Cat. No. Z918-E1-01

NE01-CCPC1-□

NE Programmer Ver. 2.0

OPERATION MANUAL

OMRON

NE01-CCPC1-□
NE Programmer Ver. 2.0

Operation Manual

Produced August 2008

Notice:

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

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

⚠ **DANGER**	Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. Additionally, there may be severe property damage.
⚠ **WARNING**	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.
⚠ **Caution**	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

OMRON Product References

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

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

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

Visual Aids

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

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

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

Trademarks and Copyrights

ControlNet and EtherNet/IP are registered trademarks of ControlNet International.

DeviceNet is a registered trademark of the Open DeviceNet Vendors Association.

Microsoft, Windows, Windows NT, Windows 2000, Windows XP, and Windows Vista are registered trademarks of the Microsoft Corporation.

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

Version Upgrade Guide

The following table lists the improvements made in the upgrade from NE Programmer version 1.7x version 2.00.

Upgrade **Note** All of the upgrades apply only to the CJ2 CPU Units.	**NE Programmer version 1.7x**	**NE Programmer version 2.00**
Support for CJ2 CPU Units • Added CJ2 USB/Serial Port to interface selections. • Enabled selecting CJ2 CPU Units: CJ2H-CPU64-EIP, CJ2H-CPU65-EIP, CJ2H-CPU66-EIP, CJ2H-CPU67-EIP, and CJ2H-CPU68-EIP.	Not supported	Supported
Support for 2-dimensional Array Variables In a CJ2 CPU Unit project, 2-dimensional array variables can be used in the following windows and lists. • Variable Windows • Watch Window • Cross-reference Pop-up Window • Variable Used List	Not supported	Supported
Reduced Restrictions in Using Variables for Array Subscripts, Upgrade 1 Variables can be specified for array subscripts in multi-level data structures (including members).	Variables could be specified only for the last subscript in an array.	Variables can be specified for any subscript in an array.
Reduced Restrictions in Using Variables for Array Subscripts, Upgrade 2 Data structure members can be specified for array subscripts.	Array variables could not be specified for elements.	Data structure members can be specified for array subscripts.
Read Protection for Programs as One Logical POU Element A password can be set for a program to prevent reading the program without password verification.	Read protection was possible only for function blocks.	Read protection is also possible for programs.
Support for Converting Physical Address Inputs to Variables • If a physical address is input directly into the Ladder Editor, a variable will be automatically created if a comment is also input. • If a variable is input or selected in the Ladder Editor and the Enter Key is pressed, a comment can be input in the same continuous operation.	Not supported	Supported
Expanded Variable Attributes • The Network Variable attribute for global variables was changed to a Convert to Network Variable attribute and a Network I/O Variable attribute. • Items for the Convert to Network Variable attribute and a Network I/O Variable attribute were added to the following: Option settings for variables, Variable Editors, and printer output.	Only a Network Variable attribute	A Convert to Network Variable attribute and a Network I/O Variable attribute
Variables Used List, Improvement 1 Comments were added to the Variables Used List and Cross-reference Report.	No comments	Comments were added.
Variables Used List, Improvement 2 The Variables Used List was changed to a window so that it can be resized.	Resizing was not possible.	Resizing is possible.
Program Upload Protection A password can be set for uploading the configuration (all programming). Verification is necessary to upload the configuration from the CPU Unit.	Not supported	Supported
In the New Watch Item Dialog Box, external variables can be specified for automatic completion.	Not supported	External variables can be specified for automatic completion.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

About this Manual:

This manual provides information required to use the NE Programmer Control and Network Support Software. The NE Programmer is an integrated programming environment used to program NE1S-series PLCs and CJ2 CPU Units.

Please read this manual and all related manuals listed in the following table and be sure you understand the information provided before attempting to use the NE Programmer.

Name	Cat. No.	Contents
NE Programmer Operation Manual (this manual)	Z918	Describes the operating procedures of the NE Programmer. Also describes programming and program elements, as well as differences and restrictions depending on the operating environment.
SYSMAC NE1S Series NE1S-CPU01 Programmable Controller Operation Manual	Z901	Provides an outlines of and describes the design, installation, maintenance, and other basic operations for the NE1S-series PLC. Also provides information on how to use the NE Programmer.
NE1S-CNS21U ControlNet Unit Operation Manual	Z902	Describes the use of the NE1S-series ControlNet Unit.
NE1S-DRM21U DeviceNet Unit Operation Manual	Z903	Describes the use of the NE1S-series DeviceNet Unit.
CS1W-EIP21/CJ1W-EIP21 CJ2H-CPU□□-EIP EtherNet/IP Unit Operation Manual	Z909	Describes the use of CS/CJ/NE1S-series EtherNet/IP Units.
CJ2 CPU Unit Hardware User's Manual	W472	Provides hardware information for the CJ2 CPU Units.
CJ2 CPU Unit Software User's Manual	W473	Provides software information for the CJ2 CPU Units.

This manual contains the following sections.

Precautions provide general precautions for using the NE Programmer and related devices.

Section 1 introduces the NE Programmer, provides NE Programmer specifications, and provides the basic operating procedure. It also outlines the differences between the NE1S and the CJ2 CPU Units.

Section 2 describes the structure of the programs.

Section 3 describes software installation.

Section 4 provides an outline of the operations and functions of the NE Programmer.

Section 5 provides details on programming.

Section 6 describes the configuration of the PLC system.

Section 7 provides the procedures for online operation.

The ***Appendices*** describe variable applications guidelines, structured text keywords, external variables, CIP message communications, the PLC Setup for CJ2 CPU Units, and Ethernet Settings.

⚠ **WARNING** Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each section in its entirety and be sure you understand the information provided in the section and related sections before attempting any of the procedures or operations given.

Read and Understand this Manual

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

Warranty and Limitations of Liability

WARRANTY

1. The warranty period for the Software is one year from either the date of purchase or the date on which the Software is delivered to the specified location.
2. If the User discovers a defect in the Software (i.e., substantial non-conformity with the manual), and returns it to OMRON within the above warranty period, OMRON will replace the Software without charge by offering media or downloading services from the Internet. And if the User discovers a defect in the media which is attributable to OMRON and returns the Software to OMRON within the above warranty period, OMRON will replace the defective media without charge. If OMRON is unable to replace the defective media or correct the Software, the liability of OMRON and the User's remedy shall be limited to a refund of the license fee paid to OMRON for the Software.

LIMITATIONS OF LIABILITY

1. THE ABOVE WARRANTY SHALL CONSTITUTE THE USER'S SOLE AND EXCLUSIVE REMEDIES AGAINST OMRON AND THERE ARE NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL OMRON BE LIABLE FOR ANY LOST PROFITS OR OTHER INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF USE OF THE SOFTWARE.
2. OMRON SHALL ASSUME NO LIABILITY FOR DEFECTS IN THE SOFTWARE BASED ON MODIFICATION OR ALTERATION OF THE SOFTWARE BY THE USER OR ANY THIRD PARTY.
3. OMRON SHALL ASSUME NO LIABILITY FOR SOFTWARE DEVELOPED BY THE USER OR ANY THIRD PARTY BASED ON THE SOFTWARE OR ANY CONSEQUENCE THEREOF.

Application Considerations

SUITABILITY FOR USE

THE USER SHALL NOT USE THE SOFTWARE FOR A PURPOSE THAT IS NOT DESCRIBED IN THE ATTACHED USER MANUAL.

Disclaimers

CHANGE IN SPECIFICATIONS
The software specifications and accessories may be changed at any time based on improvements or for other reasons.

EXTENT OF SERVICE
The license fee of the Software does not include service costs, such as dispatching technical staff.

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

PRECAUTIONS

This section provides general precautions for using the NE Programmer and related devices.

The information contained in this section is important for the safe and reliable application of the NE Programmer. You must read this section and understand the information contained before attempting to use the NE Programmer.

1 Intended Audience

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

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

2 General Precautions

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

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

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

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

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

3 Safety Precautions

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

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

⚠ WARNING Fail-safe measures must be taken by the customer to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes. Serious accidents may result from abnormal operation if proper measures are not provided.

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

⚠ Caution Confirm safety at the destination node before editing or transferring a program, PLC Setup, I/O tables, I/O memory data, or parameter data to another node. Doing either of these without confirming safety may result in unexpected operation and injury.

4 Application Precautions

Observe the following precautions when using the NE Programmer.

- Do not turn OFF the power supply to the PLC or disconnect the cable when the NE Programmer is connected online to the PLC.
- Do not turn OFF the power supply to a Unit while data is being transferred. Particularly 1) never turn OFF the power to a PLC when the Memory Card is being accessed and 2) never remove the Memory Card while it is being accessed. To remove the Memory Card, first press the Memory Card power supply switch and wait for the BUSY indicator to turn OFF before removing the Memory Card. In the worst-case scenario, the Memory Card may become unusable if the power is turned OFF while the Memory Card is being accessed or the Memory Card is removed while it is being accessed.
- Confirm that no adverse effect will occur in the system before attempting any of the following. Not doing so may result in an unexpected operation.
    - Changing the operating mode of the PLC (including changing the Startup Mode setting).
    - Force-setting/force-resetting any bit in memory.
    - Changing the present value of any word or any set value in memory.
    - Restoring the values of variables.
- Always clear the memory of a CJ2 CPU Unit before downloading programs, the PLC Setup, or the I/O tables from the NE Programmer. If the memory is not cleared before downloading the data, unexpected operation may occur in the controlled system.
- Check that the DIP switches and data memory (DM) are properly set before starting operation.
- Before actual operation, check the parameter settings and user program (such as the ladder program) for proper execution in trial operation. Always check the program before transferring it.
- Resume operation only after transferring to the new CPU Unit the contents of the DM and HR Areas required for resuming operation. Not doing so may result in an unexpected operation.
- Confirm that a Compact Flash Card containing the correct contents is inserted before starting operation.
- Be sure to set the network connection settings and network parameters correctly.

- The BKUP indicator lights when data is being written to flash memory. Do not turn OFF the power supply to the CPU Unit when the BKUP indicator is lit. The data may not be written correctly.
- Set the startup mode only after confirming that the controlled facilities will not be adversely affected.
- Do not turn OFF the power supply to the CPU Unit while a Memory Card is being accessed.
- Do not remove a Memory Card while it is being accessed. Press the Memory Card power button and confirm that the BUSY indicator goes out before removing the Memory Card.
- The user program is stored in nonvolatile memory, and operation is possible even if the Battery voltage has dropped or a Battery is not installed. (Operation will not be stopped for a memory error.) Data in the DM and EM Areas, however, will not be stable without a Battery that is fully charged. If data from the DM or EM Areas is used to control outputs from the program, used the Battery Error Flag to control outputs or perform other measures to ensure safety.

SECTION 1
Introduction

This section introduces the NE Programmer, provides NE Programmer specifications, and provides the basic operating procedure. It also outlines the differences between the NE1S and the CJ2 CPU Units.

1-1 NE Programmer Introduction

1-1-1 What Is the NE Programmer?

The NE Programmer is a programming tool that provides an integrated development environment for PLCs (i.e., Programmable Controllers) built with NE1S-series CPU Units and CJ2 CPU Units.

The NE Programmer can be used for the following CPU Units:

- NE1S-series CPU Units: NE1S-CPU01
- CJ2 CPU Units: CJ2H-CPU□□-EIP

The functions that can be used depend on the CPU Unit that is connected. For details, refer to *1-4 Differences and Restrictions between NE1S-series CPU Units and CJ2 CPU Units*.

1-1-2 NE Programmer Features

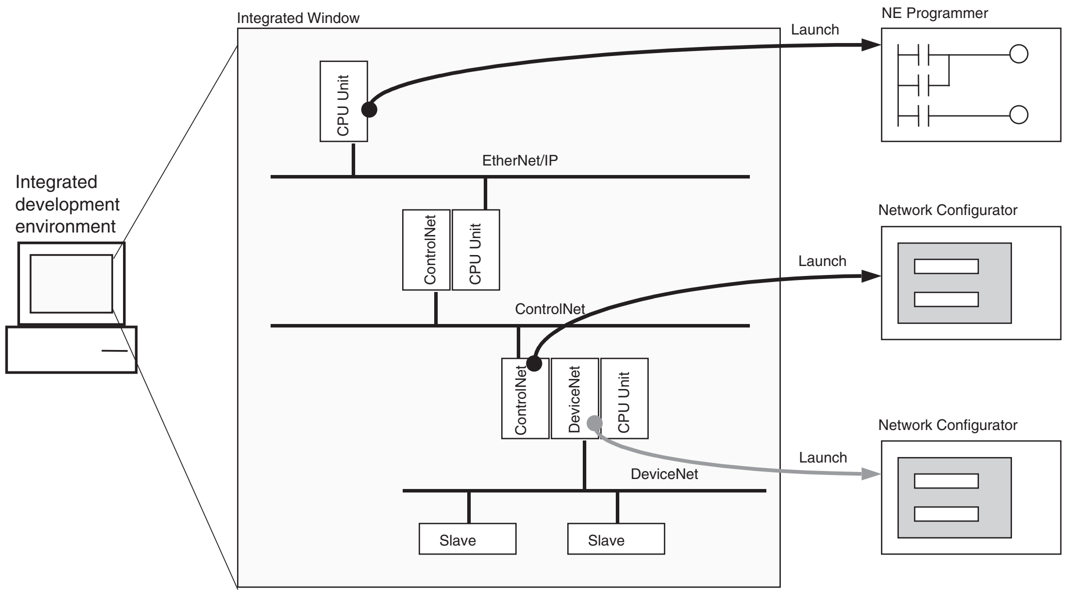
Flexible Connection Environment

You can connect the NE Programmer to an NE1S-series CPU Unit by using any of the following interfaces: USB, RS-232C, EtherNet/IP, ControlNet, or DeviceNet. (CIP communications is used for all these interfaces.)

You can also connect the NE Programmer to a CJ2 CPU Unit by using any of the following interfaces: USB, RS-232C, or EtherNet/IP. (CIP communications is used for all these interfaces.)

Integrated Development Environment

Each Programming Device can be started centrally from a window displaying the NE1S-series CPU Units connected through serial communications (USB and RS-232C), EtherNet/IP, ControlNet, and DeviceNet.



Note It is not possible to display three different network levels simultaneously on in the actual window.

Remote Programming/Monitoring from NE Programmer (Serial Connection)

The networks share the common CIP communications protocol, so that another NE1S-series CPU Unit in an EtherNet/IP, ControlNet, or DeviceNet network or a CJ2 CPU Unit on an EtherNet/IP network can be remotely programmed or monitored from NE Programmer Programming Software connected through a serial connection (USB or RS-232C).

Remote Programming/Monitoring from NE Programmer (EtherNet/IP or ControlNet Connection)

A computer running the NE Programmer Programming Software can also be connected directly to an EtherNet/IP or ControlNet network. In this case, another NE1S-series CPU Unit in an EtherNet/IP, ControlNet, or DeviceNet network can be remotely programmed or monitored from NE Programmer Programming Software connected through ControlNet

Directly Inputting Mnemonics in a Ladder Window

Instructions with text mnemonics such as LD, AND, and MOV can be entered directly by moving the cursor to the desired insertion point in the Ladder Programming Window and entering the mnemonic. Inputs and outputs can also be entered by selecting the input or output icon from the toolbar and advanced instructions can be entered by dragging and dropping the instruction from the instruction list.

Automatic Allocation of I/O Memory to Variables

Variables can be broadly divided into two categories: global variables that are shared within a PLC and local variables that are unique to a program or function block.

Physical memory addresses can be allocated to both global variables and local variables automatically with NE Programmer. The automatic allocation of I/O memory allows variables to be used in programming without dealing directly with the variables' addresses.

While it isn't necessary to deal with the variables' addresses, the user can manually specify the physical addresses of global variables if necessary.

Link from the Network Configurator

The NE Programmer can be started from the Network Configurator and data can be transferred between the two. Also, network variables registered with the NE Programmer can be shared with the Network Configurator. It is easy to make network settings using variable names in the Network Configurator by importing and exporting variables in the variable tables.

1-2 Specifications

1-2-1 NE Programmer Specifications

Item		Specifications	
System requirements	Hardware	Computer: IBM PC/AT or compatible CPU: Pentium, 300 MHz minimum (Pentium, 1 GHz minimum recommended) Memory: 512 bytes minimum Hard disk: 200 Mbytes minimum free space Monitor: SVGA (800 x 600 pixels) or better CD-ROM drive: 1 minimum	
	OS	Windows 2000, XP, or Vista *1	
	Supported languages	Japanese and English	
Method of connection to network		Board or card	3G8F7-DRM21: PCI Board 3G8F5-DRM21: ISA Board 3G8E2-DRM21: PCMCIA Card
		Serial connection (using gateway function from CIP to DeviceNet network)	Connection is made to a USB or RS-232C port on the CPU Unit of the PLC to which the DeviceNet Unit is mounted. **Note** **(1)** The Network Configurator can also be connected to a ControlNet or Ethernet network. **(2)** The NE1S-series USB driver does not support Windows Vista.
Connecting the NE Programmer using a Board or Card	Relation to network	The NE Programmer is allocated 1 node address.	
	Number of NE Programmer nodes connected to network	1 per network	
PLCs supported by NE Programmer		NE1S-series CPU Units: NE1S-CPU01 CJ-series CJ2 CPU Units: CJ2H-CPU□□-EIP	
Main functions	Creating logical POUs	Instructions and function blocks Languages: Ladder diagrams or structured text (ST) as required	
	Creating configurations	Tasks and global variables	
	PLC system configuration	PLC Setup, Ethernet settings, build settings, I/O table settings	
	Online operation	Automatic uploading (when using a serial connection), changing the connection target, creating I/O tables, uploading/downloading/comparing programs and other data, changing the operating mode, monitoring the program, force-setting and force-resetting bits, online editing, displaying errors, displaying the cycle time, tracing data, etc.	

*1 System requirements for Windows Vista: CPU: Pentium, 1 GHz min.; Memory: 1 GB

1-3 Basic Operating Procedure

This section describes the basic operating procedure and the relationship to the NE Programmer and the CPU Unit. The relationship and order is given for each step and setting.

1,2,3...

1. Installation

   Set the DIP switches on the front of each Unit as required.
   Mount the CPU Unit, Power Supply Unit, and other Units to the Backplane.
   Install the Memory Card if required.

   Note Refer to the operation manual for each Unit.

2. Wiring

   Connect the power supply wiring, I/O wiring, and Programming Device (NE Programmer). Connect communications wiring as required.

   Note Refer to the operation manual for each Unit for information on power supply wiring, I/O wiring, and the NE Programmer connection.

3. Initial Settings (Hardware)

   Set the DIP switches an Rotary switches on the front of the CPU Unit and other Units.

   Note Refer to the operation manual for each Unit.

4. Checking Initial Operation

   Turn the power on after checking the power supply wiring and voltage. Check the Power Supply Unit's POWER indicator.

5. Registering the I/O Table

   Check the Units to verify that they are installed in the right slots. With the PLC in PROGRAM mode, register the I/O table from the Programming Device (NE Programmer). (Another method is to create the I/O table in NE Programmer and transfer it to the CPU Unit.)

   Note Refer to *7-5 Online Operations for I/O Tables*.

6. PLC Setup Settings

   With the PLC in PROGRAM mode, change the settings in the PLC Setup as necessary from the Programming Device (NE Programmer). (Another method is to change the PLC Setup in NE Programmer and transfer it to the CPU Unit.)

   Note Refer to the operation manual for each Unit.

7. DM Area Settings for CPU Bus Units and Special I/O Units

   a) Use a Programming Device (NE Programmer) to make any necessary settings in the parts of the DM Area that are allocated to Special I/O Units and CPU Bus Units.

   b) Reset the power (ON → OFF → ON) or toggle the Restart Bit for each Unit.

   Note Refer to the operation manual for each CPU Bus Unit or Special I/O Unit.

8. Writing the Program

   Write the program with the NE Programmer.

9. Transferring the Program (NE Programmer Only)

   With the PLC in PROGRAM mode, transfer the program from NE Programmer to the CPU Unit.

10. Testing Operation

    a) Checking I/O Wiring

Output wiring	With the PLC in PROGRAM mode, force-set output bits and check the status of the corresponding outputs.
Input wiring	Activate sensors and switches and either check the status of the indicators on the Input Unit or check the status of the corresponding input bits with the Programming Device's Bit/Word Monitor operation.

    b) Auxiliary Area Settings (As Required)
    Check operation of special Auxiliary Area Settings such as the following:

Output OFF Bit	When necessary, turn ON the Output OFF Bit (A50015) from the program and test operation with the outputs forced OFF.
Hot Start Settings	When you want to start operation (switch to RUN mode) without changing the contents of I/O memory, turn ON the IOM Hold Bit (A50012).

    c) Trial Operation
    Test PLC operation by switching the PLC to MONITOR mode.

    d) Monitoring and Debugging
    Monitor operation from the Programming Device. Use functions such as force-setting/force-resetting bits, tracing, and online editing to debug the program.

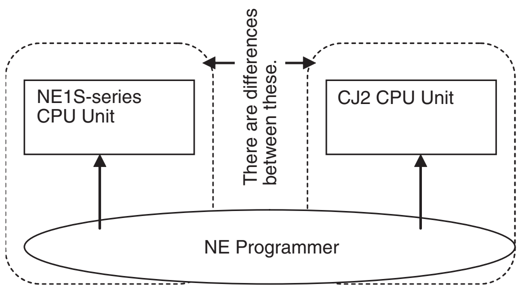
Note Refer to *SECTION 7 Online Operation*.

11. Saving and Printing the Program
12. Running the Program
    Switch the PLC to RUN mode to run the program.

1-4 Differences and Restrictions between NE1S-series CPU Units and CJ2 CPU Units

1-4-1 Improvements in CJ2 CPU Units Compared with NE1S-series CPU Units

There are differences between using the NE Programmer for a CJ2 CPU Unit in comparison to using it for an NE1S-series CPU Unit.



The following points are the main differences.

Support for Japanese Identifiers

Names in Japanese can be given to all types of identifiers in projects for the CJ2 CPU Unit. The term "identifier" refers to the following items.

• Data type names (structure names and structure member names)

- Variable names (including FBIO group names)
- Function block and instance names
- POU names (program names and configuration block names)
- Configuration names

Support for Two-dimensional Arrays

It is possible to input and display two-dimensional array variables. The Variable Editor, Edit Variables Dialog Box, and Watch Window all support two-dimensional variables.

Reduced Restrictions

The maximum number function block (FB) definitions and number of instances have been increased.

Item	Using the NE1S	Using the CJ2				
		CPU64-EIP	CPU65-EIP	CPU66-EIP	CPU67-EIP	CPU68-EIP
Maximum number of FBs instances	1,024	256	512	1,024	2,048	2,048
Area size for automatic variable allocation	8 banks	1 bank		4 banks	8 banks	

The maximum size of each variable (including array and structure) has been increased from 128 to 32,000 words.

Item	Using the NE1S	Using the CJ2
Size per variable (including structure)	128 words (256 bytes max.)	32,000 words (64 Kbytes max.)

The maximum number of array elements has been increased from 256 to 32,000 elements.

Item	Using the NE1S	Using the CJ2
Elements	1 to 255 elements	1 to 32,000 elements

The restriction on specifying variables as array subscripts has been removed.

Item	Using the NE1S	Using the CJ2
Specifying variables for array subscripts in multilevel arrays (including members)	A variable can be specified only for the rightmost array subscript.	Variables can be specified for array subscripts regardless of the position. Example: aaa[index0].bbb[index1].ccc[index2]
Specifying structure members for array subscripts	Array variables cannot be specified for elements.	Structure members can be specified for array subscripts. Example: aaa[str.member]

Read Protection

With the CJ2, it is possible to set read protection for program POUs.

Note With the NE1S it is possible to set protection only for function blocks.

Upload Protection

Password protection can be used for program uploading from the CPU Unit.

1-4-2 Restrictions for CJ2 CPU Units Compared with NE1S-series CPU Units

The following restrictions apply to CJ2 CPU Units in comparison to the NE1S-series CPU Units.

Item	Using the NE1S	Using the CJ2
Initial values for variables	Supported	Not supported
Change log	Supported	Not supported
SEND MAIL instruction (MLSND)	Supported	Not supported

SECTION 2
Program Structure

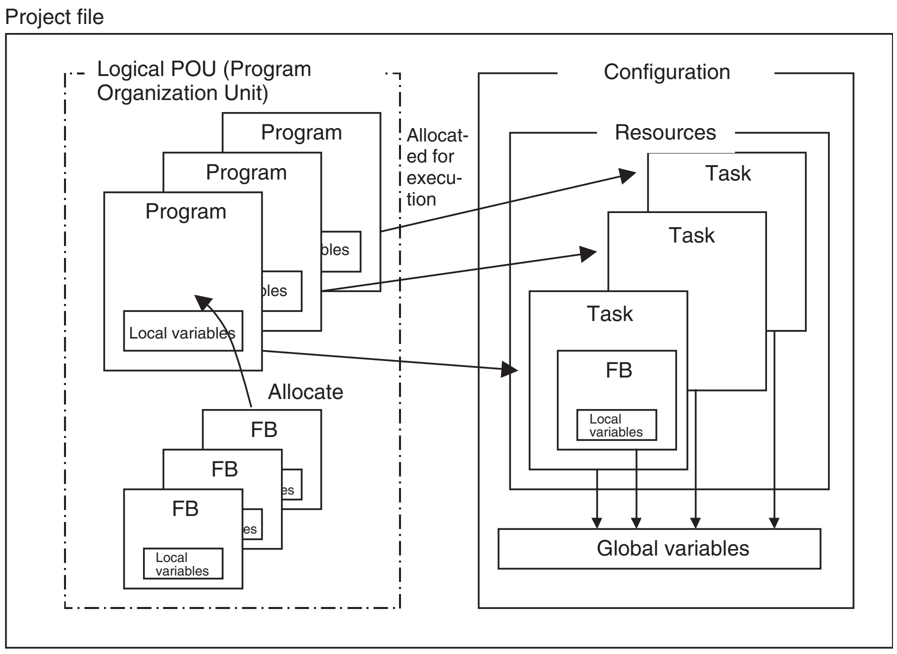
This section describes the structure of the programs.

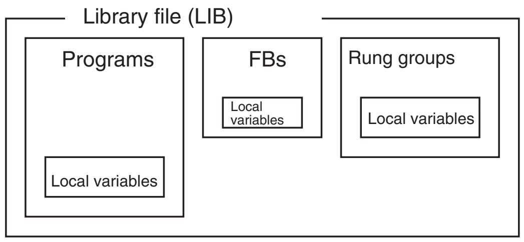
2-1 Outline of the NE Programmer

2-1-1 Project Structure

The NE Programmer is an integrated development environment for next-generation PLCs. It supports the programming specified in the IEC 61131-3 standard.

All of the data is created in a single file, known as a project file, from an integrated development environment. A project is composed of logical POUs (Program Organization Units) and the configuration. The logical POUs become executable when they are allocated in the configuration (PLC system).





Program Organization Units (Logical POUs)

Program organization units (logical POUs) include the following elements.

1,2,3...
1. Programs
2. Function Blocks (FB)

All kinds of POUs are programmed entirely with variables. Variables are automatically used in a specified area in I/O memory and are allocated in the specified area.

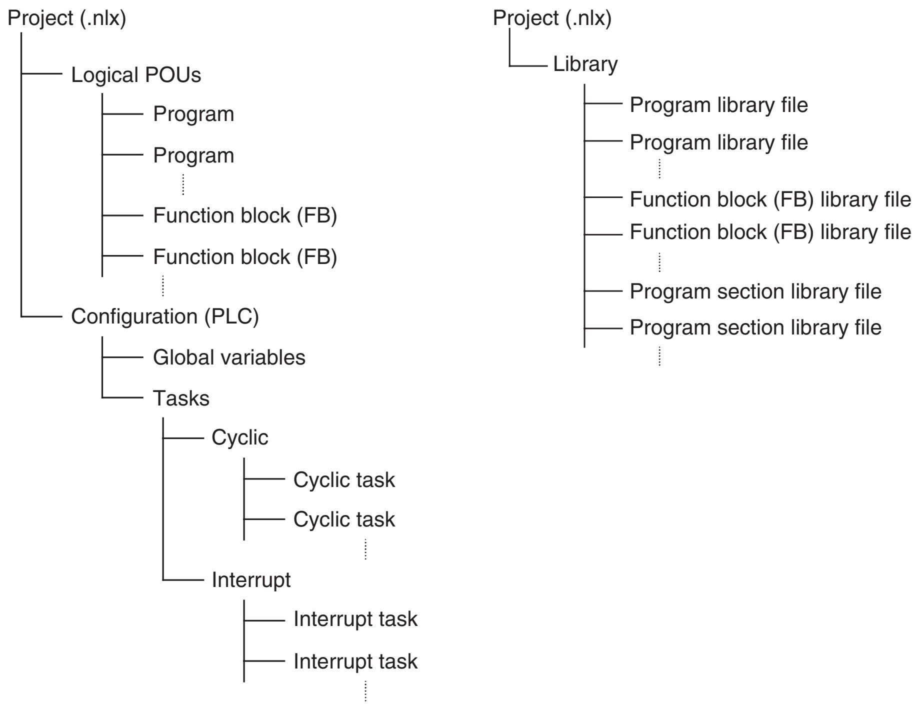
Configuration (PLC System)

The configuration (PLC system) is made up of resources (a task group of executable units) and global variables (variables that apply to all resources).

Libraries

A program, function block, or rung group (one rung or multiple rungs, including local variables) can be saved as a library function and reused. (Each unit is saved as a separate file, i.e., 1 program = 1 file, 1 function block = 1 file, or 1 rung group = 1 file.)

Note NE Programmer projects have the following directory-tree structure.



2-1-2 Logical POU (Program Organization Unit)

Logical POU Structure

A single POU is defined with the following two software elements.

1,2,3...
1. Declaration of memory usage (variable declarations)
2. Logic in an algorithm

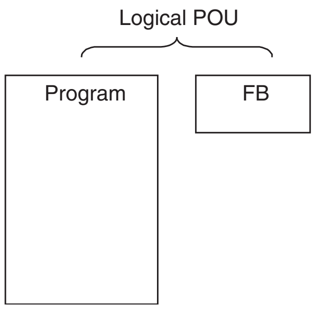
The logical POU can be reused when these two elements have been converted to a library file as a set. (It is also possible to reuse multiple rungs that have been converted to a library file with the variables used in the rungs.)

Logical POU Types

There are two types of logical POUs: programs and function blocks.

1,2,3...
1. Programs
   Programs are large functional units or units that must be executed at a specific time. Basically, the unit is entirely declared in the user interface.

2. Function Blocks
   Function blocks are components of programs and are not executed at a specific time. The unit is displayed as a single function in the user interface, so only inputs and outputs are declared.

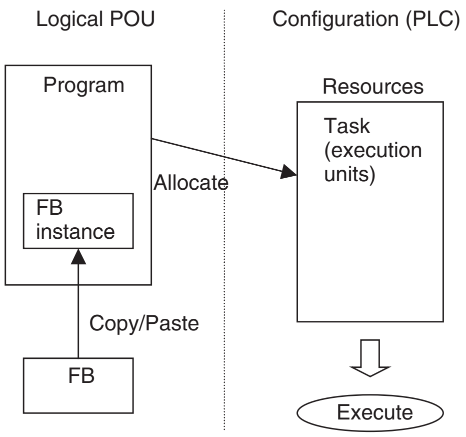


Item	Logical POU	
	Program	Function block (FB)
Unit size	Large functional unit	Functional unit smaller than program
Execution timing	(or) Requires execution at specific time.	Does not require execution at a specific time.
User interface	All disclosed.	I/O only disclosed.

Executing a Logical POU

The logical POU itself is not an executable unit; it is allocated to the configuration (PLC) first and then can be executed.

1,2,3...
1. Programs are allocated to “tasks” in the resources (creating a program instance) to become executable.
2. Function blocks are copied and pasted into a program (creating a function block instance) to become executable.



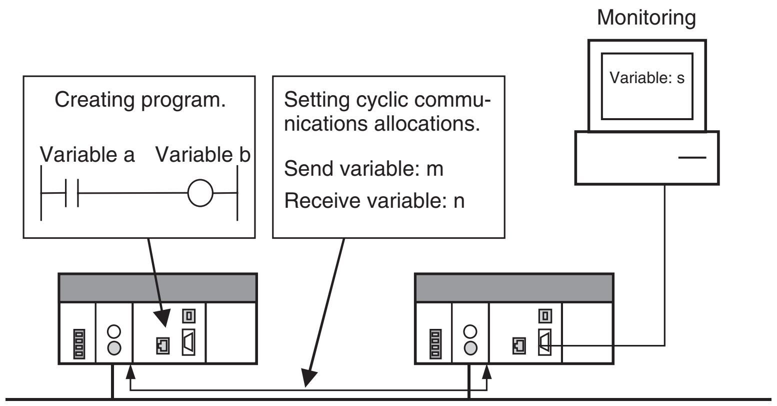
Creating a Logical POU in NE Programmer

After creating an NE Programmer project, select ***File - New*** from the menus, and input the POU name in the dialog box for creating a new POU. Set the *POU Type* to either ***Program*** or ***Function Block***.

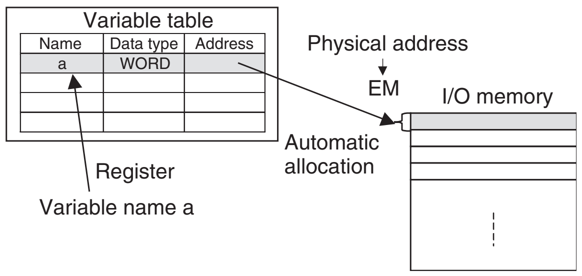
Note To paste a function block in a program (i.e., to create an instance), the function block must be saved. Once the function block has been saved (by selecting ***Save Changes to project*** from the menu), its icon will appear in the Project Explorer Bar and the icon can be dragged and dropped into the desired program.

2-1-3 Variables

With the NE Programmer, the user does not give I/O memory addresses (known as a physical addresses) directly when specifying addresses in the CPU Unit when programming, making communications settings, or monitoring. Variables are used to specify I/O memory addresses in the CPU Unit.



If the user has entered a variable in the variable table, the created variable is automatically allocated a physical address in the CPU Unit's I/O memory. It is not necessary for the user to know which physical addresses have been allocated.



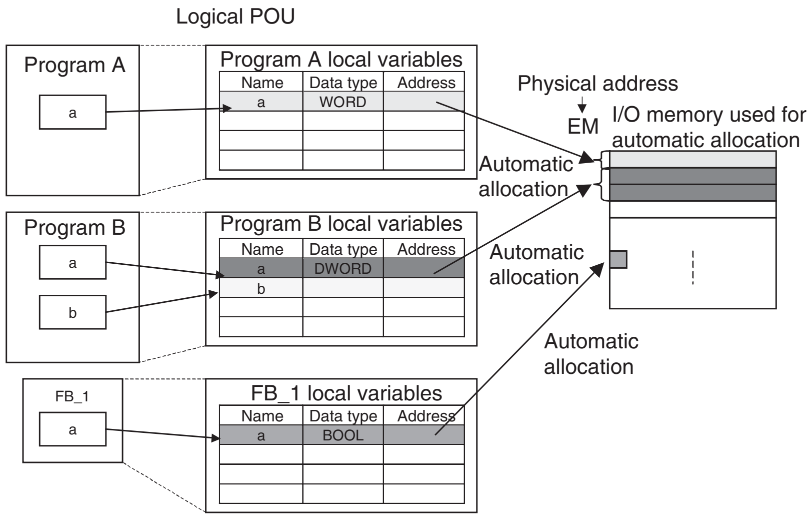
Local Variables and Global Variables

There are two basic variable types. Local variables are defined in individual logical POUs and used just within the local logical POU. Global variables are defined at the configuration level and can be used in all of the logical POUs.

Local Variables

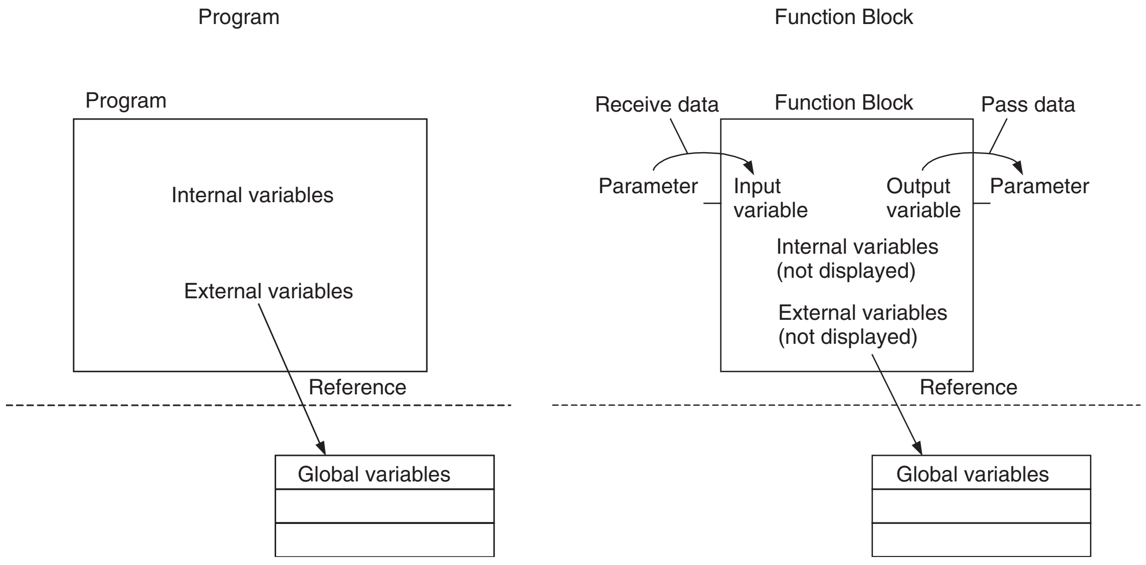
Variables defined in individual logical POUs (programs or function blocks) are known as local variables. These variables are effective only in the local logical POU.

The system automatically allocates local variables (both local program and local FB variables) to a particular memory area (EM area banks other than EM bank 0). The physical addresses are hidden.


Logical POU
Program A
a
Program A local variables
Name | Data type | Address
a | WORD
Physical address
EM
I/O memory used for automatic allocation
Automatic allocation
Program B
a
b
Program B local variables
Name | Data type | Address
a | DWORD
b
Automatic allocation
Automatic allocation
FB_1
a
FB_1 local variables
Name | Data type | Address
a | BOOL

The various kinds of local variables are listed below.

- Program:
  Internal variables (VAR) and external variables (VAR_EXTERNAL)
- Function Block:
  Internal variables (VAR), input variables (VAR_INPUT), output variables (VAR_OUTPUT), and external variables (VAR_EXTERNAL)



- Internal variables (VAR) are used only in the logical POU, they do not exchange values with external parameters, and cannot reference external global variables.
- External variables (VAR_EXTERNAL) are special variables used within the logical POU to reference global variables outside of the POU.
- Input variables (VAR_INPUT) exist only in function blocks and are used to receive the values of external parameters.
- Output variables (VAR_INPUT) exist only in function blocks and are used to pass values to external parameters.

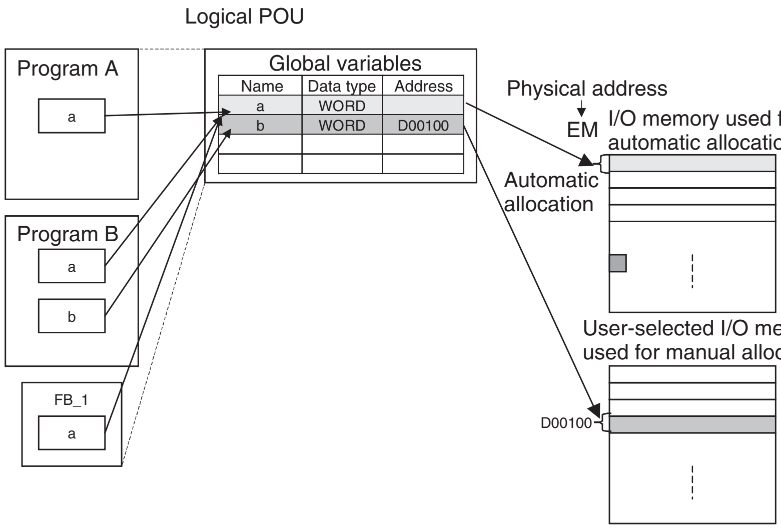
Note Even if local variables are defined with the same name in different logical POUs, they are treated as different variables and allocated different physical addresses.

Global Variables

Variables defined at the configuration level and shared by all logical POUs are known as global variables. Global variables provide an interface between logical POUs.

The system can allocate memory to global variables automatically or specific physical addresses can be allocated manually (direct specification).

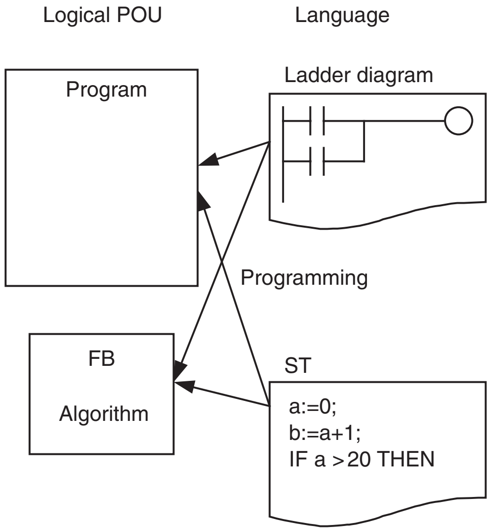
To use global variables within logical POUs, global variables must be referenced from external variables as local variables. For details on external variables, refer to *External Variables (VAR_EXTERNAL)* in *2-2-10 Details on Local Variables*.



Note Physical addresses can also be directly specified without using variables.

2-1-4 Programming Languages

The algorithms in programs and function blocks (FB) can be written as ladder diagrams (LD) or structured text (ST).



Creating Programs in NE Programmer

When creating logical POUs in NE Programmer, select **LD** when creating a ladder diagram or **ST** when creating structured text.

Note

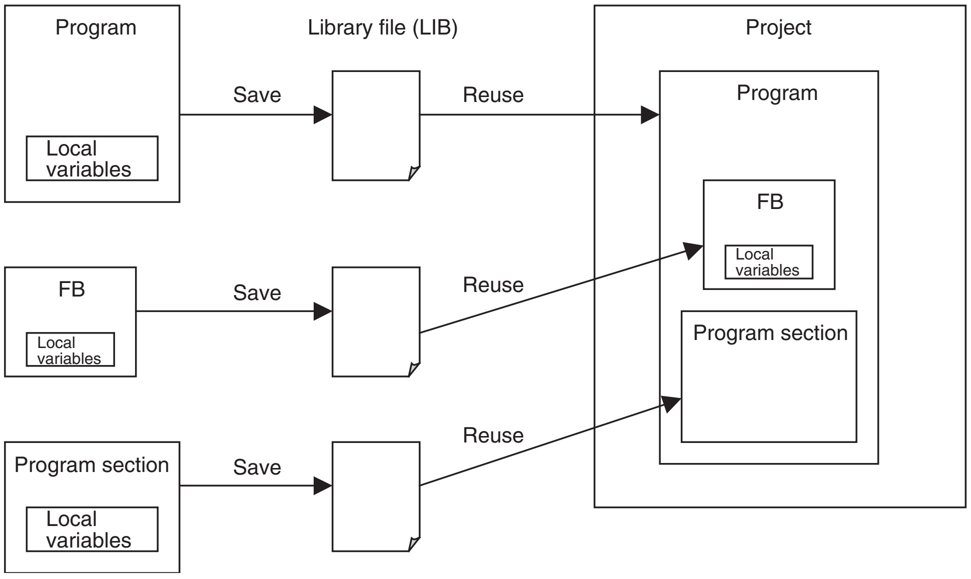
(1) When using ladder diagrams, the program can be input directly in the ladder diagram with mnemonics. Mnemonics can also be input in a mnemonics-only editing screen by selecting ***Edit - Edit Using Mnemonic Editor*** from the menus. It is not possible, however to create logical POUs that call function blocks.

(2) When using ladder diagrams, it is possible to import a text file (.txt) containing the mnemonics. To import a file, 1) Select the logical POU in program view, right-click, and select ***Change View***, and then 2) Right-click in mnemonic view, right-click, and select **Import.**

2-1-5 Libraries

Converting a Logical POU or Rungs to a Library

With the NE Programmer, logical POUs (programs or function blocks) or rungs in a program can be converted to a library file and reused.



Creating Library Files in NE Programmer:
Either select the desired logical POU and select ***POU - Register to Library*** from the Library Menu or select the rungs from the Ladder Window and select ***Rung - Register to Library***.

2-2 Variables

2-2-1 Naming Variables

The following rules apply to the CPU Units.

NE1S-series CPU Units

- CPU Units comply with the IEC 61131-3 standard.
- Allowed characters: numbers 0 to 9, letters a to z, letters A to Z, and the underscore character.
- Encoding: ASCII
- Number of characters: 48 characters max.

- Upper and lower case characters are distinguished but do not make variables different. For example, “aBc” and “AbC” are treated as the same variable, but the variable “aBc” is recorded as “aBc”.
- Variable names cannot begin with a number (0 to 9).
- There cannot be two or more consecutive underscore characters.
- Variable names cannot have the following characters followed by a number, because these combinations are treated as actual data area addresses.
  A, W, H, T, C, D, E0, @D, *D, @E□, *E□, IR, DR, TK, and TR
- Also, the actual address expressions of external variables (preassigned variables) cannot be used.
- Variables with only numbers are treated as actual addresses, e.g., “D00000” is treated as the word address D00000 and “0000” is treated as the word address CIO 0000.

CJ2 CPU Units

- Allowed characters: numbers 0 to 9, letters a to z, letters A to Z, and the underscore character.
- Encoding: UTF-8
- Number of characters: 48 characters max.
- Upper and lower case characters are distinguished but do not make variables different. For example, “aBc” and “AbC” are treated as the same variable, but the variable “aBc” is recorded as “aBc”.
- Variable names cannot begin with a number (0 to 9).
- There cannot be two or more consecutive underscore characters.
- Variable names cannot have the following characters followed by only a number, because these combinations are treated as physical addresses.
  A, W, H, T, C, D, E0, @D, *D, @E@, *E@, IR, DR, TK, and TR
  For example, “D100” is treated as word address D100.
- Variables consisting of only numbers are treated as physical addresses. For example, “100” is treated as the word address CIO 100.

2-2-2 Types of Variable

The following variable types are supported.

- Internal Variables (VAR)
  Internal variables are used only within an instance. They cannot be used pass data directly to or from parameters outside of the instance.
- Input Variables (VAR_INPUT)
  Input variables can input data from parameters outside of the instance. The default input variable is an EN (Enable) variable, which passes input condition data.
- Output Variables (VAR_OUTPUT)
  Output variables can output data to parameters outside of the instance. The default output variable is an ENO (Enable Out) variable, which passes the instance’s execution status.
- External Variables (VAR_EXTERNAL)
  External variables are local variables that are used to access global variables. They include both system variables that are registered in the NE Programmer in advance, as well as user-defined local variables. When handling I/O with variables, always define I/O as global variables and access the global variables through external variables. System variables are registered in advance as external variables. User-defined global vari-

ables are automatically registered as external variables when they are used as operands for instructions in logical POUs. If a variable is first created as an external variable, the user must manually register it as a global variable in the External Variable Tab Page of the Variable Editor to use it as a global variable. We thus recommend that the required global variables are defined before starting to program the ladder diagrams.

- Global Variables
  Global variables are defined in the in the configuration and are shared by all logical POUs. Global variables include system variables, such as the Conditions Flags and some Auxiliary Area bits, that are registered in the NE Programmer in advance, as well as user-defined global variables.

2-2-3 Variable Properties

The following table lists the variable properties.

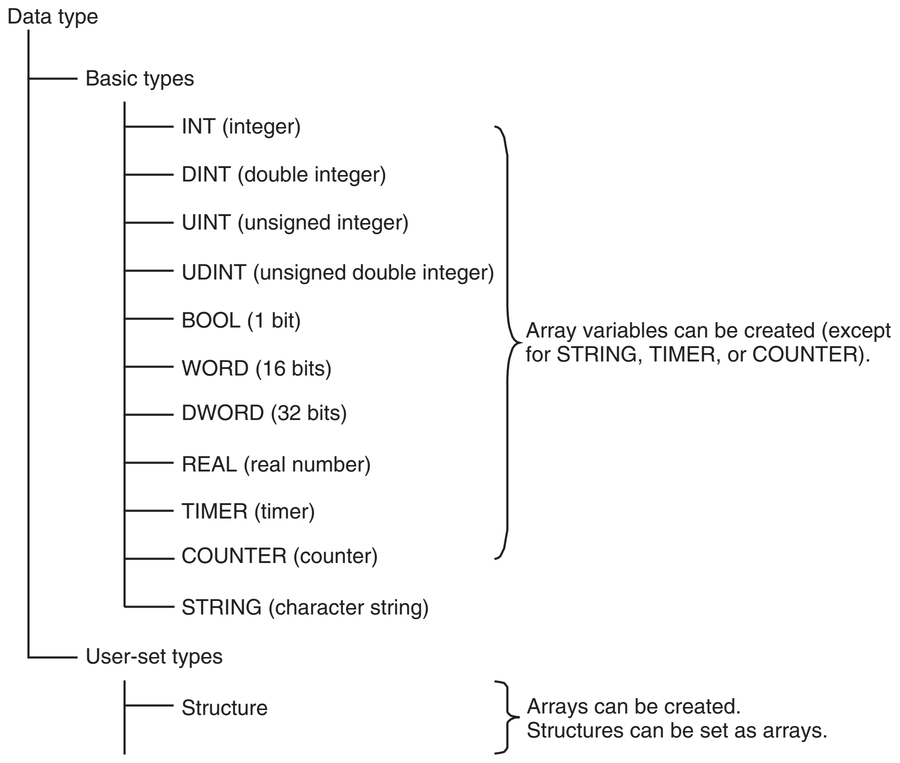
Variable property	Content	Value
Data type	Selects the variable's data type.	BOOL, INT, UINT, DINT, UDINT, WORD, DWORD, REAL, TIMER, COUNTER, STRING, or user-defined **Note** For function blocks, the logical POU name of the function block is displayed.
Array size	Sets the number of elements for a one-dimensional or two-dimensional array. Two-dimensional arrays can be set only for CJ2 CPU Units.	When not specifying an array, leave this settings blank for both a one-dimensional and two-dimensional array. When specifying a one-dimensional array, set the number of array elements for a one-dimensional array and leave the setting for a two-dimensional array blank. When specifying a two-dimensional array, set the number of array elements for a two-dimensional array and leave the setting for a one-dimensional array blank. *A two-dimensional array cannot be specified for the NE1S. *Refer to *2-2-3 Variable Properties* for other restrictions on elements.
Size	Displays the size used by the variable in the memory.	---
Initial value	With a program, this property sets the variable's value at the start of operation. With a function block, this property sets the variable's value when an instance is executed. **Note** The initial values of variables cannot be set for CJ2 CPU Units.	Set an initial value as follows, according to the variable's data type: • BOOL, WORD, or DWORD: Input the value in unsigned hexadecimal after “16#”. • INT or DINT: Input the value in unsigned decimal after “+10#” or “–10#”. • UINT or UDINT: Input the value in unsigned decimal after “10#”. • REAL: Real number. Example: +1.0, -0.23, +9.8E-3 • STRING: Character string. Example: “Data”
Address	This property sets a specific address when the address is being set manually (direct AT specification). This property cannot be changed for local variables.	---

Variable property	Content	Value
Network variable	This property sets whether to convert variables to network variables. It sets whether external access (i.e., reading or writing) is enabled for variable names. This setting can be made only for global variables. It cannot be made for local variables.	Enabled: Variable names can be accessed (i.e., read and written) externally. Disabled: Variable names cannot be accessed (i.e., read and written) externally. Local variables can always be accessed externally.
Network I/O variable	When using cyclic communications, select *Input* when disclosing the variable as an input from the network, select *Output* when disclosing the variable as an output to the network, and select *None* when the variable will not be disclosed.	None: Do not disclose the variable as a connection target. Input: Disclose the variable as an input from the network. Output: Disclose the variable as an output to the network. Global variables that have been set to *Input* or *Output* can be imported to the Network Configurator after the project has been saved. After the global variables have been imported, variable names can be used in programming to set connections for ControlNet cyclic communications. Local variables are always set to *None*.
Retain/nonretain	This property specifies whether the variable's value is retained when operation starts and the power is reset.	Retained or not retained.
Comment	Use this property to input a comment for the variable.	256 bytes max.

The properties listed in the table are described in detail below.

2-2-4 Data Type

The data format of the variable is called the data type. The following data types are supported for NE1S-series CPU Units and CJ2 CPU Units.



Note Arrays and Structures

With NE1S-series CPU Units and CJ2 CPU Units, one-dimensional array variables and user-defined data structures can be created.

- Array variables:
  A one-dimensional array variable can be created by setting the number of elements to an integer value between 1 and 255 for an NE1S-series CPU Unit or to between 1 and 32,000 for a CJ2 CPU Unit.
- Two-dimensional arrays can be created for CJ2 CPU Units.
- Structured variables:
  A structure is a set of variables containing several variables (members) with different data types. The user can configure the structure freely. Members can be specified by specifying the variable name and member name.

Data Ranges of Data Types

Data type	Contents	Size	Data range
BOOL	Bit data	1 bit	16#0 or 16#1
INT	Integer	16 bits	–10#32768 to +10#32767
UINT	Unsigned integer	16 bits	10#0 to 10#65535
DINT	Double integer	32 bits	–10#2147483648 to +10#2147483647
UDINT	Double unsigned integer	32 bits	10#0 to 10#4294967295
WORD	16-bit data	16 bits	16#0000 to 16#FFFF
DWORD	32-bit data	32 bits	16#00000000 to 16#FFFFFFFF
REAL	Real number	32 bits	Conforms to IEEE754
STRING	Character string data (ASCII data)	128 bytes or 127 characters	---
TIMER	Timer (See note 1.)	1 bit or 16 bits	16#0, 16#1, or 16#0000 to 16#FFFF
COUNTER	Counter (See note 2.)	1 bit or 16 bits	16#0, 16#1, or 16#0000 to 16#FFFF

Note

(1) When a variable is entered in the timer number (0 to 4095) operand of a timer instruction, such as TIMX or TIMHX, the data type will be TIMER. When this variable is used as an operand in another instruction, it will be treated as the timer Completion Flag if the operand takes 1-bit data or as a timer PV if the operand takes 16-bit data. The timer PVs are 16-bit binary data because NE Programmer uses only the binary format for the PVs.
The TIMER data type cannot be used in ST language function blocks.

(2) When a variable is entered in the counter number (0 to 4095) operand of a counter instruction, such as CNTX or CNTRX, the data type will be COUNTER. When this variable is used as an operand in another instruction, it will be treated as a counter Completion Flag if the operand takes 1-bit data or as a counter PV if the operand takes 16-bit data. The counter PVs are 16-bit binary data because NE Programmer uses only the binary format for the PVs.
The COUNTER data type cannot be used in ST language function blocks.

2-2-5 Array Elements (Array Specification)

A set of data with the same properties can be handled in a single structure (array). Input the array's maximum number of elements in this property to define the variable as an array.

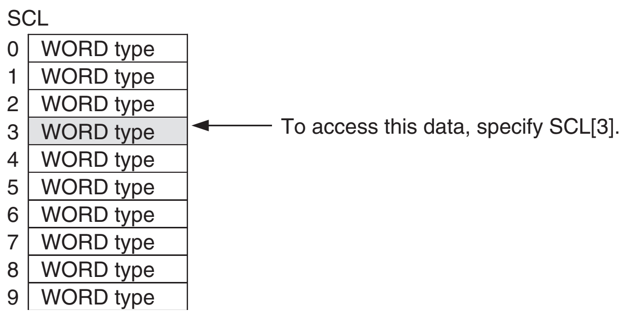
Only one-dimensional arrays can be created with the NE Programmer if a NE1S-series CPU Unit is used.

One-dimensional and two-dimensional arrays can be created if the CJ2 CPU Unit is used.

- The number of elements in an array can be set to between 1 and 255 for an NE1S-series CPU Unit and to between 1 and 32,000 for a CJ2 CPU Unit.
- The array specification can be used for internal variables (VAR), input variables (VAR_INPUT), output variables (VAR_OUTPUT), and external variables (VAR_EXTERNAL).
- When entering the variable name, specify the subscript in square brackets after the variable name.
- The subscript (for example, the subscript in variable a[ ]) can be specified in the following three ways.
  a. Directly specify the subscript number (ladder or ST language).
     Example: a[2]
  b. Specify a local variable as the subscript (ladder or ST language).
     Example: a[n], where n is a local variable
  c. Specify an arithmetic expression as the subscript (ST language only).
     Example: a[b+c], where b and c are local variables

  Note The four arithmetic operators (+, -, *, and /) can be used in expressions.
- Two-dimensional arrays can be created for CJ2 CPU Units.
  Example: a[0][0],a[i][j]

Note An array is a group of data items with the same data type. Each variable element is specified by the local variable name and a subscript. (The subscript indicates the position of an element within the array.) With a one-dimensional array, the subscript's number indicates the element in the array.
Example) An array with 10 elements and local variable name SCL
The following 10 local variables can be used: SCL[0], SCL[1], SCL[2], SCL[3], SCL[4], SCL[5], SCL[6], SCL[7], SCL[8], and SCL[9].



Array Data Types and Subscript Variable Data Types

The subscript data types that can be set for each array data type are listed in the following table.

Subscript data type / **Array data type**	**NE1S and CJ2**										**CJ2 only**		
	Basic data types										**Arrays and data structures**		
	INT	**UINT**	**DINT**	**UDINT**	**BOOL**	**WORD**	**DWORD**	**TIMER**	**COUNTER**	**REAL**	**Array *1**	**Data structure member*2**	**Data structure*3**
INT	❍	❍	❍	❍	×	❍	❍	×	×	×	Refer to the *Allowable Notations for Subscripts*, below.		
UINT	❍	❍	❍	❍	×	❍	❍	×	×	×			
DINT	❍	❍	❍	❍	×	❍	❍	×	×	×			
UDINT	❍	❍	❍	❍	×	❍	❍	×	×	×			
BOOL	❍	❍	❍	❍	×	❍	❍	×	×	×			
WORD	❍	❍	❍	❍	×	❍	❍	×	×	×			
DWORD	❍	❍	❍	❍	×	❍	❍	×	×	×			
TIMER	×	×	×	×	×	×	×	×	×	×			
COUNTER	×	×	×	×	×	×	×	×	×	×			
REAL	❍	❍	❍	❍	×	❍	❍	×	×	×			
Data structure	❍	❍	❍	❍	×	❍	❍	×	×	×			
Data structure member	❍	❍	❍	❍	×	❍	❍	×	×	×			
STRING	×	×	×	×	×	×	×	×	×	×			

*1 Example: Specifying a[i], where i is an array variable
(Notation example: d[d[0]]

*2 Example: Specifying a[i], where i is a data structure member
(Notation example: d[c.b])

*3 Example: Specifying a[i], where i is a data structure
(Notation example: d[c])

Allowable Notations for Subscripts

Notation example	**Description**	**NE1S**	**CJ2**
d[0]	Using a number for an array subscript	❍	❍
d[i]	Using a variable of an allowed basic data type for an array subscript	❍	❍
d[d[0]]	Using an array variable of an allowed basic data type for an array subscript	×	❍
d[d[i]]	Using an array variable that includes an index variable as a subscript for an array subscript	×	×
d[c]	Using a data structure variable for an array subscript	×	×
d[c.b]	Using a member of a data structure of an allowed basic data for an array subscript	×	❍

In addition to the above, formulas including addition, subtraction, multiplication, and division (+, -, *, and /) can be used for subscripts in ST programing (Example: d[x+y]).

Allowable Notations for Subscripts in Array Data Structures

Notation example	**Description**	**NE1S**	**CJ2**
c[i]	Using variables as subscripts for array data structures	❍	❍
c[1]. a[i]	Using variables in the subscripts for members of array data structures	❍	❍

Notation example	Description	NE1S	CJ2
c[i]. a[0]	Using variables as subscripts for array data structures to specify members	×	❍
c[i]. a[1]			
c[i]. b			

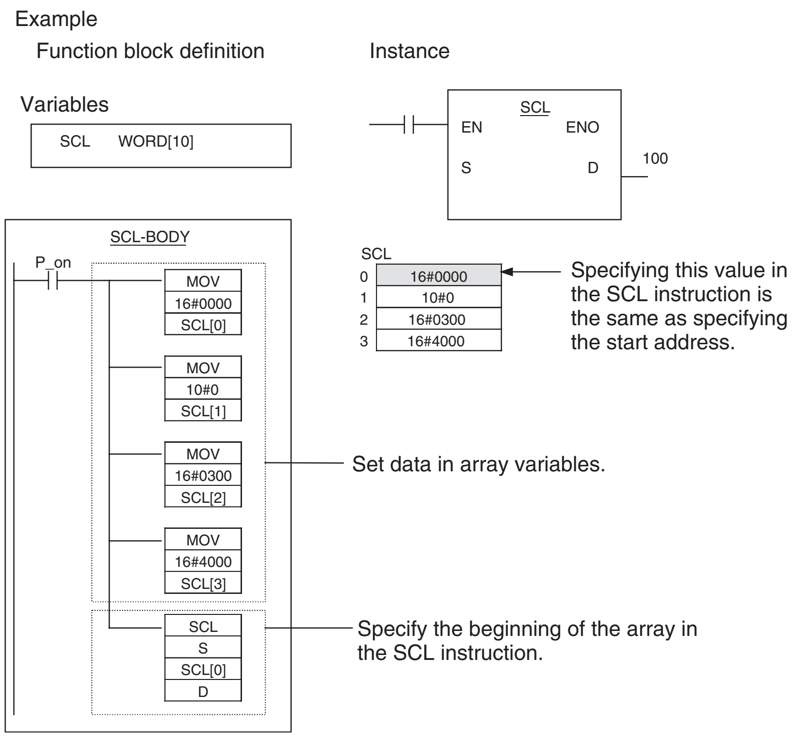
Restrictions on Arrays

Item		NE1S	CJ2
Number of elements		1 to 255	1 to 32,000
Data size per variable		255 bytes max.	64 Kbytes
Initial values	One variable	256 bytes max.	Setting not supported.
	Total data size	250 K bytes max.	Setting not supported.

Note When specifying the starting address (or end address) of multiple words in an instruction operand (see note), the address cannot be passed to a local variable by an input parameter or output parameter.

Note For example, multiple words are accessed when specifying an instruction's control data or the starting and end words of the BLOCK SET instruction (BSET(071)).

In this case, prepare arrays or structured variables with the required number of elements, set the data in the arrays or structures in the function block definition, and specify the start (or end) of the array or structure in the operand. This method effectively specifies the starting address (or end address) of multiple words.


Example
Function block definition
Instance
Variables
SCL WORD[10]
SCL-BODY
P_on
MOV 16#0000 SCL[0]
MOV 10#0 SCL[1]
MOV 16#0300 SCL[2]
MOV 16#4000 SCL[3]
SCL S SCL[0] D
SCL
EN ENO
S D
100
SCL
0 16#0000
1 10#0
2 16#0300
3 16#4000
Specifying this value in the SCL instruction is the same as specifying the start address.
Set data in array variables.
Specify the beginning of the array in the SCL instruction.

Setting the Property

- Input the number of elements (1 or more) in the *Array Size* Field in the Edit Variables Dialog Box. If the *Array Size* Field is set to 0, the variable will not be an array.
- When entering the local variable name, specify the subscript in square brackets after the local variable name.
  Example: When BOOL variable "a" is defined as an array with 8 array elements, the 7th element of variable "a" can be input as the operand bit by inputting LD a[7].
- For two-dimensional arrays (CJ2 CPU Units only), double-click the Array Elements Row and input the number of two-dimensional elements (1 to 9).

Note Basically, an array is used when the starting address (or end address) of multiple words is specified for an instruction operand. The data cannot be passed using a parameter because parameters pass values. Prepare a local array variable of the required size, set the data in the array in the function block definition, and specify the start (or end) of the local array variable for the operand. For details on using arrays to specify the starting/end addresses of multiple-word operands, refer to *Appendix D* in the *NE1S Series PLC Operation Manual*.

2-2-6 Initial Value (NE1S-series CPU Units Only) (See note.)

Note The initial values of variables cannot be set for CJ2 CPU Units.

For programs, this property specifies the variable's value when operation starts. For function blocks, this property specifies the variable's value when the instance is executed. When an instance is first executed, internal variables (VAR) and output variables (VAR_OUTPUT) are set to there initial values. Later, the variable's value may change as the instance is executed.

Automatic allocation/direct address		Initial value setting
Automatic allocation	Held	Cannot be set.
	Not held	Can be set.
Direct address	CIO Area	Can be set.
	Work Area	Can be set.
	Holding Area	Cannot be set.
	Auxiliary Area	Cannot be set.
	Data Memory (DM) Area	Cannot be set.
	Extended Memory (EM) Area	Cannot be set.
	Timer (Completion Flag/PV)	Cannot be set.
	Counter (Completion Flag/PV)	Cannot be set.

Setting the Property

Input the desired value in the *Initial Value* Field in the *Edit Variables* Dialog Box. Input a value consistent with the variable's data type, as shown in the following table.

Data type	Contents	Initial value	Initial value input method
BOOL	Bit data	16#0 or 16#1	Input as unsigned hexadecimal. Input the value after "16#".
INT	Integer	–10#32768 to +10#32767	Input as signed decimal. Input the value after "+10#" or "–10#". (If "10#" is input, a + sign will be attached automatically.)
UNIT	Unsigned integer	10#0 to 10#65535	Input as unsigned decimal. Input the value after "10#".

Data type	Contents	Initial value	Initial value input method
DINT	Double integer	–10#2147483648 to +10#2147483647	Input as signed decimal. Input the value after "+10#" or "–10#". (If "10#" is input, a + sign will be attached automatically.)
UDINT	Double unsigned integer	10#0 to 10#4294967295	Input as unsigned decimal. Input the value after "10#".
WORD	16-bit data	16#0000 to 16#FFFF	Input as unsigned hexadecimal. Input the value after "16#".
DWORD	32-bit data	16#00000000 to 16#FFFFFFFF	Input as unsigned hexadecimal. Input the value after "16#".
REAL	Real number	0.0	Input a signed numerical value. Input the value after the sign. E.g., +1.0, –0.23, +9.87E-3
STRING	Character string data (ASCII data)	---	---

⚠ **Caution** If you convert an NE1S project to a CJ2 project, any initial value settings used in the NE1S project must be set from the program. This is because the CJ2 CPU Units do not support the initial value settings. If initial values are not set from the program and the project is downloaded to a CJ2 CPU Unit, unexpected operation may occur.

2-2-7 Address (Direct Allocation of a Physical Address)

A physical address cannot be specified directly for a local variable. To specify a physical address, enter the physical address directly for the operation instead of using a local variable. Global variables are always accessed as external variables even if a physical address is specified.

When using ST language, a physical address cannot be directly allocated.

For example, if "100" is input in an ST program, it is interpreted as the decimal value 100 (10#100). If "100" is input in a ladder program, it is interpreted as CIO address CIO 0100.

2-2-8 Retain/Nonretain

This property specifies whether the variable will retain its data when the PLC is turned ON, operation starts, or a fatal error occurs. (If the CPU Unit is not equipped with a battery, the data will be cleared regardless of the Retain setting.)

The following tables show whether variable data is retained or cleared when the power is turned ON, operation starts, or a fatal error occurs regardless of the Retain and IOM Hold Bit (A50012) settings.

Retain/ Nonretain setting	Address	Initial value	Variable data
Retain	Automatic allocation	Cannot be set	Retained
Nonretain	Automatic allocation	Set	Initial value
		Not set	Cleared (0)

Effect of Forced Status Hold Bit (A50013) Setting

The status of forced variable data will be as follows when the PLC is turned ON, operation starts, or a fatal error occurs regardless of the Forced Status Hold Bit (A50013) setting:

Retain setting	Address	Hold Bit condition
Retain or Do not retain	Automatic allocation	Forced status cleared.

Setting the Property

Select ***Retain*** or ***Nonretain*** in the *Retain/Nonretain* Field of the *Edit Variables* Dialog Box.

When allocating address to global variables, they will be set to either *Retain* or *Nonretain* and cannot be changed.

2-2-9 Local Variable Properties and Types of Local Variables

The following table shows which properties must be set, can be set, and cannot be set, based on the local variable usage.

Property	Variable usage			
	Internal	Input	Output	External
Name	Must be set.	Must be set.	Must be set.	Must be set.
Type	Must be set.	Must be set.	Must be set.	Must be set.
Initial Value	Can be set.	Can be set.	Can be set.	Can be set.
Array specification	Can be set.	Can be set.	Can be set.	Can be set.
Retain	Can be set.	Can be set.	Can be set.	Can be set.
AT (direct allocation to a specific physical address, global variables only)	Cannot be set.	Cannot be set.	Cannot be set.	Cannot be set.

Note Initial values cannot be set for CJ2 CPU Units.

2-2-10 Details on Local Variables

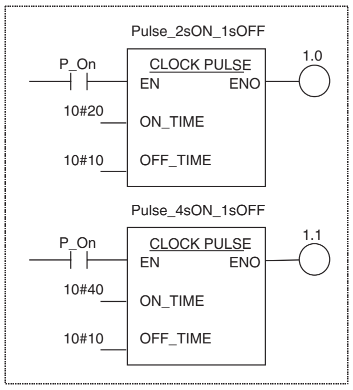
■ Internal Variables (VAR)

Internal variables are used within an instance. These variables are internal to each instance. They cannot be referenced from outside of the instance and are not displayed in the instance.

The values of internal variables are retained until the next time the instance is executed. Consequently, even if instances of the same function block definition are executed with the same I/O parameters, the result will not necessarily be the same.

Example:

The internal variable tim_a in instance Pulse_2sON_1sOFF is different from internal variable tim_a in instance Pulse_4sON_1sOFF, so the instances cannot reference and will not affect each other's tim_a value.



Variable table

Usage	Name	Type
Internal	tim_a	WORD
Internal	tim_b	WORD
Input	ON_TIME	INT
Input	OFF_TIME	INT

Note In NE Programmer, internal variables are created with a default variable name of "FI", which is a local variable that is ON the first time that an instance is executed. (The "FI" internal variable can be used to initialize the instance.)

Retaining Data through Power Interruptions and Start of Operation

Internal variables (VAR) retain the value from the last time that the instance was called. In addition, the Retain Option can be selected so that an internal variable will also retains its value when the power is interrupted or operation starts (the mode is switched from PROGRAM mode to RUN or MONITOR mode).

Initial Value

An initial value can be set for an internal variable (VAR) that is not being retained (i.e., when the Retain Option not selected). An initial value cannot be set for an internal variable if the Retain Option is selected.
Internal variables that are not being retained will be initialized to 0.

The initial value will be set, regardless of the value of the IOM Hold Bit (A50012).

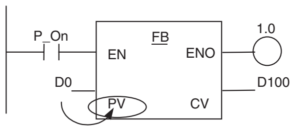
Auxiliary Area control bit		Initial value
IOM Hold Bit (A50012)	ON	Initial value is set.
	OFF	Initial value is set.

Local Variable Usage (Types)

■ Input Variables (VAR_INPUT)

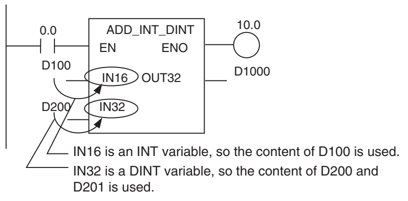
Input variables pass arguments to the instance from the outside. The input variables are displayed on the left side of the instance.

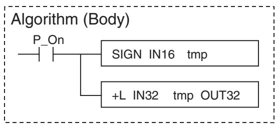
When an instance is called, the value of the input source (data contained in the specified parameter just before the instance was called) will be passed to the input variable.



The value of the parameter specified as the input (value of D0) is passed to the instance's input variable (PV).

Example


0.0
ADD_INT_DINT
EN
ENO
10.0
D100
IN16
OUT32
D1000
D200
IN32
IN16 is an INT variable, so the content of D100 is used.
IN32 is a DINT variable, so the content of D200 and D201 is used.



Variable table

Usage	Name	Type
Internal	tmp	DINT
Input	EN	BOOL
Input	IN16	INT
Input	IN32	DINT
Output	ENO	BOOL
Output	OUT32	DINT

Note (1) The same variable name cannot be assigned to an input variable (VAR_INPUT) and output variable (VAR_OUTPUT). If it is necessary to have the same variable as an input variable and output variable, register the variables with different names and transfer the value of the input variable to the output variable in the function block with an instruction such as MOV.

(2) When the instance is executed, input values are passed from parameters to input variables before the algorithm is processed. Consequently, values cannot be read from parameters to input variables while the algorithm is being processed. If it is necessary to read a value within the execution cycle of the algorithm, do not pass the value from a parameter. Assign the

value to an internal variable and use an AT setting (direct allocation of a physical address).

Initial Value

When you set an initial value for an input variable (VAR_INPUT), that value will be written to the variable when the parameter for input variable EN goes ON and the instance is executed for the first time (and that one time only). If an initial value has not been set for an input variable, the input variable will be set to 0 when the instance is first executed.

EN (Enable) Variable

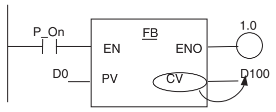
When an input variable (VAR_INPUT) is created, the default input variable is the EN variable. The instance will be executed when the parameter for input variable EN is ON.

The initial value will be set, regardless of the value of the IOM Hold Bit (A50012).

■ Output Variables (VAR_OUTPUT)

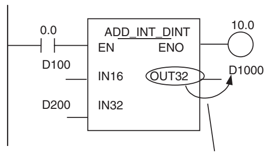
Output variables pass return values from the instance to external applications. The output variables are displayed on the right side of the instance.

After the instance is executed, the value of the output variable is passed to the specified parameter.

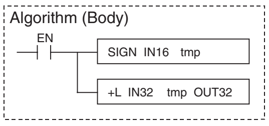


The value of the output variable (CV) is passed to the parameter specified as the output destination, which is D100 in this case.

Example



OUT32 is a DINT variable, so the variable's value is passed to D1000 and D1001.

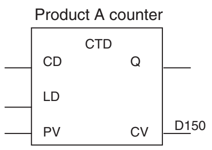


Variable table

Usage	Name	Type
Internal	tmp	DINT
Input	EN	BOOL
Input	IN16	INT
Input	IN32	DINT
Output	ENO	BOOL
Output	OUT32	DINT

Like internal variables (VAR), the values of output variables (VAR_OUTPUT) are retained until the next time the instance is executed.

Example:
In the following example, the value of output variable CV will be retained until the next time the instance is executed.



Note (1) The same variable name cannot be assigned to an input variable (VAR_INPUT) and output variable (VAR_OUTPUT). If it is necessary to have the same variable as an input variable and output variable, register the variables with different names and transfer the value of the input variable to the output variable in the function block with an instruction such as MOV.

(2) Output variable values (data) are passed to the corresponding parameters after the algorithm is processed. Consequently, values cannot be written from output variables to parameters while the algorithm is being executed. If it is necessary to write a value within the execution cycle of the algorithm, do not write the value to a parameter. Assign the value to an internal variable and use an AT setting (direct allocation of a physical address).

Initial Value

An initial value can be set for an output variable (VAR_OUTPUT) that is not being retained, i.e., when the Retain Option is not selected. An initial value cannot be set for an output variable if the Retain Option is selected.

The initial value will be set, regardless of the value of the IOM Hold Bit (A50012).

ENO (Enable Output) Variable

When a local output variable is created, the default variable is the ENO variable. The ENO output variable will be turned ON when the instance is called. The user can change this value. The ENO output variable can be used as a flag to check whether or not instance execution has been completed normally.

■ External Variables (VAR_EXTERNAL)

External variables are local variables that are used to access global variables. They include both system variables that are registered in the NE Programmer in advance, as well as user-defined local variables. When handling I/O with variables, always define I/O as global variables and access the global variables through external variables. System variables are registered in advance as external variables. User-defined global variables are automatically registered as external variables when they are used as operands for instructions in logical POUs. If a variable is first created as an external variable, the user must manually register it as a global variable in the External Variable Tab Page of the Variable Editor to use it as a global variable. We thus recommend that the required global variables are defined before starting to program the ladder diagrams.

2-2-11 Creating Variables in NE Programmer

In NE Programmer, variables are not declared by inputting a declarative statement. To create variables, insert the variables in the variable tables and input the properties.

Internal Variables

When a logical POU (program or function block) is created, the unit's individual variable table is displayed automatically. Variables can be registered in the table with either of the following methods.

1,2,3...

1. Inputting the instructions first:
   If a new variable name is input in an operand when inputting the instruction, the variable will be registered automatically in the variable table's internal variable sheet.
2. Inputting the variable table first:
   Right-click the variable table, select ***Add***, and register the variable in the variable table. When the instructions are input later, the registered variable name can be input in the operands.

Physical addresses **cannot** be input directly for local variables by inputting the physical address in the *Edit Variables* Dialog Box.

Global Variables

After creating an NE Programmer project, select ***File - New*** from the menus, change the configuration name in the Configuration Setting Window if necessary, and create a new configuration. In the project workspace, double-click the global variables under the configuration. Right-click the variable table, select ***Add***, and register the variable in the variable table. When the instructions are input later, the registered variable name can be input in the operands.

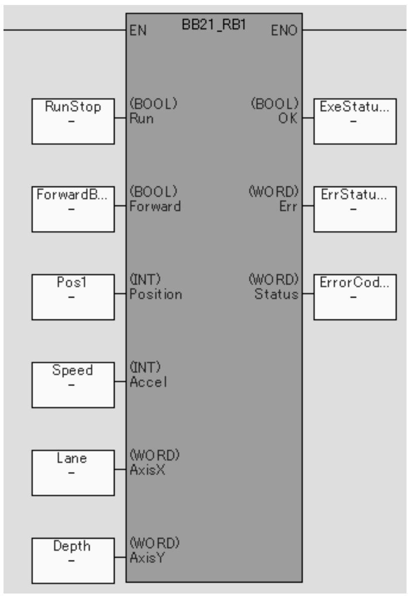
Physical addresses can be input directly for global variables by inputting the physical address in the *Address* Field in the *Edit Variables* Dialog Box.

2-2-12 Grouping Variables

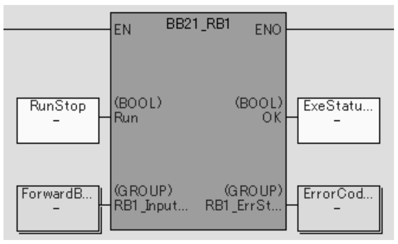
In function blocks, the input variables (VAR_INPUT) and output variables (VAR_OUTPUT) can be placed in separate groups and displayed with input and output group names.

When grouping input or output variables, the group name is displayed on the input and output side when the function block is pasted into a program.

Display Example of Ungrouped Variables



Display Example of Grouped Variables



Grouping FB I/O Variables in NE Programmer:
Select the input or output variables to be grouped, right-click, and select ***Group Input/Output Variables - Group*** from the popup menu.

2-2-13 Importing and Exporting Variables

Exporting Variables

With NE1S-series CPU Units and CJ2 CPU Units, variable data can be exported in the following file formats.

- OPC Server CSV file (for use with SCADA Software)
- Text file for use with the CX-Designer
- Text file for use with the SPU-Console
- CSV file (for use with external Programming Devices or programs such as Excel)

Exporting with NE Programmer:

- OPC Server, CX-Designer, or SPU-Console Format:
  Select ***File - Export Variable*** and select the desired file format.

- CSV Format:
  Right-click in the variable table and select ***Export***.

Importing Variables

The following data can also be imported.

- CSV files (for use with external Programming Devices or programs such as Excel)

Importing with NE Programmer:
Right-click the local or global variable table and select ***Import - CSV Format***.

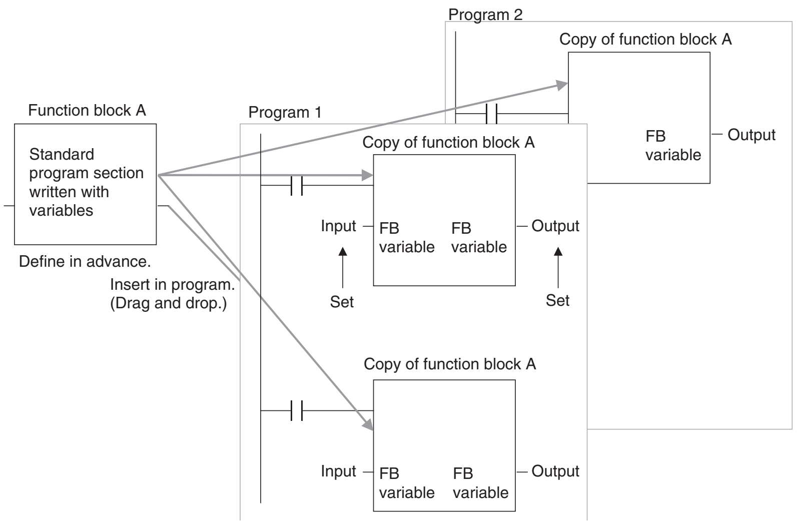
2-3 Function Blocks

2-3-1 Function Block Features and Operation

Overview

A function block is a basic program element containing a standard processing function that has been defined in advance. Once the function block has been defined, the user just has to insert the function block in the program and set the I/O in order to use the function.

As a standard process, a function block is not created with actual physical addresses, but rather with local variables. The user sets parameters (constants, variables, or physical addresses) in those variables to use the function block.



Advantages of Function Blocks

Function blocks allow complex programming units to be reused easily. Once standard program sections have been created in a function block and saved in a file, they can be reused just by placing the function block in a program and setting the parameters for the function block's I/O. Reusing standardized function blocks will reduce the time required for programming/debugging, reduce coding errors, and make the program easier to understand.

Structured Programming

Structured programs created with function blocks have better design quality and require less development time.

■ **Easy-to-read "Black Box" Design**

The I/O operands are displayed as local variable names in the program, so the pro-gram is like a "black box" when entering or reading the program and no extra time is wasted trying to understand the internal algorithm.

■ **Easily Create Different Processes from a Single Function Block**

Many different processes can be created easily from a single function block by using input variables (VAR_INPUT) for the parameters in the standard process (parameters such as timer SVs, control constants, speed settings, and travel distances).

■ **Reduce Coding Errors**

Coding mistakes can be reduced because blocks that have already been debugged can be reused.

■ **Protect Data**

The local variables in the function block cannot be accessed directly from the outside, so the data can be protected. (Data cannot be changed unintentionally.)

■ **Programming with Variables provides Improved Reusability**

The function block's I/O is entered as local variables, so the data addresses in the function block do not have to be changed as they do when copying and reusing a program section.

Creating Libraries

Processes that are independent and reusable (such as processes for individual steps, machinery, equipment, or control systems) can be saved as function block definitions and converted to library functions.

The function blocks are created with local variable names that are not tied to physical addresses, so new programs can be developed easily just by reading the definitions from the file and placing them in a new program.

Compatible with Multiple Languages

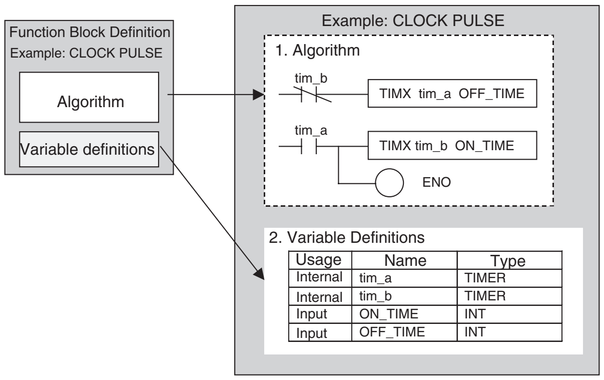
Mathematical expressions that are difficult to enter in ladder language can be entered easily in structured text (ST) language.

Function Block Structure

A function block consists of the function block definition that is created in advance and the function block instances that are actually inserted in the program.

Function Block Definition

The basic, reusable part of the function block is called the "function block definition." Each function block definition contains the algorithm and local variable definitions, as shown in the following diagram.



Usage	Name	Type
Internal	tim_a	TIMER
Internal	tim_b	TIMER
Input	ON_TIME	INT
Input	OFF_TIME	INT

1. Algorithm

The algorithm is standard programming written with variable names rather than physical I/ O memory addresses. In NE Programmer, algorithms can be written in either ladder programming or ST (structured text).

2. Local Variable Definitions

The variable table lists each variable's usage (input, output, or internal) and properties (data type, etc.). For details on local variables, refer to *2-3-2 Restrictions in Variables in Function Blocks*.

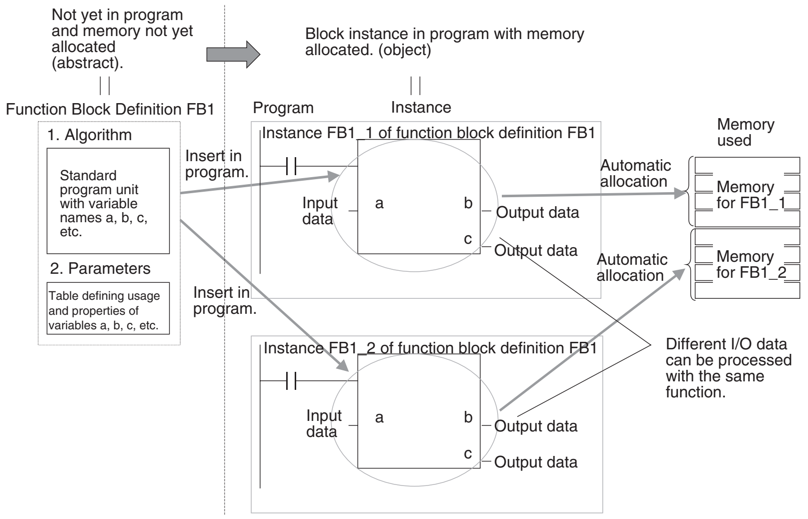
■ Number of Function Block Definitions

Up to 1,024 function block definitions can be created for one PLC (CPU Unit).

Instances

When a function block definition is inserted in a program, the function block uses a particular memory region for its local variables. Each function block definition that is inserted in the program and allocated a different memory region is called an "instance." Each instance is assigned an identifier called an "instance name."

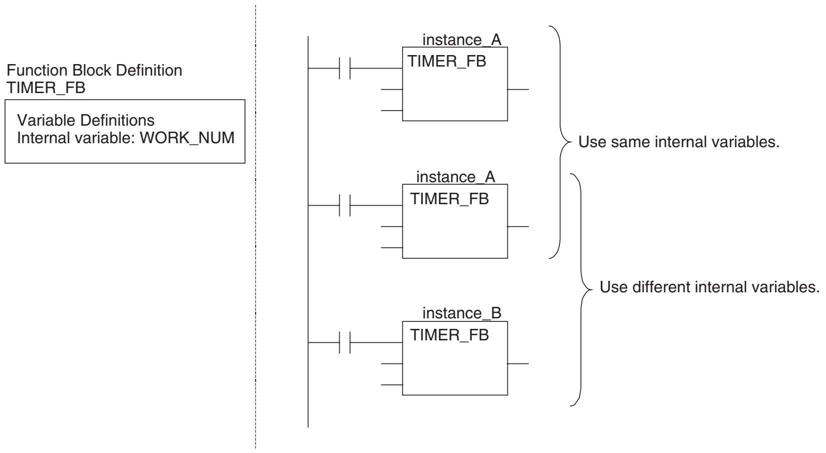
By generating instances, a single function block definition can be used to process different I/O data with the same function.


Not yet in program and memory not yet allocated (abstract).
Block instance in program with memory allocated. (object)
Function Block Definition FB1
Program
Instance
1. Algorithm
Standard program unit with variable names a, b, c, etc.
2. Parameters
Table defining usage and properties of variables a, b, c, etc.
Insert in program.
Insert in program.
Instance FB1_1 of function block definition FB1
Input data
a
b
c
Output data
Output data
Automatic allocation
Automatic allocation
Memory used
Memory for FB1_1
Memory for FB1_2
Instance FB1_2 of function block definition FB1
Different I/O data can be processed with the same function.

