OMRON

Machine Automation Controller NX1P

Practices Guide for NX1P Programming

NX1P2-[][][][]

SYSMAC-SE20[][]

Practices Guide

SYSMAC always in control

P122-E1-02

© OMRON, 2017

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this guide is subject to change without notice. Every precaution has been taken in the preparation of this guide. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Sysmac and SYSMAC are trademarks or registered trademarks of OMRON Corporation in Japan and other countries for OMRON factory automation products.
- Windows is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.
- EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany
- EtherNet/IP™ is a trademark of ODVA.
- Celeron, Intel, and Intel Core are the trademarks of Intel Corporation in the USA and other countries.
- Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.

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

Introduction

Thank you for purchasing an NX-series NX1P2 CPU Unit and the Sysmac Studio.
This *NX1P Programming Practices Guide for Beginners* (hereafter referred to as “this Guide”) describes the differences in programming between the NX1P and traditional controllers and the programming procedures using the Sysmac Studio that are required to use an NX1P2 CPU Unit for the first time. You can perform the procedures that are presented in this Guide to quickly gain a basic understanding of the NX1P2 CPU Units and the Sysmac Studio. This Guide does not contain safety information and other details that are required for actual use. Thoroughly read and understand the manuals for all of the devices that are used in this Guide to ensure that the system is used safely. Review the entire contents of these materials, including all safety precautions, precautions for safe use, and precautions for correct use.
For the startup and operating instructions for motion control, refer to the *NJ/NX-series Startup Guide for Motion Control* (Cat. No. W514).

Intended Audience

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

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

Applicable Products

This Guide covers the following products.

- NX1P2 CPU Units of NX-series Machine Automation Controllers
- Automation Software Sysmac Studio

Special Information

The icons that are used in this Guide are described below.

Precautions for Correct Use

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

Additional Information

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

Terms and Conditions Agreement

Warranty, Limitations of Liability

Warranties

- **Exclusive Warranty**

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

- **Limitations**

OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.
Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

- **Buyer Remedy**

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

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

Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.
Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.
NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

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

Disclaimers

Performance Data

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

Change in Specifications

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

Errors and Omissions

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

Automation Software Sysmac Studio

● WARRANTY

- The warranty period for the Software is one year from the date of purchase, unless otherwise specifically agreed.
- If the User discovers defect of the Software (substantial non-conformity with the manual), and return it to OMRON within the above warranty period, OMRON will replace the Software without charge by offering media or download from OMRON's website. And if the User discovers defect of media which is attributable to OMRON and return it to OMRON within the above warranty period, OMRON will replace defective media without charge. If OMRON is unable to replace defective media or correct the Software, the liability of OMRON and the User's remedy shall be limited to the refund of the license fee paid to OMRON for the Software.

● LIMITATION OF LIABILITY

- 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 PARTICULAR PURPOSE. IN NO EVENT, OMRON WILL BE LIABLE FOR ANY LOST PROFITS OR OTHER INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES
- ARISING OUT OF USE OF THE SOFTWARE. OMRON SHALL HAVE NO LIABILITY FOR DEFECT OF THE SOFTWARE BASED ON MODIFICATION OR ALTERNATION TO THE SOFTWARE BY THE USER OR ANY THIRD PARTY.
- OMRON SHALL HAVE NO LIABILITY FOR SOFTWARE DEVELOPED BY THE USER OR ANY THIRD PARTY BASED ON THE SOFTWARE OR ANY CONSEQUENCE THEREOF.

● APPLICABLE CONDITIONS

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

● CHANGE IN SPECIFICATION

The software specifications and accessories may be changed at any time based on improvements and other reasons.

● 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

- When building a system, check the specifications for all devices and equipment that will make up the system and make sure that the OMRON products are used well within their rated specifications and performances. Safety measures, such as safety circuits, must be implemented in order to minimize the risks in the event of a malfunction.
- Thoroughly read and understand the manuals for all devices and equipment that will make up the system to ensure that the system is used safely. Review the entire contents of these materials, including all safety precautions, precautions for safe use, and precautions for correct use.
- Confirm all regulations, standards, and restrictions that the system must adhere to.

Software Licenses and Copyrights

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

Related Manuals

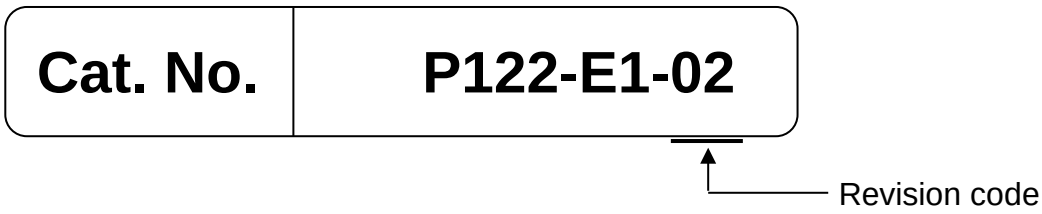
The followings are the manuals related to this manual. Use these manuals for reference.

Manual name	Cat. No.	Model	Application	Description
NX-series NX1P2 CPU Unit Hardware User's Manual	W578	NX1P2- □□□□	Learning the basic specifications of the NX1P2 CPU Units, including introductory information, designing, installation, and maintenance. Mainly hardware information is provided.	An introduction to the entire NX1P2 system is provided along with the following information on the CPU Unit. • Features and system configuration • Introduction • Part names and functions • General specifications • Installation and wiring • Maintenance and inspection
NX-series NX1P2 CPU Unit Built-in I/O and Option Board User's Manual	W579	NX1P2- □□□□	Learning about the details of functions only for an NX-series NX1P2 CPU Unit and an introduction of functions for an NJ/NX-series CPU Unit.	Of the functions for an NX1P2 CPU Unit, the following information is provided. • Built-in I/O • Serial Communications Option Boards • Analog I/O Option Boards An introduction of following functions for an NJ/NX-series CPU Unit is also provided. • Motion control functions • EtherNet/IP communications functions • EtherCAT communications functions
NJ/NX-series CPU Unit Software User's Manual	W501	NX701- □□□□ NJ501- □□□□ NJ301- □□□□ NJ101- □□□□ NX1P2- □□□□	Learning how to program and set up an NJ/NX-series CPU Unit. Mainly software information is provided.	The following information on a Controller built with an NJ/NX-series CPU Unit. • CPU Unit operation • CPU Unit features • Initial setting • Programming based on IEC 61131-3 language specifications Use this manual together with the *NX-series NX1P2 CPU Unit Hardware User's Manual* (Cat. No. W578).
NJ/NX-series Instructions Reference Manual	W502	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Learning detailed specifications on the basic instructions of an NJ/NX-series CPU Unit.	The instructions in the instruction set (IEC 61131-3 specifications) are described. When programming, use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Motion Control User's Manual	W507	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Learning about motion control settings and programming concepts.	The settings and operation of the CPU Unit and programming concepts for motion control are described. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).

Manual name	Cat. No.	Model	Application	Description
NJ/NX-series Motion Control Instructions Reference Manual	W508	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Learning about the specifications of the motion control instructions.	The motion control instructions are described. When programming, use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500), *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501),and *NJ/NX-series CPU Unit Motion Control User's Manual* (Cat. No. W507).
NJ/NX-series CPU Unit Built-in EtherCAT® Port User's Manual	W505	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Using the built-in EtherCAT port on an NJ/NX-series CPU Unit.	Information on the built-in EtherCAT port is provided. This manual provides an introduction and information on the configuration, features, and setup. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series CPU Unit Built-in EtherNet/IP™ Port User's Manual	W506	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Using the built-in EtherNet/IP port on an NJ/NX-series CPU Unit.	Information on the built-in EtherNet/IP port is provided. Information on the basic setup, tag data links, and other features is provided. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manual* (Cat. No. W501).
NJ/NX-series Troubleshooting Manual	W503	NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ NX1P2-□□□□	Learning about the errors that may be detected in an NJ/NX-series Controller.	Concepts on managing errors that may be detected in an NJ/NX-series Controller and information on individual errors are described. Use this manual together with the *NJ-series CPU Unit Hardware User's Manual* (Cat. No. W500) and *NJ/NX-series CPU Unit Software User's Manua*l (Cat. No. W501).
Sysmac Studio Version 1 Operation Manual	W504	SYSMAC-SE2 □□□	Learning about the operating procedures and functions of the Sysmac Studio.	The operating procedures of the Sysmac Studio is described.
NJ/NX-series Startup Guide for Motion Control	W514	NX1P2-□□□□ NX701-□□□□ NJ501-□□□□ NJ301-□□□□ NJ101-□□□□ SYSMAC-SE20□□ R88M-1□ R88D-1SN□-ECT	Learning startup procedures and Sysmac Studio operating procedures for someone that will use NJ/NX series motion control functions for the first time.	The operations from hardware assembly through debugging for axis parameter settings, simple one-axis positioning, and two-axis linear interpolation are described.

Revision History

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



Revision code	Date	Revised content
01	September 2017	Original production
02	December 2018	Correction of mistakes

CONTENTS

Introduction 3

Terms and Conditions Agreement 4

Precautions 7

Related Manuals 8

Revision History 10

1 Programming the NX1P 14

1-1 Overview 15

1-2 Features of NX1P Programming 16

1-2-1 Challenges in Development and Solutions Using the NX1P 16

1-2-2 Easy to Add Programs 17

1-2-3 Easy Motion Programming 18

1-2-4 Structured Text Language for Easy Mathematical Processing 19

1-3 Programming with Variables 20

1-3-1 Programming the NX1P 20

1-3-2 Data Types 23

1-3-3 Benefit of Using Data Types 24

1-3-4 International Standard IEC 61131-3 25

1-4 Programming Software 26

1-4-1 Programming Software Sysmac Studio 26

1-4-2 Simulations 27

2 Before You Begin 28

2-1 System Configuration and Devices 29

2-1-1 Overview 29

2-1-2 Wiring 30

2-2 Installing the Sysmac Studio 32

2-2-1 Installing the Sysmac Studio 32

2-2-2 Requirements for Installation 32

3 Ladder Programming 33

3-1 Programming with the Sysmac Studio 35

3-1-1 Programming Procedure 35

3-1-2 Creating a Project 35

3-2 Parts of the Sysmac Studio Window 37
3-2-1 Screen for Configurations and Setup 37
3-2-2 Screen for Programming 37
3-3 Assigning Variables to Terminals 38
3-3-1 Variable Names for Terminal Numbers 38
3-3-2 I/O Map Setting 39
3-3-3 Checking Wiring 41
3-4 Ladder Programming 42
3-4-1 Inserting Circuit Parts 42
3-4-2 Keyboard Mapping 42
3-4-3 Rules 43
3-5 Example of a Basic Ladder Program 44
3-5-1 Practice of Programming a Ladder Diagram 44
3-5-2 Writing the Algorithm 45
3-5-3 Program Check 47
3-5-4 Saving the Program 48
3-5-5 Checking Operation on the NX1P 49
3-5-6 Checking Operation on the Simulator 50
3-5-7 Example of a Program Error (Offline) 52
3-5-8 Example of an Error Occurred During Operation 52
3-6 Example of a Ladder Program Using a Timer Instruction 53
3-6-1 Self-holding Rung 53
3-6-2 On-Delay Timer (TON) Instruction 54
3-6-3 Exercise: Energy Saving Escalator 58
3-6-4 Checking the Operation of the Program 59
3-6-5 Checking the Operation of the Program (Watch Tab Page) 60
3-7 Example of a Ladder Program Using Date and Time 62
3-7-1 Programming the NX1P Using Date and Time 62
3-7-2 Exercise: Continuous Operating Time of Escalator 62
3-8 Fundamentals of Programming to Reduce Development Time 66
3-8-1 POUs (Program Organization Units) 66
3-8-2 Programs and Execution Priorities (Tasks) 66
3-8-3 Functions (FUNs) and Function Blocks (FBs) 68
3-8-4 Sections 69
3-8-5 Types of Variables 70

4 Creating Programs to Handle Data 73

4-1 Variables Used for Data Processing 74
4-1-1 Arrays 74
4-2 Programming Exercise 75
4-2-1 Application Example 75
4-2-2 Programming 75
4-2-3 Creating a Project 76
4-2-4 Configuring Analog Option Board Settings 77
4-2-5 Assigning Variables to the Option Board and Input Terminal 77
4-2-6 Program Example 78

4-2-7 Creating an Array 79
4-2-8 Entering Programming Code 80
4-2-9 Checking the Operation of the Program 81
4-2-10 Referring Values of Array Variables 83

5 Motion FB Programming 84

5-1 Motion FB Programming 85
5-1-1 Motion FB Programming 85
5-1-2 Programming Procedure 85

5-2 Adding a Servo Drive and Setting the Parameters 86
5-2-1 Registering a Servo Drive 86
5-2-2 Registering the Axis 87
5-2-3 Setting the Axis Parameters 87

5-3 Creating a Program 89
5-3-1 Overview of the Ladder Program 89
5-3-2 Motion FBs to Use 89
5-3-3 Writing the Ladder Program 90

5-4 Data Tracing 93
5-4-1 Checking the Operation with Data Traces 93

5-5 3D Simulation 95
5-5-1 Starting 3D Simulation 95

6 ST Programming 97

6-1 Overview of ST Programming 98
6-1-1 Advantages of ST Language 98
6-1-2 ST Programs Including Constructs 98
6-1-3 Structure of ST and Example 99
6-1-4 Operators 99

6-2 NX1P Programming in ST 100
6-2-1 Writing an ST Program for NX1P 100

6-3 ST Programming Exercise 101
6-3-1 Exercise of Numerical Calculation Programming 101
6-3-2 Programming Procedures 102
6-3-3 Checking the Program 104
6-3-4 Checking the Operation of the ST Program 104

1

1 Programming the NX1P

This section describes the fundamental elements of programming an NX1P Machine Automation Controller.