Note Instances are managed by names. More than one instance with the same name can also be inserted in the program. If two or more instances have the same name and in the same POU, they will use the same internal variables (VAR). If the instances are in different POUs, they will use different internal variables.

For example, if a function block that uses a timer as an internal variable (VAR) is inserted at several points in the POU, each instance must be given a different name. If two or more of the instances have the same name, the timer would be duplicated, which is not allowed.

If, however, internal variables are not used or they are used only temporarily and initialized the next time an instance is executed, the same instance name can be used to save memory.


Function Block Definition
TIMER_FB
Variable Definitions
Internal variable: WORK_NUM
instance_A
TIMER_FB
instance_A
TIMER_FB
instance_B
TIMER_FB
Use same internal variables.
Use different internal variables.

■ Creating Instances

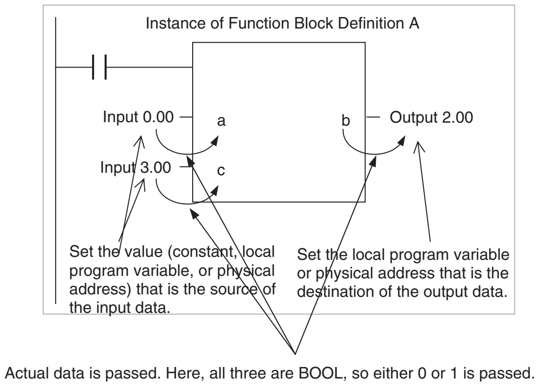
Instances can be created just by dragging and dropping the function block icon. Drag the function block icon from the logical POU in the NE Programmer workspace and drop it in the program.

■ Number of Instances

Multiple instances can be created from a single function block definition. Up to 1,024 instances can be created for a single PLC (CPU Unit). The allowed number of instances is not related to the number of function block definitions or the number of tasks in which the instances are inserted.

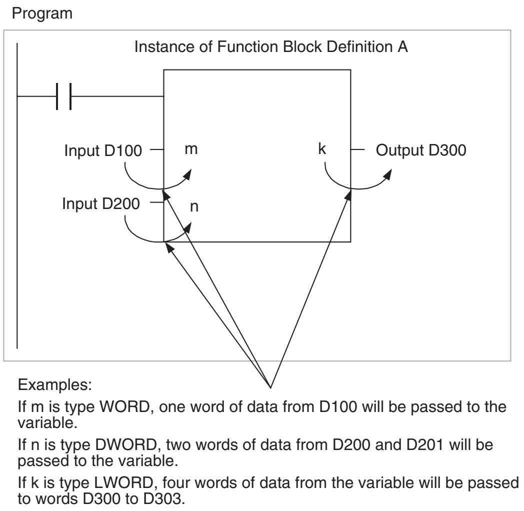
Parameters

Each time an instance is created, set the values (constants, local program variables, or physical I/O memory addresses) used to exchange data with the function block's local I/O variables. These values are called parameters.


Instance of Function Block Definition A
Input 0.00
a
b
Output 2.00
Input 3.00
c
Set the value (constant, local program variable, or physical address) that is the source of the input data.
Set the local program variable or physical address that is the destination of the output data.
Actual data is passed. Here, all three are BOOL, so either 0 or 1 is passed.

The data passed from the input parameter to the function block is not the source address, but the data within the source address (with the data format and size specified by the variable's data type). In a similar fashion, the data passed from the function block to the output parameter is not an address, but the actual data for the output address (with the data format and size specified by the variable's data type).

Even if the input parameter or output parameter is a word address, the data that is passed will be the data (with the format and size specified by the variable's data type) starting from the specified word address.


Program
Instance of Function Block Definition A
Input D100
m
k
Output D300
Input D200
n
Examples:
If m is type WORD, one word of data from D100 will be passed to the variable.
If n is type DWORD, two words of data from D200 and D201 will be passed to the variable.
If k is type LWORD, four words of data from the variable will be passed to words D300 to D303.

Note

(1) Only addresses in the following data areas can be used as parameters: CIO Area, Auxiliary Area, DM Area, EM Area (bank 0), Holding Area, and Work Area.
The following cannot be used:
a) Index and Data Registers (neither direct nor indirect specification)
b) Indirect addressing of DM or EM Area (neither binary nor BCD mode)

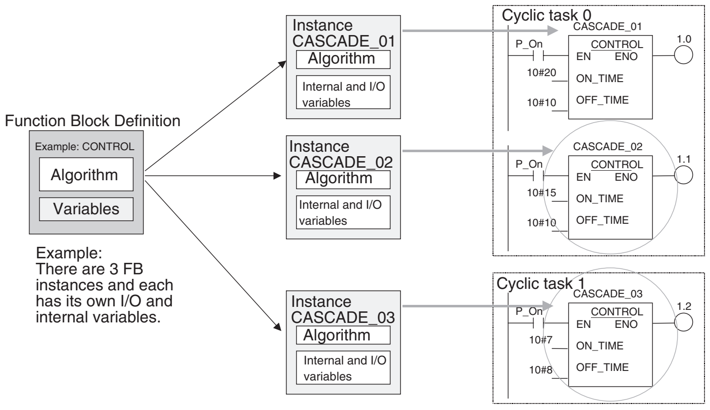
(2) Local program variables can also be specified as parameters, provided that the local program variable's data size is the same as the local variable's data size.

⚠ **Caution** Input values are passed from parameters to the input variables (VAR_INPUT) when an instance is executed but before the algorithm is processed. Consequently, a parameter cannot be used to pass a value if it is necessary to read the value during processing (within the execution cycle of the algorithm). Instead, directly input a physical address to pass the value.
Likewise, output values (VAR_OUTPUT) are passed from output variables to parameters just after the algorithm is processed. Consequently, a parameter cannot be used to pass a value if it is necessary to write the value during processing (within the execution cycle of the algorithm). Instead, directly input a physical address to pass the value.

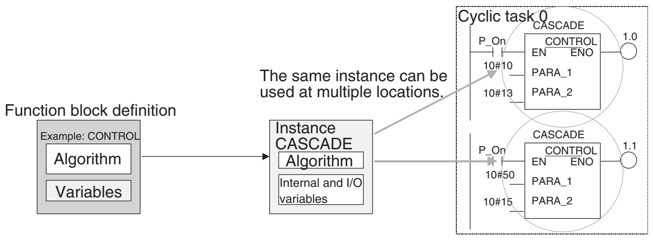
■ Reference Information

A variety of processes can be created easily from a single function block by using parameter-like elements (such as fixed values) as input variables (VAR_INPUT) and changing the values passed to the input variables for each instance.

Example: Creating 3 Instances from a Single Function Block Definition



The same instance name can be used at multiple locations in the program if internal variables (VAR) are not used or processing will not be affected even if internal variables (VAR) are used in the multiple locations.



In this case, the same memory area will be used, so some precautions are required. For example, if an instance containing a timer instruction is used in more than one program location, the same timer number will be used causing output bit duplication, and the timers will not function properly if both instructions operate at the same time.

2-3-2 Restrictions in Variables in Function Blocks

The following table shows the number of local variables that can be used and the kind of variable that is created by default for each of the variable usages.

Variable usage	Allowed number	Variable created by default
Input (VAR_INPUT)	Up to 64 per function block (not including EN)	EN (Enable): Receives an input condition. The instance is executed when the variable is ON. The instance is not executed when the variable is OFF.
Output (VAR_OUTPUT)	Up to 64 per function block (not including ENO)	ENO (Enable Output): Outputs the function block's execution status. The variable is turned ON when the instance starts being executed. It can be turned OFF by the algorithm. The variable remains OFF when the instance is not executed.
Internal (VAR)	Unlimited	FI: A local variable that turns ON the first time the instance is executed. (It can be used for initialization the first time an instance is executed.)
External variables	---	System variables

Initial Values of Input Variables (VAR_INPUT) in Function Blocks (NE1S-series CPU Units Only)

When you set an initial value for an input variable (VAR_INPUT), that value will be written to the variable when the parameter for input variable EN goes ON and the instance is executed for the first time (and that one time only). If an initial value has not been set for an input variable, the input variable will be set to 0 when the instance is first executed.

Initial Values of Output Variables (VAR_OUTPUT) in Function Blocks (NE1S-series CPU Units Only)

An initial value can be set for an output variable (VAR_OUTPUT) that is not being retained, i.e., when the Retain Option is not selected. An initial value cannot be set for an output variable if the Retain Option is selected.

Initial Values of Internal Variables (VAR) in Function Blocks (NE1S-series CPU Units Only)

An initial value can be set for an internal variable (VAR) that is not being retained (i.e., when the Retain Option not selected). An initial value cannot be set for an internal variable if the Retain Option is selected. Internal variables that are not being retained will be initialized to 0.

The initial value will be set, regardless of the value of the IOM Hold Bit (A50012).

⚠ **Caution** With a CJ2 CPU Unit, initial values cannot be specified for input variables, output variables and internal variables in function blocks. If you convert an NE1S project to a CJ2 project, any initial value settings used in the NE1S project must be set from the program or function block algorithm.

2-3-3 Function Block Specifications

Function Block Specifications

Item	Description
Number of function block definitions	1,024 max. per NE1S-series CPU Unit and 2,048 max. per CJ2 CPU Unit
Number of instances	1,024 for the NE1S and 2,048 for the CJ2
Number of instance nesting levels	8 levels max.
Number of input variables (VAR_INPUT) per function block	64 variables max.
Number of output variables (VAR_OUTPUT) per function block	64 variables max.

Function Block Elements

The following table shows the items that must be entered by the user when defining function blocks.

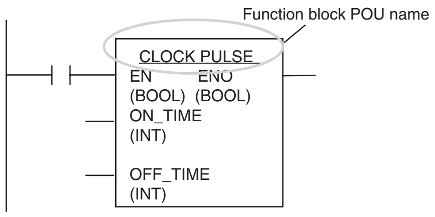
Item	Description
Function block POU name	The name of the function block definition (logical POU name)
Language	The programming language used in the function block definition. Select ladder programming or structured text

Item	Description
Local variable definitions	Variable settings, such as operands and return values, required when the function block is executed • Type (usage) of the variable • Name of the variable • Data type of the variable • Initial value of the variable
Algorithm	Enter the programming logic in ladder or structured text.
Comment	Function blocks can have comments.

Function Block Definition Name

This is the name of a function block, which is one kind of logical POU.

- Number of characters for logical POU names: 24 characters max.
- Allowed characters: Numbers 0 to 9, letters a to z, letters A to Z, and the underscore character
- Upper and lower case characters are distinguished but do not make variables different. For example, “aBc” and “AbC” are treated as the same variable, but the variable “aBc” is recorded as “aBc”.
- Variable names cannot begin with a number (0 to 9).
- There cannot be two or more consecutive underscore characters (_).



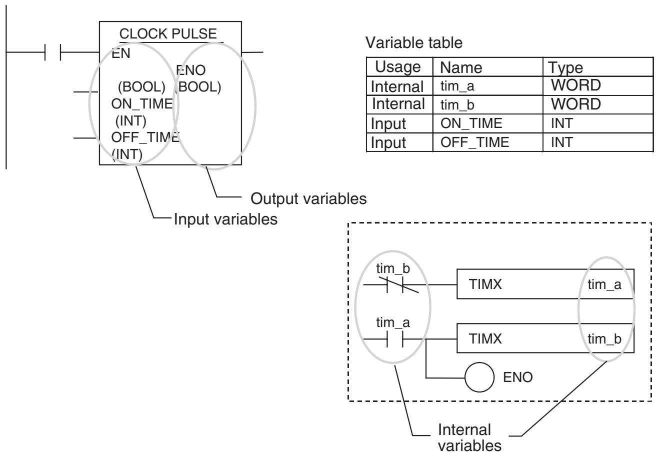
Language

Select either ladder or structured text.

Local Variable Definitions

Define the arguments and local variables used in the function block definition. Refer to *2-2-1 Naming Variables* for information on the specifications of local variables that can be made.

■ Variable Notation



Usage	Name	Type
Internal	tim_a	WORD
Internal	tim_b	WORD
Input	ON_TIME	INT
Input	OFF_TIME	INT

2-3-4 Instance Specifications

Composition of an Instance

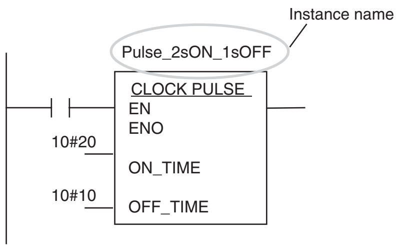
The following table lists the items that the user must set when registering an instance.

Item		Description
Instance name		Name of the instance
Parameters	Input parameters	Pass data to input variables (constants, local program variables, or physical addresses)
	Output parameters	Receive data from output variables (local program variables or physical addresses)

Instance Name

This is the name of the instance.

- Number of characters: 48 characters max.
- Allowed characters: Numbers 0 to 9, letters a to z, letters A to Z, the underscore character
- Upper and lower case characters are distinguished but do not make variables different. For example, “aBc” and “AbC” are treated as the same variable, but the variable “aBc” is recorded as “aBc”.
- Instance names cannot begin with a number (0 to 9).
- There cannot be two or more consecutive underscore characters (_).
- The instance name is displayed above the instance in the diagram.



Function Block Instance Areas

■ **Calling an Instance from Multiple Locations**

A single instance can be called from multiple locations. In this case, the internal variables (VAR) will be shared.

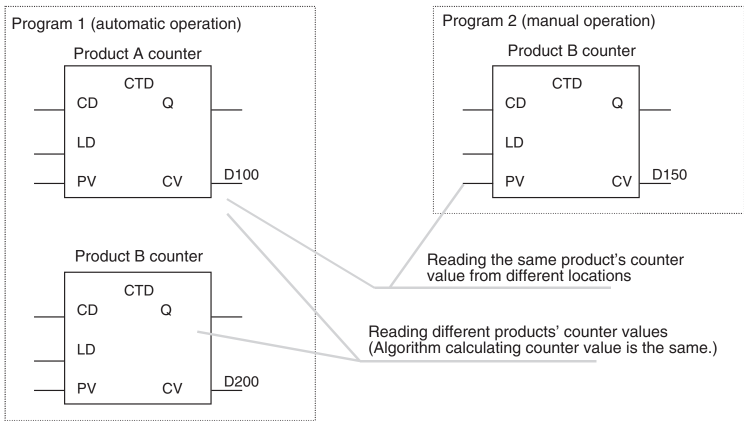
■ **Making Multiple Instances**

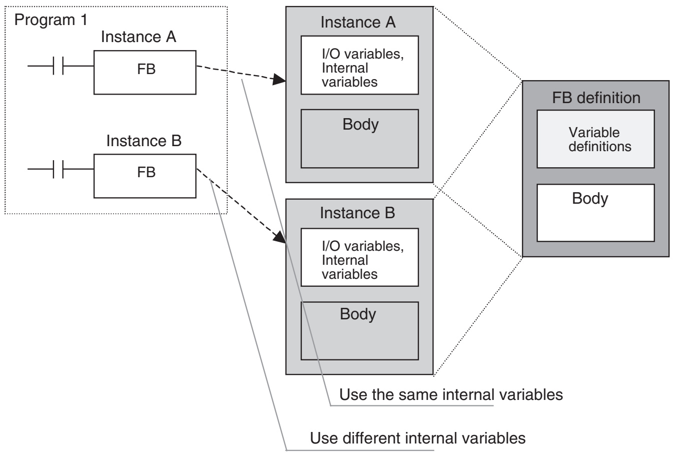
Multiple instances can be created from a single function block definition. In this case, the values of internal variables (VAR) will be different in each instance.

Example: Counting Product A and Product B

Prepare a function block definition called Down Counter (CTD) and set up counters for product A and product B. There are two types of programs, one for automatic operation and another for manual operation. The user can switch to the appropriate mode of operation.

In this case, multiple instances will be created from a single function block. The same instance must be accessible from multiple locations.

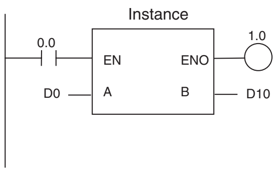

Program 1 (automatic operation)
Product A counter
CTD
CD
Q
LD
PV
CV
D100
Product B counter
CTD
CD
Q
LD
PV
CV
D200
Program 2 (manual operation)
Product B counter
CTD
CD
Q
LD
PV
CV
D150
Reading the same product's counter value from different locations
Reading different products' counter values (Algorithm calculating counter value is the same.)


Program 1
Instance A
FB
Instance B
FB
Instance A
I/O variables, Internal variables
Body
Instance B
I/O variables, Internal variables
Body
FB definition
Variable definitions
Body
Use the same internal variables
Use different internal variables

Operating Specifications

Calling Instances

The user can call an instance from any location. The instance will be executed when the input to EN is ON.

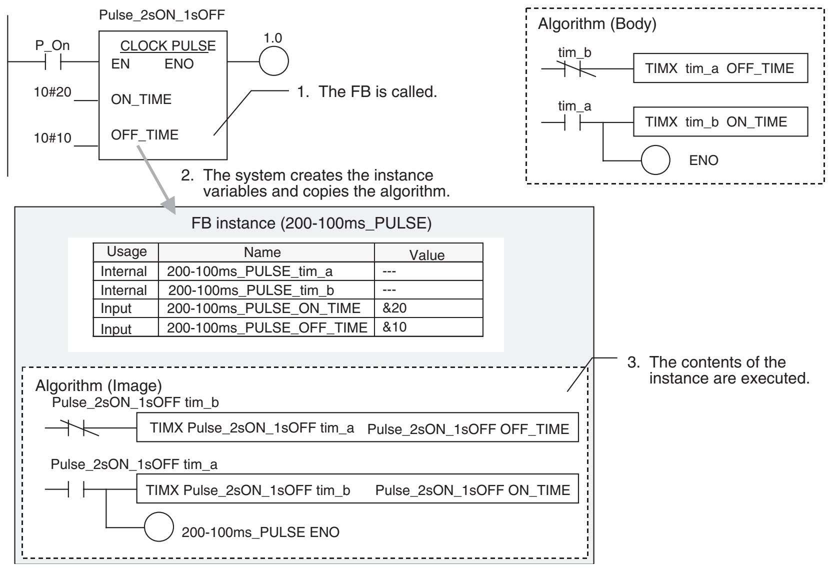


In this case, the input to EN is bit 0.0 at the left of the diagram.

- When the input to EN is ON, the instance is executed and the execution results are reflected in bit 1.0 and word D10.
- When the input to EN is OFF, the instance is not executed, bit 1.0 is turned OFF, and the content of D10 is not changed.

Operation when the Instance Is Executed

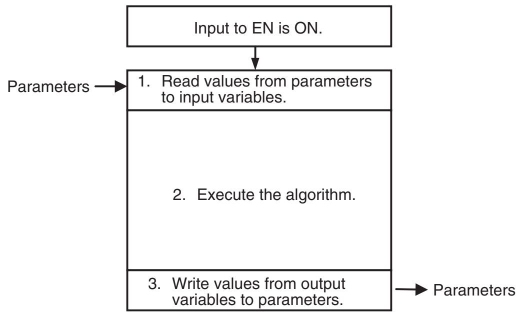
The system calls a function block when the input to the function block's EN input variable is ON. When the function block is called, the system creates the instance's local variables and copies the algorithm registered in the function block. The instance is then executed.


Pulse_2sON_1sOFF
P_On
CLOCK PULSE
EN ENO
1.0
10#20
ON_TIME
10#10
OFF_TIME
1. The FB is called.
2. The system creates the instance variables and copies the algorithm.
Algorithm (Body)
tim_b
TIMX tim_a OFF_TIME
tim_a
TIMX tim_b ON_TIME
ENO
FB instance (200-100ms_PULSE)
3. The contents of the instance are executed.
Algorithm (Image)
Pulse_2sON_1sOFF tim_b
TIMX Pulse_2sON_1sOFF tim_a Pulse_2sON_1sOFF OFF_TIME
Pulse_2sON_1sOFF tim_a
TIMX Pulse_2sON_1sOFF tim_b Pulse_2sON_1sOFF ON_TIME
200-100ms_PULSE ENO

Usage	Name	Value
Internal	200-100ms_PULSE_tim_a	---
Internal	200-100ms_PULSE_tim_b	---
Input	200-100ms_PULSE_ON_TIME	&20
Input	200-100ms_PULSE_OFF_TIME	&10

The order of execution is as follows:

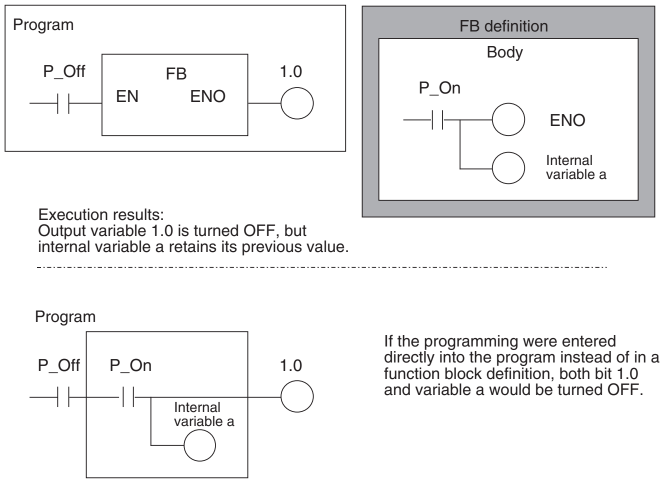
1. Read data from parameters to input variables (VAR_INPUT).
2. Execute the algorithm.
3. Write data from output variables (VAR_OUTPUT) to parameters.


Input to EN is ON.
Parameters
1. Read values from parameters to input variables.
2. Execute the algorithm.
3. Write values from output variables to parameters.
Parameters

Note Data cannot be exchanged with parameters while the algorithm is being executed. In addition, if an output variable is not changed by the execution of the algorithm, the output parameter will retain its previous value.

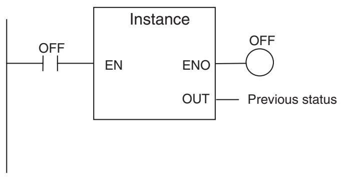
Operation when the Instance Is Not Executed

When the input to the function block's EN input variable is OFF, the function block is not called, so the internal variables of the instance do not change.


Program
P_Off
FB
EN
ENO
1.0
FB definition
Body
P_On
ENO
Internal variable a
Execution results:
Output variable 1.0 is turned OFF, but internal variable a retains its previous value.
Program
P_Off
P_On
1.0
Internal variable a
If the programming were entered directly into the program instead of in a function block definition, both bit 1.0 and variable a would be turned OFF.

Output Variables when a FB Is Not Executed

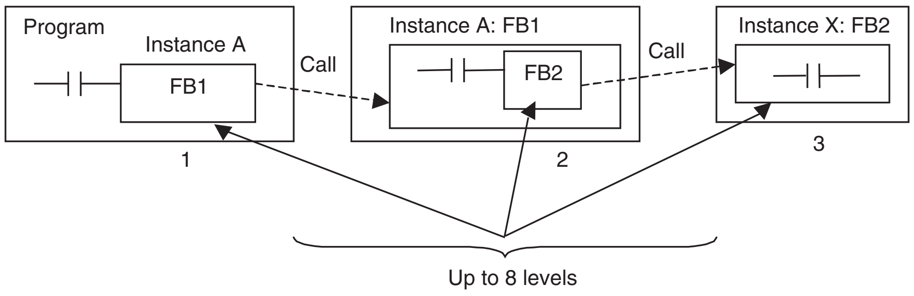
NEO will have the status of the power flow (P.F.: execution condition). User-defined output variables will retain their previous status.



⚠ Caution An instance will not be executed while its EN input variable is OFF, so Differentiation and Timer instructions will not be initialized while EN is OFF. If Differentiation or Timer instructions are being used, use the Always ON Flag (P_On) for the EN input condition and include the instruction's input condition within the function block definition.

Nesting

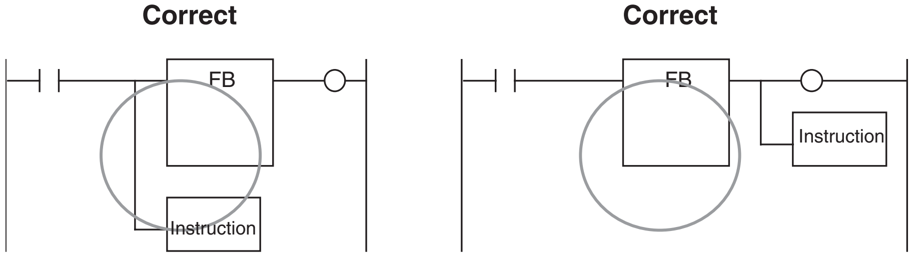
A function block can be called from another function block, i.e., nesting is supported (up to 8 levels).



Placement of Instances in Program

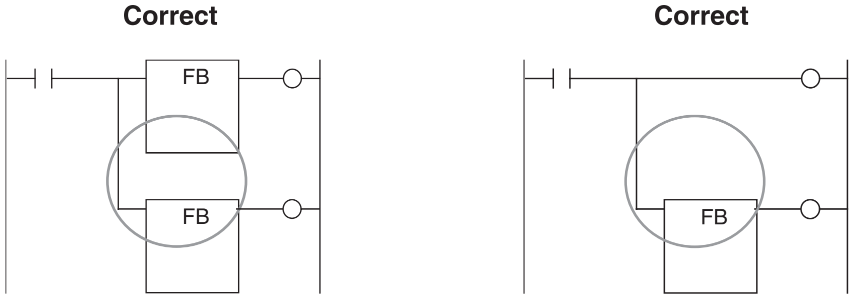
Branches Allowed on the Left and Right Sides of the Instance

Branches can be placed on the left side of the instance (EN input side) as well as the right side of the instance (to the left of ENO).



Multiple Instances in a Single Rung

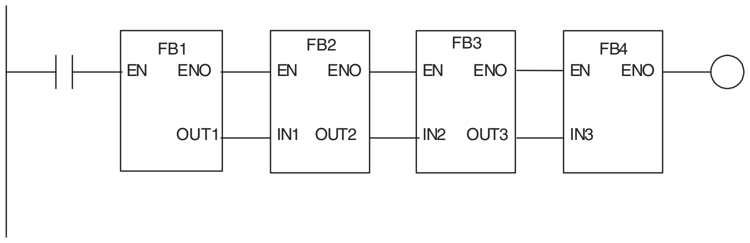
A single program rung can have more than one instance.



Connections between Instances are Allowed

An instance's ENO output can be input directly to another instance's EN input. Furthermore, an instance's output parameter can be input directly to another instance's input parameter.

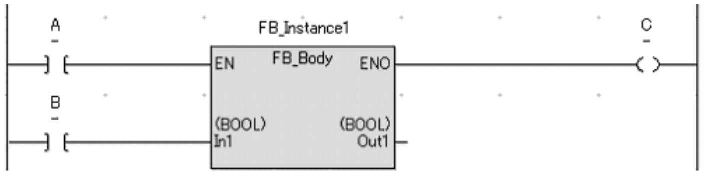
Note The data type of the input parameter must match the data type of the output parameter. As long as the data types match, other data types can be used besides BOOL-BOOL, including WORD-WORD and DWORD-DWORD.



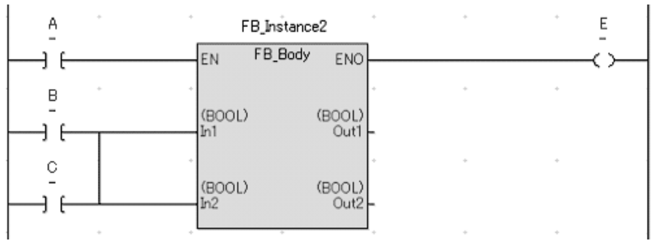
Up to 4 instances can be connected between the left and right bus bars in the ladder diagram. If more than 4 are entered, the additional instructions will wrap around. There can be up to 150 wrap-around lines per program section. Function block connections are allowed within that range.

AND and OR Conditions Allowed on Input Side

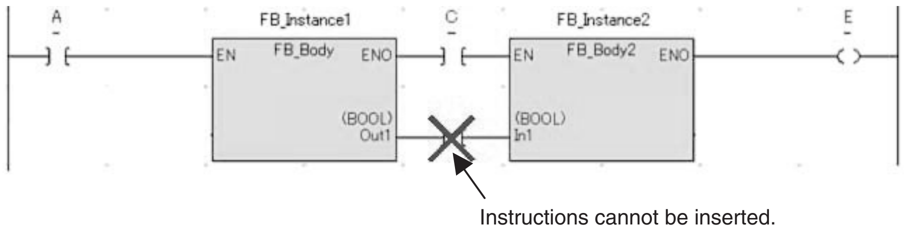
AND or OR conditions can be entered at the instance's left side (either the EN input side or the input parameter side).



Input parameters can also be connected together.



An AND or OR condition can be inserted between function blocks, as shown in the following diagram. On the other hand, instructions cannot be inserted between parameters, as explained in *Restrictions on Placement of Instances*.

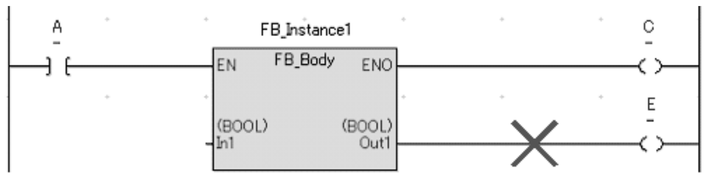


Restrictions on Placement of Instances

There are some restrictions on the placement of instances in the program.

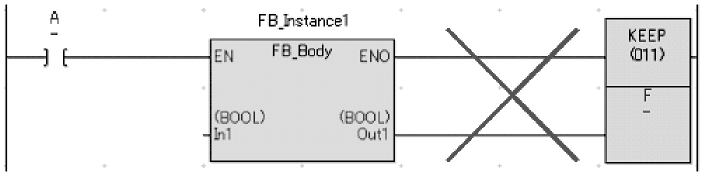
Connecting an Output Bit to an Output Parameter

An output bit cannot be connected to an output parameter.



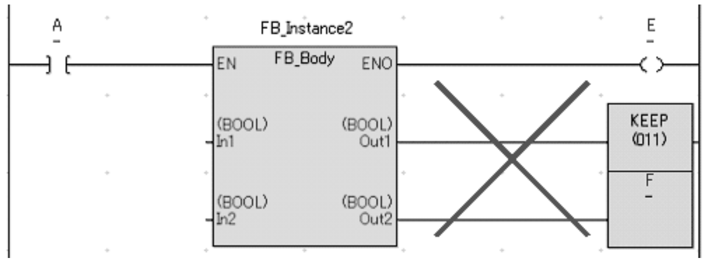
Connecting an Instruction to an ENO Output and Output Parameter

A special instruction cannot be connected to an ENO output and output parameter.



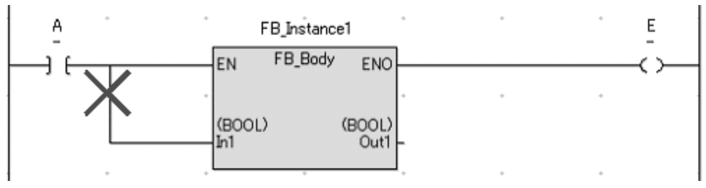
Connecting an Instruction to Output Parameters

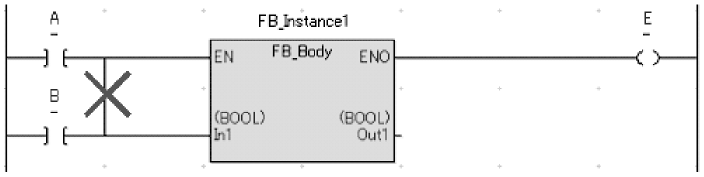
A special instruction cannot be connected to output parameters.



Connecting an EN Input with an Input Parameter

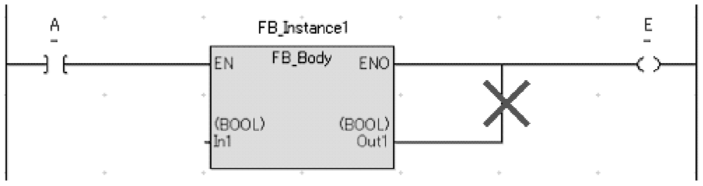
A EN input cannot be connected to an input parameter.

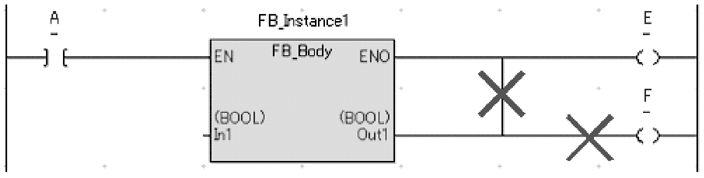




Connecting an ENO Output with an Output Parameter

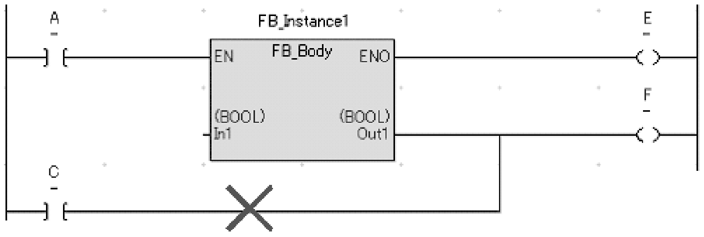
A ENO output cannot be connected to an output parameter.





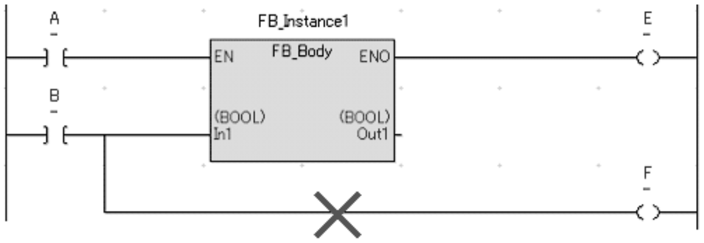
Connecting an Input to an Output Parameter

An input cannot be connected directly to an output parameter.



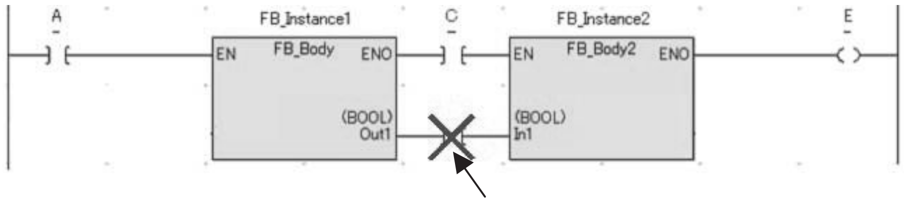
Connecting an Input Parameter to an Output Bit

An input parameter cannot be connected directly to an output.



Inserting an AND or OR Condition between Parameters

An AND or OR condition cannot be inserted between parameters.



Instructions cannot be inserted.

Parameter Specifications

Allowed Parameter Inputs

The following data can be set in input and output parameters.

Input Parameters:

- Constants (See note.)
- Local program variables (for example: aa)
  (The data type must match the corresponding input variable.)
- Physical addresses (for example: D00100 or 0000.00)

Output Parameters:

- Local program variables (for example: aa)
  (The data type must match the corresponding output variable.)
- Physical addresses (for example: D00100 or 0000.00)

Note The input range of the constant depends on the data type, as shown in the following table.

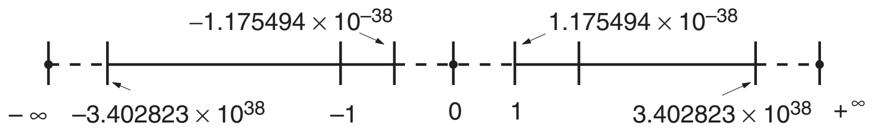
Input the desired value in the *Initial Value* Field in the *Edit Variables* Dialog Box. Input a value consistent with the variable's data type, as shown in the following table.

Data type	Content	Range of values	Initial value input method
BOOL	Bit data	16#0 or 16#1	Input as unsigned hexadecimal after “16#”.
INT	Integer	–10#32768 to +10#32767	Input as signed decimal after “+10#” or “–10#”. (If “10#” is input, a + sign will be attached automatically.)
UINT	Unsigned integer	10#0 to 10#65535	Input as unsigned decimal after “10#”.
DINT	Double integer	–10#2147483648 to +10#2147483647	Input as signed decimal after “+10#” or “–10#”. (If “10#” is input, a + sign will be attached automatically.)
UDINT	Unsigned double integer	10#0 to 10#4294967295	Input as unsigned decimal after “10#”.
WORD	16-bit data	16#0000 to16#FFFF	Input as unsigned hexadecimal after “16#”.
DWORD	32-bit data	16#00000000 to16#FFFFFFFF	Input as unsigned hexadecimal after “16#”.
REAL	Real number	Refer to the details on floating-point data below.	Input a signed numerical value. Input the value after the sign. E.g., +1.0, —0.23, +9.87E-3
STRING	Character string data (ASCII data)	127 characters	---

Floating-point Data

The following data can be expressed by floating-point data:

- $-\infty$
- $-3.402823 \times 10^{38} \le \text{value} \le -1.175494 \times 10^{-38}$
- 0
- $1.175494 \times 10^{-38} \le \text{value} \le 3.402823 \times 10^{38}$
- $+\infty$
- Not a number (NaN)



Special Numbers

The formats for NaN, ±∞, and 0 are as follows:

NaN*:	e = 255, f ≠ 0
+∞:	e = 255, f = 0, s = 0
–∞:	e = 255, f = 0, s = 1
0:	e = 0

*NaN (not a number) is not a valid floating-point number. Executing floating-point calculation instructions will not result in NaN.

Restrictions on Variable Data Types

Parameters cannot be input to TIMER or COUNTER variables or output from TIMER or COUNTER variables.

Timing of Data Transfers to/from Variables

Input data is passed from parameters to the corresponding input variables before the algorithm is processed. Output variable data is passed to the corresponding parameters after algorithm processing is completed. Consequently, values cannot be passed from parameters to input variables while the algorithm is being executed.

Restrictions on Parameter Data Quantity

The maximum amount of a parameter I/O data is 1,024 words per instance.

Connecting a Bit Control Instruction to a Parameter

For details on connecting a parameter to a Bit Control Instruction, refer to the sections *Placement of Instances in Program* and *Restrictions on Placement of Instances* above.

2-3-5 Restrictions on Function Blocks

Instructions Restricted in Ladder Programs

Instructions Prohibited in Function Block Definitions

The END (001) instruction cannot be used in function block definitions. Any other instructions that can be used in the program can also be used in function blocks.

I/O Variable Restrictions (Unsupported Data Areas)

Addresses in the following data areas cannot be used as parameters for input and output variables.

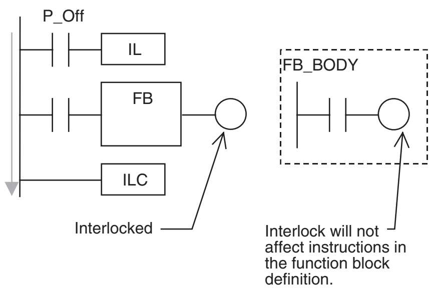
- Index Registers and Data Registers (Neither indirect nor direct addressing is supported.)
- Indirect addressing of DM or EM Area addresses (Neither binary-mode nor BCD-mode indirect addressing is supported.)

Refreshing Timer and Counter PVs

Timer and counter PVs are always stored in binary mode, so PVs of all Timer and Counter Instructions must be treated as binary data whether or not the instructions are in function blocks.

Interlocks

When a function block is called from an interlocked program section, the contents of the function block definition will not be executed. The interlocked function block will behave just like an interlocked subroutine.

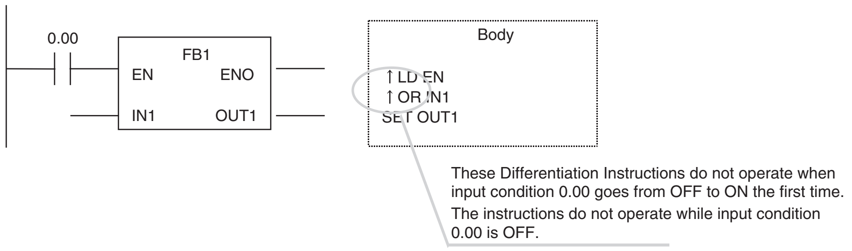

P_Off
IL
FB
ILC
FB_BODY
Interlocked
Interlock will not affect instructions in the function block definition.

Differentiation Instructions in Function Block Definitions

An instance will not be executed while its EN input variable is OFF, so the following precautions are essential when using a Differentiation Instruction in a function block definition. (Differentiation Instructions include DIFU, DIFD, and any instruction with an @ or % prefix.)

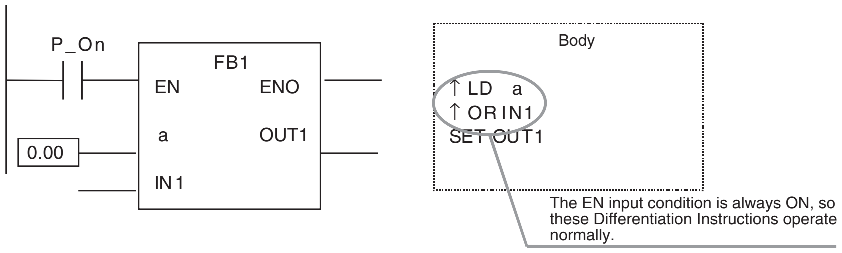
- As long as the instance's EN input variable is OFF, the execution condition will retain its previous status (the last status when the EN input variable was ON) and the Differentiation Instruction will not operate.
- When the instance's EN input variable goes ON, the present execution condition status will not be compared to the last cycle's status. The present execution condition will be compared to the last condition when the EN input variable was ON, so the Differentiation Instruction will not operate properly. (If the EN input variable remains ON, the Differentiation Instruction will operate properly when the next rising edge or falling edge occurs.)

Example:


0.00
FB1
EN
ENO
IN1
OUT1
Body
↑LD EN
↑OR IN1
SET OUT1
These Differentiation Instructions do not operate when input condition 0.00 goes from OFF to ON the first time.
The instructions do not operate while input condition 0.00 is OFF.

If Differentiation Instructions are being used, always use the Always ON Flag (P_On) for the EN input condition and include the instruction's input condition within the function block definition.

Example:

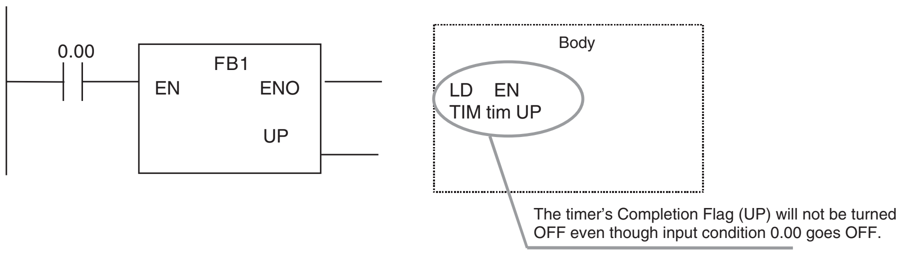

P_On
FB1
EN
ENO
a
OUT1
0.00
IN1
Body
↑LD a
↑OR IN1
SET OUT1
The EN input condition is always ON, so these Differentiation Instructions operate normally.

Timer Instructions in Function Block Definitions

An instance will not be executed while its EN input variable is OFF, so the following precautions are essential when using a Timer Instruction in a function block definition.

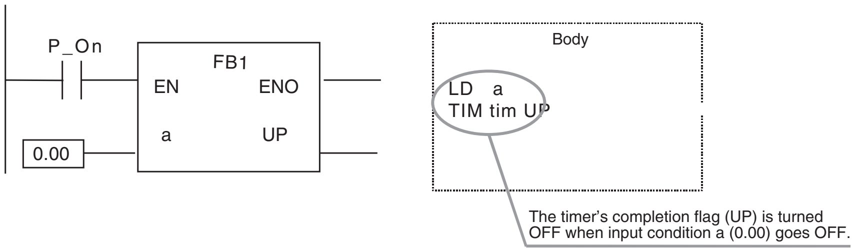
The Timer Instruction will not be initialized even though the instance's EN input variable goes OFF. Consequently, the timer's Completion Flag will not be turned OFF if the EN input variable goes OFF after the timer started operating.

Example:


0.00
FB1
EN
ENO
UP
Body
LD EN
TIM tim UP
The timer's Completion Flag (UP) will not be turned OFF even though input condition 0.00 goes OFF.

If Timer Instructions are being used in the function block definition, always use the Always ON Flag (P_On) for the EN input condition and include the instruction's input condition within the function block definition.

Example:


P_On
FB1
EN
ENO
a
UP
0.00
Body
LD a
TIM tim UP
The timer's completion flag (UP) is turned OFF when input condition a (0.00) goes OFF.

- If the same instance containing a timer is used in multiple locations at the same time, the timer will be duplicated.

ST Programming Restrictions

- Only the following statements and operators are supported.
    - Assignment statements
    - Selection statements (CASE and IF statements)
    - Iteration statements (FOR, WHILE, and REPEAT statements)
    - Arithmetic operators
    - Logical operators
    - Comparison operators
    - Comments
- The TIMER and COUNTER data types cannot be used.
- Use parentheses to indicate the priority of arithmetic operations.
  Example: Var_D:=(Var_A+Var_B)*Var_C;
- Tabs and spaces can be used to indent text.

Online Editing Restrictions

The following online editing operations cannot be performed on the user program in the CPU Unit.

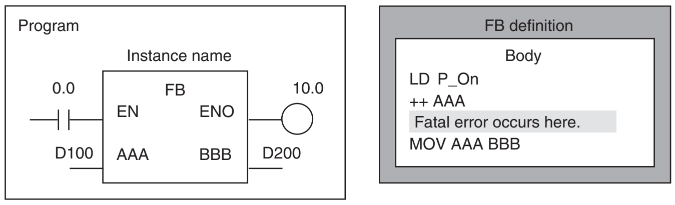
- I/O variables cannot be added, deleted, or changed.
- Internal variables cannot be deleted or changed.
- FB body names cannot be changed.
- FB instance names cannot be changed.
- FB instances cannot be added.
- Comments cannot be edited.

Note (1) Internal variables can be added, but there are restrictions on the variables that can be added.

(2) Global variables can be added.

Error-related Restrictions

If a fatal error occurs in the CPU Unit while a function block definition is being executed, ladder program execution will stop at the point where the error occurred.



In this case, the MOV AAA BBB instruction will not be executed and output variable D200 will retain the same value that it had before the function block was executed.

SECTION 3
Installation

This section describes software installation.

3-1 Installation Preparations

3-1-1 System Requirements

A computer with the following specifications is required to install the NE Programmer.

Item	Specification
Computer	IBM PC/AT or compatible
CPU	Pentium 300 MHz or higher (Pentium 1 GHz or higher recommended)*1
OS	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows Vista*1
Supported languages	Japanese/English
Memory	512 MB min.*1
HDD	200 MB min. of available space
Monitor	S-VGA or better
CD-ROM	1 min.
Communications port to connect to the NE1S	One or more of the following: USB port, RS-232C port, or Ethernet port Or one of the following: DeviceNet Interface Card (see note) or RA Communications Card (using RS Linx communications driver) **Note** Connections with NE1S CPU Units on the network is possible via USB port, RS-232C ports, or Ethernet ports.

*1 Requirements for Windows Vista:
CPU: Pentium, 1 GHz min., Memory: 1 GB

3-1-2 Installation Types

Installing the NE Programmer

For details on installing the NE Programmer, refer to *3-2 Installing the NE Programmer*. One or both of the NE Programmer and Network Configurator can be installed.

Installing the USB Driver

For details on connecting online using USB, refer to *3-3 Installing the USB Driver*.

3-2 Installing the NE Programmer

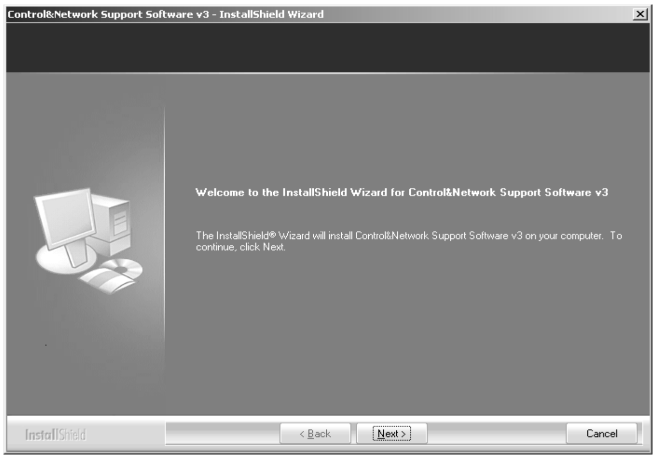
3-2-1 Installation Procedure

Use the following procedure to install the NE Programmer.

Note Administrator privileges are required to install the software.

1,2,3...

1. Insert the installation CD in the CD-ROM drive.
2. Execute setup.exe from the CD-ROM using either of the following methods.
   - Click the icon to access the CD-ROM and double-click the setup.exe file.
   - Select ***Run*** from the Start menu, browse the CD-ROM for the setup.exe file and then click the **OK** Button. The following window will be displayed.

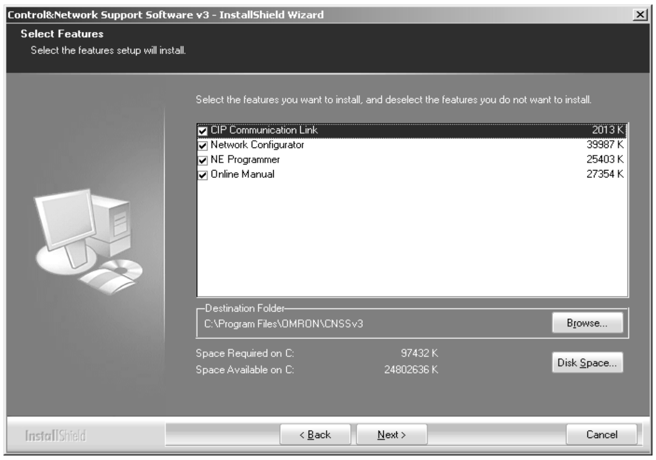


Control&Network Support Software v3 - InstallShield Wizard

Welcome to the InstallShield Wizard for Control&Network Support Software v3

The InstallShield® Wizard will install Control&Network Support Software v3 on your computer. To continue, click Next.

3. Click the **Next** Button. The License Agreement Window will be displayed.
4. Select the ***I accept the terms of the License Agreement*** option and then click the **Next** Button. The Customer Information Window will be displayed.
5. Enter the *User Name, Company Name,* and *Serial Number,* and then click the **Next** Button. The following Select Features Window will be displayed.



Control&Network Support Software v3 - InstallShield Wizard

Select Features

Select the features setup will install.

Select the features you want to install, and deselect the features you do not want to install.

✓ CIP Communication Link	2013 K
✓ Network Configurator	39987 K
✓ NE Programmer	25403 K
✓ Online Manual	27354 K

Destination Folder

C:\Program Files\OMRON\CNSSv3

Space Required on C: 97432 K

Space Available on C: 24802636 K

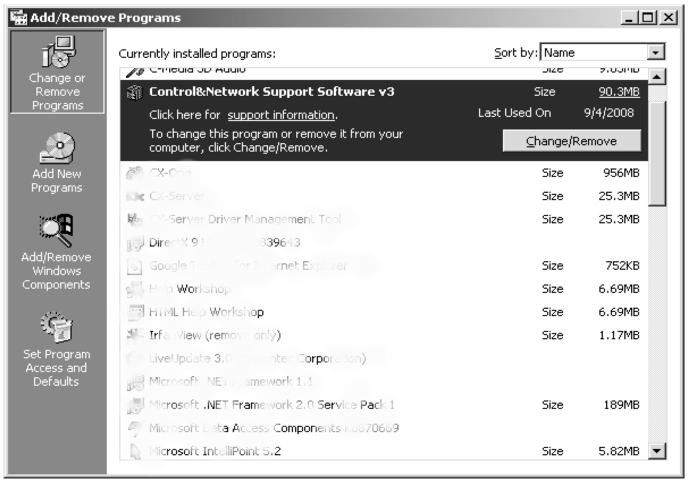
6. Select ***CIP Communication Link*** and any other programs to be installed (*CIP Communication Link* must be selected), specify the installation destination, and click the **Next** Button. The Ready to Install the Program Window will be displayed.
7. Click the **Install** Button to start installation. When installation completes normally, the InstallShield Wizard Complete Window will be displayed.
8. Click the **Finish** Button.

3-2-2 Uninstallation Procedure

Use the following procedure to uninstall the NE Programmer.

1,2,3... 1. Select ***Settings - Control Panel - Add/Remove Programs*** (or ***Add/Remove Applications***) from the Start menu.

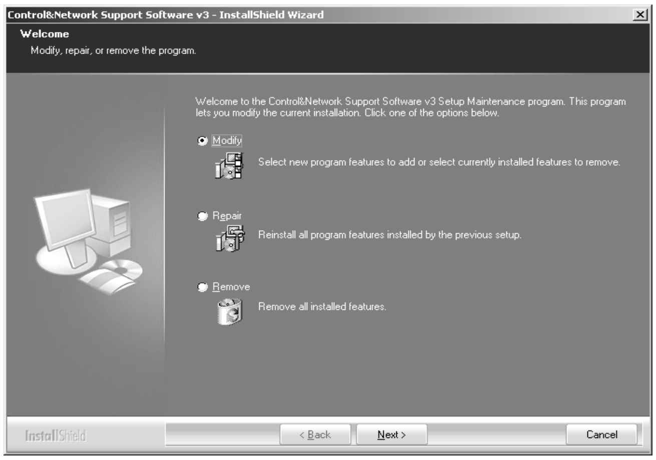
2. Select ***Control&Network Support Software v3*** in the Add/Remove Programs Dialog Box (or Add/Remove Applications Dialog Box), as shown below, and click the **Change/Remove** Button.


Add/Remove Programs
Change or Remove Programs
Add New Programs
Add/Remove Windows Components
Set Program Access and Defaults
Sort by: Name
Change/Remove

Currently installed programs:

C-Media 3D Audio	Size	9.03MB
Control&Network Support Software v3	Size	90.3MB
Click here for support information.	Last Used On	9/4/2008
To change this program or remove it from your computer, click Change/Remove.		
CX-One	Size	956MB
CX-Server	Size	25.3MB
CX-Server Driver Management Tool	Size	25.3MB
DirectX 9 Hotfix - KB839643		
Google Toolbar for Internet Explorer	Size	752KB
Help Workshop	Size	6.69MB
HTML Help Workshop	Size	6.69MB
IrfanView (remove only)	Size	1.17MB
LiveUpdate 3.0 (Symantec Corporation)		
Microsoft .NET Framework 1.1		
Microsoft .NET Framework 2.0 Service Pack 1	Size	189MB
Microsoft Data Access Components KB870669		
Microsoft IntelliPoint 5.2	Size	5.82MB

• The following window will be displayed.



Welcome
Modify, repair, or remove the program.

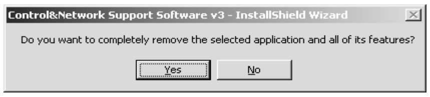
Welcome to the Control&Network Support Software v3 Setup Maintenance program. This program lets you modify the current installation. Click one of the options below.

[✓ Modify] Select new program features to add or select currently installed features to remove.

Repair Reinstall all program features installed by the previous setup.

Remove Remove all installed features.

3. Click the **Remove** Button, and then click the **Next** Button.



Do you want to completely remove the selected application and all of its features?

4. Click the **Yes** Button.
   The software will be uninstalled. When uninstallation is completed normally, the Maintenance Complete Window will be displayed.
5. Click the **Finish** Button.

3-2-3 Upgrading Software Versions

Use the following procedure to upgrade the version of the NE Programmer.

Installing an Update

Install the updated version without uninstalling the existing software.

1,2,3...

1. The updated version is provided as an executable (.exe) file, so either of the following methods can be used to execute the update file.
   - Double-click the file and click the **OK** Button.
   - Select ***Run*** from the Start menu, browse and select the update file, and click the **OK** Button.

Installing a Release

1,2,3...

1. Uninstall the software to be upgraded (refer to *3-2-2 Uninstallation Procedure*).
2. Install the new software version (refer to *3-2-1 Installation Procedure*).

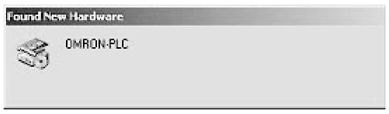
3-3 Installing the USB Driver

Note Administrator privileges are required to install the software.

Use the following procedure to install the USB driver.

1,2,3...

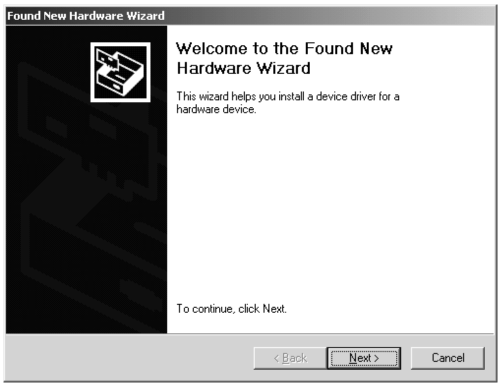
1. Connect the CPU Unit and the computer with a USB cable.
2. Turn ON the power supply to the CPU Unit.
   The following dialog box will be displayed.



Found New Hardware

OMRON-PLC

After a few moments, the following dialog box will be displayed.



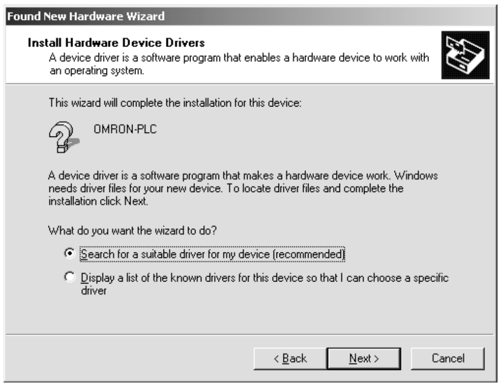
Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install a device driver for a hardware device.

To continue, click Next.

3. Click the **Next** Button.
   The following dialog box will be displayed.



Found New Hardware Wizard

Install Hardware Device Drivers
A device driver is a software program that enables a hardware device to work with an operating system.

This wizard will complete the installation for this device:

OMRON-PLC

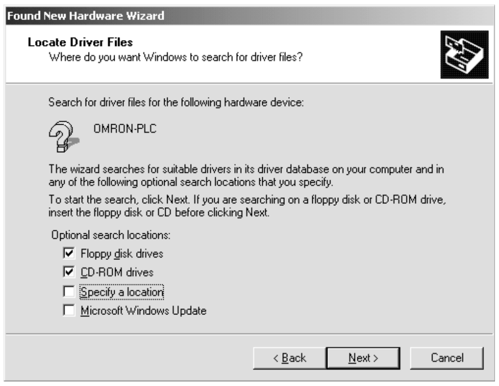
A device driver is a software program that makes a hardware device work. Windows needs driver files for your new device. To locate driver files and complete the installation click Next.

What do you want the wizard to do?

[✓ Search for a suitable driver for my device (recommended)]

Display a list of the known drivers for this device so that I can choose a specific driver

4. Select the Search for a suitable driver for my device (recommended) option and click the **Next** Button.
   The following dialog box will be displayed.



Found New Hardware Wizard

Locate Driver Files
Where do you want Windows to search for driver files?

Search for driver files for the following hardware device:

OMRON-PLC

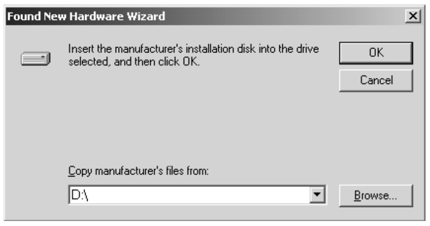
The wizard searches for suitable drivers in its driver database on your computer and in any of the following optional search locations that you specify.

To start the search, click Next. If you are searching on a floppy disk or CD-ROM drive, insert the floppy disk or CD before clicking Next.

Optional search locations:

- ✓ Floppy disk drives
- ✓ CD-ROM drives
- Specify a location
- Microsoft Windows Update

5. Select only the *Specify a location* option (do not select any other options), and click the **Next** Button.
   The following dialog box will be displayed.



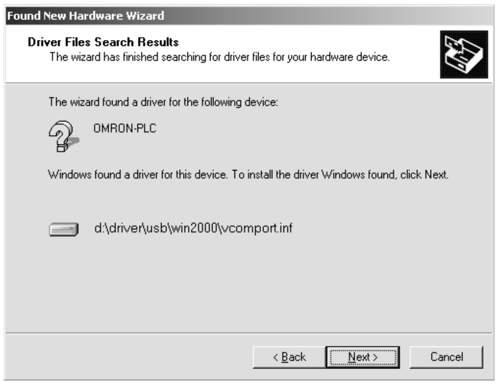
Found New Hardware Wizard

Insert the manufacturer's installation disk into the drive selected, and then click OK.

Copy manufacturer's files from:

D:\

6. Click the **Browse** Button, then in the dialog box that is displayed specify the CD-ROM\driver\NE1S_usb\win2000 folder for an NE1S-series CPU Unit or the CD-ROM\driver\CJ2H_usb folder for a CJ2 CPU Unit, and then click the **OK** Button.
   The following dialog box will be displayed.



Found New Hardware Wizard

Driver Files Search Results

The wizard has finished searching for driver files for your hardware device.

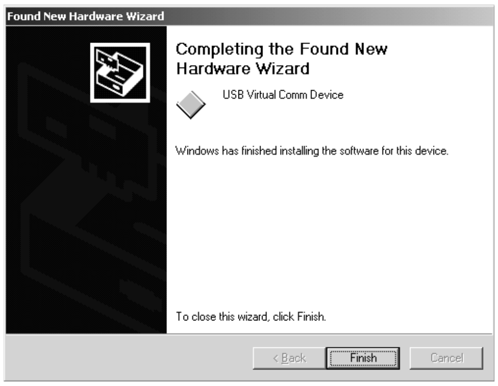
The wizard found a driver for the following device:

OMRON-PLC

Windows found a driver for this device. To install the driver Windows found, click Next.

d:\driver\usb\win2000\vcomport.inf

7. Click the **Next** Button.
   The following dialog box will be displayed if the USB driver is installed normally.



Found New Hardware Wizard

Completing the Found New Hardware Wizard

USB Virtual Comm Device

Windows has finished installing the software for this device.

To close this wizard, click Finish.

8. Click the **Finish** Button.
   This completes the installation of the USB driver.

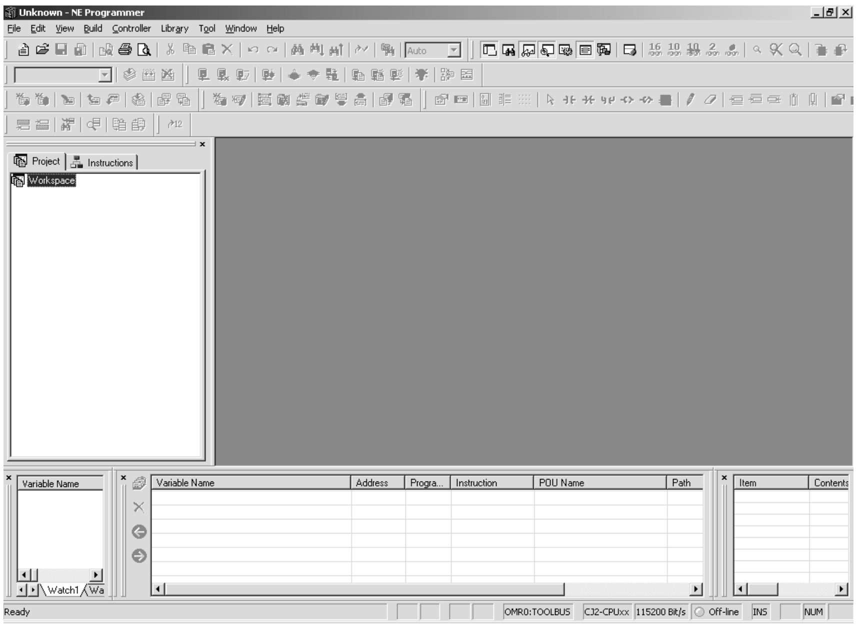
SECTION 4
Outline of Operations and Functions of the NE Programmer

This section provides an outline of the operations and functions of the NE Programmer.

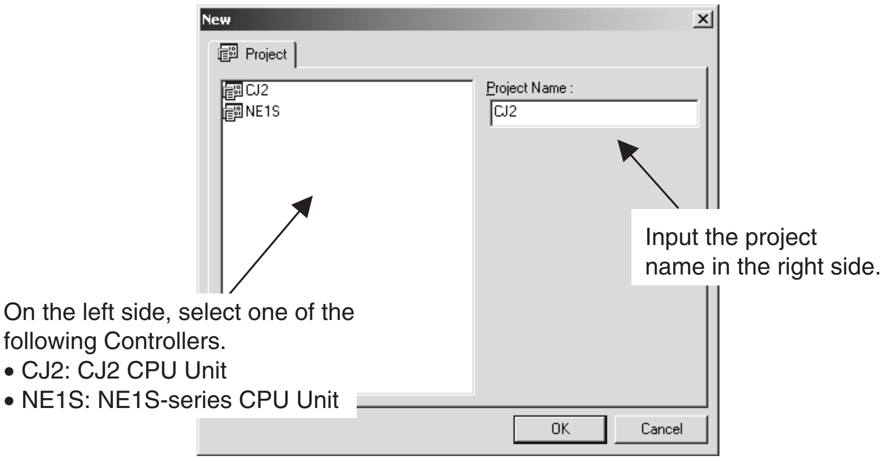
4-1 Starting the NE Programmer

1,2,3...

1. To start the Network Configurator, select ***Start - Program - OMRON Control&Network Support Software v3 - NE Programmer - NE Programmer***. The following initial window will be displayed.


Unknown - NE Programmer
File Edit View Build Controller Library Tool Window Help
Project Instructions
Workspace
Variable Name
Watch1 Wa
Variable Name Address Progra... Instruction POU Name Path
Item Contents
Ready OMR0:TOOLBUS CJ2-CPUxx 115200 Bit/s Off-line INS NUM

2. Select ***File - New - Project***. The following New Dialog Box will be displayed.
   The dialog box for creating a new project will be displayed as shown in the following figure.

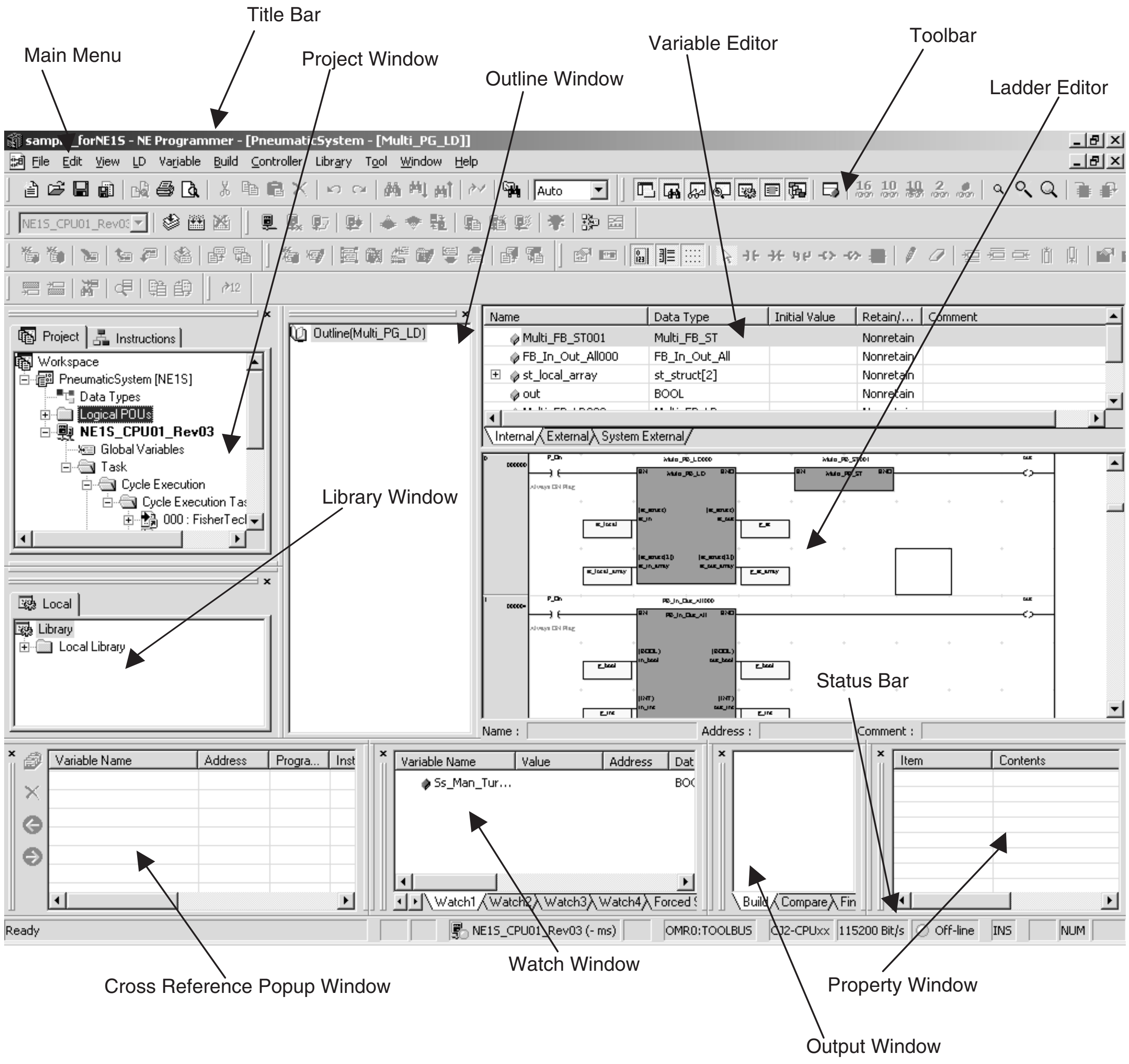


Input the project name in the right side.

On the left side, select one of the following Controllers.
• CJ2: CJ2 CPU Unit
• NE1S: NE1S-series CPU Unit

4-2 Main Window

This section describes the main functions of the Main Window.



Name	Function
Title Bar	Displays the file name.
Main Menu	Commands are selected from this menu.
Tool Bar	Functions are selected by clicking the icons.
Project Window	Used to manage programs and data.
Outline Window	Displays an outline of the ladder program currently being edited. If any part of the outline is clicked, a jump will be made to the corresponding location in the program.
Variable Editor	Used to register and edit variables. (There is a Variable Editor for local variables and one for global variables).
Ladder Editor	Used to create and edit ladder programs. Up to 15 Ladder Editor Windows can be displayed at the same time.
Library Window	Used to manage user-defined libraries (programs, function blocks, and program parts).
Cross Reference Popup Window	Displays other instructions that use the variable at the cursor location. If any cross-referenced instruction is clicked, a jump will be made to the corresponding location in the program.
Watch Window	Displays the present values of registered variables and physical addresses.

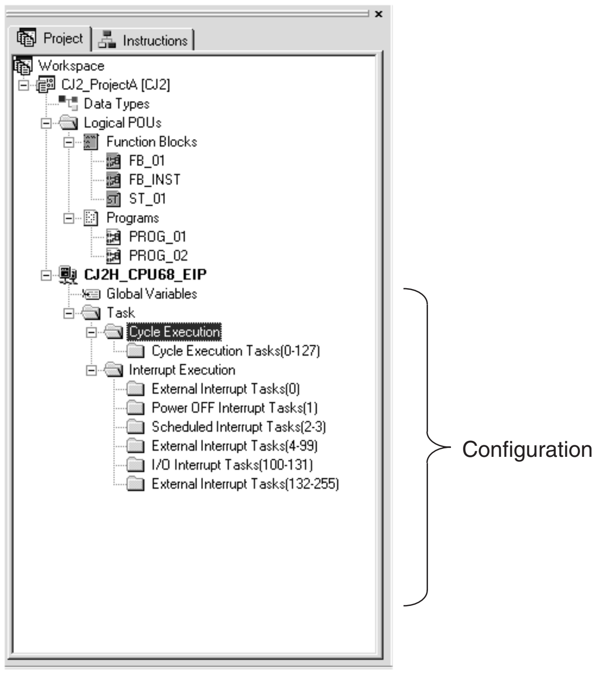
Name	Function
Output Window	Displays various information, such as compiling error information and the results of program comparison.
Property Window	Displays the properties of the instruction at the cursor position.
Status Bar	Displays information, such as the configuration name and online/offline status.

4-3 Project Window

This section describes the parts of the Project Window.

In the initial status, there are no elements created in the Project Window except for the NE Programmer workspace.

Refer to *SECTION 5 Programming* for the procedures to create elements.



```
Workspace
⊟ CJ2_ProjectA [CJ2]
    Data Types
  ⊟ Logical POUs
    ⊟ Function Blocks
        FB_01
        FB_INST
        ST_01
    ⊟ Programs
        PROG_01
        PROG_02
  ⊟ CJ2H_CPU68_EIP
      Global Variables
    ⊟ Task
      ⊟ Cycle Execution
          Cycle Execution Tasks(0-127)
      ⊟ Interrupt Execution
          External Interrupt Tasks(0)
          Power OFF Interrupt Tasks(1)
          Scheduled Interrupt Tasks(2-3)
          External Interrupt Tasks(4-99)
          I/O Interrupt Tasks(100-131)
          External Interrupt Tasks(132-255)
```

Name	Description
Workspace	Used as a workspace for the NE Programmer.
Project	A project is a unit of data consisting of the library objects, data types, logical POUs, and configurations that are being edited.
Data Types	A folder that displays the data types that can be used in the project and is used to declare data types.
Logical POUs	Logical POUs (program organization units) are units used to configure programs. POUs are the units by which software can be reused. POUs can be one of two types of element, function blocks and programs. Function blocks are called from a program and programs are allocated to tasks for execution.
Configuration	A configuration is the element that corresponds to an entire PLC system. Global variables can be declared for a configuration.
Resource	Resources consist of programs allocated to tasks and local variables. With the NE1S, one resource folder exists in the configuration.

4-4 Menu Item Lists

This section lists the functions on the main and submenus in the Main Window of the NE Programmer.

4-4-1 File Menu

Menu	Shortcut	Function
New - Project	Ctrl + N	Creates a new project.
New - Configuration		Creates a new configuration.
New - POU		Creates a new logical POU.
Open	Ctrl + O	Opens an existing project.
Save	Ctrl + S	Saves the project being edited (overwrites current project data).
Save As		Saves the project being edited under a new name.
Save Changes to project	Ctrl + Shift + S	Updates the project for the program or function block that is being edited.
Edit Data types		Opens the Data Type Editor.
Configuration - Edit Global Variables	Alt + G	Edits global variables.
Configuration - Replace physical addresses in programs	Ctrl + Shift + R	Replaces physical addresses in the program with global variables specified by that physical address.
Task - Create		Adds a task.
Task - Allocate - (*program name)*		Allocates a program to the selected task.
Task - Release		Releases a task allocation.
Task - Execute on Startup		Sets the task to be active (i.e., to be executed) at startup.
Task - Standby on Startup		Sets the task to be inactive (i.e., not to be executed) at startup.
Logical POU - Edit		Displays the Ladder Editor and Variable Editor for the currently selected program or function block.
Logical POU - Delete		Deletes the currently selected program or function block.
Logical POU - Protect		Sets read protection for a function block.
Print	Ctrl + P	Prints the specified items.
Print Preview		Displays a preview for printing.
Setup Printer		Sets the printer to use for printing.
Page Setup		Sets the margins, title, header, and footer to use for printing.
Export Variable - OMRON OPC Server File		Exports variables in external Support Software formats.
Exit		Exists the application.

4-4-2 Edit Menu

Menu	Shortcut	Function
Undo	Ctrl + Z	Undoes the previously performed operation.
Redo	Ctrl + Y	Redoes the operation that was undone with *Undo*.
Cut	Ctrl + X	Cuts the specified range of data.
Copy	Ctrl + C	Copies the specified range of data.
Paste	Ctrl + V	Pastes the contents of the clipboard.
Delete	Delete	Deletes the specified range of data.
Rename	F2	Used to change the name of a program or function block.
Select All	Ctrl + A	Selects all data.
Find	Ctrl + F	Searches for a text string.
Find Next	F3	Finds the next instance of a text string.
Find Prev	Shift + F3	Finds the previous instance of a text string.
Replace	Ctrl + H	Replaces a text string.

Menu	Shortcut	Function
Bookmark	Ctrl + B	Registers a bookmark or jumps to a bookmark.
Find in Programs	Ctrl +Shift+ F	Searches mnemonics, variables, physical addresses, instance names, line comments, and instruction comments in all POUs. The specific items to be searched can be specified. The search results are displayed in the Output Window and relevant locations can be jumped to by double-clicking on the search results.
Cross reference - Register		Registers the details currently displayed in the Cross Reference Window as log data.
Cross reference - Remove		Deletes registered log data.
Cross reference - Prev		Moves back one log.
Cross reference - Next		Moves forward one log.

Additional Menu Items when Editing Ladder Programs

Menu	Shortcut	Function
Jump - Jump	Ctrl + G	Jumps to a specified step number or rung number.
Jump - Bit Address Reference	Alt + Shift + A	Searches for program inputs and outputs that affect each other. If an input condition is specified, a search is made backward for an output with the same bit address. If an output is specified, a search is made foreword for an input condition with the same bit address.
Jump - Next Operand Reference	Alt + Shift + N	Jumps to the next operand.
Jump - Next Input	Alt + Shift + I	Jumps to the next input.
Jump - Next Output	Alt + Shift + O	Jumps to the next output.
Jump - Previous Jump Point	Alt + Shift + B	Returns to the previous instruction found with *Next Address Reference*.
Jump - Jump Variable Define		Jumps to the position in the Variable Editor where the selected instruction operand's variable is defined (declared).
Edit Instruction		Used to edit the currently selected instruction or function block instance name.
Edit Variable	F8	Used to edit the selected instruction operand's variable.
Edit Comment		Used to edit the comment for the currently selected instruction or function block.
Copy Operand	Crtl + Shift + C	Used to copy the operand only.
Update Function Block Instance		Used to update changes in the input and output variables when the input and output variables of the logical POU's function block were changed after the function block was pasted as an instance. (The ⚠: icon will appear next to an instance when there has been a change.)
Edit Using Mnemonic Editor		Used to edit one rung of a ladder program with the Mnemonic Editor.

Additional Menu Items when Editing Mnemonic Programs

Menu	Shortcut	Function
Jump - Jump	Ctrl + G	Jumps to a specified step number.
Jump - Jump Variable Define		Jumps to the position in the Variable Editor where the selected instruction operand's variable is defined (declared).
Insert upward	Ctrl + G	Inserts an instruction above the current row.
Insert downward		Inserts an instruction below the current row.

Additional Menu Items when Editing ST Programs

Menu	Shortcut	Function
Jump - Go To Line	Ctrl + G	Jumps to a specified line number.
Jump - Jump Variable Define		Jumps to the position in the Variable Editor where the selected ST program variable is defined (declared).

4-4-3 View Menu

Menu	Shortcut	Function
Toolbars: General, Build, Controller, Display, Data Type, Variable, LD, Mnemonic, ST		Displays/hides the toolbars in the Main Window.
Status Bar		Displays/hides the status bar.
Window - Workspace	Alt + 1	Displays/hides the Workspace Window.
Window - Output	Alt + 2	Displays/hides the Output Window.
Window - Watch	Alt + 3	Displays/hides the Watch Window.
Window - Cross Reference Tool	Alt + 4	Displays/hides the Cross Reference Popup Window.
Window - Library	Alt + 5	Displays/hides the Library Window.
Window - Property	Alt + 6	Displays/hides the Property Window.
Window - Outline	Alt + 7	Displays/hides the Outline Window.
Window - Initialize window position		Returns the window arrangement to its initial status (the status after the NE Programmer was first installed).
To Lower Layer	Shift + F	Jumps from a function block instance to its body program.
To Upper Layer		Jumps from the body program of a function block to the location that called the function block.
Variable Editor - Upper		Changes the Variable Editor's position to the top (default position), effective the next time the Variable Editor starts.
Variable Editor - Right		Changes the Variable Editor's position to the right side, effective the next time the Variable Editor starts.
Variable Editor - Lower		Changes the Variable Editor's position to the bottom, effective the next time the Variable Editor starts.
Variable Editor - Left		Changes the Variable Editor's position to the left side, effective the next time the Variable Editor starts.
Variable Editor - Visible		Displays/hides the Variable Editor.
Monitoring Data Type - Monitor in Hex	Alt + Shift + H	Displays the items registered in the Watch Window in hexadecimal.
Monitoring Data Type - Decimal		Displays the items registered in the Watch Window in decimal.
Monitoring Data Type - Signed Decimal		Displays the items registered in the Watch Window in signed decimal.
Monitoring Data Type - Binary Number		Displays the items registered in the Watch Window in binary.
Monitoring Data Type - Auto		Displays the items registered in the Watch Window to match the data type.
Zoom In	Alt + →	Magnifies the program display.
Zoom Out	Alt + ←	Reduces the program display.
Zoom to Fit	Alt + ↑	Fits the program display to the window width.

Items Added to the View Menu at Ladder Program Editing

Menu	Shortcut	Function
Rung Number and Program Address		Displays/hides the rung number and program address.
Instruction Description		Displays/hides the instruction description.
Grid	G	Displays/hides the grid.

4-4-4 Ladder Menu

The Ladder Menu is displayed only when the Ladder Editor is active.

Menu	Shortcut	Function
Insert -Open Contact	C	Inserts a NO input condition at the specified position.
Insert - Closed Contact	/	Inserts a NC input condition at the specified position.

Menu	Shortcut	Function
Insert - Open Contact OR	W	Inserts an ORed NO condition at the specified position.
Insert - Closed Contact OR	X	Inserts an ORed NC condition at the specified position.
Insert - Coil	O	Inserts an output instruction at the specified position.
Insert - Negative Coil	Q	Inserts a negative output instruction at the specified position.
Insert - Instruction	I	Inserts an instruction at the specified position.
Insert - Function Block	F	Inserts a function block at the specified position.
Insert - Insert Line Comment	Tab	Inserts a line comment at the specified position.
Mode - Select		Switches the editing mode to Select Mode.
Mode - Open Contact		Switches the editing mode to NO Input Condition Mode.
Mode - Closed Contact		Switches the editing mode to NC Input Condition Mode.
Mode - Open Contact OR		Switches the editing mode to Input Condition OR Mode.
Mode - Coil		Switches the editing mode to Output Mode.
Mode - Negated Coil		Switches the editing mode to Negative Output Mode.
Mode - Function Block		Switches the editing mode to Function Block Variable Input Mode.
Mode - Line		Switches the editing mode to Draw Line Mode.
Mode - Erase		Switches the editing mode to Erase Line Mode.
Rung - Select	Ctrl + Enter	Selects a rung.
Rung - Insert Row Above	Ctrl + I	Inserts an open row above the cursor position.
Rung - Insert Row Below	Ctrl + Shift + I	Inserts an open row below the cursor position.
Rung - Delete Row	Ctrl + D	Deletes the selected row.
Rung - Insert Column		Inserts a column at the cursor position.
Rung - Delete Column		Deletes the column at the cursor position.
Draw Line	Ctrl + L	Switches the editing mode to Draw Line Mode, so the cursor becomes the starting point for line drawing.
Erase Line	Ctrl + Shift + L	Switches the editing mode to Erase Line Mode, so the cursor becomes the starting point for line erasing.
Change Variable Usage - Input/ Output/Internal		Changes a variable for a function block to an input variable, output variable, or internal variable.
Immediate Refresh		Changes an instruction between a immediate refresh instructions and a normally refreshed instruction.
Invert (NOT)		Switches between NO and NC input conditions and between outputs and negative outputs.
Transition Sensing - Nothing/Positive/Negative		Sets or releases a transition (differential) condition for an input condition.
Online Edit - Begin	Ctrl + E	Starts online editing.
Online Edit - Cancel	Ctrl + U	Cancels online editing.
Online Edit - Finish	Ctrl + Shift + E	Ends online editing.
Add to Watch		Adds an instruction operand to the Watch Window.

4-4-5 ST Menu

This menu is displayed only when the ST editor is active.

Menu	Shortcut	Function
Add Variable	Ctrl + R	Adds the selected text string to the variable.
Sort - Ascending sort by name	Ctrl + Q	Sorts items in the ST Monitor Window by name in ascending order.
Sort - Descending sort by name	Ctrl + Shift + Q	Sorts items in the ST Monitor Window by name in descending order.
Sort - Ascending sort by value	Ctrl + U	Sorts items in the ST Monitor Window by value in ascending order.
Sort - Descending sort by value	Ctrl + Shift + U	Sorts items in the ST Monitor Window by value in descending order.
Display Monitor Window	Ctrl + Shift + M	Displays/hides the ST Monitor Window.

4-4-6 Mnemonic Menu (Displayed in Mnemonic Editor Only)

Menu	Shortcut	Function
Change Variable Usage - Input/Output/Internal		Changes a variable for a function block to an input variable, output variable, or internal variable.
Immediate Refresh		Changes an instruction between a immediate refresh instructions and a normally refreshed instruction.
Invert (NOT)	Ctrl + R	Switches between NO and NC input conditions and between outputs and negative output.
Differentiate - Nothing/Positive/Negative		Sets or releases a transition (differential) condition for an input condition.
Add to Watch		Adds an instruction operand to the Watch Window.
Import		Imports mnemonic data from a CSV file.
Export		Exports mnemonic data to a CSV file.

4-4-7 Variable Menu

Menu	Shortcut	Function
Add		Adds new variables.
Edit		Deletes existing variables.
Group Input/Output Variables - Group	Ctrl + G	Creates a group of I/O variables for a function block to make them the function block easier to understand.
Group Input/Output Variables - Release Group	Ctrl + Shift + G	Ungroups the I/O variables for a function block.
Group Input/Output Variables - Release Member	Ctrl + Shift + M	Removes the selected members from the group.
Rename	Ctrl + E	Changes the group name.
Change Variable Usage - Input/Output/Internal/External		Changes the variable to an input variable, output variable, internal variable, or external variable
Register Global Variables		Registers selected variables as global variables.
Check Consistent with Extern		Checks whether an external variable and global variable match.
Up	Ctrl + ↑	Shifts the selected member up one rung.
Down	Ctrl + ↓	Shifts the selected member down one rung.
Add to Watch		Adds a variable to the Watch Window.
Import - CSV Format		Imports variables from a CSV-format file.
Export - CSV Format		Exports variables to a CSV-format file.

4-4-8 Data Type Menu

The Data Type Menu is displayed only when the Data Type Editor is active.

Menu	Shortcut	Function
Insert - Struct		Inserts a data structure.
Insert - Element		Inserts an element into a data structure.
Edit		Edits a user-defined data structure.
Move Upward		Moves a user-defined data structure upward.
Move Downward		Moves a user-defined data structure downward.
Check		Checks a data structure for errors.
Import - CSV File		Imports a data structure from a CSV-format file.
Export - CSV File		Exports a data structure to a CSV-format file.

4-4-9 Build Menu

Menu	Shortcut	Function
Compile	Ctrl + F7	Compiles and performs a program check on the active program or function block.
Build	F7	Builds the entire program.
Stop build		Stops building a program.