1-1 Overview....1-15

1-2 Features of NX1P Programming....1-16

1-2-1 Challenges in Development and Solutions Using the NX1P.... 1-16

1-2-2 Easy to Add Programs.... 1-17

1-2-3 Easy Motion Programming 1-18

1-2-4 Structured Text Language for Easy Mathematical Processing 1-19

1-3 Programming with Variables....1-20

1-3-1 Programming the NX1P.... 1-20

1-3-2 Data Types 1-23

1-3-3 Benefit of Using Data Types.... 1-24

1-3-4 International Standard IEC 61131-3.... 1-25

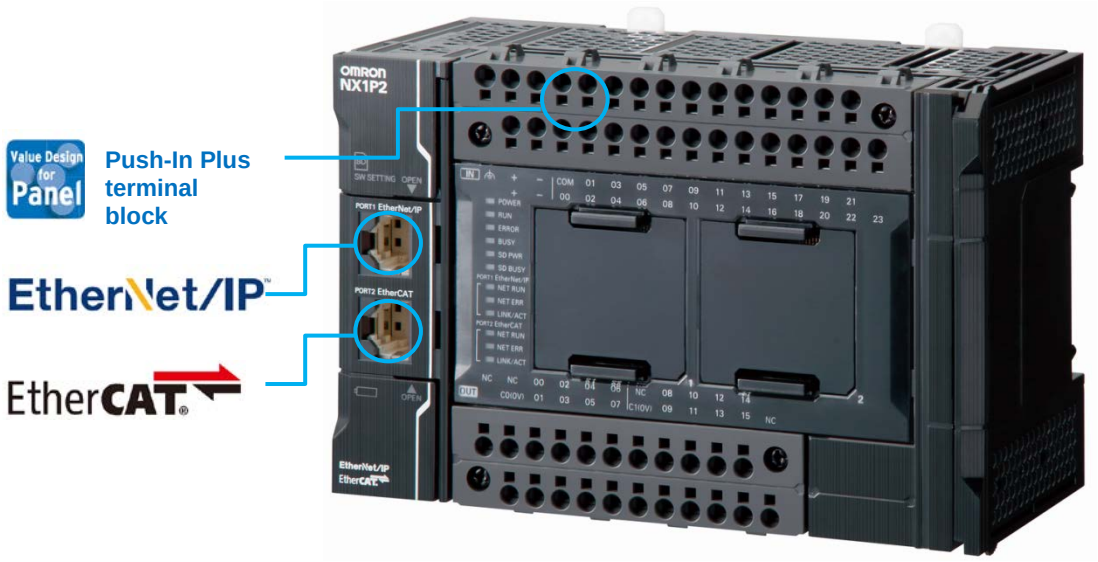
1-4 Programming Software....1-26

1-4-1 Programming Software Sysmac Studio.... 1-26

1-4-2 Simulations 1-27

1-1 Overview

The photo below shows an NX1P2 CPU Unit. Push-In Plus terminal blocks are used to connect a power supply and I/O devices.
EtherCAT and EtherNet/IP ports are built in.



Features

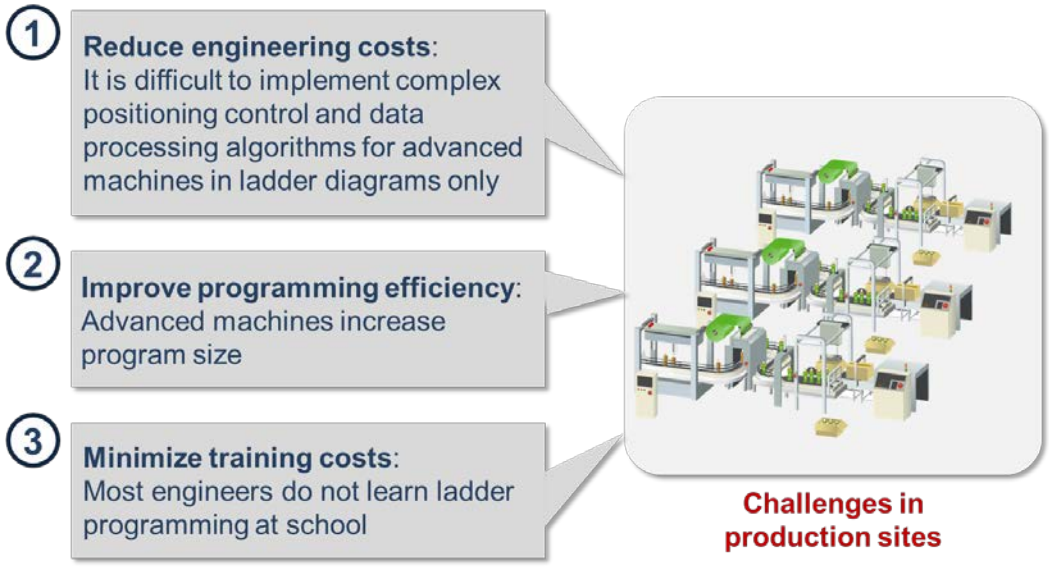
1. The built-in EtherCAT port and advanced motion control make machines faster and more precise
   - Up to four axes of motion control
   - Electronic cams and interpolation increase machine speed and precision
   - EtherCAT simplifies the wiring to up to eight servo systems including for single-axis position control
2. Networks for IoT
   - EtherNet/IP enables communications with a host PC and data links between NJ/NX-series Controllers and CJ-series PLCs
3. Push-In Plus terminal blocks
   - Push-In Plus connection reduces wiring time when a control panel is built

The environment for programming the NX1P makes development faster and easier.
This Guide describes the features of NX1P programming and how to program the NX1P using the Sysmac Studio.

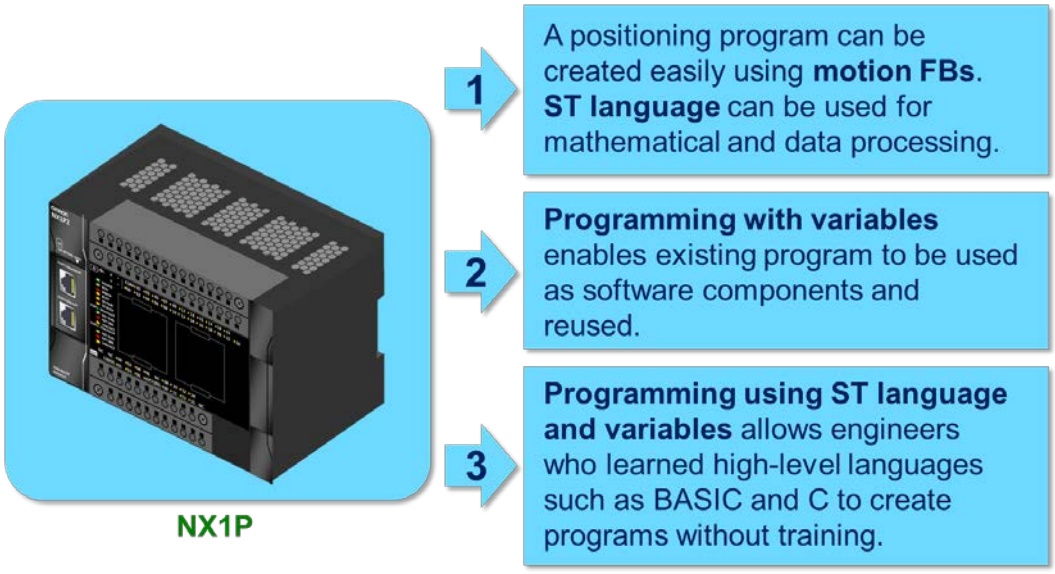
1-2 Features of NX1P Programming

1-2-1 Challenges in Development and Solutions Using the NX1P

As manufacturers need to improve productivity and quality, machines are getting more advanced and more complex. Engineers are facing challenges such as reducing engineering costs, improving programming efficiency, and minimizing training costs.



The NX1P can offer solutions to each challenge.

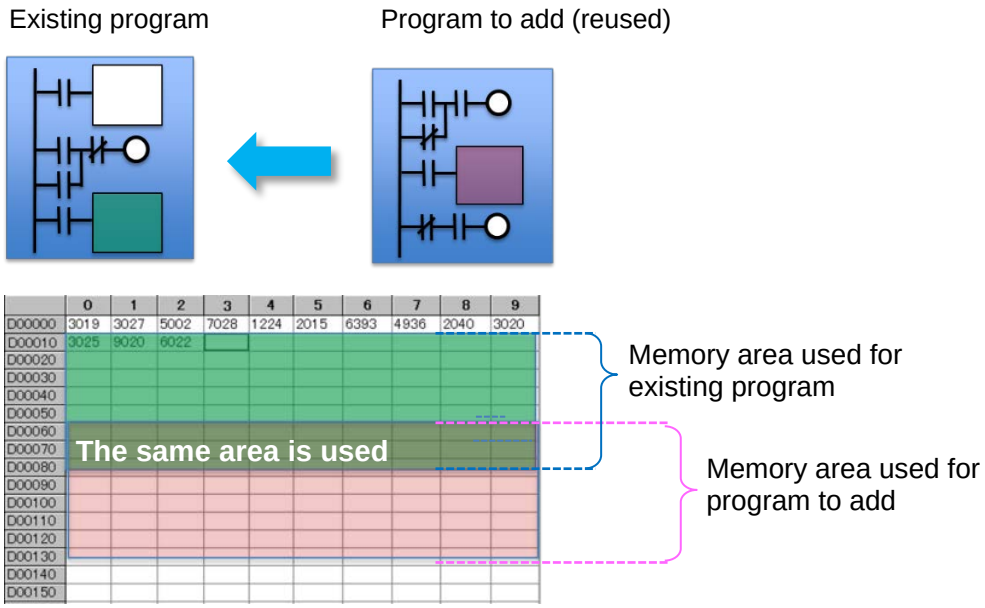


The next section gives more detailed explanation about programming the NX1P.

1-2-2 Easy to Add Programs

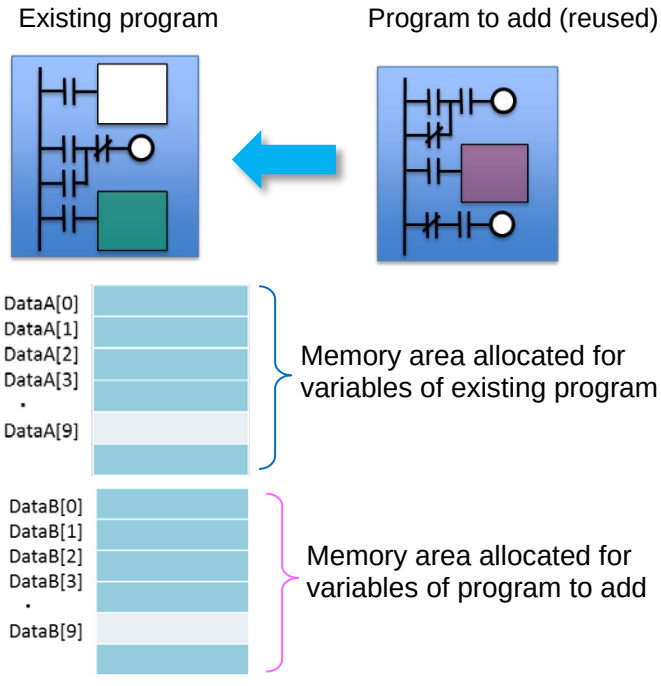
Previously

When adding a program, the user needed to check whether the I/O addresses and memory area used for the additional program had already been used. If they were used, modifications and debugging were required. These tasks reduced development productivity.



Programming the NX1P

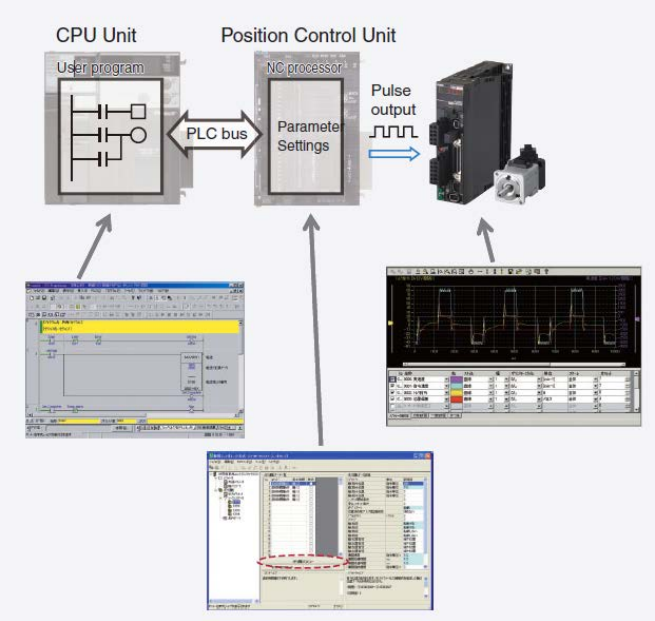
When a program is reused, the NX1P automatically allocates memory addresses in the memory area for variables. The user does not need to worry about addresses when adding or modifying the program. Debugging time can also be reduced.



1-2-3 Easy Motion Programming

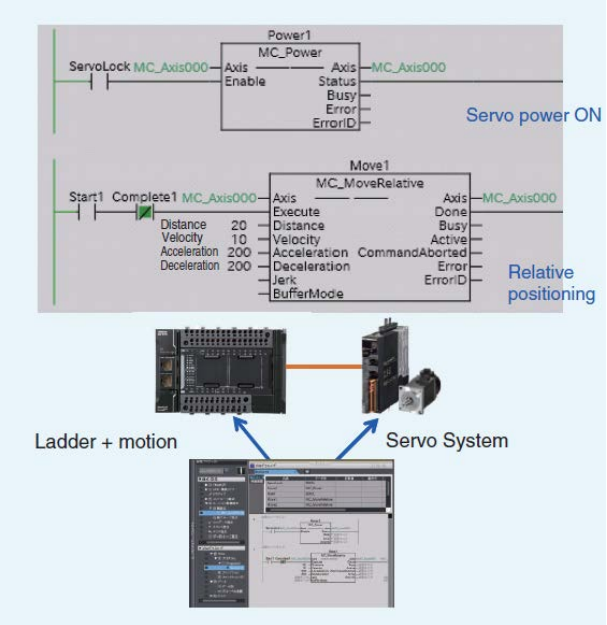
Previously

The traditional PLC (e.g., CJ2) used three different software applications for Position Control Unit settings, ladder programming, and Servo System settings. The user had to create a program while monitoring and tuning the settings.



Programming the NX1P

Operations such as turning ON the Servo, homing, and positioning can be described in one program by using motion FBs.
Processes are executed from top to bottom, which makes the program easy to read.
The Sysmac Studio integrating ladder programming, motion, and Servo configuration facilitates positioning control. Simple monitoring and modification!



1-2-4 Structured Text Language for Easy Mathematical Processing

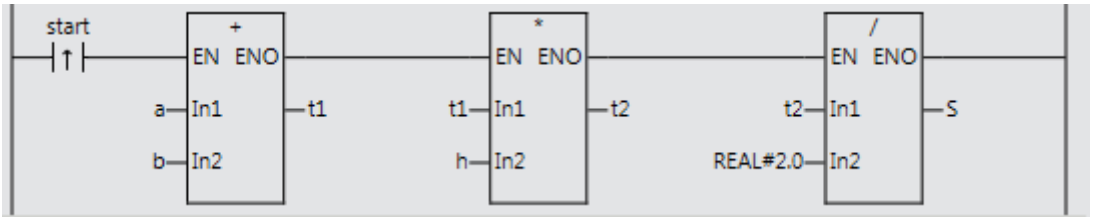
Structured Text Language

The structured text (ST) language is a high-level structured language, similar to Pascal. It is ideal for mathematical processing and nested conditional branching that are difficult to write in ladder diagrams.

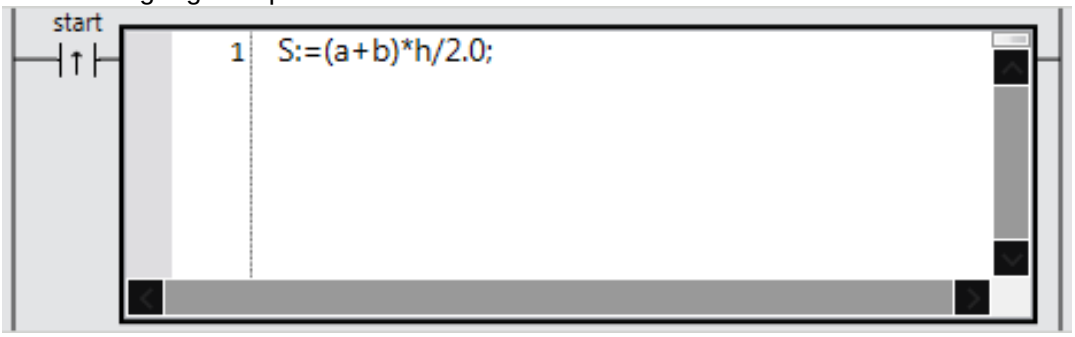
Features of ST Language

You can create easy to read programs by using two different programming languages, ladder diagram language for sequence control and ST language for mathematical processing.

Example: Calculating the area of a trapezoid
(Top length + bottom length) * height / 2



The ST language simplifies this code.



You can use ST as an element in a ladder diagram or create a program in ST only.

ST is ideal for:

1. Arithmetic operations and function calculation
   +, -, *, /, SIN, COS, TAN, etc.
2. Loop and condition constructs
   IF THEN, FOR NEXT, etc.
3. Text string processing
   Joining, extracting, searching, and replacing text strings

The next section gives more detailed explanation about programming with variables.

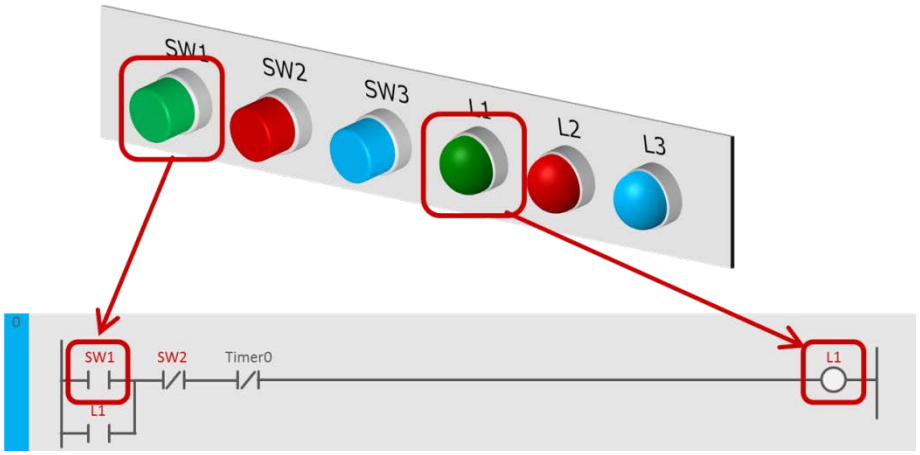
1-3 Programming with Variables

1-3-1 Programming the NX1P

Variables are names defined by the user. They are used for programming the NX1P although addresses are used for the CJ2 and other traditional PLCs.

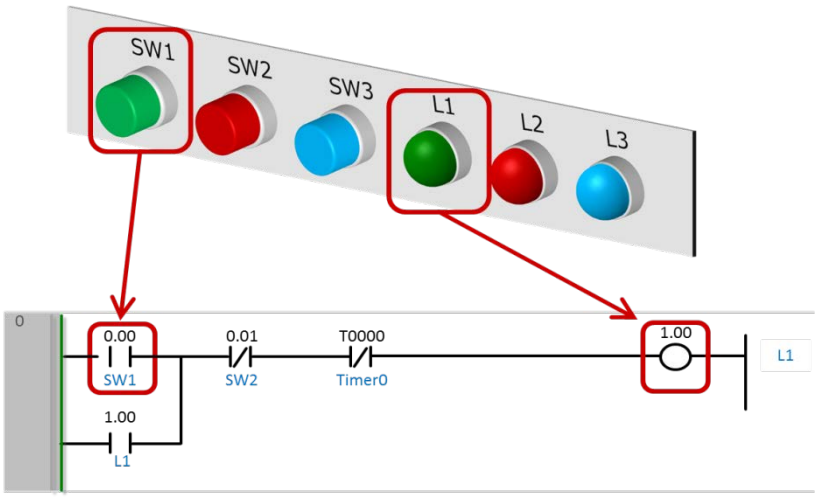
Programming the NX1P

Programming with variables eliminates the need to remember addresses and makes programming faster and easier.
Programming with variables means that you can create programs using the names on your control panel or touchscreen as shown below.



Programming Traditional PLCs

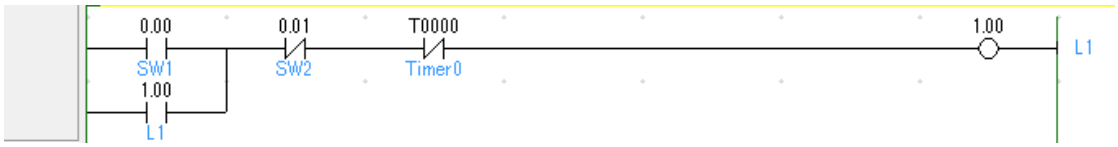
I/O numbers and timer numbers (0.00 and T0000 shown in the figure below) are used to program traditional PLCs such as CJ2.
For most PLCs, comments can be added to the numbers in order to easily understand what the numbers mean. Omron calls the comment “I/O comment”.



Difference between Programming with Addresses and with Variables

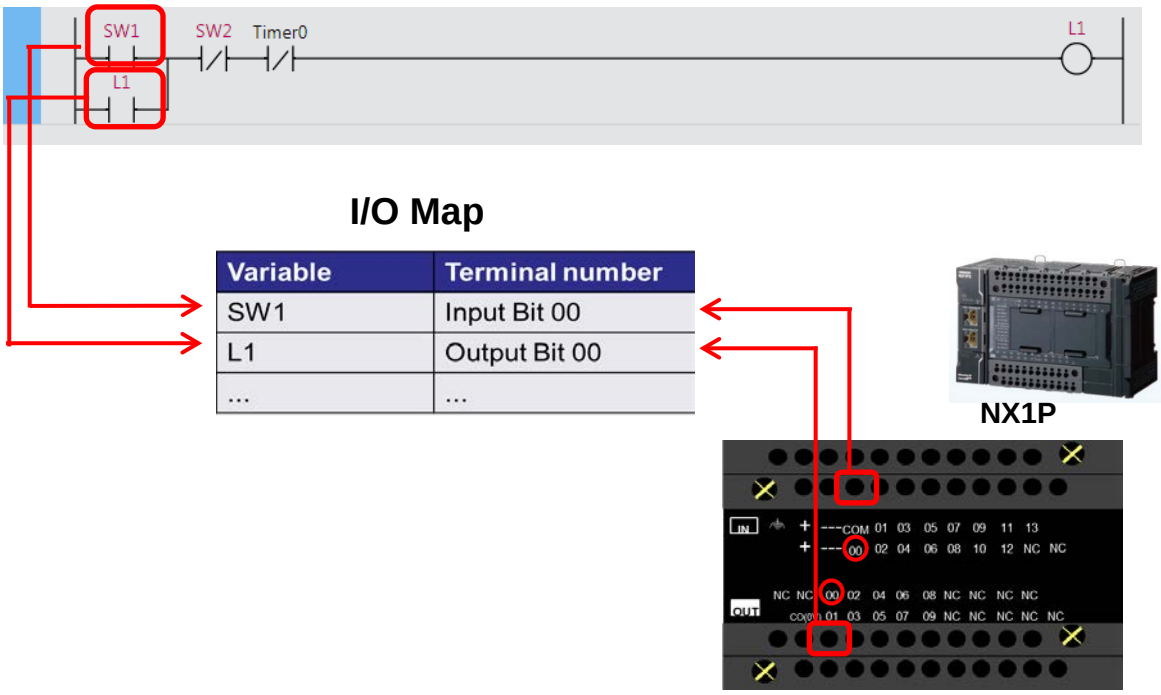
This section shows the difference between two programming methods.
The CJ2 program (created with the CX-Programmer) and NX1P program (created with the Sysmac Studio) are shown below.

CJ2 program example



NX1P program example

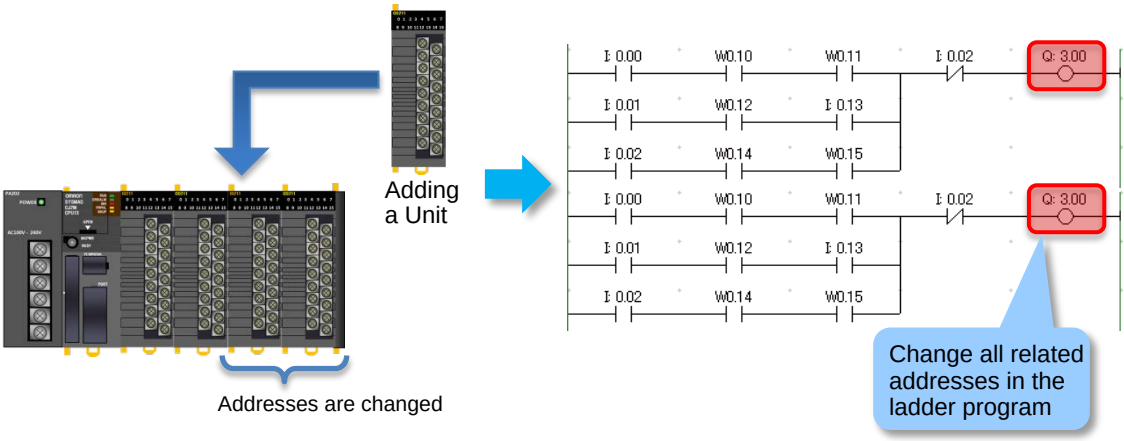
Two programs shown above were written to perform the same operation.
To program the NX1P, each variable (e.g., SW1 and L1) must be assigned in the I/O Map to the corresponding input/output terminal to which the physical device is connected.



Variables used in the program are linked with actual I/O (input/output terminals of the NX1P in this example). You can change I/O assignments by simply changing the terminal number in the I/O Map. The benefit of programming with variables is that there is no need to change the program itself.

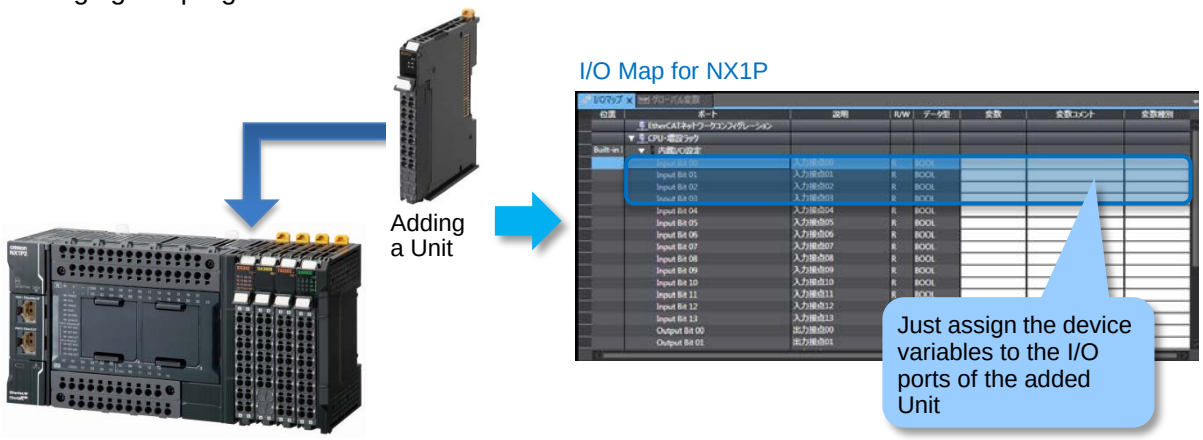
Previously

When changing the Unit configuration, you had to change addresses in the ladder program because the addresses assigned to the I/O channels were changed.



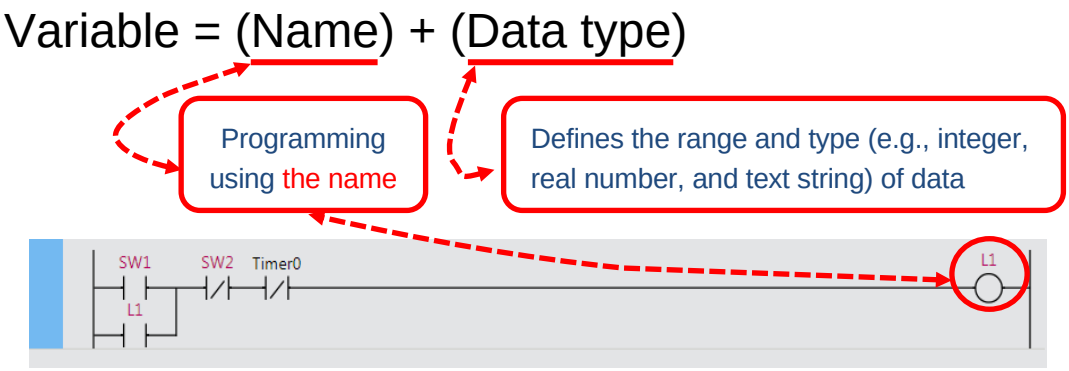
Programming the NX1P

Even when adding Units, you just assign variables to new I/O ports in the I/O Map without changing the program.



1-3-2 Data Types

For example, you define a variable called *L1* (meaning the 1st lamp). It is clear that *L1* contains ON/OFF data because *L1* is a lamp.
However, if you define a variable called *Data1*, *Data1* may contain a decimal number, decimal point number, or text string.
The data type defines the type of data that is expressed by a variable.
A variable is a container for data with a name and data type.



Examples of Data Types

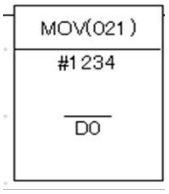
The table below lists the data types used for the NX1P. The BOOL data type is used for ON/OFF data, the INT data type for decimal integers, and the STRING data type for text strings.
Although both the INT and DINT data types represent decimal integers, they have different ranges of values.
The WORD data type for bit strings, the DATE_AND_TIME data type for date and time, and other data types can also be used.