4-4-10 Controller Menu

Menu	Shortcut	Function
Connect	Ctrl + W	Used to select the communications port and then connect online to the PLC.
Disconnect	Ctrl + Shift + W	Disconnects from the CPU Unit.
Change Controller		Changes the device to which the connection is made (eliminates the need to disconnect).
Change Controller		Changes the controller type.
Auto Upload from Controller	Ctrl + Shift + A	Automatically finds the connected PLC and communications conditions, connects the PLC online, and uploads the program.
Upload from Controller	Ctrl + Shift + T	Used to upload the program, PLC Setup, TCP/IP settings, and I/O tables from the PLC to the computer.
Download to Controller	Ctrl + T	Used to download the program, PLC Setup, TCP/IP settings, and I/O tables from the computer to the PLC.
Compare with Controller		Compares the programs on the computer and in the PLC.
System Configuration		Used to set up the PLC.
Operating Mode - Program/Monitor/Run	Ctrl + 1/Ctrl + 3/ Ctrl + 4	Changes the PLC's operating mode.
Monitor	Ctrl + M	Starts the monitor.
Backup value of variables		Saves the present variable values to a CSV file.
Restore value of variables		Restores saved variable values from a CSV file.
I/O Table - Create		Creates the real I/O tables.
I/O Table - Delete		Deletes the registered I/O tables. The CPU Unit will operate with the real I/O tables.
I/O Table - Compare		Compares the real I/O tables and registered I/O tables.
Clear Error	F4	Clears an error.
Clear Memory		Clears the CPU Unit memory, including the user program, parameter area, and I/O memory.
Restart Service		Starts the SMTP server and SNMP server.
Set - On		Sets (turns ON) a bit.
Set - Off		Resets (turns OFF) a bit.
Set - Force On	Ctrl + J	Force-sets (forces ON) a bit.
Set - Force Off	Ctrl + K	Force-resets (forces OFF) a bit.
Set - Force Cancel		Clears a force-set or force-reset bit.
Set - Cancel All Forces		Clears all force-set and force-reset bits.
Set - Value		Changes the PV of the selected variable or physical address.
Set - Timer/Counter Setting Value		Changes the PV of the timer or counter.
Differential Monitor		Executes differential monitoring.
Error Log		Displays the PLC error log.
Change Log - Enable Mode		Enables the change log.
Change Log - Disable Mode		Disables the change log.
Change Log - Change Log List		Reads the change log list.
Cycle Time		Displays the cycle time.
Data Trace		Used to execute a data trace.

Menu	Shortcut	Function
Variable Reference Report		Displays a list showing the usage of variables.
Set Clock		Used to set the clock in the CPU Unit.
Release Access Rights		Forcibly releases the access rights held by the PLC.

4-4-11 Library Menu

Menu	Shortcut	Function
POU - Register to Library		Registers a POU in the library.
POU - Add to Project		Adds a POU registered in the library to a project.
Rung - Register to Library		Registers a rung group in the library.
Rung - Insert to Program		Adds a rung group registered in the library to a program.
Map Folder		Changes the library folder allocated in the computer.
Disconnect Folder		Clears the library folder allocation.
Create Folder		Creates a folder.
Property		Displays information on the library.
Update		Redisplays the library folder tree.
Edit		Used to edit library properties.
Delete		Deletes a library object or folder.
Option		Sets the operation used for the library function.

4-4-12 Tool Menu

Menu	Shortcut	Function
Select Interface - CJ2 USB/Serial Port NE1S Serial Port DeviceNet I/F Ethernet I/F RSLinx I/F		Used to select the communications interface. • CJ2 USB/Serial Port: USB port or RS-232 port • NE1S Serial Port: USB port • DeviceNet Interface: DeviceNet • Ethernet Interface: Ethernet port • RSLinx interface: RSLinx port
Key Customize		Used to change shortcut key allocations.
Option		Sets various options for NE Programmer displays and operations.

4-4-13 Window Menu

Menu	Shortcut	Function
Next Docked	Alt + 0	Switches to the next window as the active window in the following order: Workspace, Outline, Output, Watch, Property, Cross Reference, Library, MDI Window.
Previous Docked	Alt + Shift + 0	Switches to the next window as the active window in the following order: MDI Window, Library, Cross Reference, Property, Watch, Output, Outline, Workspace.
Toggle Split Window	F6	Switches between the Variable Editor and Program Editor.
Toggle POU/Task Window	Alt + F6	Switches between the program window and the task window.
Close All		Closes all windows.
Cascade		Cascades all open windows.
Tile Horizontally		Tiles all open windows vertically.
Tile Vertically		Tiles all open windows horizontally.
Arrange Icons		Arranges the icons for minimized windows.

4-4-14 Help Menu

Menu	Shortcut	Function
Topics		Displays the NE Programmer help function.
Instruction Reference		Displays the instruction reference.
About		Displays information on the NE Programmer version.

4-5 Shortcut Keys

The following tables list the NE Programmer's shortcut key operations.

4-5-1 Window/View Operations

Shortcut keys	Menu name	Function
Ctrl + PageUp/ PageDown	---	Switches the Tab Page.
Shift + F10	---	Displays the popup menu at the cursor position (same as right-clicking).
Ctrl + F4	---	Closes the program window.
F6	Window - Toggle Split Window	Switches between the Variable Editor and Program Editor.
Alt + F6	Window - Toggle POU/Task Window	Switches between the program window and the task window.
Ctrl + Tab	Window - *(open window name)*	Switches to an open program window.
Alt + 0	Window - Next Docked	Switches to the next window as the active window in the following order: Workspace, Outline, Output, Watch, Property, Cross Reference, Library, MDI Window.
Alt + Shift + 0	Window - Previous Docked	Switches to the next window as the active window in the following order: MDI Window, Library, Cross Reference, Property, Watch, Output, Outline, Workspace.
Alt + 1	View - Window - Workspace	Displays/hides the Workspace Window.
Alt + 2	View - Window - Output	Displays/hides the Output Window.
Alt + 3	View - Window - Watch	Displays/hides the Watch Window.
Alt + 4	View - Window - Cross Reference Tool	Displays/hides the Cross Reference Popup Window.
Alt + 5	View - Window - Library	Displays/hides the Library Window.
Alt + 6	View - Window - Property	Displays/hides the Property Window.
Alt + 7	View - Window - Outline	Displays/hides the Outline Window.
Shift + F	View - To Lower Layer	Jumps from a function block instance to its body program.
Alt + →	View - Zoom In	Magnifies the program display.
Alt + ←	View - Zoom Out	Reduces the program display.
Alt + ↑	View - Zoom to Fit	Fits the program display to the window width.
G	View - Grid	Displays/hides the grid.
Ctrl + Shift M	ST - Display Monitoring Window	Displays/hides the ST Monitor Window.

4-5-2 File Operations

Shortcut keys	Menu name	Function
Ctrl + N	File - New - Project	Creates a new project.
Ctrl + O	File - Open	Opens an existing project.
Ctrl + S	File - Save	Saves the project being edited (overwrites current project data).
Ctrl + Shift + S	File - Save Changes to project	Updates the project for the program or function block that is being edited.

Shortcut keys	Menu name	Function
Alt + G	File - Configuration - Edit Global Variables	Edits global variables.
Ctrl + Shift + R	File - Configuration - Replace physical addresses in programs	Replaces physical addresses in the program with global variables specified by that physical address.
Ctrl + P	File - Print	Prints the specified items.

4-5-3 Edit Operations

Shortcut keys	Menu name	Function
Ctrl + Z	Edit - Undo	Undoes the previously performed operation.
Ctrl + Y	Edit - Redo	Redoes the operation that was undone with *Undo*.
Ctrl + X	Edit - Cut	Cuts the specified range of data.
Ctrl + C	Edit - Copy	Copies the specified range of data.
Ctrl + V	Edit - Paste	Pastes the contents of the clipboard.
Delete	Edit - Delete	Deletes the specified range of data.
F2	Edit - Rename	Used to change the name of a project, function block, program, or configuration.
Ctrl + A	Edit - Select All	Selects all data in the selected window.
Ctrl + F	Edit - Find	Searches for a text string.
F3	Edit - Find Next	Finds the next instance of a text string.
Shift + F3	Edit - Find Prev	Finds the previous instance of a text string.
Ctrl + H	Edit - Replace	Replaces a text string.
Ctrl + B	Edit - Bookmark	Edits a bookmark.
Ctrl + Shift + F	Edit - Find in Programs	Searches for the specified item.
Ctrl + G	Edit - Jump - Jump Edit - Jump - Go To Link Edit - Jump - Jump	• In Ladder Editor, jumps to a specified step number or rung number. • In ST Editor, adds the selected text as a variable. • In Mnemonic Editor, inverts the specified instruction.
Alt + Shift + A	Edit - Jump - Bit Address Reference	In Ladder Editor, searches for corresponding input conditions or outputs associated with the selected operand. If an input condition is specified, a search is made backward for an output with the same bit address. If an output is specified, a search is made foreword for an input condition with the same bit address.
Alt + Shift + N	Edit - Jump - Next Operand Reference	In Ladder Editor, jumps to the next operand with the same address/variable.
Alt + Shift + I	Edit - Jump - Next Input	In Ladder Editor, jumps to the next input with the same address/variable.
Alt + Shift + O	Edit - Jump - Next Output	In Ladder Editor, jumps to the next output with the same address/variable.
Alt + Shift + B	Edit - Jump - Previous Jump Point	In Ladder Editor, returns to the position before the jump.
F8	Edit - Edit Variable	Used to edit the selected instruction operand's variable.
Ctrl + Shift + C	Edit - Copy Operand	Used to copy the operand only.

4-5-4 Offline/Programming Operations

Shortcut keys	Menu name	Function
C	LD - Insert - Open Contact	Inserts a NO input condition at the specified position.
/	LD - Insert - Closed Contact	Inserts a NC input condition at the specified position.
W	LD - Insert - Open Contact OR	Inserts an ORed NO condition at the specified position.
X	LD - insert - Closed Contact OR	Inserts an ORed NC condition at the specified position.
O	LD - Insert - Coil	Inserts an output instruction at the specified position.
Q	LD - Insert - Negative Coil	Inserts a negative output instruction at the specified position.
I	LD - Insert - Instruction	Inserts an instruction at the specified position.
F	LD - Insert - Function Block	Inserts a function block at the specified position.
Tab	LD - Insert - Insert Line Comment	Inserts a line comment at the specified position.
Ctrl + Enter	LD - Rung - Select	Selects a rung when using the Ladder Editor.
Ctrl + R	ST - Add Variable Mnemonic - Invert	• In ST Editor, adds the selected text as a variable. • In Mnemonic Editor, inverts the specified instruction.
Ctrl + I	Ladder - Rung - Insert Row Above	Inserts an open row above the cursor position.
Ctrl + Shift + I	Ladder - Rung - Insert Row Below	Inserts an open row below the cursor position.
Ctrl + D	Ladder - Rung - Delete Row	Deletes the selected row.
Ctrl + L	Ladder - Draw Line	Switches the editing mode to Draw Line Mode.
Ctrl + Shift + L	Ladder - Erase Line	Switches the editing mode to Erase Line Mode.
Ctrl + →/ ←/↑/↓	---	In a ladder program, draws a line with the cursor position as a starting point.
Ctrl + Shift + →/ ←/ ↑/↓	---	In a ladder program, deletes a line with the cursor position as a starting point.
/	Ladder - Invert (NOT)	Inverts the specified instruction.
Ctrl + E	Ladder - Online Edit - Begin	Starts online editing.
Ctrl + U	Ladder - Online Edit - Cancel	Cancels online editing.
Ctrl + Shift + E	Ladder - Online Edit - Finish	Ends online editing.

4-5-5 Variable Operations

Shortcut keys	Menu name	Function
Ctrl + G	Variable - Group Input/Output Variables - Group	In Variable Editor or Global Variable Editor, groups the selected I/O variables.
Ctrl + Shift + G	Variable - Group Input/Output Variables - Release Group	In Variable Editor or Global Variable Editor, ungroups all of the I/O variables in the group.
Ctrl + Shift + M	Variable - Group Input/Output Variables - Release Member	In Variable Editor or Global Variable Editor, removes the selected member from the group.
Ctrl + E	Variable - Rename	Changes the group name.
Ctrl + ↑	Variable - Up	Shifts the function block's selected input variable or output variable up one position in the Variable Editor.
Ctrl + ↓	Variable - Down	Shifts the function block's selected input variable or output variable down one position in the Variable Editor.

4-5-6 Build Operations

Shortcut keys	Menu name	Function
Ctrl + F7	Build - Compile	Compiles the active program or function block.
F7	Build - Build	Builds the project.

4-5-7 Online/Controller Operations

Shortcut keys	Menu name	Function
Ctrl + W	Controller - Connect	Connects to the Controller.
Ctrl + Shift + W	Controller - Disconnect	Disconnects from the Controller.
Ctrl + Shift + A	Controller - Auto upload from Controller	Automatically uploads from the connected Controller.
Ctrl + Shift + T	Controller - Upload from Controller	Uploads from a Controller.
Ctrl + T	Controller - Download to Controller	Downloads to a Controller.
Ctrl + 1	Controller - Operating Mode - Program	Switches the Controller to PROGRAM mode.
Ctrl + 3	Controller - Operating Mode - Monitor	Switches the Controller to MONITOR mode.
Ctrl + 4	Controller - Operating Mode - Run	Switches the Controller to RUN mode.
Ctrl + M	Controller - Monitor	Starts monitoring.
F4	Controller - Clear Error	Clears errors.
Ctrl + J	Controller - Set - Force On	Force-sets (forces ON) a bit.
Ctrl + K	Controller - Set - Force Off	Force-resets (forces OFF) a bit.

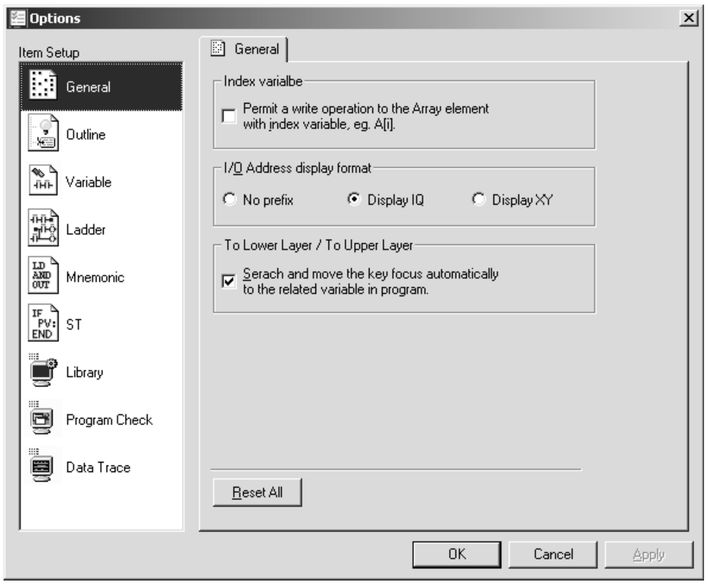
4-6 Option Settings

This menu contains various settings for NE Programmer displays and operations.

Select ***Tool - Option***. The following Integrated Options Dialog Box will be displayed.

Item	Description
General	Sets general displays and operation for the NE Programmer.
Outline	Sets display items in the Outline Window and timing linked with the ladder editor.
Variable	Specifies which variable properties will be displayed.
Ladder	Specifies whether variable names, physical addresses, and comments will be displayed/hidden in the Ladder Editor, and how many lines will be displayed. Displays/hides grid lines, sets fonts, cell widths, and colors for items in the Ladder Editor. Specifies whether instance names will be generated automatically.
Mnemonic	Specifies whether values, comments, addresses, and data types will be displayed/hidden in the Mnemonic Editor.
ST	Sets the font properties and colors of items in the ST Editor.
Library	Specifies whether to edit local variables when rung elements are inserted. Also specifies whether prefixes are added to global variables when using the library's POUs.
Program Check	Sets the program check level.
Data Trace	Sets the colors of items in the data trace display.

4-6-1 General Window



Index varialbe
[ ] Permit a write operation to the Array element with index variable, eg. A[i].

I/O Address display format
○ No prefix [✓ Display IQ] ○ Display XY

To Lower Layer / To Upper Layer
[✓] Serach and move the key focus automatically to the related variable in program.

Index Variable

Select this option to permit writing (i.e., PV changes, force-setting/resetting, and setting/resetting) array variable elements with a variable index, e.g., A[i].

Note An array variable's index value is based on the index value the last time that the value was monitored, so a write operation may operate on a different array element if the variable's index value was changed since the last write operation.

I/O Address Display

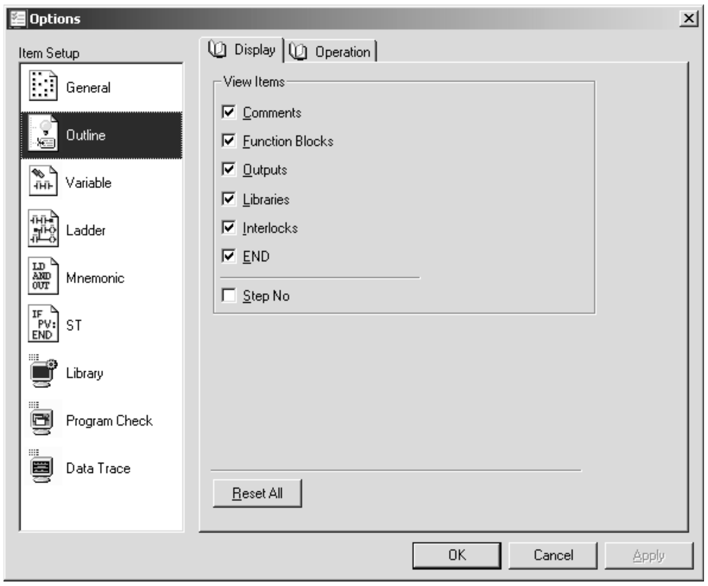
If I/O Units are allocated in I/O tables, the corresponding I/O bits will be displayed in the ladder window with I/O addresses.
Set whether to display the I/O address and the display format.
No display: The I/O address will not be displayed.
Display IQ: The I/O address will be displayed with "I" for inputs and "Q" for outputs.
Display XY: The I/O address will be displayed with "X" for inputs and "Y" for outputs.

To Lower Layer/To Upper Layer

Select this option to automatically search for function block instance variables (default: enabled). If this option is disabled, the automatic variable search function will not operate.

4-6-2 Outline

Display Tab Page



[✓] Comments
[✓] Function Blocks
[✓] Outputs
[✓] Libraries
[✓] Interlocks
[✓] END
[ ] Step No

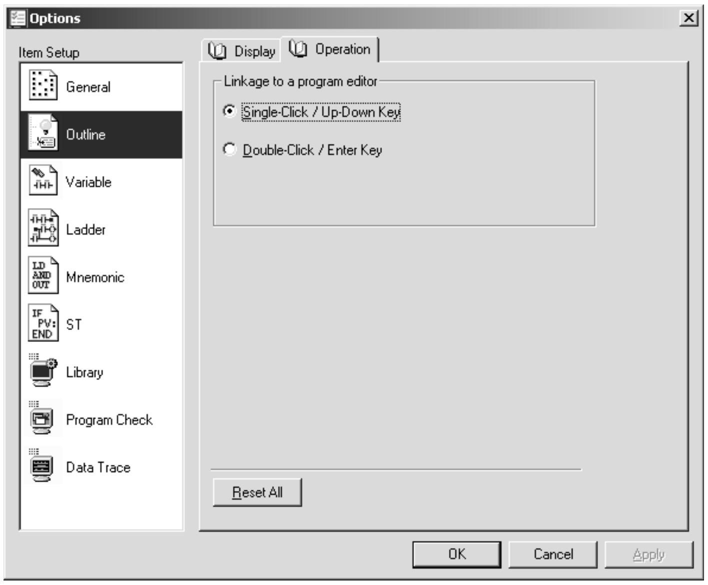
View Items

Select the items to display in the Outline Window: Comments, Function Blocks, Outputs, Libraries, Interlocks, END, Step No.

Reset All

Use this button to reset all view item settings. (The settings will be as shown in the above figure.)

Operation Tab Page



[✓ Single-Click / Up-Down Key]
Double-Click / Enter Key

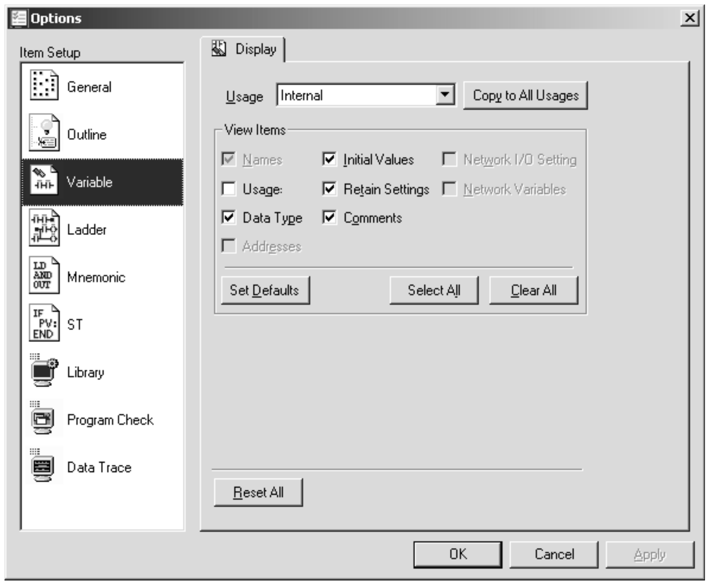
Linkage to a Program Editor

Select the timing for linking to the Program Editor.

Reset All

Use this button to reset all settings. (The settings will be as shown in the above figure.)

4-6-3 Variable Window



Usage: Internal

[✓] Names
[ ] Usage
[✓] Data Type
[ ] Addresses
[✓] Initial Values
[✓] Retain Settings
[✓] Comments
[ ] Network I/O Setting
[ ] Network Variables

Usage

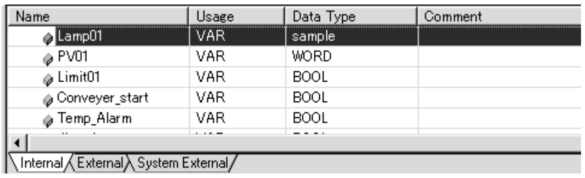
Select the variable type: *Internal*, *Global*, *System*, *Input*, *Output*, *External*, or *System External*.

View Items

Select the variable properties to be displayed in the Variable Editor Window. (Different properties can be set for each variable type.)

> *Names, Usage*, *Data Type*, *Addresses*, *Initial Values*, *Retain Settings*, *Comments, Network I/O Setting, Network Variables*

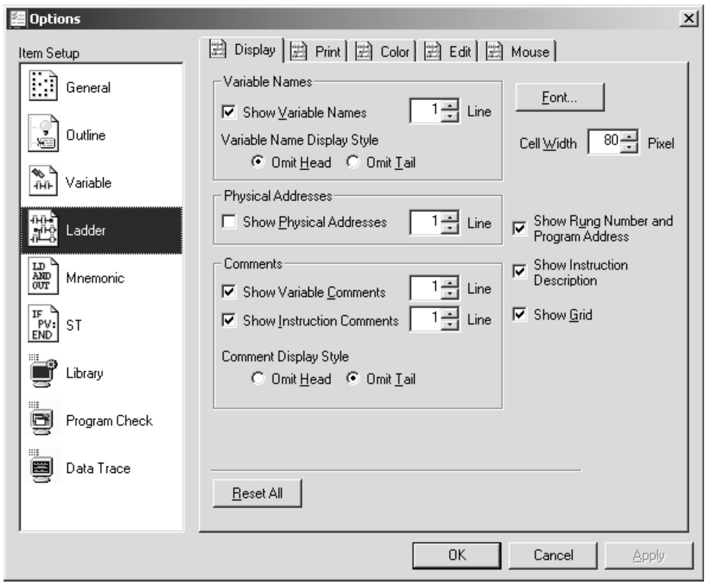
In the following example, the *Usages*, *Data Types* and *Comments* Options have been selected.



Name	Usage	Data Type	Comment
Lamp01	VAR	sample	
PV01	VAR	WORD	
Limit01	VAR	BOOL	
Conveyer_start	VAR	BOOL	
Temp_Alarm	VAR	BOOL	

4-6-4 Ladder Window

Display Tab Page



Variable Names
[✓] Show Variable Names 1 Line
Variable Name Display Style
[✓ Omit Head] Omit Tail

Physical Addresses
[ ] Show Physical Addresses 1 Line

Comments
[✓] Show Variable Comments 1 Line
[✓] Show Instruction Comments 1 Line
Comment Display Style
Omit Head [✓ Omit Tail]

Cell Width 80 Pixel
[✓] Show Rung Number and Program Address
[✓] Show Instruction Description
[✓] Show Grid

Variables

Show Variable Name:
Select this option to display the variable names. The style of the variable name display can be selected.

Variable Name Display Style:

- Omit Head
  If all of the variable name cannot be displayed, select this option to display only the last part of the variable name.
- Omit Tail
  If all of the variable name cannot be displayed, select this option to display only the first part of the variable name.

Physical Addresses

Show Physical Addresses:
Select this option to display the physical addresses in addition to variable names (default: OFF). A physical address will be displayed, however, if it is directly specified to the bit. The number of lines of physical addresses can be set.

Comments

Show Variable Comments:
Select this option to display the variable comments. The number of lines of variable comments can be set.

Show Instruction Comments:
Select this option to display the instruction comments. The number of lines of instruction comments can be set.

Comment Display Style:

- Omit Head
  If all of the comment cannot be displayed, select this option to display only the last part of the comment.
- Omit Tail
  If all of the comment cannot be displayed, select this option to display only the first part of the comment.

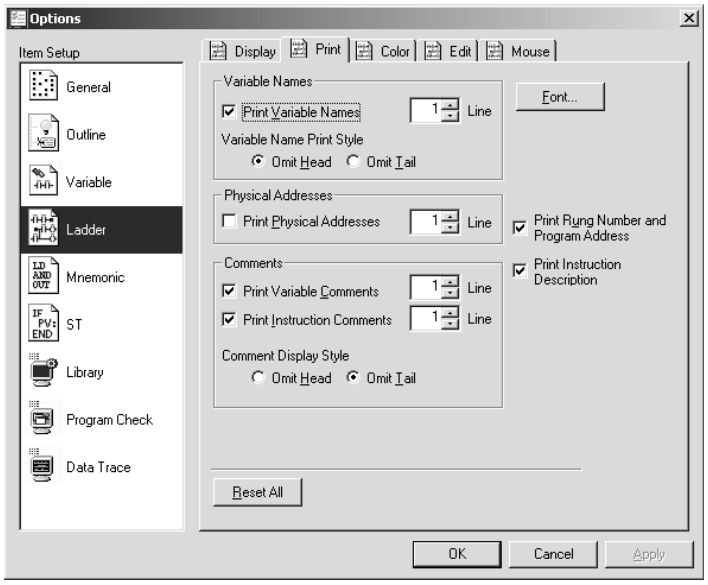
Font

Click the **Font** Button to set the font used in the Ladder Editor.

Cell Width: Sets the width of cells in the Ladder Editor in pixels.
Show Rung Number and Program Address: Clear this option to increase the ladder display width by not displaying the rung number or step number.
Show Instruction Description: Clear this option to not display descriptions of special instructions.

Show Grid: Clear this option to hide the Ladder Editor's grid.

Print Tab Page



Variable Names
[✓] Print Variable Names 1 Line
Variable Name Print Style
[● Omit Head] ○ Omit Tail

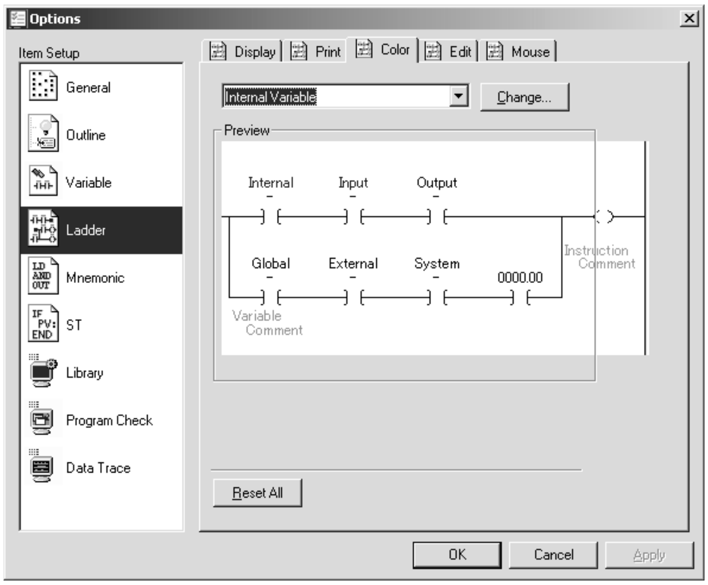
Physical Addresses
[ ] Print Physical Addresses 1 Line

Comments
[✓] Print Variable Comments 1 Line
[✓] Print Instruction Comments 1 Line
Comment Display Style
○ Omit Head [● Omit Tail]

[✓] Print Rung Number and Program Address
[✓] Print Instruction Description

Contains the same settings as the Display Tab Page.

Color Tab Page



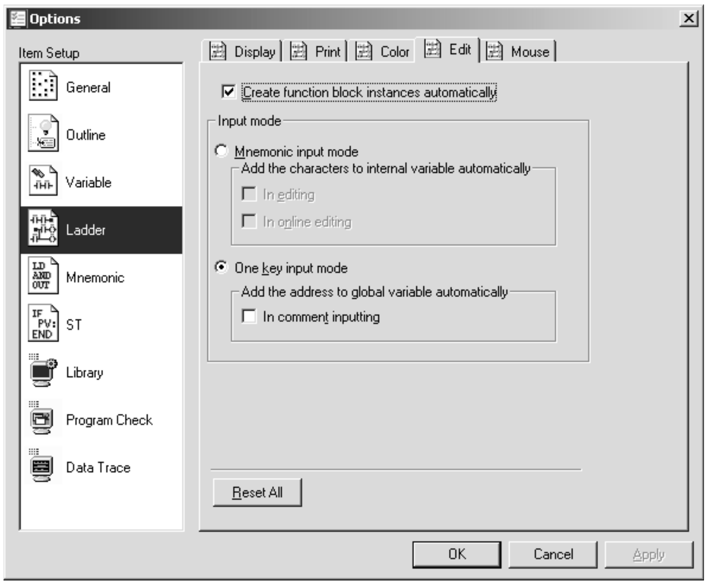
The colors can be selected for the following Ladder Editor items. The *Preview* Area displays an example of the selected colors.

Internal Variable, *Input Variable*, *Output Variable*, *Global Variable*, *External Variable*, *System Variable*, *Variable Comment*, etc.

Change

Click the **Change** Button to set the color. The Set Color Dialog Box will be displayed. Set the colors for the selected Ladder Editor items.

Edit Tab Page



[✓] Create function block instances automatically

Input mode

( ) Mnemonic input mode

Add the characters to internal variable automatically

[ ] In editing

[ ] In online editing

(●) One key input mode

Add the address to global variable automatically

[ ] In comment inputting

Create function block instances automatically

Select this option to automatically generate the instance name when a function block is pasted in the program. (The instance name is the function block name with a consecutive number attached.)

Input Mode

Select the input mode to be used when editing bits and special instructions.
Mnemonic Input Mode: Mode for directly inputting mnemonics in the ladder display. With this mode, input characters are automatically added as internal variables. The timing for automatic addition can be set to either of the following two options.

In editing:
The input variable name is automatically added as an internal variable whenever editing is performed.

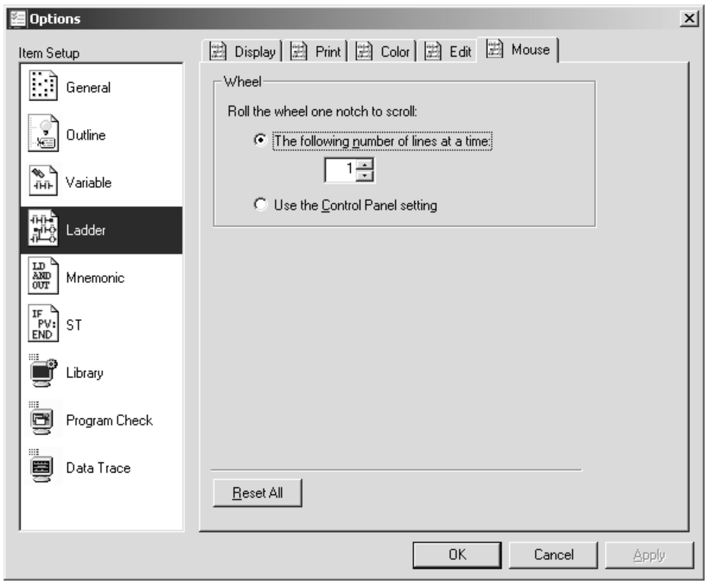
In online editing:
The input variable name is automatically added as an internal variable when editing is performed online.

One-key Input Mode: Mode for starting input by using shortcut keys assigned to bits and instructions. Defaults include the following: C Key: LD Instructions, O Key: OUT instructions, I Key: Special Instructions. Input addresses are automatically added as global variables. (This is the default input mode.)

Note The shortcut keys can be changed as desired in the Key Customize Window by selecting ***Tool - Key Customize***.

In comment inputting: Support Option for Converting to Physical Addresses to use support for converting to physical addresses (default: disabled). When a physical address is directly input in the ladder editor, this function is used to automatically generate the AT-specified global variable of that address and the corresponding external variable. It is not possible to use this function, however, if the configuration is not already created in the project.

Mouse Tab Page



Wheel

Roll the wheel one notch to scroll:

(●) The following number of lines at a time: 1

( ) Use the Control Panel setting

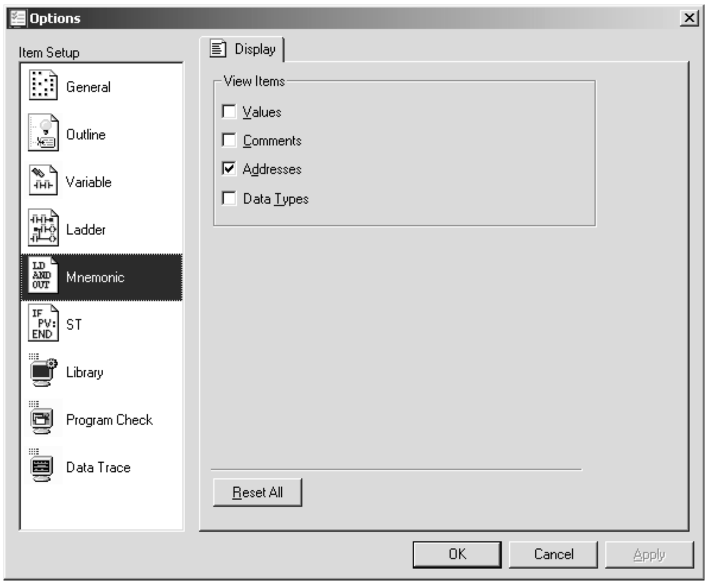
Wheel

Set the amount to scroll for each notch on the mouse wheel.

Roll the wheel one notch to scroll: Input the number of lines to scroll for each notch (default: 1).

Use the Control Panel setting: Use the setting for amount of scrolling in the Control Panel.

4-6-5 Mnemonic Window



View Items

☐ Values

☐ Comments

☑ Addresses

☐ Data Types

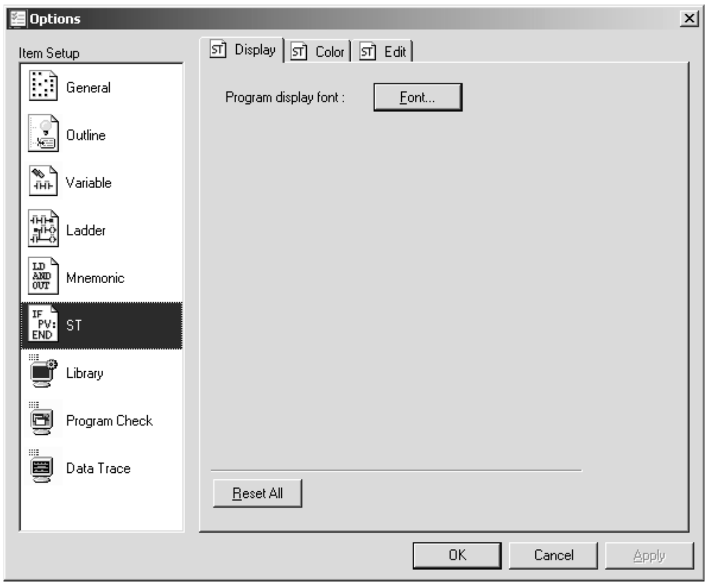
Display Tab Page

Select the following items to be displayed in the Mnemonic Editor.

Values, *Comments*, *Addresses*, and *Data Type*

4-6-6 ST Window

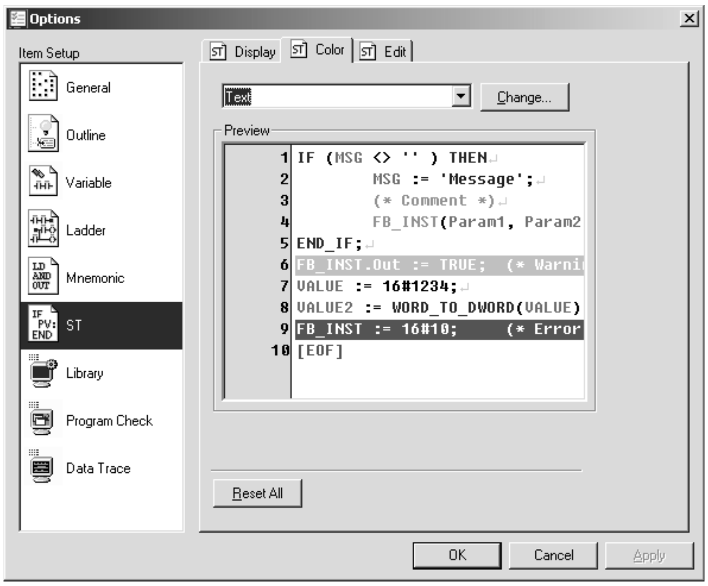
Display Tab Page



Font

Click the **Font** Button to set the font used in the ST Editor.

Color Tab Page



```
IF (MSG <> '' ) THEN
        MSG := 'Message';
        (* Comment *)
        FB_INST(Param1, Param2
END_IF;
FB_INST.Out := TRUE;  (* Warni
VALUE := 16#1234;
VALUE2 := WORD_TO_DWORD(VALUE)
FB_INST := 16#10;     (* Error
[EOF]
```

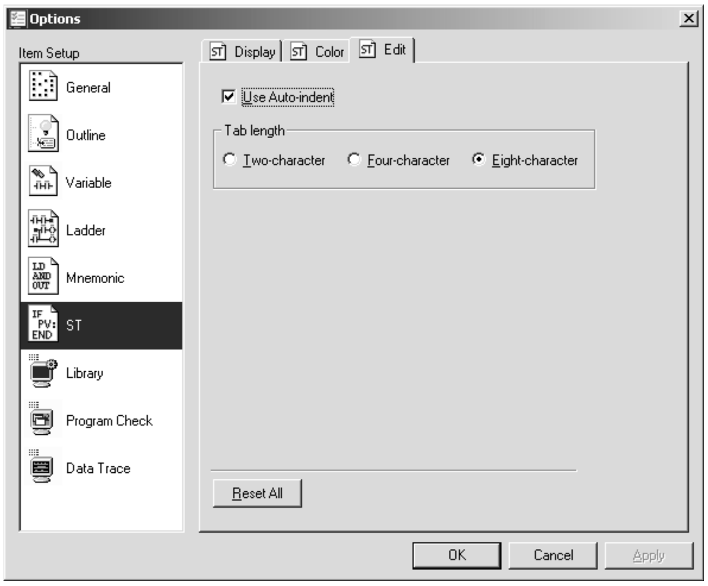
The colors can be selected for the following ST Editor items. The *Preview* Area displays an example of the selected colors.

Text, Background, Error Line, Warning Line, Error Line Text, Warning Line Text, Comment, Keyword, Fixed String, Line Feed, End of File, Function, Function Block Instance, and *Variable Name*

Change

Click the **Change** Button to set the color. The Set Color Dialog Box will be displayed. Set the colors for the selected ST Editor items.

Edit Tab Page



[✓] Use Auto-indent

Tab length

Two-character Four-character [✓ Eight-character]

Perform settings for editing operations using the ST Editor.

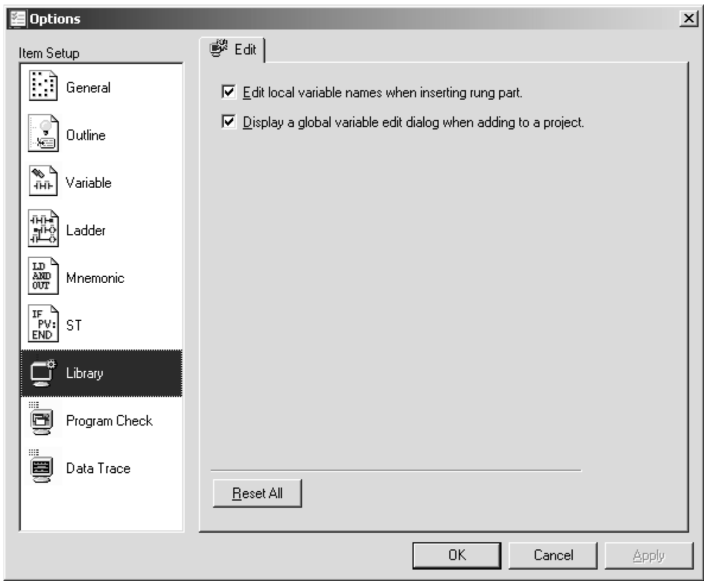
Use Auto-indent

Select this option to automatically indent when performing a line feed while editing programming using the ST Editor (default: enabled).

Tab Length

Set the tab length to be used when the Tab Key is used while editing programming using the ST Editor. Select one of the following options: *Two-character, Four-character,* or *Eight-character* (default: *Eight-character*).

4-6-7 Library Window



[✓] Edit local variable names when inserting rung part.

[✓] Display a global variable edit dialog when adding to a project.

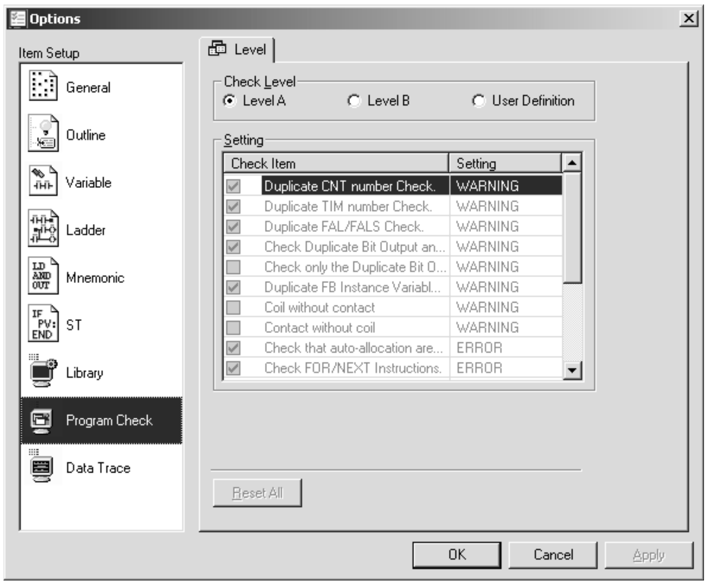
Edit local variable names when inserting rung part

When this option is selected, the Edit Local Variables Dialog Box will be displayed when one of the library's rung elements is dragged and dropped into the program.

Display a global variable edit dialog box when adding to a project

When this option is selected, the Add Prefix/Suffix Dialog Box will be displayed and a prefix and suffix will be added to all global variables in a POU when a library POU is added to a project.

4-6-8 Program Check Window



Check Level
(● Level A) ○ Level B ○ User Definition

Setting

	Check Item	Setting
✓	Duplicate CNT number Check.	WARNING
✓	Duplicate TIM number Check.	WARNING
✓	Duplicate FAL/FALS Check.	WARNING
✓	Check Duplicate Bit Output an...	WARNING
☐	Check only the Duplicate Bit O...	WARNING
✓	Duplicate FB Instance Variabl...	WARNING
☐	Coil without contact	WARNING
☐	Contact without coil	WARNING
✓	Check that auto-allocation are...	ERROR
✓	Check FOR/NEXT Instructions.	ERROR

Check Level

The program check level can be set to *Level A* (default setting), *Level B*, or *User Definition*. If level A is selected, a stricter program check will be performed. (Refer to the following table.)

The check items can be customized by selecting *User Definition*. (Refer to the following table for check items.)

- Select whether to perform the check by selecting the check item in the list.
- Select either *Error* or *Warning* as the error notification.
- If the Reset All Button is pressed, the levels will be set to level A.

Note Check items cannot be selected and "error" or "warning" cannot be specified if level A or level B is selected.

Level A

Check item	Normal POUs		Function block POUs	
	Compiling	Building	Compiling	Building
Duplicate CNT number Check	Yes	Yes	Yes	Yes
Duplicate TIM number Check	Yes	Yes	Yes	Yes
Duplicate FAL/FALS Check	Yes	Yes	Yes	Yes
Check Duplicate Bit Output and Channel Output	Yes	Yes	Yes	Yes
Check only the Duplicate Bit Output	No	No	No	No
Duplicate FB Instance Variable Check	Yes	Yes	Yes	Yes
Coil without contact	No	No	No	No
Contact without coil	No	No	No	No
Check that auto-allocation area is not used	Yes	Yes	Yes	Yes
Check FOR/NEXT Instructions	Yes	Yes	Yes	Yes

Check item	Normal POUs		Function block POUs	
	Compiling	**Building**	**Compiling**	**Building**
Check the following of END Instructions	Yes	Yes	No	No
Check IL-ILC Instructions	Yes	Yes	Yes	Yes
Check JMP/JME Instructions	Yes	Yes	Yes	Yes
Check the Auto-variables	Yes	Yes	Yes	Yes
Check operand size and variable type	Yes	Yes	Yes	Yes
Check operand size and array length when the element number is 0	No	No	No	No
Check the unused FB instance variables	Yes	Yes	Yes	Yes

Level B

Check item	Normal POUs		Function block POUs	
	Compiling	**Building**	**Compiling**	**Building**
Duplicate CNT number Check	No	No	No	No
Duplicate TIM number Check	No	No	No	No
Duplicate FAL/FALS Check	No	No	No	No
Check Duplicate Bit Output and Channel Output	No	No	No	No
Check only the Duplicate Bit Output	No	No	No	No
Duplicate FB Instance Variable Check	No	No	No	No
Coil without contact	No	No	No	No
Contact without coil	No	No	No	No
Check that auto-allocation area is not used	Yes	Yes	Yes	Yes
Check FOR/NEXT Instructions	Yes	Yes	Yes	Yes
Check the following of END Instructions	Yes	Yes	No	No
Check IL/ILC Instructions	Yes	Yes	Yes	Yes
Check JMP/JME Instructions	Yes	Yes	Yes	Yes
Check the Auto-variables	Yes	Yes	Yes	Yes
Check operand size and variable type	Yes	Yes	Yes	Yes
Check operand size and array length when the element number is 0	No	No	No	No
Check the unused FB instance variables	Yes	Yes	Yes	Yes

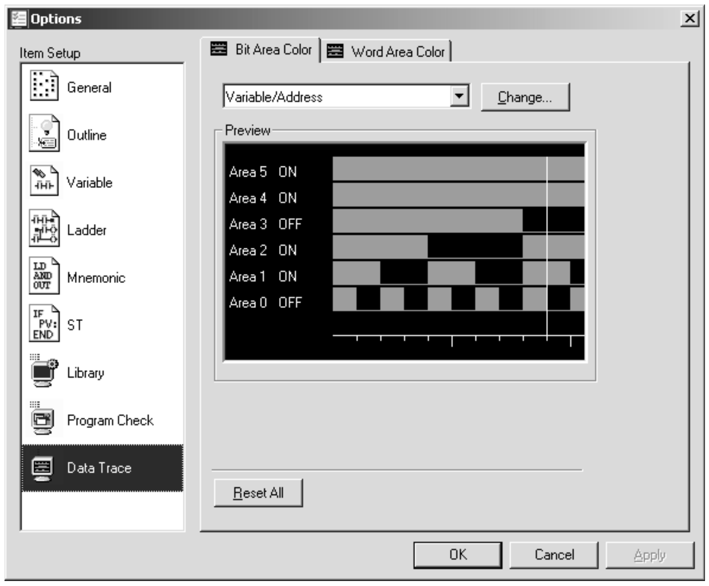
User Definition

Execution/non-execution of check items and error notification settings (either ERROR or WARNING) can be configured.

For details, refer to *Tool - Option, Program Check*.

4-6-9 Data Trace Window

Bit Area Colors Tab Page



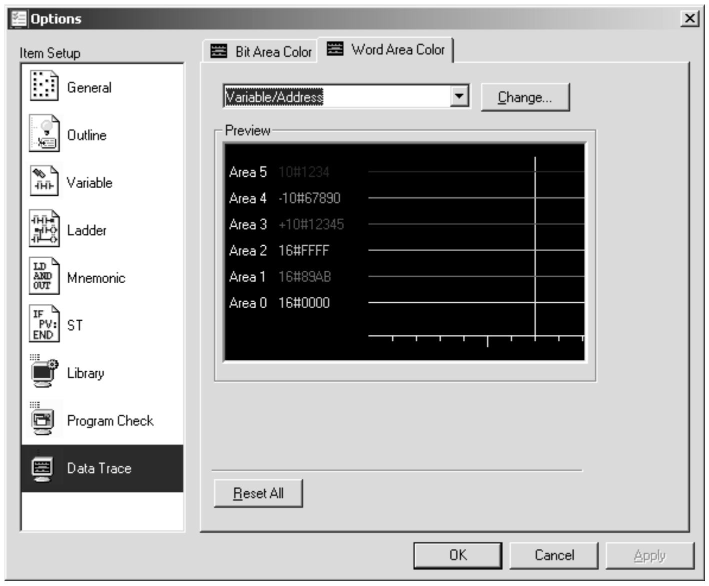
The colors can be selected for the following items in the data trace bit area. The *Preview* Area displays an example of the selected colors.

Variable/Address, *Value*, *Background*, *Grid Line*, *Graph*

Change

Click the **Change** Button to set the color. The Set Color Dialog Box will be displayed. Set the colors for the selected data trace bit area items.

Word Area Colors Tab Page



The colors can be selected for the following items in the data trace word area. The *Preview* Area displays an example of the selected colors.

Variable/Address, *Area 1*, *Area 2*, *Area 3*, *Area 4*, *Area 5*, *Area 6*, *Background*, *Grid Line*

Change

Click the **Change** Button to set the color. The Set Color Dialog Box will be displayed. Set the colors for the selected data trace word area items.

SECTION 5
Programming

This section provides details on programming.

5-1 Overview

5-1-1 Basic Flow of Programming

Reference

1. Create a project. ***File - New - Project***

↓ *5-2-1*

2. Create logical POUs (programs or function blocks). ***File - New - POU***

↓ *5-2-2*

3. Programming in ladder diagrams or standard text and registering/editing variables with local variable editor.

↓ *5-3*

4. Saving edited data in project. ***File - Save Changes to project***

↓ *5-2-3*

5. Creating function blocks and pasting them into programs. Creating function blocks and then dragging and dropping them into programs.

↓ *5-4*

See note.

6. Creating a configuration. ***File - New - Configuration***

↓ *5-6-1*

7. Editing global variables.

↓ *5-6-2*

8. Pasting programs into tasks. Dragging and dropping programs into task folders.

↓ *5-6-3*

9. Saving the project. ***File - Save As***

5-2-4

Note The configuration can be created and the global variables can be edited before the logical POUs (programs and function blocks) are created.

5-2 Creating Projects and Logical POUs

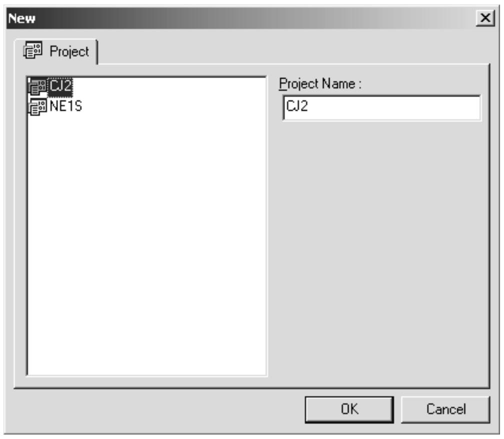
Before actual programming, a project and logical POUs must be created.

5-2-1 Creating Projects

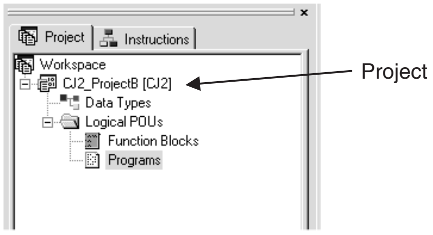
Use the following procedure to create a project.

1,2,3...

1. Right-click the Workspace in the Project Window and select ***Create Project***. Alternately, press the **Ctrl+N** Keys or select ***File - New - Project***. The following New Dialog Box will be displayed.



2. Select either a CJ2 CPU Unit or an NE1S-series CPU Unit. The restrictions on programs that can be created and instructions that can be used depend on the series of the CPU Unit. Therefore, strict restrictions apply to changing the series of the CPU Unit after the project has been created. It is not possible to convert from a project for a CJ2 CPU Unit to a project for an NE1S-series CPU Unit.
3. Inputting the Project Name
   If the name *NE1S* is acceptable, click the **OK** Button. To change the project, input the project name and then click the **OK** Button.
4. The following will be displayed in the Project Window.



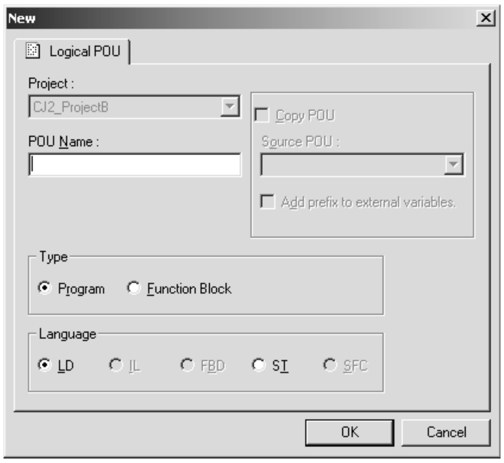
This completes creating a project.

5-2-2 Creating Logical POUs (Programs or Function Blocks)

Use the following procedure to create logical POUs (programs or function blocks).

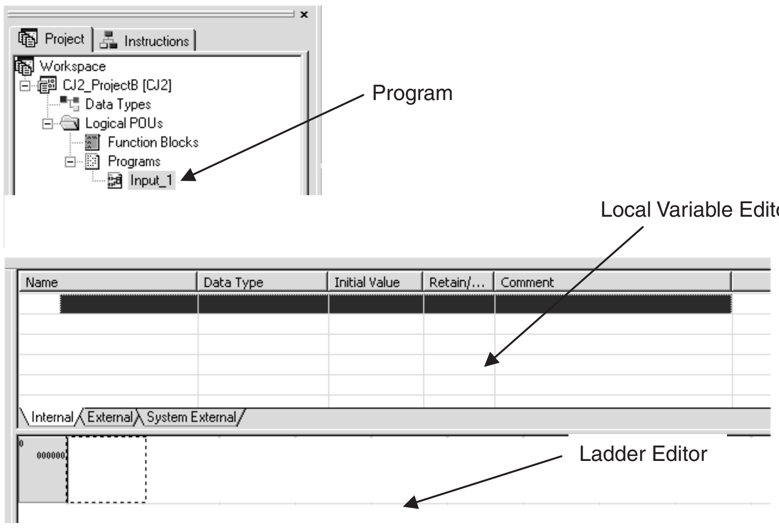
1,2,3...

1. Right-click the ***Logical POUs*** in the Project Window and select ***Create POU***. Alternately, select ***File - New - POU***.
   The following New Dialog Box will be displayed.



2. Input the program name in the *POU Name* field.
3. For *Type*, select ***Program***.

   Note If *Function Block* is selected a new function block will be created.
4. For *Language*, select the desired language. Select ***LD*** to program a ladder diagram or ***ST*** to program in standard text.
5. Click the **OK** Button.
   The logical POUs that are created will be displayed under *Logical POUs* in the Project Window. Also, a Ladder Editor and Local Variable Editor will be displayed for the program that was created.



   This completes creating a program.
6. Finally, code the program.
   Refer to *5-3 Programming Methods* for information on programming methods.

Note (1) By default, the Local Variable Editor will be display connected to the top of the Ladder Editor as shown above.
(2) To hide the Local Variable Editor, select ***View - Variable Editor - Visible***.
(3) To display the Local Variable Editor, select ***View - Variable Editor - Visible*** again.

5-2-3 Saving Edited Data in the Project

After coding programs or function blocks, be sure to save any data that was edited in the program.

To save the data, select ***File - Save Changes to project***. Alternately, press the **Ctrl+Shift+S** Keys.

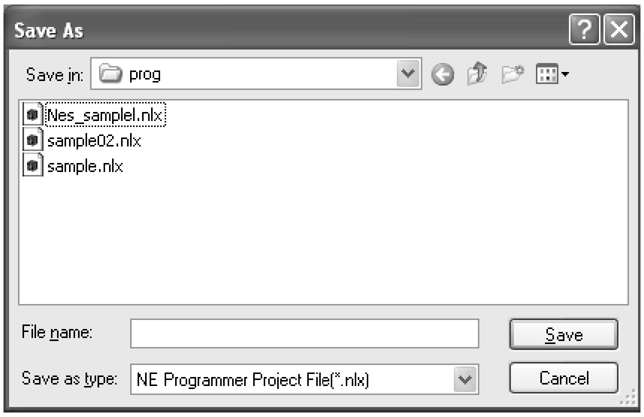
An alarm will be displayed if there is an error in the program. Correct the error and then select ***File - Save Changes to project*** again.

5-2-4 Saving the Project

Use the following procedure to save the project.

1,2,3...

1. Select ***File - Save*** or ***File - Save As***.
   The Save As Dialog Box will be displayed.



2. Specify the file location and file name to save in and then click the **Save** Button.
   The project will be saved with an .nlx filename extension.

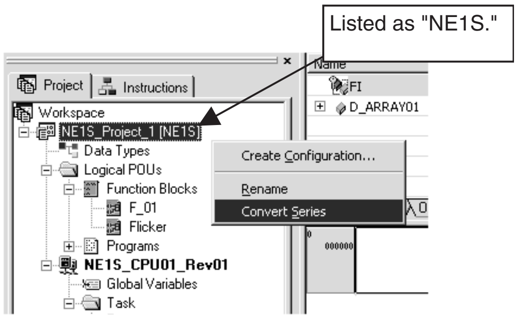
5-2-5 Changing the Controller Series

A project for an NE1S-series CPU Unit can be converted to a project for a CJ2 CPU Unit by changing the Controller series.

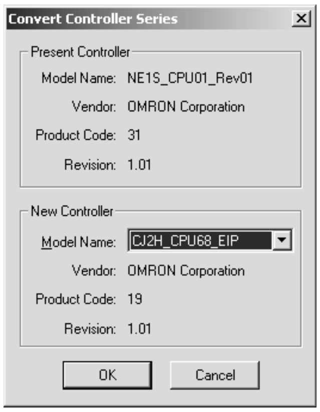
⚠ **Caution** The CJ2 CPU Units do not support initial value settings for variables. If you convert an NE1S project to a CJ2 project, any initial value settings used in the NE1S project will be lost and must be set from the program. Also, the instruction set and execution timing are different between the NE1S-series PLCs and the CJ2 CPU Units. After conversion, sufficiently check the user program (e.g., ladder program) and parameters to be sure that they operate correctly before using them for actual system operation. Always perform a program check on the program before downloading it to the PLC.

Note A project for a CJ2 CPU Unit cannot be changed to a project for a NE1S-series CPU Unit.

1,2,3... 1. Right-click to select the project in the Project Window.



2. Select ***Convert Series*** from the menu. The Convert Controller Series Dialog Box will be displayed. Select the type of new Controller and click the **OK** Button.

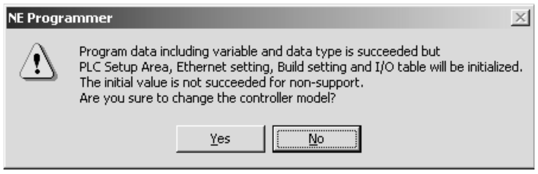

Convert Controller Series

Present Controller
Model Name: NE1S_CPU01_Rev01
Vendor: OMRON Corporation
Product Code: 31
Revision: 1.01

New Controller
Model Name: CJ2H_CPU68_EIP
Vendor: OMRON Corporation
Product Code: 19
Revision: 1.01

OK　Cancel

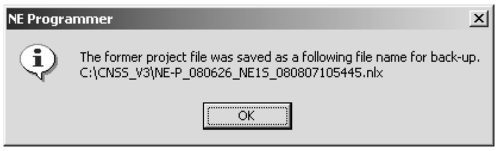
3. The following dialog box will be displayed. Click the **Yes** button.


NE Programmer

Program data including variable and data type is succeeded but
PLC Setup Area, Ethernet setting, Build setting and I/O table will be initialized.
The initial value is not succeeded for non-support.
Are you sure to change the controller model?

Yes　No

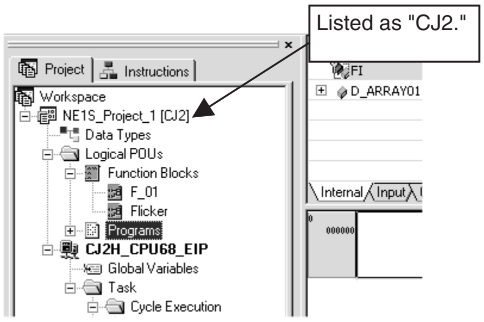
Next, a message box that gives the backup location of the project file before the change will be displayed. Click the **OK** Button.


NE Programmer

The former project file was saved as a following file name for back-up.
C:\CNSS_V3\NE-P_080626_NE1S_080807105445.nlx

OK

4. The Controller series (i.e., CPU Unit Series) will be changed. Confirm that the listing in the Project Window has changed.



5-3 Programming Methods

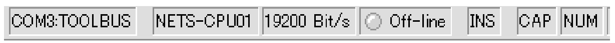
This section describes how to program using ladder diagrams or standard text, and how to edit local variables.

5-3-1 Ladder Diagrams

This section describes basic ladder programming methods. Refer to *SECTION 2* for details on variables and methods for inputting constants and operand numbers.

Note

(1) The default input mode is the insert mode. Press the **Insert** Key to change to overwrite mode. The mode will switch between overwrite and insert mode every time the Insert Key is pressed. Either INS or OVR will be displayed in the status bar to indicate the input mode.



(2) After programming, always select ***File - Save Changes to project***. Alternately, press the **Ctrl+Shift+S** Keys.
An alarm will be displayed if there is an error in the program. Correct the error and then select ***File - Save Change to project*** again. Alternately, press the **Ctrl+Shift+S** Keys.

(3) Always place an END instruction at the end of each program.

Changing the Display Zoom

The display zoom can be changed by using the View Menu, shortcut keys, or the toolbar. The display can be magnified or reduced in 20% increments from 20% to 200%.

Magnifying the Ladder Editor

Select ***View - Zoom In*** (or Alt + →).
The **Zoom In** Icon in the toolbar can also be clicked.

Reducing the Ladder Editor

Select ***View - Zoom Out*** (or Alt + ←).
The **Zoom Out** Icon in the toolbar can also be clicked.

Zooming to Fit the Window Width

Select ***View - Zoom to Fit*** (or Alt + ↑).
The **Zoom to Fit** Icon in the toolbar can also be clicked.

Displaying and Hiding the Grid

The Ladder Editor grid can be displayed or hidden.
Select ***View - Grid***.

Changing the Input Mode

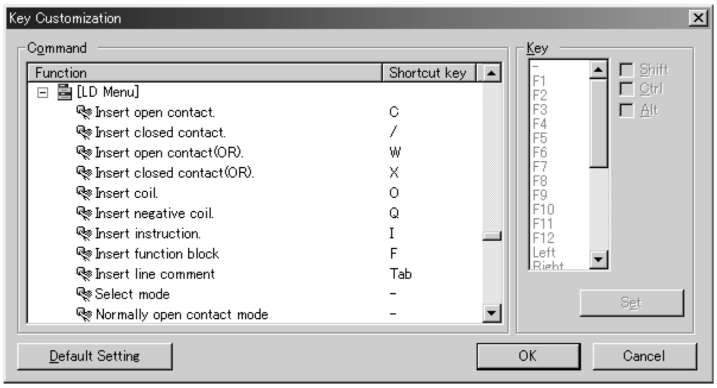
The input mode can be selected when bits or instructions are selected. The two following input modes can be used.

1. Mnemonic input mode
2. One-key input mode (default)

Select ***Tools - Option***, and perform the setting in the Edit Tab Page of the Ladder Dialog Box that will be displayed.

Assigning Keys for One-key Input

The following keys are assigned for one-key input by default.



Function	Shortcut key
[LD Menu]	
Insert open contact.	C
Insert closed contact.	/
Insert open contact(OR).	W
Insert closed contact(OR).	X
Insert coil.	O
Insert negative coil.	Q
Insert instruction.	I
Insert function block	F
Insert line comment	Tab
Select mode	-
Normally open contact mode	-

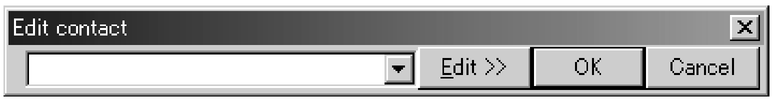
Keys assigned for one-key input can be changed as desired by using the Key Customization Window. Select ***Tool - Key Customize*** to access this window.

Inputting NO Conditions

Using One-key Input

1,2,3...

1. On the Ladder Editor, move the cursor to the cell where the NO condition is to be input.
2. Press the **C** Key. Lowercase can also be used. All other instructions are also not case sensitive.
   The Edit Contact Dialog Box will be displayed.

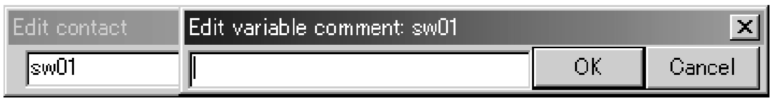
   

3. Enter the variable name and press the **Enter** Key (or click the **OK** Button).

   Example: sw01

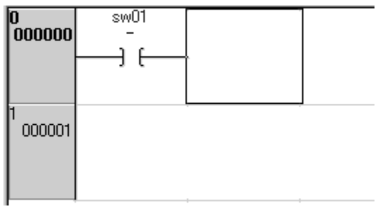
   Variable names are also not case sensitive, even though they are registered with the case that is input. Regardless of the case that is displayed, variables are not case sensitive. Example: The following all specify the same variable: abc, Abc, and ABC.

   The Edit Contact Dialog Box will be displayed.

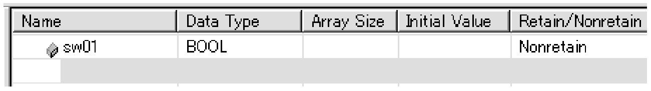
   

4. Press the **Enter** Key (or click the **OK** Button) after inputting a comment or leaving the entry blank. In this example, the **Enter** Key is pressed with the entry left blank.

As shown below, a NO input condition and the variable name will be displayed and the cursor will move to the next cell on the right.



If the variable that was specified above is not registered in the Variable Editor, it will automatically be registered as an internal variable in the Local Variable Editor.

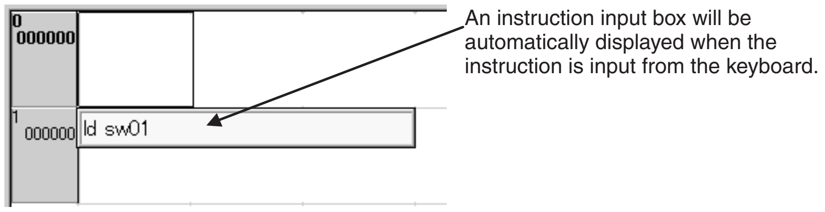


Name	Data Type	Array Size	Initial Value	Retain/Nonretain
sw01	BOOL			Nonretain

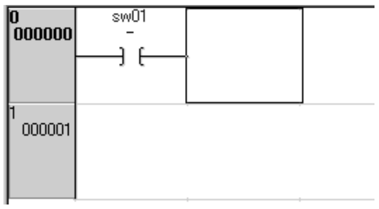
Using Mnemonic Input

1,2,3...
1. On the Ladder Editor, move the cursor to the cell where the NC condition is to be input.
2. Input ***ldnot****(space)(variable_name)* from the keyboard and then press the **Enter** Key. Either ***ld*** or ***LD*** may be input. Inputs for all other instructions are not case-sensitive.
   Variable names are also not case-sensitive, even though they are registered with the case that is input. Regardless of the case that is displayed, variables are not case-sensitive. Example: The following all specify the same variable: abc, Abc, and ABC.

Example: ld sw01


0
000000
1
000000
ld sw01
An instruction input box will be automatically displayed when the instruction is input from the keyboard.

When the Enter Key is pressed, the NO condition and variable will be displayed as shown below and the cursor will move to the next cell to the right.

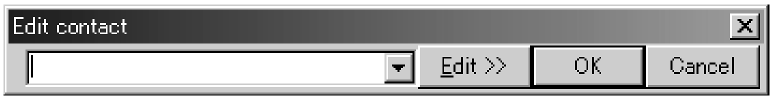


Inputting NC Conditions

Using One-key Input

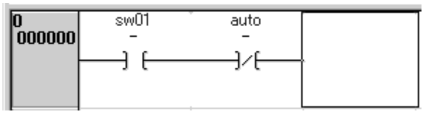
1,2,3...
1. On the Ladder Editor, move the cursor to the cell where the NC condition is to be input.

2. Press the **Forward Slash** Key.
   The Edit Contact Dialog Box will be displayed.



3. Enter the variable name and press the **Enter** Key (or click the **OK** Button).
   Example: auto
   The Edit Comment Dialog Box will be displayed.
4. Press the **Enter** Key (or click the **OK** Button) after inputting a comment or leaving the entry blank. In this example, the **Enter** Key is pressed with the entry left blank.

   As shown below, a NC input condition and the variable name will be displayed and the cursor will move to the next cell on the right.

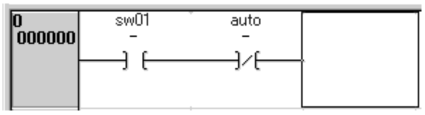


1,2,3...

1. On the Ladder Editor, move the cursor to the cell where the NC condition is to be input.
2. Input ***ldnot****(space)(variable_name)* from the keyboard and then press the **Enter** Key.

   The NC condition and variable will be displayed as shown below and the cursor will move to the next cell to the right.

   Example: ldnot auto

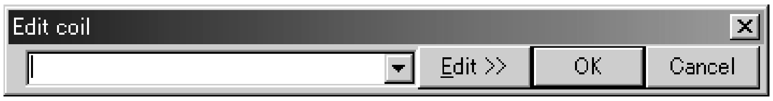


Inputting Outputs (Coils)

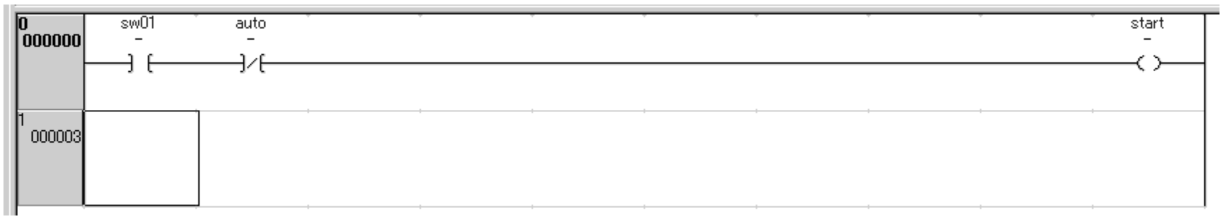
Using One-key Input

1,2,3...

1. Move the cursor to the cell to the right on the input conditions of the line in which to write the output, as indicated by the position of the cursor in the above figure.
2. Press the **O** Key (i.e., letter O).
   The Edit Coil Dialog Box will be displayed.



3. Enter the variable name and press the **Enter** Key (or click the **OK** Button).
   Example: start
   The Edit Comment Dialog Box will be displayed.
4. Press the **Enter** Key (or click the **OK** Button) after inputting a comment or leaving the entry blank. In this example, the **Enter** Key is pressed with the entry left blank.
   As shown in the following figure, the output and variable name will be displayed on the far right, and the cursor will move to the beginning of the next line.



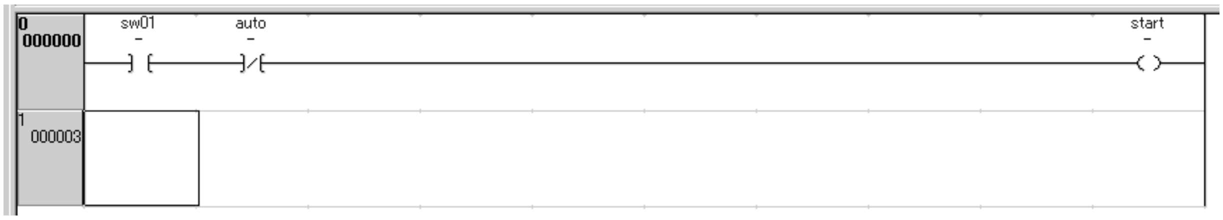
Using Mnemonic Input Mode

1,2,3...

1. Move the cursor to the cell to the right on the input conditions of the line in which to input the output, as indicated by the position of the cursor in the above figure.
2. Input ***outnot****(space)(variable_name*) from the keyboard and then press the **Enter** Key. To specify a negated output condition, input as follows: ***outnot****(space)(variable_name)*.

   The output and variable will be displayed in the rightmost cell as shown below and the cursor will move to the beginning of the next line.

   Example: out start



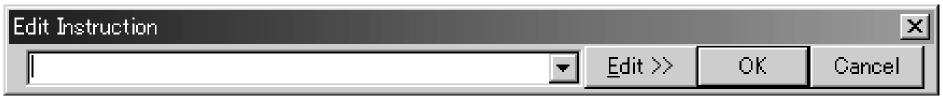
Note I/O Display for I/O Bits
"I" for inputs and "Q" for outputs or "X" for inputs and "Y" for outputs can be displayed before input bit addresses in the ladder window by selecting *Display IQ* or *Display XY* in the *Display I/O Address* Field of the General Tab Page in the Option Menu.

Inputting Special Instructions

Using One-key Input Mode

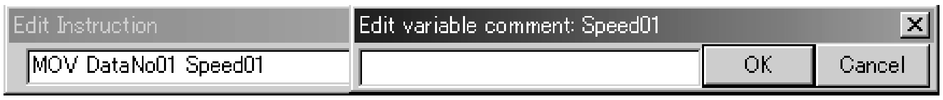
1,2,3...

1. Move the cursor to the cell to the right on the input conditions of the line in which to input the special instruction.
2. Press the **I** Key.
   The Edit Instruction Dialog Box will be displayed.



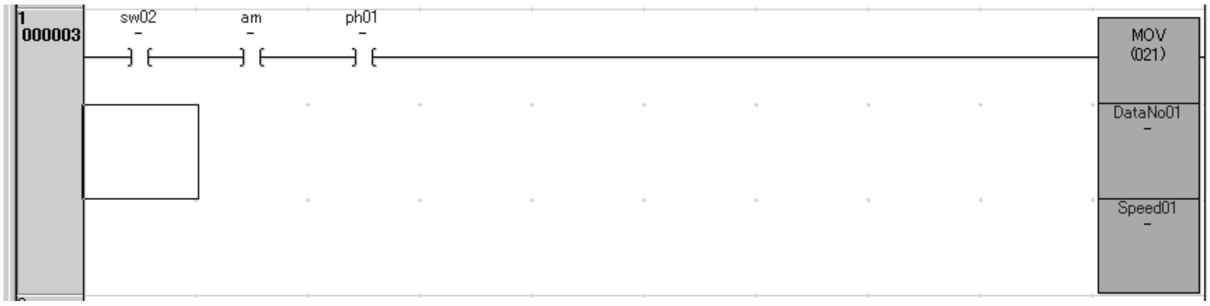
3. Input (special instruction)(space)(operand variable name) [(space)(operand variable name)] from the keyboard, and then press the **Enter** Key (or click the **OK** Button).
   Example: mov DataNo01 Speed01

   The Edit Variable Comment Dialog Box will be displayed.



MOV DataNo01 Speed01

4. Press the **Enter** Key (or click the **OK** Button) after inputting a comment or leaving the entry blank. In this example, the **Enter** Key is pressed with the entry left blank.
   As shown in the following figure, the special instruction and the operand variable name will be displayed on the far right, and the cursor will move to the beginning of the next line.



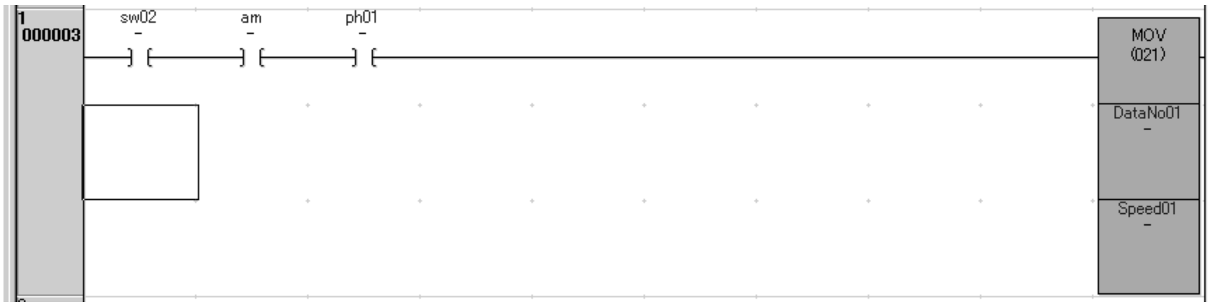
Using Mnemonic Input Mode

1,2,3...

1. Move the cursor to the cell to the right on the input conditions of the line in which to input the instruction.
2. From the keyboard,
   input *(instruction)(space)(variable_name_of_operand)*...,
   repeated "*(space)(variable_name_of_operand)*"
   for each operand required by the instruction, and then press the **Enter** Key.

   The instructions and operand variables will be displayed in the rightmost cell as shown below and the cursor will move to the beginning of the next line.

   Example: mov DataNo01 Speed01



Inputting Special Instructions from the Function Window

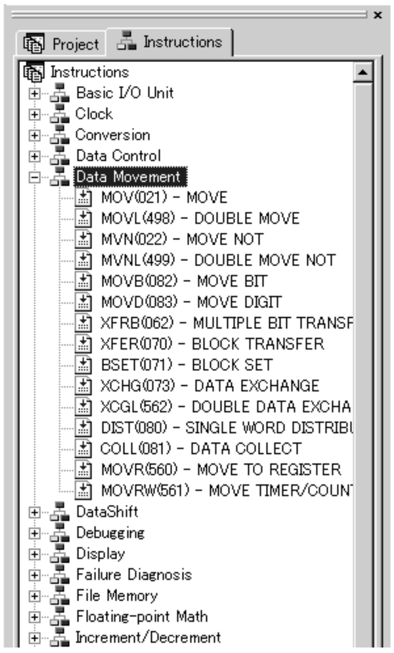
Special instructions can also be added to the program by dragging and dropping them from the Function Window onto the Ladder Editor.

The following two display methods can be used for the Function Window.

- Categorized display (categorization)
- Alphabetical display (ascending alphabetical order)

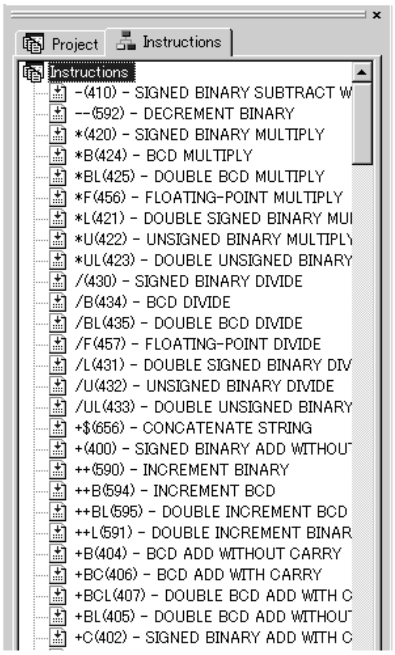
The display method can be switched by right-clicking the **Instructions** Tab in the Function Window and selecting *View - Classified* or *View - Alphabetical.*

• Classified



```
Project   Instructions
Instructions
  Basic I/O Unit
  Clock
  Conversion
  Data Control
  Data Movement
    MOV(021) - MOVE
    MOVL(498) - DOUBLE MOVE
    MVN(022) - MOVE NOT
    MVNL(499) - DOUBLE MOVE NOT
    MOVB(082) - MOVE BIT
    MOVD(083) - MOVE DIGIT
    XFRB(062) - MULTIPLE BIT TRANSF
    XFER(070) - BLOCK TRANSFER
    BSET(071) - BLOCK SET
    XCHG(073) - DATA EXCHANGE
    XCGL(562) - DOUBLE DATA EXCHA
    DIST(080) - SINGLE WORD DISTRIB
    COLL(081) - DATA COLLECT
    MOVR(560) - MOVE TO REGISTER
    MOVRW(561) - MOVE TIMER/COUN
  DataShift
  Debugging
  Display
  Failure Diagnosis
  File Memory
  Floating-point Math
  Increment/Decrement
```

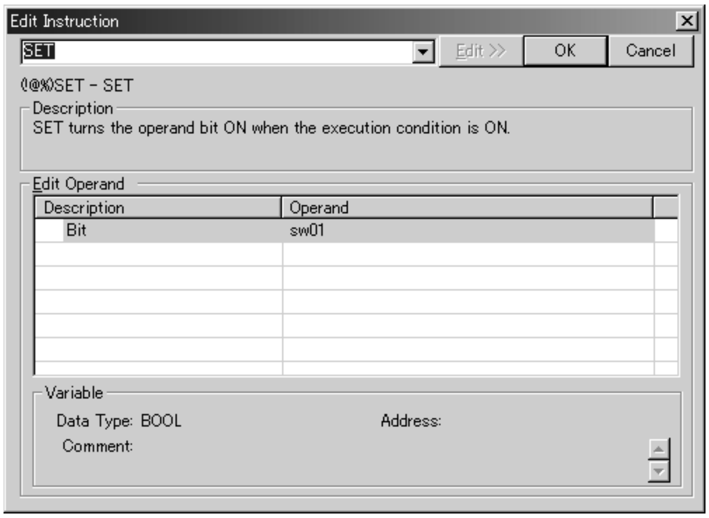
• Alphabetical



```
Project   Instructions
Instructions
  -(410) - SIGNED BINARY SUBTRACT W
  --(592) - DECREMENT BINARY
  *(420) - SIGNED BINARY MULTIPLY
  *B(424) - BCD MULTIPLY
  *BL(425) - DOUBLE BCD MULTIPLY
  *F(456) - FLOATING-POINT MULTIPLY
  *L(421) - DOUBLE SIGNED BINARY MU
  *U(422) - UNSIGNED BINARY MULTIPLY
  *UL(423) - DOUBLE UNSIGNED BINARY
  /(430) - SIGNED BINARY DIVIDE
  /B(434) - BCD DIVIDE
  /BL(435) - DOUBLE BCD DIVIDE
  /F(457) - FLOATING-POINT DIVIDE
  /L(431) - DOUBLE SIGNED BINARY DIV
  /U(432) - UNSIGNED BINARY DIVIDE
  /UL(433) - DOUBLE UNSIGNED BINARY
  +$(656) - CONCATENATE STRING
  +(400) - SIGNED BINARY ADD WITHOU
  ++(590) - INCREMENT BINARY
  ++B(594) - INCREMENT BCD
  ++BL(595) - DOUBLE INCREMENT BCD
  ++L(591) - DOUBLE INCREMENT BINAR
  +B(404) - BCD ADD WITHOUT CARRY
  +BC(406) - BCD ADD WITH CARRY
  +BCL(407) - DOUBLE BCD ADD WITH C
  +BL(405) - DOUBLE BCD ADD WITHOU
  +C(402) - SIGNED BINARY ADD WITH C
```

Editing Instructions

1,2,3... 1. Double-click the instruction to be edited. Alternately, right-click the instruction and select ***Edit Instruction***. Alternately, move the cursor to the instruction and press the **Enter** Key.
The Edit Instruction Dialog Box will be displayed.



Edit Instruction

SET | Edit >> | OK | Cancel

(@%)SET - SET

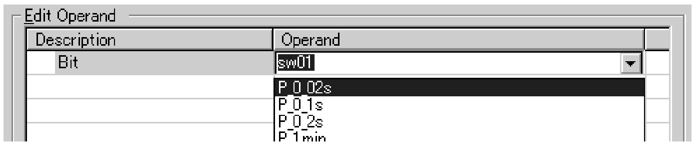
Description
SET turns the operand bit ON when the execution condition is ON.

Edit Operand

Description	Operand
Bit	sw01

Variable
Data Type: BOOL   Address:
Comment:

2. Double-click an operand to edit it. (Alternately, select the operand and press the **Enter** Key.
A list of selections for the operand will be displayed as shown below.



Edit Operand

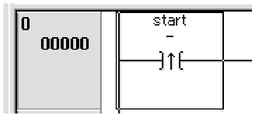
Description	Operand
Bit	sw01

```
P_0_02s
P_0_1s
P_0_2s
P_1min
```

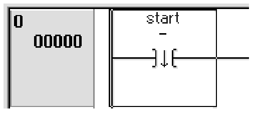
3. Complete editing the instruction and then click the **OK** Button.

Inputting Differentiated Conditions

1,2,3...
1. Input the input condition.
2. Right-click the input condition, select ***Transition-Sensing - Positive*** or ***Transition-Sensing - Negative***.
   - Example of Positive Transition Sensing



   - Example of Negative Transition Sensing

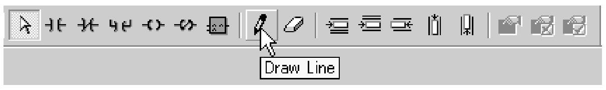


Note Input can also be made in the Details Display of the Edit Instruction Dialog Box. In addition, an upward (positive) transition can be specified using @ + instruction and a downward (negative) transition can be specified using the % + instruction in mnemonic input mode.

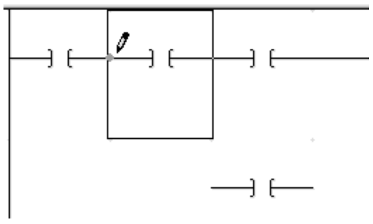
Inputting Vertical and Horizontal Lines: Line Connection Mode

Inputting Lines with the Mouse

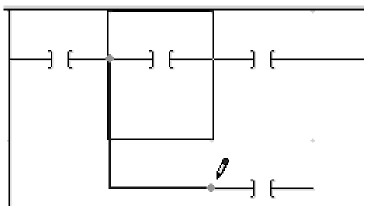
1,2,3...
1. Click the **Draw Line** Icon in the toolbar to enter line connection mode.



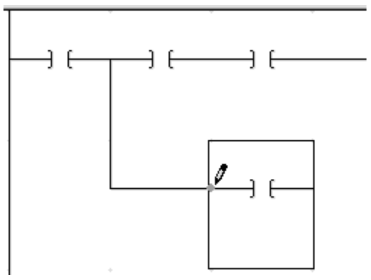
2. Move the cursor to the starting point of the line (indicated as a light green dot) and click the left mouse button to set the starting point.



3. Move the cursor to the end point of the line. The line to be created will be indicated in blue.



4. Click the end point of the line.
   The line will be displayed.



To leave line connection mode, click the **Select** Icon.

Inputting Lines from the Keyboard

1,2,3...

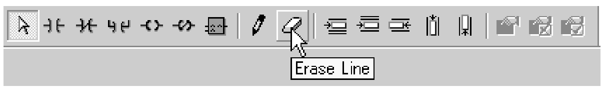
1. Move the cursor to the starting point for the line and then press the **Ctrl+L** Keys.
2. Move the cursor to the end point of the line using the cursor keys. The line to be created will be indicated in blue.
3. Press the **Ctrl+L** Keys at the end point for the line.
   The line will be displayed.

Deleting Vertical and Horizontal Lines: Line Deletion Mode

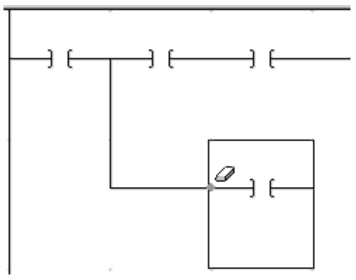
Deleting Lines with the Mouse

1,2,3...

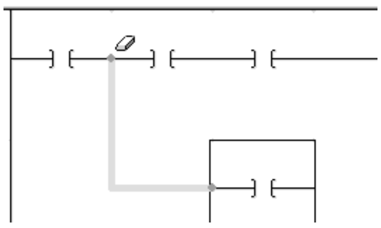
1. Click the **Erase Line** Icon in the toolbar to enter line deletion mode.



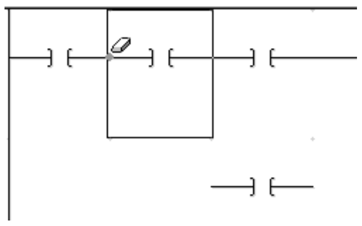
2. Move the cursor to the starting point of the line to be deleted (indicated as a light green dot) and click the left mouse button to set the starting point.



3. Move the cursor to the end point of the line to be deleted. The line to be deleted will be indicated in gray.



4. Click the end point of the line to be deleted.
   The line will be deleted.



To leave line deletion mode, click the **Select** Icon.

Deleting Lines from the Keyboard

1,2,3...
1. Move the cursor to the starting point for the line and then press the **Ctrl+Shift+L** Keys.
2. Move the cursor to the end point of the line using the cursor keys. The line to be deleted will be indicated in gray.
3. Press the **Ctrl+Shift+L** Keys at the end point for the line.
   The line will be deleted.

Deleting Instructions and Lines

Instructions and lines can be deleted by pressing the **Backspace** or **Delete** Key in the same way as for standard text editors or word processors.

Lines can also be deleted in line deletion mode. Refer to *Deleting Vertical and Horizontal Lines: Line Deletion Mode* for the procedure.

Registering and Editing Local Variables

Registering Variables

When a logical POU (program or function block) is created, a Local Variable Editor will be displayed for each one. Variables can be registered or edited in the Local Variable Editor using either of the following methods.

1,2,3...
1. Inputting Instructions First: As shown above under *Inputting NO Conditions*, if a new variable is specified as an operand when inputting an instruction, the variable will be registered as an internal variable in the Local Variable Editor. If necessary, the parameters of these variables can later be edited in the Local Variable Editor. (Refer to *Editing Variables* later in this section.)
2. Entering Variables in Variable Editor First: The Edit Variables Dialog Box can be displayed by right-clicking in the Local Variable Editor and selecting *Add* (or alternately by double-clicking the table) to enable registering variables in the Local Variable Editor. Refer to *Editing Variables* later in this section for information on the Edit Variable Dialog Box.
   The variables that were registered can then be input for operands when inputting instructions.

Editing Variables

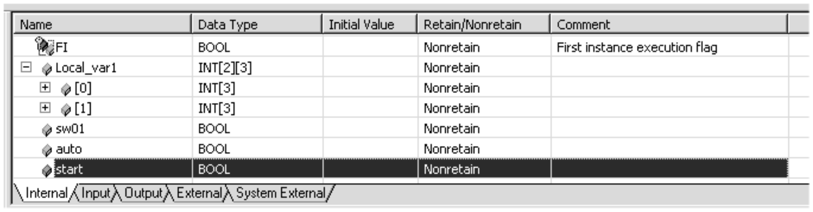
This section describes methods for editing variables and variable parameters.

■ **Displaying the Local Variable Editor**

If the Local Variable Editor is not displayed, double-click the program name or function block name in the Project Window.

The Local Variable Editor and Ladder Editor will be displayed.

If the Local Variable Editor is still not displayed, select ***View - Variable Editor - Visible***.

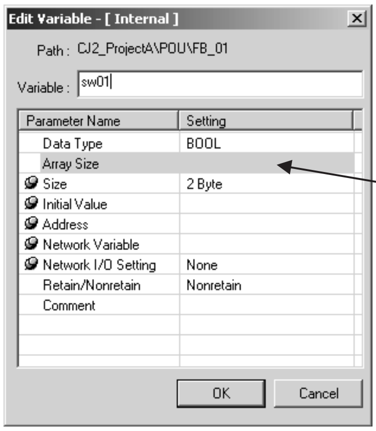


Name	Data Type	Initial Value	Retain/Nonretain	Comment
FI	BOOL		Nonretain	First instance execution flag
Local_var1	INT[2][3]		Nonretain	
[0]	INT[3]		Nonretain	
[1]	INT[3]		Nonretain	
sw01	BOOL		Nonretain	
auto	BOOL		Nonretain	
start	BOOL		Nonretain	

■ Editing Variable Parameters

1,2,3...

1. Double-click the variable to be edited. Alternately, right-click the instruction and select ***Edit***. The Edit Variables Dialog Box will be displayed.



Path : CJ2_ProjectA\POU\FB_01

Variable : sw01

Parameter Name	Setting
Data Type	BOOL
Array Size	
Size	2 Byte
Initial Value	
Address	
Network Variable	
Network I/O Setting	None
Retain/Nonretain	Nonretain
Comment	

Double-click the parameter to edit. Parameters with the following mark cannot be edited:

2. To edit other parameters, double-click the parameter to be edited.
   For local variables, physical addresses can be directly input in the *Address* Field of the Edit Variables Dialog Box.
3. Edit the parameter.
4. When all editing has been completed, click the **OK** Button.
   Variable parameters are given in the following table.

Variable Parameter	Description	Values
Variable name	Displays the name of the variable being edited. The name of the variable can also be changed.	Refer to *2-2-1 Naming Variables* for details.
Data Type	Set the data type of the variable.	BOOL, INT, UINT, DINT, UDINT, WORD, DWORD, TIMER, COUNTER, STRING, REAL, or user-defined For a function block, the name of the logical POU of the function block will be displayed.
Array Size	Set the number of elements for a one-dimensional array or a two-dimensional array. Two-dimensional arrays can be set only for CJ2 CPU Units.	When not specifying an array, leave this settings blank for both a one-dimensional and two-dimensional array. When specifying a one-dimensional array, set the number of array elements for a one-dimensional array and leave the setting for a two-dimensional array blank. When specifying a two-dimensional array, set the number of array elements for a two-dimensional array and leave the setting for a one-dimensional array blank. *A two-dimensional array cannot be specified for the NE1S. *Refer to *2-2-3 Variable Properties* for other restrictions on elements.
Size	Displays the size used by the variable in the memory.	---

Variable Parameter	Description	Values
Initial Value	For programs, set the initial value of the variable when program execution is started. For function blocks, set the value of the variable when an instance of the function block is executed. **Note** The initial values of variables cannot be set for CJ2 CPU Units.	Set the initial value of the variable according to the data type. • BOOL, WORD, or DWORD: Unsigned hexadecimal Input the number after "16#". • INT or DINT: Signed decimal Input the number after "+10#" or "–10#". • UINT or UDINT: Unsigned decimal Input the number after "10#". • REAL: Real number (e.g., +1.0, –0.23, +9.8E-3) • STRING: Character string, e.g., "Data"
Address	This property cannot be changed for local variables.	---
Network Variable	This property cannot be changed for local variables.	Always *Enabled*.
Network I/O Setting	This property cannot be changed for local variables.	Always *None*.
Retain/Nonretain	Specify whether to maintain the value of the variable when power is turned OFF and ON, and when operation is started.	Retain or Nonretain
Comment	Input a comment for the variable.	256 characters max.

Note (1) When the Cross Reference Pop-up Window has been displayed (by selecting ***View - Window - Cross Reference Tool***), a variable's cross reference information (program address, instruction name, program name, etc.) can be displayed just by selecting that variable in the Variable Editor.

(2) Refer to *2-2 Variables* for detailed variable specifications.

■ Inserting Variables

1,2,3...
1. Right-click the line at which to insert a variable and select ***Insert*** from the pop-up menu. Alternately, double-click an empty row.
   The Edit Variables Dialog Box will be displayed.
2. Set the parameters of the Variable and then click the **OK** Button.
   The variable will be inserted.

■ Deleting Variables

1,2,3...
1. Select the variable to be deleted and press the **Delete** Key.
2. A dialog box will appear to confirm the deletion. Click the **Yes** Button.
   The variable will be deleted.

Specifying Arrays

Arrays can be used to handle an array as a group of data elements with the same properties. To create an array variables, set the *Array Size* in the Edit Variables Dialog Box to a value between 1 and 255.

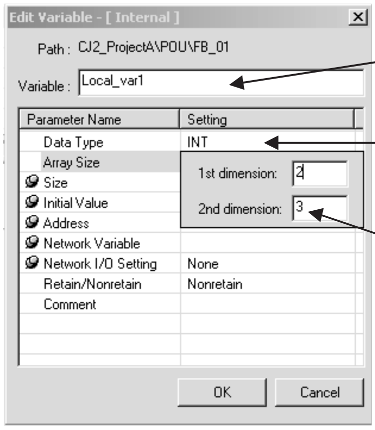
- Arrays can be specified for internal variable (VAR), input variables (VAR_INPUT), output variables (VAR_OUTPUT), or external variables (VAR_EXTERNAL).
- With NE1S-series CPU Units, only one-dimensional arrays can be created. With CJ2 CPU Units, one-dimensional and two-dimensional arrays can be created.

- When specifying the name of an array variable in a program, the index must be placed in brackets after the variable name (example: ARRAY[0]). Refer to *2-2-5 Array Elements (Array Specification)* for details on using indices.

Use the following procedure to specify an array variable.

1,2,3...

1. Add a variable in the Variable Editor. Alternately, double-click an empty row. To edit an existing variable, double-click the variable to be edited.
   The Edit Variables Dialog Box will be displayed. Click the *Array Size Row*. The dialog box for setting the number of elements for one-dimensional and two-dimensional arrays will be displayed.



Path : CJ2_ProjectA\POU\FB_01

Variable : Local_var1

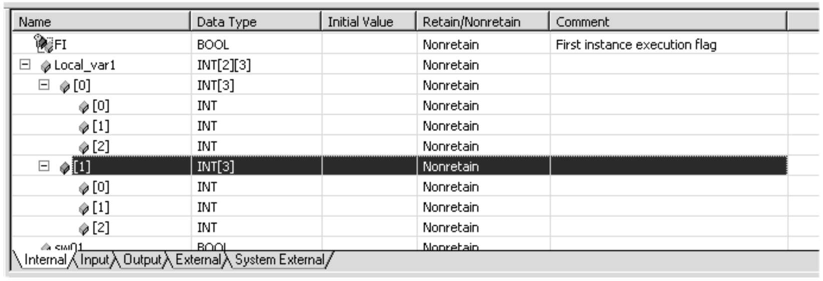
Parameter Name	Setting
Data Type	INT
Array Size	1st dimension: 2 / 2nd dimension: 3
Size	
Initial Value	
Address	
Network Variable	
Network I/O Setting	None
Retain/Nonretain	Nonretain
Comment	

Input variable name.

Double-click to change data type.

Click here and then enter the number of array elements. For the NE1S, only a one-dimensional array can be set. A two-dimensional array cannot be set. For the CJ2, a one-dimensional or two-dimensional array can be set.

2. Set the *1st Dimension* Field and *2nd Dimension* Field to a number from 1 to 9 and click the **OK** Button. The variable will be defined as an array as shown in the following figure. In this example, the numbers of array elements are set to 2 and 3.



Name	Data Type	Initial Value	Retain/Nonretain	Comment
FI	BOOL		Nonretain	First instance execution flag
⊟ Local_var1	INT[2][3]		Nonretain	
⊟ [0]	INT[3]		Nonretain	
[0]	INT		Nonretain	
[1]	INT		Nonretain	
[2]	INT		Nonretain	
⊟ [1]	INT[3]		Nonretain	
[0]	INT		Nonretain	
[1]	INT		Nonretain	
[2]	INT		Nonretain	
sw01	BOOL		Nonretain	

The variable will be one-dimensional if no entry is made in the 2nd Dimension Field.

Creating Data Structures

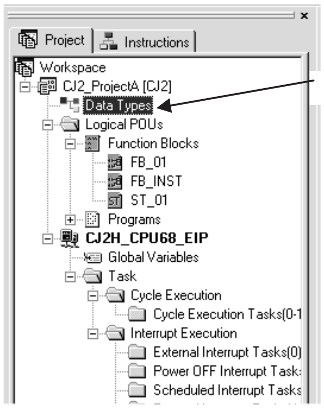
A data structure is a variable consisting of elements with different data types that are treated as a single variable. The user can define data structures as required. The variable name and element names can be specified for specific elements.

Use the following procedure to create a data structure.

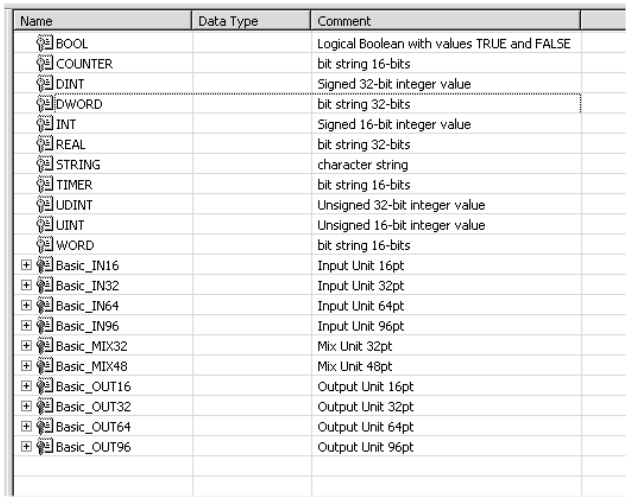
Inserting a Data Structure

1,2,3...

1. Double-click ***Data Types*** in the Workspace.

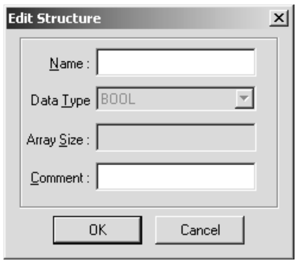


The Data Structure Table will be displayed. All data structures that are currently registered will be displayed in the Data Structure Table.



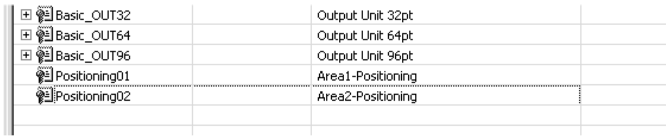
Name	Data Type	Comment
BOOL		Logical Boolean with values TRUE and FALSE
COUNTER		bit string 16-bits
DINT		Signed 32-bit integer value
DWORD		bit string 32-bits
INT		Signed 16-bit integer value
REAL		bit string 32-bits
STRING		character string
TIMER		bit string 16-bits
UDINT		Unsigned 32-bit integer value
UINT		Unsigned 16-bit integer value
WORD		bit string 16-bits
Basic_IN16		Input Unit 16pt
Basic_IN32		Input Unit 32pt
Basic_IN64		Input Unit 64pt
Basic_IN96		Input Unit 96pt
Basic_MIX32		Mix Unit 32pt
Basic_MIX48		Mix Unit 48pt
Basic_OUT16		Output Unit 16pt
Basic_OUT32		Output Unit 32pt
Basic_OUT64		Output Unit 64pt
Basic_OUT96		Output Unit 96pt

2. Right-click the last line (where nothing is registered) and select ***Insert - Struct***.
   The Edit Structure Dialog Box will be displayed.



3. Input the name of the structure and any comment that is required and then click the **OK** Button.

   The data structure will be inserted as a data type and displayed as shown below.

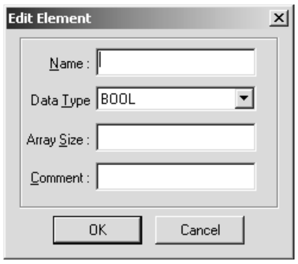


Basic_OUT32		Output Unit 32pt
Basic_OUT64		Output Unit 64pt
Basic_OUT96		Output Unit 96pt
Positioning01		Area1-Positioning
Positioning02		Area2-Positioning

Inserting Elements

1,2,3...

1. Right-click the data structure that was added (*Positioning01* in the above example) and select ***Insert - Element***.
   The Edit Element Dialog Box will be displayed.



Edit Element

Name :

Data Type BOOL

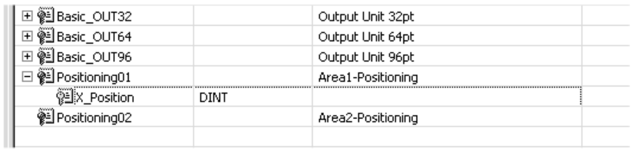
Array Size :

Comment :

2. Input the name of the element, the data type, the array size (if required), and any comment that is required and then click the **OK** Button.

   Note If an array size is specified, the element will be defined as an array inside the data structure.

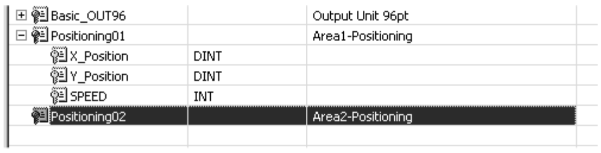
   The element will be inserted and displayed as shown below.



⊞ Basic_OUT32		Output Unit 32pt
⊞ Basic_OUT64		Output Unit 64pt
⊞ Basic_OUT96		Output Unit 96pt
⊟ Positioning01		Area1-Positioning
X_Position	DINT	
Positioning02		Area2-Positioning

3. Repeat the above steps to add other elements.

   Setting Example



⊞ Basic_OUT96		Output Unit 96pt
⊟ Positioning01		Area1-Positioning
X_Position	DINT	
Y_Position	DINT	
SPEED	INT	
Positioning02		Area2-Positioning

Grouping Variables

Input variables (VAR_INPUT) or output variables (VAR_OUTPUT) can be grouped for function block to display a group name for the inputs or outputs. Refer to page 121 in *5-4-2 Programming a Function Block* for details.

5-3-2 List of Inputs for Instructions

Use the following keys to input execution conditions.

Key	Execution condition
@	Upward differentiation
%	Downward differentiation
!	Immediate refreshing

Use the following inputs to specify instructions.

Spaces are indicated by □.

Instruction	Input
LD	LD□ *variable_name*
OR	OR□ *variable_name*
AND	AND□ *variable_name*
LDNOT	LDNOT□ *variable_name*

Instruction	Input
ORNOT	ORNOT□ *variable_name*
ANDNOT	ANDNOT□ *variable_name*
OUT	OUT□ *variable_name*
OUTNOT	OUTNOT□ *variable_name*
!LD	! LD□ *variable_name*
!AND	! AND□ *variable_name*
!OR	! OR□ *variable_name*
!LDNOT	! LDNOT□ *variable_name*
!ANDNOT	! ANDNOT□ *variable_name*
%LD	% LD□ *variable_name*
%AND	% AND□ *variable_name*
%OR	% OR□ *variable_name*
!@LD	! @ LD□ *variable_name*
!@AND	! @ AND□ *variable_name*
!@OR	! @ OR□ *variable_name*
!%LD	! % LD□ *variable_name*
!%AND	! % AND□ *variable_name*
!%OR	! % OR□ *variable_name*
!OUT	! OUT□ *variable_name*
!OUTNOT	! OUTNOT□ *variable_name*
TIMER (TIMX)	TIMX□*timer_number*□10# *set_value* or 16# *set_value*
COUNTER (CNTX)	CNTX□*counter_number*□10# *set_value* or 16# *set_value*
Special Instructions executed each scan	*instruction*□*operand*□*operand*□*operand*
Upwardly differentiated instructions	@ *instruction*□*operand*□*operand*□*operand*
Downwardly differentiation instructions	% *instruction*□*operand*□*operand*□*operand*
Immediate refresh instructions	! *instruction*□*operand*□*operand*□*operand*

Refer to *5-3-4 Inputting Constants* and *5-3-5 Inputting Operand Numbers* for the methods for inputting constants and operand numbers.

5-3-3 Inputting Bit and Word Addresses

The inputs used to specify bit and word addresses are listed in the following table.

Refer to *5-3-4 Inputting Constants* and *5-3-5 Inputting Operand Numbers* for the methods for inputting constants and operand numbers.

Area	Bit address	Word address
CIO Area	*bit-address*	*word-address*
Work Area (word bits)	W*bit-address*	**W***word-address*
Holding Area	**H***bit-address*	**H***word-address*
Auxiliary Area	**A***bit-address*	**A***word-address*
Timer Area	**T***timer_number*	**T***timer-number*
Counter Area	**C***counter-number*	**C***counter-number*
Task Area	**TK***task-number*	---
DM Area	---	**D***word-address*

Area	Bit address	Word address
EM Area	---	**E***word-address*
Indirect address in DM Area	---	**@D***word-address*
Indirect address in EM Area	---	***E0**_*word-address* **@E0**_*word-address,*
Constants	---	10# +*number*16# +number, etc.
Data Register Area	---	**DR***address*
Index Register Area	---	**IR***address*
Indirect address in Index Register Area	---	,IR* ,IR*+ ,IR*++ ,–IR* ,—IR*DR*, IR* XXXX, IR*

5-3-4 Inputting Constants

When inputting a constant in the following cases, always include the prefix code corresponding to the data type (16#, 10#, or -10# for integers, or decimal point or □E□ for real numbers).

- Inputting an instruction operand (other than timer/counter number inputs)
- Inputting I/O memory data
- Inputting a constant for an initial value
- Inputting function block parameters

Unsigned Hexadecimal: Input 16#

Input the prefix 16# before a hexadecimal constant.
For example, input “16#1A” to input the hexadecimal value 1A.

Positive or Zero Decimal: Input 10#

Input the prefix 10# before a positive decimal constant (or 0).
For example, input “10#123” to input the decimal value 123.

Negative Decimal: Input -10#

Input the prefix -10# before a negative decimal constant.
For example, input “-10#123” to input the decimal value -123.

Real Number: Decimal or □E□

Real numbers can be input with decimal points or as □E□. For example, the input “-0.123” or “-1.23E-1” is acceptable.

To input a fixed character string, insert it between single quotation marks (').

Example: 'abcdefg'

Binary (ST Program Only): Input 2#

Binary numbers can be input in ST language only. Input the prefix 2# before a binary constant.
For example, input “2#010” to input the binary value 010.

Octal (ST Program Only): Input 8#

Octal numbers can be input in ST language only. Input the prefix 8# before an octal constant.
For example, input “8#10” to input the octal value 10.

⚠ **Caution** If a number is input without a 16#, 10#, or -10# prefix code, it is treated as a word address in the CIO Area.
For example, an input of “100” specifies CIO 0100 in the CIO Area.

5-3-5 Inputting Operand Numbers

Inputting Timer/Counter Numbers

Input timer and counter numbers (and only these numbers) as a plain decimal value without a 10# prefix code. (A prefix such as 16# or 10# cannot be input.)

For example, to enter a timer with timer number 2 and decimal SV of 200, input TIMX 2 10#200.

Inputting Jump Numbers

Always input jump numbers with a 16# or 10# prefix code. If the prefix is omitted, the number indicates a CIO Area word address and the content of that address is used as the jump number.

For example, to enter a JMP instruction with jump number 2, input JMP 10#2 or JMP 16#2. (If JMP 2 is input, the content of CIO 0002 is used as the jump number.)

Inputting Task Numbers

Always input a decimal task numbers with the 10# prefix code. (A prefix code of 16# cannot be used.) If the prefix is omitted, the number indicates a CIO Area word address and the content of that address is used as the task number.

For example, to enter a TASK ON instruction with task number 3, input TKON 10#3. (If TKON 3 is input, the content of CIO 0003 is used as the task number.)

5-3-6 Converting Specified Physical Addresses to Variables

If a ladder program was created with physical addresses instead of variable names and global variables for the physical addresses were added later, the physical addresses in the program can all be converted to global variables at once. In that case, global variables will be automatically added to external variables.

Example Application:
This function is useful when a program is created first with physical addresses and then corresponding variable names are later assigned altogether.

Note

(1) If there is a physical address duplication error (build error), the conversion will not be performed because the replacement global variable cannot be determined.

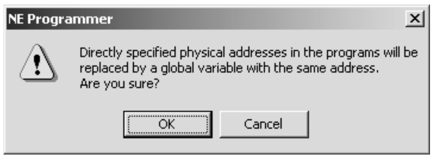
(2) A global variable will not be replaced if there is another kind of variable (local variable in the ladder program or external variable) with the same name as the global variable.

Procedure

1,2,3...

1. Select **File - Configuration - Replace physical addresses in programs** or right-click a *Global Variables* in the Project Window and select **Replace physical addresses in programs** from the popup menu.

   A dialog box will be displayed to confirm that the physical addresses in the program will be replaced by the specified variables. Click the **OK** Button to continue.
2. A message box for confirmation will be displayed. Click the **OK** Button.


NE Programmer

Directly specified physical addresses in the programs will be replaced by a global variable with the same address.
Are you sure?

3. The Find Tab of the Output Window will show the progress of the conversion (replacing or completed).

5-3-7 Support for Converting to Physical Addresses

This function automatically creates an AT-specified global variable for a physical address when a physical address is input for an input condition or output bit and then converts it to a corresponding external variable.

If the physical address is directly input into the Ladder Editor and then a comment is input, the variable will be automatically created.

It is also possible at the same time to input comments during continued operation by inputting or selecting a variable in the Ladder Editor and pressing the Enter Key.

Enabling Support for Converting to Physical Addresses

The following settings must be made in Option Settings to enable support for converting to physical addresses.

- Select the *One Key Input Mode* Option in the *Input Mode* Field on the Edit Tab Page of the Ladder Window and select the *In comment inputting* Option (default: OFF).

The following precautions apply to using this function.

- To use this function, there must be a configuration.
- The network variable conversion attribute (described below) of the variable being created must be set to disable network variable conversion.
- It may not be possible to automatically create variables if there are variables with the same name but different data types.

Automatically created variables can be checked in the Variable Editor.

⚠ **Caution** Support for converting to physical addresses is achieved by inputting a comment. If a comment is not input, variables will not be automatically created even if the settings have been made and the necessary conditions have been met. If a physical address is created without a comment, however, variables can be created automatically if a comment is input later.

Automatically Created Variable Names

The following variable names are created automatically.

Word Addresses

"_address_"

An underscore character is inserted before and after the physical address.

Example: _D10000_, _E0_10000_

Bit Addresses

"_wordaddress_bitaddress_"

An underscore character is inserted before and after the physical address and also between the word address and bit address.

Example: _0000_12_, _A100_15_

Variables that are automatically created from physical addresses cannot be accessed externally.

Note Physical addresses may remain unchanged if variables cannot be automatically generated due to mismatching data types with external variables of the same name.

Input Procedure Example

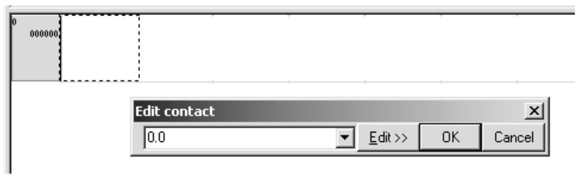
Using Bit Address Variables

There are two input methods. (The differences are given in step 3.)

1,2,3...

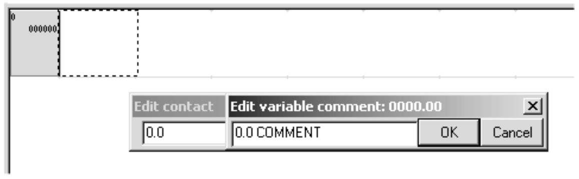
1. In the Ladder Editor, right-click and select an instruction from the Insert Menu. (In this example, a normally open condition is selected.)

2. Enter a variable name in the Edit Contact Dialog Box. (In this example, 0.0 is entered.)



• Press the **Enter** Key to input a comment.

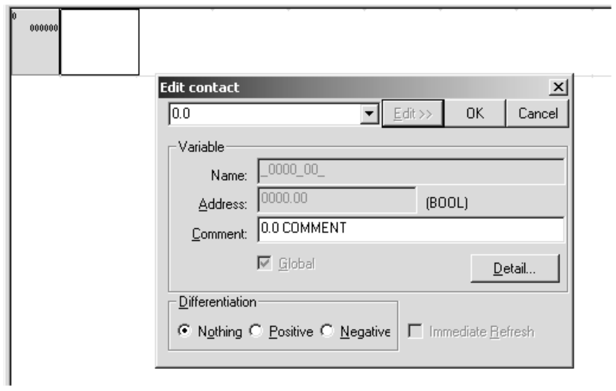
3. Press the **Enter** Key. The Edit Variable Comment Dialog Box will be displayed. Enter a comment and click the **OK** Button.



• Enter a comment with the **Edit** Button.

3. Click the **Edit** Button.

Detailed settings will be displayed in the Variable Area and elsewhere. Enter a comment and click the **OK** button.



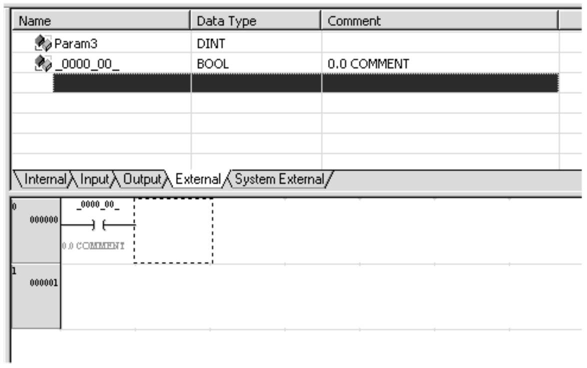
0.0

Name: _0000_00_

Address: 0000.00 (BOOL)

Comment: 0.0 COMMENT

4. Automatically created variable names and input comments will be displayed in the Variable Editor and Ladder Editor.



Name	Data Type	Comment
Param3	DINT	
_0000_00_	BOOL	0.0 COMMENT

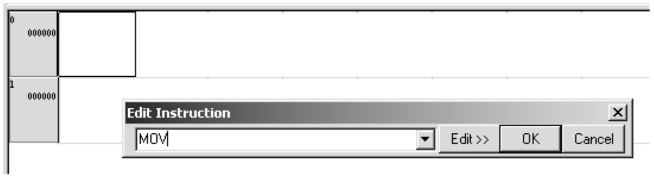
⚠ **Caution** The Edit Variable Comment Dialog Box will not be displayed when the Enter Key is pressed in step 2 if the *In comment inputting* Option has not been selected on the Edit Tab Page in Option Settings of the Ladder Window. In the same way, the *Comments* Option in the Variable Area will not be enabled, and it will not be possible to edit comments.

Converting Word Addresses to Variables

There are two input methods. (The differences are given from step 2.)

1,2,3...

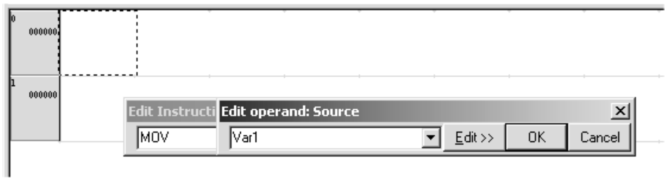
1. In the Ladder Editor, right-click and select an instruction from the Insert Menu. (In this example, Instruction is selected and *MOV* is input.)



MOV

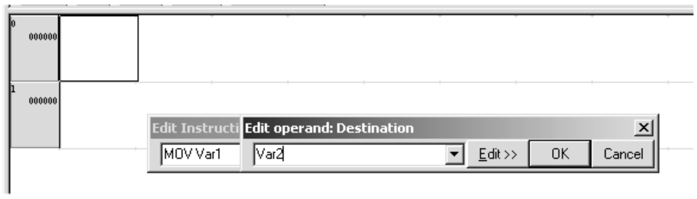
• Press the **Enter** Key to input a comment.

2. Press the **Space** Key.
   The Edit Operand Dialog Box will be displayed. Enter operand 1 and press the **Enter** Key.



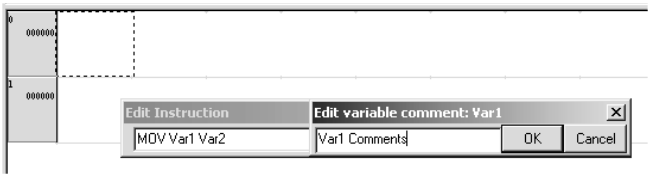
MOV
Var1

3. The Edit Operand Dialog Box will be displayed again. Enter operand 2 and press the **Enter** Key.



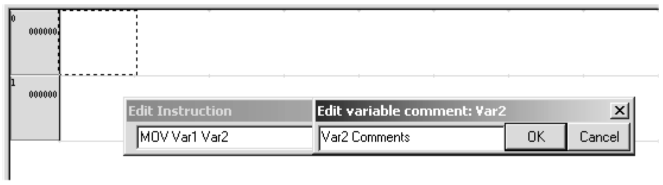
MOV Var1
Var2

4. Enter a comment for operand 1 (for Var1 in this example) and press the **Enter** Key.



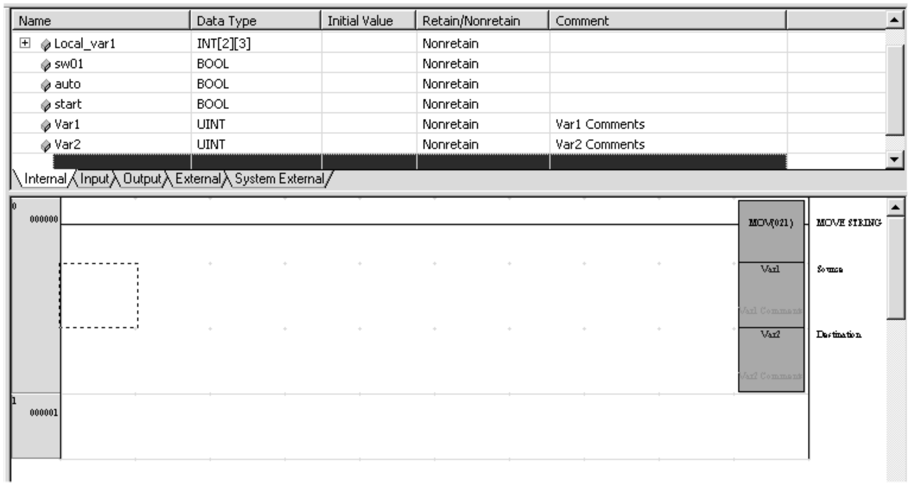
MOV Var1 Var2
Var1 Comments

5. Enter a comment for operand 2 (for Var2 in this example), and press the **Enter** Key or click the **OK** Button.



MOV Var1 Var2
Var2 Comments

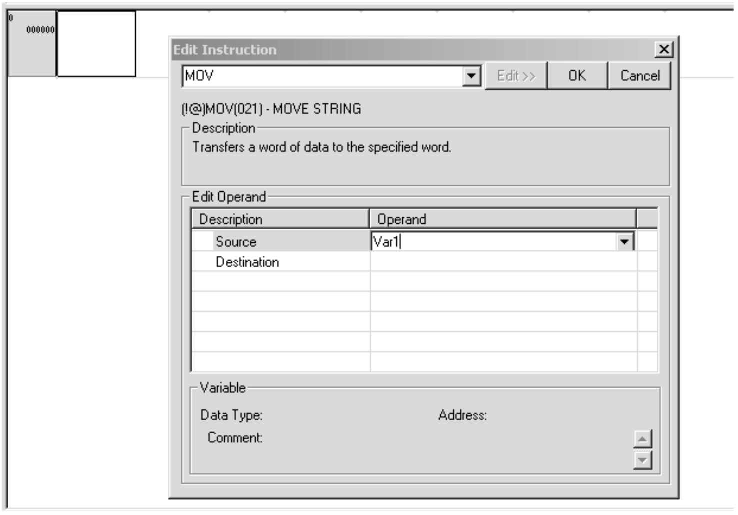
6. Automatically created variable names and input comments will be displayed in the Variable Editor and Ladder Editor.



Name	Data Type	Initial Value	Retain/Nonretain	Comment
Local_var1	INT[2][3]		Nonretain	
sw01	BOOL		Nonretain	
auto	BOOL		Nonretain	
start	BOOL		Nonretain	
Var1	UINT		Nonretain	Var1 Comments
Var2	UINT		Nonretain	Var2 Comments

• Clicking the **Edit** Button

2. Input the variables for each operand in the *Edit Operand* Area. Once the variables for the operands have been set, move the cursor to the field that has an instruction input, and press the **Enter** Key.



Edit Instruction

MOV

(!@)MOV(021) - MOVE STRING

Description

Transfers a word of data to the specified word.

Edit Operand

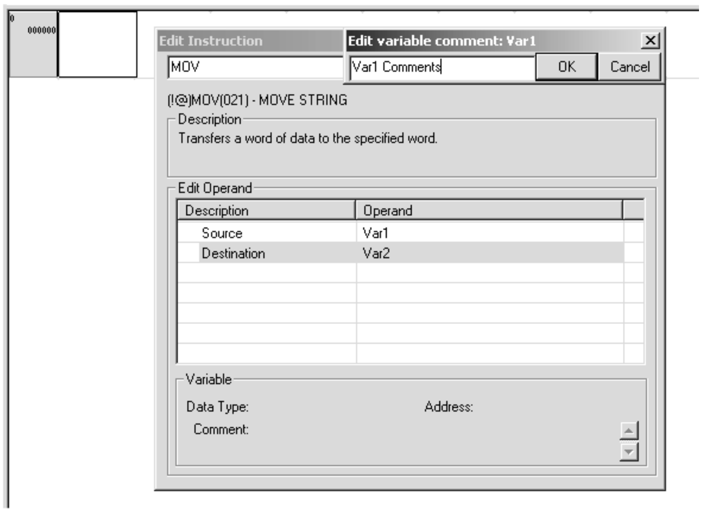
Description	Operand
Source	Var1
Destination	

Variable

Data Type: Address:

Comment:

3. The Edit Variable Comment Dialog Box will be displayed. Enter a comment for the variable.



MOV

Var1 Comments

(!@)MOV(021) - MOVE STRING

Description

Transfers a word of data to the specified word.

Edit Operand

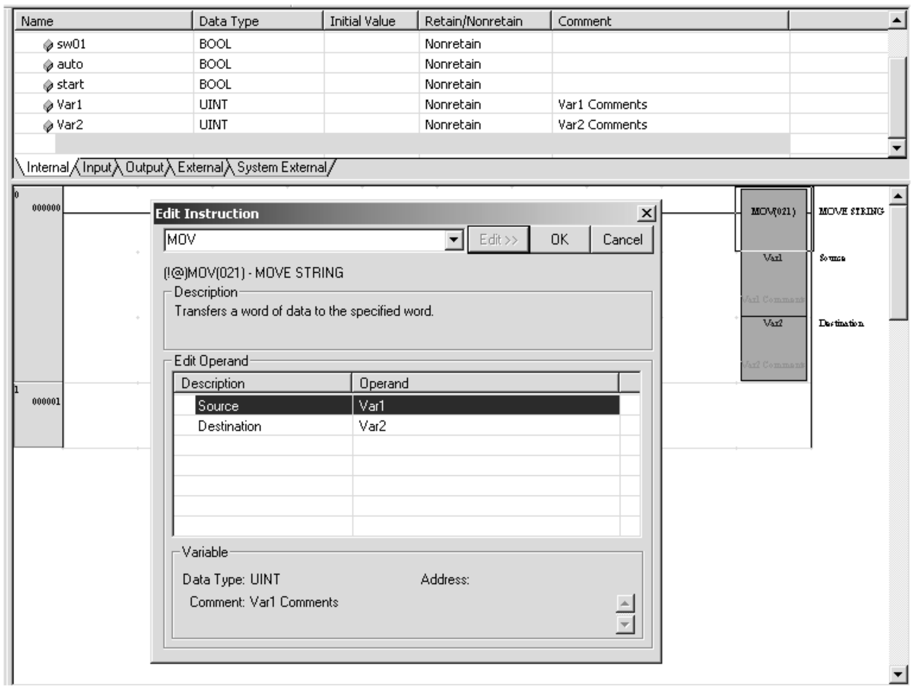
Description	Operand
Source	Var1
Destination	Var2

Variable

Data Type: Address:

Comment:

4. Automatically created variable names and input comments will be displayed in the Variable Editor and Ladder Editor.



Name	Data Type	Initial Value	Retain/Nonretain	Comment
sw01	BOOL		Nonretain	
auto	BOOL		Nonretain	
start	BOOL		Nonretain	
Var1	UINT		Nonretain	Var1 Comments
Var2	UINT		Nonretain	Var2 Comments

MOV

(!@)MOV(021) - MOVE STRING

Description

Transfers a word of data to the specified word.

Edit Operand

Description	Operand
Source	Var1
Destination	Var2

Variable

Data Type: UINT Address:

Comment: Var1 Comments

5-3-8 Programming in Standard Text Language

Character Set

The character set for characters other than those for identifiers used in programs conforms to IEC 61131-3.

- Characters are not case-sensitive.
- The character set conforms to the *Basic Code Table* in ISO 646.
- Keywords can combine uppercase and lowercase letters (e.g., iF or if).

Identifiers

Overview

Identifiers are text strings used to express the following language elements.

- Naming program control units
- Naming I/O and variables
- Naming functions and function blocks

Text Allowed for Identifiers

The same characters and the same number of characters as used for variables in ladder diagram programming can be used for identifiers in ST programming.

Note Refer to *2-2 Variables* for detailed variable specifications.

Restrictions

The following restrictions apply to identifiers.

- The first character must not be a number.
- Two underscores (_) must not be used consecutively.
- Spaces cannot be used.
- In addition to the restrictions on characters and number of characters for variables in ladder programming, reserved words for ST language programming cannot be used. Refer to *ST Language Reserved Words* on page 225 for details on reserved words for ST language programming.

Data Types

Basic Data Types

The basic data types and their sizes are listed in the following table.

Variables are edited with the Variable Editor. Variables can be registered with the ST Editor, but they cannot be changed.

Data type	Meaning	Size	Words allocated	Description
INT	Integer	16 bits	1 word	(–32768 to +32767)
DINT	Double-word integer	32 bits	2 words	(–2147483648 to 2147483647) Word 0: Lower 16 bits Word 1: Upper 16 bits
UINT	Unsigned integer	16 bits	1 word	(0 to 65535)
UDINT	Unsigned double-word integer	32 bits	2 words	(0 to 4294967295) Word 0: Lower 16 bits Word 1: Upper 16 bits
BOOL	Bit string of 1 bit	1 bit	1 bit	1 or 0
WORD	Bit string of 16 bits	16 bits	1 word	
DWORD	Bit string of 32 bits	32 bits	2 words	Word 0: Lower 16 bits Word 1: Upper 16 bits
REAL	Real number	32 bits	2 words	Conforms to IEEE754.
STRING	Text string	64 words	64 words	

Basic Data Types That Are Not Supported

The following basic data types are not supported.

Data type	Meaning
SINT	8-bit integer
USINT	Unsigned 8-bit integer
LINT	64-bit integer
ULINT	Unsigned 64-bit integer
LREAL	Double-word real number
BYTE	Bit string of 8 bits
LWORD	Bit string of 64 bits

Declaring Variables and Data Structures

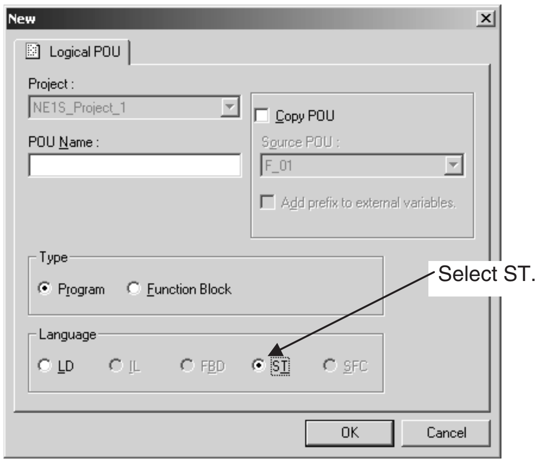
Variables and data structures are edited with the Variable Editor and Structure Editor. Direct address (AT) specifications and variable parameters (e.g., retain/nonretain) are set with the Variable Editor.

Variables and data structures cannot be declared (VAR - END_VAR) in the ST Editor.

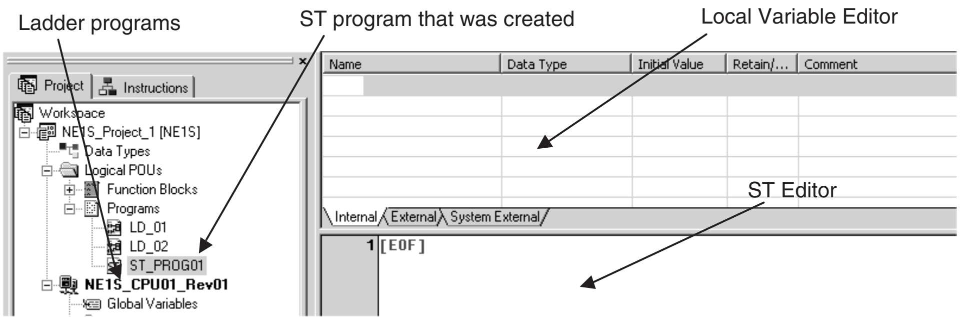
Creating an ST Program

1,2,3...

1. Right-click the ***Logical POUs*** in the Project Window and select ***Create POU***. Alternately, select ***File - New - POU***.
   The following New Dialog Box will be displayed.



2. Input the program name or function name in the *POU Name* field. In this example, *ST_PROG01* is used.
3. For *Type*, select ***Program*** or ***Function Block***. In this example *Program* is selected.
4. For *Language*, select ***ST***.
5. Click the **OK** Button.
   The logical POUs that are created will be displayed under *Logical POUs* in the Project Window. Also, an ST Editor will be displayed for the program or function block that was created.



   This completes creating a program.
6. Finally, code the program.
   Refer to *Appendix B* for operators and conditional statements.
   Refer to page 103 in *5-3-1* for procedures to use the Local Variable Editor.

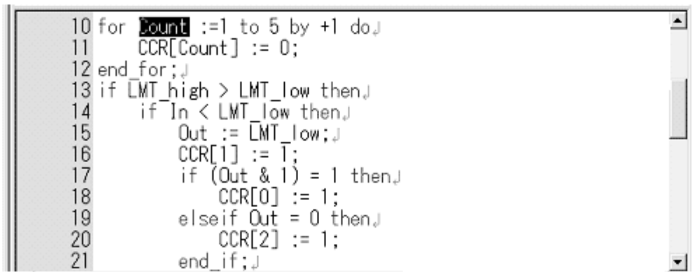
Note After coding a program or function block, always select ***File - Save Changes to project*** to save the changes to the project. Alternately, press the **Ctrl+Shift+S** Keys.

Registering Variables from the ST Editor

Variables and data structures are edited with the Variable Editor and Structure Editor, but variables can be registered from the ST Editor. Use the following procedure.

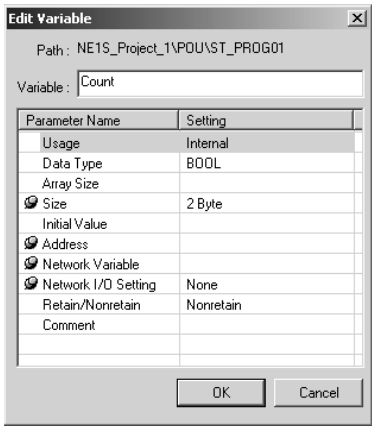
1,2,3...

1. Select the variable in the ST Editor as shown below by clicking and dragging.



```
for Count :=1 to 5 by +1 do
    CCR[Count] := 0;
end_for;
if LMT_high > LMT_low then
    if In < LMT_low then
        Out := LMT_low;
        CCR[1] := 1;
        if (Out & 1) = 1 then
            CCR[0] := 1;
        elseif Out = 0 then
            CCR[2] := 1;
        end_if;
```

2. Right-click and select ***Add Variable***.
   The following Edit Variables Dialog Box will be displayed.



Edit Variable

Path : NE1S_Project_1\POU\ST_PROG01

Variable : Count

Parameter Name	Setting
Usage	Internal
Data Type	BOOL
Array Size	
Size	2 Byte
Initial Value	
Address	
Network Variable	
Network I/O Setting	None
Retain/Nonretain	Nonretain
Comment	

3. Set the parameters and then click the **OK** Button.
   The variable will be registered in the variable editor.

   Refer to page 103 in *5-3-1* for procedures to use the Edit Variables Dialog Box and the Variable Editor.

5-4 Creating Function Blocks and Pasting Them into Programs

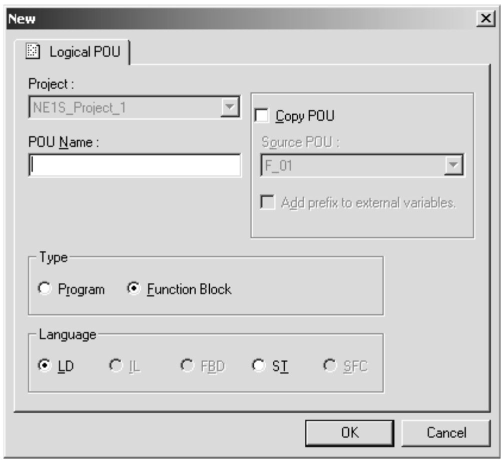
This section describes basic methods for creating function blocks.

Refer to *2-3* for information on the function of function blocks.

5-4-1 Creating Function Blocks

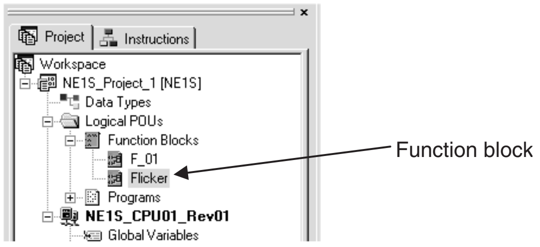
1,2,3...

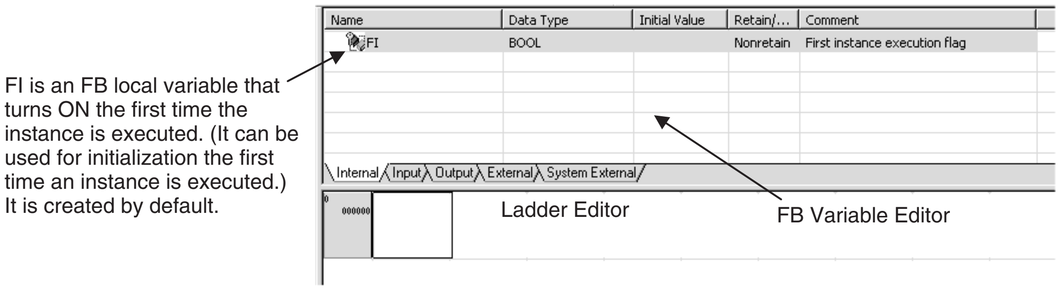
1. Right-click the ***Logical POUs*** in the Project Window and select ***Create POU***. Alternately, select ***File - New - POU***.
   The following New Dialog Box will be displayed.



2. Input the function name in the *POU Name* field. In this example, *Flicker* is used.
3. For *Type*, select ***Function Block***.

   Note If *Program* is selected a new program will be created.
4. Click the **OK** Button.
   The function block will be created and displayed in the Workspace as shown below. Also, a Ladder Editor and FB Variable Editor will be displayed for the function block that was created.





Name	Data Type	Initial Value	Retain/...	Comment
FI	BOOL		Nonretain	First instance execution flag

FI is an FB local variable that turns ON the first time the instance is executed. (It can be used for initialization the first time an instance is executed.) It is created by default.

This completes creating a function block.

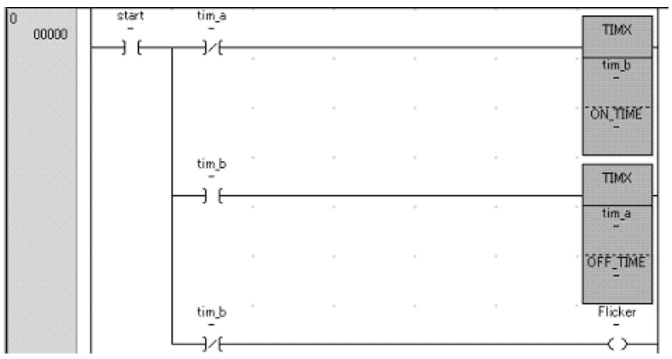
5-4-2 Programming a Function Block

The procedures for programming the contents of a function block are the same as those described in *5-3 Programming Methods*.

Note Refer to *2-3 Function Blocks* for information on the function of function blocks.

1,2,3...
1. As an example, ladder diagram program will be used to create a function block for the following flicker rung.

Variable name	Type	Data type
tim_a	VAR	TIMER
tim_b	VAR	TIMER
ON_TIME	VAR_INPUT	UINT
OFF_TIME	VAR_INPUT	UINT
start	VAR_INPUT	BOOL
Flicker	VAR_OUTPUT	BOOL

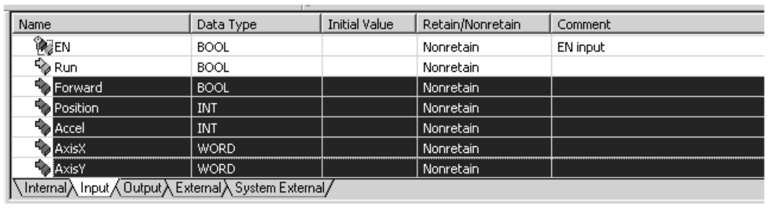


2. Select ***File - Save Change to project***. Alternately, press the **Ctrl+Shift+S** Keys.
   The function block will be saved in the project.
   A program cannot be assigned to a function block unless the function block has been saved in the project.

Reference Information: Grouping Input Variables or Output Variables for Function Blocks

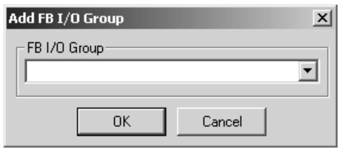
When a function block is pasted into a program with the procedure described later in this section, the program can be very difficult to read if there are too many input variables in the operand input box or too many output variables in the output operand box. The following procedure can be used to simplify the display and use one input variable or one output variable to represent all of the input or output variables.

1,2,3...
1. On the Input Tab Page or Output Tab Page of the Local Variable Editor, select the variables to be grouped.
   - The following example shows a selection to group input variables.

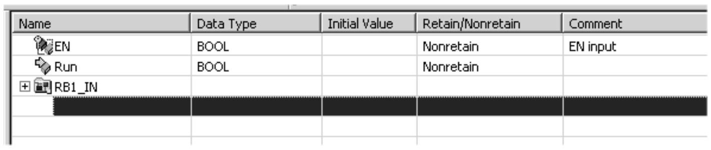


Name	Data Type	Initial Value	Retain/Nonretain	Comment
EN	BOOL		Nonretain	EN input
Run	BOOL		Nonretain	
Forward	BOOL		Nonretain	
Position	INT		Nonretain	
Accel	INT		Nonretain	
AxisX	WORD		Nonretain	
AxisY	WORD		Nonretain	

2. Right-click the variables that were selected and select ***Group Input/Output Variables - Group***.
   The Add FB I/O Group Dialog Box will be displayed.

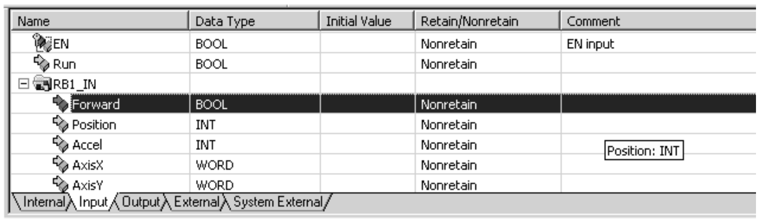


3. Input the name of the I/O group and then click the **OK** Button.
   As shown below, the variables have been grouped under the specified I/O group name.
   - The following example shows a group called RB1_IN.



Name	Data Type	Initial Value	Retain/Nonretain	Comment
EN	BOOL		Nonretain	EN input
Run	BOOL		Nonretain	
⊞ RB1_IN				

   - To ungroup the variables, right-click the I/O group name and select ***Group Input/Output Variables - Release Group***.
   - To delete elements from the group, open the group folder, right-click the element, and then select ***Release Member*** from the ***Group Input/Output Variables*** Menu. To change the display order of the group elements, right-click the element, and then select ***Up***, or ***Down***.



Name	Data Type	Initial Value	Retain/Nonretain	Comment
EN	BOOL		Nonretain	EN input
Run	BOOL		Nonretain	
⊟ RB1_IN				
Forward	BOOL		Nonretain	
Position	INT		Nonretain	
Accel	INT		Nonretain	Position: INT
AxisX	WORD		Nonretain	
AxisY	WORD		Nonretain	

Reference Information: Changing Variable Types

Internal variables can be changed to input, output, or external variables, and output, input, or external variables can be changed to internal variables. This is performed on the Ladder Editor.

The following example shows how to change an internal variable to an input variable.

Right-click the internal variable to be changed on the Ladder Editor and select ***Change Variable Usage - Input***.

The variable type will be changed from an internal variable to an input variable. When the variable type is change, the variable will be removed from the Internal Tab Page of the Variable Editor and placed on the Input Tab Page.

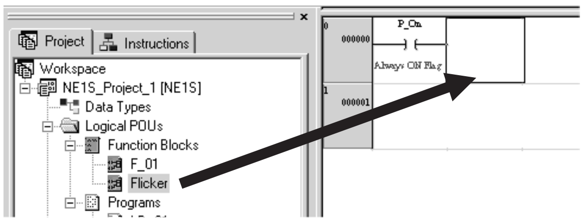
5-4-3 Pasting Function Blocks into Programs

Use the following procedure to paste a function block into a program.

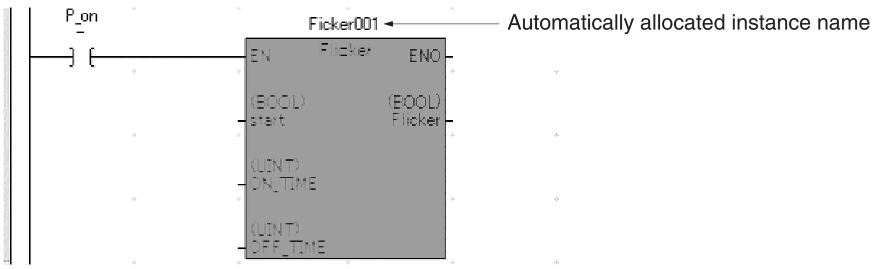
1,2,3...

1. Drag the function block from the Workspace and drop it at the insertion point in the ladder diagram. When using one-key input mode, the function block can be inserted by pressing the **F** Key.

   If the insertion point is selected and clicked in Function Block Mode (entered by selecting ***Ladder - Mode - Function Block***), the Edit Function Block Dialog Box will be displayed. Select the function block, input the instance name, and click the **OK** Button.

   The function block will be inserted into the ladder diagram and displayed as shown below.

   Automatically Generated Instance Variables

   If the *Create Function Block Instances Automatically* setting is enabled (default) in the Edit Tab of the Ladder Window under *Tool - Option*, instance variables will be automatically generated and an instance name will be automatically displayed, as shown in the above diagram. The instance name consists of the FB body name + _XXX (where XXX indicates a serial number starting from 001). When the *Create Function Block Instances Automatically* setting is enabled (default), the procedure described in step 3 is not required.

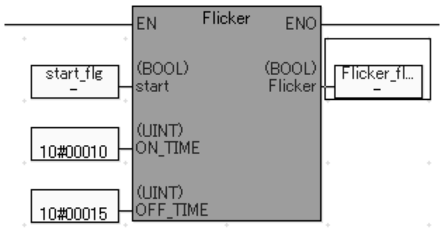
2. Input the function block operands.
   - Method 1:
     Double-click the operand input position (or select it and press the **Enter** Key) to display the Edit Function Block Argument Dialog Box. Input the function block operands in this dialog box.
   - Method 2:
     It is also possible to select the input position and input the value directly from the keyboard, or drag and drop the variable from the Variable Editor to the operand.

Note

(1) Addresses can also be specified for operands, but data types and sizes will not be checked. Always use the proper data type so that data is not corrupted.

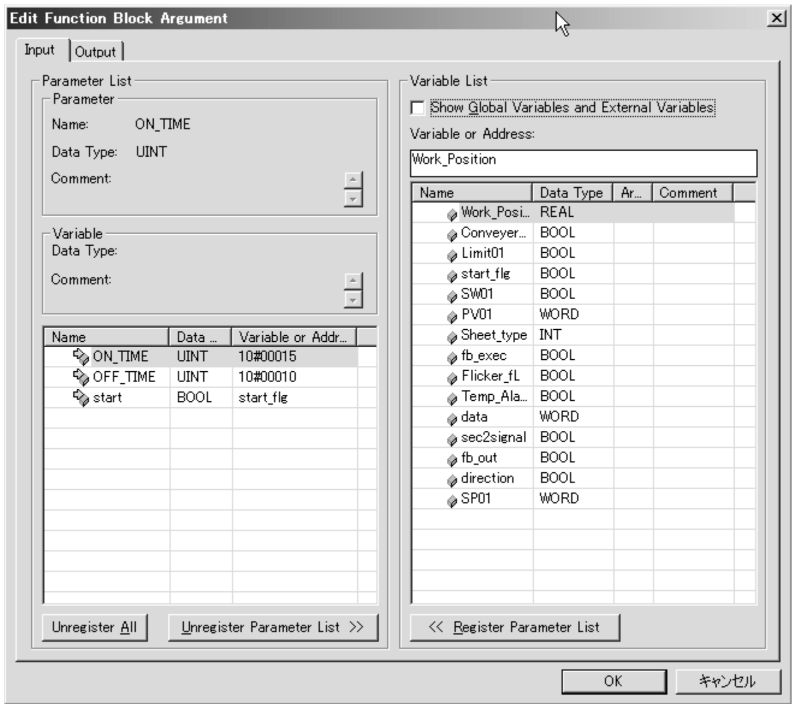
(2) Refer to *5-3-4 Inputting Constants* and *5-3-5 Inputting Operand Numbers* for the methods for inputting constants and operand numbers.

• Input Example



Operand Input Method 1

a. Double-click the function block operand input position to display the Edit Function Block Argument Dialog Box.


Edit Function Block Argument
Input
Output
Parameter List
Parameter
Variable
Variable List
Show Global Variables and External Variables
Unregister All
Unregister Parameter List >>
<< Register Parameter List
OK
キャンセル

Name: ON_TIME
Data Type: UINT
Comment:

Variable Data Type:
Comment:

Name	Data ...	Variable or Addr...
ON_TIME	UINT	10#00015
OFF_TIME	UINT	10#00010
start	BOOL	start_flg

Variable or Address: Work_Position

Name	Data Type	Ar...	Comment
Work_Posi...	REAL		
Conveyer...	BOOL		
Limit01	BOOL		
start_flg	BOOL		
SW01	BOOL		
PV01	WORD		
Sheet_type	INT		
fb_exec	BOOL		
Flicker_fL	BOOL		
Temp_Ala...	BOOL		
data	WORD		
sec2signal	BOOL		
fb_out	BOOL		
direction	BOOL		
SP01	WORD		

b. In the *Parameter List* area, select the desired variable in the function block. The selected variable will be highlighted in gray. (In this example, select an input variable such as *start* in the Input Tab Page. Select the output variable *Flicker* in the Output Tab Page.)

c. In the *Variable List* at the upper-right, select the variable that you want to be the input source and click the **Register Parameter List** Button. (In this example, *start_flg* is registered to input variable *start*.)

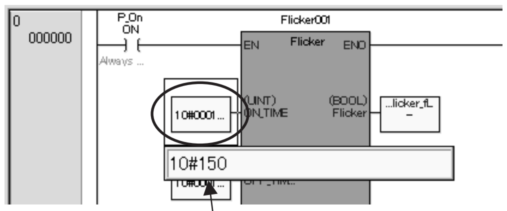
If you want to input a constant, input the constant directly in the *Variable or Address* Field. (In this example, *10#00010* is input directly for input variable *ON_TIME* and *10#00015* is input directly for input variable *OFF_TIME*.

In the *Variable List* at the upper-right, select the variable that you want to be the output destination and click the **Register Parameter List** Button. (In this example, *Flicker_fL* is registered to output variable *Flicker*.)

Operand Input Method 2

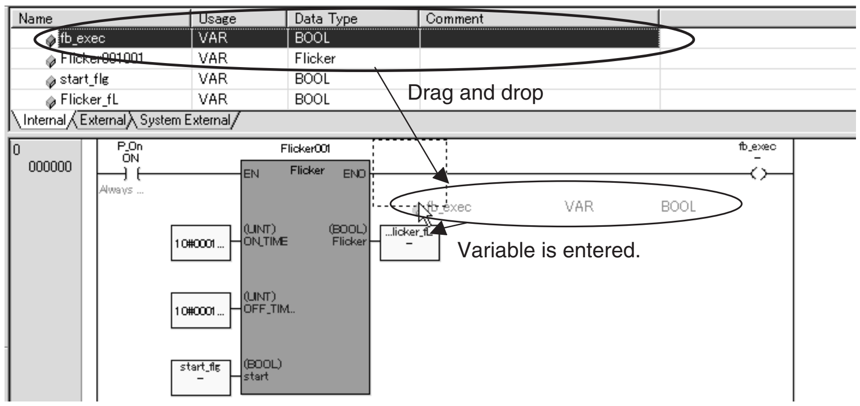
Direct Input

Select the position where you want to input the function block operand and directly input the value from the keyboard (in mnemonic input mode only).



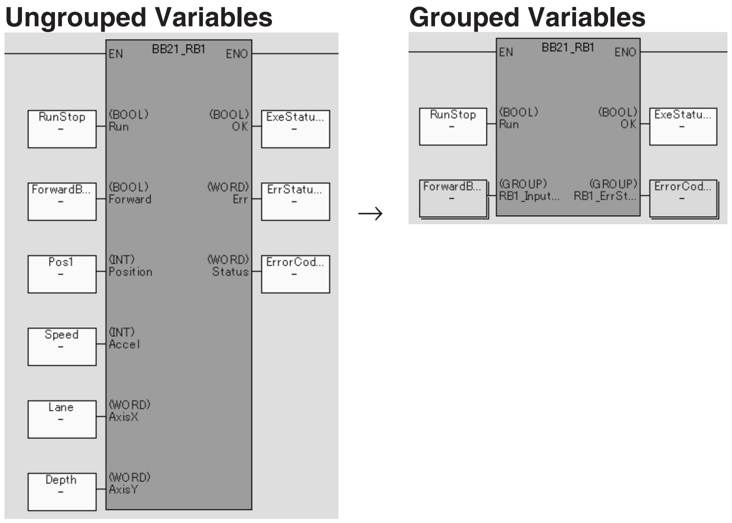
Dragging and Dropping a Variable

Drag and drop the variable from the Variable Editor to the desired input position. The data types of the source and destination variables must be the same.



Name	Usage	Data Type	Comment
fb_exec	VAR	BOOL	
Flicker001001	VAR	Flicker	
start_flg	VAR	BOOL	
Flicker_fL	VAR	BOOL	

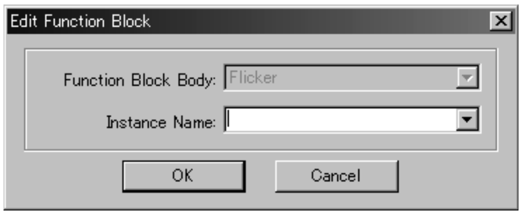
Note If I/O groups have been created for input variables or output variables, the group names will be displayed as shown below when the function block is pasted into a program.



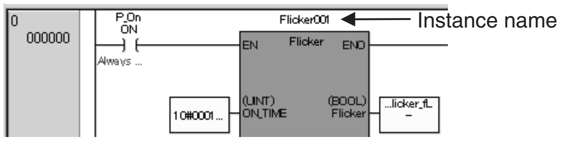
3. Assign an instance name to the function block.

   Note The procedure in step 3 is not required if the *Create Function Block Instances Automatically* setting is enabled (default) in the Edit Tab of the Ladder Window under *Tool - Option*. Proceed to step 4.

   • Select the function block in the ladder diagram and select ***Edit Instructions*** from the pop-up menu.The following dialog box will be displayed.



- Input the instance name (*Flicker01* in this example) and then click the **OK** Button. The instance name will be displayed on the function block as shown below.



4. Select ***File - Save Change to project***. Alternately, press the **Ctrl+Shift+S** Keys.
   The program will be saved in the project.
   A program cannot be assigned to a task unless the program has been saved in the project.

This completes pasting a function block.

Note (1) After programming, always select ***File - Save Changes to project***. Alternately, press the **Ctrl+Shift+S** Keys.
An alarm will be displayed if there is an error in the program. Correct the error and then select ***File - Save Change to project*** again.
(2) Read-protection can be set for a function block. Refer to *5-5 Read Protection for Logical POUs* for the procedure.

5-4-4 Editing the Function Block Body after Pasting

Changes to a function block definition itself (editing of input variables or output variables) can be reflected in an instance even if the function block body was already pasted as an instance. Function block instances can be updated by selecting ***Edit - Update Function Block Instance***.

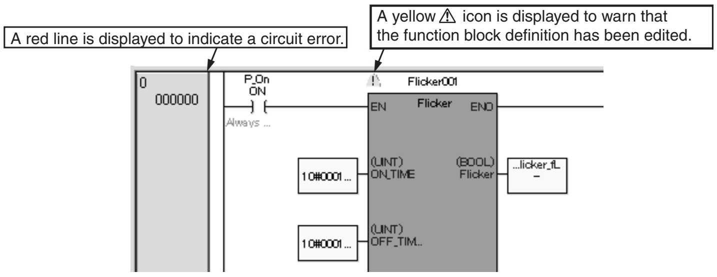
With this menu command, it is no longer necessary to delete and repaste instances in the program after editing the function block definition itself. It is also unnecessary to input the argument/operand values again.

Procedure

Use the following procedure to update function block instances.

1,2,3... 1. After pasting the function block body as an instance, edit the function block body's input variables or output variables (change, add, or delete) and save the project.
   - A confirmation dialog box will be displayed, indicating the following: *Function block was modified. Please check the output window.* At the same time, the Find Tab Page in the Output Window will show the program addresses of that function block's instances.

- A yellow ⚠: icon will be displayed at the upper-left corner of instances in the program, warning that there has been a change, and a circuit error will occur.


A red line is displayed to indicate a circuit error.
A yellow ⚠ icon is displayed to warn that the function block definition has been edited.
0
000000
P_On
ON
Always ...
Flicker001
EN
Flicker
ENO
10#0001...
(UINT) ON_TIME
(BOOL) Flicker
...licker_fL
10#0001...
(UINT) OFF_TIM..

2. Either select ***Edit - Update Function Block Instance*** or select the instance, right-click, and select ***Update Function Block Instance*** from the popup menu. The changes to the function block definition will be reflected in the program's instances and the yellow warning icons and circuit errors will be cleared.

⚠Caution Be sure to thoroughly check the arguments of locations used (marked with a ⚠) if input variables or output variables are increased or decreased or if the order is changed.

5-4-5 Moving to the Function Block Body (Internal)

Moving to the Function Block Body Use the following procedure to move the cursor to the body of the function block body selected with the Ladder Editor.

1,2,3...
1. Click the function block instance in the Ladder Editor.
2. Select ***View - To Lower Layer***. (Alternatively, double-click, press the **Enter** Key, or click the toolbar icon.) The cursor will move to the corresponding function block body.

Note If the operation is performed by clicking the input variables (or output variables) of the function block instance, the cursor will automatically move to the position in the program where the variable is used (Ladder Editor only). This function can be disabled using a setting under ****Tool - Options****.

Moving to the Location That Called the Function Block

Use the following procedure to move the cursor from the function block body to the location that called the function block.

Select ***View - To Upper Layer***. (Alternatively, click the toolbar icon.) The cursor will move to the function block instance of the call source.

5-5 Read Protection for Logical POUs

Passwords can be set to set read protection for logical POUs in function blocks or programs.

This function is also enabled offline.

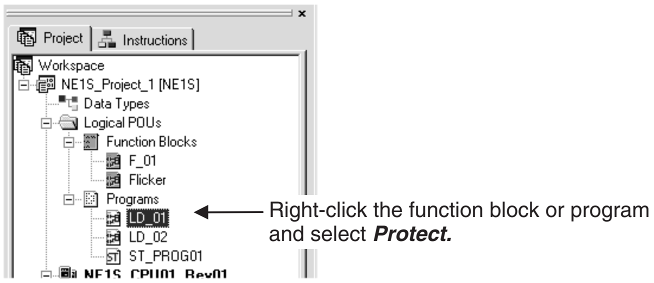
If read protection is set, a function block cannot be displayed or edited without inputting the password.

Printing can be performed only if Read Protection is OFF or is ON but temporarily disabled.

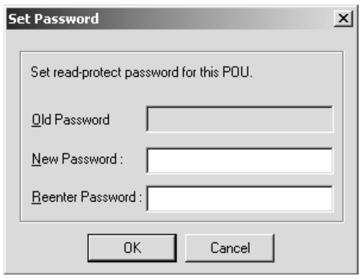
Note Read-protection is ON but temporarily disabled when a read-protected function block is displayed after inputting the correct password.

1,2,3...

1. Right-click the function block or program to be read-protected and select **Protect** from the popup menu.



The following Set Password Dialog Box will be displayed.


Set Password

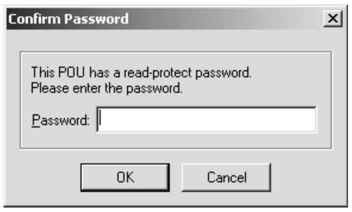
Set read-protect password for this POU.

Old Password

New Password :

Reenter Password :

2. Input the password and then click the **OK** Button.
   A password will be set for the function block specified in step 1. After the password is set, the function block cannot be displayed or edited without inputting the password.
3. The following message will be displayed when you attempt to open a function block or program for which a password has been set. Input the password to temporarily release the protection and perform the editing.


Confirm Password

This POU has a read-protect password.
Please enter the password.

Password:

Display Icons

The three display icons for function blocks are shown in the following table.

POU type	Protection OFF	Protection ON	Protection ON but temporarily disabled
Function block	▦	🔒▦	🔓▦
Program	▦	🔒▦	🔓▦

Clearing Read Protection

When protection is temporarily disabled, input the current password in the Old Password Field, input an empty (i.e., blank) password in the New Password Field, and click the **OK** Button. The set read protection will be cleared.

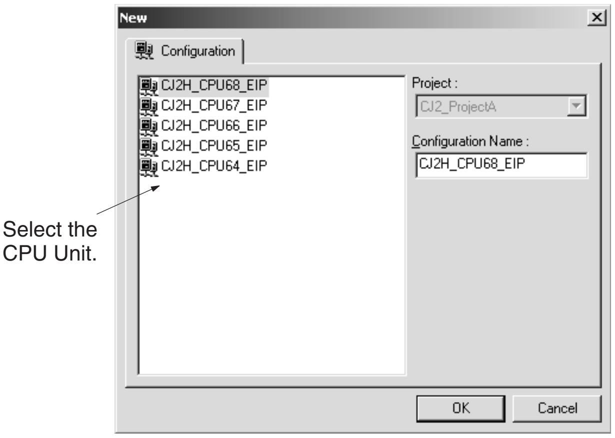
5-6 Creating Configurations and Assigning Programs to Tasks

5-6-1 Creating a Configuration

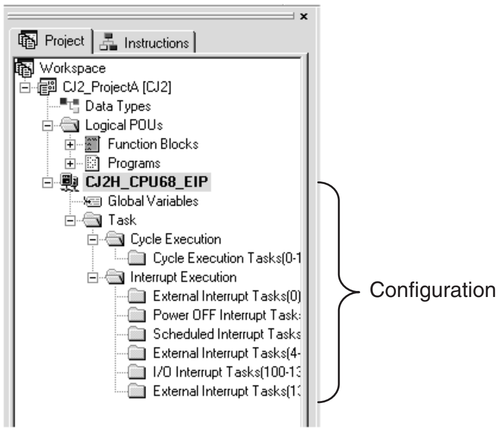
Use the following procedure to create a new configuration.

1,2,3...

1. 1 Right-click the project name in the Workspace and select ***Create Configuration***. Alternately, select ***File - New - Configuration***.
   The following New Dialog Box will be displayed.



2. Select the CPU Unit.
   The CPU Units displayed here are the CPU Units for the series selected when the project was created. (The figure above applies to a project with a CJ2 CPU Unit.)
3. Input the configuration name and then click the **OK** Button. The new configuration will be displayed in the Workspace.

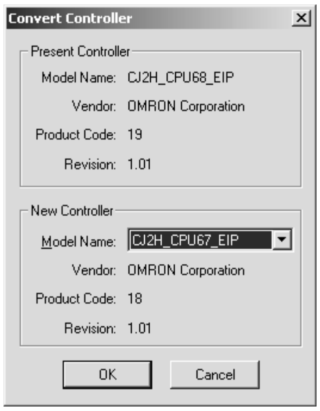


This completes creating a configuration.

Changing the CPU Unit Type (PLC Type)

1,2,3...

1. To change the type of CPU Unit, select ***Controller*** – ***Convert Controller***. The following Convert Controller Dialog Box will be displayed.

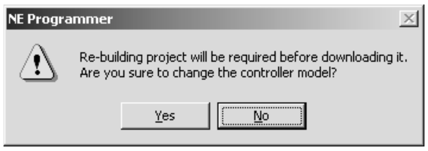


Convert Controller

Present Controller
Model Name: CJ2H_CPU68_EIP
Vendor: OMRON Corporation
Product Code: 19
Revision: 1.01

New Controller
Model Name: CJ2H_CPU67_EIP
Vendor: OMRON Corporation
Product Code: 18
Revision: 1.01

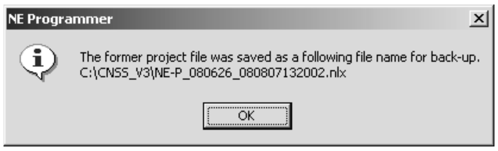
2. Select the Controller from the list box and click the **OK** Button.
3. The following confirmation message will be displayed. The **Yes** Button.



NE Programmer

Re-building project will be required before downloading it.
Are you sure to change the controller model?

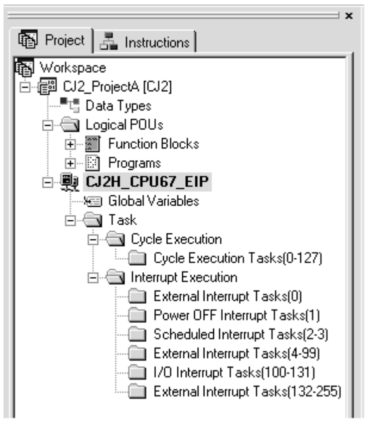
4. A backup file of the project before converting the type of CPU Unit will be automatically created with the name "previous project name_0xxxxxxxxxxx.nlx". The Xs in the filename are numbers for the year, month, day, hour, minutes, and seconds.



NE Programmer

The former project file was saved as a following file name for back-up.
C:\CNSS_V3\NE-P_080626_080807132002.nlx

5. The CPU Unit specified as the type of CPU Unit for the new controller will be set.

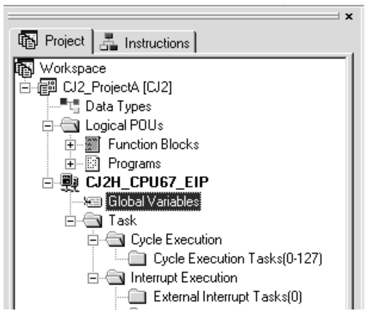


Project
Instructions
Workspace
CJ2_ProjectA [CJ2]
Data Types
Logical POUs
Function Blocks
Programs
CJ2H_CPU67_EIP
Global Variables
Task
Cycle Execution
Cycle Execution Tasks(0-127)
Interrupt Execution
External Interrupt Tasks(0)
Power OFF Interrupt Tasks(1)
Scheduled Interrupt Tasks(2-3)
External Interrupt Tasks(4-99)
I/O Interrupt Tasks(100-131)
External Interrupt Tasks(132-255)

5-6-2 Creating and Editing Global Variables

Displaying the Global Variable Editor

1,2,3... 1. Create a new configuration (*File - New - Configuration*).
The global variables will be displayed in the Workspace.

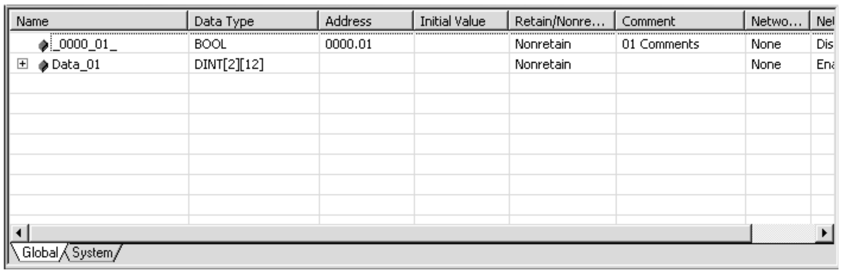


Project | Instructions

```
Workspace
└ CJ2_ProjectA [CJ2]
  ├ Data Types
  ├ Logical POUs
  │ ├ Function Blocks
  │ └ Programs
  └ CJ2H_CPU67_EIP
    ├ Global Variables
    └ Task
      ├ Cycle Execution
      │ └ Cycle Execution Tasks(0-127)
      └ Interrupt Execution
        └ External Interrupt Tasks(0)
```

2. Double-click ***Global Variables*** in the Workspace.
The Global Variable Editor will be displayed.

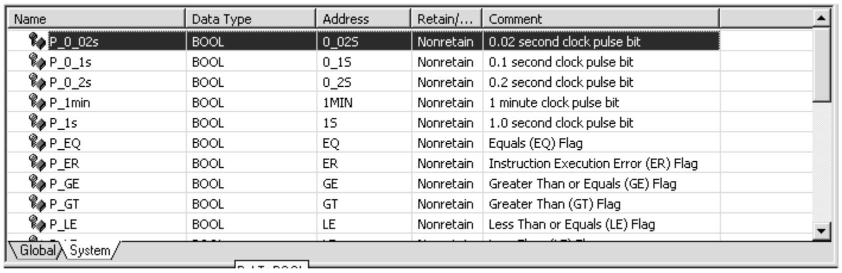
■ **Global Sheet**



Name	Data Type	Address	Initial Value	Retain/Nonre...	Comment	Netwo...	Ne
_0000_01_	BOOL	0000.01		Nonretain	01 Comments	None	Dis
Data_01	DINT[2][12]			Nonretain		None	En

Global / System

■ **System Sheet**



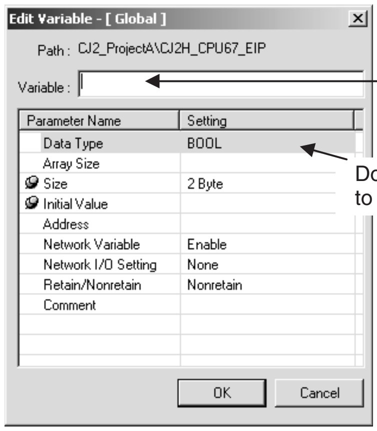
Name	Data Type	Address	Retain/...	Comment
P_0_02s	BOOL	0_02S	Nonretain	0.02 second clock pulse bit
P_0_1s	BOOL	0_1S	Nonretain	0.1 second clock pulse bit
P_0_2s	BOOL	0_2S	Nonretain	0.2 second clock pulse bit
P_1min	BOOL	1MIN	Nonretain	1 minute clock pulse bit
P_1s	BOOL	1S	Nonretain	1.0 second clock pulse bit
P_EQ	BOOL	EQ	Nonretain	Equals (EQ) Flag
P_ER	BOOL	ER	Nonretain	Instruction Execution Error (ER) Flag
P_GE	BOOL	GE	Nonretain	Greater Than or Equals (GE) Flag
P_GT	BOOL	GT	Nonretain	Greater Than (GT) Flag
P_LE	BOOL	LE	Nonretain	Less Than or Equals (LE) Flag

Global / System

System global variables are registered in advance in the System Sheet. System global variables cannot be changed.

Adding Global Variables

1,2,3... 1. Right-click on the Global Variable Sheet and select ***Add*** from the pop-up menu. Alternately, double-click an empty row.
The Edit Variables Dialog Box will be displayed.



Path : CJ2_ProjectA\CJ2H_CPU67_EIP

Variable :

Parameter Name	Setting
Data Type	BOOL
Array Size	
Size	2 Byte
Initial Value	
Address	
Network Variable	Enable
Network I/O Setting	None
Retain/Nonretain	Nonretain
Comment	

2. Input the variable name.
3. Set the variable parameters.
4. When all settings have been completed, click the **OK** Button. Parameters are displayed as described in the following table.

Note Refer to *2-2* for detailed variable specifications.

Parameter	Meaning	Values
Path	Displays the valid scope of the variable.	Global variables: Project_name/ configuration_name
Data Type	Set the data type of the variable.	BOOL, INT, UINT, DINT, UDINT, WORD, DWORD, TIMER, COUNTER, STRING, REAL, or user-defined
Array Size	Set the number of elements for a one-dimensional array or a two-dimensional array. Two-dimensional arrays can be set only for CJ2 CPU Units.	When not specifying an array, leave this settings blank for both a one-dimensional and two-dimensional array. When specifying a one-dimensional array, set the number of array elements for a one-dimensional array and leave the setting for a two-dimensional array blank. When specifying a two-dimensional array, set the number of array elements for a two-dimensional array and leave the setting for a one-dimensional array blank. *A two-dimensional array cannot be specified for the NE1S. *Refer to *2-2-3 Variable Properties* for other restrictions on elements.
Size	Displays the size used by the variable in the memory	---
Initial Value	Set the values for when operation is started. **Note** The initial values of variables cannot be set for CJ2 CPU Units.	Set the initial value of the variable according to the data type. • BOOL, WORD, or DWORD: Unsigned hexadecimal Input the number after “16#”. • INT or DINT: Signed decimal Input the number after “+10#” or “–10#”. • UINT or UDINT: Unsigned decimal Input the number after “10#”. • REAL: Real number (e.g., +1.0, –0.23, +9.8E-3) • STRING: Character string, e.g., “Data”
Address	Set when a specific address is manually set for the variable (AT designation). This setting is supported only for global variables.	

Parameter	Meaning	Values
Network Variable	This property sets whether to convert variables to network variables. It sets whether external access (i.e., reading or writing) is enabled for variable names. This setting can be made only for global variables. It cannot be made for local variables.	Enabled: Variable names can be accessed (i.e., read and written) externally. Disabled: Variable names cannot be accessed (i.e., read and written) externally.
Network I/O Setting	When using cyclic communications, select *Input* when disclosing the variable as an input from the network, select *Output* when disclosing the variable as an output to the network, and select *None* when the variable will not be disclosed.	None: Do not disclose the variable as a connection target. Input: Disclose the variable as an input from the network. Output: Disclose the variable as an output to the network. Global variables that have been set to Input or *Output* can be imported to the Network Configurator after the project has been saved. After the global variables have been imported, variable names can be used in programming to set connections for ControlNet cyclic communications.
Retain/Nonretain	Specify whether to maintain the value of the variable when power is turned OFF and ON, and when operation is started.	Retain or Nonretain
Comment	Input a comment for the variable.	256 characters max.

Note When the Cross Reference Pop-up Window has been displayed (by selecting ***View - Window - Cross Reference Tool***), a variable's cross reference information (program address, instruction name, program name, etc.) can be displayed just by selecting that variable in the Variable Editor.

Editing Global Variables

1,2,3...
1. Double-click the variable to be edited on the Global Sheet. Alternately, right-click the variable and select ***Edit***.
   The Edit Variables Dialog Box will be displayed.
2. Edit the parameters of the Variable and then click the **OK** Button.

Deleting Global Variables

1,2,3...
1. Select the variable to be deleted and press the **Delete** Key.
2. A dialog box will appear to confirm the deletion. Click the **Yes** Button.
   The variable will be deleted.

5-6-3 Pasting Programs into Tasks

Tasks control the timing of program execution. Logical POU programs are assigned to tasks to execute them.

There are two types of tasks: cyclic tasks and interrupt tasks (power OFF interrupt, scheduled interrupt, I/O interrupt, and external interrupt tasks). Extra cyclic tasks are not supported.

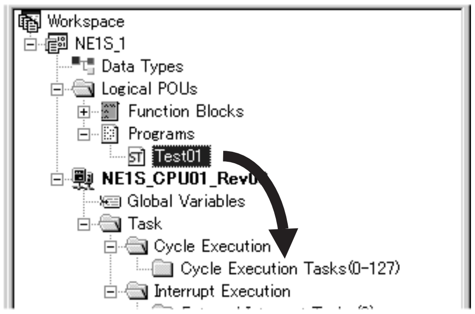
Up to 128 cyclic tasks (task numbers 0 to 127) can be used. When specifying the task number in the operand of a TASK ON (TKON) or TASK OFF (TKOF) instruction, the task number can be specified directly by inputting the number in decimal after "10#" or it can be specified indirectly by inputting an I/O memory address containing the task number.
If TKONα is used, set the value of variable a to the task number.

Example: When the instruction "TKON 3" is input, the task number is the value in CIO 0003.