Classification	Used for	Data type		Range of values	Notation
Boolean	ON/OFF status of inputs and outputs	BOOL		0 to 1	·1 or 0 ·TRUE or FALSE
Decimal number (integer)	Numeric operation	Signed	INT	-32768 to +32767	+30000, -20000
			DINT	-2147483648 to +2147483647	+12345678, -20000000
		Unsigned	UINT	0 to 65535	60000
			UDINT	0 to 4294967295	20000000
Floating-point number	Real number	REAL		Single-precision floating-point values	0.15625
		LREAL		Double-precision floating-point values	1.0000000000000002
Text string	Text string displayed on HMI	STRING		Text strings (UTF-8) Approx. 2,000 bytes	‘OMRON’ ‘Failure rate’

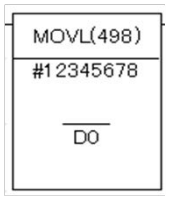
1-3-3 Benefit of Using Data Types

When special instructions are used for a traditional PLC such as CJ2, different instructions must be used for different types of data.
With the NX1P, operands of special instructions are specified with variables. As the variables contain data types, there is no need to use different instructions for different data types.
Even when the data length is changed from 16 bits to 32 bits, all you have to do is change the data type. You don't need to change the program or allocate memory.

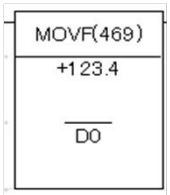
- **CJ2 or traditional PLC**



Move 16-bit value : MOV
(Moved data: 1234)



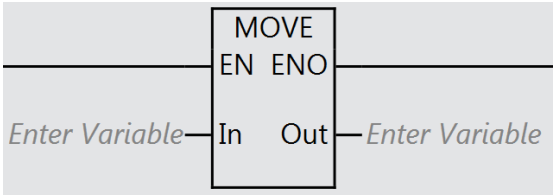
Move 32-bit value : MOVL
(Moved data: 12345678)



Move floating-point value : MOVF
(Moved data: +123.4)

Different special instructions for different types or lengths of data.
MOV, MOVL, MOVF, etc.

- **NX1P**



MOVE instruction only.
Specify the appropriate data types for move source *In* and move destination *Out* when the type or length of data is changed.

Move 16-bit value	**⇒ INT**
Move 32-bit value	**⇒ DINT**
Move floating-point value	**⇒ REAL**

The concepts of "programming with variables" and "data types" based on the international standard IEC 61131-3 are rapidly spreading.

1-3-4 International Standard IEC 61131-3

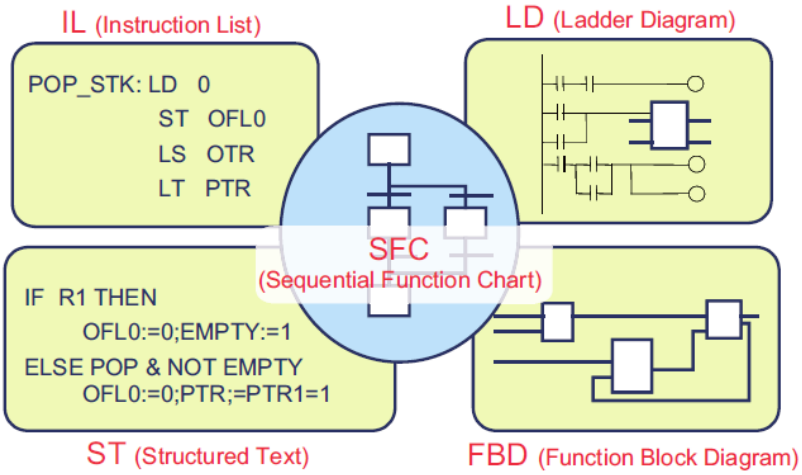
IEC 61131-3

IEC 61131-3[*1] is an international standard that is initially published in 1993.

- Manufacturer and hardware-independent
- Reusable software components
- Five programming languages for a variety of purposes and skill levels

*1. IEC (International Electrotechnical Commission)

Five programming languages according to IEC 61131-3



- IL (Instruction List): A low-level text language similar to assembly
- LD (Ladder Diagram): A graphical language written in a form similar to electrical circuits
- ST (Structured Text): A high-level structured language similar to Pascal
- FBD (Function Block Diagram):
  A graphical language to describe the function as a set of elementary blocks
- SFC (Sequential Function Chart):
  A graphical language used to program processes that can be split into steps

The NX1P supports LD and ST.

Adoption of the IEC 61131-3 standard

The adoption of the IEC 61131-3 standard is widespread from Europe and North America to Asia.
The NX1P support programming languages based on IEC 61131-3. Engineers can be trained easily thanks to familiar programming languages.

1-4 Programming Software

1-4-1 Programming Software Sysmac Studio

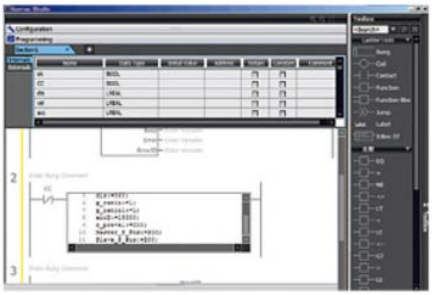
The Sysmac Studio

The Sysmac Studio provides an integrated development environment to configure, program, debug, and maintain NJ/NX-series Machine Automation Controllers.

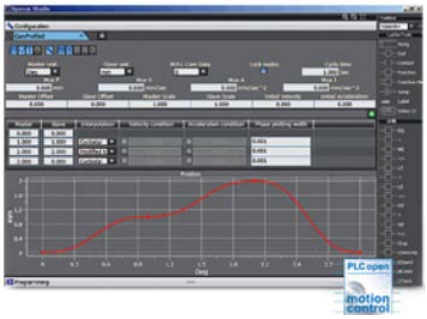
Features

1. One software integrates configuration, programming and monitoring
2. Programming with variables. Supports the ladder and ST languages and FBs[*1] based on IEC 61131-3
3. PLCopen function blocks for easy programming of complex motion profiles, and Cam Editor for quick implementation of cam motion profiles
4. Integrated simulation and debugging
   Motion trajectories in 3D can be pre-tested, and simulation of programs can be performed. This reduces set-up and tuning time.

*1. ST language (Structured Text language), FB (Function Block)

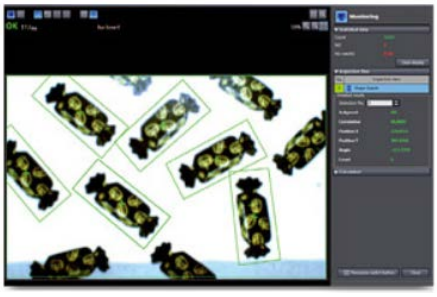


Programming

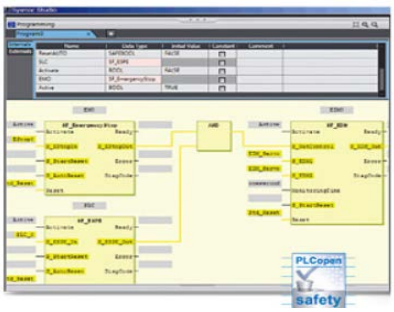


Motion control

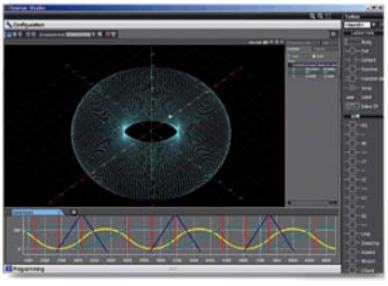
HMI



Vision sensors



Safety

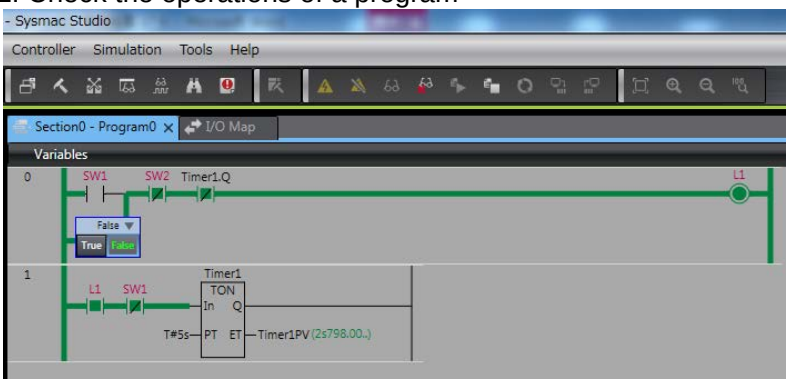


Simulation

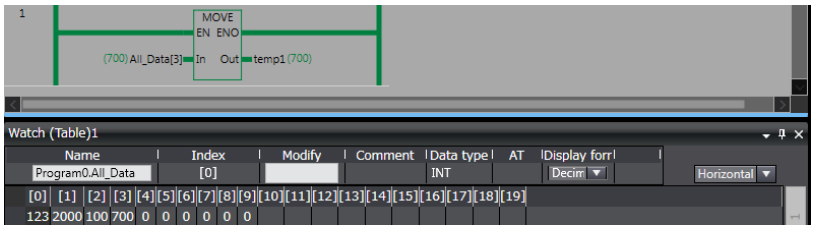
1-4-2 Simulations

The Sysmac Studio provides a variety of simulations.
The Simulator in the Sysmac Studio allows you to test programs without connecting physical devices.

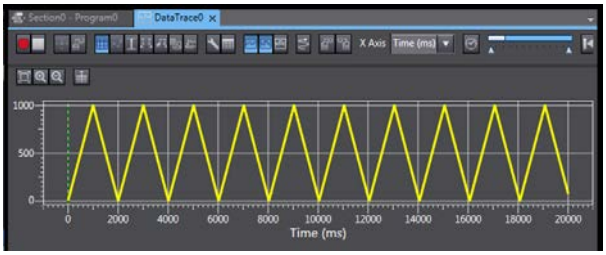
1. Check the operations of a program



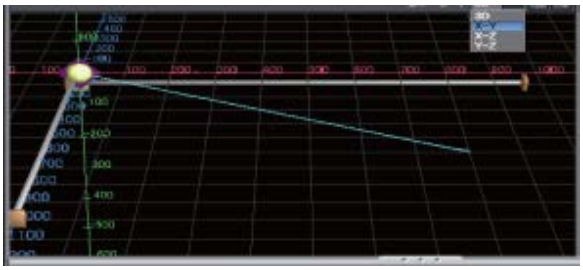
2. Monitor variables in the Watch Tab Page without connecting devices



3. Check a motion program by viewing the changes of positions and velocities sampled by data tracing



4. Check motion trajectories by performing 3D motion monitoring, without connecting physical devices



2

2 Before You Begin

This section describes the process of hardware mounting and wiring and the installation of the Sysmac Studio.

2-1 System Configuration and Devices....2-29
2-1-1 Overview.... 2-29
2-1-2 Wiring.... 2-30
2-2 Installing the Sysmac Studio....2-32
2-2-1 Installing the Sysmac Studio 2-32
2-2-2 Requirements for Installation.... 2-32

2-1 System Configuration and Devices

2-1-1 Overview

Connect a Power Supply, Pushbutton Switches, and Indicators to the NX1P and create a ladder program in *Section 3 Ladder Programming*.

Automation Software Sysmac Studio Standard Edition Version 1.17 or higher	Machine Automation Controller NX1P	Ethernet cable (100Base-TX/10Base-T)
SYSMAC-SE200D (DVD only) SYSMAC-SE201L (1 license)	NX1P2-□□□□	-

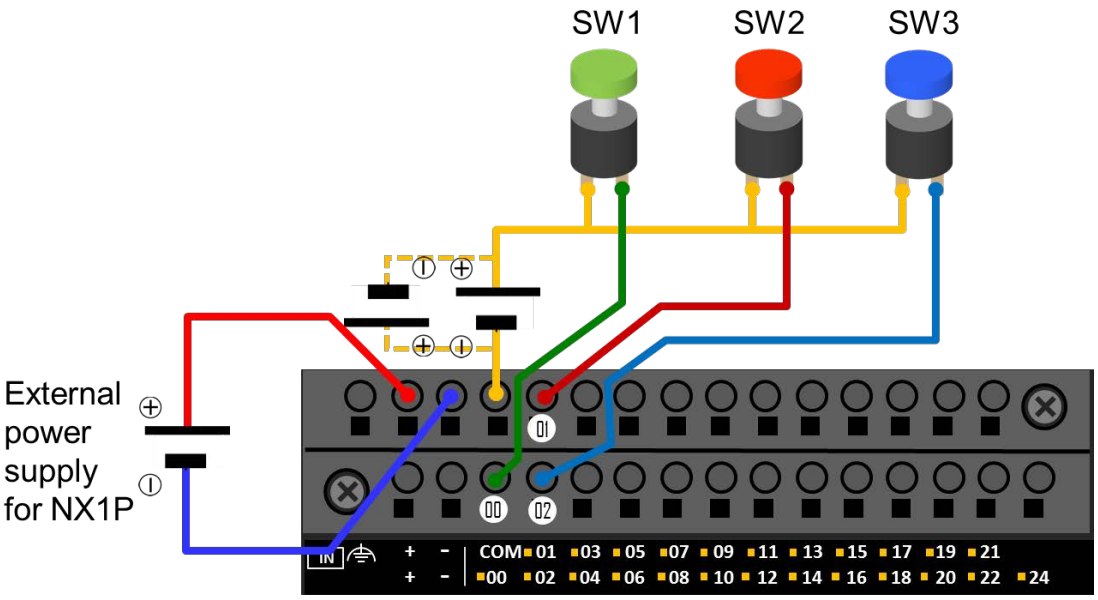
Pushbutton Switch	Indicator	Power Supply
A22NL etc.	M22N etc.	S8VK-S06024 etc.

The physical devices such as NX1P, Pushbutton Switches, and Indicators will help you understand programming concepts.
Even if there is no physical device, you can check operation using the Simulator in the Sysmac Studio.
Section 4, 5, and 6 explain about simulations.

2-1-2 Wiring

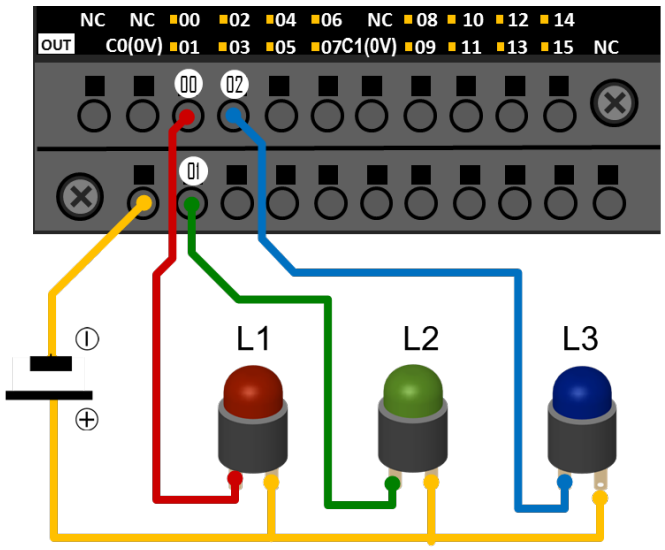
Wiring Pushbutton Switches

Wire Pushbutton Switches to the NX1P as shown below.