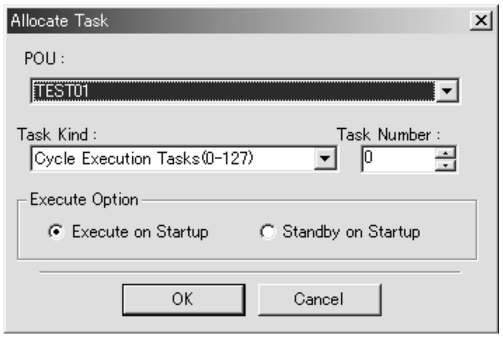
Use the following procedure to paste a program into a task.

1,2,3...

1. Drag the program from the Workspace and drop it on the task folder. In this example, the program TEST01 is assigned to *Cycle Execution Task (0-127)*.

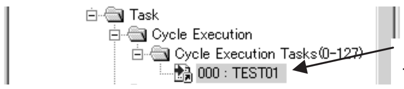


The following Allocate Task Dialog Box will be displayed.



POU : TEST01
Task Kind : Cycle Execution Tasks(0-127)
Task Number : 0
Execute Option: [✓ Execute on Startup] Standby on Startup

2. Set the *Execute Option* and *Task Number* (these are automatically set if the program is dragged and dropped), and then click the **OK** Button.
   The program will be assigned to the task and displayed as shown below.



Program assigned to a task.

Assigning a program to a task makes it possible to execute the task.

Note Logical POUs and tasks can be switched in the workspace. The window can be switched by selecting ***Window - Toggle POU/Task Window*** or by pressing **Alt** Key and **F6** Key. (Function blocks, however, can be switched from tasks to logical POUs only.)
The window can also be changed when the Controller is connected online.

5-6-4 Checking External Variables for Consistency

The NE Programmer can check whether there are any inconsistencies between the configuration's global variables and the external variables of the logical POUs assigned to that configuration as tasks. Inconsistencies can occur for either of the following two reasons.

1. Variables with the same name exist in both programs, but the data types or sizes do not match.
2. External variables exist, but global variables do not exist.

If there is an inconsistency, a build error will occur.

The External Variable Consistency Check function checks for the two inconsistencies described above and helps correct the inconsistencies if any are detected.

1,2,3...

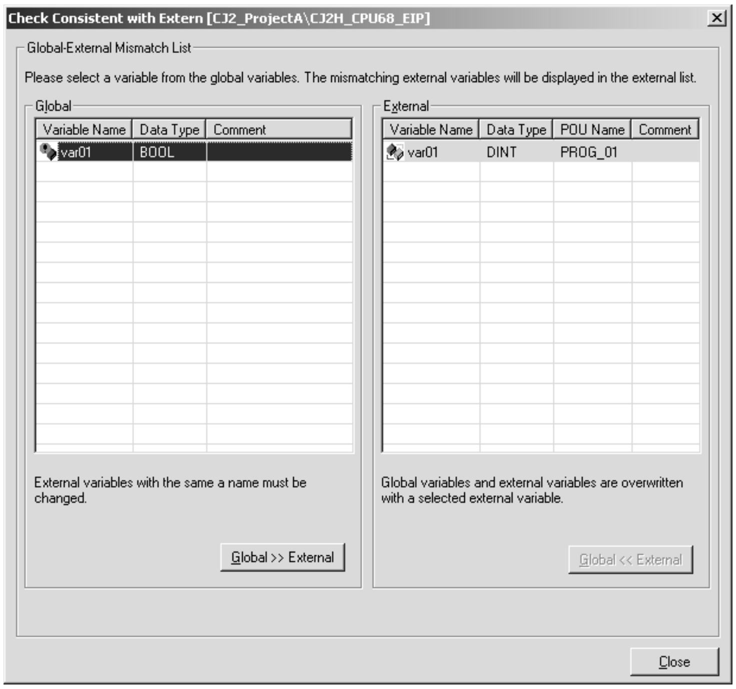
1. Either select ***Variable - Check Consistent with Extern*** from the Menu Bar or select ***Check Consistent with Extern*** from the popup menu in the Variable Window or Program Window.

   If no inconsistencies are found, the message *A problem is not in adjustment* will be displayed. In this case, click the **OK** Button to continue.
2. If any inconsistencies are found, the following Check Consistent with Extern Dialog Box will be displayed.

   a. Unifying to the Global Variable's Data Definition

      Select the variable in the *Global* variable mismatch list and click the **Global >> External** Button. Click the **Yes** Button in the confirmation dialog box that is displayed.

   Note For consistency, the external variable's comment is overwritten by the selected global variable's comment.



Check Consistent with Extern [CJ2_ProjectA\CJ2H_CPU68_EIP]

Global-External Mismatch List

Please select a variable from the global variables. The mismatching external variables will be displayed in the external list.

Global

Variable Name	Data Type	Comment
var01	BOOL	

External variables with the same a name must be changed.

External

Variable Name	Data Type	POU Name	Comment
var01	DINT	PROG_01	

Global variables and external variables are overwritten with a selected external variable.

   b. Unifying to the External Variable's Data Definition

      Select the variable in the *External* variable mismatch list and click the **Global << External** Button. Click the **Yes** Button in the confirmation dialog box that is displayed.

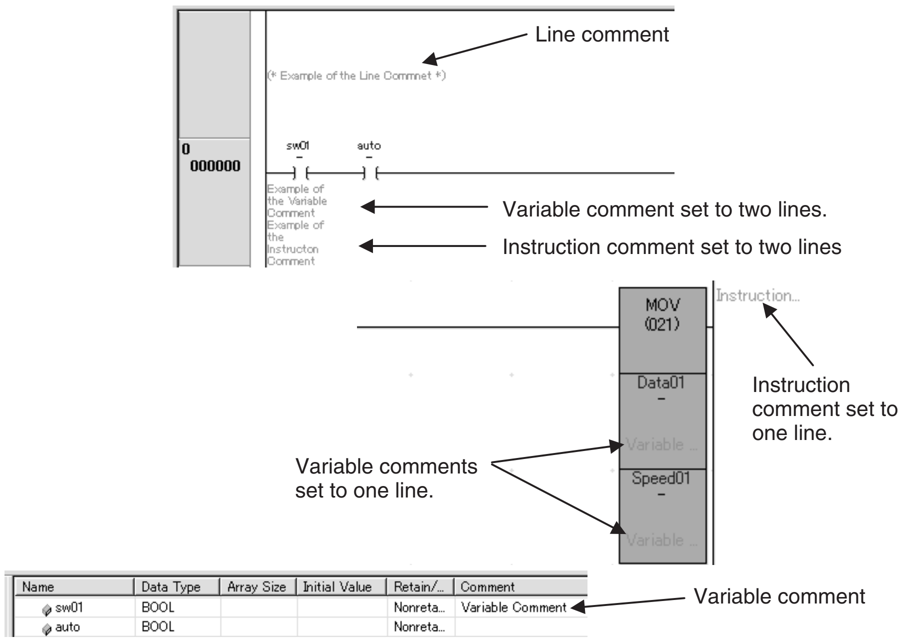
   Note For consistency, the comments of the external variable/global variable are overwritten by the selected external variable's comment. In addition, the global variable's AT specification, initial value, and network variable settings will be deleted.

5-7 Editing Comments

5-7-1 Overview

The following comments can be input and edited.

Comment type	Description	Display location
Variable comments	Comments for variables	Refer to the following figure.
Instruction com-ments	Comments for instructions	
Line comments	Comments that are input on comment lines inserted above or below rungs	

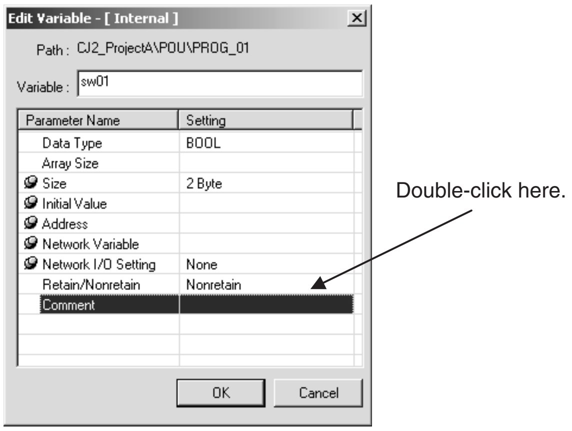

Line comment
(* Example of the Line Commnet *)
0
000000
sw01
auto
Example of the Variable Comment
Example of the Instructon Comment
Variable comment set to two lines.
Instruction comment set to two lines
MOV (021)
Instruction...
Data01
Variable ...
Speed01
Variable ...
Instruction comment set to one line.
Variable comments set to one line.
Variable comment

Name	Data Type	Array Size	Initial Value	Retain/...	Comment
sw01	BOOL			Nonreta...	Variable Comment
auto	BOOL			Nonreta...	

Note (1) Variable comments and instruction comments can be displayed or hidden and the number of display lines can be displayed on the Options Dialog Box displayed when ***Tool - Option*** is selected.

5-7-2 Inputting Variable Comments

1,2,3...
1. Double-click the variable to which a comment is to be added on the Variable Editor.
   The Edit Variables Dialog Box will be displayed.



Path : CJ2_ProjectA\POU\PROG_01

Variable : sw01

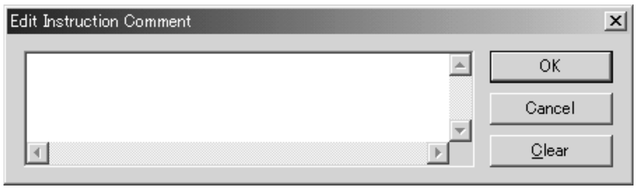
Parameter Name	Setting
Data Type	BOOL
Array Size	
Size	2 Byte
Initial Value	
Address	
Network Variable	
Network I/O Setting	None
Retain/Nonretain	Nonretain
Comment	

2. Double-click the comment field.
   Editing the comment will be enabled.
3. Input the comment and then click the **OK** Button.

In addition to the method described above, comments can be input by using the support for converting variables to physical addresses. For information on that method, refer to *5-3-7 Support for Converting to Physical Addresses.* For display examples, refer to *5-7-1 Overview* for display examples.

5-7-3 Inputting Instruction Comments

1,2,3...
1. Right-click the instruction to which a comment is to be added on the Ladder Editor and then select ***Edit Comment***.
   The Edit Instruction Comment Dialog Box will be displayed.

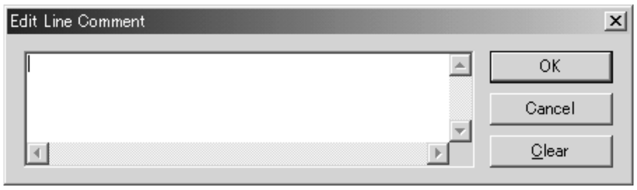


2. Input the comment and then click the **OK** Button.

Refer to *5-7-1 Overview* for display examples.

5-7-4 Inputting Line Comments

1,2,3...
1. Right-click the line where a comment is to be added and then select ***Insert - Insert Line Comment***.
   The Edit Line Comment Dialog Box will be displayed.



2. Input the comment and then click the **OK** Button.

Note Up to 2,000 characters can be input for a line comment.

Refer to *5-7-1 Overview* for display examples.

5-8 Search/Replace Function

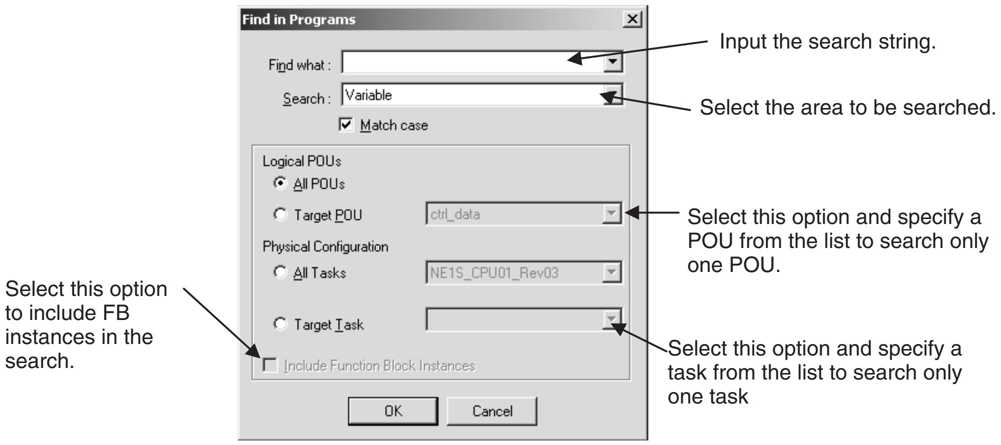
5-8-1 Overview

The following search and replace operations can be performed.

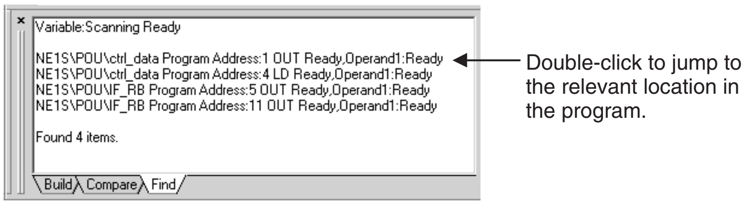
Type	Function
Searching programs	Mnemonics, variables, physical addresses, instance names, line comments, instruction comments, etc. can be searched in all or specific POUs. ST programs are not searched. The search results are output to the Output Window. The locations that were found can be jumped to by double-clicking in the Output Window.
Search/replace/jump operations in the Ladder Editor	Search/replace operations can be performed in the active Ladder Editor. Jumping is possible to specified step numbers (program addresses) or rung numbers.
Search/replace/jump operations in the Variable Editor	Search/replace operations can be performed in the active Variable Editor.

5-8-2 Searching Programs

1,2,3... 1. Press the **Ctrl+Shift+F** Keys. Alternately select ***Edit - Find in Programs***. The Find in Programs Dialog Box will be displayed.


Find in Programs
Find what :
Search :
Variable
Match case
Logical POUs
All POUs
Target POU
ctrl_data
Physical Configuration
All Tasks
NE1S_CPU01_Rev03
Target Task
Include Function Block Instances
OK
Cancel
Input the search string.
Select the area to be searched.
Select this option and specify a POU from the list to search only one POU.
Select this option to include FB instances in the search.
Select this option and specify a task from the list to search only one task

2. Set the range to be searched, the text string to search for, and the search conditions, and then click the **OK** Button.
The search results will be displayed in the Output Window.



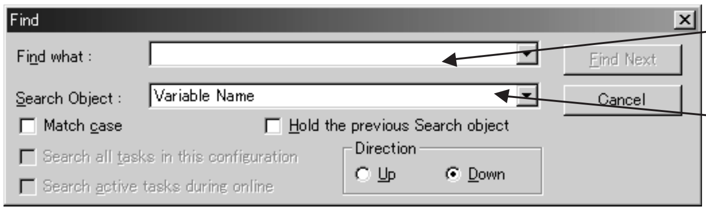
```
Variable:Scanning Ready

NE1S\POU\ctrl_data Program Address:1 OUT Ready,Operand1:Ready
NE1S\POU\ctrl_data Program Address:4 LD Ready,Operand1:Ready
NE1S\POU\IF_RB Program Address:5 OUT Ready,Operand1:Ready
NE1S\POU\IF_RB Program Address:11 OUT Ready,Operand1:Ready

Found 4 items.
```

Build
Compare
Find
Double-click to jump to the relevant location in the program.

5-8-3 Search/Replace/Jump Operations in the Ladder Editor

Searching in the Ladder Editor

1,2,3...
1. Open the Ladder Editor to be searched.
2. Press the **Ctrl+F** Keys. Alternately select ***Edit - Find***.
   The Find Dialog Box will be displayed.


Find
Find what :
Search Object :
Variable Name
Match case
Hold the previous Search object
Search all tasks in this configuration
Search active tasks during online
Direction
Up
Down
Find Next
Cancel
Input the search string.
Select the object to search.

Note When the CPU Unit is connected online, a search operation was executed in the Ladder Editor, and you want to search through all tasks, select the Search all tasks in this configuration Option. If you want to search only the active tasks, select the Search active tasks during online Option.

3. Set the range to be searched, the text string to search for, and the search conditions, and then click the **Find Next** Button.
   - If the text string is found, the line in the Variable Editor containing it will be highlighted (selected).
   - Press the **F3** Key to find the next occurrence.
   - Press the **Shift+F3** Keys to return to the previously found location.

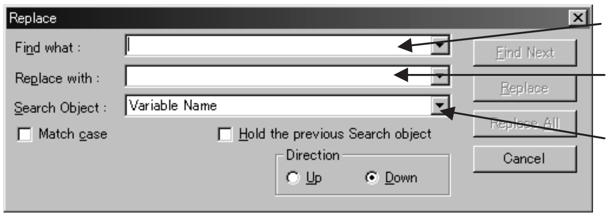
Jumping to a Variable Declaration from the Ladder Editor

When a variable is selected in the Ladder Editor, the NE Programmer can jump directly to that variable's declaration position (in the Variable Editor).

1,2,3...
1. Select the variable in the Ladder Editor.
2. Select ***Edit - Jump - Jump Variable Define***. The NE Programmer will jump to the same variable name in the Variable Editor.

Replacing in the Ladder Editor

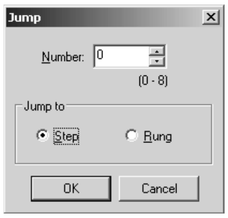
1,2,3...
1. Open the Ladder Editor in which to perform the replace operation.
2. Press the **Ctrl+H** Keys. Alternately select ***Edit - Replace***.
   The Replace Dialog Box will be displayed.



3. Set the range to be searched, the text string to search for, the replacement text string, and the search conditions, and then click the **Find Next** Button. If the text string is found, the instruction in the Ladder Editor containing it will be highlighted (selected).
4. Click the **Replace** Button.
   The text string will be replaced.

Jumping in the Ladder Editor

1,2,3...
1. Make the Ladder Editor active and press the **Ctrl+G** Keys. Alternately select ***Edit - Jump***.
   The Step/Rung No. Jump Dialog Box will be displayed.



2. Set the step number (program address) or rung number and then click the **OK** Button.
   The cursor will jump to the specified step or rung and the instruction or rung will be highlighted (selected).

Searching for Address References in the Ladder Editor (Shift+Alt+A Key)

Jumps can be made from an input instruction at the cursor to an output instruction using the same variable for from an output instruction to an input instruction using the same variable.

1,2,3...
1. In the Ladder Editor, select the input or output instruction for which to search for address references.
2. Press the **Shift+Alt+A** Key. Alternately, select ***Edit - Jump - Bit Address Reference***.
   The cursor will jump to an output or input instruction using the same variable. The cursor will jump to the next output or input instruction each time the **Shift+Alt+A** Key is pressed.

Other Search Operation in the Ladder Editor

Next Operand (Shift+Alt+N Keys)

A jump can be made from the instruction at the cursor to an instruction with the same operand.

1,2,3...
1. In the Ladder Editor, select the instruction with the address for which to search.
2. Press the **Shift+Alt+N** Key. Alternately, select ***Edit - Jump - Next Operand Reference***.
   The cursor will jump to an instruction with the same address.

Next Input (Shift+Alt+I Keys)

A jump can be made from the instruction at the cursor to an input instruction with the same variable.

1,2,3...
1. In the Ladder Editor, select the instruction with the variable for which to search.
2. Press the **Shift+Alt+I** Key. Alternately, select ***Edit - Jump - Next Input***.
   The cursor will jump to an input instruction with the same variable.

Next Output (Shift+Alt+O Keys)

A jump can be made from the instruction at the cursor to an output instruction with the same variable.

1,2,3...
1. In the Ladder Editor, select the instruction with the variable for which to search.
2. Press the **Shift+Alt+O** Key. Alternately, select ***Edit - Jump - Next Output***.
   The cursor will jump to an output instruction with the same variable.

Back (Shift+Alt+B Keys)

The cursor can be returned to the previous instruction from which a search was made.

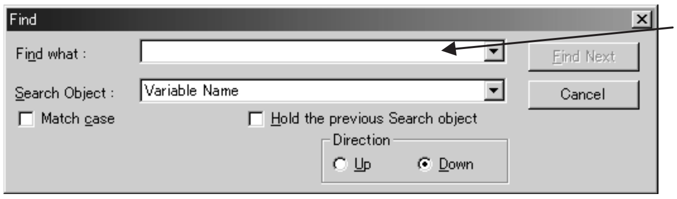
Press the **Shift+Alt+B** Key. Alternately select ***Edit - Jump - Previous Jump Point***.
The cursor will return to the previous instruction.

5-8-4 Search/Replace Operations in the Variable Editor

Search Operations in the Variable Editor

1,2,3...

1. Place the cursor in the Variable Editor to be searched.
2. Press the **Ctrl+F** Keys. Alternately select ***Edit - Find***.
   The Find Dialog Box will be displayed.

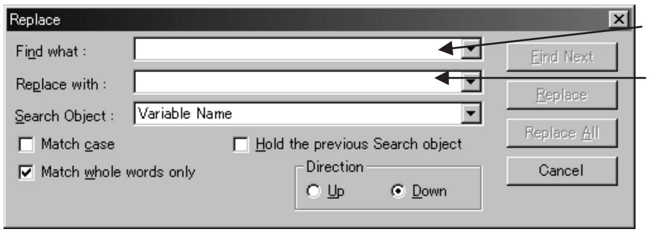


3. Set the range to be searched, the text string to search for, and the search conditions, and then click the **Find Next** Button.
   - If the text string is found, the line in the Variable Editor containing it will be highlighted.
   - Press the **F3** Key to find the next occurrence.
   - Press the **Shift+F3** Keys to return to the previously found location.

Replace Operations in the Variable Editor

1,2,3...

1. Place the cursor in the Variable Editor in which to perform a replace operation.
2. Press the **Ctrl+H** Keys. Alternately select ***Edit - Replace***.
   The Replace Dialog Box will be displayed.



3. Set the range to be searched, the text string to search for, the replacement text string, and the search conditions, and then click the **Find Next** Button.
   If the text string is found, the line in the Variable Editor containing it will be highlighted.
4. Click the **Replace** Button.
   The text string will be replaced.

5-9 Cross Reference Function

5-9-1 Overview

When the cursor is over a variable or address in the Ladder Editor or Variable Table, the cross reference function can display a list (in the Cross Reference Pop-up Window) showing the corresponding instructions in which that variable or address is used. Jumps can be made to the instructions by double-clicking items displayed in the list.

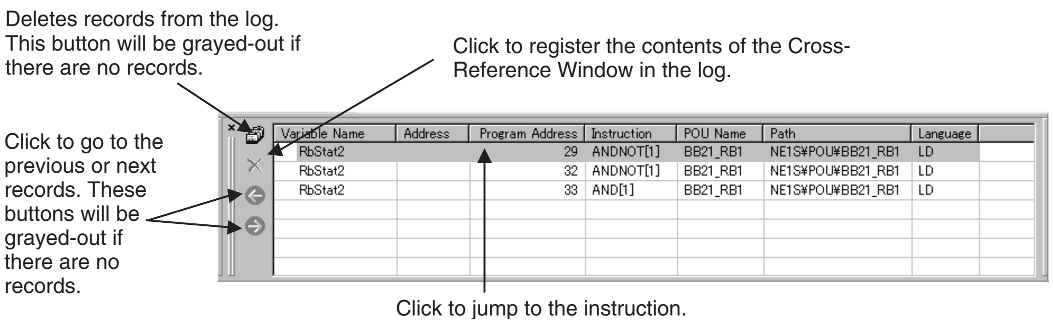
A log is also kept of the instructions that are found to enable jumping to them by selected instructions registered in the log.

5-9-2 Cross Reference Pop-up Window Displays and Operations

1,2,3...

1. If the Cross Reference Window is not currently displayed, select ***View - Window - Cross Reference Tool***.
2. Move the cursor to the variable or address for which to search.
   A list will be displayed to show the instructions in which the variable or address at the cursor is used, as shown below.

   In this example, the cursor is located over the variable *RbStat2*.


Deletes records from the log. This button will be grayed-out if there are no records.
Click to register the contents of the Cross-Reference Window in the log.
Click to go to the previous or next records. These buttons will be grayed-out if there are no records.
Click to jump to the instruction.

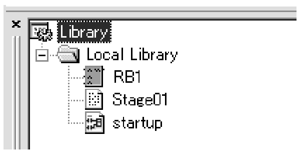
Variable Name	Address	Program Address	Instruction	POU Name	Path	Language
RbStat2		29	ANDNOT[1]	BB21_RB1	NE1S¥POU¥BB21_RB1	LD
RbStat2		32	ANDNOT[1]	BB21_RB1	NE1S¥POU¥BB21_RB1	LD
RbStat2		33	AND[1]	BB21_RB1	NE1S¥POU¥BB21_RB1	LD

Note When some of the function block's I/O variables have been grouped, the Cross Reference Pop-up Window will display all of the cross reference information for the group members. Also, when an array variable is displayed, the Cross Reference Pop-up Window will display all of the cross reference information for the other array variables with the same variable name.

5-10 Using the Library

5-10-1 Overview

Logical POUs (programs or function blocks) and groups of rungs can be saved as library files so that they can be reused.



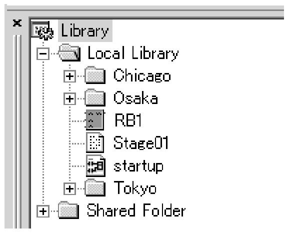
As shown above, programs, function blocks, and rung groups (one or more rungs) can be stored and reused with one file for each program, one file for each function block, and one file for each rung group.

Each library contains information such as the local variables (internal, input, output, and external variables), function blocks, data type definitions, and global variables.

By default, the library files are stored under a *Local Library* folder in the installation directory, as shown above.

As shown below, groups of library files can be registered and managed in a directory hierarchy.

You can also right-click the *Library* Directory and select ***Map Folder*** to register another folder in the first level of the library. System development by multiple engineers is possible by specifying a shared folder on a server in the first level of the library directory.



```
Library
  Local Library
    Chicago
    Osaka
    RB1
    Stage01
    startup
    Tokyo
  Shared Folder
```

- Program or function block library files that have been registered can be added to projects by right-clicking and selecting ***Register to Library***.
- To reuse rung groups, they can be dragged and dropped on the ladder diagram program, or alternately the rung groups can be added by right-clicking and selecting ***Insert to Program***.

  Note "Local Library" is the installation folder. It requires system administrator rights. To allow users without system administrator rights to access the local library, create a folder with appropriate access rights, and allocate that folder to the local library.

Note Library parts cannot be edited after they have been registered in the library. They must be re-registered in order to change them.

Libraries by Type of Controller

The library display and operation in the Library Window depend on the controller series (CPU Unit series); they do not depend on the type of library. If the currently open project is for the CJ2, items for the NE1S will be displayed with a "disabled" icon, and they cannot be added to the project for the CJ2 CPU Unit. In the same way, if a project for a NE1S-series CPU Unit is open, library items for the CJ2 will be disabled and you cannot add them to the project for the NE1S-series CPU Unit.

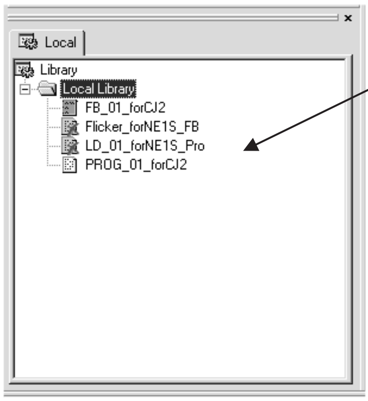
5-10-2 Displaying the Library Window

1,2,3...
1. If the Library Window is not currently displayed, select ***View - Window - Library***.
2. The Library Window will be displayed.
   The Library Window will be automatically displayed when a library file is registered as described in sections below.

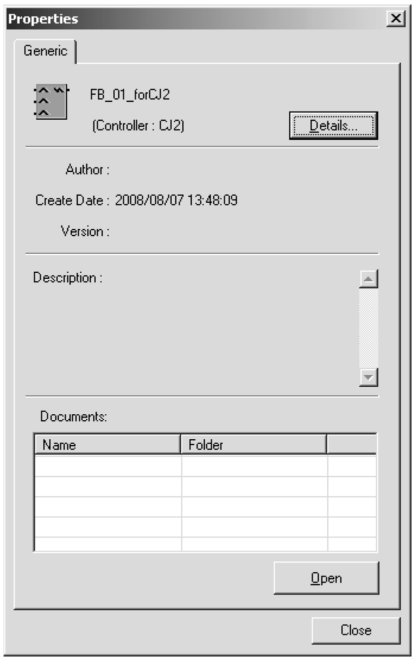
Display Showing the Controller Series

The display elements in the Library Window that depend on the controller series (CPU Unit series) are shown below.

■ **Example: Library Window Opened for Project for CJ2 CPU Unit**

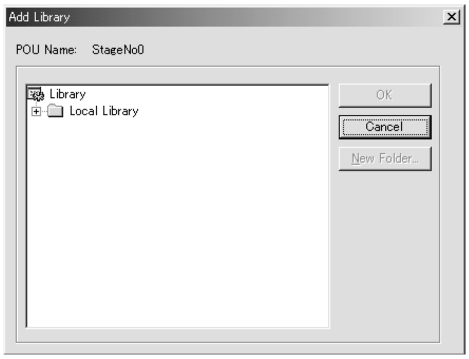

Local
Library
Local Library
FB_01_forCJ2
Flicker_forNE1S_FB
LD_01_forNE1S_Pro
PROG_01_forCJ2
The library elements for the NE1S are marked with an icon showing that the library cannot be used because the Library Window that is open is for a project for the CJ2 CPU Unit.

Check the library properties to determine whether the library items are for the NE1S or the CJ2. Double-click or right-click and select ***Properties*** from the pop-up menu.



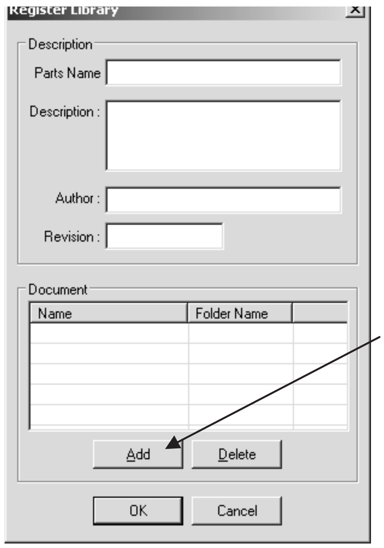
5-10-3 Registering Logical POUs in the Library

1,2,3... 1. Right-click the logical POU (program or function block) in the Workspace and select ***Register to Library*** from the popup menu.
The following dialog box will be displayed.



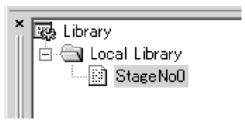
- To create a new folder in the library tree, select the insertion location and click the **New Folder** Button.

2. Select the folder in which to register the library part (Local Library, the default folder, in this example) and then click the **OK** Button.
   The Register Library Dialog Box will be displayed.



- When the logical POU used to register a library has a text file, the library can be linked to the relevant text file.
- Click the **Add** Button and specify the text file.
- The added text file can be opened from the Properties Window for the registered library (right-click on the library and select ***Properties***).

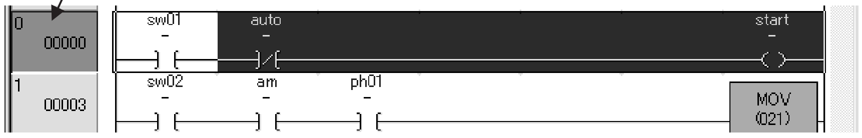
3. Input the name of the part and any other required items and then click the **OK** Button.
   The part will be registered in the library and displayed in the Library Window as shown below.



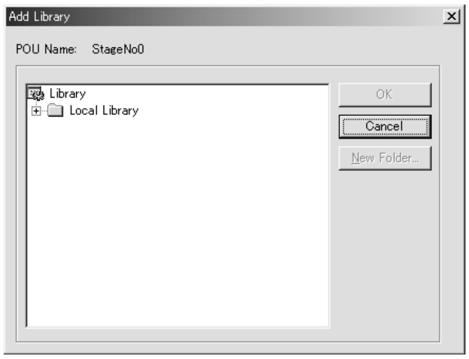
5-10-4 Registering Rung Groups in the Library

1,2,3... 1. Select one or more rungs in the Ladder Editor and select ***Library - Rung - Register to Library***.

Click the rung header area to select the rung. Click the rung header area while holding down the Shift Key to select multiple rungs.

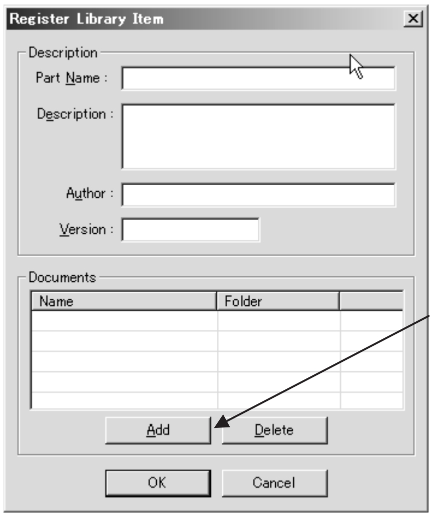


The following dialog box will be displayed.



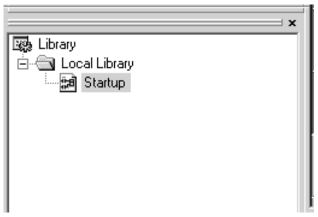
- To create a new folder in the library tree, select the insertion location and click the **New Folder** Button.

2. Select the folder in which to register the library part (Local Library, the default folder, in this example) and then click the **OK** Button.
   - The Register Library Dialog Box will be displayed.



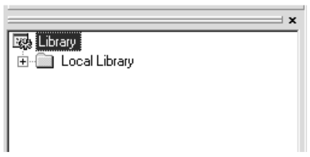
- When the rung group used to register a library has a text file, the rung group can be linked to the relevant text file.
- Click the **Add** Button and specify the text file.
- The added text file can be opened from the Properties Window for the registered rung group (right-click on the rung group and select ***Properties***).

3. Input the name of the part and any other required items and then click the **OK** Button.
   The part will be registered in the library and displayed in the Library Window as shown below.

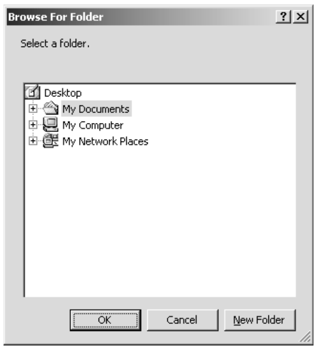


5-10-5 Registering Folders in the Library

1,2,3... 1. Right-click the Library and select ***Map Folder***.



The following Browse Folder Dialog Box will be displayed.



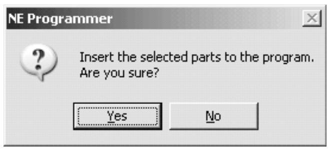
2. Select the folder to be added and then click the **OK** Button.
   The specified folder will be added to the library tree.

5-10-6 Using the Library

POUs (Programs and Function Blocks)

Program sections and function blocks can be inserted in projects.

1,2,3...
1. Right-click the program or function block and select ***Add to Project***. The following confirmation dialog box will be displayed.

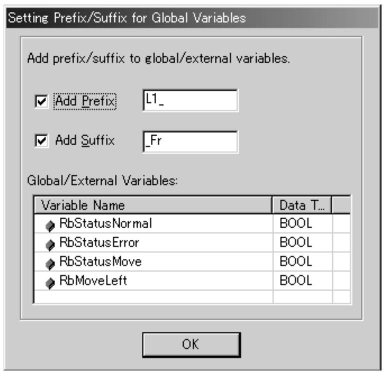

NE Programmer

Insert the selected parts to the program.
Are you sure?

2. Click the **Yes** Button.
   - If the POU contains a global variable, the following Setting Prefix/Suffix for Global Variables Dialog Box will be displayed.
   - Use this dialog box to attach prefixes and suffixes to all global variables contained in POUs at once.

   Note This function enables the variable names for global variables to be changed altogether, making reuse of global variables easy.

   Example: Adding the prefix L1 and the suffix _Fr.


Setting Prefix/Suffix for Global Variables

Add prefix/suffix to global/external variables.

☑ Add Prefix: L1_

☑ Add Suffix: _Fr

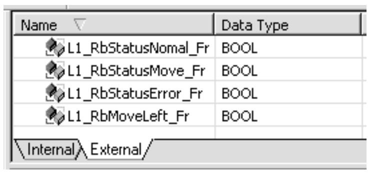
Global/External Variables:

Variable Name	Data T...
RbStatusNormal	BOOL
RbStatusError	BOOL
RbStatusMove	BOOL
RbMoveLeft	BOOL

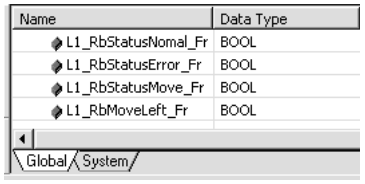
3. After setting, click the **OK** Button. When prefix/suffix settings are not required, click the **OK** Button without making any settings.
   - The program or function block will be added to the project. The POU name will be displayed in the project (do not use a library name).

   Note If the same POU name already exists in the project, a dialog box will be displayed to confirm that it is OK to overwrite it.

- The prefix/suffix settings in the above example will be displayed as follows:



Name	Data Type
L1_RbStatusNomal_Fr	BOOL
L1_RbStatusMove_Fr	BOOL
L1_RbStatusError_Fr	BOOL
L1_RbMoveLeft_Fr	BOOL

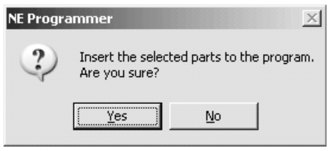


Name	Data Type
L1_RbStatusNomal_Fr	BOOL
L1_RbStatusError_Fr	BOOL
L1_RbStatusMove_Fr	BOOL
L1_RbMoveLeft_Fr	BOOL

Rung Groups

Rung groups can be simply dragged and dropped into ladder programs. Delete variables and change variable names as required.

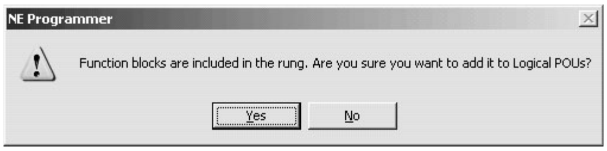
1,2,3...
1. Drag and drop the rung group into the ladder program (or right-click on the rung group and select ***Insert to Program***). The following confirmation dialog box will be displayed.



NE Programmer

Insert the selected parts to the program.
Are you sure?

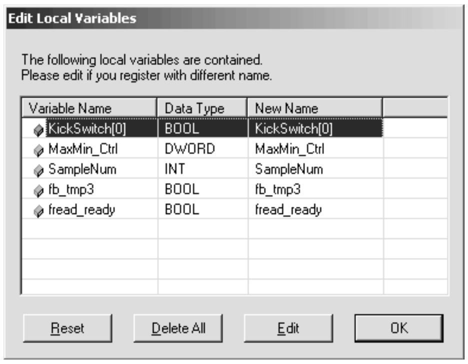
2. Click the **Yes** Button.
   - If the rung contains function blocks, the following confirmation dialog box will be displayed.



NE Programmer

Function blocks are included in the rung. Are you sure you want to add it to Logical POUs?

   - Click the **Yes** Button.
   - The following Edit Local Variables Dialog Box will be displayed.



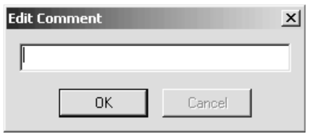
Edit Local Variables

The following local variables are contained.
Please edit if you register with different name.

Variable Name	Data Type	New Name
KickSwitch[0]	BOOL	KickSwitch[0]
MaxMin_Ctrl	DWORD	MaxMin_Ctrl
SampleNum	INT	SampleNum
fb_tmp3	BOOL	fb_tmp3
fread_ready	BOOL	fread_ready

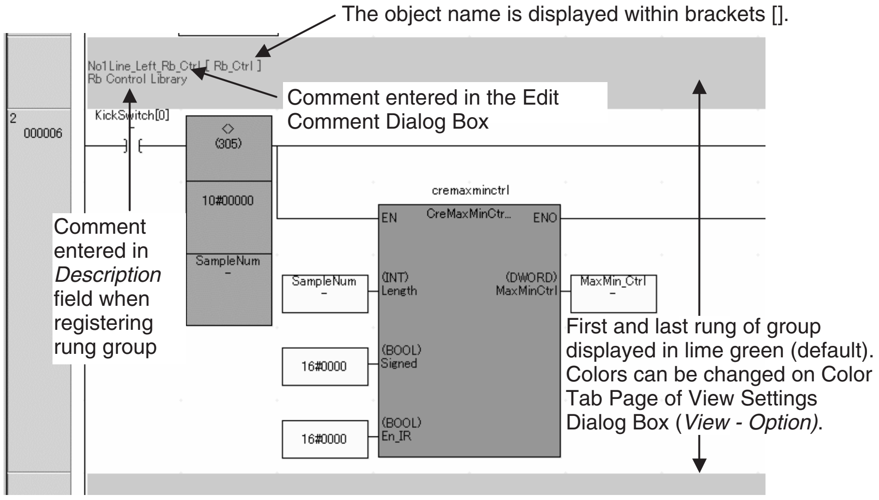
3. To use the variables without any changes, click the **OK** Button.
   - To change the variable name, double-click the variable name and change the name.
   - To delete all variables, click the **Delete All** Button.

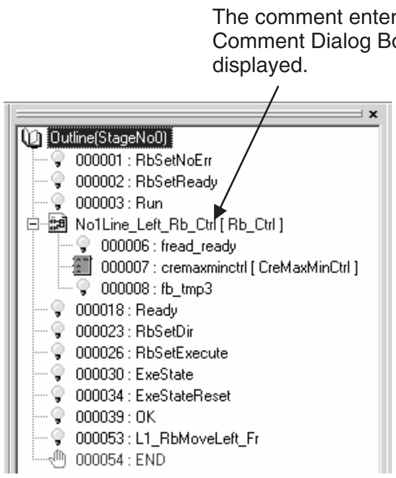
- To return to the original settings after editing the variables, click the **Reset** Button.
  Click the **OK** Button to display the Edit Comment Dialog Box.



4. Enter a comment and click the **OK** Button. The rung will be inserted into the ladder program.

   Example


The object name is displayed within brackets [].
No1Line_Left_Rb_Ctrl [Rb_Ctrl]
Rb Control Library
Comment entered in the Edit Comment Dialog Box
KickSwitch[0]
000006
(305)
10#00000
SampleNum
cremaxminctrl
EN
CreMaxMinCtr...
ENO
SampleNum
(INT) Length
(DWORD) MaxMinCtrl
MaxMin_Ctrl
16#0000
(BOOL) Signed
(BOOL) En_IR
Comment entered in *Description* field when registering rung group
First and last rung of group displayed in lime green (default). Colors can be changed on Color Tab Page of View Settings Dialog Box (*View - Option*).



5-10-7 Transferring Library Items for the NE1S to CJ2

Library items for NE1S-series CPU Units cannot be added to projects for CJ2 CPU Units without making changes. Perform the following procedure to convert library items for CJ2 CPU Units into library items for NE1S-series CPU Units.

The following procedure is recommended to convert the items.

1,2,3...

1. Create a project for an NE1S-series CPU Unit and add library items for the NE1S-series CPU Unit to be converted for a CJ2 CPU Unit.
2. Change the Controller series (CPU Unit series) and convert the project from one for a NE1S-series CPU Unit to one for a CJ2 CPU Unit.

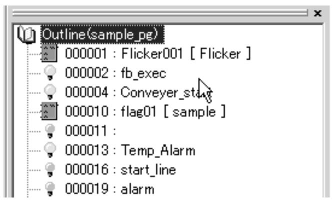
3. Initial settings may become disabled or other operation may change when the series is changed. Therefore, debug the programming and verify operation after converting to a project for the CJ2 CPU Unit.
4. Register the library items as library items for the CJ2 CPU Unit.

Note CJ2 projects cannot be converted to NE1S projects. Therefore, CJ2 CPU Unit library items cannot be used in projects for NE1S-series CPU Units.

5-11 Outline Window

5-11-1 Outline Window

The Outline Window displays an outline of the logical POU (program or function block) that is being edited. Jumping to any instruction in the Ladder Editor or ST Editor is possible by clicking it in the Outline Window.



```
Outline(sample_pg)
  000001 : Flicker001 [ Flicker ]
  000002 : fb_exec
  000004 : Conveyer_start
  000010 : flag01 [ sample ]
  000011 :
  000013 : Temp_Alarm
  000016 : start_line
  000019 : alarm
```

As shown above, line comments, outputs, function blocks, rung groups, the interlock instruction, and the END instruction are displayed in the Outline Window.

Note For structured text, only the line comments and function blocks are displayed in the Outline Window.

Option Settings Display items in the Outline Window and the timing of operation related to the Ladder Editor can be changed. To make changes, select ***Tool - Option - Outline***. For details, refer to *4-6-2 Outline*.

5-11-2 Displaying the Outline Window

1,2,3...

1. If the Outline Window is not currently displayed, select ***View - Window - Outline***.
   The Outline Window will be displayed. If a logical POU is being edited at the time, an outline will be displayed in the Outline Window.
2. To display an outline of a specific logical POU, double-click it in the Workspace.
   When the logical POU is opened, an outline will be displayed in the Outline Window at the same time.

5-12 Building and Compiling Programs

5-12-1 Building and Compiling

A project is build, all of the programs in the project are checked and an executable file for the CPU Unit is generated. Always build the project before downloading it to the CPU Unit.

Compiling can be used to perform program checks on individual logical POUs (programs or function blocks). Only one active logical POU (i.e., the one being edited) can be compiled at the same time.

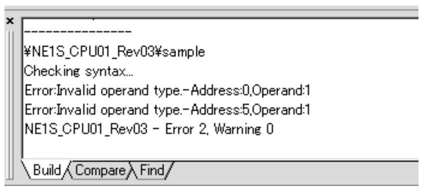
The program check level can be set.

5-12-2 Building

Press the **F7** Key. Alternately select ***Build - Build***.

The project will be built. Any errors or warning generated during building will be displayed in the Output Window.

Display Example

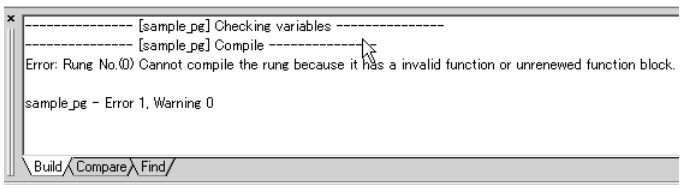


```
---------------
¥NE1S_CPU01_Rev03¥sample
Checking syntax...
Error:Invalid operand type.-Address:0,Operand:1
Error:Invalid operand type.-Address:5,Operand:1
NE1S_CPU01_Rev03 - Error 2, Warning 0
```

5-12-3 Compiling

1,2,3...
1. Make the program or function block to be checked active (i.e., open it).
2. Press the **Ctrl+F7** Keys. Alternately select ***Edit - Compile***.
   The program will be checked and any errors or warning generated during compilation will be displayed in the Output Window.

Display Example



```
--------------- [sample_pg] Checking variables ---------------
--------------- [sample_pg] Compile ---------------
Error: Rung No.(0) Cannot compile the rung because it has a invalid function or unrenewed function block.

sample_pg - Error 1, Warning 0
```

5-12-4 Level Settings for Program Check

Use the following procedure to set the level for program checking. If level A is set, a stricter program check will be performed. (The default is level A.)

For details on the program check, refer to *10-6-8 Program Check*.

5-12-5 Detailed Build Information

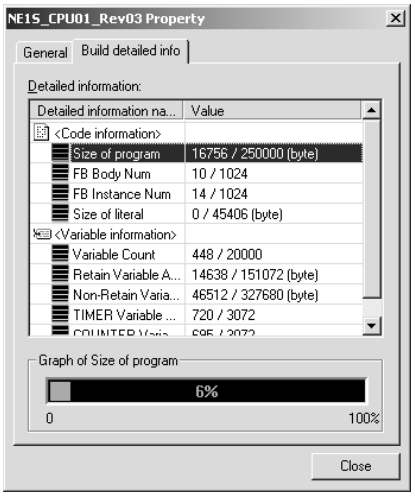
There is a tab page for build information in the configuration properties. The build information for each element is displayed here when building a program. If the program has not been built, a message will be displayed to say that the information cannot be displayed and that building the program is required to display the information.

The display shows the degree to which the user program is consuming the resources of the selected CPU Unit for each element. Build information is mainly useful for the following points.

- Judgment criteria when changing the model of CPU Unit
- As part of hardware management

1,2,3...

1. Right-click the configuration in the Project Window and select ***Properties*** from the menu. Click the **Build Detailed Info** Tab.



NE1S_CPU01_Rev03 Property

General | Build detailed info

Detailed information:

Detailed information na...	Value
<Code information>	
Size of program	16756 / 250000 (byte)
FB Body Num	10 / 1024
FB Instance Num	14 / 1024
Size of literal	0 / 45406 (byte)
<Variable information>	
Variable Count	448 / 20000
Retain Variable A...	14638 / 151072 (byte)
Non-Retain Varia...	46512 / 327680 (byte)
TIMER Variable ...	720 / 3072
COUNTER Varia...	695 / 3072

Graph of Size of program

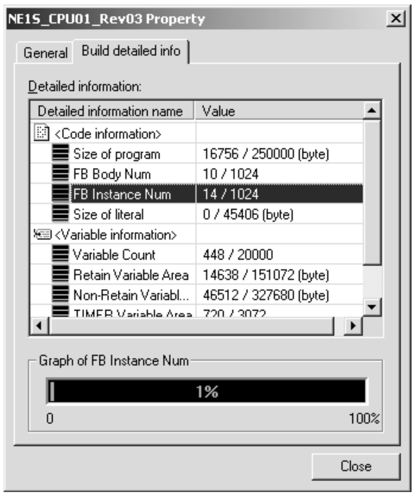
6%

0 100%

Close

Note The Build Detailed Info Tab Page will be displayed only if the program has been built without error.

2. The percentage graph at the bottom of the dialog box will switch according to the elements selected in the list.



NE1S_CPU01_Rev03 Property

General | Build detailed info

Detailed information:

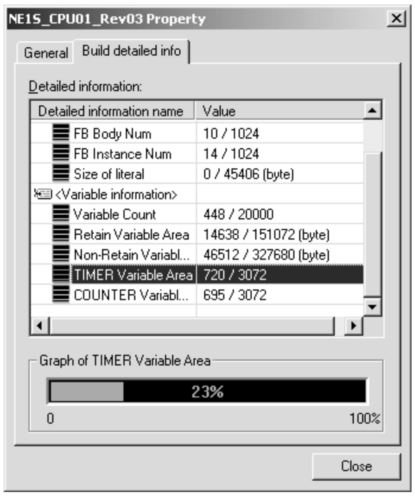
Detailed information name	Value
<Code information>	
Size of program	16756 / 250000 (byte)
FB Body Num	10 / 1024
FB Instance Num	14 / 1024
Size of literal	0 / 45406 (byte)
<Variable information>	
Variable Count	448 / 20000
Retain Variable Area	14638 / 151072 (byte)
Non-Retain Variabl...	46512 / 327680 (byte)
TIMER Variable Area	720 / 3072

Graph of FB Instance Num

1%

0 100%

Close



NE1S_CPU01_Rev03 Property

General | Build detailed info

Detailed information:

Detailed information name	Value
FB Body Num	10 / 1024
FB Instance Num	14 / 1024
Size of literal	0 / 45406 (byte)
<Variable information>	
Variable Count	448 / 20000
Retain Variable Area	14638 / 151072 (byte)
Non-Retain Variabl...	46512 / 327680 (byte)
TIMER Variable Area	720 / 3072
COUNTER Variabl...	695 / 3072

Graph of TIMER Variable Area

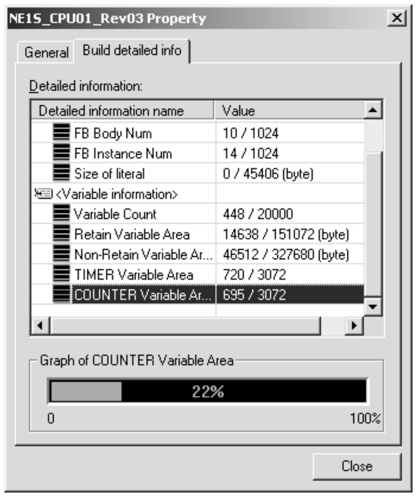
23%

0 100%

Close

Note There will be a build error if the graph of the selection item exceeds 100% for the program. If a build error occurs, information on the type of insufficient resource will be displayed in an error message in the Output Window. Take measures for each item of information.

If the resources are insufficient, revise the program to reduce the amount of resources used or change the CPU Unit to a model with a larger capacity.



Detailed information name	Value
FB Body Num	10 / 1024
FB Instance Num	14 / 1024
Size of literal	0 / 45406 (byte)
<Variable information>	
Variable Count	448 / 20000
Retain Variable Area	14638 / 151072 (byte)
Non-Retain Variable Ar...	46512 / 327680 (byte)
TIMER Variable Area	720 / 3072
COUNTER Variable Ar...	695 / 3072

⚠ **Caution** If there is a build error, the Build Detailed Info Tab Page will not be displayed in the configuration properties.

5-13 Importing and Exporting

5-13-1 Overview

The following logical POUs can be imported and variables can be imported/ exported from the Variable Editor.

Element	Import/Export files
Mnemonics	Import: Text (.txt) files in special formats
	Export: Text (.txt) files in special formats
Variables	Import: CSV files in special formats
	Export: • CSV files in special formats • Text files for SPU Console Also, files that can be exported depend on the CPU Unit for the following items. • NE1S Projects • CSV files for NE OPC Servers • Text files for CX-Designer of version 2.x or earlier • Text files for CX-Designer version 2.03H for NS-Runtime version 1.00H • CJ2 Projects • CSV files for SYSMAC OPC servers of version 3.x • Text files for CX-Designer version 3.0

5-13-2 Importing Mnemonics

Import

Reads mnemonic data stored in text files in proprietary formats.

1,2,3... 1. Select ***Mnemonic - Import*** in the Mnemonic Editor, specify the desired file name, and select ***Open***.

Export

Saves edited mnemonic data in a text file in proprietary formats.

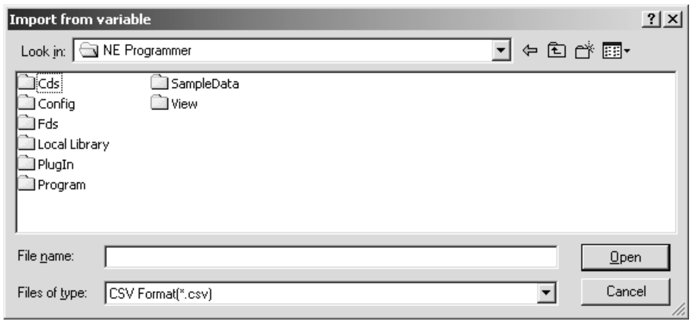
1,2,3... 1. Select ***Mnemonic - Export*** in the Mnemonic Editor, specify the desired file name, and select ***Save***.

5-13-3 Importing and Exporting Variables

Importing

Use to following procedure to import CSV files created on external Programming Devices, Excel, etc.

1,2,3... 1. Either select ***Variable - Import - CSV Format***. from the Menu Bar, or right-click the Local Variable Editor or Global Variable Editor and select ***Import - CSV Format*** from the popup menu.
The following dialog box will be displayed.

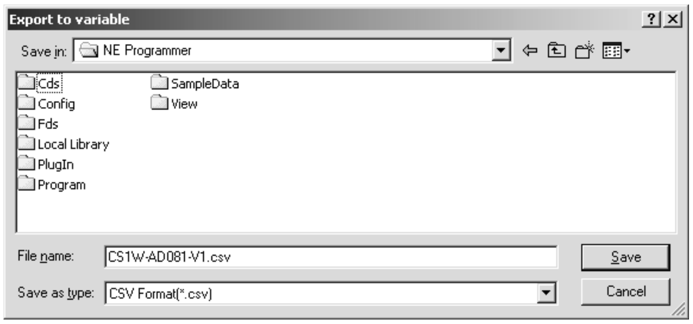


2. Specify the location and name of the file and click the **Open** Button. The variables will be imported.

Exporting CSV Files

Variable data can be saved in CSV flies for use on external Programming Devices, Excel, etc.

1,2,3... 1. Either select ***Variable - Export - CSV Format***. from the Menu Bar, or right-click the Local Variable Editor or Global Variable Editor and select ***Export - CSV Format*** from the popup menu.
The following dialog box will be displayed.



File name: CS1W-AD081-V1.csv

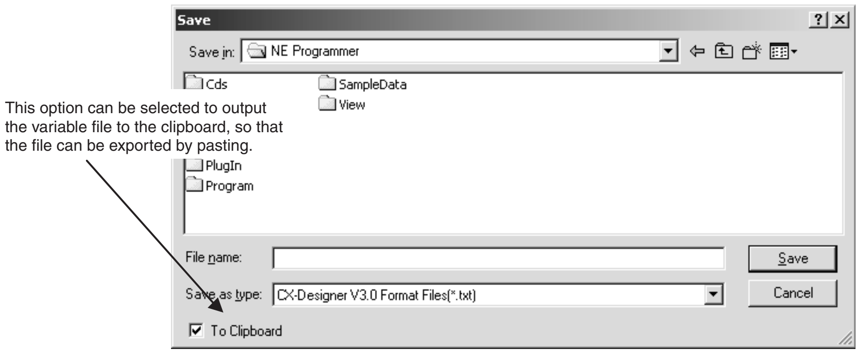
2. Specify the location and name of the file to save and click the **Save** Button. The variables will be saved in a CSV file.

Exporting in OPC Server, CX-Designer, or SPU-Console Format

Use the following procedure to save variable data to a CSV or text file for use with OPC Server, CX-Designer, or SPU-Console.

1,2,3...

1. Select ***File - Export Variables***. The Save Dialog Box will be displayed.



This option can be selected to output the variable file to the clipboard, so that the file can be exported by pasting.

Save as type: CX-Designer V3.0 Format Files(*.txt)

2. Select the file type in the *Save as Type* Field.
3. Specify the location and name of the file to save and click the **Save** Button. The variables will be saved in the specified application's format.

Note
(1) If CX-Designer Format Files (*.txt) or SPU Console Format Files (*.txt) is selected, only global variables with specified physical addresses will be exported.
(2) With CX-Designer V3.0 Format Files (*.txt), all global variables will be exported regardless of whether a physical address is specified. Local variables of function blocks will not be exported.

5-14 Printing

All kinds of project data can be printed.

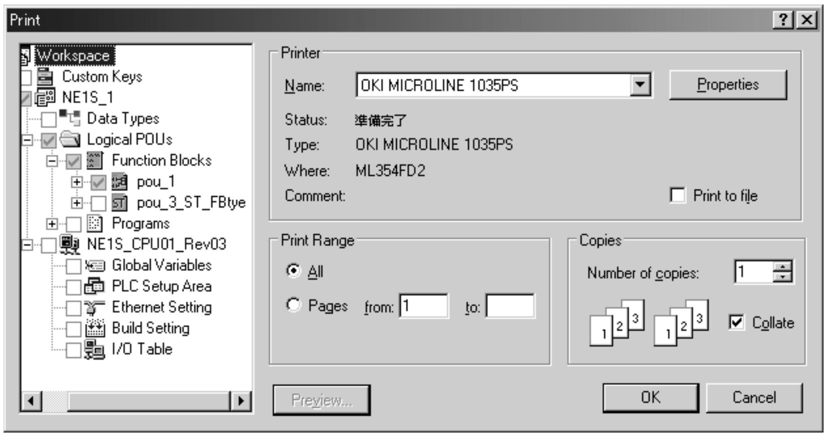
5-14-1 Page Setup

1,2,3... 1. Select ***File - Page Setup***.

Tab page	Contents
Margin	Use to set the page margins in the *Top*, *Bottom*, *Left*, *Right*, *Header*, *and Footer* Fields.
Title Setting	Use to input the title text and set the number of lines, position, and font.
Header	Header text can be input in the *Left*, *Center*, or *Right* Fields.
Footer	Footer text can be input in the *Left*, *Center*, or *Right* Fields.

5-14-2 Printing

1,2,3... 1. Select ***File - Print***.



Name: OKI MICROLINE 1035PS
Status: 準備完了
Type: OKI MICROLINE 1035PS
Where: ML354FD2
Comment:
Print Range: All; Pages from: 1 to:
Number of copies: 1; Collate ✓

2. Select the item to be printed in the Project Workspace at the left side of the Window. The contents of data types, logical POUs (programs or function blocks), global variables, PLC Setup, Ethernet settings, build settings, I/O tables, or Custom Keys can be printed.

   Note If read protection is set for a logical POU (program or function block), a protection icon will be displayed and it will not be possible to select the logical POU for printing.

Note Sub-items can be added or removed from the print job by right-clicking the item and selecting or removing the sub-items (such as ladder diagrams or variable types) in the popup menu.

5-14-3 Print Preview

1,2,3... Click the **Preview** Button at the bottom of the Print Window or select ***File - Print Preview*** from the Menu Bar.

SECTION 6
PLC System Configuration

This section describes the configuration of the PLC system.

6-1 Overview

This section provides an overview of the settings and describes how to display the setting windows.

6-1-1 Settings

The following PLC system configuration settings can be made.

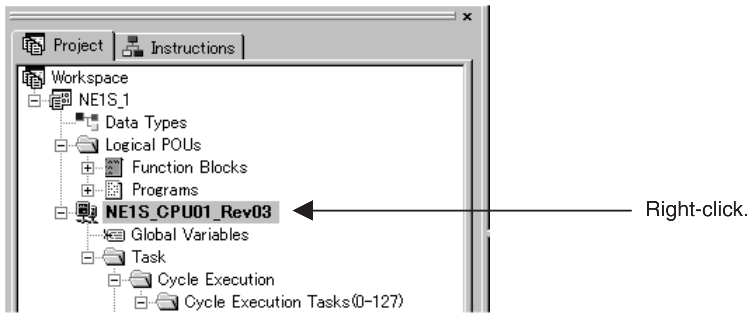
Setting tab name	Setting
PLC Setup	Includes settings such as startup settings, CPU Unit settings, timer/interrupt settings, Special I/O Unit refresh settings, and communications settings.
Ethernet	Includes the built-in Ethernet Setup.
Build	Includes settings such as detailed timer/counter settings and IR/DR sharing settings for tasks.
I/O Table	Used to create I/O tables online or edit I/O tables offline.

6-1-2 Displaying the Setup Window for the System Configuration

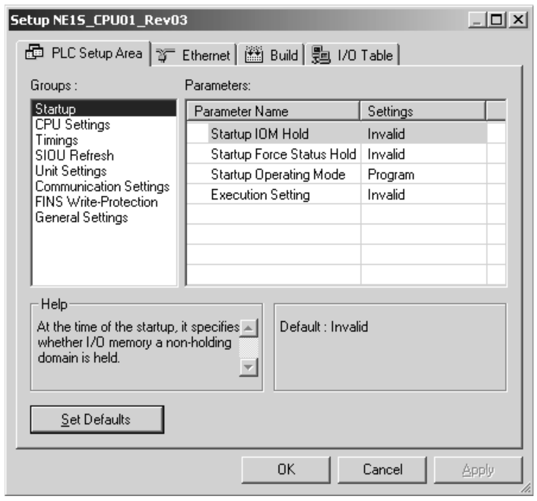
Use the following procedure to display the Setup Window for the system configuration.

1,2,3...

1. Right-click the configuration name (i.e., the PLC name) in the Project Workspace.



2. Select ***System Configuration*** from the popup menu.
   The following Setup Window will be displayed.



Parameter Name	Settings
Startup IOM Hold	Invalid
Startup Force Status Hold	Invalid
Startup Operating Mode	Program
Execution Setting	Invalid

At the time of the startup, it specifies whether I/O memory a non-holding domain is held.

Default : Invalid

The setting on each tab page are described in the following sections.

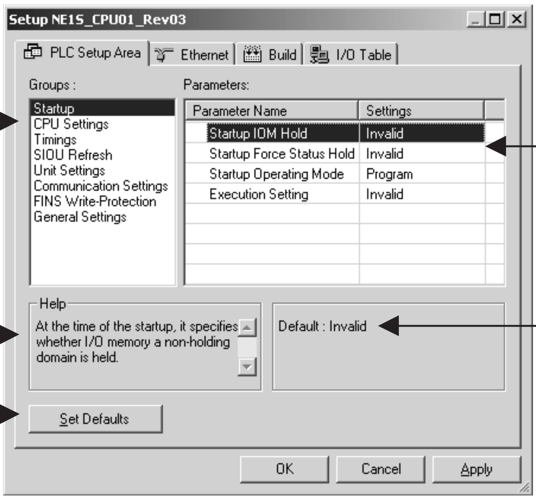
6-2 PLC Setup (PLC Setup Area Tab Page)

This section describes the setting procedure for the PLC Setup.

For details on the PLC Setup for NE1S CPU Units, refer to the *SYSMAC NE1S Series Operation Manual* (Cat. No. Z901).

For details on the PLC Setup for CJ2 CPU Units, refer to *Appendix E* and *Appendix F*.

1,2,3...
1. Click the **PLC Setup** Tab from the Configuration Setting Window.
   The PLC Setup Tab Page, shown below, will be displayed.



Groups: Startup, CPU Settings, Timings, SIOU Refresh, Unit Settings, Communication Settings, FINS Write-Protection, General Settings

Parameter Name	Settings
Startup IOM Hold	Invalid
Startup Force Status Hold	Invalid
Startup Operating Mode	Program
Execution Setting	Invalid

Help: At the time of the startup, it specifies whether I/O memory a non-holding domain is held.

Default : Invalid

Select the group.

Basic help is displayed on the current parameter.

Returns all parameters to their default settings.

The current settings are displayed. Double-click here to change the settings. (Parameters with a mark cannot be changed.)

The default value and setting range for the current parameter are displayed.

2. Make all of the required settings and then click the **OK** Button.

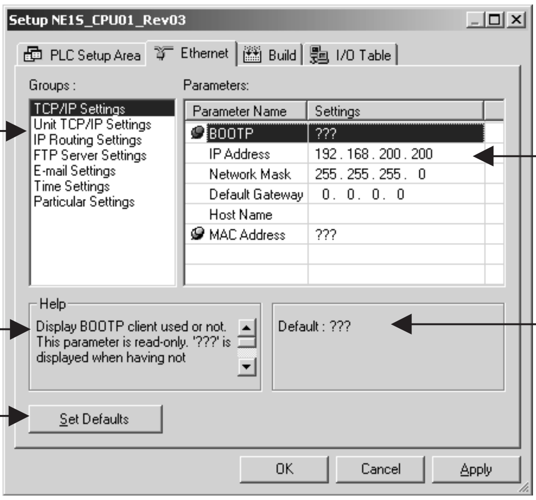
6-3 Ethernet Setup (Ethernet Tab Page)

This section describes the setting procedure for the Ethernet Setup.

For details on Setup for NE1S CPU Units, refer to the *SYSMAC NE1S Series Operation Manual* (Cat. No. Z901).

For details on Ethernet settings for CJ2 CPU Units, refer to *Appendix F Ethernet Settings for CJ2 CPU Units*.

1,2,3...
1. Click the **Ethernet** Tab from the Configuration Setting Window.
   The Ethernet Tab Page, shown below, will be displayed.



Groups: TCP/IP Settings, Unit TCP/IP Settings, IP Routing Settings, FTP Server Settings, E-mail Settings, Time Settings, Particular Settings

Parameter Name	Settings
BOOTP	???
IP Address	192 . 168 . 200 . 200
Network Mask	255 . 255 . 255 . 0
Default Gateway	0 . 0 . 0 . 0
Host Name	
MAC Address	???

Help: Display BOOTP client used or not. This parameter is read-only. '???' is displayed when having not

Default : ???

Select the group.

Basic help is displayed on the current parameter.

Returns all parameters to their default settings.

The current settings are displayed. Double-click here to change the settings. (Parameters with a mark cannot be changed.)

The default value and setting range for the current parameter are displayed.

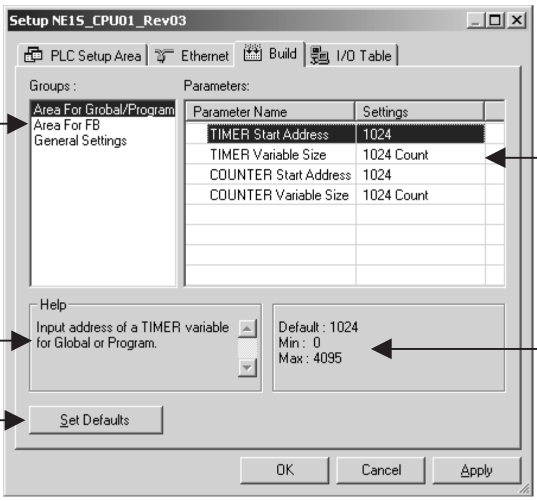
2. Make all of the required settings and then click the **OK** Button.

6-4 Build Settings (Build Tab Page)

This section describes the setting procedure for the Build Settings.

Refer to *6-4-1 Build Settings* for details on the individual settings.

1,2,3... 1. Click the **Build** Tab from the Configuration Setting Window. The Build Tab Page, shown below, will be displayed.



Setup NE1S_CPU01_Rev03

PLC Setup Area | Ethernet | Build | I/O Table

Groups:

- Area For Grobal/Program
- Area For FB
- General Settings

Parameters:

Parameter Name	Settings
TIMER Start Address	1024
TIMER Variable Size	1024 Count
COUNTER Start Address	1024
COUNTER Variable Size	1024 Count

Help: Input address of a TIMER variable for Global or Program.

Default : 1024
Min : 0
Max : 4095

Set Defaults | OK | Cancel | Apply

Select the group.

The current settings are displayed. Double-click here to change the settings. (Parameters with a mark cannot be changed.)

Basic help is displayed on the current parameter.

The default value and setting range for the current parameter are displayed.

Returns all parameters to their default settings.

2. Make all of the required settings and then click the **OK** Button.

6-4-1 Build Settings

NE1S-series CPU Unit Build Settings

Group	Setting	Default	Setting range
Area for Global/Program	Timer Start Address	1,024	0 to 4,095
	Timer Variable Size	1,024	0 to 4,096
	Counter Start Address	1,024	0 to 4,095
	Counter Variable Size	1,024	0 to 4,096
Area for FB	Timer Start Address for FB	2,048	0 to 4,095
	Timer Variable Size for FB	2,048	0 to 4,096
	Timer Variable Reserve Size for Online Edit	1	0 to 4,096
	Counter Start Address for FB	2,048	0 to 4,095
	Counter Variable Size for FB	2,048	0 to 4,096
	Counter Variable Reserve Size for Online Edit	1	0 to 4,096
General Settings	IR/DR Area Shared Between Tasks	Independent	Independent or share

CJ2 CPU Unit Build Settings

Group	Setting	Default	Setting range
Area for Variables	Timer Start Address	1,024	0 to 4,095
	Timer Variable Size	3,072	0 to 4,096
	Counter Start Address	1,024	0 to 4,095
	Counter Variable Size	3,072	0 to 4,096
General Settings	IR/DR Area Shared Between Tasks	Independent	Independent or share

6-5 I/O Table Settings (I/O Table Tab Page)

This section describes the setting procedure for the I/O table settings.

For details on I/O allocations for NE1S-series CPU Units, refer to the *SYS-MAC NE1S Series Operation Manual* (Cat. No. Z901).

For details on I/O allocations for CJ2 CPU Units, refer to the *CJ2 CPU Unit Software User's Manual* (Cat. No. W473).

I/O table settings can be perform online (mostly to create the I/O tables) or they can be performed offline (e.g., editing the I/O tables).

The I/O setting procedures for both online operation and offline operation are described in this section.

6-5-1 Online Operations: Creating, Deleting, Comparing

Creating I/O Tables

Use the following procedure to create I/O tables.

1,2,3...
1. Select ***Controller - Connect*** to connect online to the PLC.
2. Select ***Controller - I/O Table - Create***.
   The real I/O tables will be created.

Deleting I/O Tables

Use the following procedure to delete I/O tables.

1,2,3...
1. Select ***Controller - Connect*** to connect online to the PLC.
2. Select ***Controller - I/O Table - Delete***.
   The real I/O tables will be deleted.

Comparing I/O Tables

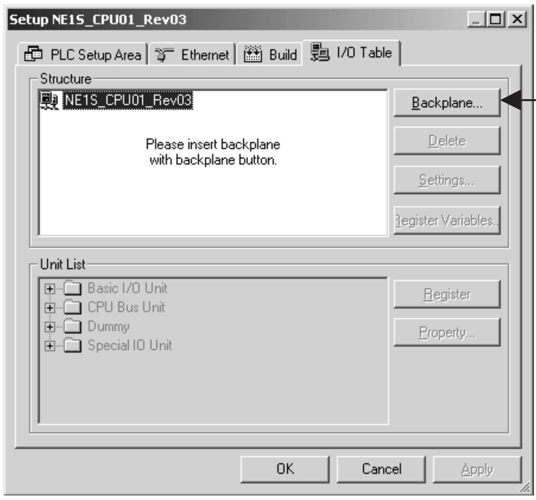
Use the following procedure to compare the real I/O tables and the registered I/O tables.

1,2,3...
1. Select ***Controller - Connect*** to connect online to the PLC.
2. Select ***Controller - I/O Table - Compare***.
   The real I/O tables and the registered I/O tables will be compared.

6-5-2 Offline Operations: Editing I/O Tables

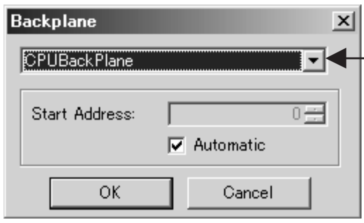
Use the following procedure to edit I/O tables offline.

1,2,3...
1. Click the **I/O Table** Tab from the Configuration Setting Window.
   The following window will be displayed.



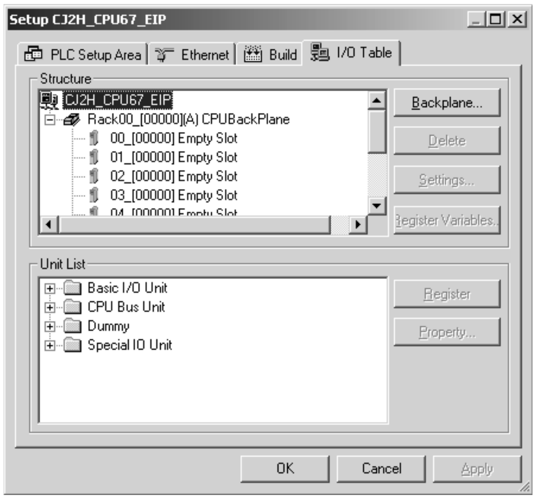
First, click here to add a CPU Backplane. Once the CPU Rack has been completed, click here again to add any Expansion Backplanes needed by the system.

2. Click the **Backplane** Button.
   The Backplane Setting Dialog Box will be displayed.
   A CJ-series PLC (like the CJ2) is a building block PLC. It does not have a Backplane, but a Backplane can be virtually set using settings in the NE Programmer.

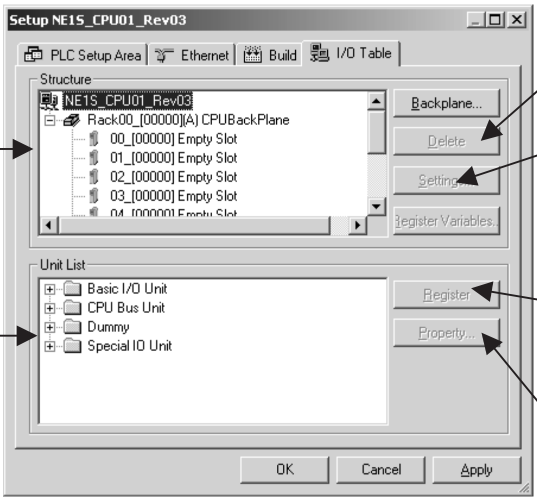


The CPU Backplane is added first. The display will automatically change to Expansion Backplanes after a CPU Backplane is added.

To set a specific start address, clear the check mark from *Automatic* and input the start address for the Rack.
With a CJ2 CPU Unit, the Backplane will be added immediately after the project is created.



3. Click the **OK** Button.
   The following CPU Backplane Window will be displayed.



Deletes a Unit registered in the structure.

Used to make settings for a registered Unit.

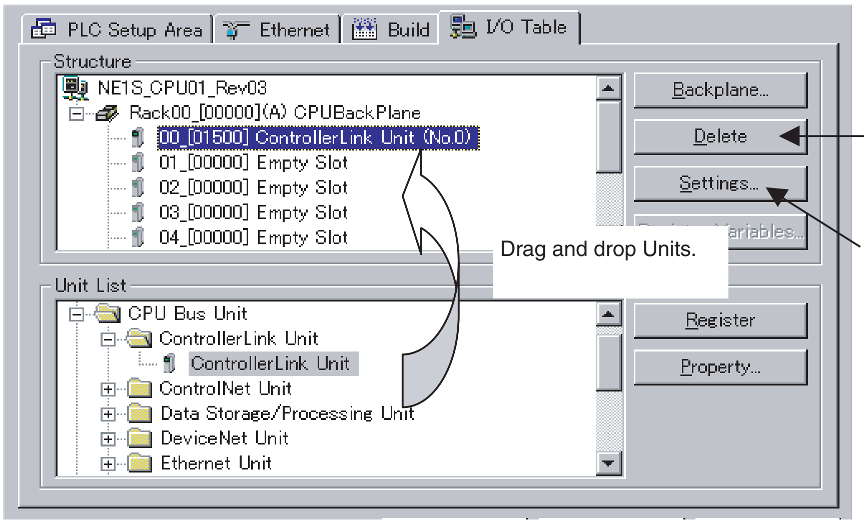
In this example, the first address is set to Automatic, so Rack 00 is assigned.

Drag and drop Units to register them in the Structure at the top of the window.

Select the slot in the Structure Area and the Unit in the Unit List Area and click this button.

Displays the properties of the selected Unit.

4. Drag and drop any of the Units from the *Unit List Area* to an *Empty Unit* in the *Structure* Area.
   The Unit will be registered in the slot.

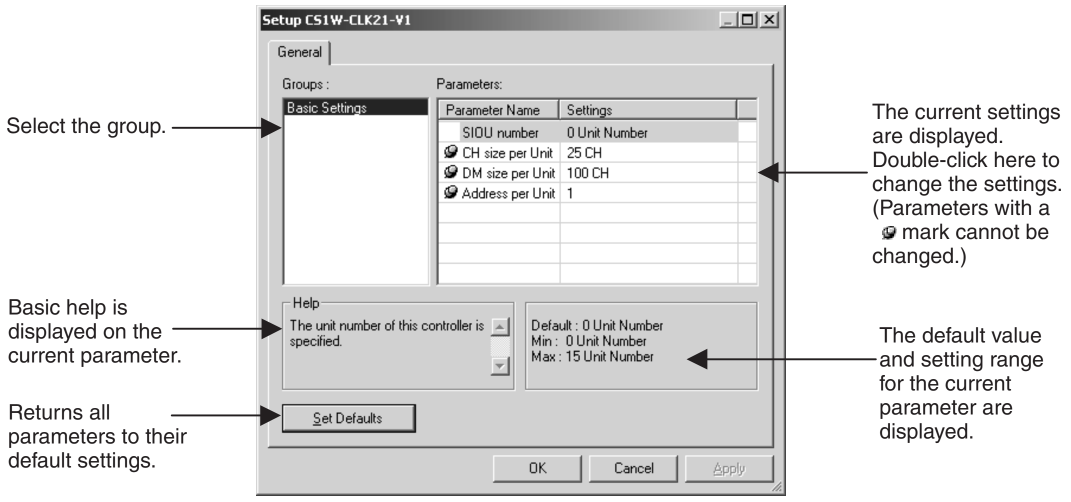


Drag and drop Units.

Deletes a Unit registered in the structure.

Used to make settings for a registered Unit.

5. Settings for the Unit must be made next. Click the **Setting** Button. The Unit Setting Dialog Box will be displayed.



Basic Settings

Parameter Name	Settings
SIOU number	0 Unit Number
● CH size per Unit	25 CH
● DM size per Unit	100 CH
● Address per Unit	1

Help: The unit number of this controller is specified.

Default : 0 Unit Number
Min : 0 Unit Number
Max : 15 Unit Number

Select the group.

Basic help is displayed on the current parameter.

Returns all parameters to their default settings.

The current settings are displayed. Double-click here to change the settings. (Parameters with a ● mark cannot be changed.)

The default value and setting range for the current parameter are displayed.

6. Make all of the required settings and then click the **OK** Button. The I/O Table Tab Page will return.
7. Make all of the required settings and then click the **OK** Button.

SECTION 7
Online Operation

This section provides the procedures for online operation.

7-1 Connecting via Serial Communications (USB/RS-232C)

This section describes how to connect the NE Programmer online to the CPU Unit using serial communications (USB/RS-232C).

The first time USB is used to connect to the CPU Unit, the USB driver will need to be installed. To connect using RS-232C communications, go to *7-1-2*.

7-1-1 Installing the USB Driver

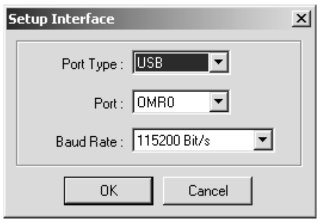
Refer to *3-3 Installing the USB Driver* for details on installing the USB driver.

7-1-2 Connecting Online via USB or RS-232C

Use the following procedure to connect the NE Programmer online to the CPU Unit using USB or RS-232C communications.

1,2,3...

1. Select ***Tool*** – ***Select Interface - NE1S Serial PORT*** to connect to an NE1S-series CPU Unit and select ***Tool - Select Interface - CJ2 USB/Serial Port*** to connect to a CJ2 CPU Unit.
2. Select ***Controller - Connect***. Alternately, press the **Ctrl+W** Keys.
   The following dialog box will be displayed.



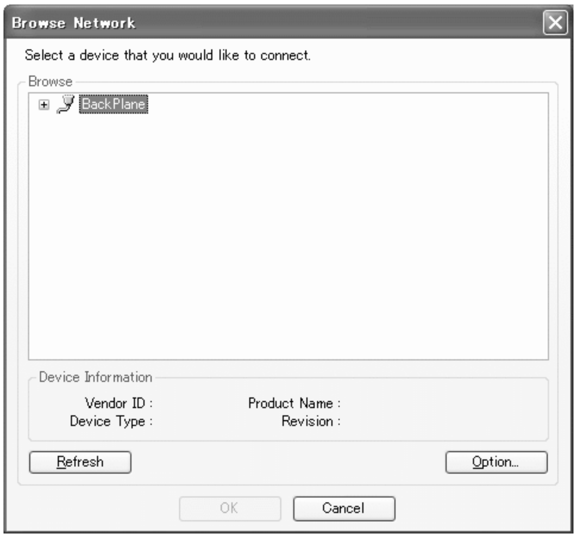
Setup Interface

Port Type : USB

Port : OMR0

Baud Rate : 115200 Bit/s

3. Select the USB or RS-232C port number for the *Serial Port* and then click the **OK** Button.
   The following dialog box will be displayed.



Browse Network

Select a device that you would like to connect.

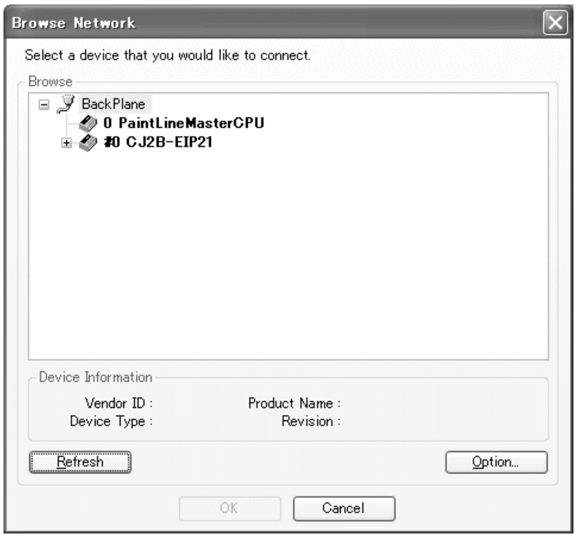
Browse

BackPlane

Device Information

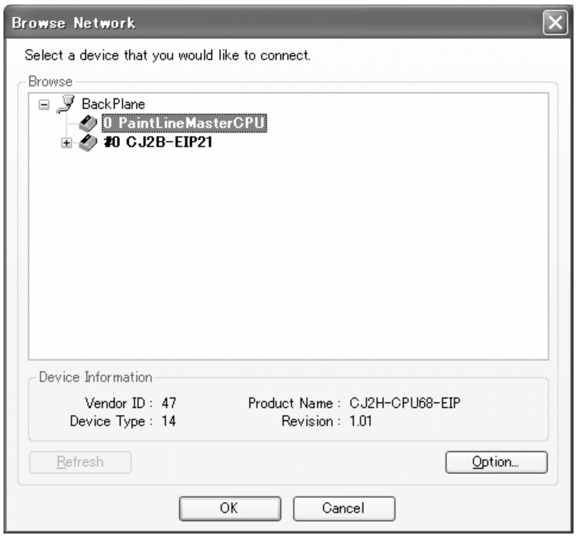
Vendor ID :
Device Type :
Product Name :
Revision :

4. Click the **Refresh** Button.
   The CPU Unit and Communications Unit mounted to the CPU Rack will be displayed as shown below.



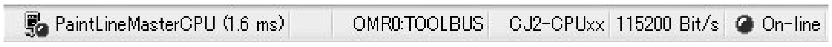
5. Select the CPU Unit and then click the **OK** Button.

   Note This connection path can be saved. (See the note at the end of this procedure.)



Vendor ID : 47
Device Type : 14
Product Name : CJ2H-CPU68-EIP
Revision : 1.01

   If an online connection is made normally, On-line will be displayed in the status bar at the bottom of the window.



PaintLineMasterCPU (1.6 ms) | OMR0:TOOLBUS | CJ2-CPUxx | 115200 Bit/s | On-line

6. To go offline again, select ***Controller - Disconnect***. Alternately, press the **Ctrl+Shift+W** Keys.

Note When connected online, there is a setting in the Browse Network Dialog Box to select whether or not to save the connection path.

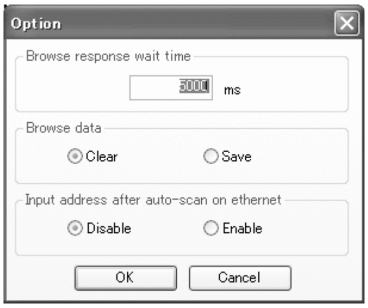
- Clear (default)
  Each time the NE Programmer connects online, it is necessary to browse to a network path.

- Save
  The connection path is displayed from the last time that the NE Programmer was connected. In this case, the window in step 5 is displayed when step 3 is completed, so you can connect online to the last network path just by clicking the **OK** Button.

Use the following procedure to set this option.

1,2,3...

1. Click the **Option** Button in the lower-right corner of the Browse Network Dialog Box. The following window will be displayed.



Option

Browse response wait time: 5000 ms

Browse data: [✓ Clear] Save

Input address after auto-scan on ethernet: [✓ Disable] Enable

2. Select *Clear* or *Save* and click the **OK** Button.

7-2 Connecting Online via Ethernet

This section describes how to connect the NE Programmer online to the CPU Unit using Ethernet.

⚠ **Caution** If the destination node address (IP address) is not set correctly, the NE Programmer may connect unexpectedly to another PLC, and invalid device parameters may be set. Always verify that the correct PLC is connected before downloading data.

7-2-1 Connecting Online via Ethernet

There are several methods that can be used when connecting online to the CPU Unit via Ethernet for the first time. Here, the IP address of the computer will be changed to make the online connection.

To change the IP address of the CPU Unit (i.e., to match an existing network address) after once having connected online to the CPU Unit via Ethernet, refer to the manual for the relevant CPU Unit. (See note.)

1,2,3...

1. Refer to the manual for the relevant CPU Unit for information on the default IP address of an NE1S-series CPU Unit or CJ2 CPU Unit. (See note.) In this example, we will assume the IP address of the computer will be changed to 192.168.200.10.
   Refer to the user's manual for your operating system for the method of changing the IP address of your computer.
2. Connect the CPU Unit and the computer to Ethernet.
3. Turn ON the power supply to the CPU Unit.

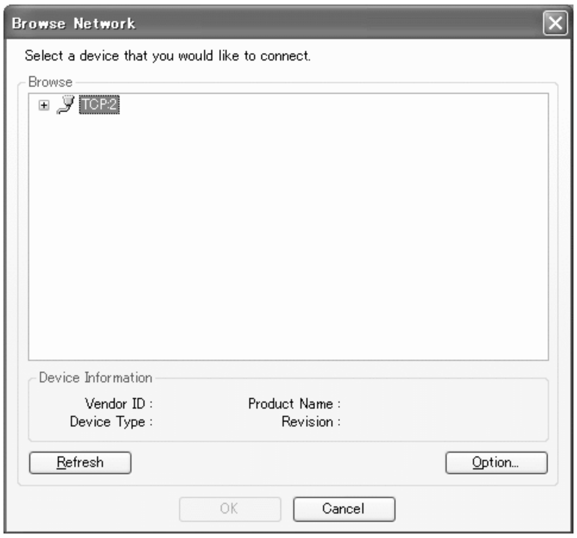
Note Refer to the following manuals for the CPU Units.
NE1S CPU Units:
SYSMAC NE1S Series Operation Manual (Cat. No. Z901)
CJ2 CPU Units:
SYSMAC CJ Series CJ2 CPU Unit Hardware User's Manual (Cat. No. W472)

7-2-2 Connecting Online via Ethernet

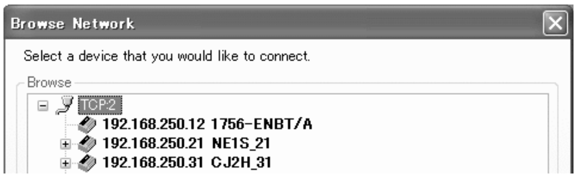
Use the following procedures to connect the NE Programmer online to the CPU Unit using Ethernet.

1,2,3...

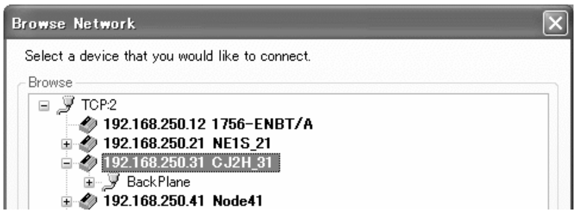
1. Select ***Tool - Select Interface - Ethernet I/F***.
2. Select ***Controller - Connect***. Alternately, press the **Ctrl+W** Keys.
   The following dialog box will be displayed.



3. Click the **Refresh** Button.
   The network will be browsed and the results of browsing will be displayed as shown in the following figure.

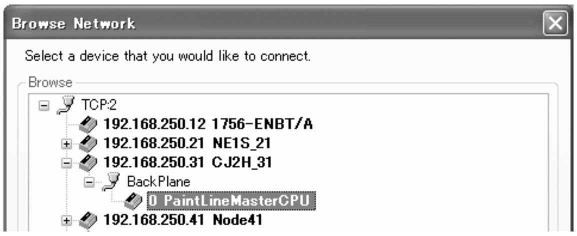

Browse Network
Select a device that you would like to connect.
Browse
TCP:2
192.168.250.12 1756-ENBT/A
192.168.250.21 NE1S_21
192.168.250.31 CJ2H_31

4. Click the expand button (i.e., the + button) for the CPU Unit to expand the options. In the following figure, 192.168.250.31 CJ2H-31 is selected.

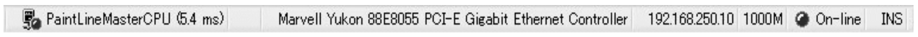


Note With the CJ2, the built-in Ethernet port is also called the EtherNet/IP Unit on the backplane. Therefore, the built-in EtherNet/IP port (program name: CJ2B-EIP21) is given directly under TCP:2. The backplane will be displayed when the built-in EtherNet/IP Unit option is expanded.

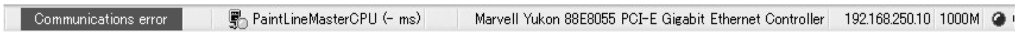
5. A CJ2 CPU Unit or other Unit will be displayed when you expand the backplane option under the built-in EtherNet/IP port. Select the CPU Unit and then click the OK Button.



If an online connection is made normally, *On-line* will be displayed in the status bar at the bottom of the window, as shown in the following figure.


PaintLineMasterCPU (5.4 ms)
Marvell Yukon 88E8055 PCI-E Gigabit Ethernet Controller
192.168.250.10
1000M
On-line
INS

Note *Communications error* will be displayed in the status bar at the bottom of the window as shown in the following figure if a communications error, such as due to a cable disconnection, is detected during online connection.


Communications error | PaintLineMasterCPU (- ms) | Marvell Yukon 88E8055 PCI-E Gigabit Ethernet Controller | 192.168.250.10 | 1000M

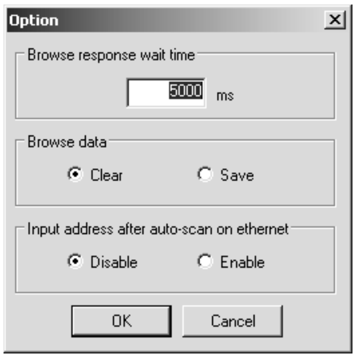
6. To go offline again, select ***Controller - Disconnect***. Alternately, press the **Ctrl+Shift+W** Keys.

Note When connected online, there is a setting in the Browse Network Dialog Box to select whether or not to save the connection path.

- Clear (default)
  Each time the NE Programmer connects online, it is necessary to browse to a network path.
- Save
  The connection path is displayed from the last time that the NE Programmer was connected. In this case, the window in step 5 is displayed when step 3 is completed, so you can connect online to the last network path just by clicking the **OK** Button.

Use the following procedure to set this option.

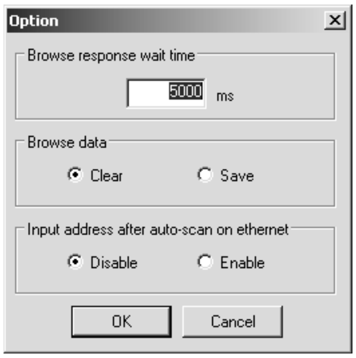
1,2,3... 1. Click the **Option** Button in the lower-right corner of the Browse Network Dialog Box. The following window will be displayed.


Option
Browse response wait time: 5000 ms
Browse data: [✓ Clear] Save
Input address after auto-scan on ethernet: [✓ Disable] Enable

2. Select *Clear or Save in the Browse data* Area and click the **OK** Button.

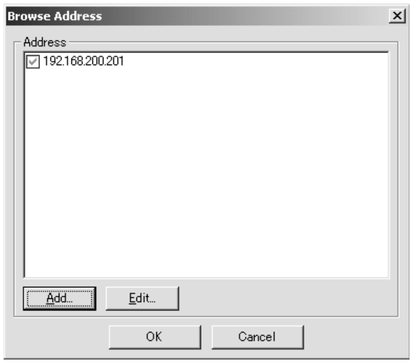
Note To connect online to an NE1S CPU Unit on a different segment, use the following procedure to input the IP address directly.

1. Click the **Option** Button in the Browse Network Dialog Box. The following dialog box will be displayed.

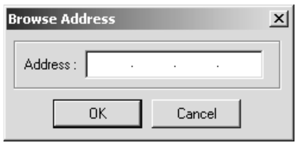

Option
Browse response wait time: 5000 ms
Browse data: [✓ Clear] Save
Input address after auto-scan on ethernet: [✓ Disable] Enable

2. Select *Disable* in the *Input address after auto-scan on ethernet* Area and then click the **OK** Button.

3. Click the **Refresh** Button in the Browse Network Dialog Box.
   The following Browse Address Dialog Box will be displayed along with the IP addresses found in the Browse Network Dialog Box.



4. Click the **Add** Button. The following Browse Address Dialog Box will be displayed.



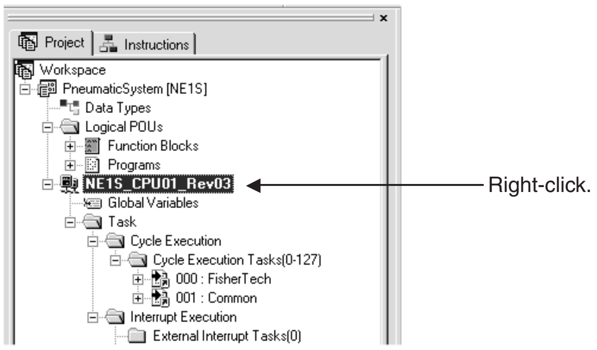
5. Directly input the IP address in the *Address* Field.

7-2-3 Ethernet Setup

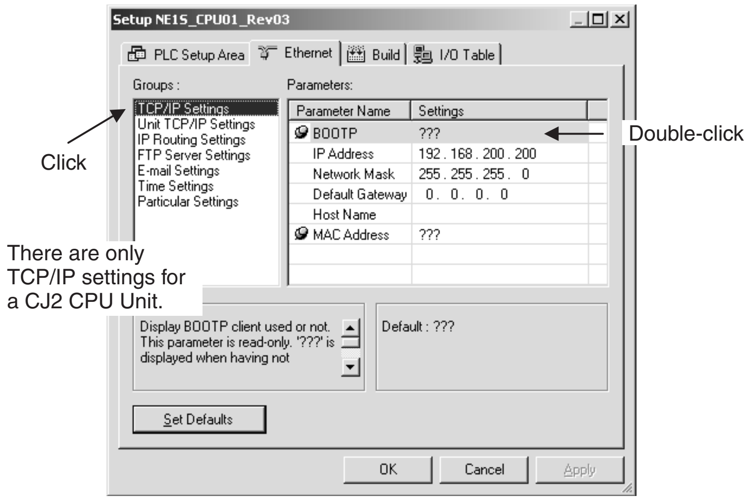
To change the network or host portion of the IP address, first change them in the Ethernet Setup (IP address settings) from the NE Programmer and then download the Ethernet Setup to the CPU Unit.

Refer to the manual for the relevant CPU Unit for information on setting the rotary switches.

1,2,3... 1. Right-click the configuration name (i.e., the PLC name) in the Project Workspace. (Alternately, select ***Controller - System Configuration***.)



2. Select ***System Configuration*** from the popup menu.
   The System Configuration Setting Window shown in step 4 will be displayed.
3. Click the **Ethernet** Button.
4. Select ***TCP/IP Settings*** in the *Group* List.
   The following dialog box will be displayed to set TCP/IP.



Groups:
- TCP/IP Settings
- Unit TCP/IP Settings
- IP Routing Settings
- FTP Server Settings
- E-mail Settings
- Time Settings
- Particular Settings

Parameters:

Parameter Name	Settings
BOOTP	???
IP Address	192 . 168 . 200 . 200
Network Mask	255 . 255 . 255 . 0
Default Gateway	0 . 0 . 0 . 0
Host Name	
MAC Address	???

Display BOOTP client used or not. This parameter is read-only. '???' is displayed when having not

Default : ???

5. Double-click *IP Address* and enter the IP address. If required, set the network mask.
6. Click the **OK** Button.

This completes setting the IP address. Refer to the following procedure *Downloading Ethernet Settings* and transfer the settings that have been made to the CPU Unit.

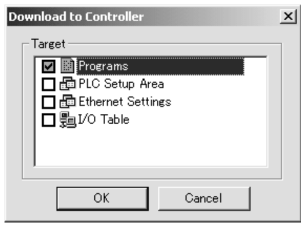
7-2-4 Downloading Ethernet Settings

Ethernet settings can be transferred to the CPU Unit.

Use the following procedure to download the Ethernet settings online to the CPU Unit.

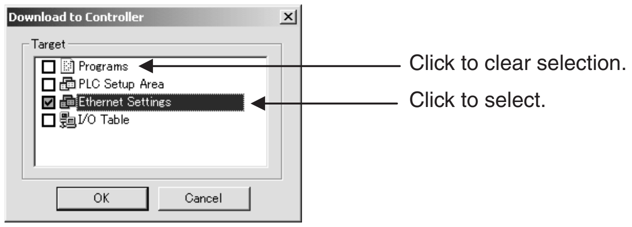
1,2,3...

1. Connect online to the CPU Unit. (A serial connection can be used if desired.)
2. Select ***Controller - Download to Controller.*** Alternately, press the **Ctrl+T** Keys.
   The following dialog box will be displayed.



- ☑ Programs
- ☐ PLC Setup Area
- ☐ Ethernet Settings
- ☐ I/O Table

3. Select only the *Ethernet Settings* option (do not select any other options), and click the **OK** Button.



- ☐ Programs
- ☐ PLC Setup Area
- ☑ Ethernet Settings
- ☐ I/O Table

The Ethernet settings will be downloaded. If the download ends normally, the following dialog box will be displayed.

Note The Ethernet settings are automatically backed up in flash memory inside the CPU Unit.
Do not turn OFF the power supply to the CPU Unit while the settings are being backed up.
The BKUP indicator on the front of the CPU Unit will light during the automatic backup operation.

4. Restart the CPU Unit according to the manual for the relevant CPU Unit.
   SYSMAC NE1S Series Operation Manual (Cat. No. Z901)
   SYSMAC CJ Series CJ2 CPU Unit Hardware User's Manual (Cat. No. W472)

7-3 Automatic Upload Function

7-3-1 Overview

If the personal computer and CPU Unit are directly connected via USB or RS-232C communications, the NE Programmer can be automatically connected online to the CPU Unit. After going online, you can automatically upload and monitor the program as well.

Refer to information starting in *7-6 Uploading, Downloading, and Comparing Programs and Other Data* and *7-8 Monitoring* for details on transferring and monitoring the program.

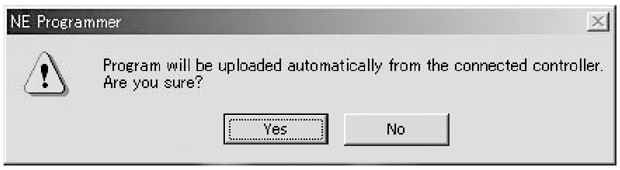
Note
(1) The automatic upload function cannot be used if another application is using the serial port.
(2) The automatic upload function can be used only with a direct serial connection.

7-3-2 Executing the Automatic Upload Function

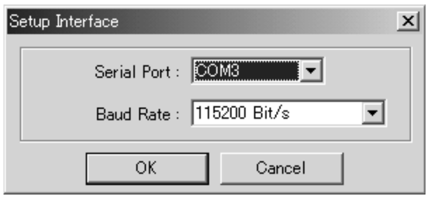
Use the following procedure to execute the automatic upload function.

Note The automatic upload function cannot be used if another application is using the serial port on the personal computer.

1,2,3...
1. Select ***Tool - Select Interface*** and then ***NE1S Serial PORT*** for an NE1S-series CPU Unit or ***CJ2 USB/Serial Port*** for a CJ2 CPU Unit.
2. Select ***Controller - Auto Upload from Controller***.
   The following dialog box will be displayed.


NE Programmer
Program will be uploaded automatically from the connected controller.
Are you sure?

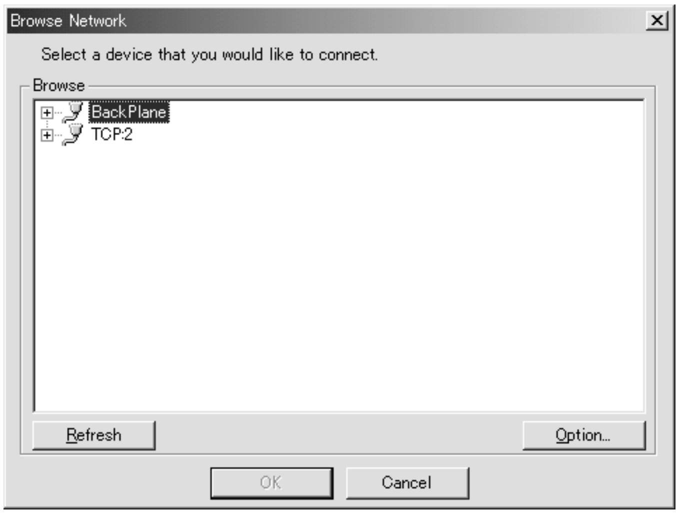
3. Click the **Yes** Button. The Setup Interface Dialog Box will be displayed.


Setup Interface
Serial Port : COM3
Baud Rate : 115200 Bit/s

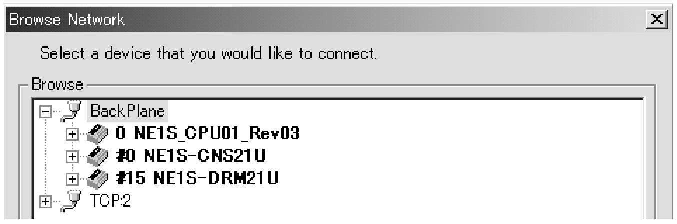
Note In some cases, this dialog box will not be displayed. If it is not displayed, proceed to step 4.

This software retains the interface settings from the previous upload (automatic upload or regular upload). When the **Yes** Button is clicked in step 3, the NE Programmer checks for a connection at this port. If there is a connection, the Setup Interface Dialog Box will not be displayed; the Browse Network Dialog Box (step 4) will be displayed instead.

4. Set the USB port's communications port and baud rate and click the **OK** Button. The Browse Network Dialog Box will be displayed.

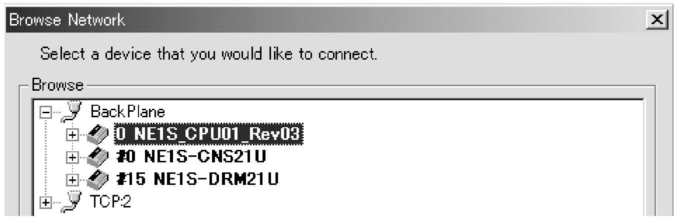


5. Click the **Refresh** Button.
   The CPU Unit and Communications Units mounted in the CPU Rack will be displayed, as shown in the following diagram.



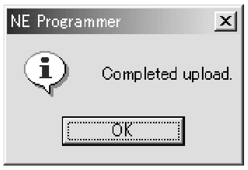
6. Select the ***NE1S-CPU01*** Icon as shown below and then click the **OK** Button.

   Note This connection path can be saved. (See the note at the end of this procedure.)



- If there is unsaved data, a dialog box will be displayed that prompts you to save the data. Follow the directions and save the data.

After connecting online to the CPU Unit, the program, PLC Setup, Ethernet settings, and I/O table will be uploaded from the CPU Unit. The following dialog box will be displayed when the data has been uploaded.


NE Programmer

Completed upload.

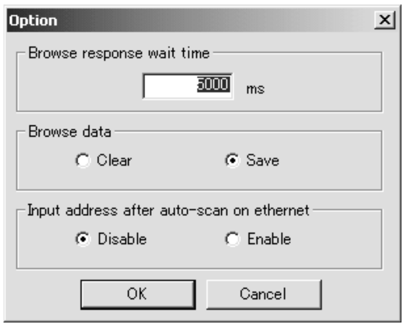
7. Click the **OK** Button.

Note When connected online, there is a setting in the Browse Network Dialog Box to select whether or not to save the connection path.

- Clear (default)
  Each time the NE Programmer connects online, it is necessary to browse to a network path.
- Save
  The connection path is displayed from the last time that the NE Programmer was connected. In this case, the window in step 5 is displayed when step 3 is completed, so you can connect online to the last network path just by clicking the **OK** Button.

Use the following procedure to set this option.

1,2,3... 1. Click the **Option** Button in the lower-right corner of the Browse Network Dialog Box. The following window will be displayed.


Option

Browse response wait time: 5000 ms

Browse data: ( ) Clear (●) Save

Input address after auto-scan on ethernet: (●) Disable ( ) Enable

2. Select *Clear* or *Save* and click the **OK** Button.

7-4 Changing the CPU Unit That Is Connected

7-4-1 Overview

Use this function to connect to a different CPU Unit when the NE Programmer is already connected online to a CPU Unit. When the CPU Unit connection is changed, the automatic upload function will be executed after connecting to the new CPU Unit.

There are two ways to change the CPU Unit:

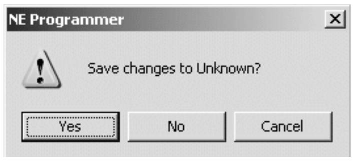
- Changing to a CPU Unit on the same network. For example, changing to a CPU Unit connected to the same Ethernet network as the CPU Unit that is currently connected. Refer to *7-4-2*.
- Changing to a CPU Unit on a different network. For example, changing to a CPU Unit connected to an Ethernet network when a CPU Unit is currently connected via USB. Refer to *7-4-3*.

7-4-2 Changing to a CPU Unit on the Same Ethernet Network

Use the following procedure to change the connection to a CPU Unit connected to the same Ethernet network as the CPU Unit that is currently connected.

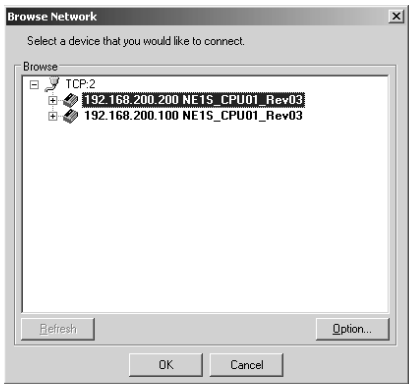
1,2,3...

1. Select ***Controller - Change Controller***.
   The following dialog box will appear if there is any unsaved data.



Save changes to Unknown?

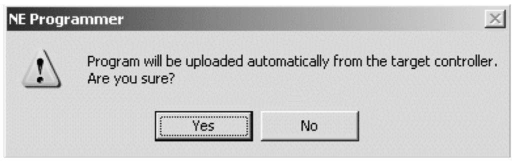
2. Click the **Yes** Button and save the data on the NE Programmer if the data is required.
   Click the **No** Button is the data is not required.
   - The Browse Network Dialog Box will be displayed.



Select a device that you would like to connect.
192.168.200.200 NE1S_CPU01_Rev03
192.168.200.100 NE1S_CPU01_Rev03

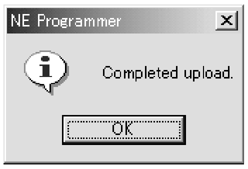
   - If the IP address of the CPU Unit to be connected is not displayed, click the **Refresh** Button.

3. Select the IP address of the CPU Unit to be connected.
4. Click the **OK** Button.
   The following dialog box will be displayed.



Program will be uploaded automatically from the target controller. Are you sure?

5. Click the **Yes** Button.
   The automatic upload function will be performed after connecting to the specified CPU Unit. The following dialog box will be displayed if the automatic upload function is executed normally.



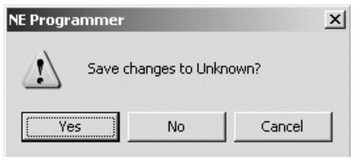
Completed upload.

7-4-3 Changing to a CPU Unit on a Different Network

Use the following procedure to change the connection to a CPU Unit connected to an Ethernet network when a CPU Unit is currently connected via USB.

1,2,3...

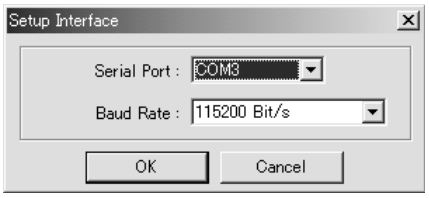
1. Select ***Controller - Change Controller***.
   The following dialog box will be displayed if there is any unsaved data.



NE Programmer

Save changes to Unknown?

2. Click the **Yes** Button and save the data on the NE Programmer if the data is required.
   Click the **No** Button is the data is not required.
   - The Setup Interface Dialog Box will be displayed.

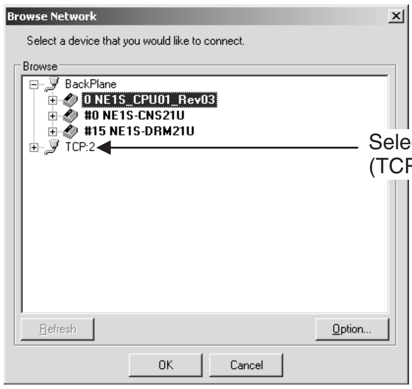


Setup Interface

Serial Port : COM3

Baud Rate : 115200 Bit/s

3. The serial port and baud rate being used for the current connection will be displayed. Check the settings and then click the **OK** Button.
   The Browse Network Dialog Box will be displayed.



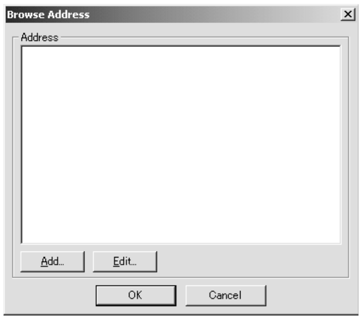
Browse Network

Select a device that you would like to connect.

Browse
- BackPlane
  - 0 NE1S_CPU01_Rev03
  - #0 NE1S-CNS21U
  - #15 NE1S-DRM21U
- TCP:2

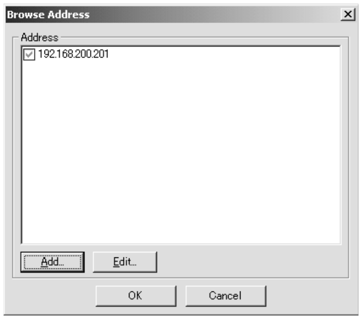
Select the Ethernet port (TCP2) to connect externally.

4. Select ***TCP:2*** (Ethernet port) and then click the **Refresh** Button.
   The Browse Address Dialog Box will be displayed.

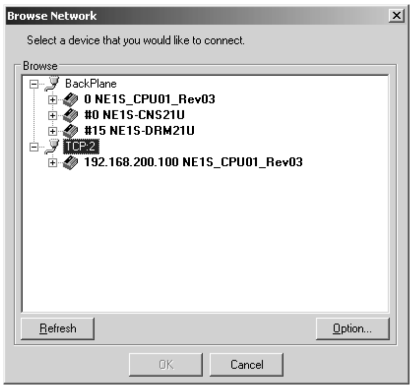


Browse Address

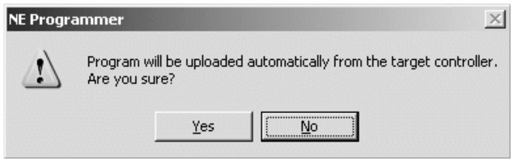
5. Click the **Add** Button and input the IP address in the dialog box that will be displayed. The Browse Address Dialog Box will return.
   The IP address that was input will be displayed as shown below.



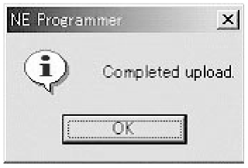
6. Click the **OK** Button.
   The IP address that was input will be displayed in the Browse Network Dialog Box.


Select a device that you would like to connect.

7. Select the IP address of the CPU Unit to be connected.
8. Click the **OK** Button.
   The following dialog box will be displayed.


Program will be uploaded automatically from the target controller.
Are you sure?

9. Click the **Yes** Button.
   The automatic upload function will be performed after connecting to the specified CPU Unit. The following dialog box will be displayed if the automatic upload function is executed normally.


Completed upload.

7-5 Online Operations for I/O Tables

This section describes creating, deleting, and comparing I/O tables online.

7-5-1 I/O Tables

I/O tables are tables that list the models and locations of the Units mounted in the PLC.

The I/O tables registered in the CPU Unit are used to allocate I/O memory in the CPU Unit to real I/O (i.e., Basic I/O Units), Special I/O Units, and CPU Bus Units.

Always create I/O tables and register them in the CPU Unit after adding any new Unit to the PLC or after removing any Unit from the PLC.

7-5-2 Real I/O Tables and Registered I/O Tables

Real I/O Tables

The real I/O tables are tables that are created by the CPU Unit in the PLC to show the Units actually mounted in the PLC.

When the I/O table creation operation is performed, the real I/O tables are registered in the CPU Unit as the registered I/O tables. The real I/O tables cannot be changing from the NE Programmer.

Registered I/O Tables

The registered I/O tables in the CPU Unit are used by the CPU Unit when allocating I/O. Use either of the following procedures to register I/O tables in the CPU Unit.

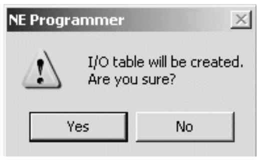
- Execute the I/O table creation operation online to register the real I/O tables in the CPU Unit as the registered I/O tables.
- Create and edit I/O tables offline and then transfer them to the CPU Unit.

7-5-3 Creating I/O Tables

Use the following procedure to create the real I/O tables online.

1,2,3...

1. Connect online to the CPU Unit.
2. Select ***Controller - I/O Table - Create***.
   The following dialog box will be displayed.


NE Programmer

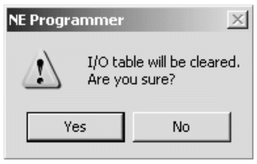
I/O table will be created.
Are you sure?

3. Click the **Yes** Button.
   The real I/O tables will be registered in the CPU Unit as registered I/O tables.

7-5-4 Deleting I/O Tables

Use the following procedure to delete the registered I/O tables online.

1,2,3...
1. Connect online to the CPU Unit.
2. Select ***Controller - I/O Table - Delete***.
   The following dialog box will be displayed.



NE Programmer

I/O table will be cleared.
Are you sure?

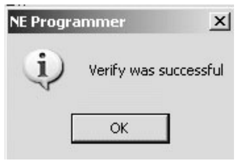
3. Click the **Yes** Button.
   The registered I/O tables will be created.

7-5-5 Verifying I/O Tables

Use the following procedure to compare the registered I/O tables and the real I/O tables online.

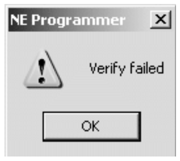
1,2,3...
1. Connect online to the CPU Unit.
2. Select ***Controller - I/O Table - Compare.***

The registered I/O tables will be compared to the real I/O tables and the results will be displayed.



NE Programmer

Verify was successful



NE Programmer

Verify failed

7-6 Uploading, Downloading, and Comparing Programs and Other Data

7-6-1 Overview

The following data can be uploaded to the NE Programmer, downloaded to the CPU Unit, or compared between the CPU Unit and NE Programmer: I/O tables, PLC Setup, and Ethernet Setup.

The CPU Unit operating modes during which uploading, downloading, and comparing are possible are shown in the following table.

Menu command	RUN	MONITOR	PROGRAM
Controller - Upload from Controller	Yes	Yes	Yes
Controller - Download to Controller	No	No	Yes
Controller - Compare with Controller	Yes	Yes	Yes

⚠ Caution Confirm safety at the destination node before editing or transferring a program, PLC Setup, I/O table, I/O memory data, or parameter data to another node. Doing either of these without confirming safety may result in unexpected operation and injury.

⚠ Caution Before actual operation, check the parameter settings and user program (such as the ladder program) for proper execution in trial operation. Always check the program before transferring it.

⚠ Caution Always clear the memory of a CJ2 CPU Unit before downloading programs, the PLC Setup, or the I/O tables from the NE Programmer. If the memory is not cleared before downloading the data, unexpected operation may occur in the controlled system.

Note (1) Confirm that the controlled system will not be adversely affected before changing the operating mode of the CPU Unit.

(2) Valuable programs may be lost if the direction of program transfer is not correct. Double-check the transfer direction before transferring data.

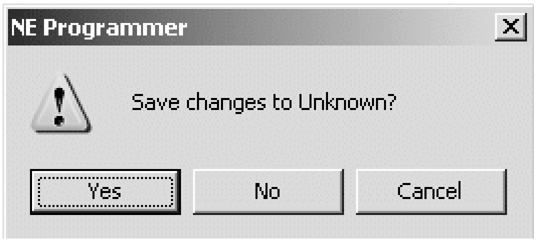
Perform all of the following operations online. Before starting, connect the NE Programmer online to the CPU Unit.

7-6-2 Uploading

Use the following procedure to upload any of the following from the CPU Unit to the NE Programmer: Program, PLC Setup, Ethernet settings, and I/O tables.

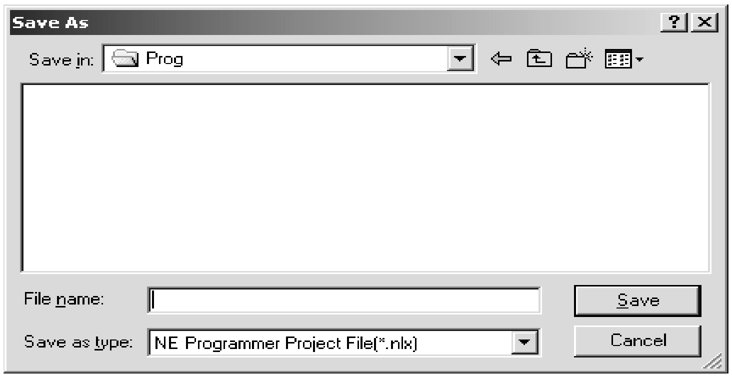
1,2,3...

1. Select ***Controller - Upload from Controller***.
   A confirmation dialog box will be displayed asking if you want to save the project.

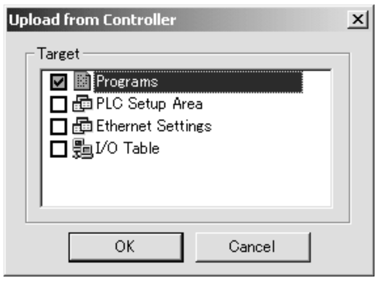

NE Programmer

Save changes to Unknown?

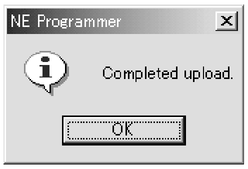
2. To save the current project, click the **Yes** Button, enter the project file name, and then click the **Save** Button.



3. Select the data to be uploaded and then click the **OK** Button.



4. The data selected in step 3 will be uploaded from the CPU Unit to the NE Programmer.
   The following dialog box will be displayed if the data is uploaded normally.



Completed upload.

5. Click the **OK** Button.

Upload Protection for Programs

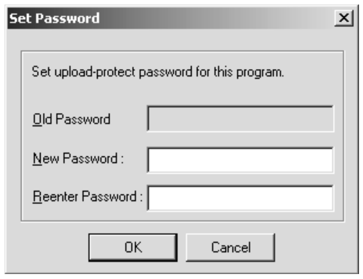
It is possible to set an upload protection password for the program. Upload protection will be enabled when a program in a computer that has a set password is downloaded to the PLC.

This enables prohibiting uploading of a program with a set password if the password does not match when it is checked. The following describes the basic operation.

- Only programs can be upload-protected. The PLC Setup, Ethernet settings, and I/O table settings can be uploaded and edited.
- The password will not be checked if the password in the computer does not match the password in the PLC.
- Downloading and online editing can be performed and settings can be changed regardless of whether there is a password.

Setting the Password

Right-click the configuration in the Workspace and select ***Program Upload Protect (P)*** from the pop-up menu. The following dialog box will be displayed to set the password.



Set upload-protect password for this program.

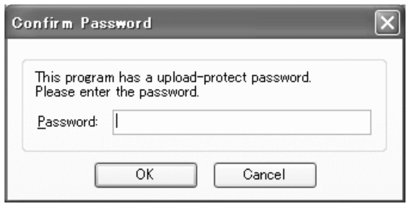
Input the password and then click the **OK** Button.

Releasing the Password

Input the current password into the *Old Password* Field, leave the *New Password* Field and *Reenter Password* Field blank, and then click the **OK** Button.

Checking the Password at Upload

The following dialog box will be displayed to check the password if an upload protection password has been set and the password does not match when it is checked. The password will be verified when upload is performed from the Controller or verification is performed with the Controller.


Confirm Password

This program has a upload-protect password.
Please enter the password.

Password:

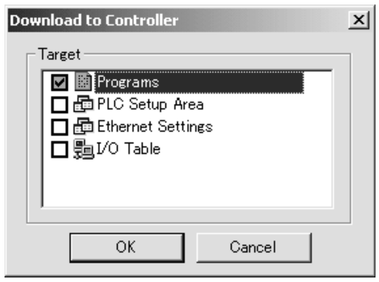
OK Cancel

7-6-3 Downloading

Use the following procedure to download any of the following from the NE Programmer to the CPU Unit: Program, PLC Setup, Ethernet settings, and I/O tables.

1,2,3...

1. Select ***Controller - Download to Controller***.
   The following dialog box will be displayed.

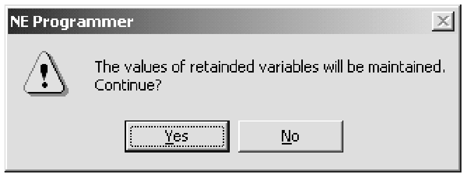

Download to Controller

Target

- ☑ Programs
- ☐ PLC Setup Area
- ☐ Ethernet Settings
- ☐ I/O Table

OK Cancel

2. Select the data to be downloaded and then click the **OK** Button.
   The following dialog box will be displayed if there are already retained variables in the CPU Unit with the same data type as retained variables in a program being downloaded.

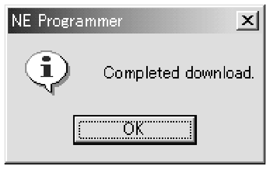

NE Programmer

The values of retainded variables will be maintained.
Continue?

Yes No

Refer to *Preserving the Values of Retained Variables* below for details on preserving the PVs of retained variables.

3. Select the data to be downloaded and then click the **OK** Button.
   The selected data selected will be downloaded from the NE Programmer to the CPU Unit. The following dialog box will be displayed if the data is downloaded normally.


NE Programmer

Completed download.

OK

4. Click the **OK** Button.

Preserving the Values of Retained Variables

If there are already retained variables in the CPU Unit with the same data type as retained variables in a program being downloaded, the retained variables in the CPU Unit can be preserved.

Note Observe the following precautions.

- The force-set/force-reset status of automatically allocated variables will not be preserved.
- The force-set/force-reset status of automatically allocated variables is cleared when the CPU Unit is switched from PROGRAM mode to MONITOR or RUN mode.
- If the program is downloaded when automatically allocated variables have been force-set/force-reset, the variables may have unexpected force-set/force-reset status after the download is completed. We recommend clearing the forced status of all automatically allocated variables before downloading the program.
- Network I/O variables will be held even if they are set to not be held.

This list shows various requirements for the PVs of retained variables to be preserved after downloading. There is a table following the list with examples for cases *1 to *10.

- Variables in the downloaded program are set to be retained and automatically allocated. Variables with address specifications or non-retained variables are not preserved. (*1)
- The variable names are being used in the program in the Controller before the download. (*2, *3, *4)
- The variables and data types in the Controller before the download match the ones in the program being downloaded. (*5)
    - For structures, the data types match for each member. Also, members are valid even if the member's declaration location changed. (*6)
    - For array variables, it depends on whether the number of elements increased or decreased. If the number of elements decreased, all variables will be valid. If the number of elements increased, only the number of elements before the download will be valid. (*7)
- There are no changes to task allocations or nesting of function block instances. (*8, *9, *10)
- The variables are not written from another source while the download is in progress.

When a variable has been overwritten from another application, the variable may be returned to its previous value.

	Status	Information in program (downloaded information)	Information in Controller (information before download)	PV status after download
1	Retain/Non-retain and Address specification	BOOL VarA (with address specification)	BOOL VarA (automatic allocation/retain/non-retain)	Not retained
		BOOL VarA (with address specification)	BOOL VarA (with address specification)	Not retained
		BOOL VarA (automatic allocation/retain)	BOOL VarA (with address specification)	Retained
		BOOL VarA (automatic allocation/retain)	BOOL VarA (automatic allocation/non-retain)	Retained
		BOOL VarA (automatic allocation/non-retain)	BOOL VarA (automatic allocation/retain)	Not retained
2	Variable exists in both project and Controller.	VarA	VarA	Retained
3	Variable does not exist in Controller.	VarA	None	Not retained
4	Variable does not exist in project.	None	VarA	Not retained
5	Data types match.	BOOL VarA	BOOL VarA	Retained
6	Data types do not match.	BOOL VarA	WORD VarA	Not retained
		DWORD VarA	WORD VarA	Not retained

		TYPE stA:
		STRUCT
		Member1: BOOL;
		Member2: WORD;
		Member3: UINT;
		Member4: DWORD;
		END_STRUCT;
		END_TYPE
		VAR
		VarA: stA;
		END_VAR;
			TYPE stA:
			STRUCT
			Member2: BOOL;
			Member1: WORD;
			Member3: UDINT;
			END_STRUCT;
			END_TYPE
			VAR
			VarA: stA;
			END_VAR;
				Retained
				Retained
				Not retained
				Not retained

		TYPE stA:
		STRUCT
		Member2: BOOL;
		Member1: WORD;
		Member3: UDINT;
		END_STRUCT;
		END_TYPE
		VAR
		VarA: stA;
		END_VAR;
			TYPE stA:
			STRUCT
			Member1: BOOL;
			Member2: WORD;
			Member3: UINT;
			Member4: DWORD;
			END_STRUCT;
			END_TYPE
			VAR
			VarA: stA;
			END_VAR;
				Retained
				Retained
				Not retained
				Not retained

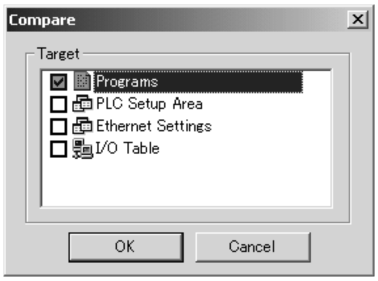
7	Number of array elements is different.	BOOL VarA[3] VarA[0] VarA[1] VarA[2]	BOOL VarA[3] VarA[0] VarA[1]	Retained Retained Not retained
		BOOL VarA[3] VarA[0] VarA[1]	BOOL VarA[3] VarA[0] VarA[1] VarA[2]	Retained Retained Not retained

	Status	Information in program (downloaded information)	Information in Controller (information before download)	PV status after download
8	Task allocation was changed.	\Cyclic task (0 to 127) 0 \VarA	\Cyclic task (0 to 127) 1 \VarA	Not retained
9	FB instance name was changed.	\Cyclic task (0 to 127) 0 \InstA\VarA	\Cyclic task (0 to 127) 0 \InstB\VarA	Not retained
10	FB instance nesting was changed.	\Cyclic task (0 to 127) 0 \InstA\VarA	\Cyclic task (0 to 127) 0 \InstA\InstB\VarA	Not retained

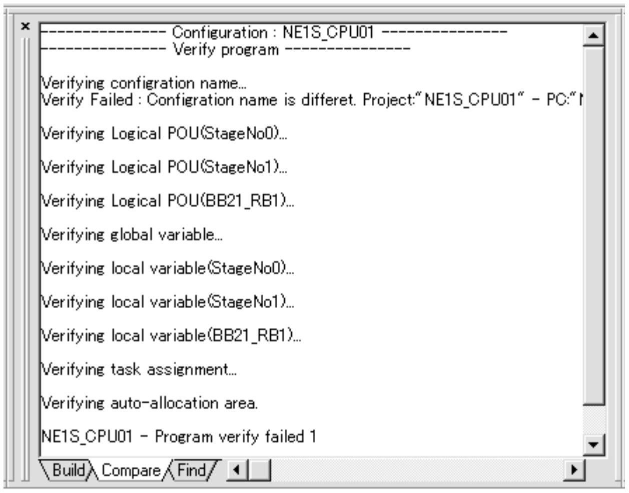
7-6-4 Comparing

Use the following procedure to copy any of the following between the NE Programmer and the CPU Unit: Program, PLC Setup, Ethernet settings, and I/O tables.

1,2,3... 1. Select ***Controller - Compare with Controller***.
The following dialog box will be displayed.



2. Select the data to be compared and then click the **OK** Button.
The selected data selected will be compared between the NE Programmer and the CPU Unit.
The results of comparison will be displayed on the Compare Tab Page of the Output Window as shown below.



```
--------------- Configuration : NE1S_CPU01 ---------------
--------------- Verify program ---------------

Verifying configration name...
Verify Failed : Configration name is differet. Project:"NE1S_CPU01" - PC:"

Verifying Logical POU(StageNo0)...

Verifying Logical POU(StageNo1)...

Verifying Logical POU(BB21_RB1)...

Verifying global variable...

Verifying local variable(StageNo0)...

Verifying local variable(StageNo1)...

Verifying local variable(BB21_RB1)...

Verifying task assignment...

Verifying auto-allocation area.

NE1S_CPU01 - Program verify failed 1
```

7-7 Changing the Operating Mode

Use the following menu commands to change the operating mode of the CPU Unit from the NE Programmer.

Note Confirm that the controlled system will not be adversely affected before changing the operating mode of the CPU Unit.

Operating mode	Menu command
PROGRAM	Controller - Operating Mode - Program
MONITOR	Controller - Operating Mode - Monitor
RUN	Controller - Operating Mode - Run

7-8 Monitoring

7-8-1 Overview

The monitoring function enables ladder program execution to be monitored in a window.

The NE Programmer supports the following two types of monitoring.

- Monitoring status and present values on the Ladder Editor
- Monitoring present values of specified variables (or addresses) in the I/O memory of the CPU Unit in the Watch Window.

The following operations are used in the above monitoring windows.

- Status/PV Monitor
- Force ON/Force OFF
- Differential Monitor (detecting ON to OFF and OFF to ON transitions in bit status)
- PV Change
- Program Change while Monitoring: Refer to *7-14 Online Editing*.
- Timer/Counter SV Change: Refer to *7-14 Online Editing*.

⚠ **Caution** Confirm safety sufficiently before starting to monitor status or PVs on the Ladder Editor or in the Watch Window.
Operating errors, e.g., of shortcut keys that result in forcing ON or OFF bits or turning ON or OFF bits may result in operating errors in the controlled system connected to Output Units regardless of the operating mode of the CPU Unit.

Relationship between Monitoring Types and Operations

Operation	Type of monitoring		Operating mode of CPU Unit
	Ladder Editor	Watch Window	
Status/PV Monitor	Yes	Yes	All modes
Force ON/Force OFF	Yes	Yes	All modes except RUN
Differential monitor	Yes	Yes	All modes
PV Change for timers, counters, I/O areas, DM Area, or EM Area	Yes	Yes	All modes except RUN
Online editing (Including Timer/Counter SV Change)	Yes	No	All modes except RUN

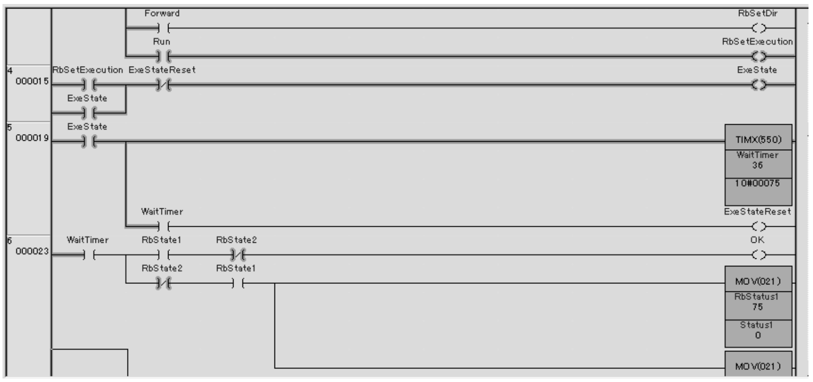
7-8-2 Starting Monitoring Functions

When the NE Programmer is connected online to a CPU Unit, monitoring will automatically be started in the Ladder Editor and Watch Window.

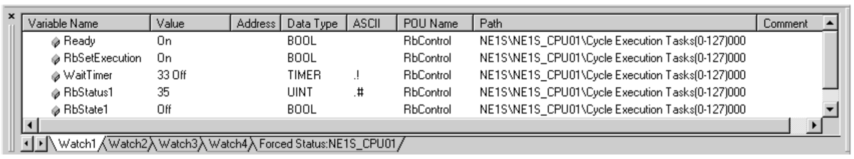
Note
(1) The program in the NE Programmer and the program in the CPU Unit must be the same to enable monitoring. If they are not the same, transfer the program. For details on transferring programs, refer to *7-6 Uploading, Downloading, and Comparing Programs and Other Data*.

(2) The automatic upload function can be used to automatically upload the program from the CPU Unit and start monitoring. (The automatic upload function can be used only when the NE Programmer and CPU Unit are connected directly via USB or RS-232C communications. Refer to *7-3 Automatic Upload Function* for details on the automatic upload function.

1,2,3...
1. Connect the NE Programmer online to the CPU Unit.
2. Double-click the task to be monitored in the Project Window to display the Ladder Editor.
   Monitoring will automatically be started in the Ladder Editor and Watch Window.



- ON execution status (i.e., the power flow) in the ladder diagram will be indicated in light green. (Light green is the default color.)
- If the CPU Unit is in MONITOR or RUN mode and the current task is active, the background will be light blue. (Light blue is the default color.)



Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
Ready	On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbSetExecution	On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
WaitTimer	33 Off		TIMER	.!	RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbStatus1	35		UINT	.#	RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbState1	Off		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	

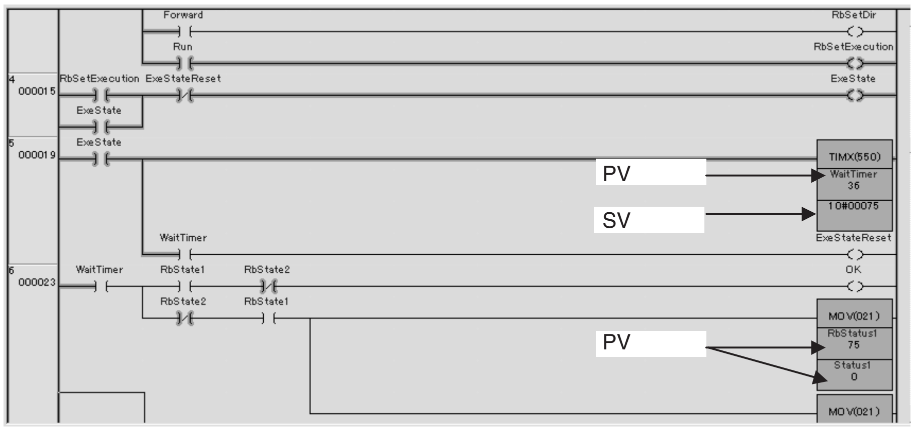
- If the Watch Window is not currently displayed, select ***View - Window - Watch***.

Operations that can be performed in the Ladder Editor or Watch Window during monitoring are described in the following sections.

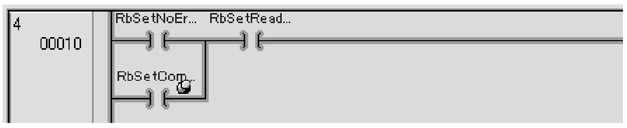
7-8-3 Monitoring in the Ladder Editor

In online status, double-click the task to be monitored in the Project Window to display the Ladder Editor.

The following Ladder Editor Status Monitor Window will be displayed.



- PV: Present value, SV: Set value
- Depending on the operating status of the CPU Unit, ON execution status (i.e., the power flow) in the ladder diagram will be indicated in light green. (Light green is the default color.)
- If the CPU Unit is in MONITOR or RUN mode, the background will be light blue. (Light blue is the default color.)
- To change the display colors, select ***Tool - Option*** to display the Integrated Options Window, click the **Ladder** Icon, and click the **Color** Tab.
- The icon shown below will be displayed to indicated forced status if a bit is forced ON or forced OFF.



Note The following indications are used for operand data for function blocks that are being monitored.

NE Programmer version 1.53 and higher	NE Programmer version 1.60 and lower
Decimal: Number after 10# Hexadecimal: Number after 16#	Decimal: Number after plus or minus sign Hexadecimal: HEX after number
Examples	
10#1234 16#1234	+1234 1234 HEX

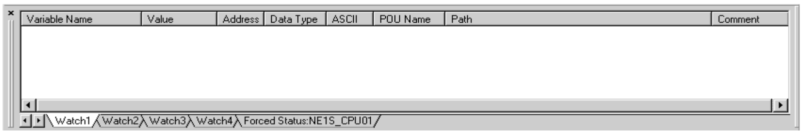
Constants are input using “10#” and “16#”.

7-8-4 Monitoring in the Watch Window

Monitoring by Registering Variables (Bits or Words) in the Editor

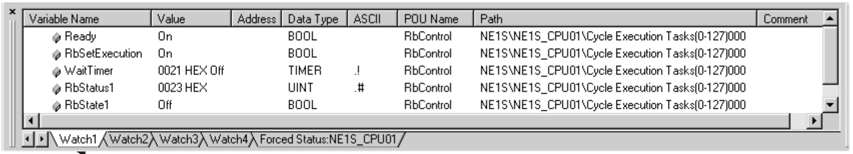
1,2,3... 1. In online status, double-click the task to be monitored in the Project Window to display the Ladder Editor.

2. If the Watch Window is not currently displayed, select ***View - Window - Watch***.
   The Watch Window will be displayed.



3. In the Ladder Editor, select one or more variables (bits or words) to be monitored, right click, and select **Add to Watch** from the popup menu.
   The specified bit or word will be added to the Watch Window and its present value will be automatically monitored as shown below.

   Example of Setting Multiple Variables



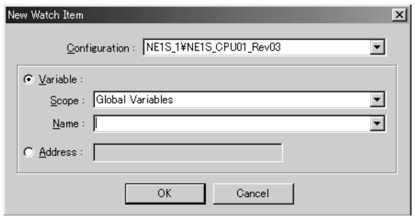
Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
Ready	On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbSetExecution	On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
WaitTimer	0021 HEX Off		TIMER	.!	RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbStatus1	0023 HEX		UINT	.#	RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbState1	Off		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	

Right-click the tab to select *Create Tab*, *Delete Tab*, *Move*, or *Rename.*

- *"S On"* will be displayed for bits that are forced ON and *R Off* will be displayed for bits that are forced ON.
- If the project is saved, information on the variables that have been added to the Watch Window will be saved as well.
- The tab pages from *Watch 1* to *Watch 4* can be used to group the variables that are being monitored.
- Any variables that have been forced ON or OFF will be displayed on the *Force Status* Tab Page.
  Execute ***Update Force Status*** the first time the Force Status Tab Page is selected. (A dialog box will be displayed automatically when the tab is clicked.)
  Once a bit has been forced ON or OFF, it will remain on the Force Status Tab Page even if its forced status is cleared. To remove a bit from the Force Status Tab Page, right-click the Watch Window and select ***Update Status***.
- If multiple BOOL , TIMER, or COUNTER variables have been selected, all the variables can be set, reset, force-set, or force-reset.

Monitoring by Registering Variables (Bits or Words) in the Watch Window

1,2,3... 1. Right-click the Watch Window and select ***Add to Watch*** from the popup menu.
   The ***New Watch Item*** Dialog Box will be displayed.



New Watch Item

Configuration : NE1S_1¥NE1S_CPU01_Rev03

[✓ Variable :]

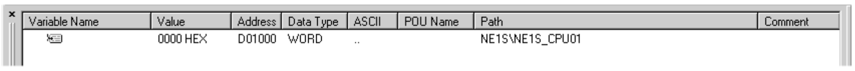
Scope : Global Variables

Name :

Address :

2. To register a variable, select the *Variable* Option, input the variable name, and then click the **OK** Button.
   To register a bit or word, select the Address Option, input the address, and then click the **OK** Button.
   The specified variable (bit or word) will be added to the Watch Window and the present value will be monitored.

   Example when *D100* Is Designated



Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
	0000 HEX	D01000	WORD	..		NE1S\NE1S_CPU01	

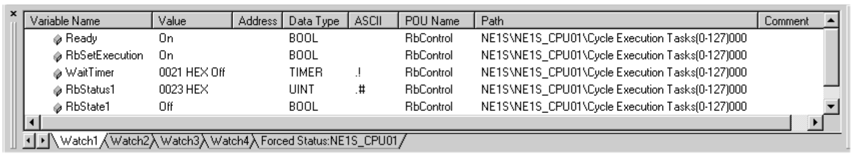
3. Repeat the previous step to monitor multiple variables (bits or words).

   Note Consecutive addresses (e.g., D101, D102 . . .) can be easily input when watch items are repeatedly added after registering addresses of bits or words.

Changing the Display Format

To change the display format, select the items in the Watch Window (multiple items can be selected), right-click, and select ***Monitoring Data Type*** and then ***Monitor in Hex, Decimal, Signed Decimal, Binary Number,*** or ***Data Type*** from on the popup menu.

Example when *Data Type* Is Selected

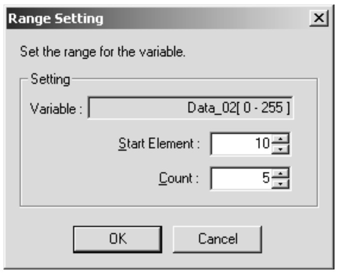


Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
Ready	On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbSetExecution	On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
WaitTimer	0021 HEX Off		TIMER	.!	RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbStatus1	0023 HEX		UINT	.#	RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
RbState1	Off		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	

- The display format for PVs of items in the Watch WIndow can be individually specified by selecting the following commands from the View Menu or toolbar.
  Select ***View - Monitoring Data Type*** and then ***Monitor in Hex***, ***Decimal, Signed Decimal***, ***Binary Number***, or ***Data Type***.

Setting the Watch Range

A dialog box will be displayed to ask the display range if the number of elements for the specified array is 256 or higher.



Range Setting

Set the range for the variable.

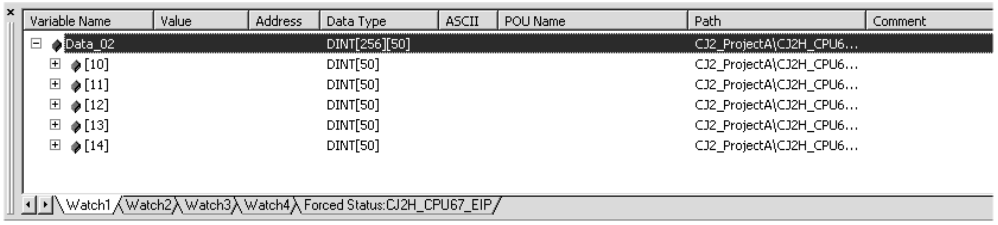
Setting

Variable : Data_02[0 - 255]

Start Element : 10

Count : 5

Input the number of the start element and the number of elements to register, and then click the **OK** Button.



Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
⊟ Data_02			DINT[256][50]			CJ2_ProjectA\CJ2H_CPU6...	
⊞ [10]			DINT[50]			CJ2_ProjectA\CJ2H_CPU6...	
⊞ [11]			DINT[50]			CJ2_ProjectA\CJ2H_CPU6...	
⊞ [12]			DINT[50]			CJ2_ProjectA\CJ2H_CPU6...	
⊞ [13]			DINT[50]			CJ2_ProjectA\CJ2H_CPU6...	
⊞ [14]			DINT[50]			CJ2_ProjectA\CJ2H_CPU6...	

7-9 Saving and Restoring Variable PVs

7-9-1 Function

This function reads all of the variable PVs saved in the CPU Unit and saves the data in a CSV file; it also restores the saved variable to the CPU Unit.

All of the variables are saved and restored together, including global, local, retained, and non-retained variables.

⚠WARNING Restore variables only after confirming that doing so will not adversely affect the system.

Note Observe the following precautions.

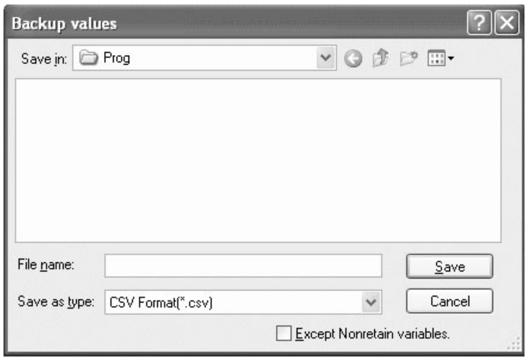
- The force-set/force-reset status of automatically allocated variables will not be preserved.
- The force-set/force-reset status of automatically allocated variables is cleared when the CPU Unit is switched from PROGRAM mode to MONITOR or RUN mode.

7-9-2 Procedure

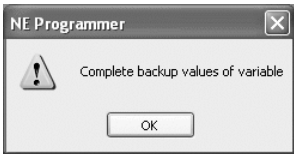
Saving PVs

1,2,3...

1. Select ***Controller - Backup value of variables***. The Backup values Dialog Box will be displayed.



2. Specify the file location and file name and click the **Save** Button. The following dialog box will be displayed if the save operation was completed normally.


NE Programmer

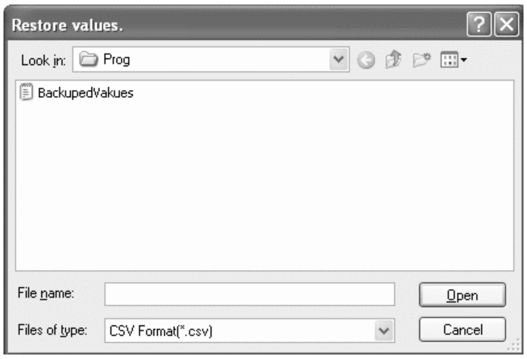
Complete backup values of variable

OK

Restoring PVs

1,2,3...

1. Select ***Controller - Restore value of variables***. The Restore values Dialog Box will be displayed.



Restore values.

Look in: Prog

BackupedVakues

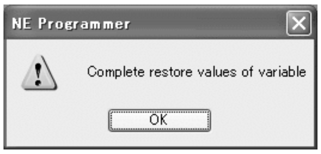
File name:

Files of type: CSV Format(*.csv)

Open

Cancel

2. Specify the source file and click the **Open** Button. The following dialog box will be displayed if the restore operation was completed normally.

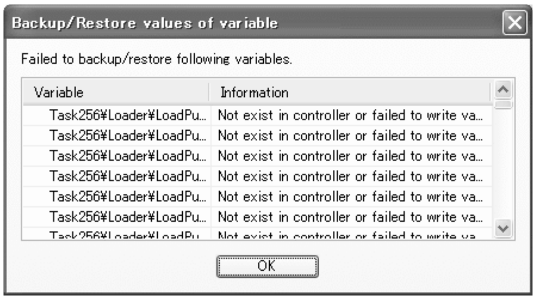


NE Programmer

Complete restore values of variable

OK

If none of the saved variables exist in the CPU Unit, the following error message will be displayed.



Backup/Restore values of variable

Failed to backup/restore following variables.

Variable	Information
Task256¥Loader¥LoadPu...	Not exist in controller or failed to write va...
Task256¥Loader¥LoadPu...	Not exist in controller or failed to write va...
Task256¥Loader¥LoadPu...	Not exist in controller or failed to write va...
Task256¥Loader¥LoadPu...	Not exist in controller or failed to write va...
Task256¥Loader¥LoadPu...	Not exist in controller or failed to write va...
Task256¥Loader¥LoadPu...	Not exist in controller or failed to write va...
Task256¥Loader¥LoadPu	Not exist in controller or failed to write va

OK

7-10 Forcing Bits ON and OFF (Force-set and Force-reset)

7-10-1 Overview

Force ON/Force OFF

When the CPU Unit is online in MONITOR or PROGRAM mode, input bits, output bits, and timer/counter Completion Flags can be forced ON or OFF from the Ladder Editor or Watch Window.

The forced status is maintained until it is cleared or until it is forced ON or OFF again. Forced status will not change regardless of external input status or the results of program execution.

Force ON/Force OFF

When the CPU Unit is online in MONITOR or PROGRAM mode, input bits, output bits, and timer/counter Completion Flags can be turned ON or OFF from the Ladder Editor or Watch Window.

The set status, however, is not forced and will change if the external input status changes or if the status changes as a result of ladder program execution. This is the difference between turning bits ON or OFF normally and forcing bits ON or OFF.

Note

(1) Confirm that the controlled system will not be adversely affected before changing the status of any bit in memory, including turning bits ON and OFF, forcing bits ON and OFF, and resetting forced status.

(2) The status of bits forced ON and OFF while online will be maintained even after the NE Programmer is taken offline.

(3) Before going offline, check the forced status on the Force Status Tab Page and clear the forced status.

(4) Do not unintentionally go offline while there is forced status remaining in the CPU Unit.

(5) The program in the NE Programmer and the program in the CPU Unit must be the same to enable forcing bits ON or OFF. If they are not the same, upload the program.

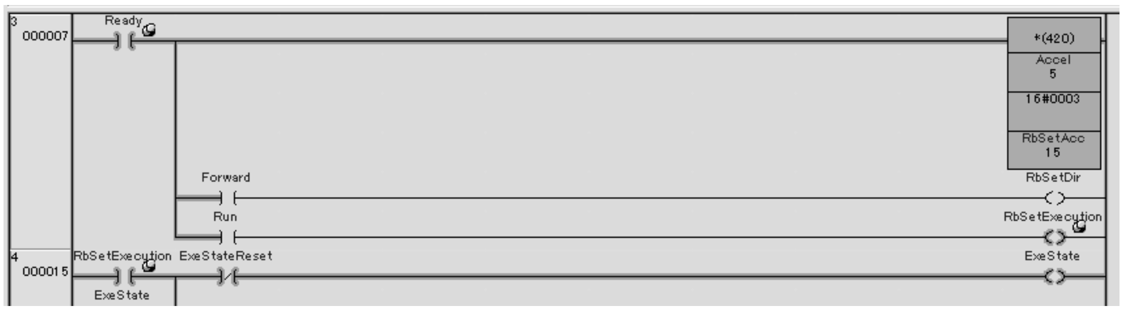
An option in the Integrated Options Window's General Tab Page must be selected in order to allow write operations (PV change, force-set/reset, and set/reset operations) on array variables specified with an index, such as A[i]. To enable these write operations, select ***Tool - Option*** to display the Integrated Options Window, click the **General** Icon, and select the *Permit a write operation to the Array element with index variable, e.g., A[i].* Option.

Note An array variable's index value is based on the index value the last time that the value was monitored, so a write operation may operate on a different array element if the variable's index value was changed since the last write operation.

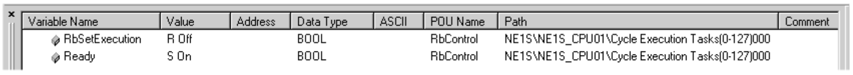
7-10-2 Turning Bits ON/OFF, Forcing Bits ON/OFF, and Clearing Forced Status

To change the status of a bit, select it in the Ladder Editor or Watch Window and select one of the following from the ***Set*** Menu: ***On, Off, Force On, Force Off, Release Force Status***.

Example for Force ON/Force OFF



• The icon shown below will be displayed on the Ladder Editor to indicate forced status if a bit is forced ON or forced OFF.



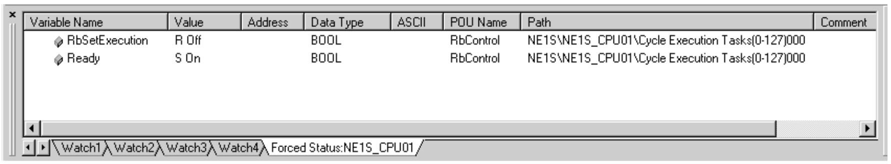
Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
RbSetExecution	R Off		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
Ready	S On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	

• In the Watch Window, *S On* will be displayed for bits that are forced ON and *R Off* will be displayed for bits that are forced ON.

Note When bits are turned ON or OFF, the icon, *S On*, and *R Off* will not be displayed in the Ladder Editor or Watch Window and only the normal display for ON and OFF status will be displayed.

7-10-3 Forced Status Display

Click the **Force Status** Tab in the Watch Window to display all the forced statuses for connected CPU Units. The following screen shows an example of the forced status display.



Variable Name	Value	Address	Data Type	ASCII	POU Name	Path	Comment
RbSetExecution	R Off		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	
Ready	S On		BOOL		RbControl	NE1S\NE1S_CPU01\Cycle Execution Tasks(0-127)000	

The forced status can be read from the CPU Unit using the following two methods:

- Click the **Force Status** Tab the first time the NE Programmer is started.
- Select ***Scan Force Status*** after right-clicking the Force Status Tab.

When the forced status for the connected CPU Unit is changed by the NE Programmer, the change is immediately shown on the Force Status Tab Page. When the forced status is changed from another NE Programmer installation, the change will not be shown until the forced status is refreshed with the most recent data.

7-11 Changing the PVs of Variables

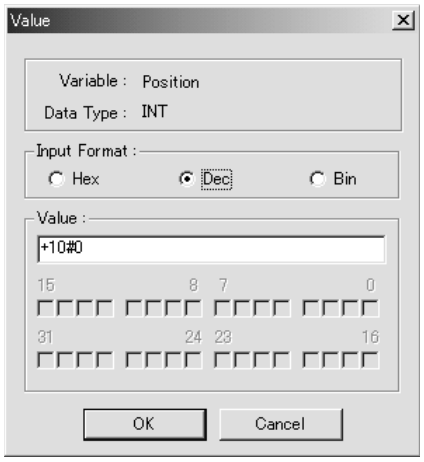
When the CPU Unit is online in MONITOR or PROGRAM mode, the present values of words can be changed from the Watch Window or the Ladder Editor.

Any present values that are changed, however, are not forced and will change if the external input status changes or if the status changes as a result of ladder program execution.

Note Confirm that the controlled system will not be adversely affected before changing a present value.

1,2,3...

1. Right-click the PV to be changed in the Watch Window and select ***Set - Value*** from the popup menu.
   The Value Dialog Box will be displayed.



Variable : Position
Data Type : INT
Input Format : Hex [✓ Dec] Bin
Value : +10#0

2. Input the desired value and then click the **OK** Button.
   The specified value will be displayed in the Watch Window and Ladder Editor.

An option in the Integrated Options Window's General Tab Page must be selected in order to allow write operations (PV change, force-set/reset, and set/reset operations) on array variables specified with an index, such as A[i]. To enable these write operations, select ***Tool - Option*** to display the Integrated Options Window, click the **General** Icon, and select the *Permit a write operation to the Array element with index variable, e.g., A[i].* Option.

Note An array variable's index value is based on the index value the last time that the value was monitored, so a write operation may operate on a different array element if the variable's index value was changed since the last write operation.

7-12 Changing Timer/Counter Set Values

When the CPU Unit is online in MONITOR or PROGRAM mode, timer and counter set values can be changed with an online editing operation. Timer and counter set values can also be changed from constants to external settings such as a CIO word, Work word, or DM word.

Note Before changing a timer or counter set value, verify that the change will not adversely effect the system or equipment.

Supported Instructions

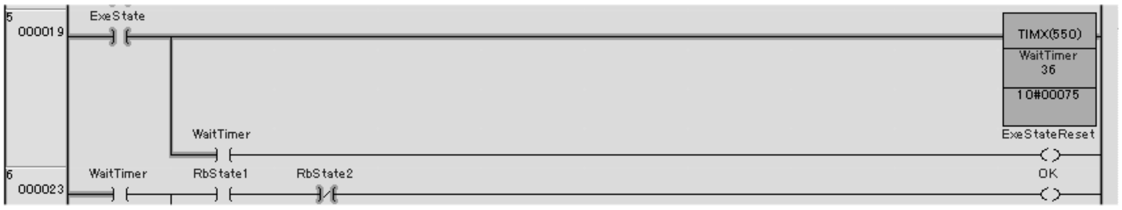
Set values can be changed in the following instructions.

Timer/Counter Operation Mode: Binary Mode

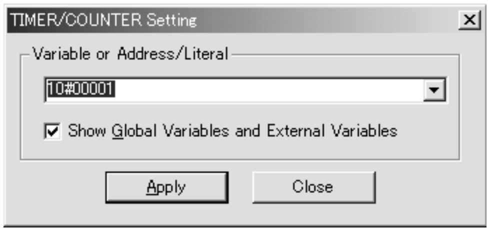
Timer instructions: TIMX, TIMHX(551), TIMLX(553), TMHHX(552), and TTIMX(555)
Counter instructions: CNTX and CNTRX(548)

The following example procedure shows how to change a timer's set value.

1,2,3... 1. Monitor the rung containing the timer that will be edited.



2. Select the TIMX instruction's set value, right-click, and select ***Set - Timer/ Counter Setting Value*** from the popup menu. The following dialog box will be displayed.


TIMER/COUNTER Setting
Variable or Address/Literal
10#00001
Show Global Variables and External Variables
Apply
Close

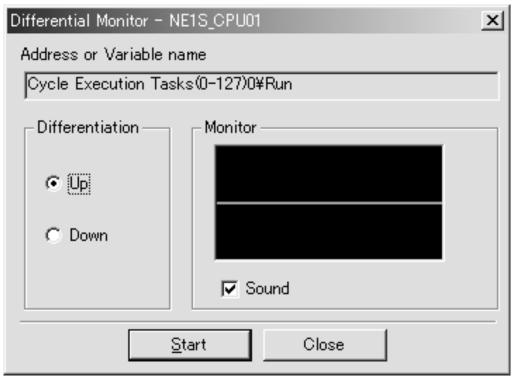
3. Change the variable or address/value and click the **Apply** Button. The set value will be changed. (The timer set value can be changed repeatedly until the **Close** Button is clicked.)

7-13 Differential Monitor

ON to OFF and OFF to ON transitions in bits can be detected online from the Ladder Editor and Watch Window.

1,2,3...

1. Right-click the variable for differential monitoring in the Watch Window or Ladder Editor and select ***Differential Monitor*** from the popup menu.
   The Differential Monitor Dialog Box will be displayed.



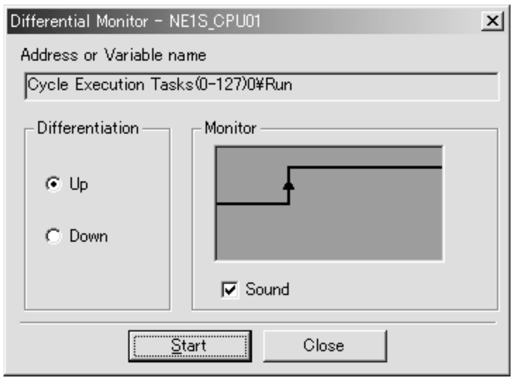
Differential Monitor - NE1S_CPU01

Address or Variable name: Cycle Execution Tasks(0-127)0¥Run

Differentiation: (●) Up ( ) Down

Monitor: [✓] Sound

2. Select the *Differentiation Up/Down* and then click the **Start** Button.
   When the differentiation condition is met, the display will change as shown below for an Up condition.



Differential Monitor - NE1S_CPU01

Address or Variable name: Cycle Execution Tasks(0-127)0¥Run

Differentiation: (●) Up ( ) Down

Monitor: [✓] Sound

7-14 Online Editing

7-14-1 Online Editing

When the CPU Unit is online in MONITOR or PROGRAM mode, multiple ladder rungs can be edited simultaneously from the Ladder Editor while monitoring status.

Note With NE1S-CPU01 Rev. 3.0 or earlier revisions, multiple ladder rungs cannot be edited even if there are line comments included with the rungs.

Broadly speaking, the following two types of online editing operations are supported.

- Editing or deleting the selected rung
- Inserting rungs before or after the selected rung

The following editing functions are supported for online editing.

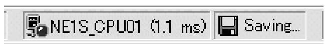
- Inserting, deleting, and changing basic instructions
- Inserting, deleting, and changing special instructions
- Adding external or internal variables
- Changing differentiated instructions (setting/releasing upward or downward differentiation or adding/deleting/changing differentiated instructions)

- Adding global variables
- Adding, changing, and deleting function block parameters

⚠ **Caution** Confirm that the controlled system will not be adversely affected even if the cycle time is increased before performing online editing. An increased cycle time may prevent input signals from being read.

Note

(1) When online editing is performed, changes will be written first to the normal RAM in the CPU Unit and then they will be backed up in the flash memory in the CPU Unit. Do not turn OFF the power supply to the CPU Unit while the settings are being backed up to flash memory (i.e., while the BKUP indicator is lit). The following display will appear in the status bar while data is being written to flash memory.



NE1S_CPU01 (1.1 ms) Saving...

(2) If the CPU Unit and NE Programmer do not contain the same program, and error message will be displayed and online editing will not be possible.

(3) To may major corrections to the program or to move rungs, edit the program offline and then download it to the CPU Unit.

(4) Do not edit the program in a way that would cause the maximum cycle time set in the PLC Setup to be exceeded.
If the maximum cycle time is exceeded, a cycle time exceeded error will occur and operation will stop.

If a cycle time exceeded error occurs, operation will stop.

- To start operation again, switch to PROGRAM mode and then return to RUN or MONITOR mode.

If a cycle time exceeded error persists, do the following.

- Change the program or increase the maximum cycle time setting.
- Perform the operation to clear error displays.

The CPU Unit will stop for a short period of time when any of the following instructions is inserted or deleted.
JMP, JME, or END

(5) Refer to *7-15 Clearing Errors* and *7-18 Displaying Errors and the Error Log* for information on clearing error displays.

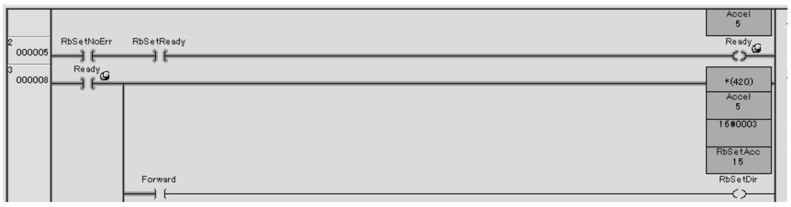
(6) There must be at least one END instruction in a program. Leave at least one END instruction in the program when performing online editing.

7-14-2 Online Editing Procedures

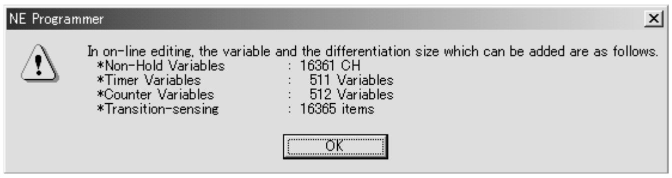
The following example shows how to input an OR using online editing.

1,2,3...

1. While monitoring the program, display the rung where online editing is to be performed.



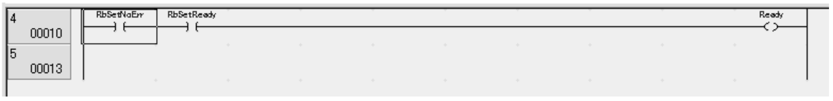
2. Right-click the rung to be edited and select ***Online Edit - Begin*** from the popup menu. It does not matter what portion of the rung is selected. In this example, rung 4 will be edited.
   The following dialog box will be displayed.


NE Programmer

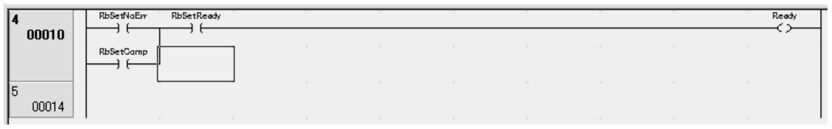
In on-line editing, the variable and the differentiation size which can be added are as follows.
*Non-Hold Variables : 16361 CH
*Timer Variables : 511 Variables
*Counter Variables : 512 Variables
*Transition-sensing : 16365 items

Note The number of differentiations that can be added will not be displayed for program POUs of a CJ2 CPU Unit.

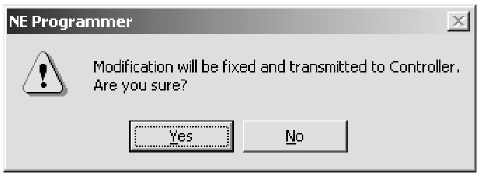
3. Click the **OK** Button.
   The selected rung will be displayed with a yellow background as shown below. (Yellow is the default color.)



4. Edit the rung. Editing methods are the same as for offline operations.
   In this example, an OR will be added for RbSetComp for the NC input condition at the beginning of the above rung.

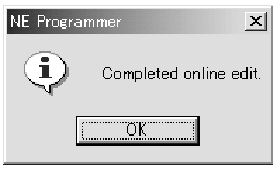


5. Right-click the Edit Window and select ***Online Edit - Finish*** from the popup menu.
   A confirmation dialog box will be displayed.


NE Programmer

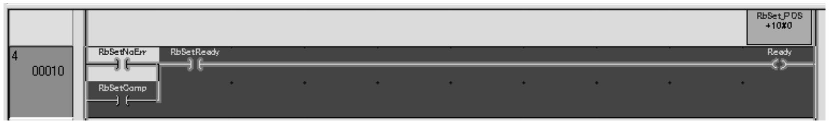
Modification will be fixed and transmitted to Controller.
Are you sure?

6. Click the **Yes** Button.
   The changes will be downloaded and the following dialog box will be displayed when the download has been finished.


NE Programmer

Completed online edit.

7. Click the **OK** Button.
   The monitor window will be displayed with the changed applied.



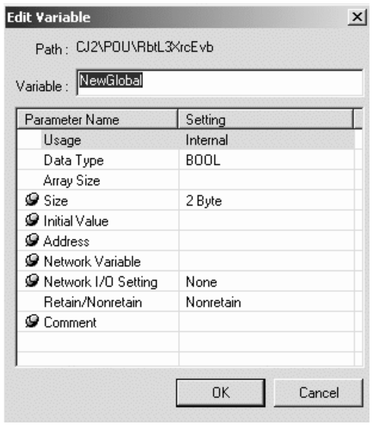
This completes the online editing.

7-14-3 Adding Global Variables

Use the following procedure to add global variables in online editing. The restrictions that apply are also given below.

Procedure

1,2,3... 1. When a new variable is added during online editing, the following Edit Variable Dialog Box will be displayed.

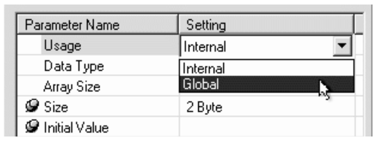

Edit Variable

Path : CJ2\POU\RbtL3XrcEvb

Variable : NewGlobal

Parameter Name	Setting
Usage	Internal
Data Type	BOOL
Array Size	
Size	2 Byte
Initial Value	
Address	
Network Variable	
Network I/O Setting	None
Retain/Nonretain	Nonretain
Comment	

2. Change the *Usage* setting to *Global*.



Parameter Name	Setting
Usage	Internal
Data Type	Internal
Array Size	Global
Size	2 Byte
Initial Value	

   Addresses and network variables can be set for global variables.

3. Click the **OK** Button. The new variable will be registered as a global variable. The new variable will also be registered as an external variable.

Precautions

The added global variable cannot be edited or deleted during online editing.

- The addition of a new variable to the global and external variables can be cancelled during online editing by ending global editing.
- When online editing has been completed and changes have been transferred, it is necessary to close the connection, edit/delete the global variables, build the program, and download the project.

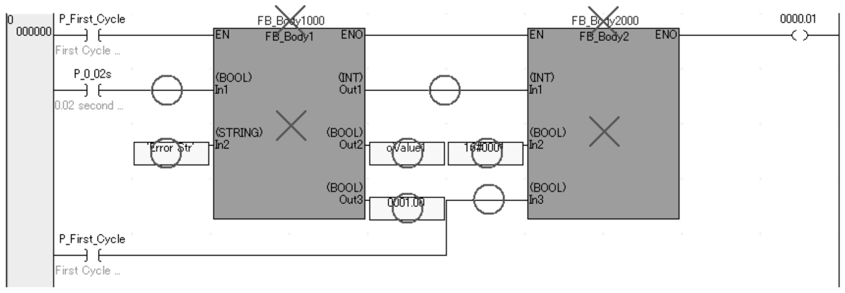
7-14-4 Editing Function Block Parameters Online

Function block parameters (i.e., the inputs and outputs of the function block; refer to the following figure) can be edited online. The following operations can be edited.

- Changing from a variable (e.g., VarA) to a different variable (e.g., VarB)
- Changing from a variable to a constant (e.g., 10#1234) or an address (e.g., 0000.00)
- Changing from a constant or address to a constant or address
- Changing from a constant or address to a variable or address
- Deleting a variable or constant to (i.e., changing from allocate to omit)
- Adding a variable, constant, or address (i.e., changing from omit to allocate)
- Creating direct connections

    Connection lines from the left bus bar to function block input variables
    Connection lines from function block output variables to function block input variables
- Deleting direct connection lines

The parameters shown with circles in the following figure can be edited.

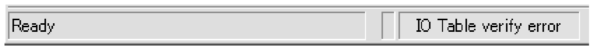


7-15 Clearing Errors

Use the following procedure to clear error displays.

1,2,3...
1. If an error is displayed, first remove the cause of the error.
2. Select ***Controller - Clear Error***. Alternately, press the **F4** Key.
   Errors, however, cannot be cleared when another Programming Devices has the access right or the CPU Unit is in RUN mode.

The information will be displayed in the status bar when an error occurs.


Ready | IO Table verify error

Information will also be display on the Error Tab Page of the Error Log Window.

Refer to *7-18 Displaying Errors and the Error Log* for information on errors and the Error Log Window.

Controller - Clear Error is the same function as that executed for the **Clear** Button on the Error Tab Page in the Error Log Window.

7-16 Clearing Memory

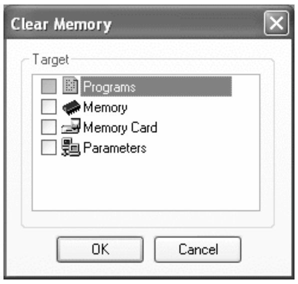
Memory in the CPU Unit can be cleared in the following units.

- All user programming (multiple programs)
- I/O Memory Area
- Parameter Area

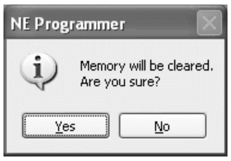
Perform all of the following operations online. Before starting, connect the NE Programmer online to the CPU Unit.

1,2,3...

1. Select ***Controller - Clear Memory***.
   The following dialog box will be displayed.



2. Select the data to be cleared and then click the **OK** Button.
   The following dialog box will be displayed.



Memory will be cleared.
Are you sure?

3. Click the **Yes** Button.
   The data specified in step 2 will be cleared.

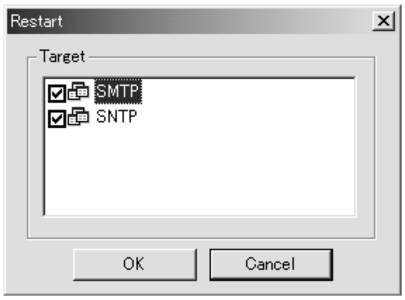
7-17 Restarting Services

Use the following procedures to restart the SMTP and SNTP services.

Note This function can be used to make changes to the email settings (SMTP) and time settings (SNTP) in the Ethernet Setting valid, eliminating the need to restart the CPU Unit.
Refer to the *NE1S Series Operation Manual* (Cat. No. Z901) for information on making the SMTP email settings and SNTP time settings.

1,2,3...

1. Select ***Controller - Restart Service***.
   The Restart Dialog Box will be displayed.



2. Select the service or services to be restarted and then click the **OK** Button.
   The specified service or services will be restarted.

CJ2 CPU Units do not support the SMTP service or as SNTP service. Therefore, service restart menu items cannot be used for CJ2 CPU Unit projects.

7-18 Displaying Errors and the Error Log

This section describes the procedures to display current errors and delete error log displays. Errors and the error log are displayed in the Error Log Window. Messages generated by the MSG instruction are also displayed.

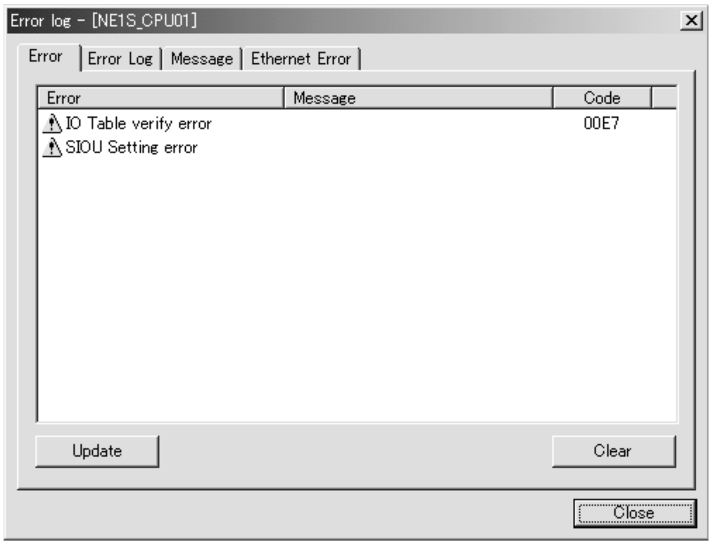
When an error that is stored in the error log occurs, an error message will flash in red in the status bar.