The power supply terminals for input devices do not have polarity

Wiring Indicators



The power supply terminals have polarity

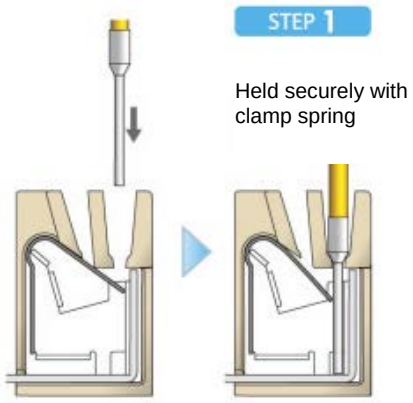
Push-In Plus Terminal Blocks

A push-in terminal block allows you to connect wires (e.g. ferrule) by just pushing them in. Reducing wiring work can greatly reduce the time required to build control panels.
Push-In Plus Terminal Blocks were independently developed by Omron for easier wire insertion and firmer wire holding ability than standard push-in terminal blocks.

You can connect and remove a wire (solid or ferrule) to/from a Push-In Plus Terminal Block by following the procedure below. (Refer to the manual for connection and removal of a stranded wire)

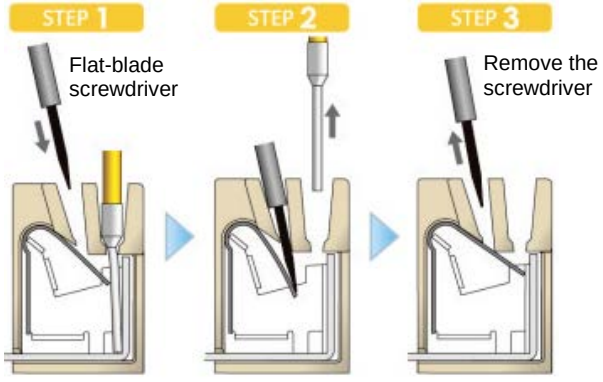
■ Connecting a wire

Just push the wire into the terminal block until stopping. When connecting a stranded wire, use a ferrule, or insert the wire after loosening the clamp spring with a tool and then remove the tool.



■ Removing a wire

Press a flat-blade screwdriver diagonally into the release hole to loosen the clamp spring and then remove the wire. Remove the flat-blade screwdriver.



2-2 Installing the Sysmac Studio

2-2-1 Installing the Sysmac Studio

The Sysmac Studio is the support software to configure, program, debug, and simulate NJ/NX-series Controllers.

Use the following procedure to install the Sysmac Studio on your computer.

1. Set the Sysmac Studio installation disk into the DVD-ROM drive.
   The setup program is started automatically and the Select Setup Language Dialog Box is displayed.
2. Select the language to use, and then click the **OK** Button.
   The Sysmac Studio Setup Wizard is started.
3. Follow the wizard to install the Sysmac Studio.
4. Restart your computer after the Sysmac Studio has been successfully installed.

2-2-2 Requirements for Installation

The system requirements for the Sysmac Studio are given in the following table.

OS	CPU		RAM	Display
Windows 7 32-bit or 64-bit edition Windows 8 32-bit or 64-bit edition Windows 8.1 32-bit or 64-bit edition Windows 10 32-bit or 64-bit edition	Minimum	IBM AT or compatible with Intel® Celeron® processor 540 (1.8 GHz)	2 GB	XGA 1,024 x 768 16 million colors
	Recommended	IBM AT or compatible with Intel® Core™ i5 M520 processor (2.4 GHz) or the equivalent	4 GB min.	WXGA 1,280 × 800 16 million colors

Precautions for Correct Use

When the CX-One version 4 or lower has been installed, uninstall it before installing the Sysmac Studio.

3

3 Ladder Programming

This section describes how to write ladder programs using the Sysmac Studio.

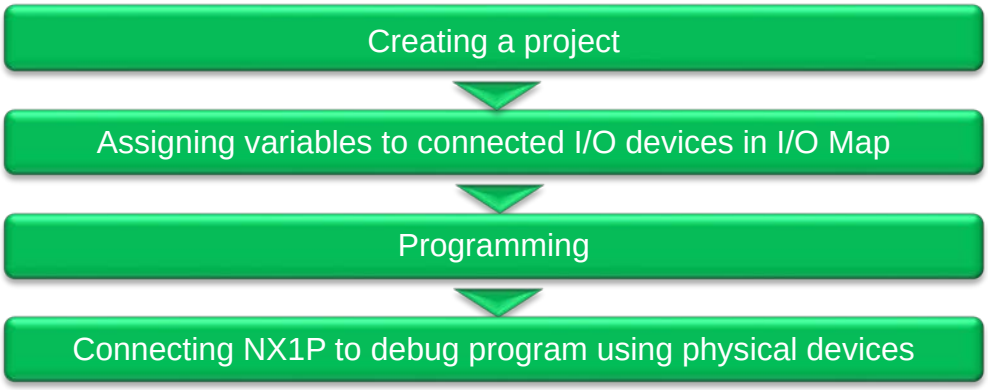
3-1 Programming with the Sysmac Studio 3-35
3-1-1 Programming Procedure 3-35
3-1-2 Creating a Project 3-35
3-2 Parts of the Sysmac Studio Window 3-37
3-2-1 Screen for Configurations and Setup 3-37
3-2-2 Screen for Programming 3-37
3-3 Assigning Variables to Terminals 3-38
3-3-1 Variable Names for Terminal Numbers 3-38
3-3-2 I/O Map Setting 3-39
3-3-3 Checking Wiring 3-41
3-4 Ladder Programming 3-42
3-4-1 Inserting Circuit Parts 3-42
3-4-2 Keyboard Mapping 3-42
3-4-3 Rules 3-43
3-5 Example of a Basic Ladder Program 3-44
3-5-1 Practice of Programming a Ladder Diagram 3-44
3-5-2 Writing the Algorithm 3-45
3-5-3 Program Check 3-47
3-5-4 Saving the Program 3-48
3-5-5 Checking Operation on the NX1P 3-49
3-5-6 Checking Operation on the Simulator 3-50
3-5-7 Example of a Program Error (Offline) 3-52
3-5-8 Example of an Error Occurred During Operation 3-52
3-6 Example of a Ladder Program Using a Timer Instruction 3-53
3-6-1 Self-holding Rung 3-53

3-6-2 On-Delay Timer (TON) Instruction 3-54
3-6-3 Exercise: Energy Saving Escalator 3-58
3-6-4 Checking the Operation of the Program 3-59
3-6-5 Checking the Operation of the Program (Watch Tab Page) 3-60
3-7 Example of a Ladder Program Using Date and Time 3-62
3-7-1 Programming the NX1P Using Date and Time 3-62
3-7-2 Exercise: Continuous Operating Time of Escalator 3-62
3-8 Fundamentals of Programming to Reduce Development Time 3-66
3-8-1 POUs (Program Organization Units) 3-66
3-8-2 Programs and Execution Priorities (Tasks) 3-66
3-8-3 Functions (FUNs) and Function Blocks (FBs) 3-68
3-8-4 Sections 3-69
3-8-5 Types of Variables 3-70

3-1 Programming with the Sysmac Studio

3-1-1 Programming Procedure

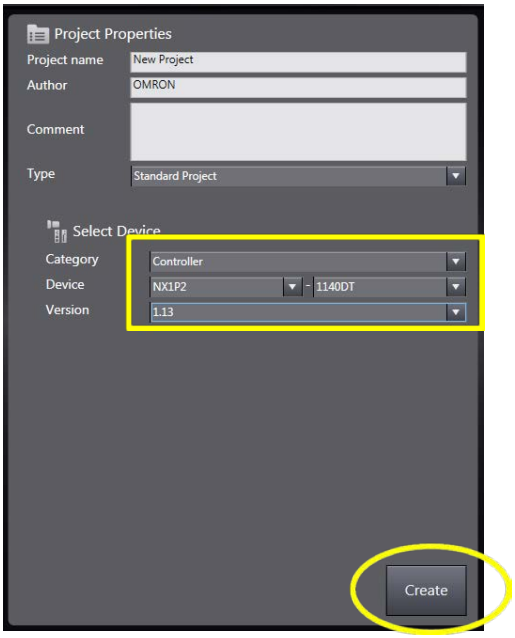
This section describes how to create a simple ladder program using pushbutton switches and lamps.
The overall programming procedure is given below.



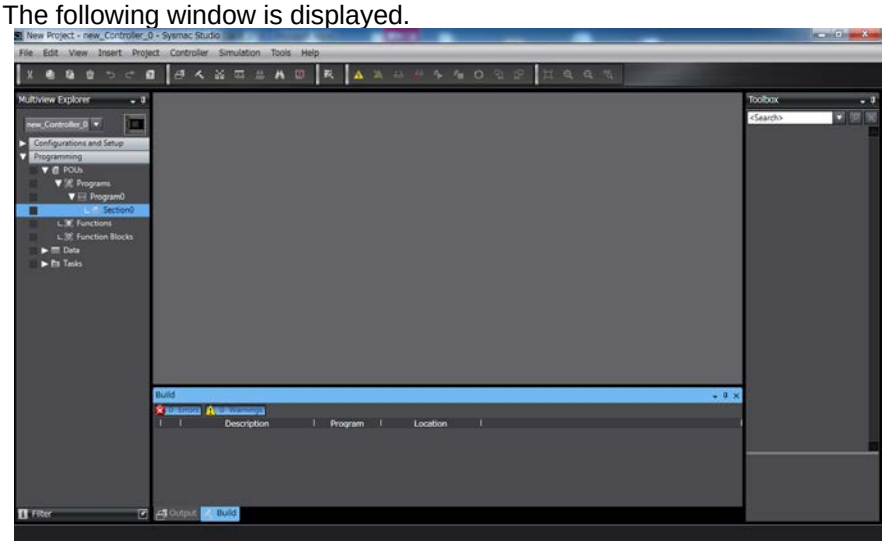
3-1-2 Creating a Project

1. Start the Sysmac Studio.

2. Enter the project name.
Select *NX1P2, 9024DT/1140DT* for the *device* parameter and *1.13* (version indicated on the NX1P) for the *version* parameter, and then click the **Create** Button.

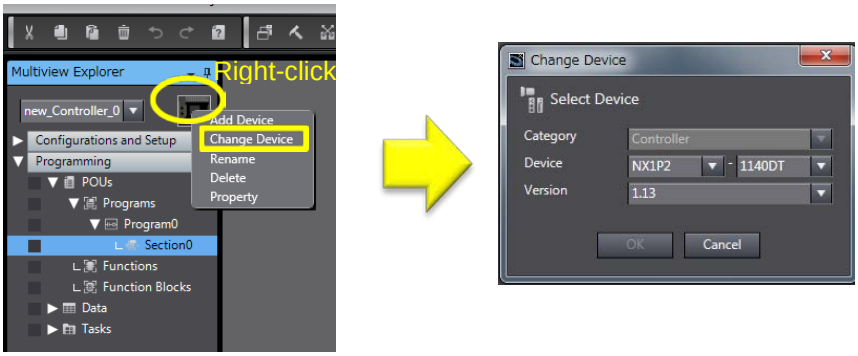


The following window is displayed.



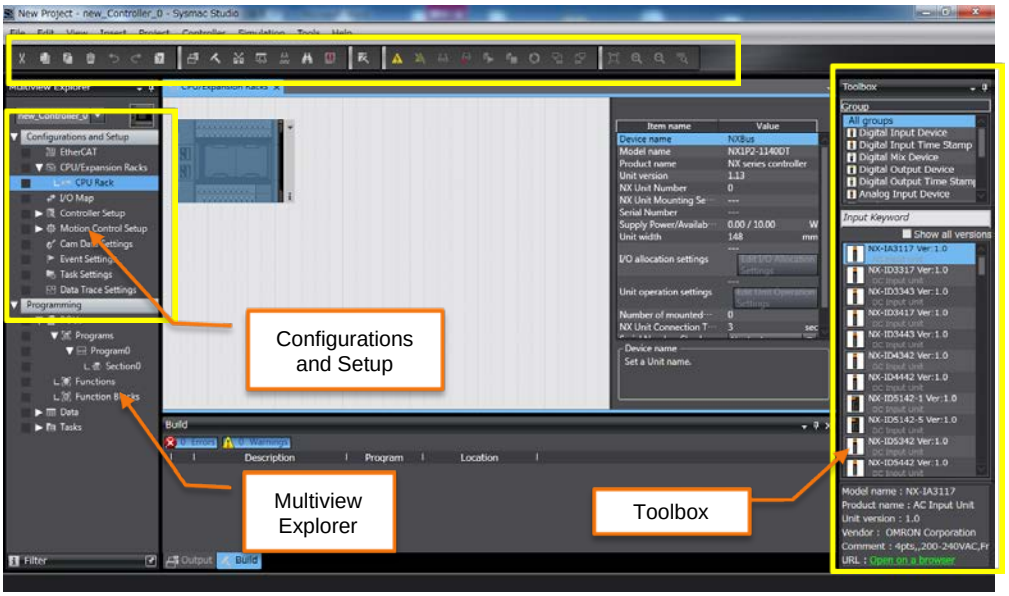
Additional Information

You can change the model, version, and other properties after creating a project file.