Perform all of the following operations online. Before starting, connect the NE Programmer online to the CPU Unit.

7-18-1 Displaying Current Errors

Use the following procedure to display current errors.

1,2,3... 1. Select ***Controller - Error Log***.
The Error Tab Page of the Error Log Window will be displayed as shown below.



Error	Message	Code
IO Table verify error		00E7
SIOU Setting error		

- Current errors will also be display on the Error Tab Page of the Error Log Window.
- Errors that occur after this window has been displayed will automatically be added to the display.
- The error level of each error (fatal or nonfatal) will also be indicated.

- The errors that may be displayed as follows:

Fatal Errors (in Order of Priority)	Nonfatal Errors (in Order of Priority)
Memory error	System error (FAL)
I/O bus error	Interrupt task error
Unit/Rack number duplication error	PLC Setup error
Too many I/O error	I/O verification error
I/O setting error	CPU Bus Unit error
Program error	Special I/O Unit error
Cycle time exceeded error	Battery error
System error (FALS)	CPU Bus Unit setting error
	Special I/O Unit setting error
	Communications interface error

2. If an error is displayed, first remove the cause of the error.
3. Click the **Clear** Button. All error displays will be cleared. (This is the same function as that executed for ***Controller - Clear Error***.

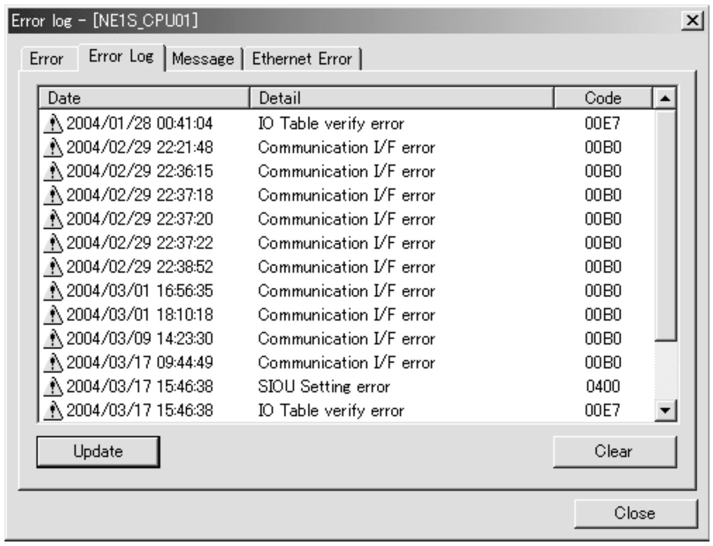
Errors, however, cannot be cleared when another Programming Devices has the access right or the CPU Unit is in RUN mode.

7-18-2 Displaying the Error log

Use the following procedure to display the error log.

Click the **Error Log** Tab in the Error Log Window.

An error log like the following one will be displayed.



Date	Detail	Code
2004/01/28 00:41:04	IO Table verify error	00E7
2004/02/29 22:21:48	Communication I/F error	00B0
2004/02/29 22:36:15	Communication I/F error	00B0
2004/02/29 22:37:18	Communication I/F error	00B0
2004/02/29 22:37:20	Communication I/F error	00B0
2004/02/29 22:37:22	Communication I/F error	00B0
2004/02/29 22:38:52	Communication I/F error	00B0
2004/03/01 16:56:35	Communication I/F error	00B0
2004/03/01 18:10:18	Communication I/F error	00B0
2004/03/09 14:23:30	Communication I/F error	00B0
2004/03/17 09:44:49	Communication I/F error	00B0
2004/03/17 15:46:38	SIOU Setting error	0400
2004/03/17 15:46:38	IO Table verify error	00E7

- Error errors displayed on the Error Log Tab Page are the same as those display on the Error Tab Page. Refer to the previous section for details.
- The error log contains up to 20 records. If more than 21 error occur, the oldest records will be deleted.
- The error log will be cleared if the **Clear** Button is pressed.

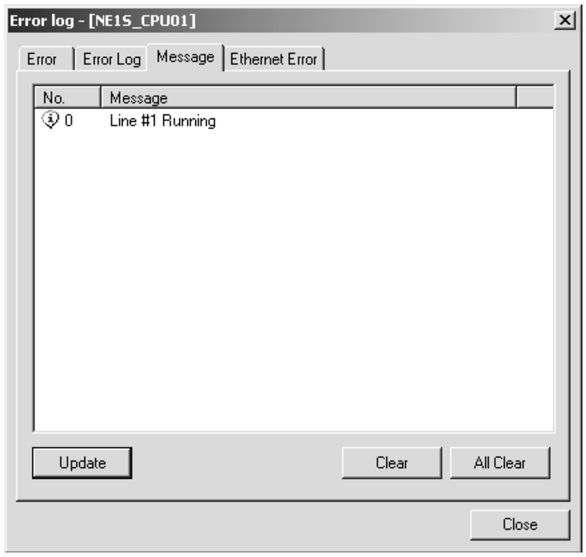
Note Current errors and the error log are not displayed in the Error Log Window when there is a CPU standby error (fatal). CPU standby status occurs when the POWER indicator on the Power Supply Unit is lit, but the RUN and ERR/ ALM indicators on the front of the CPU Unit are both not lit.

7-18-3 Displaying Messages

Use the following procedure to display messages generated by the MSG instruction.

Click the **Message** Tab in the Error Log Window.

A message list like the following one will be displayed.



No.	Message
0	Line #1 Running

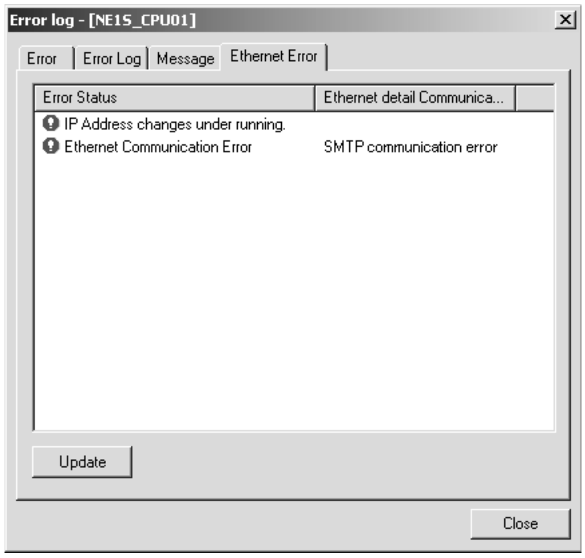
- The message numbers will be displayed along with the text of the message.
- Up to 8 messages will be displayed.
- The currently selected message will be cleared if the **Clear** Button is pressed.
- All messages will be cleared if the **All Clear** Button is pressed.
- Double-byte codes can be displayed.
- Messages generated by the FAL and FALS instructions will not be displayed.

7-18-4 Displaying Ethernet Errors

Use the following procedure to display Ethernet errors.

Click the **Ethernet Error Log** Tab in the Error Log Window.

Ethernet errors, like those shown below, will be displayed.



Error Status	Ethernet detail Communica...
IP Address changes under running.	
Ethernet Communication Error	SMTP communication error

Error Status

The error status is indicated as follows:

IP Address Settings Error
Unit IP Address Settings Error
FTP Server Error
SMTP Settings Error
SNTP Settings Error
Unit IP Address Settings Over
IP Routing Settings Error
IP Address Duplicate Error
IP Address Changes Under Running
Ethernet Communications Error

Ethernet Communications Errors

The following error status is displayed for the Ethernet communications error.

SMTP Communications Error
SNTP Communications Error
FINS/TCP Connection Error
FINS/TCP Send Error
FINS/TCP Receive Error
FINS/UCP Send Error
FINS/UCP Receive Error

7-19 Change Log

7-19-1 Overview (NE1S CPU Units Only)

The change log function cannot be used with CJ2 CPU Units.
The log data created when downloading, editing online, creating/clearing I/O tables, or clearing memory can be recorded in the CPU Unit.

Item	Details
Save timing	Downloading, executing online editing, creating/clearing I/O tables, clearing memory, and changing Timer/Counter SVs
Number of saved data records	100 records
Saved data	Time, operation (download, online edit, I/O table create/clear, memory clear), version, author, comments (120 bytes)
Location of saved data	Recorded data = Flash ROM in CPU Unit
Simple backup	Supports simple backup operations

- The settings can be made to send the data by email when the change log is recorded.
- Recorded change log entries can be deleted by the user.
- Recorded log data can also be overwritten during online editing.
- The change log can be enabled/disabled by the user (default: disabled).

7-19-2 Enabling/Disabling the Change Log

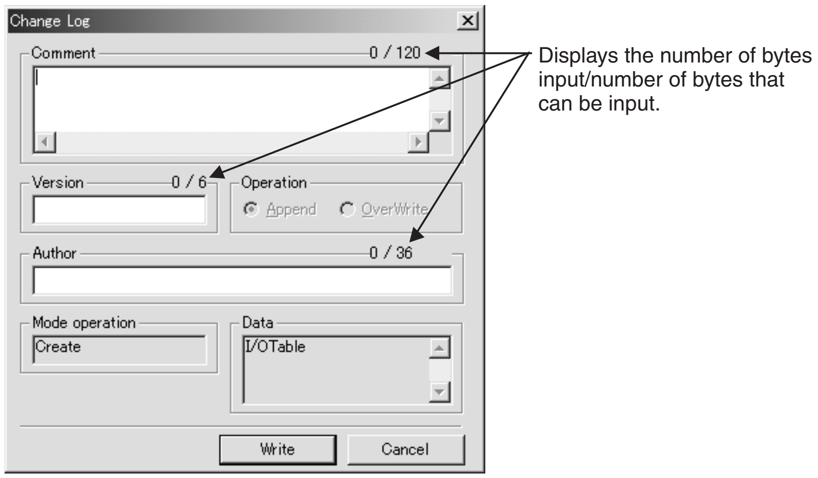
Set whether to enable or disable the change log. The default setting is disabled.

Select ***Controller - Change Log*** and then either ***Enable Mode*** or ***Disable Mode***. When *Enable Mode* is selected for change log, the Change Log Dialog Box will be displayed when downloading, editing online, creating/clearing I/O tables, or clearing memory.

7-19-3 Change Log Input Examples

Creating I/O Tables

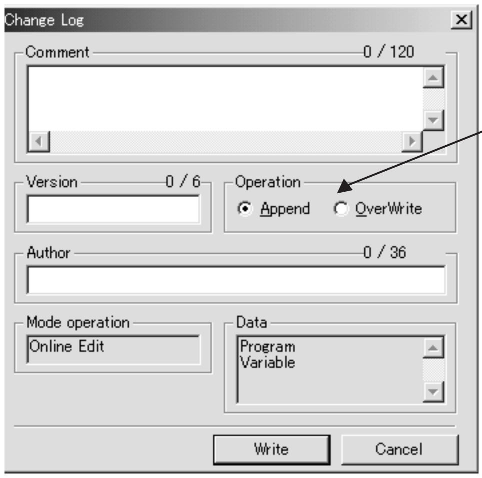
- When creating I/O tables, the following Change Log Dialog Box will be displayed.



- Input the comment and other required data, and then click the **Write** Button.

Editing Online

- When online editing is completed, the following Change Log Dialog Box will be displayed.



Operation: [✓ Append] OverWrite

Mode operation: Online Edit

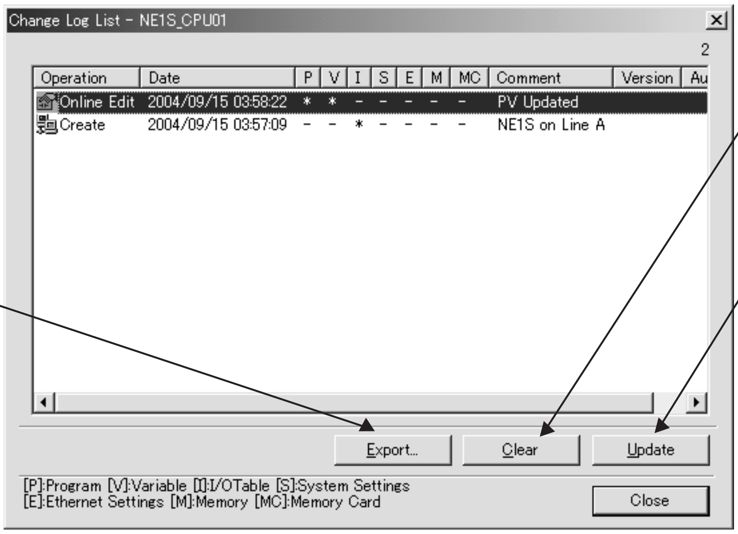
Data: Program, Variable

Select ***Append*** or ***OverWrite***. This operation is enabled during online editing only.

- Input the comment and other required data, select ***Append*** or ***OverWrite***, and then click the **Write** Button.
  To record several online editing operations as a single change log, select ***Overwrite***. To record each operation as a separate change log, select ***Append***.

7-19-4 Change Log Display

Select ***Controller - Change Log - Change Log List*** to display the Change Log List Window.



Operation	Date	P	V	I	S	E	M	MC	Comment	Version	Au
Online Edit	2004/09/15 03:58:22	*	*	-	-	-	-	-	PV Updated		
Create	2004/09/15 03:57:09	-	-	*	-	-	-	-	NE1S on Line A		

[P]:Program [V]:Variable [I]:I/OTable [S]:System Settings
[E]:Ethernet Settings [M]:Memory [MC]:Memory Card

Saves the change log as a CSV file.

Clears the change log.

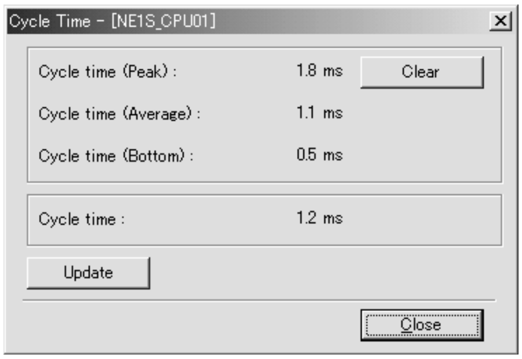
Refreshes the change log display.

7-20 Displaying the Cycle Time

The following procedure can be used to measure and display the cycle time of the program that is being executed. The CPU Unit must be in RUN or MONITOR mode to use this function.

Select ***Controller - Cycle Time***.

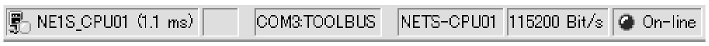
The following Cycle Time Dialog Box will be displayed.



```
Cycle Time - [NE1S_CPU01]
Cycle time (Peak) :        1.8 ms   [Clear]
Cycle time (Average) :     1.1 ms
Cycle time (Bottom) :      0.5 ms

Cycle time :               1.2 ms
[Update]
                                    [Close]
```

- The peak, average, bottom, and current cycle times will be cleared if the **Clear** Button is pressed.
- The peak, average, bottom, and current cycle times will be remeasured and displayed again if the **Update** Button is pressed.
- The Cycle Time Dialog Box will be closed if the **Close** Button is pressed.

Note Even if the above procedure is not performed, the average cycle time will be displayed in the status bar when the CPU Unit is in RUN or MONITOR mode.



```
NE1S_CPU01 (1.1 ms) | | COM3:TOOLBUS | NETS-CPU01 | 115200 Bit/s | On-line
```

7-21 Data Tracing

7-21-1 Data Tracing

Data tracing displays the operating status of the CPU Unit on the NE Programmer. Operating status includes the status of bits and the present values (PVs) of words.

When the specified trigger condition is met after starting data tracing, the CPU Unit will store the status of specified bits and the PVs of specified words in trace memory in the CPU Unit according to sampling conditions. The NE Programmer can read the data stored in the CPU Unit and display it on time charts.

Note The Data Trace Start Flag will turn ON when the data trace operation is executed.

- Any of three trigger conditions can be selected.

  Bit OFF to ON transition, bit ON to OFF transition, or specified word contents
- Either of two sampling conditions can be selected.

  Each cycle or when the TRSM instruction is executed.
- Data tracing will stop when the trace memory become full and then the trace data will be transferred to the NE Programmer and displayed.

- Data tracing is performed by the CPU Unit itself, which enables high-speed sampling in comparison to time chart monitoring.

■ **Number of traceable bits and words:**

Bits: 31, Words: 6

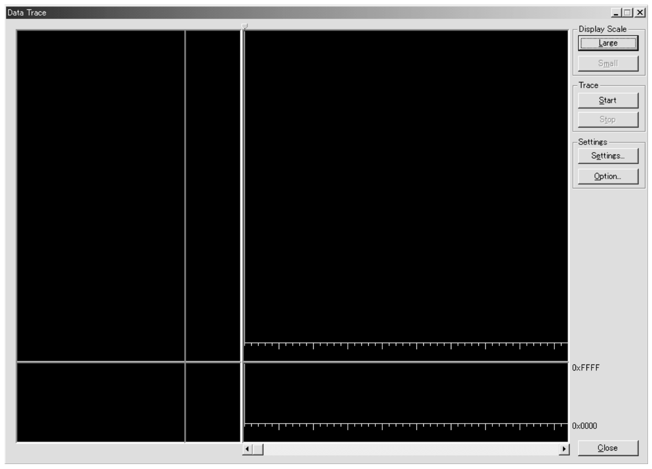
7-21-2 Opening and Closing the Data Trace Window

Monitoring data traces is performed in the Data Trace Window.

Opening

Select ***Controller - Data Trace***.

The following Data Trace Window will be displayed.



Closing

Click the **Close** Button in the Data Trace Window.

The Data Trace Window will be closed.

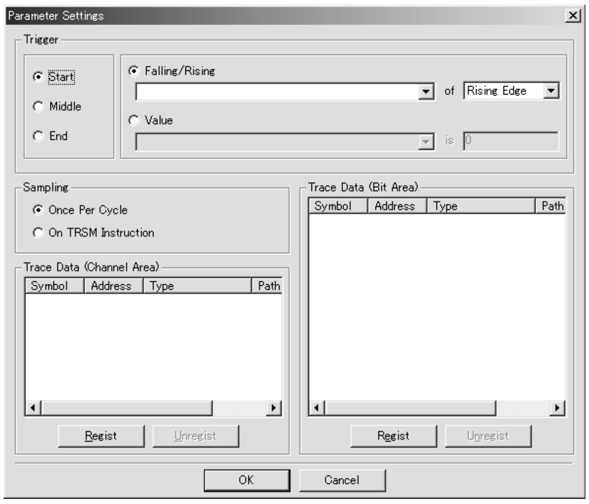
7-21-3 Setting Data Trace Parameters

This section described how to set the data trace parameters.

Displaying the Parameter Settings Dialog Box

1,2,3...

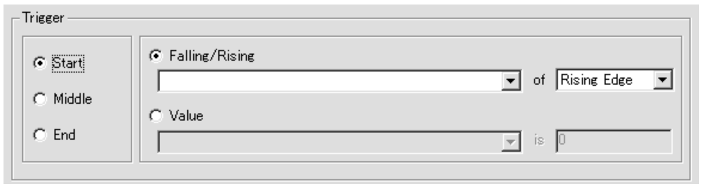
1. Click the **Settings** Button in the Data Trace Window.
   The following Parameter Settings Dialog Box will be displayed.



2. Make all of the required settings and then click the **OK** Button.
   The settings are described individually below.

Trigger

The *Trigger* fields are used to set the type of trigger and the condition for the bit or word trigger.



Select *Start*, *Middle*, or *End* as the type of trigger.

Start:	Used to display status after the trigger.
Middle:	Used to display status before and after the trigger.
End:	Used to display status before the trigger.

■ Using a Bit Trigger

Select ***Falling/Rising***, specify the variable or address, and select ***Falling*** or ***Rising***.

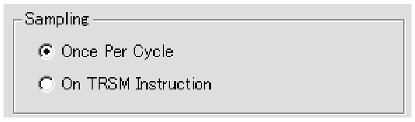
■ Using a Word Trigger

Select ***Value***, specify the variable or address, and input a hexadecimal value.

If *TIM/CNT* is input, the timer/counter Completion Flag will be used.

Sampling

Set the sampling condition in the *Sampling* Area.

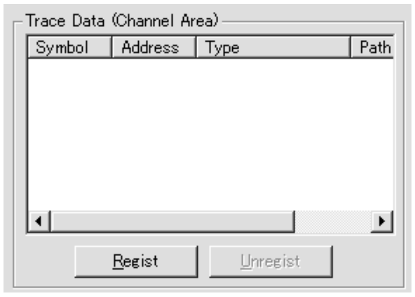


Select one of the two sampling methods.

Sampling method	Function	Contents of sampled data
Once Per Cycle	Sampling will be performed once per cycle.	I/O data after the END instruction is executed
On TRSM Instruction	Sampling will be performed when the TRSM instruction is executed.	I/O data when the END instruction is executed

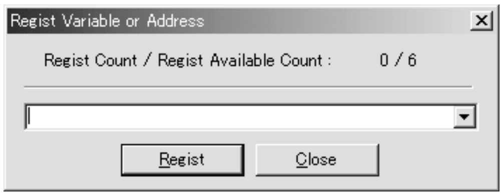
Trace Data (Channel Area)

Use the following procedure to specify the word variables or addresses to be sampled in the *Trace Data (Channel Area)*.



1,2,3...

1. Click the **Regist** Button.
   The following Regist Variables or Address Dialog Box will be displayed.



Regist Variable or Address

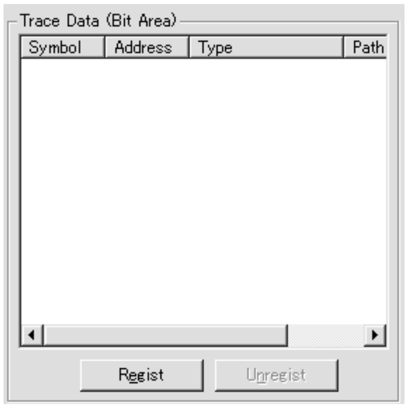
Regist Count / Regist Available Count : 0 / 6

2. Input the word variable or address to be sampled and then click the **Regist** Button.
3. Repeat step 2 to monitor more than one word. Up to 6 word variables and addresses can be registered.
4. Set all the required variables or addresses and then click the **Close** Button.

Note To delete a variable or address that has been set, select the variable or address and click the **Unregist** Button.

Trace Data (Bit Area)

Use the following procedure to specify the bit variables or addresses to be sampled in the *Trace Data (Bit Area).*

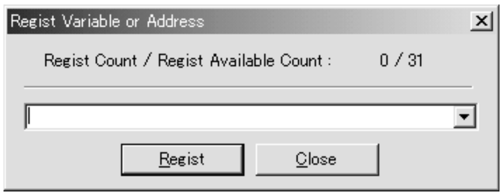


Trace Data (Bit Area)

Symbol	Address	Type	Path

1,2,3...

1. Click the **Regist** Button.
   The following Regist Variables or Address Dialog Box will be displayed.



Regist Variable or Address

Regist Count / Regist Available Count : 0 / 31

2. Input the bit variable or address to be sampled and then click the **Regist** Button.
3. Repeat step 2 to monitor more than one bit. Up to 31 bit variables and addresses can be registered.
4. Set all the required variables or addresses and then click the **Close** Button.

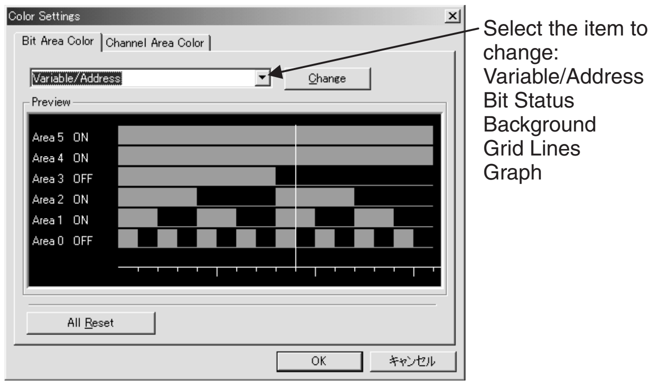
Note To delete a variable or address that has been set, select the variable or address and click the **Unregist** Button.

7-21-4 Setting Data Trace Display Colors

This section described how to set the colors displayed on the Data Trace Window.

Bit Area Colors

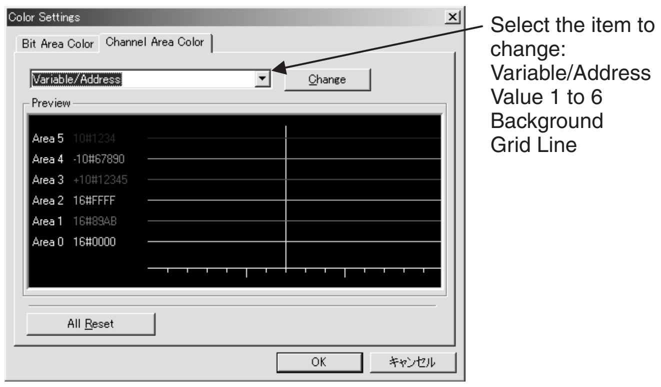
1,2,3...
1. Click the **Option** Button in the Data Trace Window.
   The following Bit Area Color Tab Page of the Color Settings Dialog Box will be displayed.



2. Select the item for which to change the color and then click the **Change** Button.
3. The standard Windows color setting dialog box will be displayed. Set the desired color.

Word Area Colors

1,2,3...
1. Click the **Channel Area Color** Tab in the Color Settings Dialog Box.



2. Select the item for which to change the color and then click the **Change** Button.
3. The standard Windows color setting dialog box will be displayed. Set the desired color.

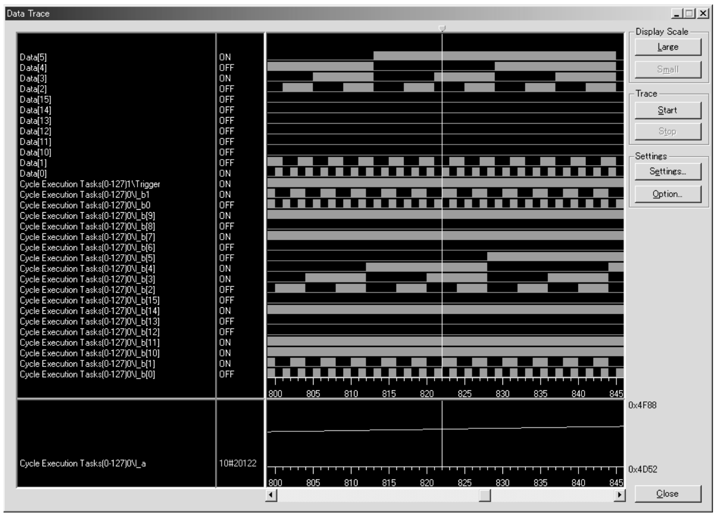
7-21-5 Executing the Data Trace Monitor Function

Use the following procedure to execute the data trace monitor function.

Be sure to set the data trace parameters before executing this procedure.

1,2,3...
1. Click the **Start** Button in the Data Trace Window.
   - The data trace will be started.

- Sampling will be started when the trigger condition is met, and traced data will be stored in the trace memory in the CPU Unit.
- When the trace memory becomes full, sampling will stop automatically. Traced data will be read from the CPU Unit and the trace data will be displayed in the Data Trace Window as shown below.



Data[5]	ON
Data[4]	OFF
Data[3]	ON
Data[2]	OFF
Data[15]	OFF
Data[14]	OFF
Data[13]	OFF
Data[12]	OFF
Data[11]	OFF
Data[10]	OFF
Data[1]	OFF
Data[0]	ON
Cycle Execution Tasks(0-127)1\Trigger	ON
Cycle Execution Tasks(0-127)0\I_b1	ON
Cycle Execution Tasks(0-127)0\I_b0	OFF
Cycle Execution Tasks(0-127)0\I_b[9]	ON
Cycle Execution Tasks(0-127)0\I_b[8]	OFF
Cycle Execution Tasks(0-127)0\I_b[7]	ON
Cycle Execution Tasks(0-127)0\I_b[6]	OFF
Cycle Execution Tasks(0-127)0\I_b[5]	OFF
Cycle Execution Tasks(0-127)0\I_b[4]	ON
Cycle Execution Tasks(0-127)0\I_b[3]	ON
Cycle Execution Tasks(0-127)0\I_b[2]	OFF
Cycle Execution Tasks(0-127)0\I_b[15]	OFF
Cycle Execution Tasks(0-127)0\I_b[14]	ON
Cycle Execution Tasks(0-127)0\I_b[13]	OFF
Cycle Execution Tasks(0-127)0\I_b[12]	OFF
Cycle Execution Tasks(0-127)0\I_b[11]	ON
Cycle Execution Tasks(0-127)0\I_b[10]	ON
Cycle Execution Tasks(0-127)0\I_b[1]	ON
Cycle Execution Tasks(0-127)0\I_b[0]	OFF
Cycle Execution Tasks(0-127)0\I_a	10#20122

- To stop the data trace before trace memory becomes full, click the **Stop** Button. The Data Trace Window will be displayed when the Stop Button is clicked.

Changing the Display Scale

The trace data display scale can be enlarged or reduced by clicking the **Large** or **Small** Button while the data trace data is being displayed.

7-22 Variable Reference List

The Variable Reference List Window displays the Variable Usage Report and Cross Reference Report. This window can be accessed both offline and online.

Variable Usage Report

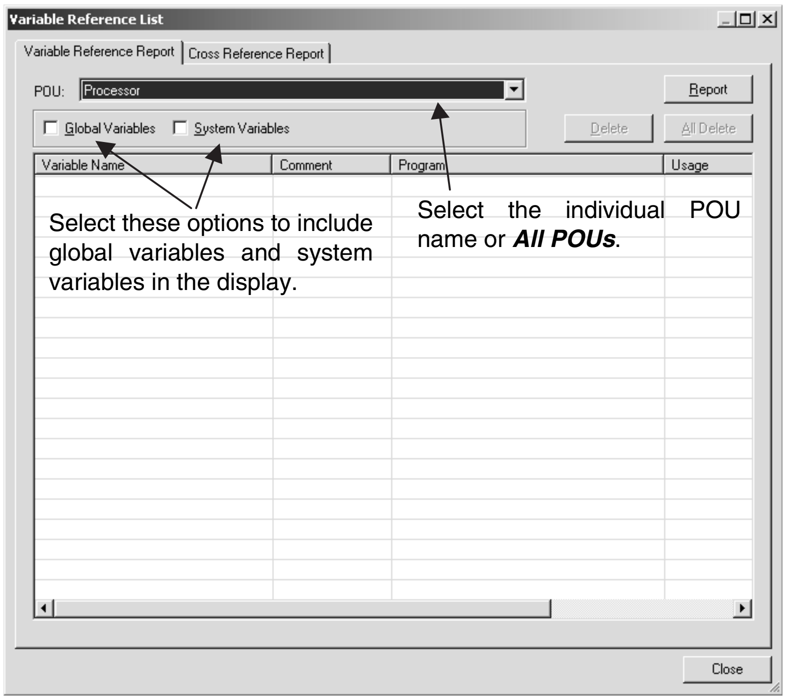
The variables registered in the project are listed in this report together with whether the registered variables are being used in the program. The variables not being used can be deleted.

Cross Reference Report

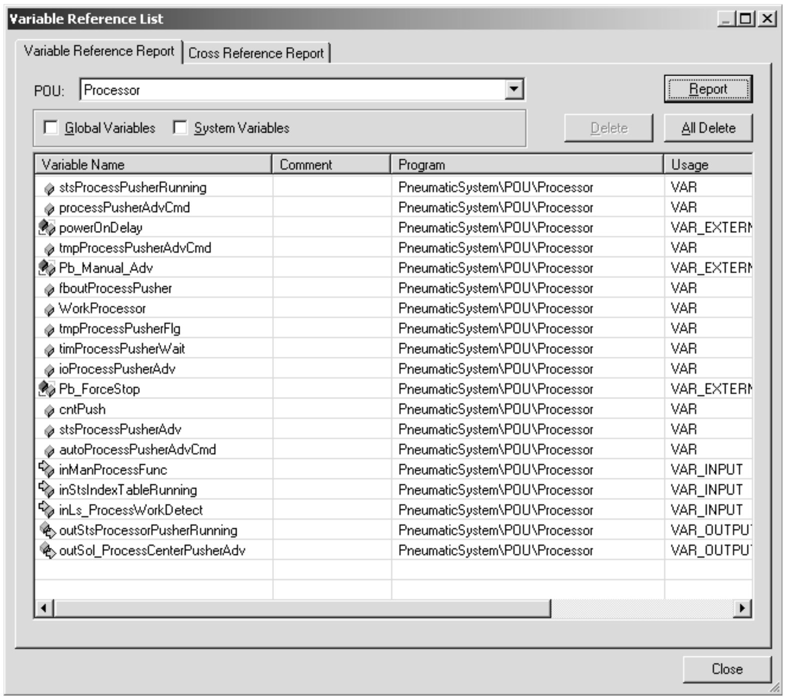
This report lists in which instructions, location, and program the variables are being used. The usage status of variables can be listed for individual POUs or for all POUs.

7-22-1 Variable Usage List

1,2,3... 1. Select ***Controller - Variable Reference***. The Variable Usage List Tab will be displayed in the Variable Reference List Window, as shown here.


Variable Reference List
Variable Reference Report
Cross Reference Report
POU:
Processor
Report
Global Variables
System Variables
Delete
All Delete
Variable Name
Comment
Program
Usage
Select these options to include global variables and system variables in the display.
Select the individual POU name or ***All POUs***.
Close

2. Select a POU name from the *POU* field, and then click the **Report** Button. The following Variable Usage Report will be displayed.



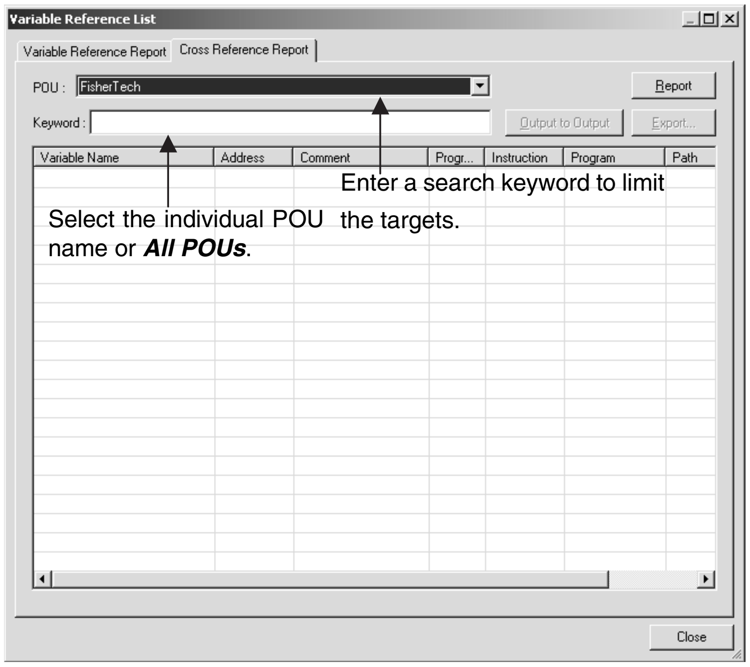
Variable Name	Comment	Program	Usage
stsProcessPusherRunning		PneumaticSystem\POU\Processor	VAR
processPusherAdvCmd		PneumaticSystem\POU\Processor	VAR
powerOnDelay		PneumaticSystem\POU\Processor	VAR_EXTER
tmpProcessPusherAdvCmd		PneumaticSystem\POU\Processor	VAR
Pb_Manual_Adv		PneumaticSystem\POU\Processor	VAR_EXTER
fboutProcessPusher		PneumaticSystem\POU\Processor	VAR
WorkProcessor		PneumaticSystem\POU\Processor	VAR
tmpProcessPusherFlg		PneumaticSystem\POU\Processor	VAR
timProcessPusherWait		PneumaticSystem\POU\Processor	VAR
ioProcessPusherAdv		PneumaticSystem\POU\Processor	VAR
Pb_ForceStop		PneumaticSystem\POU\Processor	VAR_EXTER
cntPush		PneumaticSystem\POU\Processor	VAR
stsProcessPusherAdv		PneumaticSystem\POU\Processor	VAR
autoProcessPusherAdvCmd		PneumaticSystem\POU\Processor	VAR
inManProcessFunc		PneumaticSystem\POU\Processor	VAR_INPUT
inStsIndexTableRunning		PneumaticSystem\POU\Processor	VAR_INPUT
inLs_ProcessWorkDetect		PneumaticSystem\POU\Processor	VAR_INPUT
outStsProcessorPusherRunning		PneumaticSystem\POU\Processor	VAR_OUTPU
outSol_ProcessCenterPusherAdv		PneumaticSystem\POU\Processor	VAR_OUTPU

• Click the **All Delete** Button to delete all unused variables.

• To delete unused variables individually, select the variable to be deleted (multiple variables can also be selected), and click the **Delete** Button.

7-22-2 Cross Reference Report

1,2,3... 1. Click the Cross Reference Report Tab in the Variable Reference List Window to display the following window.



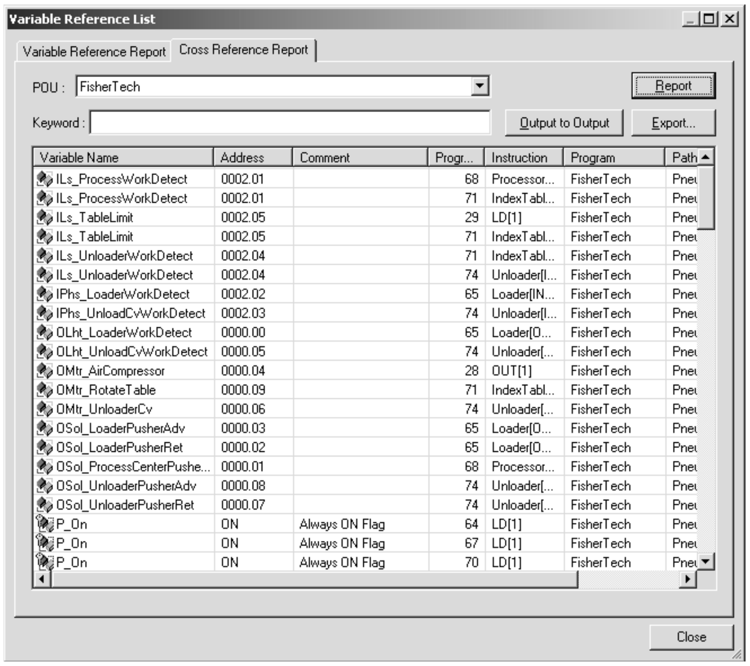
POU : FisherTech

Keyword :

Select the individual POU name or ***All POUs***.

Enter a search keyword to limit the targets.

2. Select a POU name from the *POU* field, and then click the **Report** Button. The following Cross Reference Report will be displayed, and the same information will be displayed as text in the Find Tab Page of the Output Window.



POU : FisherTech

Keyword :

Variable Name	Address	Comment	Progr...	Instruction	Program	Path
ILs_ProcessWorkDetect	0002.01		68	Processor...	FisherTech	Pneu
ILs_ProcessWorkDetect	0002.01		71	IndexTabl...	FisherTech	Pneu
ILs_TableLimit	0002.05		29	LD[1]	FisherTech	Pneu
ILs_TableLimit	0002.05		71	IndexTabl...	FisherTech	Pneu
ILs_UnloaderWorkDetect	0002.04		71	IndexTabl...	FisherTech	Pneu
ILs_UnloaderWorkDetect	0002.04		74	Unloader[I...	FisherTech	Pneu
IPhs_LoaderWorkDetect	0002.02		65	Loader[IN...	FisherTech	Pneu
IPhs_UnloadCvWorkDetect	0002.03		74	Unloader[I...	FisherTech	Pneu
OLht_LoaderWorkDetect	0000.00		65	Loader[O...	FisherTech	Pneu
OLht_UnloadCvWorkDetect	0000.05		74	Unloader[...	FisherTech	Pneu
OMtr_AirCompressor	0000.04		28	OUT[1]	FisherTech	Pneu
OMtr_RotateTable	0000.09		71	IndexTabl...	FisherTech	Pneu
OMtr_UnloaderCv	0000.06		74	Unloader[...	FisherTech	Pneu
OSol_LoaderPusherAdv	0000.03		65	Loader[O...	FisherTech	Pneu
OSol_LoaderPusherRet	0000.02		65	Loader[O...	FisherTech	Pneu
OSol_ProcessCenterPushe...	0000.01		68	Processor...	FisherTech	Pneu
OSol_UnloaderPusherAdv	0000.08		74	Unloader[...	FisherTech	Pneu
OSol_UnloaderPusherRet	0000.07		74	Unloader[...	FisherTech	Pneu
P_On	ON	Always ON Flag	64	LD[1]	FisherTech	Pneu
P_On	ON	Always ON Flag	67	LD[1]	FisherTech	Pneu
P_On	ON	Always ON Flag	70	LD[1]	FisherTech	Pneu

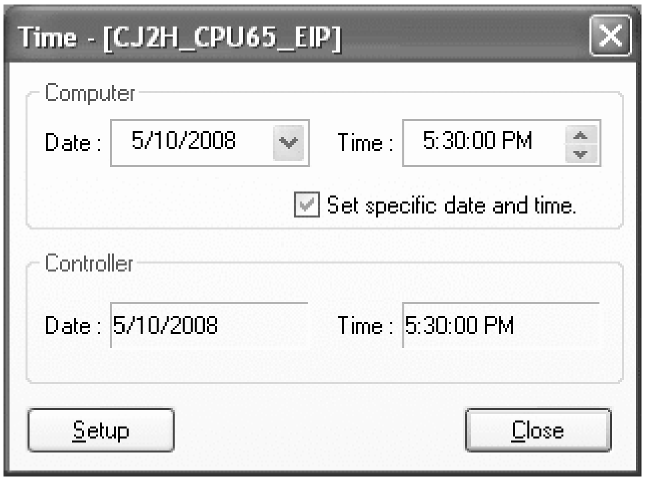
- Click the **Export** Button to save the contents of the report as a CSV file.

Note After closing the Variable Reference List Dialog Box, you can jump to the instruction by double-clicking the text output to the Output Window.

7-23 Setting the CPU Unit Clock

Use the following procedure to synchronize the CPU Unit's clock with the personal computer's clock.

1,2,3... 1. Select ***Controller - Set Clock***. The following Time Dialog Box will be displayed.



Date : 5/10/2008 Time : 5:30:00 PM

☑ Set specific date and time.

Date : 5/10/2008 Time : 5:30:00 PM

2. Click the **Setup** Button. The CPU Unit's clock will be synchronized with the personal computer's clock.

   The CPU Unit's clock can also be set by setting the *Date* and *Time* fields in this dialog box, and then clicking the Setup Button.

7-24 Forcibly Releasing the Access Right

The access right to the CJ2 CPU Unit can be forcibly released.

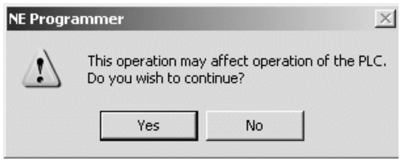
- Perform troubleshooting if the access right is not released.
- This function is not required for NE1S-series CPU Units and the menu commands will be disabled.

The access right is used to perform exclusive access control to a CPU Unit from Support Software on multiple computers. CPU Unit access rights are obtained while communications are being processed for a series of operations, such as downloading and online editing, from the Support Software. It is impossible to perform downloading or online editing from Support Software on another computer that does not have the access right.

The access rights may be obtained if an error occurs, such as when the communications cable is disconnected, during downloading, or during online editing. If this occurs, forcibly release the access right.

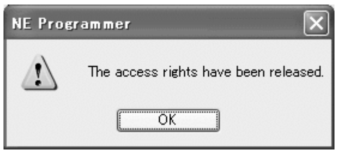
Use the following procedure to forcibly release the access right.

1,2,3... 1. Select ***Controller - Release Access Rights***. The following dialog box will be displayed for confirmation.



This operation may affect operation of the PLC.
Do you wish to continue?

2. Click the **Yes** Button.



The access rights have been released.

Appendix A
Variable Applications Guidelines

This section provides guidelines for using function blocks with the NE Programmer.

Using Variable Data Types

Integer Data Types (1, 2, or 4-word Data)

Use the following data types when handling single numbers in 1, 2, or 4-word units.

- INT and UINT
- DINT and DINT

Note Use signed integers if the numbers being used will fit in the range.

Word Data Types (1, 2, or 4-word Data)

Use the following data types when handling groups of data (non-numeric data) in 1, 2, or 4-word units.

- WORD
- DWORD

Array Settings

Array Variables
Use for First or End Addresses of Word Ranges

When specifying an instruction operand that is the first address or end address of a range of words (see note), the required values cannot be passed to variables through input parameters or output parameters.

Note Refer to the *SYSMAC NE1S Series Operation Manual* (Cat. No. Z901) and *Appendix D Instruction Support and Operand/Variable Restrictions* to determine which instruction operands must have array variables because they specify the first/end address of a range of words.

In this case, prepare an array variable with the required number of array elements, set the data in each array element in the function block, and specify the beginning (or end) array variable in the operand. Using an array variable allows you to specify the first address or end address of a range of words.

Handling a Single String of Data in Multiple Words

In this example, an array contains the directory and filename (operand S2) for an FREAD instruction.

- Variable
  Internal variable (VAR), data type = WORD, array setting with 10 elements, variable names = filename[0] to filename[9]
- Ladder Programming

```
MOV 16#5C31 file_name[0]  ⎫
MOV 16#3233 file_name[1]  ⎬←— Set data in each array element.
MOV 16#0000 file_name[2]) ⎭
FREAD (omitted) (omitted) file_name[0] (omitted) ←— Specify the first element
                                                    of the array in the instruction
                                                    operand.
```

Handling Control Data in Multiple Words

In this example, an array contains the number of words and first source word (operand S1) for an FREAD instruction.

- Variable
  Internal variable (VAR), data type = DINT, array setting with 3 elements, variable names = read_num[0] to read_num[9]
- Ladder Programming

```
MOVL 10#100 read_num[0] (No._of_words)    }← Set data in each array element.
MOVL 10#0 read_num[1] (1st_source_word)   }

FREAD (omitted) (omitted) file_name[0] (omitted) ← Specify the first element of the array
                                                    in the instruction operand.
```

Handling a Block of Read Data in Multiple Words

The allowed amount of read data must be determined in advance and an array must be prepared that can handle the maximum amount of data. In this example, an array receives the FREAD instruction's read data (operand D).

- Variable
  Internal variable, data type = WORD, array setting with 100 elements, variable names = read_data[0] to read_data[99]
- Ladder Programming

```
FREAD (omitted) (omitted) (omitted) read_data[0]
```

Division Using Integer Array Variables (Ladder Programming Only)

A two element array can be used to store the result from a ladder program's SIGNED BINARY DIVIDE (/) instruction. The result from the instruction is D (quotient) and D+1 (remainder). This method can be used to obtain the remainder from a division operation in ladder programming.

Note When ST language is used, it isn't necessary to use an array to receive the result of a division operation. Also, the remainder can't be calculated directly in ST language. The remainder must be calculated as follows:
Remainder = Dividend – (Divisor × Quotient)

Specifying External Variables or Physical Addresses for Function Blocks

Specify either the external variable or the physical address to enable reading or writing of Auxiliary Area bits within the algorithm (i.e., within the execution cycle) in the function block. (Auxiliary Area bits can also be used to receive and pass I/O variables.)

Example: Using Communications Port Enabled Flags (A20200 to A20207) in Function Blocks

The function block input variables are executed by referencing the items copied to the instances when the function block is called. The Communications Port Enabled Flags turn ON/OFF asynchronously with the program execution. When these bits are referenced as input variables in a function block, the Communications Port Enabled Flags are copied to the Instance Area when the function block is called. After copying, even if the value is changed prior to referencing the Communications Port Enabled Flags, detection is not possible in the function block program. Therefore, either define the Communications Port Enabled Flags as local variables and reference them as external variables from the function block, or specify the physical addresses (e.g., A20200).

Appendix B

Structured Text Keywords

Operators

Operation	Symbol	Data types supported by operator	Example	Value (example)	Priority 1: Lowest 11: Highest
Parentheses and brackets	(*expression*), *array*[*index*]	---	(2 + 3) * (4 + 5)	45	1
Function evaluation	*identifier* (*operand_list*)	---	---	---	2
Exponential	**	Not supported	---	---	3
Complement	–	INT, DINT	–10		4
Boolean complement	NOT	BOOL, WORD, DWORD	NOT TRUE	FALSE	4
Multiplication	*	INT, DINT, UINT, UDINT	10 * 3	30	5
Division	/	INT, DINT, UINT, UDINT	6/2	3	5
Remainder calculation	MOD	INT, DINT, UINT, UDINT	17 MOD 10	7	5
Addition	+	INT, DINT, UINT, UDINT, STRING	2 + 3	5	6
Subtraction	–	INT, DINT, UINT, UDINT	4 – 2	2	6
Comparisons	<, >, <=, >=	BOOL, WORD, DWORD, INT, DINT, UINT, UDINT, STRING	4 > 12	FALSE	7
Equality	=	BOOL, WORD, DWORD, INT, DINT, UINT, UDINT, STRING	10#16 = 16#10	TRUE	8
Non-equality	<>	BOOL, WORD, DWORD, INT, DINT, UINT, UDINT	8 <> 16	TRUE	8
Boolean AND	&, AND	BOOL, WORD, DWORD	TRUE & FALSE	FALSE	9
Boolean exclusive OR	XOR	BOOL, WORD, DWORD	TRUE XOR FALSE	TRUE	10
Boolean OR	OR	BOOL, WORD, DWORD	TRUE OR FALSE	TRUE	11

Restrictions

- Ladder programming special instructions cannot be used.
- Writing "100" means "10#100."

Conditional Statements

Keyword	Example	Function
RETURN	RETURN;	Return. Leaves the called function block and returns to the calling POU. With the NE Programmer, the RETURN statement can be used in the function block only.

IF	IF a < b THEN c: = 1;
	ELSIF a = b THEN C: = 2;
	ELSE c: = 3;
	END_IF;
		Selection.
		Evaluates group expression when the condition (a<b) is true. When the condition is false, the expression is not evaluated and the group following ELSE is evaluated.

CASE	CASE f OF
	1: a: = 3;
	2..5: a: = 4;
	ELSE a: = 0;
	END_CASE;
		Selection.
		Evaluates one group according to the value in the expression following the keyword CASE. The variable or expression *f* must be an INT data type.

Keyword	Example	Function

FOR	FOR a: = 1 TO 10 BY 3 DO
	f[a]: = b;
	END_FOR;
		Repetition.
		Evaluates the expressions in the range of DO to END_FOR, starting with 1 for variable *a* and sequentially adding 3 each execution, repeating until *a* reaches 10. It starts with the value of variable *a*, and ends with the value following TO, increasing with the value of BY. All values must be ANY_INT types.
		Note If BY is omitted, the default is 1 and all data must be INT type.

WHILE	WHILE b > 1 DO b: = b/2 END_WHILE;	Repetition. Evaluates the expressions of one group, repeating until the condition (b>1) is false. The condition for this expression is evaluated at the beginning of the loop, and when it is not true, the loop is not evaluated.

REPEAT	REPEAT
	a: = a*b;
	UNTIL a < 10000
	END_REPEAT;
		Repetition.
		Evaluates the expressions of one group, repeating until the condition (a<10000) is true. The condition for this expression is evaluated at the end of the loop, i.e., even if it is not true, the loop is evaluated at least once.

EXIT	FOR a: = 1 TO 2 DO
	IF flag THEN EXIT;
	END_IF
	SUM: = SUM + a
	END_FOR
		End.
		This statement can be used to exit an evaluation for a repetition statement.

Functions

Type	Name	Description
Math	ADD	Adds
	MUL	Multiplies
	SUB	Subtracts
	MOD	Finds remainder
	DIV	Divides
	MOVE	Assigns
Bit manipulation	SHL	Shifts 1 bit left
	SHR	Shifts 1 bit right
	ROR	Rotates 1 bit right without carry bit
	ROL	Rotates 1 bit left without carry bit
Logic operation	AND	Logical AND
	OR	Logical OR
	XOR	Logical exclusive OR
	NOT	Logical NOT
Character string manipulation	LEFT	Gets character string from the left
	RIGHT	Gets character string from the right
	MID	Gets character string from any position
	DELETE	Deletes character string
	CONCAT	Concatenates character strings
	INSERT	Inserts character string
	REPLACE	Replaces character string
	LEN	Gets character string length
	FIND	Finds character string j
Selection	MAX	Gets maximum value
	MIN	Gets minimum value

Type	Name	Description
Data type conversion	INT_TO_UINT	Converts INT to UINT
	INT_TO_DINT	Converts INT to DINT
	INT_TO_UDINT	Converts INT to UDINT
	INT_TO_WORD	Converts INT to WORD
	UINT_TO_INT	Converts UINT to INT
	UINT_TO_DINT	Converts UINT to DINT
	UINT_TO_UDINT	Converts UINT to UDINT
	UINT_TO_WORD	Converts UINT to WORD
	DINT_TO_INT	Converts DINT to INT
	DINT_TO_UINT	Converts DINT to UINT
	DINT_TO_UDINT	Converts DINT to UDINT
	DINT_TO_DWORD	Converts DINT to DWORD
	UDINT_TO_INT	Converts UDINT to INT
	UDINT_TO_UINT	Converts UDINT to UINT
	UDINT_TO_DINT	Converts UDINT to DINT
	UDINT_TO_DWORD	Converts UDINT to DWORD
	WORD_TO_INT	Converts WORD to INT
	WORD_TO_UINT	Converts WORD to UINT
	WORD_TO_DWORD	Converts WORD to DWORD
	DWORD_TO_INT	Converts DWORD to INT
	DWORD_TO_UINT	Converts DWORD to UINT
	DWORD_TO_WORD	Converts DWORD to WORD

ST Language Reserved Words

The following ST language reserved words cannot be used as identifiers.

N	END_RESOURCE	DATE	F_TRIG	~
R	RETAIN	TIME_OF_DAY	CTU	*
L	RETURN	TIME	CTD	/
D	STEP	AND	CTUD	MOD
P	END_STEP	OR	TP	+
SD	STRUCT	NOT	TON	-
DS	END_STRUCT	SHL	TOF	<
SL	TASK	SHR	RTC	>
ACTION	TRANSITION	ROR	LD	<=
END_ACTION	FROM	ROL	LDN	>=
ARRAY	TO	SUB	ST	=
AT	END_TRANSITION	MUL	STN	<>
CASE	TRUE	MOD	S	&
OF	TYPE	EXPT	R	VAR_SYSTEM
ELSE	END_TYPE	ABS	ANDN	CHANNEL
END_CASE	VAR	SQRT	AND(	FI
CONFIGURATION	END_VAR	LN	ANDN(	
END_CONFIGURATION	VAR_INPUT	LOG	ORN	
CONSTANT	VAR_IN_OUT	EXP	OR(	
EN	VAR_OUTPUT	SIN	ORN(	
ENO	VAR_EXTERNAL	COS	XOR	
EXIT	VAR_ACCESS	TAN	XORN	
FALSE	VAR_GLOBAL	ASIN	XOR(	

F_EDGE	VAR_TEMP	ACOS	XORN(	
FOR	WHILE	ATAN	ADD	
BY	END_WHILE	USINT_TO_DINT	ADD(	
DO	WITH	BOOL_TO_BYTE	SUB(	
END_FOR	ANY	SEL	MUL(	
FUNCTION	ANY_NUM	MIN	DIV	
END_FUNCTION	ANY_REAL	MAX	DIV(	
FUNCTION_BLOCK	LREAL	LIMIT	GT	
END_FUNCTION_BLOCK	REAL	MUX	GT(	
IF	ANY_INT	T	GE(	
THEN	SINT	GE	EQ(	
ELSEIF	INT	EQ	NE(	
ELSE	DINT	LT	LE	
END_IF	LINT	NE	LE(	
INITIAL_STEP	USINT	LEN	JMP	
END_STEP	UINT	LEFT	JMPNC	
PROGRAM	ULINT	RIGHT	JMPC	
WITH	UDINT	MID	CAL	
END_PROGRAM	ANY_BIT	CONCAT	CALNC	
R_EDGE	BOOL	INSERT	CALC	
READ_ONLY	BYTE	DELETE	RET	
READ_WRITE	WORD	REPLACE	RETNC	
REPEAT	DWORD	FIND	RETC	
UNTIL	LWORD	SR	(	
END_REPEAT	STRING	RS	)	
RESOURCE	ANY_DATE	SEMA	Function	
ON	DATE_AND_TIME	R_TRIG	**	

Appendix C
External Variables

Classification	Name	External variable in NE Programmer	Data type	Address
Conditions Flags	Greater Than or Equals (GE) Flag	P_GE	BOOL	CF00
	Not Equals (NE) Flag	P_NE	BOOL	CF001
	Less Than or Equals (LE) Flag	P_LE	BOOL	CF002
	Instruction Execution Error (ER) Flag	P_ER	BOOL	CF003
	Carry (CY) Flag	P_CY	BOOL	CF004
	Greater Than (GT) Flag	P_GT	BOOL	CF005
	Equals (EQ) Flag	P_EQ	BOOL	CF006
	Less Than (LT) Flag	P_LT	BOOL	CF007
	Negative (N) Flag	P_N	BOOL	CF008
	Overflow (OF) Flag	P_OF	BOOL	CF009
	Underflow (UF) Flag	P_UF	BOOL	CF010
	Access Error Flag	P_AER	BOOL	CF011
	Always OFF Flag	P_Off	BOOL	CF114
	Always ON Flag	P_On	BOOL	CF113
Clock Pulses	0.02 second clock pulse bit	P_0_02s	BOOL	CF103
	0.1 second clock pulse bit	P_0_1s	BOOL	CF100
	0.2 second clock pulse bit	P_0_2s	BOOL	CF101
	1 minute clock pulse bit	P_1mim	BOOL	CF104
	1.0 second clock pulse bit	P_1s	BOOL	CF102
Auxiliary Area Flags/ Bits	First Cycle Flag	P_First_Cycle	BOOL	A200.11
	First Task Execution Flag	P_First_Cycle_Task	BOOL	A200.15
	Maximum Cycle Time	P_Max_Cycle_Time	DWORD	A262
	Present Scan Time	P_Cycle_Time_Value	DWORD	A264
	Cycle Time Error Flag	P_Cycle_Time_Error	BOOL	A401.08
	Low Battery Flag	P_Low_Battery	BOOL	A402.04
	Output OFF Bit	P_Output_Off_Bit	BOOL	A500.15

Appendix D
CIP Message Communications

This document describes CIP message transmission for NE1S Series by using CSND instructions.

CIP Object on page 229 through *Example of Use for NE1S Series* on page 238 describe the basic information required for CIP to use CSND instructions to help you understand CSND instruction specifications deeply.

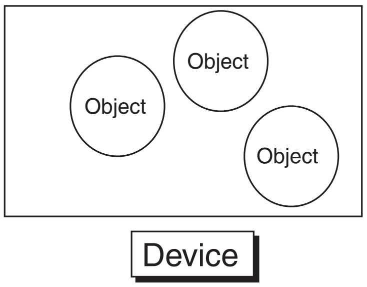
Refer to *Data Access for NE1S Series* on page 244 when the using CSND instructions.

Refer to *Data Type* on page 264 and *Response Code* on page 268 as required for a lists of data types and error codes supported by NE1S Series.

CIP Object

Object Model

Each device is modeled as a group of "Objects" in the conception of CIP. Object represents something that a particular element of a device is abstracted.



You should access to each Object when accessing from the outside.

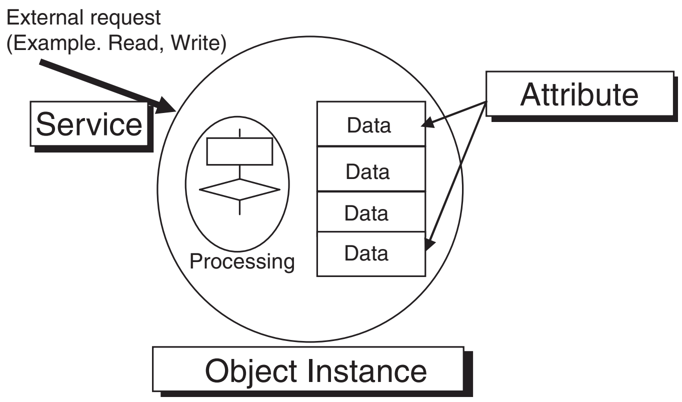
Object represents the processing and the data resulted from abstraction of a function in the device.

A request from the outside of Object, such as Read Data, is called "Service."

Data belonging to Object is called "Attribute."

The entity of Object is called "Instance" or "Object Instance."

When Object is generalized, it is called "Class." For example, "Japan" is one of Instances (Object Instances) of Class "Nation."



Reference Information

In CIP Common Specifications, “Object,” “Class,” “Instance,” “Attribute” and “Service” are explained as follows:

(Extracts from CIP Common Specifications)

Object	An abstract representation of a particular component within a product.
Class	A set of objects that all represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values.
Instance	A specific and real (physical) occurrence of an object. For example: NewZealand is an instance of the object class Country. The terms Object, Instance, and Object Instance all refer to a specific Instance.
Attribute	A description of an externally visible characteristic or feature of an object. Typically, attributes provide status information or govern the operation of an Object. For example: the ASCII name of an object; and the repetition rate of a cyclic object.
Service	A function supported by an object and/or object class. CIP defines a set of common services and provides for the definition of Object Class and/or Vendor Specific services.

Designation of Object Address

This is the concept to access to Object or Attribute.

Each Object Class has “Class ID”.

There are two types of “Class ID”; one is standardized by ODVA and the other is decided independently by each device vendor.

Each Object Instance also has ID. This is called “Instance ID.” Different Instance ID is assigned to each Object. As for Object Class standardized by ODVA, Instance ID is given to it according to the ODVA method. On the other hand, vendor's own Instance ID is decided independently by the vendor.

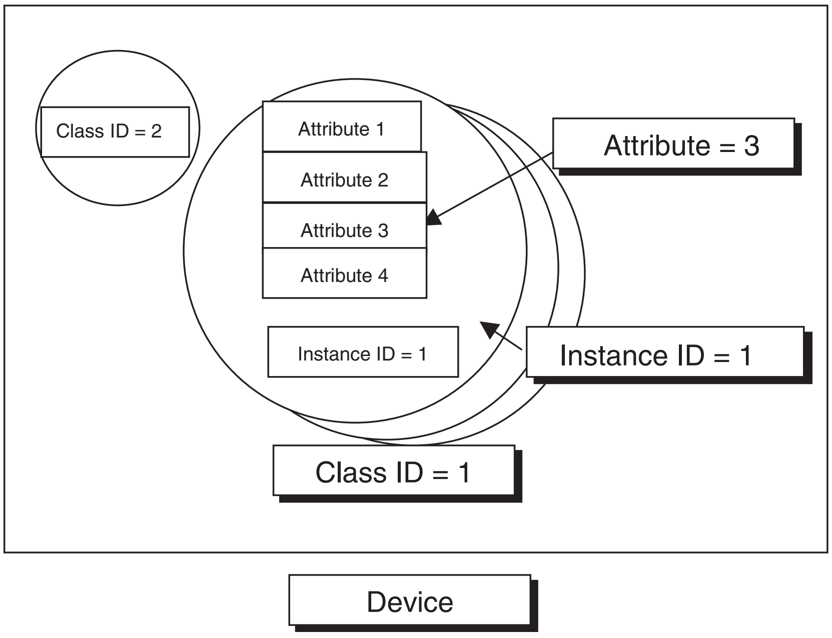
Each Attribute also has “Attribute ID.”

Each Object is accessed to by using “Class ID,” “Instance ID,” and “Attribute ID.”

In the device, you can designate Object by specifying these three IDs.

When requesting “Service,” you should specify “Class ID,” “Instance ID,” and “Attribute ID.” (Instance ID and Attribute ID may not be required, depending on the Service.)

These three IDs are called “IOI (Internal Object Identifier)” because they identify the location of Object in the device.



Link Path

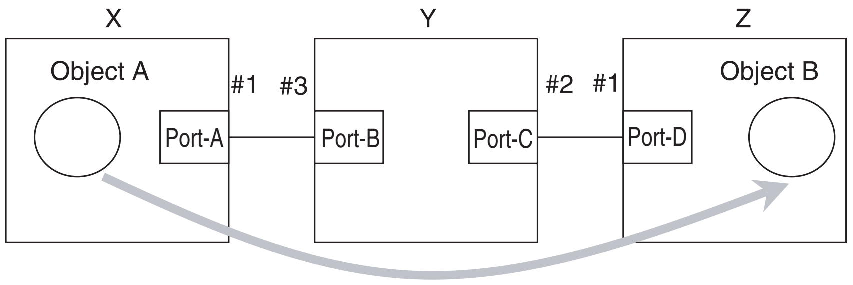
Link Path

For CIP, different from the internet protocol, the relay route from the transmission node to the reception node is all described in the transmission frame.

The described route is called "Link Path." Link Path is described as "EPATH type."

The conception of Link Path is as follows:

First of all, designate a network port of a transmission channel with the destination network, and designate node address on that network, which is called Link Address. For the relay channel, similarly, designate a network port with the destination network and node address on that network. Then, repeat the same procedure to the final destination.



When sending data from X to Z.

Link Path = Port A: #3, Port C: #1

Send data from the network port of X (Port-A) to #3 on that circuit, and the data reaches Y. Then, send it from the network port of Y (Port-C) to #1 on that circuit. Through this procedure, the destination node Z can be designated.

Description by EPATH Type

For CIP, EPATH type is employed for describing Link Path and IOI.

This is the method of dividing Link Path or IOI into segments and assigning a value to each of them.

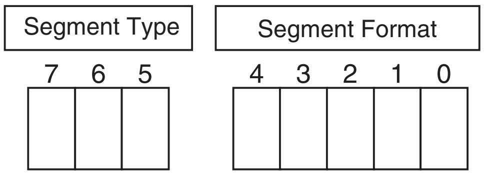
Therefore, Link Path description indicates the final destination by joining data called segments.

The segment includes the segment type information and the segment data.

Segment 1	Segment 2	Segment 3	Segment 4	· · · ·

Details of Segment Type

The interpretation method of a segment is included in the first 1 byte, which consists of tow parts; 3 bits of "Segment Type" and 5 bits of "Segment Format."



According to CIP Specifications, the Segment Type specifications are decided as follows:

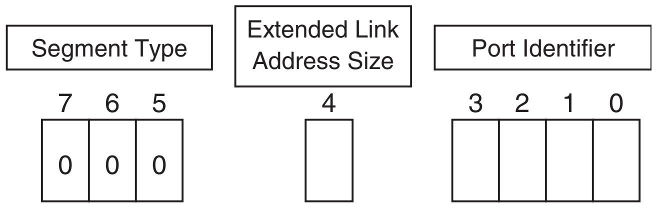
Segment Type			Description
7	**6**	**5**	
0	0	0	Port Segment
0	0	1	Logical Segment
0	1	0	Network Segment
0	1	1	Symbolic Segment
1	0	0	Data Segment
1	0	1	Data Type
1	1	0	Data Type
1	1	1	Reserved

The specifications of Segment Format are different for each Segment Type.

The following sections describe Port Segment, Logical Segment, and Data Segment which are to be required for using CSND instructions.

Port Segment

Port Segment is employed for describing the above-mentioned path.

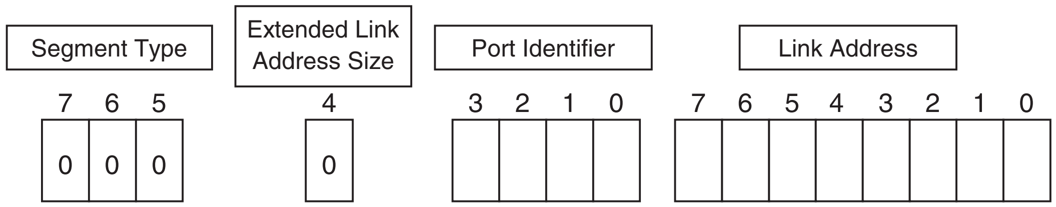


Set ID of that port in Port Identifier.

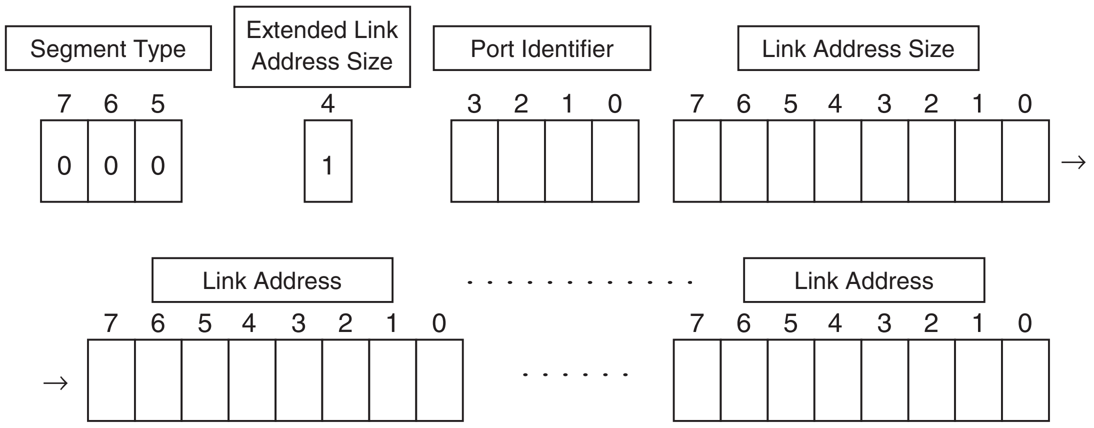
Port Identifier is 4-bit, so that it can take a value of 0 to 15. “0” is reserved and not available. “1” is to indicate the backplane port. “15” has a special meaning, indicating that the size of Port Identifier is larger than 1 byte. In this case, Port Identifier is followed by 2-byte Port Identifier. This case is not explained here because, for EN1S Series, Port Identifier will not exceed 1 byte.

Set “1” in Extended Link Address Size when Link Address of that port is larger than 1 byte.

Shown below is the description method of Port Segment when “0” is set in Extended Link Address Size.



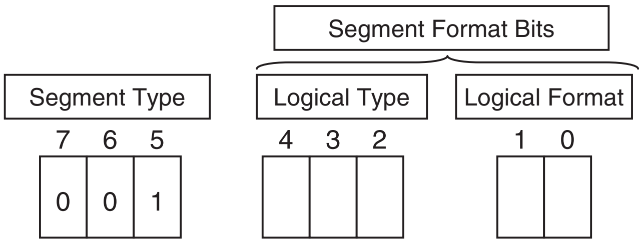
Designate the size of Link Address when “1” is set in Extended Link Address Size. Shown below is the description method of Port Segment.



Set even-number byte in Link Address without fail. If it is an odd number, surely change it to an even number by padding with “00.”

Logical Segment

Logical Segment is employed for describing IOI.



Logical Type			Description
4	**3**	**2**	
0	0	0	Class ID
0	0	1	Instance ID
0	1	0	Member ID
0	1	1	Connection Point
1	0	0	Attribute ID
1	0	1	Special (Do not use the logical addressing definition for the Logical Format.)
1	1	0	Service ID (Do not use the logical addressing definition for the Logical Format.)
1	1	1	Reserved

Logical Format		Description
1	**0**	
0	0	8-bit logical address
0	1	16-bit logical address
1	0	32-bit logical address
1	1	Reserved

The 32-bit logical address of Logical Format is reserved and not available.

The 8-bit and 16-bit logical addresses are available for Class ID and Instance ID which indicate IOI.

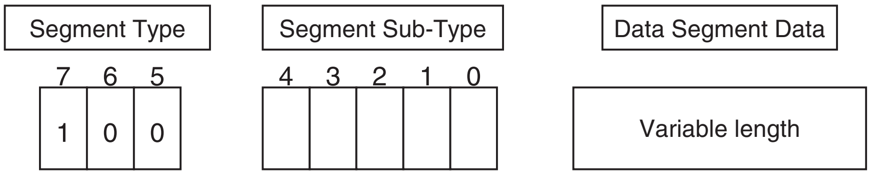
The 8-bit logical address is available for Attribute ID.

This is used for requesting Service to an optional Object of an optional device.

This can be also used for directly reading/writing IO memory of NE1S by designating the address.

Data Segment

Data Segment is employed for reading/writing variables of CPU Unit.

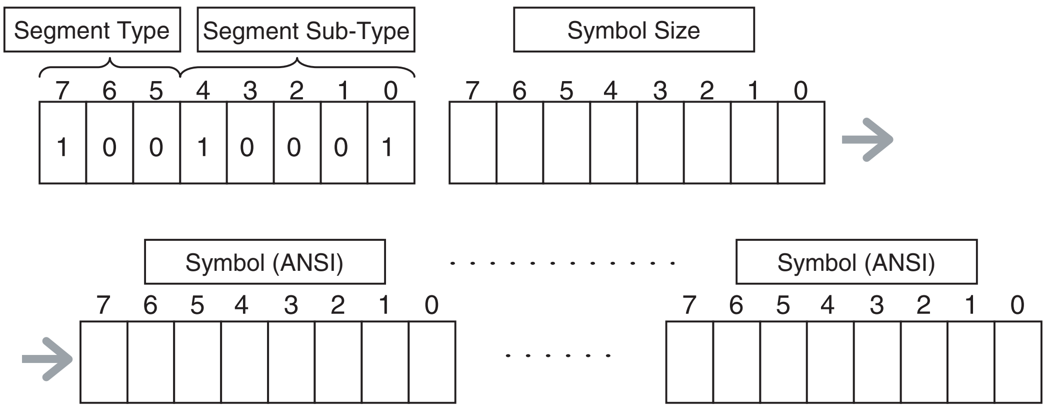


Segment Sub-Type					Description
4	**3**	**2**	**1**	**0**	
0	0	0	0	0	Simple Data Segment
1	0	0	0	1	ANSI Extended Symbol Segment
					All Segment Sub-Types are reserved except the above.

ANSI Extended Symbol Segment is mainly used for Data Segment.

Variable data is read/written by using the segment of this type.

ANSI Extended Symbol Segment



Variables and IOI

As mentioned above, Object Model is used in the concept of CIP. A device is recognized as a group of Objects. You have to identify the data you want to access to by using Class/Instance.

When accessing to a variable by using a variable name, the variable name must be converted to Class/ Instance. For NE1S Series, CPU Unit performs the conversion process, so that the user doesn't have to do it. All the user need to do is just to designate a variable in spite of Class/Instance.

Inputting a Network Path

Network Path

The network path indicates the path from the local CPU Unit to another PLC in the network. The NE1S-series PLCs can use the network path for routing.

When a Network Path is Required

A network path can be specified in the following case.

- SEND CIP COMMAND Instruction (CSND)
  When reading or writing the I/O memory of another node in the network with the SEND CIP COMMAND Instruction (CSND), the network path can be specified in the instruction's control data.

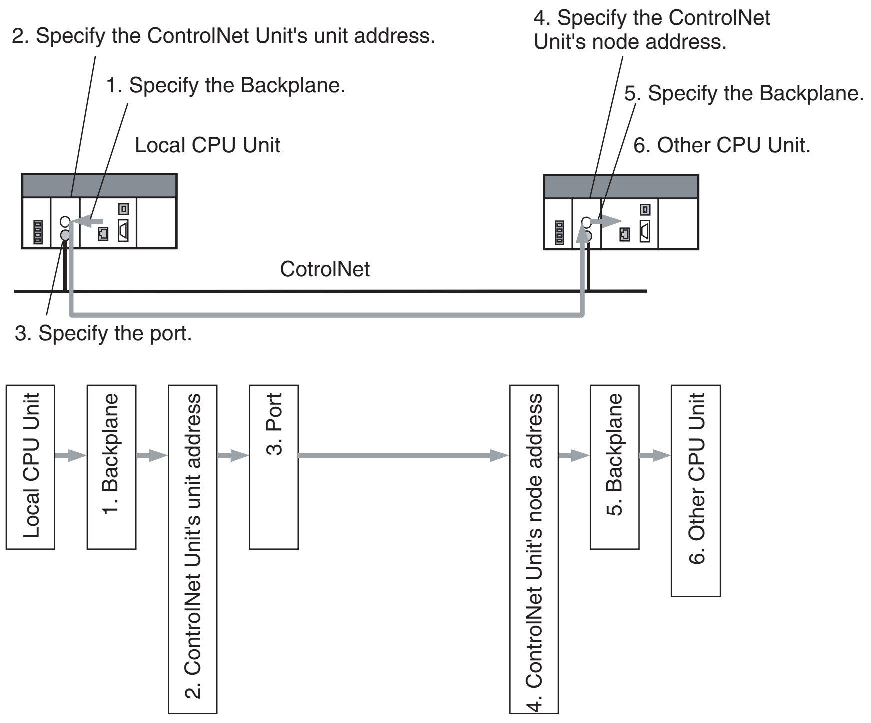
The I/O memory of another PLC in the network can be accessed through a variable by specifying the network path and variable name.

Specifying the Network Path

Specify the network as shown in the following table.

Target device	Specifying the network path for a SEND CIP COMMAND Instruction (CSND)
Backplane	01 hex
Specifying a Unit mounted to the local CPU Unit	Specify the unit address in hexadecimal. For a CPU Bus Unit, specify the unit number + 10 hex.
Specifying the network port	Ethernet port, e.g., 02 hex
Specifying a Unit in the network	Specify the network node address in hexadecimal.
Specifying the other PLC’s CPU Unit	00 hex (unit address)

Example:
Specifying the path from the local CPU Unit to a local variable in another CPU Unit in a ControlNet Network


2. Specify the ControlNet Unit's unit address.
1. Specify the Backplane.
Local CPU Unit
4. Specify the ControlNet Unit's node address.
5. Specify the Backplane.
6. Other CPU Unit.
CotrolNet
3. Specify the port.
Local CPU Unit
1. Backplane
2. ControlNet Unit's unit address
3. Port
4. ControlNet Unit's node address
5. Backplane
6. Other CPU Unit

Example		**Specifying the network path for a SEND CIP COMMAND Instruction (CSND)**
Specification method	1. Backplane	16#01
	2. ControlNet Unit's unit address (Example: Unit address = 3)	16#13
	3. Port (Example: Port = Port B)	16#02
	4. ControlNet Unit's node address (Example: Node address = 5)	16#5
	5. Backplane	16#01
	6. Other CPU Unit	16#00

Refer to *Data Access for NE1S Series* for information on the CSND instruction.

Example of Use for NE1S Series

Setup of Link Path

Port Number

Described below is the network port for designating Link Path.

For NE1S Series, the base unit (backplane) is also recognized as part of the network when designating Link Address.

CPU Unit

CPU Unit has two ports.

One is a backplane port and the other is Ethernet port.

The backplane port is CS1 bus (base unit). Communication from CPU Unit via a CPU Bus Unit surely goes through the backplane.

Port	Port Number
Backplane	1
Ethernet	2

ControlNet Unit

ControlNet Unit has two ports.

One is a backplane port and the other is ControlNet port.

The backplane port is CS1 bus (base unit). Communication via CPU Unit or other CPU Bus Unit surely goes through the backplane.

Port	Port Number
Backplane	1
ControlNet	2

DeviceNet Unit

DeviceNet Unit has two ports.

One is a backplane port and the other is DeviceNet port.

The backplane port is CS1 bus (base unit). Communication via CPU Unit or other CPU Bus Unit surely goes through the backplane.

Port	Port Number
Backplane	1
DeviceNet	2

Link Address

Link Address is a node address on the network for designating Link Path.

The method to set up Link Address is different for each network.

For NE1S Series, the base unit (backplane) is also recognized as part of the network when designating Link Address.

Backplane (Base Unit)

- **CPU Bus Unit**
  For NE1S Series, the base unit is recognized as a backplane port.
  CPU Bus Units, such as ControlNet Unit or DeviceNet Unit, are also recognized as nodes on the backplane port.
  Link Address of a CPU Bus Unit, on the backplane, is “Unit No. + 10 hex.” For example, when Unit No. is 0, Link Address is 10 hex. When Unit No. is F, Link Address is 1F hex.

- **CPU Unit**
  Link Address of CPU Unit, on the backplane, will be surely 00 hex.
  IP address is employed to describe Link Address of Ethernet port of CPU Unit, which is the same as that of Ethernet port mentioned below.

Network

- **Ethernet**
  IP address is employed to describe Link Address of Ethernet port.
  All of IP address must be described with ASCII code.
  For example, IP address of 192.168.200.200 will be [31] [39] [32] [2E] [31] [36] [38] [2E] [32] [30] [30] [2E] [32] [30] [30].
- **ControlNet**
  The node address of ControlNet is described as Link Address of ControlNet port.
  The node address of ControlNet is “1” to “99” (01 hex to 63 hex). For ControlNet, Link Address doesn't have to be described with ASCII code.
- **DeviceNet**
  The node address of DeviceNet is described as Link Address of DeviceNet port.
  The node address of DeviceNet is “0” to “63” (00 hex to 3F hex). For DeviceNet, Link Address doesn't have to be described with ASCII code.

Example of Path

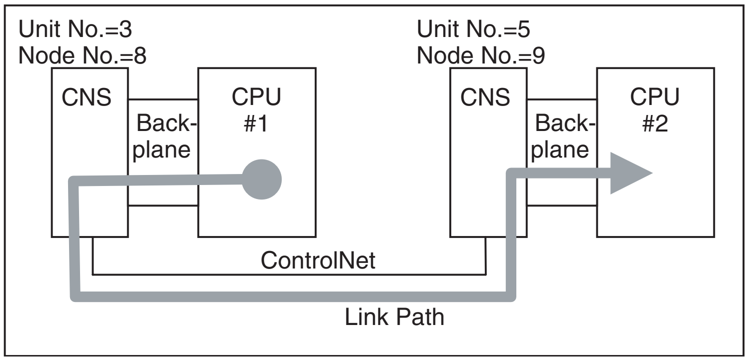
Example 1:

In case of accessing from CPU#1 to CPU#2 via ControlNet Unit

[Structure]

ControlNet Unit of CPU#1: Unit No. = 3, Node Address = 8

ControlNet Unit of CPU#2: Unit No. = 5, Node Address = 9



The route is as follows:

Backplane Port of CPU#1 → ControlNet Unit (Unit No. = 3) → ControlNet Port of ControlNet Unit → ControlNet Unit (Node Address =9) → Backplane Port of ControlNet Unit → CPU#2

The route above is explained below.

Port Segment is used for setting Link Path. Therefore, the top 3 bits (Segment Type) of the first byte will be “0” inevitably.

The backplane port of CPU#1 comes first. Port No. of the backplane port is “1.” Because it falls into 1 byte, Extended Link Address Size will be “0.”

Therefore, the first byte will be [01].

Unit No. of ControlNet Unit of CPU#1 is 03 hex, so that Link Address will be

10 hex + 03 hex = 13 hex. Therefore, the second will be [13].

Port No. of ControlNet Port of ControlNet Unit is “2,” so that the third will be [02]. Node Address of the target ControlNet Unit (ControlNet Unit of CPU#2) is “9,” so that the fourth byte will be [09].

Link Address of CPU Unit, on the backplane, is “0.” Therefore, the fifth and sixth bytes will be [01] and [00], respectively.

Link Path will be as follows:

[01] [13] [02] [09] [01] [00]

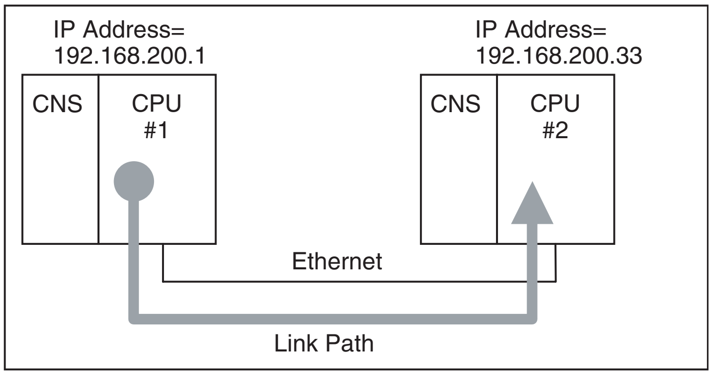
Example 2:

In case of accessing from CPU#1 to CPU#2 via Ethernet

[Structure]

IP Address of CPU#1 = 192.168.200.1

IP Address of CPU#2 = 192.168.200.33



The route is as follows:

Ethernet Port of CPU#1 → CPU#2 (IP Address = 192.168.200.33)

The route above is explained below.

Port Segment is used for setting Link Path. Therefore, the top 3 bits (Segment Type) of the first byte will be “0” inevitably.

In this example, data is transmitted via Ethernet Port. Port No. of Ethernet Port is “2.”

IP Address is used for describing Link Address on Ethernet. IP Address is larger than 1 byte, so that Extended Link Address Size will be “1.” Therefore, the first byte will be [12].

Link Address Size falls into the second byte. IP Address of the target CPU#2 is “192.168.200.33.” The number of the letters of this IP Address should be counted, including dot (“.”) because the whole IP Address must be described with ASCII code. In this case, there are 14 letters (= 0E hex).

Link Path is as follows:

[12] [0E] [31] [39] [32] [2E] [31] [36] [38] [2E] [32] [30] [30] [2E] [33] [33]

Example 3:

In case of accessing from CPU#1 to CPU#3 via CPU#2 and ControlNet Unit

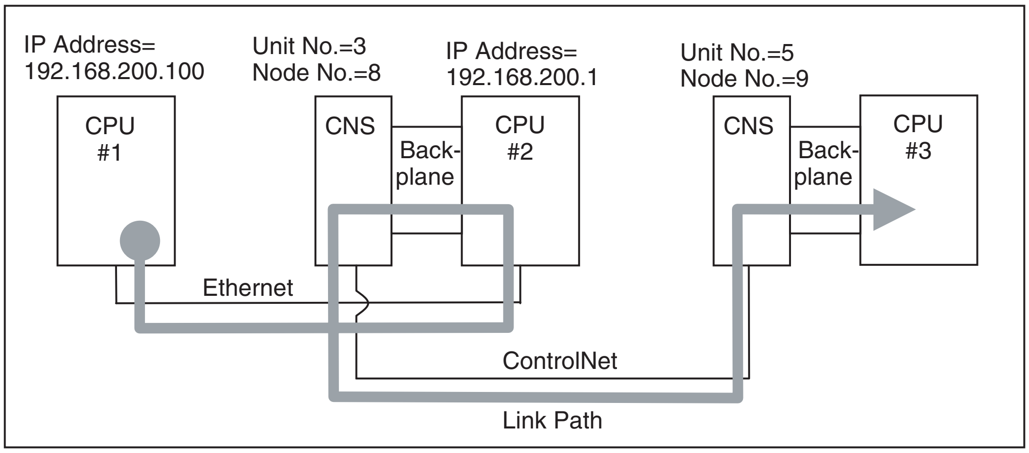
[Structure]

IP Address of CPU#1 = 192.168.200.100

IP Address of CPU #2 = 192.168.200.1

ControlNet Unit of CPU #2: Unit No. = 3, Node Address = 8

ControlNet Unit of CPU #3: Unit No. = 5, Node Address = 9



The route is as follows:

Ethernet Port of CPU#1 → CPU#2 (IP Address = 192.168.200.1) → ControlNet Unit (Unit No. = 3) → ControlNet Port of ControlNet Unit → ControlNet Unit (Node Address = 9) → Backplane Port of ControlNet Unit → CPU#3

The route above is explained below.

Port Segment is used for setting Link Path. Therefore, the top 3 bits (Segment Type) of the first byte will be “0” inevitably.

In this example, data is sent via Ethernet Port. Port No. of Ethernet Port is “2.”

IP Address is employed for Link Address on Ethernet. Because IP Address is larger than 1 byte, Extended Link Address Size will be “1.” Therefore, the first byte will be [12].

Link Address Size falls into the second byte. IP Address of the target CPU#2 is “192.168.200.1.” The number of the letters of this IP Address should be counted, including dot (“.”) because the whole IP Address must be described with ASCII code. In this case, there are 13 letters (= 0D hex). Thirteen (byte) is an odd number, so that [00] should be added at the end in order to make it an even number.

So far, Link Path is [12] [0D] [31] [39] [32] [2E] [31] [36] [38] [2E] [32] [30] [30] [2E] [31] [00].

Now data reaches CPU#2. Then, it is sent to CPU#3 via ControlNet Unit of CPU#2.