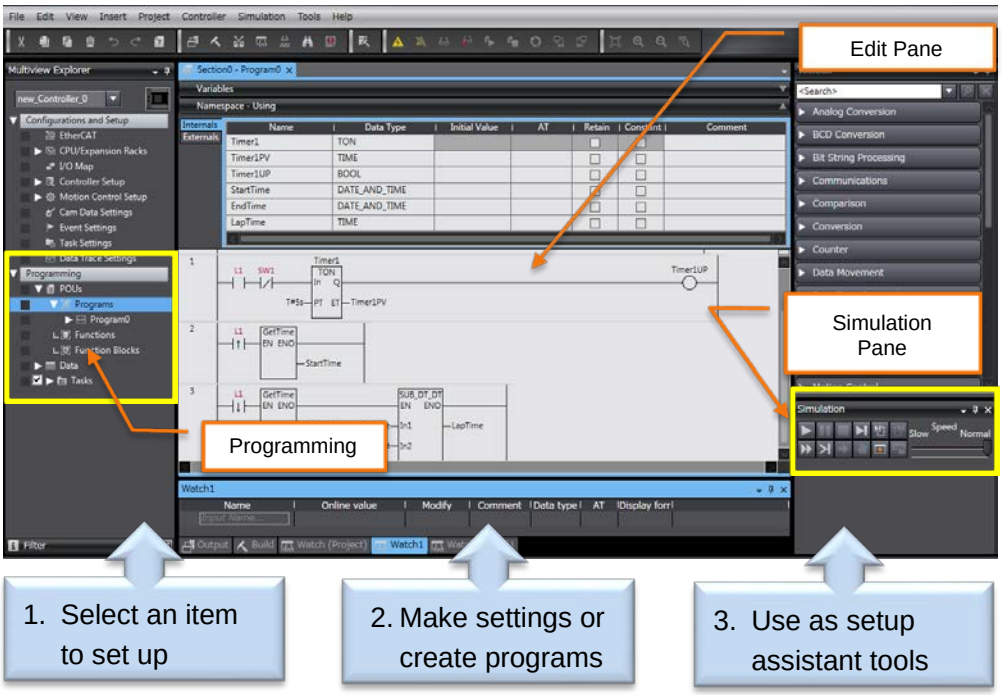


3-2 Parts of the Sysmac Studio Window

3-2-1 Screen for Configurations and Setup



3-2-2 Screen for Programming

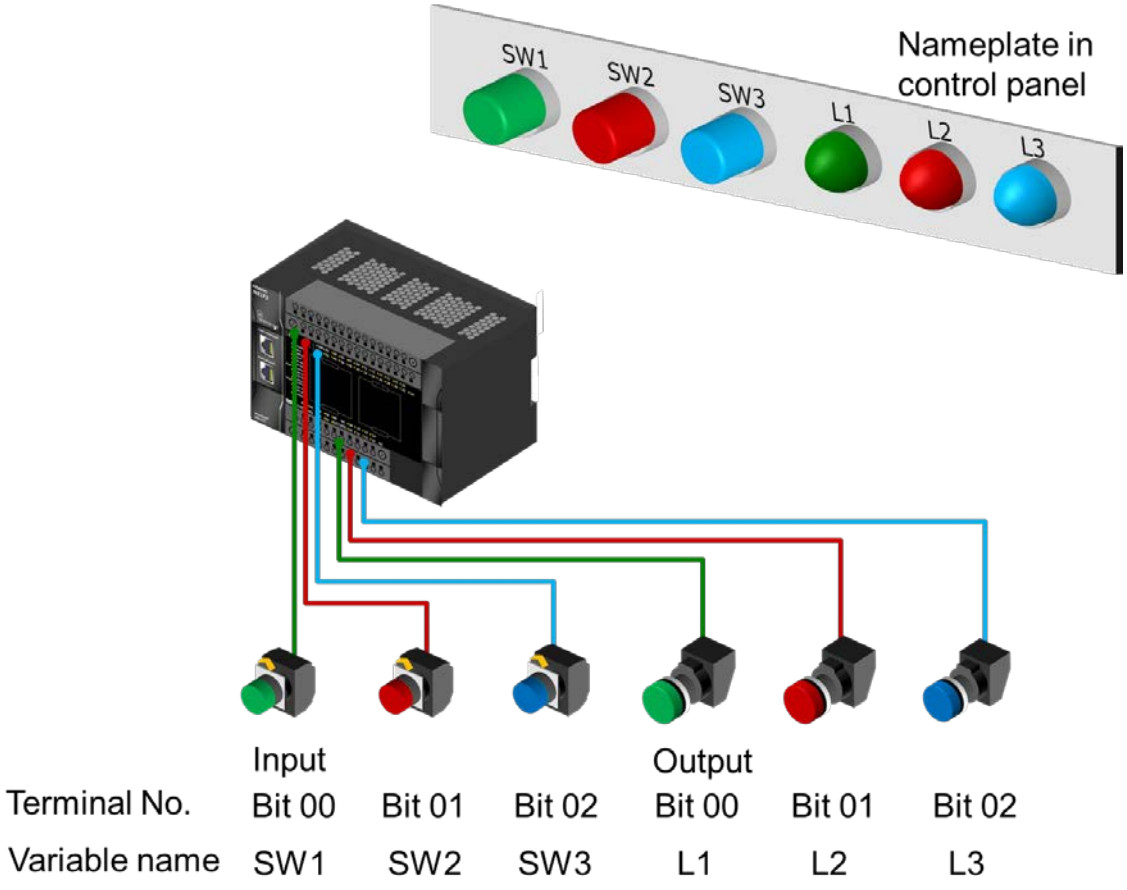


3-3 Assigning Variables to Terminals

3-3-1 Variable Names for Terminal Numbers

Although Pushbutton Switches and Indicators are physically connected to the input and output terminals of the NX1P, they cannot be used for programming now.
In order to create a program using the connected devices (I/O), you need to assign variable names for the numbers of terminals to which devices are connected.

Any name can be assigned. Names related to the device type or processing are recommended. For example, you can use “SW1” for a Pushbutton Switch connected to the input terminal 00 of the NX1P because “SW1” is its name on the nameplate in the control panel. This makes it easy to identify the device.

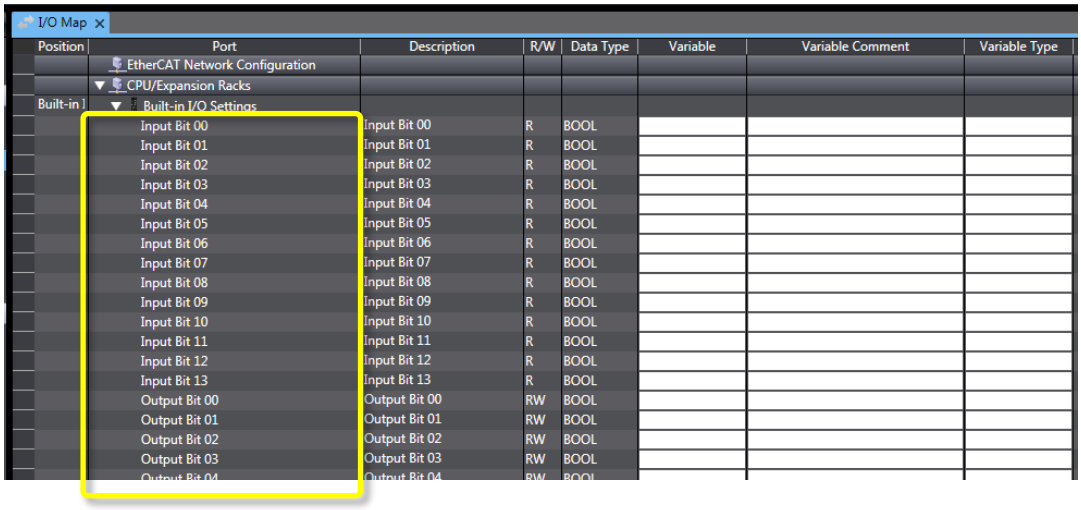


3-3-2 I/O Map Setting

Set variables for terminals.

I/O map setting means that variables used for a program are assigned to terminals (called “I/O ports” in the Sysmac Studio) of the NX1P to which devices (I/O) are connected.

1. Double-click **I/O Map** under **Configurations and Setup** on the Multiview Explorer. The I/O Map is displayed.



As the NX1P is selected for the device, all input/output terminals (I/O ports: Input Bit 00 etc.) of the NX1P are displayed in the I/O Map.

2. Double-click an I/O port to enter a variable name.

Set variable names for input/output terminals as shown below.

Variable names for input terminals

Terminal No.	R/W	Variable name	Data type
Input bit 00	R	SW1*1	BOOL
Input bit 01	R	SW2	BOOL
Input bit 02	R	SW3	BOOL

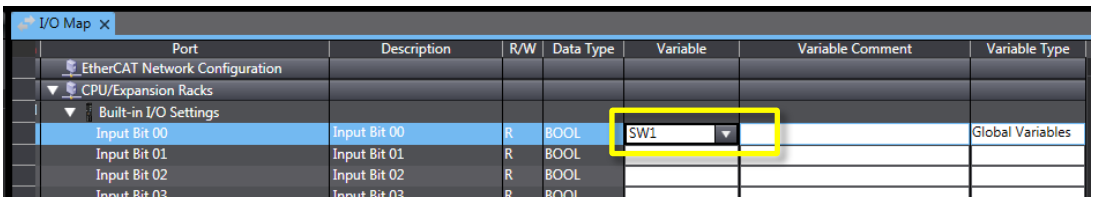
Variable names for output terminals

Terminal No.	R/W	Variable name	Data type
Output bit 00	R	L1*2	BOOL
Output bit 01	R	L2	BOOL
Output bit 02	R	L3	BOOL

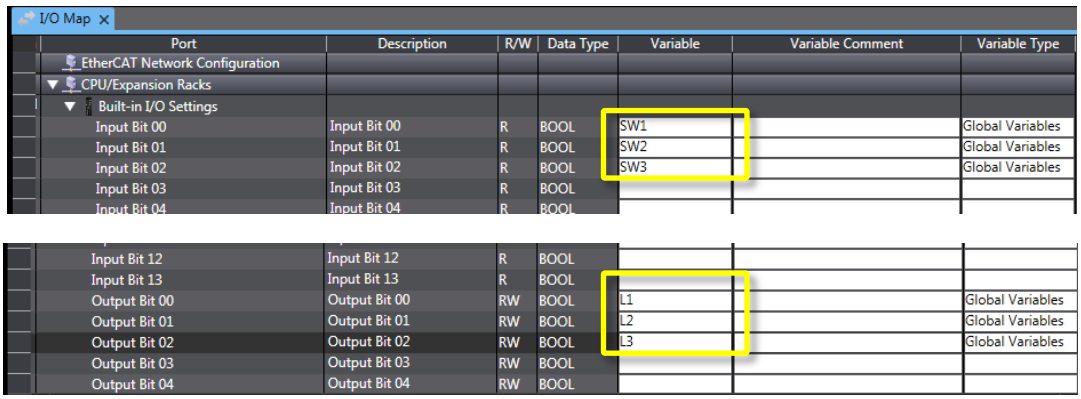
*1. SW means a switch (Pushbutton Switch).

*2. L means a lamp (Indicator).

3. Enter “SW1” in the *Variable* Column of Input Bit 00.



4. In the same way, set SW2 and SW3 for Input Bit 01 and 02 and L1 to L3 for Output Bit 00 to 02.



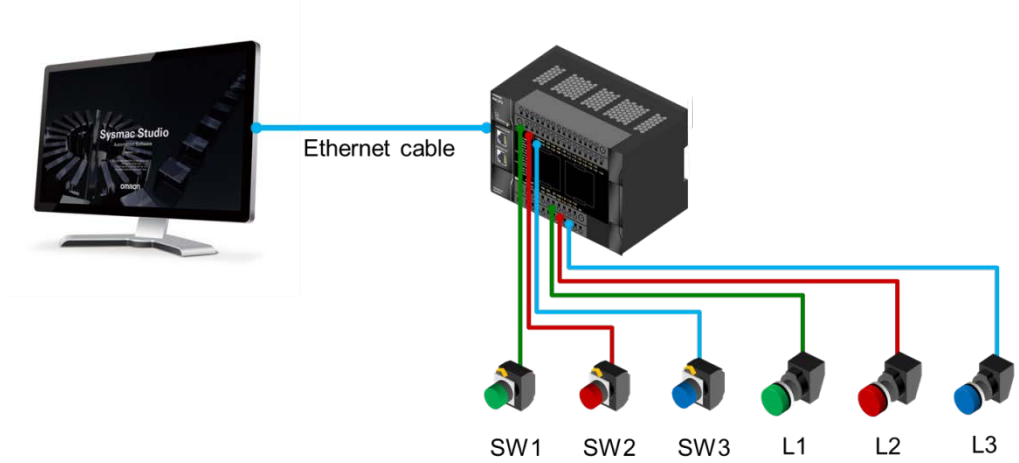
I/O Map

Port	Description	R/W	Data Type	Variable	Variable Comment	Variable Type
EtherCAT Network Configuration						
CPU/Expansion Racks						
Built-in I/O Settings						
Input Bit 00	Input Bit 00	R	BOOL	SW1		Global Variables
Input Bit 01	Input Bit 01	R	BOOL	SW2		Global Variables
Input Bit 02	Input Bit 02	R	BOOL	SW3		Global Variables
Input Bit 03	Input Bit 03	R	BOOL			
Input Bit 04	Input Bit 04	R	BOOL			
Input Bit 12	Input Bit 12	R	BOOL			
Input Bit 13	Input Bit 13	R	BOOL			
Output Bit 00	Output Bit 00	RW	BOOL	L1		Global Variables
Output Bit 01	Output Bit 01	RW	BOOL	L2		Global Variables
Output Bit 02	Output Bit 02	RW	BOOL	L3		Global Variables
Output Bit 03	Output Bit 03	RW	BOOL			
Output Bit 04	Output Bit 04	RW	BOOL			

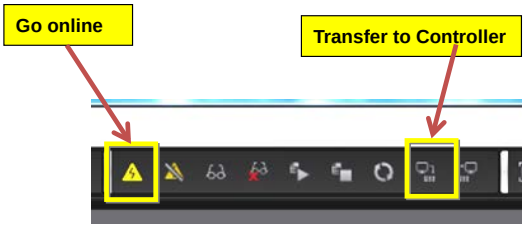
The variable names have been linked with the terminal numbers.

3-3-3 Checking Wiring

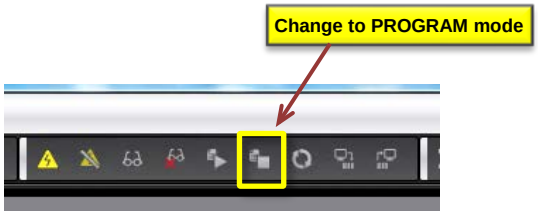
1. Connect the NX1P to the computer (Sysmac Studio) via an Ethernet cable.



2. Go online, and then transfer the I/O map settings to the NX1P.



3. Change the operating mode to PROGRAM mode to prevent the program from being executed while checking I/O wiring.



4. Press the Pushbutton Switches to check whether the values of the input bits change in the I/O Map.

5. Select the I/O port. Right-click and select ***Set*** or ***Reset*** from the menu to check whether the Indicator turns ON or OFF.

3-4 Ladder Programming

3-4-1 Inserting Circuit Parts

Inserting a Program Input or Output in an AND Structure

Shortcut key: Select a connecting line and press the shortcut key
Example: N.O. Input - **C** Key, output - **O** Key

Toolbox: Drag a circuit part from the Toolbox

Right-click: Right-click a connecting line and select ***Insert Input*** or ***Insert Output*** from the Menu.

Inserting a Program Input in an OR Structure

Shortcut key: Select an input and press the **W** Key. (The N.O. input is inserted in an OR structure)

Drag and drop: Drag the connecting line and select a circuit part from the pop-up menu

Inserting a Rung

Shortcut key: Select the start of a rung and press the **R** Key.

Right-click: Right-click a rung and select ***Insert rung above*** or ***Insert rung below***.

3-4-2 Keyboard Mapping

The following table lists the shortcut keys that you can use when creating ladder programs.

Operation	Shortcut keys	Reference (Shortcut key in CX-Programmer)
Entering an N.O. input	**C**	Same key * **C** or **L** in CX-Programmer
Entering an N.C. input	**/**	Same key
Entering an OR with an N.O. input	**W**	Same key * Different cursor position
Entering an OR with an N.C. input	**X**	Same key * Different cursor position
Entering an output	**O**	Same key
Entering a NOT output	**Q**	Same key
Calling a function block	**F**	Same key * **F** for both FUN and FB
Calling a function	**I**	Different key * First letter of instruction
Inserting a rung below the cursor	**R**	Same key
Inserting a rung above the cursor	**Shift** + **R**	

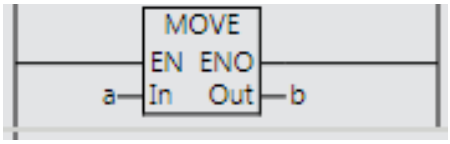
* Select ***Keyboard Mapping Reference*** from the Help Menu to display the Keyboard Mapping Reference.

3-4-3 Rules

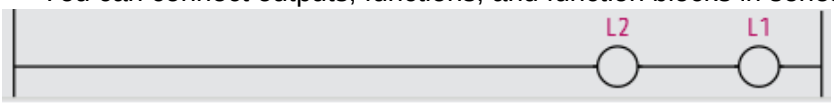
You Can

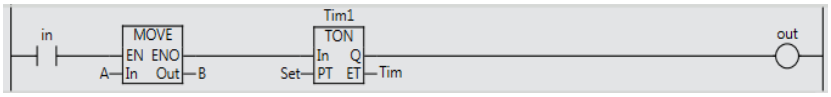
With the NX1P, you can program a ladder diagram that cannot be programmed with the CJ2 or other traditional PLC.

- You can insert an output without inserting an input. (Always ON Flag is not required) Functions and function blocks can also be inserted without inserting an input.



- You can connect outputs, functions, and function blocks in series.

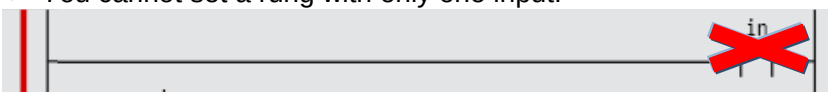




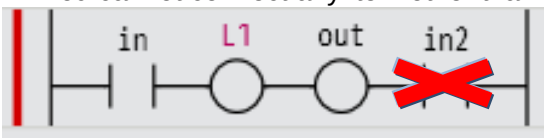
You Cannot

- You cannot set a rung without any circuit parts.

- You cannot set a rung with only one input.



- You cannot connect any item other than output after an output.

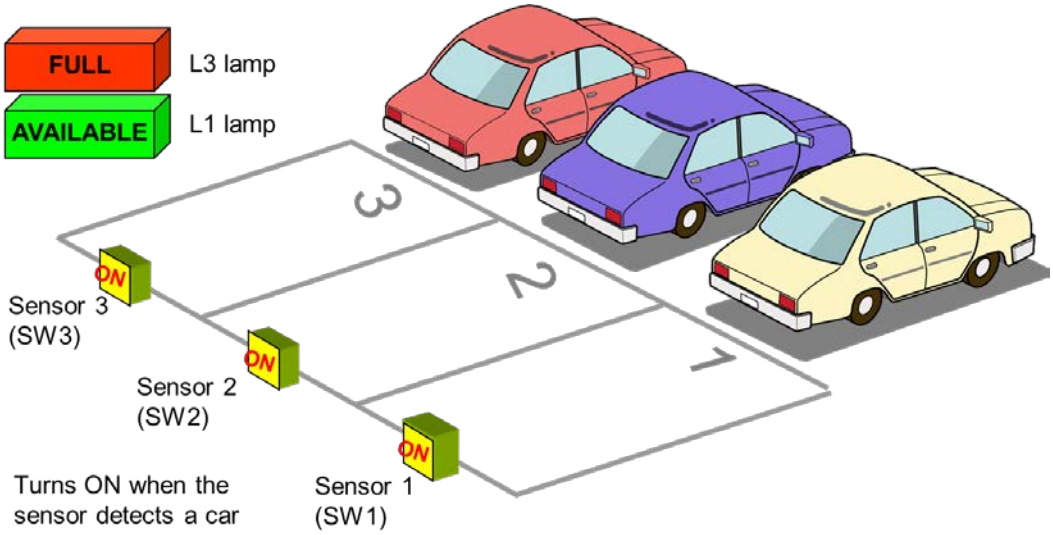


3-5 Example of a Basic Ladder Program

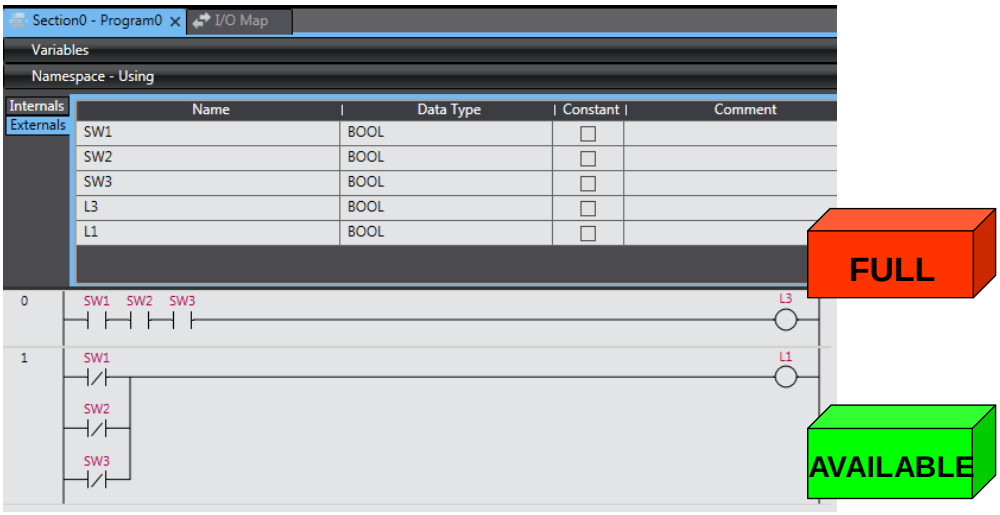
3-5-1 Practice of Programming a Ladder Diagram

[Exercise] Coin Operated Parking Space

Three cars can be parked in the parking space. Create a program to turn ON the FULL lamp (L3 (red lamp)) if the parking lots are full and turn ON the AVAILABLE lamp (L1 (green lamp)) if one or more parking lots are available.
SW1, SW2, and SW3 are used as sensors to detect presence of cars.

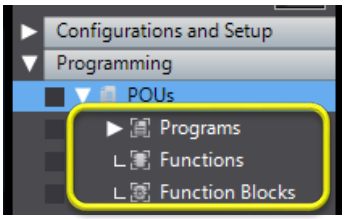


●Completed program

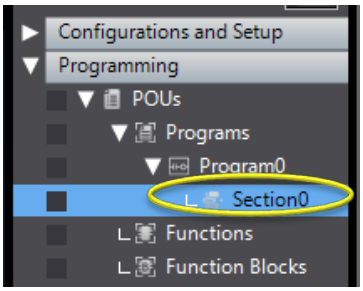


3-5-2 Writing the Algorithm

1. Click **POUs** under **Programming** in the Multiview Explorer.
Programs, *Functions*, and *Function Blocks* are displayed under *POUs*.

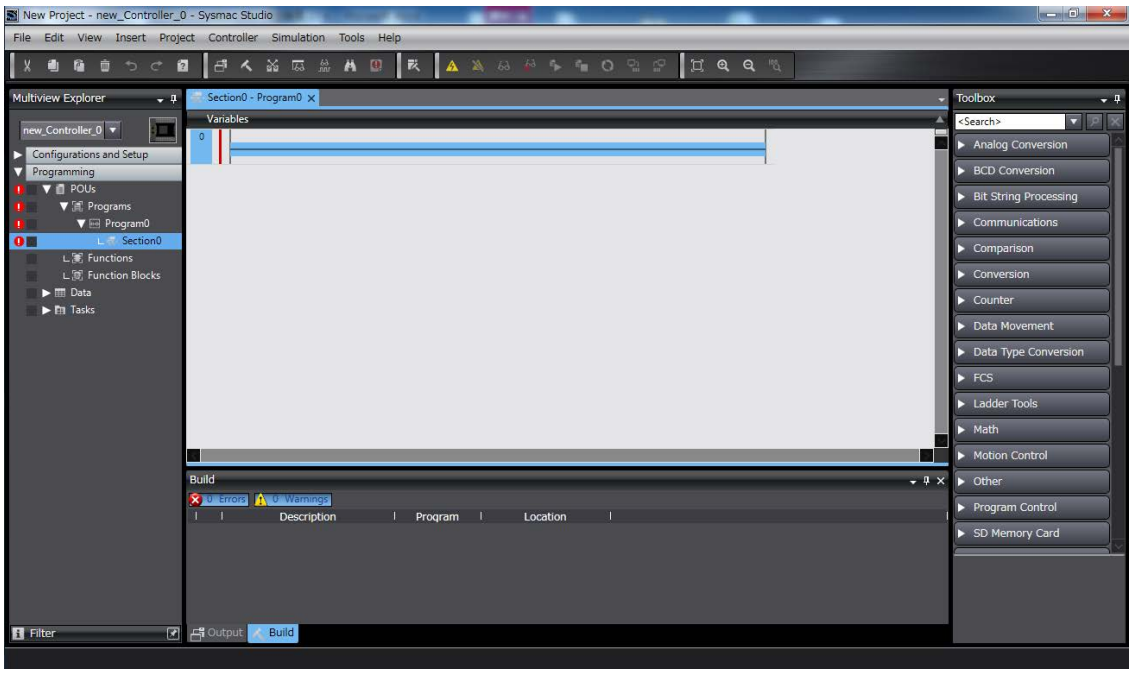


2. Double-click **Section0** under **Programs - Program0**.



The Ladder Editor is displayed.

3. Write a ladder program on the Ladder Editor.



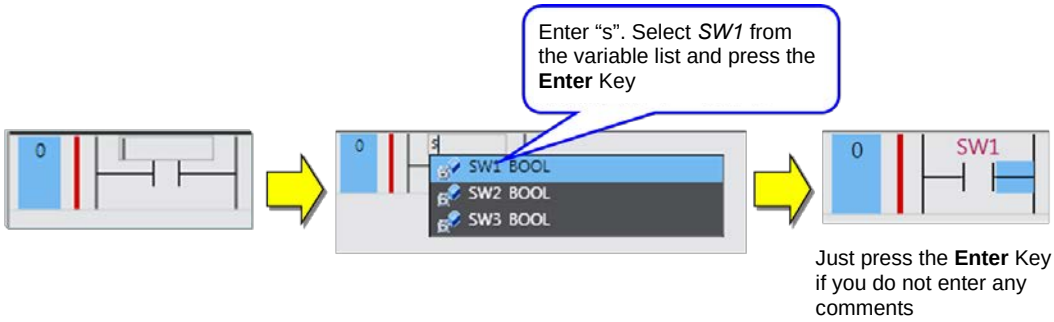
Additional Information

When a new ladder program is created, Section0 will be marked with a red ! mark ( ). This mark means that the program contains an error. It will disappear when the program is written correctly.

(1) Insert a program input in an AND structure

Insert an input and enter the variable name.

1. Press the **C** Key or right-click a connecting line and select ***Insert Input*** from the Menu.

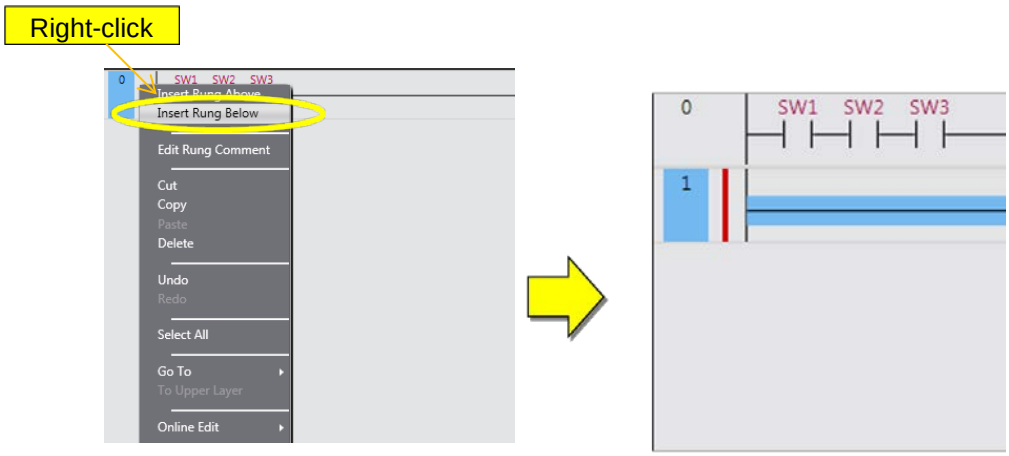


In the same way insert circuit parts as shown below.

2. To insert an output, press the **O** Key or right-click a connecting line and select ***Insert Output*** from the Menu.

(2) Insert a rung below.

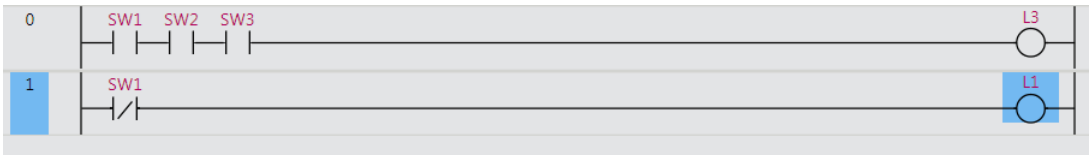
1. Select the start of a rung and press the **R** Key, or right-click a rung and select ***Insert rung below***.