In order to go through ControlNet Unit of CPU#2, data must go through Backplane Port of CPU#2. Port No. of Backplane Port is inevitably “1.” Because it falls into 1 byte, Extended Link Address Size will be “0.” Therefore, the first byte will be [01].

Link Address is 10 hex + 03 hex=13 hex because Unit No. of ControlNet Unit of CPU#2 is 03 hex, so that the second byte will be [13].

Because Port No. of ControlNet Port of ControlNet Unit is “2,” the third byte will be [02]. Node Address of the target ControlNet Unit (ControlNet Unit of CPU#3) is “9,” so that the fourth byte will be [09].

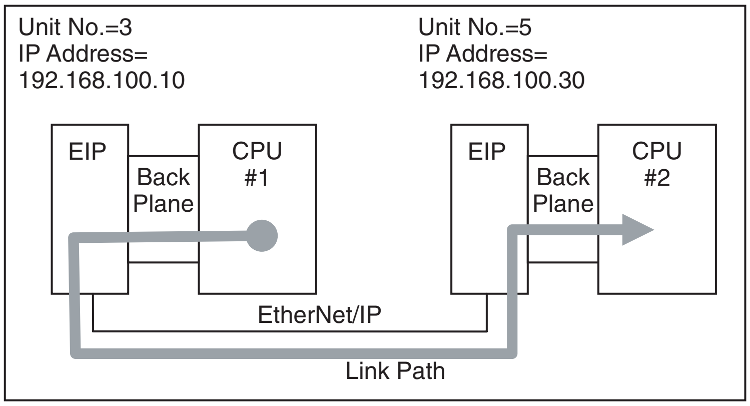
Link Address of CPU Unit, on Backplane, is “0,” so that the fifth and the sixth bytes will be [01] and [00], respectively.

Link Path of this part will be [01] [13] [02] [09] [01] [00].

The above-mentioned two Link Paths being joined, the whole Link Path is as follows:

[12] [0D] [31] [39] [32] [2E] [31] [36] [38] [2E] [32] [30] [30] [2E] [31] [00] [01] [13] [02] [09] [01] [00]

Example 4:
In case of accessing CPU#2 from CPU#1 via EtherNet/IP Unit
[Structure]
EtherNet/IP Unit of CPU#1: Unit No. = 3, IP Address = 192.168.100.10
EtherNet/IP Unit of CPU#2: Unit No. = 5, IP Address = 192.168.100.30



The route is as follows:

Backplane Port of CPU#1 → EtherNet/IP Unit (Unit No. = 3) → EtherNet Port of EtherNet/IP Unit → Ether-Net/IP Unit (IP Address = 192.168.100.30) → Backplane Port of EtherNet/IP Unit → CPU#2

The route above is as follows:
Port Segment is used for setting the Link Path. Therefore, the top 3 bits (Segment Type) of the first byte will always be “0”.
First in the route is the Backplane port of CPU#1. The port number of the Backplane port is always “1”. The number will not be larger than 1 byte, and so the Extended Link Address Size will be “0”. Therefore, first will be “01”.
The unit number of the EtherNet/IP Unit for CPU#1 is “03 hex”, so the Link Address will be 10 hex + 03 hex = 13 hex, and second will be “13”.
The Link Path so far is [01][13].
The port number of the EtherNet/IP port for the EtherNet/IP Unit is “2”, so the third byte is [02]. The IP Address of the target EtherNet/IP Unit is 192.168.100.30. The dots (“.”) must also be counted when counting the number of characters in the IP Address because the whole IP Address must be described with ASCII. In this case, there are 14 characters (= 0E hex).
The Link Path for this part is [12][0E][31][39][32][2E][31][36][38][2E][31][30][30][2E][33][30]. At this point the EtherNet/IP Unit of CPU#2 has been reached.
The Link Address of the CPU Unit on the backplane is “0”, and so the Link Path for this part is [01][00].

Joining the two Link Paths above results in the following complete Link Path:

[01] [13] [12] [0E] [31] [39] [32] [2E] [31] [36] [38] [2E] [31] [30] [30] [2E] [33] [30] [01] [00]

Designation of Variable Name

ANSI Extended Symbol Segment of Data Segment is employed for designating Variable Name.

Example 1:

A variable of “ABCDE” is designated by the following description. The last part, [00], is the padding for making the number of bytes an even number.

[91] [05] [41] [42] [43] [44] [45] [00]

Example 2:

The whole structure of an array variable is the same as in Example 1.

The whole of an array variable, ABC [10], is as follows:

[91] [03] [41] [42] [43] [00]

Example 3:

An element of an array variable is treated as "a member of Instance." Member ID of Logical Segment is employed for describing Instance Member.

An element of an array variable, ABC [3], is described as follows:

[91] [03] [41] [42] [43] [00] [28] [03]

Example 4:

For describing a member of a variable of the structured data type, ANSI Extended Symbol Segment is employed for every dot, ".", which divides members of a structure.

ABC.DE is described as follows:

[91] [03] [41] [42] [43] [00] [91] [02] [44] [45]

Data Access for NE1S Series

CSND Instruction

Overview of CSND Instruction

For NE1S Series, variable data can be accessed to by using CSND Instruction.

CSND Instruction is an instruction to send/receive a message of CIP.

CSND Instruction has three parameters; S data, D data, and C data.

Service data to be sent and transmission control information such as address are set up in S data and C data, respectively. When CSND Instruction is executed, received response data is stored in D data.

FUNC No.	Mnemonic	Instruction Name	Overview of Function
489	(@)CSND	CIP Transmission Instruction	Instruction to send CIP Explicit Message

Execution Condition/Immediate Refreshing Specification		
Execution Condition	Execute cyclically every time it goes to ON.	CSND
	Execute 1 cycle at rising up.	@CSND
	Execute 1 cycle at falling down.	Not supported.
Immediate Refreshing Specification		Not supported.

(@)CSND(489)	
S	Data to be sent
D	Data received.
C	Control data

S Data Details

Offset (word)	
S	Data Size to be sent
S+1	Service Code
S+2	Service Data
:	:
:	:

↑ Send data size (in bytes) ↓

Data Size to be sent	Data Length of Service Code and the following data (unit: byte)
Service Code	CIP Explicit Message Service Code
Service Data	Service Data of Explicit Message Service of CIP Content is different for each Service Code. Service Data is stored in the order from lower byte to upper byte.

D Data Details

Offset (word)		
D	Size of Data received	
D+1	Service Code	↑
D+2	General Status	Receive data size (in bytes)
D+3	Additional Status	
D+4	Response Data	
:	:	
:	:	↓

Size of Data received	Data Length of Service Code and the following data (unit: byte)
Service Code	Transmitted Explicit Message Service Code of CIP The 8th bit is turned ON. For example, when Service Code is 4C hex, it will be CC hex.
General Status	Execution result of the transmitted Service 00 hex indicates a correct end. Other values indicate an error.
Additional Status	Additional information of General Status When Additional Status is larger than 1 Word, only the top 1 Word is stored. Content is different for each Object addressed and Service.
Response Data	Response Data received Received data is stored in the order from lower byte to upper byte.

C Data Details

The C data depends on how the addresses data is specified.
The following section describes how to specify the data using a variable, Class/Instance/Attribute ID, or EPATH type.

Specifying with Variables

Specify the data in the following format if it is accessed using variables.

Offset (word)					
C	Reception Buffer Size				
C+1	0	Comm. Port No.	0	0	
C+2	0	0	0	0	
C+3	Service Execution Time				
C+4	8	0	IOI Size (N)		
C+5	9	1	Variable Name Size		↑
C+6	Variable Name_1		Variable Name _2		IOI Size (in words)
:	:		:		
C+(4+N)	Variable Name _X		Variable Name _Y		↓
C+(4+N)+1	Link Path Size (M)				
C+(4+N)+2	LinkPath_1		LinkPath_2		↑
:	:		:		Link Path Size (in words)
:	:		:		
C+(4+N)+1+M	LinkPath_X		LinkPath_Y		↓

Reception Buffer Size	Size of area for storing received data (area specified by D data). (Unit: WORD) When a response larger than the reception buffer size is received, excess part of the received data will be annulled.
Comm. Port No.	Internal logic port number. There are 8 ports, 0 to 7.
Service Execution Time	Service execution time in the other node. Usually, 0000 hex should be specified.
Variable Name Size	Size of Variable Name to be accessed to. (Unit: Byte) When Variable Name is “ABC,” Variable Name Size will be “3.”
Variable Name_1 to Variable Name _Y	Variable Name to be accessed to. It should be designated with ASCII Code. When Variable Name is “ABC,” Variable Name_1 = 41 hex, Variable Name_2 = 42 hex, Variable Name _3 = 43 hex, and Variable Name _4 = 00 hex.
Link Path Size	Specifies length of Link Path. (Unit: WORD)
Link Path_1 to LinkPath_Y	Link Path Up to 16 nodes can be relayed by specifying the link path. CSND instructions may time out even when network and node status is normal if the message is relayed through more than 16 nodes. If the link path is specified with more than 16 nodes, a constant time must be specified for the service execution time (normally 0000 hex).

Communications Port Number

Eight logical communications ports are provided. Eight communications instructions can be executed simultaneously.

Only one instruction can be executed at one time for each communications port. When executing 9 or more communications instructions, you have to prepare exclusive control.

This communications port number is shared with NETWORK COMMUNICATIONS instruction (CSND) and PROTOCOL MACRO instruction (PMCR). Therefore, you must be careful not to designate the same number for these instructions.

Specifying with Class/Instance/Attribute ID

Specify the data in the following format if it is accessed using Class/Instance/Attribute ID. The IOI is automatically created in EPATH format and sent when the instruction is executed. The logical value, however, is always specified as 16 bits.

Offset (word)				
C	Reception Buffer Size			
C+1	0	Comm. Port No.	0	0
C+2	0	0	0	0
C+3	Service Execution Time			
C+4	8	0	0	
C+5	Class			
C+6	Instance			
C+7	Attribute			
C+8	Link Path Size (M)			
C+9	LinkPath_1		LinkPath_2	
:	:		:	
:	:		:	
C+8+M	LinkPath_X		LinkPath_Y	

↑ Link Path size (in words) ↓

Reception Buffer Size	Size of area for storing received data (area specified by D data). (Unit: WORD) When a response larger than the reception buffer size is received, the part that exceeds the buffer will be cancelled.
Comm. Port No.	Internal logic port number. Eight ports from 0 to 7.
Service Execution Time	Service execution time in the other node. Normally, specify 0000 hex.
Class	Specifies the Class.
Instance	Specifies the Instance ID.
Attribute	Specifies the Attribute. The attribute will be 00 if it is not specified.
Link Path Size	Specifies the length of the Link Path. (Unit: WORD)
Link Path_1 to LinkPath_Y	Link Path Up to 16 nodes can be relayed by specifying the link path. CSND instructions may time out even when network and node status is normal if more than 16 nodes are relayed. If the link path is specified with more than 16 nodes, a constant time must be specified for the service execution time (normally 0000 hex).

Communications Port Number

Eight logical communications ports are provided. Eight communications instructions can be executed simultaneously.

Only one instruction can be executed at one time for each communications port. When executing 9 or more communications instructions, you have to prepare exclusive control.

This communications port number is shared with NETWORK COMMUNICATIONS instruction (CSND) and PROTOCOL MACRO instruction (PMCR). Therefore, you must be careful not to designate the same number for these instructions.

Specifying with EPATH Type

Specify the data in the following format if it is accessed using EPATH type.
Use this format is used if the data is specified using Class/Instance/Attribute and the size of the logical value is specified in detail. If the logical values are all 16 bits, the same message will be sent as when specifying with Class/Instance/Attribute described above.

Offset (word)					
C	Reception Buffer Size				
C+1	0	Comm. Port No.	0	0	
C+2	0	0	0	0	
C+3	Service Execution Time				
C+4	8	0	IOI size (N)		
C+5	IOI_1				↑
:	:				IOI Size (in words)
:	:				
C+(4+N)	ION_N				↓
C+(4+N)+1	Link Path Size (M)				
C+(4+N)+2	LinkPath_1		LinkPath_2		↑
:	:		:		Link Path Size (in words)
:	:		:		
C+(4+N)+1+M	LinkPath_X		LinkPath_Y		↓

Reception Buffer Size	Size of area for storing received data (area specified by D data). (Unit: WORD) When a response larger than the reception buffer size is received, the part that exceeds the buffer will be cancelled.
Comm. Port No.	Internal logic port number. Eight ports from 0 to 7.
Service Execution Time	Service execution time in the other node. Normally, specify 0000 hex.
IOI Size	Specifies the size of IOI in word increments.
IOI_1 to IOI_N	Specifies IOI in EPATH format.
Link Path Size	Specifies the length of the Link Path. (Unit: WORD)
Link Path_1 to LinkPath_Y	Link Path Up to 16 nodes can be relayed by specifying the link path. CSND instructions may time out even when network and node status is normal if more than 16 nodes are relayed. If the link path is specified with more than 16 nodes, a constant time must be specified for the service execution time (normally 0000 hex).

Communications Port Number

Eight logical communications ports are provided. Eight communications instructions can be executed simultaneously.

Only one instruction can be executed at one time for each communications port. When executing 9 or more communications instructions, you have to prepare exclusive control.

This communications port number is shared with NETWORK COMMUNICATIONS instruction (CSND) and PROTOCOL MACRO instruction (PMCR). Therefore, you must be careful not to designate the same number for these instructions.

Specifying IOI

Specify the value for displaying Class/Instance/Attribute when specifying IOI in EPATH format. When specifying in EPATH format, specify three items: Segment Type, Segment Format, and Logical Value. The Logical Value can be specified in 8 bits or 16 bits. Whether 8 bits or 16 bits is used, however, depends on the addressed device. Check the specifications of the addressed device.
For the NE1S Series, operation will be normal if the logical value is 255 or less whether 8 bits or 16 bits is specified. If the logical value is 256 to 65535, specify 16 bits.

• Specifying 8 Bits

Offset (word)		
C+4+n	SegmentType+SegmentFormat	LogicalValue

• Specifying 16 Bits

Offset (word)		
C+4+n	SegmentType+SegmentFormat	00
C+4+(n+1)	LogicalValue	

Examples of Segment Type and Segment Format

	SegmentType + SegmentFormat	
Type	**Specifying 8 bits**	**Specifying 16 bits**
Class ID	20 hex	21 hex
Instance ID	24 hex	25 hex
Attribute ID	30 hex	31 hex

Example Specifying Class ID = 5, Instance ID = 2, and Attribute ID = 1

Offset (word)			
C+4	8	0	03
C+5	20		05
C+6	24		02
C+7	30		01

All are specified in 8 bits.

Example Specifying Class ID = 5 and Instance ID = 2

Offset (word)			
C+4	8	0	04
C+5	21		00
C+6	0005		
C+7	25		00
C+8	0002		

All are specified in 16 bits.

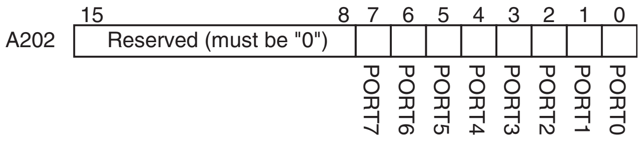
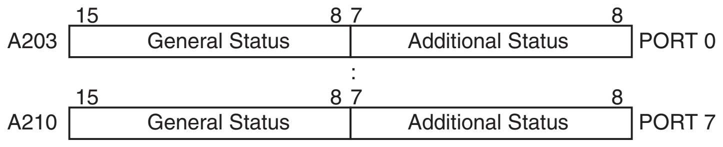
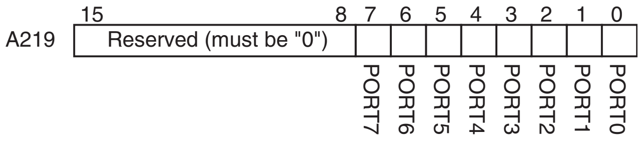
Example Specifying Class ID = 5, Instance ID = 2, and Attribute ID = 1

Offset (word)			
C+4	8	0	05
C+5	21		00
C+6	0005		
C+7	25		00
C+8	0002		
C+6	30		01

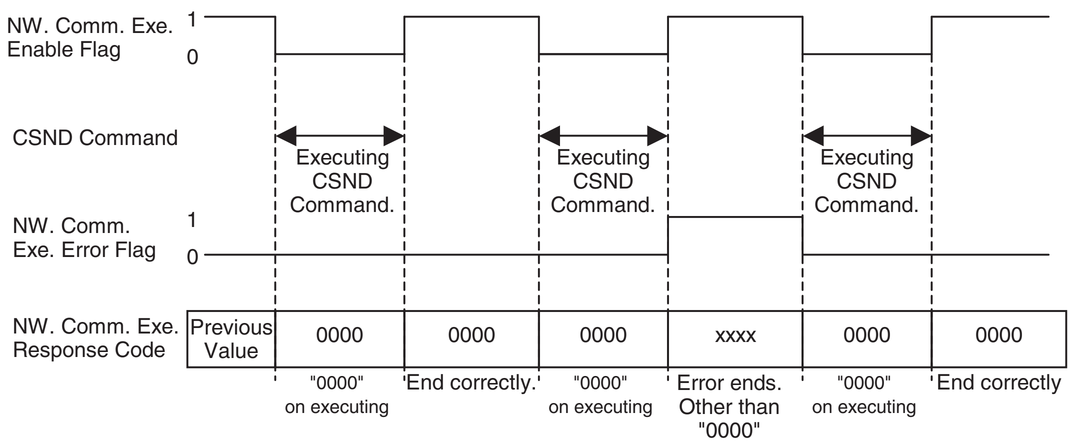
Class ID and Instance ID are specified in 16 bits. Attribute ID is specified in 8 bits.

Flag and Status

The related flags are shown below.

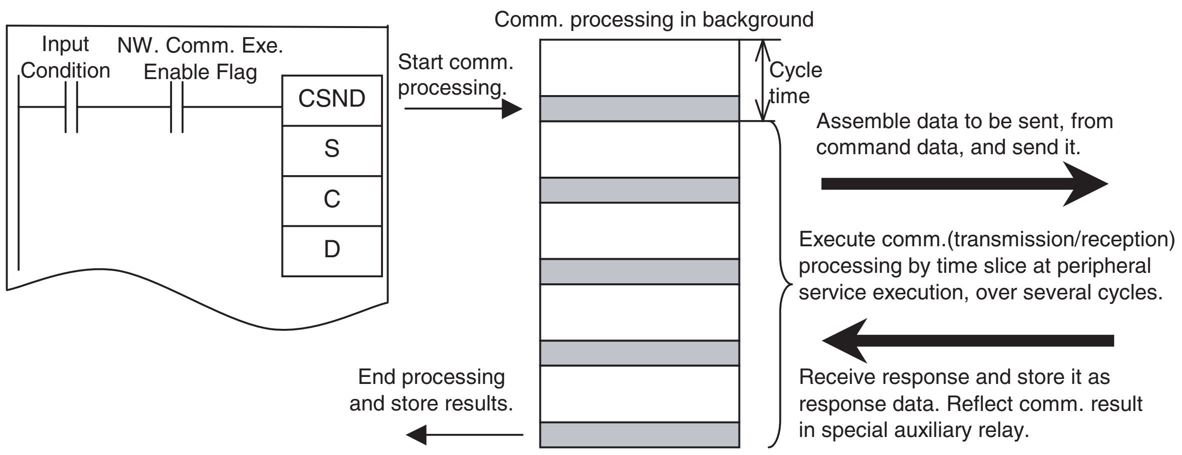
Name	Address	Description
Error Flag	ER	Goes to ON when address range of S, D, and C areas is too large.Goes to ON when Network Instruction Execution Enable Flag is OFF for the communications port specified by C. Otherwise, goes to OFF.
Network Communications Instruction Execution Enable Flag	A20200 to A20207	Goes to 1 (ON) when network communications (CSND or PMCR instruction) is executable.Each bit indicates a communications port. Goes to 0 (OFF) during execution of network communication, and goes to 1 (ON) when execution ends in either case of correct or error.  15 8 7 6 5 4 3 2 1 0 A202 Reserved (must be "0") PORT7 PORT6 PORT5 PORT4 PORT3 PORT2 PORT1 PORT0
Network Communications Response Code	A20300 to A21000	0 during execution of CIP instruction. When processing ends, the value is stored. General Error Code and Additional Error Code are stored in 1 byte of High side and 1 byte of Low side, respectively. When such an error occurs that can be detected in CPU Unit, such as time out or incorrect format, its response is stored only in this area. Note that it is not stored in the response area.  15 8 7 8 A203 General Status Additional Status PORT 0 : 15 8 7 8 A210 General Status Additional Status PORT 7
Network Communications Execution Error Flag	A21900 to A21907	Goes to 1 (ON) when an error occurs during execution of network communication. Each bit corresponds to each communications port. This status is held until next execution of network communication. Note that this bit goes to ON when an error occurs during communications (no data in response area) or when receiving an error response from Target (some data in response area).  15 8 7 6 5 4 3 2 1 0 A219 Reserved (must be "0") PORT7 PORT6 PORT5 PORT4 PORT3 PORT2 PORT1 PORT0

The figure below shows the relation between execution of CSND Instruction and each flag.



Execution Timing

For instructions for network communication, when input conditions are satisfied, the communications processing gets just started and it is in “Communications Port Service” of the peripheral services, in the background, that the actual processing is executed.


Input Condition
NW. Comm. Exe. Enable Flag
CSND
S
C
D
Start comm. processing.
Comm. processing in background
Cycle time
Assemble data to be sent, from command data, and send it.
Execute comm.(transmission/reception) processing by time slice at peripheral service execution, over several cycles.
Receive response and store it as response data. Reflect comm. result in special auxiliary relay.
End processing and store results.

1. When the input conditions are satisfied, if Network Communications Instruction Execution Enable Flag (A20200 to A20207) is 1 (ON) at this moment, each instruction sets 0 (OFF) in Network Communications Instruction Execution Enable Flag (A20200 to A20207), 0 (OFF) in Network Communications Execution Error Flag (A21900 to A21907), and 0000 hex in Network Communications Response Code (A203 to A210), reads C, and starts the communications processing (CIP Instruction Issue/Response Reception).
2. In the peripheral service processing, data to be sent is created based on the operand (See note 1.), and CIP Instruction to the Communications Units are issued.
3. If the issue processing is not completed in one peripheral service, that processing will be executed, by time slice, in the next communications port service.
4. When a response is returned, the response data specified by the operand is updated in the peripheral service (See note 2.). At this moment, Network Communications Instruction Execution Enable Flag (A20200 to A20207) of the special auxiliary relay goes to 1 (ON), and Network Communications Execution Error Flag (A21900 to A21907) and Network Communications Response Code (A203 to A210) are updated.

Note (1) In case of CSND Instruction, it reads S and creates an optional CIP Instruction.

(2) In case of CSND Instruction, D is updated with CIP Response.

Read Service by Variables

Predefined Data Type

Shown below is the case that the data type of a variable is INT/WORD/UINT/UDINT/DWORD/BOOL/REAL.

Service Code=4C hex

Request Data

Word	
S+1	004C
S+2	0100

Response Data

Word		
D+4	Type	00
D+5	Data	
	:	

Type: Data Type Code of a variable which was read. Refer to the following “Data Type Code.”

Structure

The situation is a little different for structured variables.

In case of a structured variable, it has to be confirmed whether or not the specified structured variable is defined correctly. CRC Code (Cyclic Redundant Code) calculated from the structure definition is used for confirming that it is identified with the structure definition.

Although the format of Request Data is the same, that of Response Data is different.

“Type” will be A0 hex. CRC Code is stored in the channel next to Type Field.

Service Code=4C hex

Request Data

Word	
S+1	004C
S+2	0100

Response Data

Word		
D+4	A0	02
D+5	CRC	
D+6	Data	
	:	

Array Variable

In case of an array variable, access to the whole of an array variable is the same as in the case of Predefined Data Type.

Access to an array element is access to “a member of Instance.”

Service Code=4C hex

Request Data

Word	
S+1	004C
S+2	0100

Response Data

Word		
D+4	Type	00
D+5	Data	
:	:	
D+n		

Type: Data Type Code of a variable which was read. Refer to the following “Data Type Code.”

Example 1

In case that an array variable with ten INT-type elements, ArrayData[10], exists and that you access to the whole of this array variable:

Service Code=4C hex

Request Data

Word	
S+1	004C
S+2	0100

Response Data

Word		
D+4	C3	00
:	Data	
	:	
	:	
D+15	Data	

Control Data

Word			
C+5	91	09	
C+6	41	72	“Ar”
C+7	72	61	“ra”
C+8	79	44	“yD”
C+9	61	74	“at”
C+10	61	00	“a”

Example 2

In case that an array variable with ten INT-type elements, ArrayData[10], exists and that you access to the 3rd element of this array variable, ArryData:[2]:

Service Code=4C hex

Request Data

Word	
S+1	004C
S+2	0100

Response Data

Word		
D+4	C3	00
D+5	Data	

Control Data

Word			
C+5	91	09	
C+6	41	72	“Ar”
C+7	72	61	“ra”
C+8	79	44	“yD”
C+9	61	74	“at”
C+10	61	00	“a”
C+11	28	02	28 hex = Logical Segment: Member ID is specified.

02 hex = Member ID = “2.” The 3rd member is specified.

Write Service by Variables

Predefined Data Type

Shown below is the case that the data type of a variable is INT/WORD/UINT/UDINT/DWORD/BOOL/REAL.

Service Code=4D hex

Request Data

Word		
S+1	004D	
S+2	Type	00
S+3	0100	
S+4	Data to be written	
S+5	:	

Response Data

Word	
D+4	None

Type: Data Type Code of a variable which was read. Refer to the following “Data Type Code.”

Data to be written: Set the data to be written in order from the lower byte to the upper byte.

Structure

The situation is a little different for structured variables.

In case of a structured variable, it has to be confirmed whether or not the specified structured variable is defined correctly. CRC Code (Cyclic Redundant Code) calculated from the structure definition is used for confirming that it is identified with the structure definition.

Although the format of Request Data is the same, that of Response Data is different.

“Type” will be A0 hex. CRC Code is stored in the channel next to Type Field.

Service Code=4D hex

Request Data

Word		
S+1	004D	
S+2	A0	02
S+3	CRC	
S+4	0100	
S+5	Data to be written	
	:	

Response Data

Word	
D+4	None

Array Variable

In case of an array variable, access to the whole of an array variable is the same as in the case of Predefined Data Type.

Access to an array element is access to “a member of Instance.”

Service Code=4D hex

Request Data

Word		
S+1	004D	
S+2	Type	00
S+3	0100	
S+4	Data to be written	
:	:	
S+n	Data to be written:	

Response Data

Word	
D+4	None

Type: Data Type Code of a variable which was read. Refer to the following “Data Type Code.”

Data to be written: Set the data to be written in order from the lower byte to the upper byte.

Example 1

In case that an array variable with ten INT-type elements, ArrayData[10], exists and that you access to the whole of this array variable:

Service Code=4D hex

Request Data

Word		
S+1	004D	
S+2	C3	00
S+3	0100	
S+4	Data to be written	
:	:	
S+13	Data to be written:	

Response Data

Word	
D+4	None

Control Data

Word			
C+5	91	09	
C+6	41	72	“Ar”
C+7	72	61	“ra”
C+8	79	44	“yD”
C+9	61	74	“at”
C+10	61	00	“a”

Example 2

In case that an array variable with ten INT-type elements, ArrayData[10], exists and that you access to the 3rd element of this array variable, ArryData:[2]:

Service Code=4D hex

Request Data

Word		
S+1	004D	
S+2	C3	00
S+3	0100	
S+4	Data to be written	

Response Data

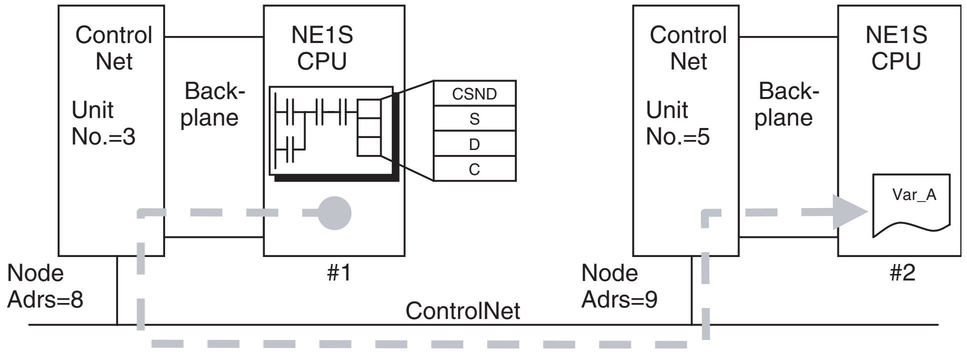
Word	
D+4	None

Control Data

Word			
C+5	91	09	
C+6	41	72	“Ar”
C+7	72	61	“ra”
C+8	79	44	“yD”
C+9	61	74	“at”
C+10	61	00	“a”
C+11	28	02	28 hex = Logical Segment: Member ID is specified. 02 hex = Member ID = “2”. The 3rd member is specified.

Use Example of CSND Instruction

Read WORD-Type Variable via ControlNet



Read a variable of the right CPU Unit #2, “Var_A,” from the left CPU Unit #1, and store it in a variable of #1, “Var_B.”

The route is as follows:

Backplane Port of CPU#1 → ControlNet Unit (Unit No. = 3) → ControlNet Port of ControlNet Unit → ControlNet Unit (Node Address = 9) → Backplane Port of ControlNet Unit → CPU#2

Therefore, the path is described as follows:

[01] [13] [02] [09] [01] [00]

A variable to be accessed to is “Var_A” and it is described as follows:

[91] [05] [56] [61] [72] [5F] [41] [00]

Shown below is the parameter setting for CSND Instruction.

Service Code and Service Data are described in S data. Link Path and IOI (variable name) are described in C data. The received response is stored in D data.

S Data

S	0004	Request Data Size (unit: byte)
S+1	004C	Service Code
S+2	0100	Service Data

C Data

C	000A	Reception Buffer Size (unit: word) (when area of 10 words is specified)
C+1	0300	Set 3 in Communications Port No. of CSND Instruction.
C+2	0000	Reserved (0000 fixed)
C+3	0000	Service Execution Monitoring Timer (usually “0000”)
C+4	8004	Set “1” in the top bit. Describe the size of Variable Name in the unit of word.
C+5	9105	Describe Variable Name according to the method above.
C+6	5661	“Va”
C+7	725F	“r_”
C+8	4100	“A” (Variable Name ends here.)
C+9	0003	Describe Link Path Size in the unit of word.
C+10	0113	Describe Link Path according to the method above.
C+11	0209	
C+12	0100	Link Path ends here.

When S data and C data are set up and the CSND instruction is executed, received data is stored in D. The execution result is stored in General Status area. For the details, refer to the *General Status Code* on page 268. In some cases, not only General Status Code but also Additional Status Code may be added.

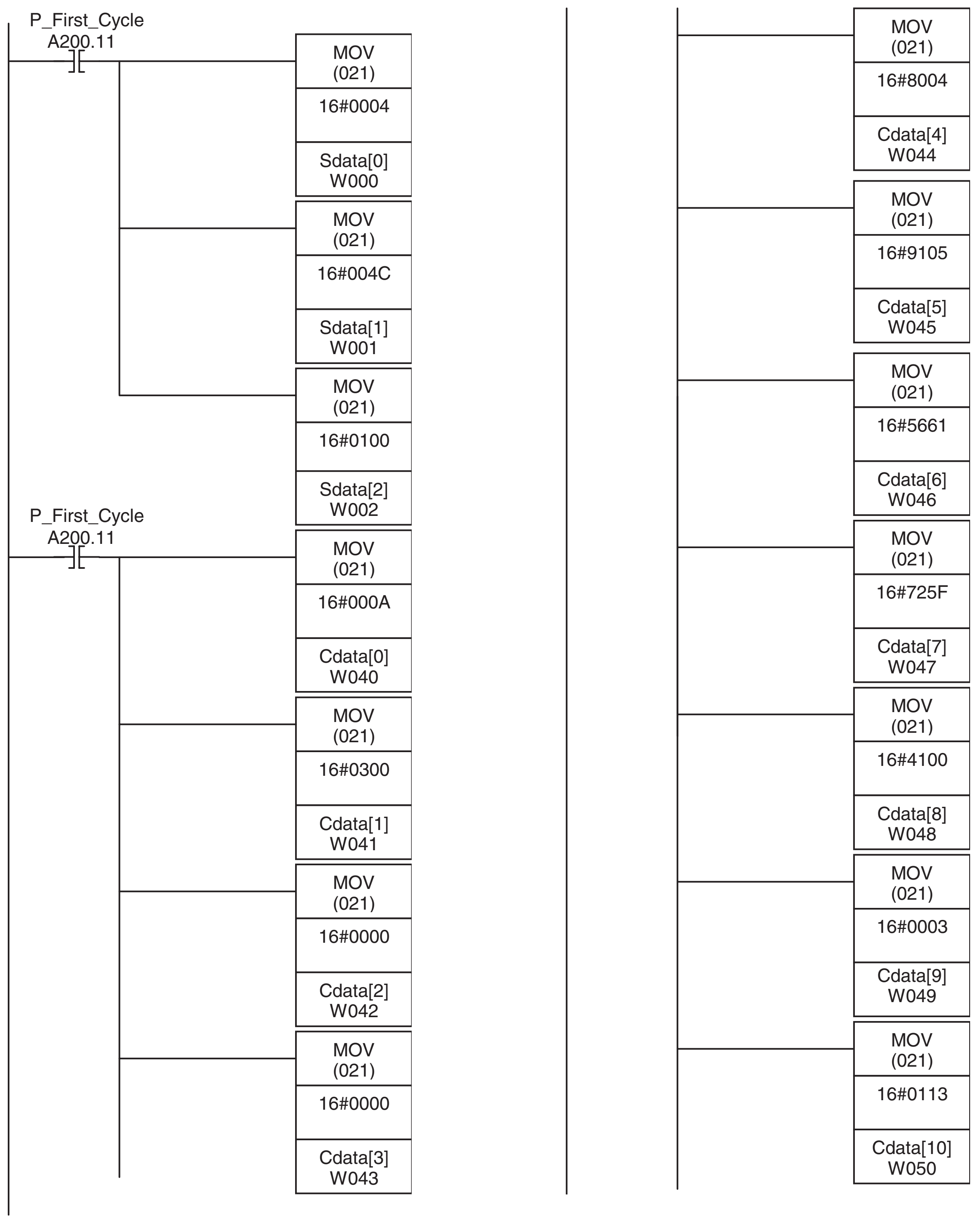
The data which was read is stored in Data of D+4. It is stored in the way of Little Endian (lining in order from the lower byte to the upper byte).

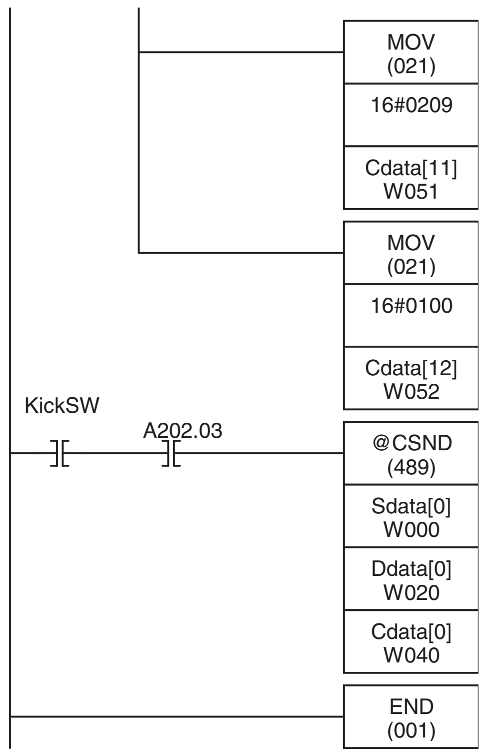
D Data

D	0008	Response Data Size (unit: byte)
D+1	00CC	Service Code (The 8th bit goes to On. 4C=>CC)
D+2	0000	General Status (0000 hex indicates correct end.)
D+3	0000	Additional Status (The upper 2 bytes will be stored if any additional information.)
D+4	D200	Data Type Code (WORD Type = D2)
D+5	Data	Data that was read.

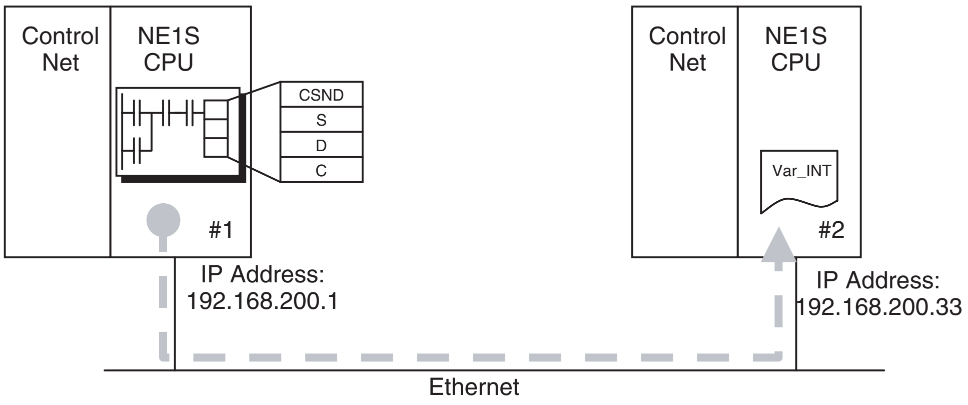
Example of Ladder Program

Name	Data Type	No. of Elements	Address	Comment
Sdata	WORD	16	W000	Operand S of CSND Instruction (data to be sent)
Ddata	WORD	16	W020	Operand S of CSND Instruction (data received)
Cdata	WORD	20	W040	Operand C of CSND Instruction (control)
KickSW	BOOL			Switch to start CSND Instruction
			A202.03	Communications Instruction Execution Enable Flag


P_First_Cycle
A200.11
MOV (021) 16#0004 Sdata[0] W000
MOV (021) 16#004C Sdata[1] W001
MOV (021) 16#0100 Sdata[2] W002
P_First_Cycle
A200.11
MOV (021) 16#000A Cdata[0] W040
MOV (021) 16#0300 Cdata[1] W041
MOV (021) 16#0000 Cdata[2] W042
MOV (021) 16#0000 Cdata[3] W043
MOV (021) 16#8004 Cdata[4] W044
MOV (021) 16#9105 Cdata[5] W045
MOV (021) 16#5661 Cdata[6] W046
MOV (021) 16#725F Cdata[7] W047
MOV (021) 16#4100 Cdata[8] W048
MOV (021) 16#0003 Cdata[9] W049
MOV (021) 16#0113 Cdata[10] W050



Write INT-Type Variable via Ethernet



Write data into a variable of the right CPU Unit #2, “Var_INT,” from the left CPU Unit #1.

The route is as follows:

Ethernet Port of CPU#1 → CPU#2 (IP Address = 192.168.200.33)

The path is described as follows:

[12] [0E] [31] [39] [32] [2E] [31] [36] [38] [2E] [32] [30] [30] [2E] [33] [33]

A variable to be accessed, “Var_INT,” is described as follows:

[91] [07] [56] [61] [72] [5F] [49] [4E] [54] [00]

Shown below is the parameter setting for CSND Instruction.

Service Code and Service Data are described in S data. Link Path and IOI (variable name) are described in C data. The received response is stored in D data.

The data to be written is stored in the way of Little Endian (lining in order from the lower byte to the upper byte).

S Data

S	0008	Request Data Size (unit: byte)
S+1	004D	Service Code
S+2	C300	Data Type Code = C3
S+3	0100	No. of Elements = 1
S+4	Data	Data to be written

C Data

C	000A	Reception Buffer Size (unit: word) (when area of 10 words is specified.)
C+1	0000	When Communications Port No. of CSND Instruction is set to 0.
C+2	0000	Reserved (0000, fixed)
C+3	0000	Service Execution Monitoring Timer (usually, “0000”)
C+4	8005	Set the top bit to “1.” Describe the size of Variable Name in the unit of word.
C+5	9107	Describe Variable Name according to the method above.
C+6	5661	“Va”
C+7	725F	“r_”
C+8	494E	“IN”
C+9	5400	“T” (Variable Name ends here.)
C+10	0008	Describe Link Path Size in the unit of word.
C+11	120E	Describe Link Path according to the method above.
C+12	3139	“19”
C+13	322E	“2.”
C+14	3136	“16”
C+15	382E	“8.”
C+16	3230	“20”
C+17	302E	“0”
C+18	3333	“33” (Link Path ends here.)

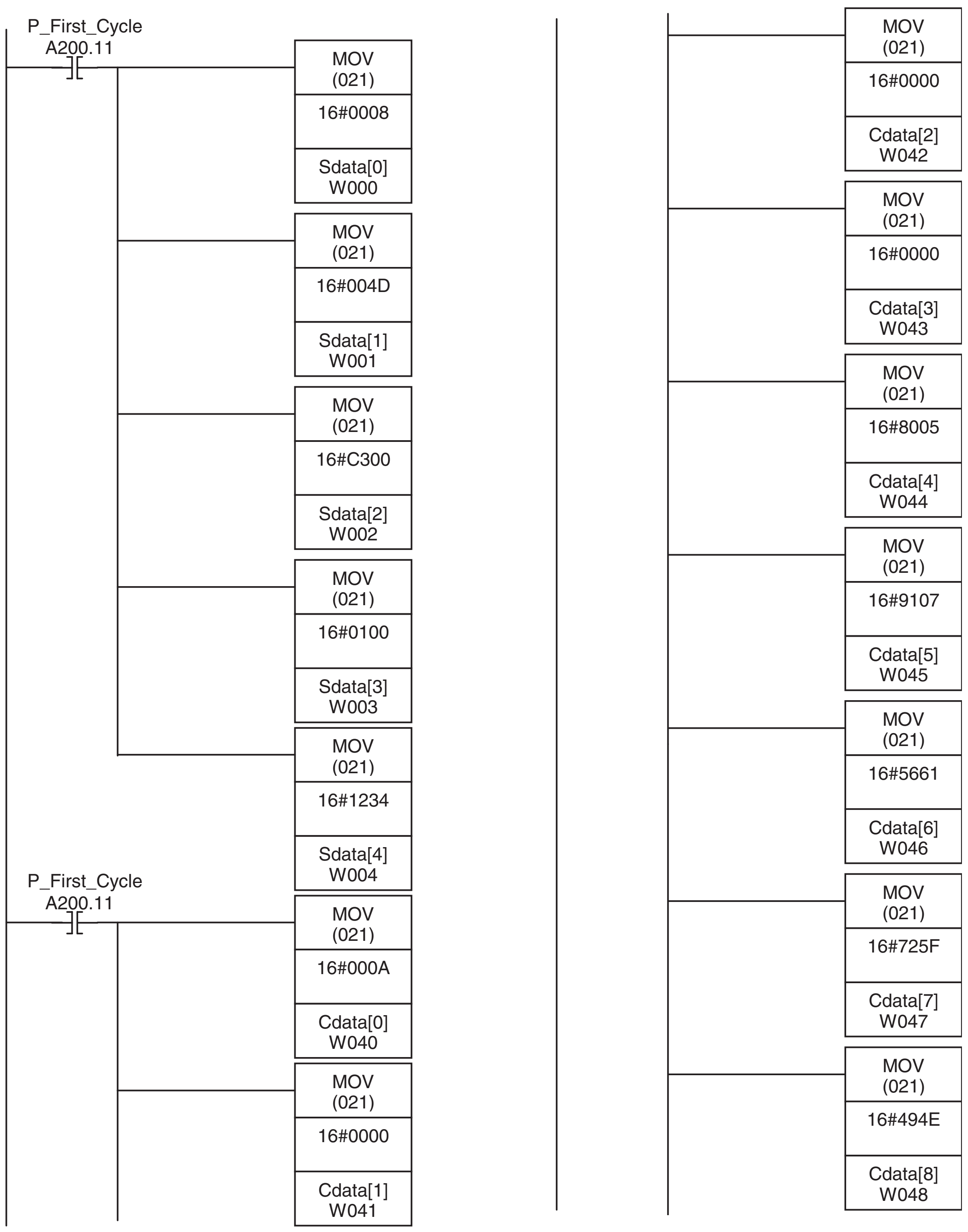
When S data and C data are set up and CSND Instruction is executed, received data is stored in D data. The execution result is stored in General Status area. For the details, refer to the following “General Status Code.” In some cases, not only General Status Code but also Additional Status Code may be added. If no Additional Code exists, “0000” is stored.

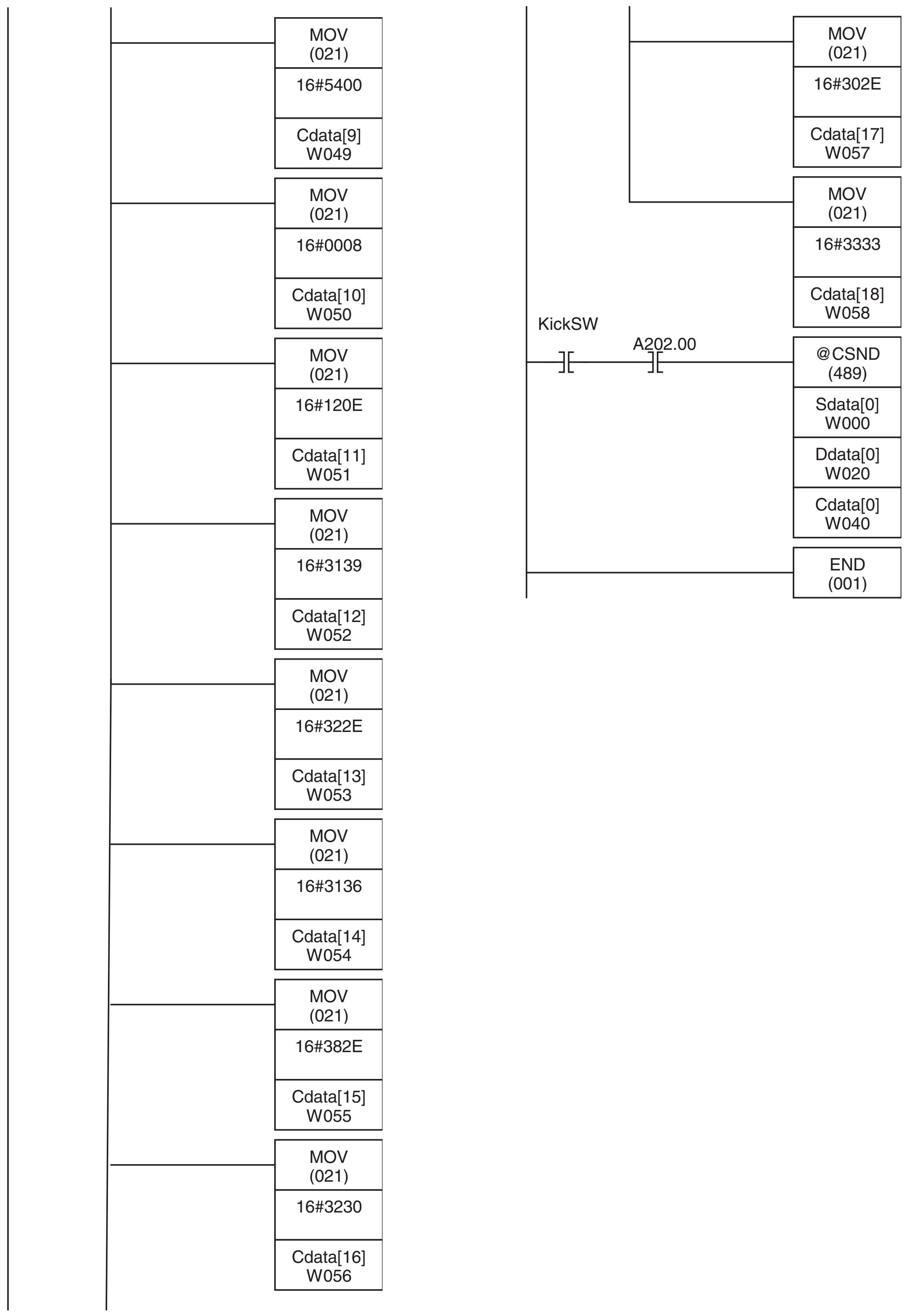
D Data

D	0006	Response Data Size (unit: byte)
D+1	00CD	Service Code (The 8th bit goes to On. 4D => CD)
D+2	0000	General Status (0000 hex indicates correct end.)
D+3	0000	Additional Status (The upper 2 bytes will be stored if any additional information.)

Example of Ladder Program

Name	Data Type	No. of Elements	Address	Comment
Sdata	WORD	16	W000	Operand S of CSND Instruction (data to be sent)
Ddata	WORD	16	W020	Operand S of CSND Instruction (data received)
Cdata	WORD	20	W040	Operand C of CSND Instruction (control)
KickSW	BOOL			Switch to start CSND Instruction
			A202.03	Communications Instruction Execution Enable Flag


P_First_Cycle
A200.11
MOV
(021)
16#0008
Sdata[0]
W000
MOV
(021)
16#004D
Sdata[1]
W001
MOV
(021)
16#C300
Sdata[2]
W002
MOV
(021)
16#0100
Sdata[3]
W003
MOV
(021)
16#1234
Sdata[4]
W004
P_First_Cycle
A200.11
MOV
(021)
16#000A
Cdata[0]
W040
MOV
(021)
16#0000
Cdata[1]
W041
MOV
(021)
16#0000
Cdata[2]
W042
MOV
(021)
16#0000
Cdata[3]
W043
MOV
(021)
16#8005
Cdata[4]
W044
MOV
(021)
16#9107
Cdata[5]
W045
MOV
(021)
16#5661
Cdata[6]
W046
MOV
(021)
16#725F
Cdata[7]
W047
MOV
(021)
16#494E
Cdata[8]
W048


MOV
(021)
16#5400
Cdata[9]
W049
MOV
(021)
16#0008
Cdata[10]
W050
MOV
(021)
16#120E
Cdata[11]
W051
MOV
(021)
16#3139
Cdata[12]
W052
MOV
(021)
16#322E
Cdata[13]
W053
MOV
(021)
16#3136
Cdata[14]
W054
MOV
(021)
16#382E
Cdata[15]
W055
MOV
(021)
16#3230
Cdata[16]
W056
MOV
(021)
16#302E
Cdata[17]
W057
MOV
(021)
16#3333
Cdata[18]
W058
KickSW
A202.00
@CSND
(489)
Sdata[0]
W000
Ddata[0]
W020
Cdata[0]
W040
END
(001)

Data Type

Data Type Code

No.	Data Type Name	Code [hex]	Description
1.	TIMER	01	OMRON Specific Data Type for Timer Instruction in which UP flag and Present Counter are involved.
2.	COUNTER	02	OMRON Specific Data Type for Counter Instruction in which UP flag and Present Counter are involved.
3.	BOOL	C1	Logical Boolean with values TRUE and FALSE TRUE = 01 hex, FALSE = 00 hex
4.	INT	C3	Signed 16-bit integer value
5.	DINT	C4	Signed 32-bit integer value
6.	UINT	C7	Unsigned 16-bit integer value
7.	UDINT	C8	Unsigned 32-bit integer value
8.	REAL	CA	32-bit floating point value
9.	WORD	D2	bit string - 16-bits
10.	DWORD	C3	bit string - 32-bits
11.	STRUCT	A0	Structured variable
12.	STRING	D0	String of letters

Data Placement

The data placement for each data type is described below. The data placement differs between the CPU Unit memory and the data sent and received using the CSND instruction. Be sure to reorder the data when sending or receiving data with the CSND instruction.

BOOL Data

CPU Unit Data Placement

BOOL data in the CPU Unit is at the specified bit location for fixed allocations using address specifications. For automatic allocations, the bit location is automatically allocated. Information on forced set/reset status cannot be read or written from the program.

CSND Instruction Data Placement

Bit

Wd	15 8	7 0
+0	Data	Information on Forced Set/Reset Status

Data: True = 01 hex, False = 00 hex

Information on Forced Set/Reset Status: Forced = 01 hex, Not Forced = 00 hex

Only data can be written, i.e., forced status information cannot be written.

BOOL Data (Whole Array)

CPU Unit Data Placement

Bit

Wd	15							8	7							0
+0	D15							D8	D7							D8
+1															D17	D16
:								:								

D0 to Dn: True = 01 hex, False = 00 hex

Allocations always start a bit 0 for arrays. It is not possible to start allocations from any other bit (e.g., starting from bit 4 is not possible).

CSND Instruction Data Placement

Bit

Wd	15							8	7							0
+0	D7							D0	D15							D8
+1							D17	D16								
:								:								

D0 to Dn: True = 01 hex, False = 00 hex

WORD, INT, or UINT Data

CPU Unit Data Placement

Bit

Wd	15 8	7 0
+0	Data (H)	Data (L)

CSND Instruction Data Placement

Bit

Wd	15 8	7 0
+0	Data (L)	Data (H)

DWORD, DINT, or UDINT Data

CPU Unit Data Placement

Bit

Wd	15 8	7 0
+0	Data (LH)	Data (LL)
+1	Data (HH)	Data (HL)

CSND Instruction Data Placement

Bit

Wd	15 8	7 0
+0	Data (LL)	Data (LH)
+1	Data (HL)	Data (HH)

TIMER Data

CPU Unit Data Placement

The data size and meaning of TIMER variables depend on the instruction that is used. When using instructions that require a bit operand (e.g., LD, AND, OR, or, OUT), the Completion Flag is accessed. When using instructions that require other operands, the PV is accessed.

Bit

Wd	15 8	7 0
+0	Current Value (H)	Current Value (L)

CSND Instruction Data Placement

Bit

Wd	15 8	7 0
+0	Current Value (L)	Current Value (H)
+1	Up Flag	Information of Forced Reset/Reset

Current Value: Current value of the timer

Up Flag: Time up = 01 hex, Others = 00 hex

Information of Forced Reset/Reset: Forced = 01 hex, Not Forced = 00 hex

Only current values can be written.

COUNTER Data

CPU Unit Data Placement

The data size and meaning of COUNTER variables depend on the instruction that is used. When using instructions that require a bit operand (e.g., LD, AND, OR, or, OUT), the Completion Flag is accessed. When using instructions that require other operands, the PV is accessed.

Bit

Wd	15 8	7 0
+0	Current Value (H)	Current Value (L)

CSND Instruction Data Placement

Bit

Wd	15 8	7 0
+0	Current Value (L)	Current Value (H)
+1	Up Flag	Information of Forced Reset/Reset

Current Value: Current value of the timer

Up Flag: Count up = 01 hex, Others = 00 hex

Information of Forced Reset/Reset: Forced = 01 hex, Not Forced = 00 hex

Only current values can be written.

STRING Data

CPU Unit Data Placement

Bit

Wd	15 8	7 0
+0	Data 1	Data 2
+1	:	:
:	Data m	Data n
+N	0 hex	0 hex

All data up to 00 hex is treated as string data. If there is an odd number of characters in the string, 00 hex is stored in the lower byte of the word.

CSND Instruction Data Placement

Bit

Wd	15 8	7 0
+0	Size	00
+1	Data 1	Data 2
:	:	:
+N	Data m	Data n

Size: Data Size (unit = byte)

Data: Data of letters. When Data Size is an odd number, pad the lower side of the last word with 00 hex.

REAL Data

READ data conforms to the definition of single-precision floating-point data in IEEE 754. Single-precision data uses 32 bits in the following format.

Actual value = (–1) s2^{e-127} (1.f)

s	e		f	
31	30	23	22	0

CPU Unit Data Placement

Bit

Wd	15		8	7	0
+0	f bits 08 to 15			f bits 00 to 07	
+1	s	e		f bits 16 to 22	

CSND Instruction Data Placement

Bit

Wd	15		8	7	0
+0	f bits 00 to 07			f bits 08 to 15	
+1	e	f bits 16 to 22		e	

Response Code

General Status Code

General Status Code is stored in the response reception area, D+2, after execution of CSND Instruction is completed.

This code is also reflected in the upper byte of A203 to A210. When Additional Code is added, only part for 1 byte is reflected in the lower byte of A203 to A230.

General Status Code (hex)	Status Name	Description of Status
00	Success	Service was successfully performed by the object specified.
01	Connection failure	A connection related service failed along the connection path.
02	Resource unavailable	Resources needed for the object to perform the requested service were unavailable.
03	Invalid parameter value	See Status Code 20 hex, which is the preferred value to use for this condition.
04	Path segment error	The path segment identifier or the segment syntax was not understood by the processing node. Path processing shall stop when a path segment error is encountered.
05	Path destination unknown	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. Path processing shall stop when a path destination unknown error is encountered.
06	Partial transfer	Only part of the expected data was transferred.
07	Only part of the expected data was transferred.	The messaging connection was lost.
08	Service not supported	The requested service was not implemented or was not defined for this Object Class/Instance.
09	Invalid attribute value	Invalid attribute data detected.
0A	Attribute list error	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non-zero status.
0B	Already in requested mode/state	The object is already in the mode/state being requested by the service.
0C	Object state conflict	The object cannot perform the requested service in its current mode/state.
0D	Object already exists	The requested instance of object to be created already exists.
0E	Attribute not settable	A request to modify a non-modifiable attribute was received.
0F	Privilege violation	A permission/privilege check failed.
10	Device state conflict	The device's current mode/state prohibits the execution of the requested service.
11	Reply data too large	The data to be transmitted in the response buffer is larger than the allocated response buffer
12	Fragmentation of a primitive value	The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type.
13	Not enough data	The service did not supply enough data to perform the specified operation.
14	Attribute not supported	The attribute specified in the request is not supported.
15	Too much data	The service supplied more data than was expected.
16	Object does not exist	The object specified does not exist in the device.
17	Service fragmentation sequence not in progress	The fragmentation sequence for this service is not currently active for this data.
18	No stored attribute data	The attribute data of this object was not saved prior to the requested service.
19	Store operation failure	The attribute data of this object was not saved due to a failure during the attempt.

General Status Code (hex)	Status Name	Description of Status
1A	Routing failure (request packet too large)	The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service.
1B	Routing failure (response packet too large)	The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service.
1C	Missing attribute list entry data	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behavior.
1D	Invalid attribute value list	The service is returning the list of attributes supplied with status information for those attributes that were invalid.
1E	Embedded service error	An embedded service resulted in an error.
1F	Vendor specific error	A vendor specific error has been encountered. The Additional Code Field of the Error Response defines the particular error encountered. Use of this General Error Code should only be performed when none of the Error Codes presented in this table or within an Object Class definition accurately reflect the error.
20	Invalid parameter	A parameter associated with the request was invalid. This code is used when a parameter does not meet the requirements of this specification and/or the requirements defined in an Application Object Specification.
21	Write-once value or medium already written	An attempt was made to write to a write-once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.
22	Invalid Reply Received	An invalid reply is received (e.g. reply service code does not match the request service code, or reply message is shorter than the minimum expected reply size). This status code can serve for other causes of invalid replies.
23-24		Reserved by CIP for future extensions
25	Key Failure in path	The Key Segment that was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.
26	Path Size Invalid	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.
27	Unexpected attribute in list	An attempt was made to set an attribute that is not able to be set at this time.
28	Invalid Member ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute.
29	Member not settable	A request to modify a non-modifiable member was received.
2A	Group 2 only server general failure	This error code may only be reported by DeviceNet group 2 only servers with 4K or less code space and only in place of Service not supported, Attribute not supported and Attribute not settable.
2B-CF		Reserved by CIP for future extensions
D0-FF	Reserved for Object Class and service errors	This range of error codes is to be used to indicate Object Class specific errors. Use of this range should only be performed when none of the Error Codes presented in this table accurately reflect the error that was encountered.

Example of Additional Status in case that General Status Is 01 Hex. (Status of Connection Manager Object)

General Status (hex)	Additional Status (hex)	Explanation
01	0100	Connection in Use or Duplicate Forward Open.
01	0103	Transport Class and Trigger combination not supported
01	0106	Ownership Conflict
01	0107	Connection not found at target application.
01	0108	Invalid Connection Type. Indicates a problem with either the Connection Type or Priority of the Connection.
01	0109	Invalid Connection Size
01	0110	Device not configured
01	0111	RPI not supported. May also indicate problem with connection time-out multiplier, or production inhibit time.
01	0113	Connection Manager cannot support any more connections
01	0114	Either the Vendor Id or the Product Code in the key segment did not match the device
01	0115	Product Type in the key segment did not match the device
01	0116	Major or Minor Revision information in the key segment did not match the device
01	0117	Invalid Connection Point
01	0118	Invalid Configuration Format
01	0119	Connection request fails since there is no controlling connection currently open.
01	011A	Target Application cannot support any more connections
01	011B	RPI is smaller than the Production Inhibit Time.
01	0203	Connection cannot be closed since the connection has timed out
01	0204	Unconnected Send timed out waiting for a response.
01	0205	Parameter Error in Unconnected Send Service
01	0206	Message too large for Unconnected message service
01	0207	Unconnected acknowledge without reply
01	0301	No buffer memory available
01	0302	Network Bandwidth not available for data
01	0303	No Tag filters available
01	0304	Not Configured to send real-time data
01	0311	Port specified in Port Segment Not Available
01	0312	Link Address specified in Port Segment Not Available
01	0315	Invalid Segment Type or Segment Value in Path
01	0316	Path and Connection not equal in close
01	0317	Either Segment not present or Encoded Value in Network Segment is invalid.
01	0318	Link Address to Self Invalid
01	0319	Resources on Secondary Unavailable
01	031A	Connection already established
01	031B	Direct connection already established
01	031C	Miscellaneous
01	031D	Redundant connection mismatch
01	031F	No connection resources exist for target path
01	0320-07FF	Vendor specific

Error Code Unique to CSND Instruction

When CSND Instruction itself turns to be an error (incorrect parameter, etc.), this code is reflected in A203 to A210.

Error Code (hex)	Error Name	Cause	Remarks
2001	Inappropriate Request Data Length	The value for the Request Data Size in the first field of S is less than 4 (specified as less than 4 bytes), or that the total request data size exceeds the specified amount (512 bytes).	The message will not be sent and the Communications port Error Flag for the corresponding port will turn ON.
2002	Inappropriate Response Data Length	The value for the Response Data Size in the first field of D is less than 8 (specified as less than 8 bytes).	
2003	Inappropriate Control Data Length	The Link Address Size in the sixth field of C exceeds the specified amount (512 bytes).	
0201	Insufficient Response Area	The CIP response data length exceeds the Response Buffer Size in the first field of C.	The response will be discarded and the Communications Port Error Flag for the corresponding port will turn ON.
---	EPATH Error	CIP Segment Encoding Errors (EPATH Errors) are listed below. For details on CIP segments, refer to Appendix C: Data Management in CIP Common Specification Volume 1 Release 1.0.	The message will not be sent and the Communications port Error Flag for the corresponding port will turn ON.
2041		The value specified for the Path Type in the IOI Path area of operand C (C+4, bit 15) is not 1 or 0 (i.e., Class/Instance/Attribute or EPATH is not specified).	
2042		The value for the IOI Size is 0 when the Path Type is specified as 1 in the IOI Path Area of operand C (C+4, bit 15).	
0401		Logical Segment/Data Segment is not specified when the Path Type is specified as 1 in the IOI Path Area of operand C (C+4, bit 15).	
2043		The Logical Type is not specified as Class/Instance/Attribute.	
0411		The Logical Format is not specified as 8, 16, or 32bits.	
0421		The Segment Sub-Type is not specified as ANSI Extended Symbol Segment.	
2044		The IOI length is set to 0x100 or higher for the IOI Path in operand C.	
1F03	Maximum Send/Receive Length Exceeded	The send request exceeded the CIP maximum message size.	
1F02	Link Path Error	An undefined link path was specified. An undefined port such as port No. 0x05 or 0x10 was specified.	

Error Code (hex)	Error Name	Cause	Remarks
0281	Internal Error	An internal error occurred.	The message will not be sent and the Communications port Error Flag for the corresponding port will turn ON.
0282			
0283			
0284			
1F81			
1F01	Response Timeout	A timeout occurred in the processing at the target. The message was discarded during communications processing (the message frame length exceeded the maximum length, etc.).	The Communications Port Error Flag for the corresponding port will turn ON.

Appendix E
PLC Setup for CJ2 CPU Units

The section describes the PLC Setup items for CJ2 CPU Units that can be set from the NE Programmer. For information on PLC Setup items for CJ2 CPU units, refer to the *CJ2 CPU Unit Software User's Manual* (Cat. No. W473).

To display the settings window for the PLC Setup, right-click the configuration and select ***System Setup***.

Startup Settings

Startup IOM Hold

Settings	Function	Related flags and words	New setting's effectiveness
Invalid Valid Default: Invalid	This setting determines whether or not the status of the IOM Hold Bit (A50012) is retained at startup. When you want all of the data in I/O Memory to be retained when the power is turned ON, turn ON the IOM Hold Bit and set this setting to 1 (ON).	A50012 (IOM Hold Bit)	At startup

Startup Force Status

Settings	Function	Related flags and words	New setting's effectiveness
Invalid Valid Default: Invalid	This setting determines whether or not the status of the Forced Status Hold Bit (A50013) is retained at startup. When you want all of the bits that have been force-set or force-reset to retain their forced status when the power is turned ON, turn ON the Forced Status Hold Bit and set this setting to *Retained*.	A50013 (Forced Status Hold Bit)	At startup

Startup Operating Mode

Settings	Function	Related flags and words	New setting's effectiveness
PROGRAM MONITOR RUN Default: RUN	This setting determines whether the Startup Mode will be the mode set on the Programming Console's mode switch or the mode set here in the PLC Setup.	---	At startup

Execution Setting

Settings	Function	Related flags and words	New setting's effectiveness
Invalid: Do not start operation. Valid: Start operation. Default: Invalid	This property sets the startup condition. It sets whether to start CPU Unit operation without waiting for Special I/O Units or CPU Bus Units to finish starting when the power supply is turned ON in RUN mode or MONITOR mode.	---	At startup

CPU Settings

Detect Battery Error

Settings	Function	Related flags and words	New setting's effectiveness
Detect Nondetect Default: Detect	This setting determines whether CPU Unit battery errors are detected. If this setting is set to *Detect* and a battery error is detected, the ERR/ALM indicator on the CPU Unit will flash and the Battery Error Flag (A40204) will be turned ON, but CPU Unit operation will continue.	A40204 (Battery Error Flag)	Next cycle

Detect Duplicate Refresh Error

Settings	Function	Related flags and words	New setting's effectiveness
Detect Nondetect Default: Detect	This setting determines whether interrupt task errors are detected. If this setting is set to *Detect* and an interrupt task error is detected, the ERR/ALM indicator on the CPU Unit will flash and the Interrupt Task Error Flag (A40213) will be turned ON, but CPU Unit operation will continue.	A40213 (Interrupt Task Error Flag)	Next cycle

Register FAL to Error Log

Settings	Function	Related flags and words	New setting's effectiveness
Register Unregister Default: Register	This setting determines if user-defined FAL errors created with FAL(006) and time monitoring for FPD(269) will be recorded in the error log (A100 to A199).	---	Whenever FAL(006) is executed (every cycle)

Table Data Process Instructions

Settings	Function	Related flags and words	New setting's effectiveness
Disable Enable Default: Disable	This setting determines if Table Data Instructions will be processed over multiple cycle times (i.e., processed in the background).	---	Start of operation

String Data Process Instructions

Settings	Function	Related flags and words	New setting's effectiveness
Disable Enable Default: Disable	This setting determines if Text String Data Instructions will be processed over multiple cycle times (i.e., processed in the background).	---	Start of operation

Data Shift Process Instructions

Settings	Function	Related flags and words	New setting's effectiveness
Disable Enable Default: Disable	This setting determines if Data Shift Instructions will be processed over multiple cycle times (i.e., processed in the background).	---	Start of operation

Com Port Number

Settings	Function	Related flags and words	New setting's effectiveness
0 to 7: Communications ports 0 to 7 (internal logical ports) Default: 0	The communications port number (internal logical port) that will be used for background execution. The same port number is used for all instructions.	---	Start of operation

Stop CPU on Instruction Error _

Settings	Function	Related flags and words	New setting's effectiveness
Continue Stop Default: Continue	This setting determines whether instruction errors (instruction processing errors (ER) and illegal access errors (AER)) are treated as non-fatal or fatal errors. When this setting is set to *Stop*, CPU Unit operation will be stopped if the ER or AER Flags is turned ON (even when the AER Flag is turned ON for an indirect DM/EM BCD error). Related Flags: A29508 (Instruction Processing Error Flag) A29509 (Indirect DM/EM BCD Error Flag) A29510 (Illegal Access Error Flag)	A29508, A29509, A29510 (If this setting is set to *Continue*, these flags won't be turned ON even if an instruction error occurs.)	Start of operation

Timings Settings

Schedule Interrupt Interval

Settings	Function	Related flags and words	New setting's effectiveness
10 ms 1.0 ms 0.1 ms Default: 10 ms	Sets the time interval for the scheduled interrupt task.	---	Start of operation. . (Can't be changed during operation.)

Cycle Monitor Time (Minimum Cycle Time

Settings	Function	Related flags and words	New setting's effectiveness
0.2 to 32000.0 ms (0.1-ms increments) Default: 0 ms	Set to the desired value to specify a minimum cycle time. If the cycle time is less than this setting, it will be extended until this time passes. Leave this setting at 0 ms for a variable cycle time.	---	Start of operation. (Can't be changed during operation.)

Cycle Time

Settings	Function	Related flags and words	New setting's effectiveness
Disabled Enabled Default: Disabled (1,000 ms)	Set to *Enabled* to use a cycle monitor time setting other than the default value of 1 s.	A264 and A265 (Present Cycle Time)	Start of operation. (Can't be changed during operation.)

Set Cycle Monitor Time

Settings	Function	Related flags and words	New setting's effectiveness
10 to 4,000 ms Default: 1,000 ms	This setting is valid only when *Enable Minimum Cycle Time Setting* is set to *Enable*. The Cycle Time Too Long Flag (A40108) will be turned ON if the cycle time exceeds this setting.	A40108 (Cycle Time Too Long Flag)	Start of operation. (Can't be changed during operation.)

Power OFF Interrupt

Settings	Function	Related flags and words	New setting's effectiveness
Invalid Valid Default: Invalid	When this setting is set to *Enabled*, the power OFF interrupt task will be executed when power is interrupted.	---	At startup or at start of operation. (Can't be changed during operation.)

Power OFF Detection Time

Settings	Function	Related flags and words	New setting's effectiveness
0 to 10 ms (1-ms increments) Default: 0	This setting determines how much of a delay there will be from the detection of a power interruption (approximately 10 to 25 ms after the power supply voltage drops below 85% of the rated value for an AC power supply or below 80% for a DC power supply) to the confirmation of a power interruption. The default setting is 0 ms. When the power OFF interrupt task is enabled, it will be executed when the power interruption is confirmed. If the power OFF interrupt task is disabled, the CPU will be reset and operation will be stopped.	---	At startup or at start of operation. (Can't be changed during operation.)

SIOU Refresh Settings

Item	Settings	Function	Related flags and words	New setting's effectiveness
Cyclic Refreshing of Units 0 to 95	Enabled Disabled Default: Enabled	These settings determine whether data will be exchanged between the specified Unit and the Special I/O Unit's allocated words (10 words/Unit) during cyclic refreshing for Special I/O Units.	---	Start of operation.

Unit Settings

Item	Settings	Function	Related flags and words	New setting's effectiveness
Rack 0, Slot 0	No filter 0.5 ms 1 ms 2 ms 4 ms 8 ms 16 ms 32 ms Default: 8 ms	Sets the input response time (ON response time = OFF response time) for CJ-series Basic I/O Units. The default setting is 8 ms and the setting range is 0.5 ms to 32 ms. This value can be increased to reduce the effects of chattering and noise, or it can be reduced to allow reception of shorter input pulses.	A220 to A259: Actual input response times for Basic I/O Units	At startup.
Rack 0, Slot 1				
Rack 0, Slot 2				
Rack 0, Slot 3				
Rack 0, Slot 4				
Rack 0, Slot 5				
Rack 0, Slot6				
Rack 0, Slot 7				
Rack 0, Slot 8				
Rack 0, Slot 9				
Rack 1, Slots 0 to 9				
Rack 2, Slots 0 to 9				
Rack 3, Slots 0 to 9				
Rack 4, Slots 0 to 9				
Rack 5, Slots 0 to 9				
Rack 6, Slots 0 to 9				
Rack 7, Slots 0 to 9				

Communications Settings

⚠ **Caution** For information on how to make serial port settings, refer to the *CJ2 CPU Unit Hardware User's Manual* (Cat. No. W472).

Serial Port Port Mode

Settings	Function	Related flags and words	New setting's effectiveness
Default (SYSWAY) Tool bus SYSWAY Serial gateway NT Link (1:N)	This setting determines which serial communications mode will be used for the USB port.		Next cycle.

Serial Port Data Length

Settings	Function	Related flags and words	New setting's effectiveness
7 bits 8 bits Default: 7 bits	These settings are valid only when the communications mode is set to *Host link*. These settings are also valid only when *USB Port Port Settings* is set to *Manual*.		Next cycle.

Serial Port Stop Bits

Settings	Function	Related flags and words	New setting's effectiveness
2 bits 1 bit Default: 2 bits	These settings are valid only when the communications mode is set to Host link. These settings are also valid only when *USB Port Port Settings* is set to *Manual*.		Next cycle.

Serial Port Parity

Settings	Function	Related flags and words	New setting's effectiveness
Parity - Even Parity - Odd Parity - None Default: Parity - Even	These setting is valid only when the communications mode is set to Host link. These settings are also valid only when *USB Port Port Settings* is set to *Manual*.		Next cycle.

Serial Port Baud Rate (bps)

Settings	Function	Related flags and words	New setting's effectiveness
300 600 1,200 2,400 4,800 9,600 19,200 38,400 57,600 115,200 Default: 9,600	Refer to the CPU Unit manual to set the appropriate baud rate for the mode to be used.		Next cycle.

Serial Port SYSWAY Mode Unit Number

Settings	Function	Related flags and words	New setting's effectiveness
0 to 31 Default: 0	This setting determines the CPU Unit's unit number when it is connected in a 1-to-N (N=2 to 32) Host Link.		Next cycle.

Serial Port NT Link Max

Settings	Function	Related flags and words	New setting's effectiveness
0 to 7 Default: 1	This setting determines the highest unit number of PT that can be connected to the PLC.		Next cycle.

Serial Port Response Monitor Time

Settings	Function	Related flags and words	New setting's effectiveness
0 to 25,500 ms Default: 0 ms	Set the response monitor time in increments of 100 ms for the setting of the RS-232C port.		Next cycle.

FINS Write Protection

FINS Write Protection Settings

Settings	Function	Related flags and words	New setting's effectiveness
Enable or disable FINS write protection Default: Disable	This setting specifies whether or not to enable write protection on FINS commands sent to the CPU Unit through the network. (The write protection does not apply to direct serial connections.)	---	As soon as changed.
The following settings can be used to allow write commands from up to specified nodes in the specified networks. (Write-protection will not apply to the specified nodes.) If the following settings are not made, write-protection will apply to all nodes other than the local node.			
Network address: 0 to 127 Node address: 1 to 255 (255 = all nodes) Default: Network address 0, node address 1	Specifies the network address and node address of the 1st node that is not write-protected.	---	As soon as changed.
:	:	:	:
Network address: 0 to 127 Node address: 1 to 255 (255 = all nodes) Default: Network address 0, node address 1	Specifies the network address and node address of the 32nd node that is not write-protected.	---	As soon as changed.

General Settings

Enabling the Total Service Time Setting

Settings	Function	Related flags and words	New setting's effectiveness
Default (10%) Default: 10%	Use this setting to set the total peripheral service time to a value other than 10%.	---	Next cycle.

Setting the Total Service Time

Settings	Function	Related flags and words	New setting's effectiveness
0.1 to 3276.7 ms (0.1-ms increments) Default: 0	This setting sets the total service time.	---	Next cycle.

Appendix F
Ethernet Settings for CJ2 CPU Units

List of Settings

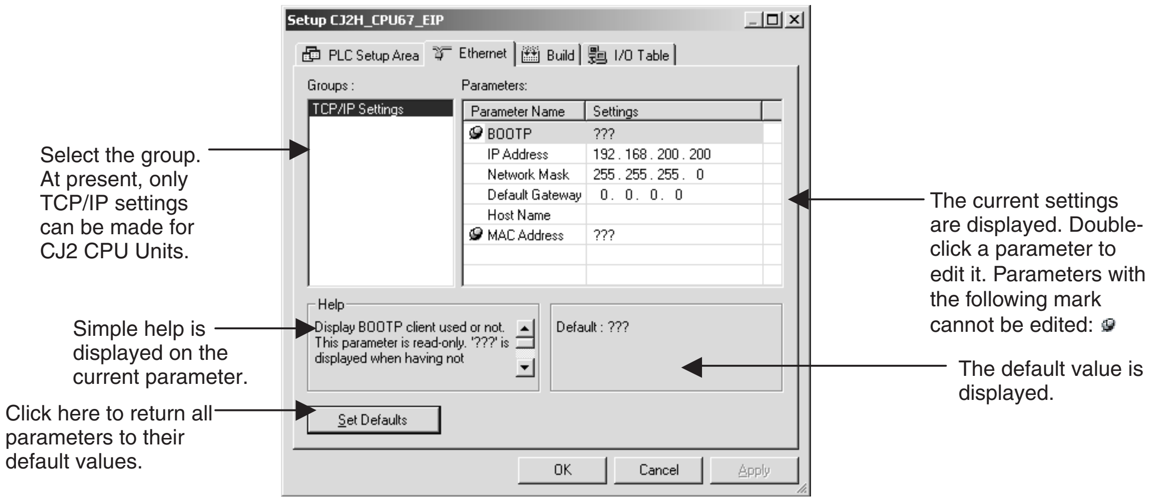
Group	Setting	Default
TCP/IP Settings	BOOTP	Unused
	IP Address	192.168.200.200
	Network Mask	255.255.255.0
	Default Gateway	0.0.0.0
	Host Name	
	MAC Address	

Basic Setting Procedures for Ethernet Settings

Ethernet Settings are made on the Ethernet Tab Page of the Configuration Setting Window of NE Programmer. The Ethernet Tab Page is shown below.

1,2,3...

1. Right-click the configuration name (i.e., the PLC name) in the Project Workspace and select ***System Configuration*** from the popup menu.
   The Configuration Setting Window will be displayed.
2. Click the **Ethernet** Tab in the Configuration Setting Window.
   The following Ethernet Setting Window will be displayed.



Parameter Name	Settings
BOOTP	???
IP Address	192 . 168 . 200 . 200
Network Mask	255 . 255 . 255 . 0
Default Gateway	0 . 0 . 0 . 0
Host Name	
MAC Address	???

Display BOOTP client used or not. This parameter is read-only. '???' is displayed when having not

Default : ???

Select the group. At present, only TCP/IP settings can be made for CJ2 CPU Units.

Simple help is displayed on the current parameter.

Click here to return all parameters to their default values.

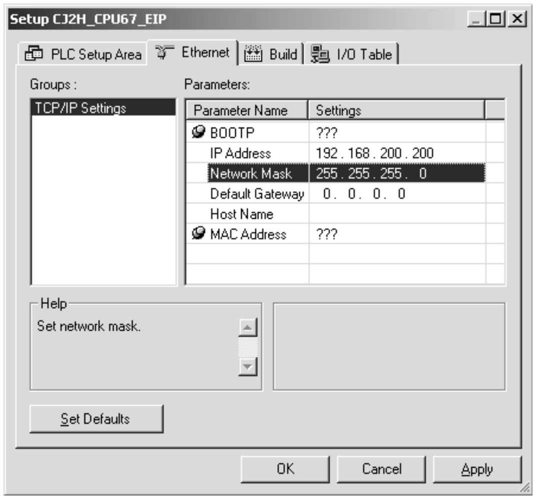
The current settings are displayed. Double-click a parameter to edit it. Parameters with the following mark cannot be edited:

The default value is displayed.

3. Select the group and set the parameters.
4. After completing all settings, click the **OK** Button.
5. Download the settings to the CPU Unit. Select ***Controller - Download to Controller***, select ***Ethernet Setting***, and then click the **OK** Button.
   Refer to *SECTION 7 Online Operation* for information on downloading and other online operations for the NE Programmer.
6. Restart the CPU Unit.

TCP/IP Settings

The settings in the TCP/IP group are shown below.


Setup CJ2H_CPU67_EIP

Parameter Name	Settings
BOOTP	???
IP Address	192 . 168 . 200 . 200
Network Mask	255 . 255 . 255 . 0
Default Gateway	0 . 0 . 0 . 0
Host Name	
MAC Address	???

Help

Set network mask.

These settings are described individually below.

BOOTP

This window only displays whether the BOOTP function is enabled or not. The BOOTP setting cannot be changed here.

Note BOOTP
The BOOTP function enables automatically setting the IP address, network mask, and default gateway by obtaining IP configuration information from the BOOTP server on the network.
A BOOTP server must be installed separately on the network.

IP Address

- As a rule, the IP address is set on the rotary switches on the CPU Unit first and then the setting on the Ethernet Setting Tab Page from the NE Programmer is set to match the setting on the rotary switches.
- The network portion of the IP address, however, must be changed using the Ethernet Settings on the NE Programmer.

CPU Unit Rotary Switch Settings

Changes to the Ethernet Settings are effective the next time the CPU Unit is started.

For information on setting rotary switch for CJ2 CPU Units, refer to the *SYSMAC CJ Series CJ2 CPU Unit Hardware User's Manual* (Cat. No. W472).

Network Mask

A subnet mask can be set.

Class	Subnet mask
Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

Set the same subnet mask for all nodes on the same subnetwork.

Default Gateway

Select the IP address of the default gateway.

Do not set anything if a default gateway is not being used.

Host Name

Set a host name for the IP address of the CPU Unit.

Do not set anything if a host name is not being used.

MAC Address

The MAC Address of the CPU Unit is displayed.

Index

Revision History

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

Cat. No. Z918-E1-01

Revision code

The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
01	August 2008	Original production

OMRON

Terms and Conditions of Sale

1. Offer; Acceptance. These terms and conditions (these "Terms") are deemed part of all quotes, agreements, purchase orders, acknowledgments, price lists, catalogs, manuals, brochures and other documents, whether electronic or in writing, relating to the sale of products or services (collectively, the "Products") by Omron Electronics LLC and its subsidiary companies ("Omron"). Omron objects to any terms or conditions proposed in Buyer's purchase order or other documents which are inconsistent with, or in addition to, these Terms.
2. Prices; Payment Terms. All prices stated are current, subject to change without notice by Omron. Omron reserves the right to increase or decrease prices on any unshipped portions of outstanding orders. Payments for Products are due net 30 days unless otherwise stated in the invoice.
3. Discounts. Cash discounts, if any, will apply only on the net amount of invoices sent to Buyer after deducting transportation charges, taxes and duties, and will be allowed only if (i) the invoice is paid according to Omron's payment terms and (ii) Buyer has no past due amounts.
4. Interest. Omron, at its option, may charge Buyer 1-1/2% interest per month or the maximum legal rate, whichever is less, on any balance not paid within the stated terms.
5. Orders. Omron will accept no order less than $200 net billing.
6. Governmental Approvals. Buyer shall be responsible for, and shall bear all costs involved in, obtaining any government approvals required for the importation or sale of the Products.
7. Taxes. All taxes, duties and other governmental charges (other than general real property and income taxes), including any interest or penalties thereon, imposed directly or indirectly on Omron or required to be collected directly or indirectly by Omron for the manufacture, production, sale, delivery, importation, consumption or use of the Products sold hereunder (including customs duties and sales, excise, use, turnover and license taxes) shall be charged to and remitted by Buyer to Omron.
8. Financial. If the financial position of Buyer at any time becomes unsatisfactory to Omron, Omron reserves the right to stop shipments or require satisfactory security or payment in advance. If Buyer fails to make payment or otherwise comply with these Terms or any related agreement, Omron may (without liability and in addition to other remedies) cancel any unshipped portion of Products sold hereunder and stop any Products in transit until Buyer pays all amounts, including amounts payable hereunder, whether or not then due, which are owing to it by Buyer. Buyer shall in any event remain liable for all unpaid accounts.
9. Cancellation; Etc. Orders are not subject to rescheduling or cancellation unless Buyer indemnifies Omron against all related costs or expenses.
10. Force Majeure. Omron shall not be liable for any delay or failure in delivery resulting from causes beyond its control, including earthquakes, fires, floods, strikes or other labor disputes, shortage of labor or materials, accidents to machinery, acts of sabotage, riots, delay in or lack of transportation or the requirements of any government authority.
11. Shipping; Delivery. Unless otherwise expressly agreed in writing by Omron:
    a. Shipments shall be by a carrier selected by Omron; Omron will not drop ship except in "break down" situations.
    b. Such carrier shall act as the agent of Buyer and delivery to such carrier shall constitute delivery to Buyer;
    c. All sales and shipments of Products shall be FOB shipping point (unless otherwise stated in writing by Omron), at which point title and risk of loss shall pass from Omron to Buyer; provided that Omron shall retain a security interest in the Products until the full purchase price is paid;
    d. Delivery and shipping dates are estimates only; and
    e. Omron will package Products as it deems proper for protection against normal handling and extra charges apply to special conditions.
12. Claims. Any claim by Buyer against Omron for shortage or damage to the Products occurring before delivery to the carrier must be presented in writing to Omron within 30 days of receipt of shipment and include the original transportation bill signed by the carrier noting that the carrier received the Products from Omron in the condition claimed.
13. Warranties. (a) 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. (b) 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. (c) 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.omron247.com or contact your Omron representative for published information.
14. 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.
15. Indemnities. Buyer shall indemnify and hold harmless Omron Companies and their employees from and against all liabilities, losses, claims, costs and expenses (including attorney's fees and expenses) related to any claim, investigation, litigation or proceeding (whether or not Omron is a party) which arises or is alleged to arise from Buyer's acts or omissions under these Terms or in any way with respect to the Products. Without limiting the foregoing, Buyer (at its own expense) shall indemnify and hold harmless Omron and defend or settle any action brought against such Companies to the extent based on a claim that any Product made to Buyer specifications infringed intellectual property rights of another party.
16. Property; Confidentiality. Any intellectual property in the Products is the exclusive property of Omron Companies and Buyer shall not attempt to duplicate it in any way without the written permission of Omron. Notwithstanding any charges to Buyer for engineering or tooling, all engineering and tooling shall remain the exclusive property of Omron. All information and materials supplied by Omron to Buyer relating to the Products are confidential and proprietary, and Buyer shall limit distribution thereof to its trusted employees and strictly prevent disclosure to any third party.
17. Export Controls. Buyer shall comply with all applicable laws, regulations and licenses regarding (i) export of products or information; (iii) sale of products to "forbidden" or other proscribed persons; and (ii) disclosure to non-citizens of regulated technology or information.
18. Miscellaneous. (a) Waiver. No failure or delay by Omron in exercising any right and no course of dealing between Buyer and Omron shall operate as a waiver of rights by Omron. (b) Assignment. Buyer may not assign its rights hereunder without Omron's written consent. (c) Law. These Terms are governed by the law of the jurisdiction of the home office of the Omron company from which Buyer is purchasing the Products (without regard to conflict of law principles). (d) Amendment. These Terms constitute the entire agreement between Buyer and Omron relating to the Products, and no provision may be changed or waived unless in writing signed by the parties. (e) Severability. If any provision hereof is rendered ineffective or invalid, such provision shall not invalidate any other provision. (f) Setoff. Buyer shall have no right to set off any amounts against the amount owing in respect of this invoice. (g) Definitions. As used herein, "including" means "including without limitation"; and "Omron Companies" (or similar words) mean Omron Corporation and any direct or indirect subsidiary or affiliate thereof.

Certain Precautions on Specifications and Use

1. 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 but the following is a non-exhaustive list of applications for which particular attention must be given:
   (i) Outdoor use, uses involving potential chemical contamination or electrical interference, or conditions or uses not described in this document.
   (ii) Use in consumer products or any use in significant quantities.
   (iii) Energy control systems, combustion systems, railroad systems, aviation systems, medical equipment, amusement machines, vehicles, safety equipment, and installations subject to separate industry or government regulations.
   (iv) Systems, machines and equipment that could present a risk to life or property. Please know and observe all prohibitions of use applicable to this Product.
   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'S PRODUCT IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.
2. Programmable Products. Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.
3. 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.
4. 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.
5. 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.

OMRON

Automation...simple...powerful.