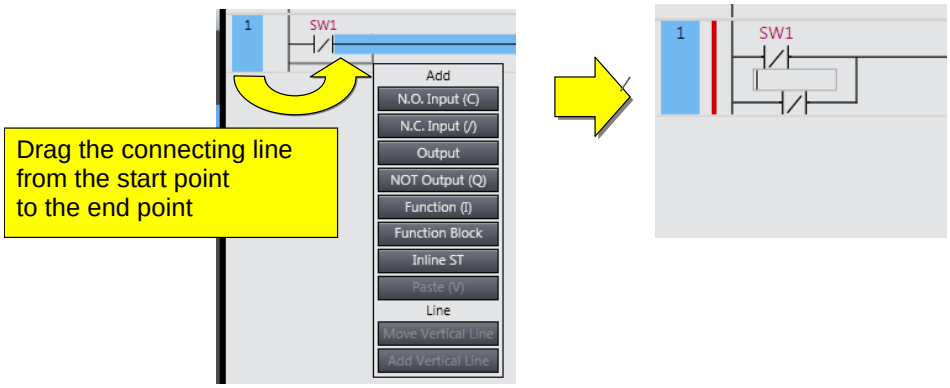
(3) Insert a program input in an OR structure

1. Insert an N.C. input and an output.

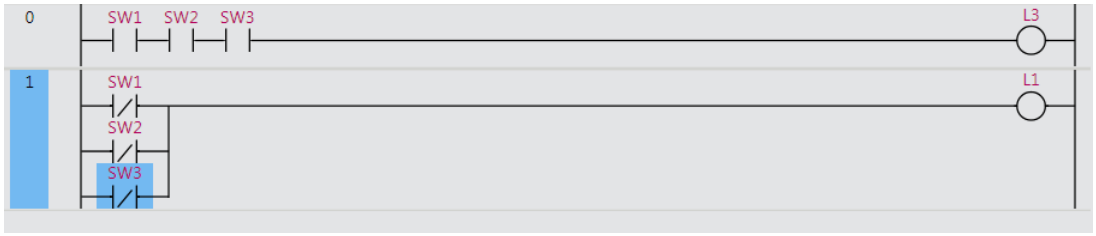
Insert an N.C. input by pressing the ***/*** Key and then insert an output.



2. Inset an N.C. input in an OR structure. Select *SW1* and press the **X** Key, or drag the connecting line from the start point to the end point and select ***N.C. Input*** from the menu.



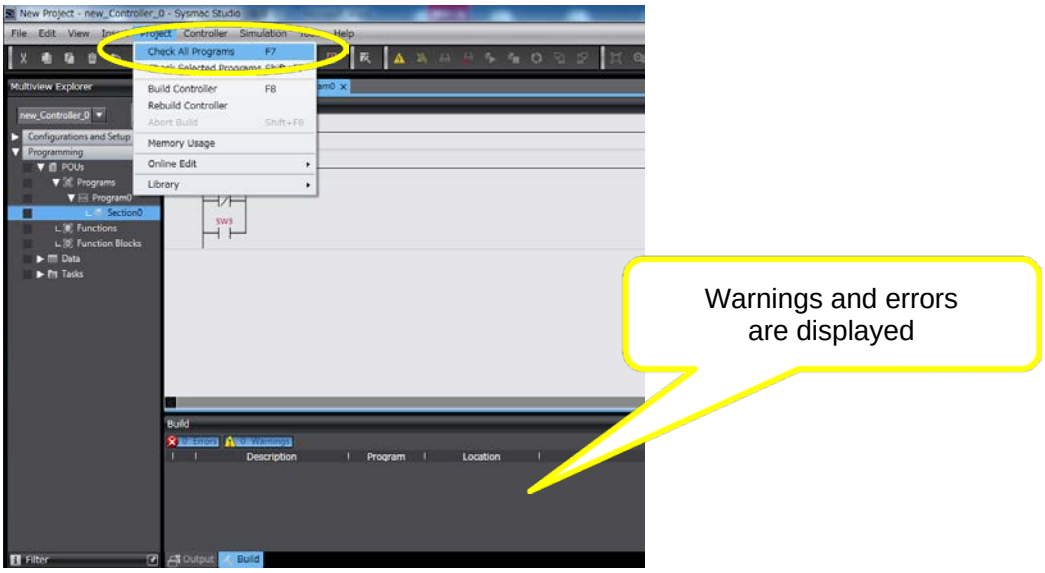
3. Insert another N.C. Input in the same way.



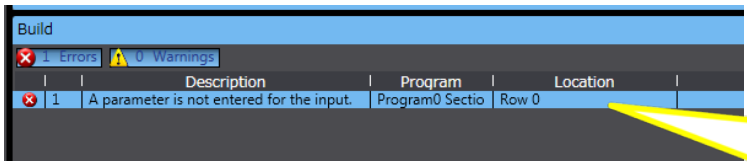
The program is completed.

3-5-3 Program Check

1. Select ***Check All Programs*** from the Project Menu.



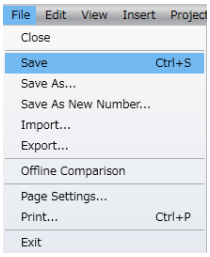
2. Modify the program if an error is found.
The results of the program check are displayed in the Build Tab Page.
Modify the program if an error is found.



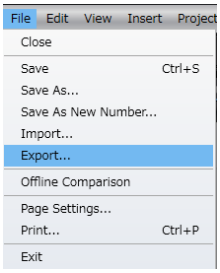
< Useful function >
Double-click the line of the error to jump to the location of the error

3-5-4 Saving the Program

1. Save and export the project file before taking the next step.
Select ***Save*** from the File Menu.



2. Select ***Export*** from the File Menu. Enter a file name and export the file to the desktop.



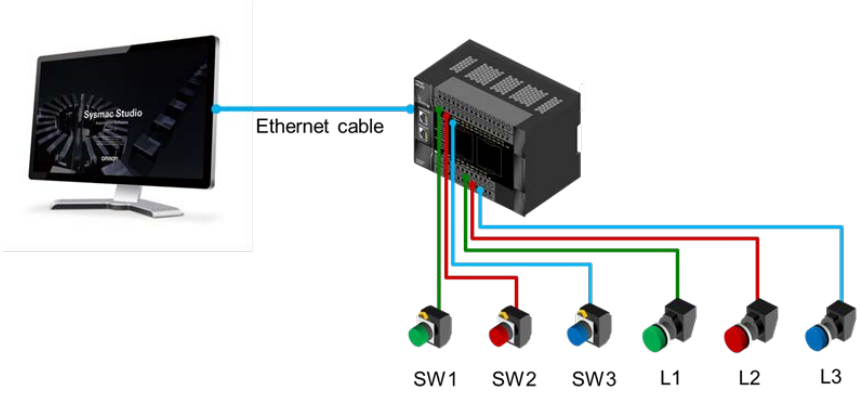
Additional Information Difference between Save and Export

Save	Export
The project is saved in the default folder. The user does not need to know where the project is saved.	The project is Saved as a file.
To open the saved project, click **Open Project** on the Start page.	To open the saved project, double-click the file icon or Click **Import** on the Start page.

3-5-5 Checking Operation on the NX1P

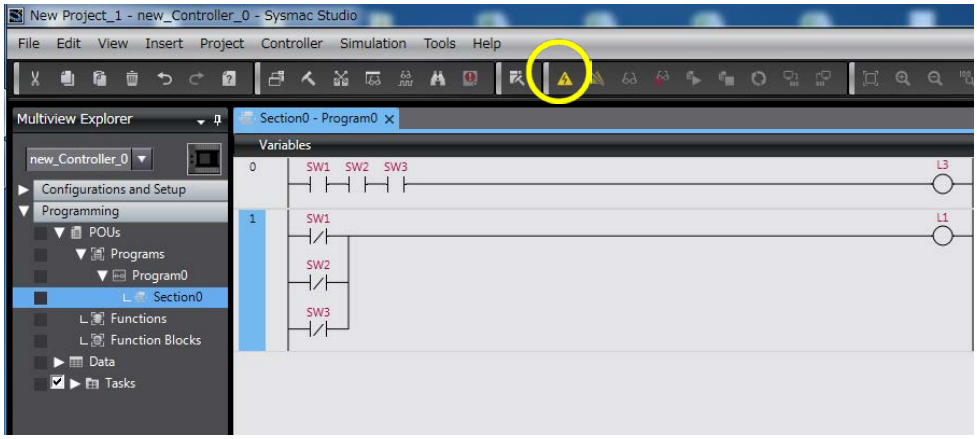
Connect the NX1P and the Sysmac Studio and download the project (all data including the program) from the Sysmac Studio to the NX1P to check operation.

1. Connect the NX1P to the computer (Sysmac Studio) via an Ethernet cable.



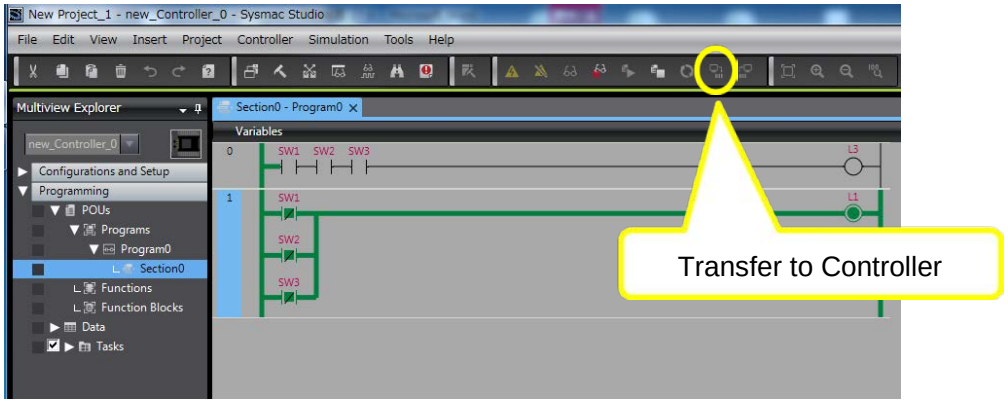
2. Go online.

Start the Sysmac Studio and go online with the NX1P.



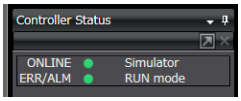
3. Download the project.

Click the **Transfer to Controller** Button.



4. Change the operating mode to RUN mode.

Check the Controller Status Pane. If the mode is PROGRAM mode, change the operating mode to RUN mode.



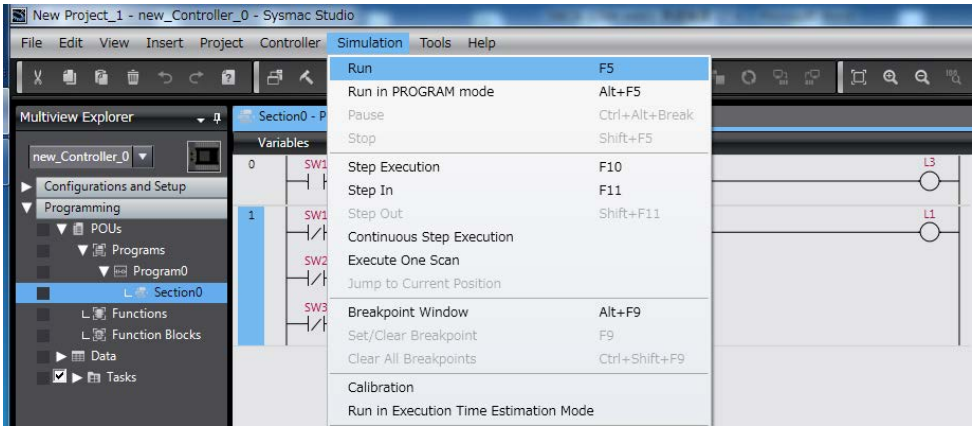
5. Check operation.

Turn ON and OFF the SW1, SW2, and SW3 to check whether the AVAILABLE lamp (L1) and FULL lamp (L3) turn ON and OFF.

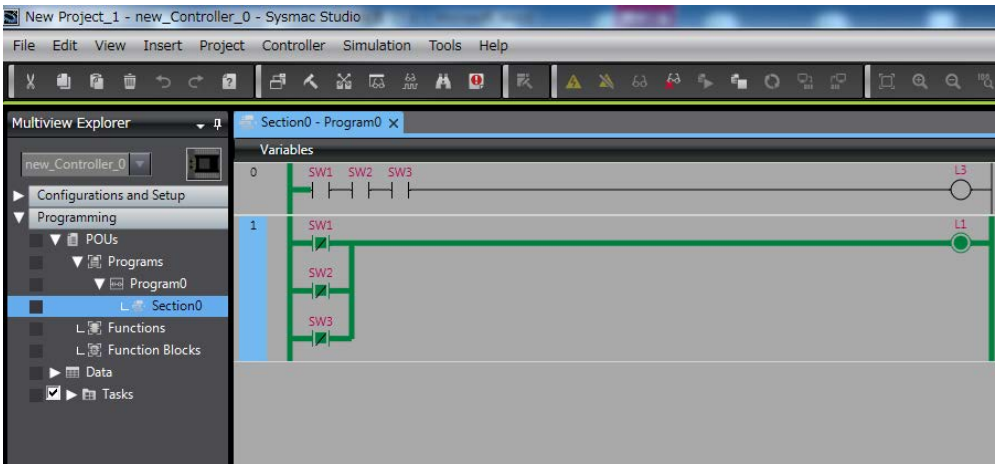
3-5-6 Checking Operation on the Simulator

You can check operation using the Simulator in the Sysmac Studio, without connecting the NX1P (offline debugging).

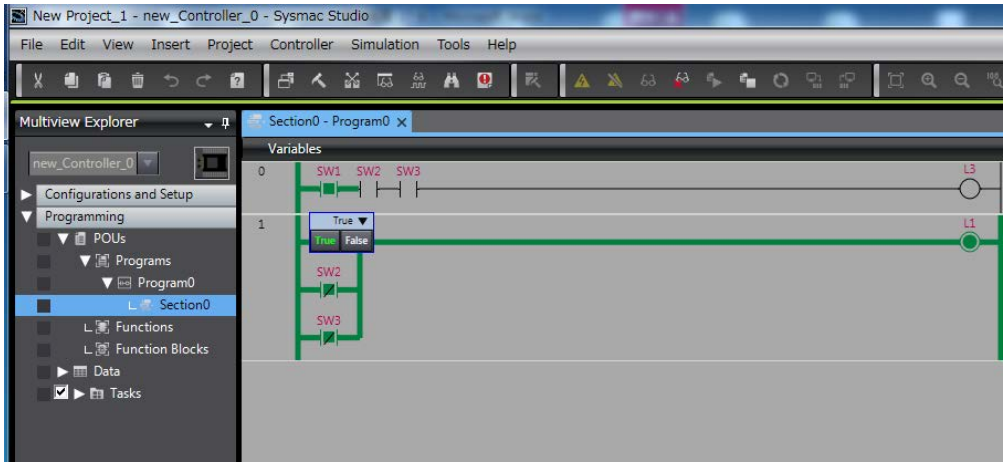
1. Select ***Run*** from the Simulation Menu to start the Simulator.



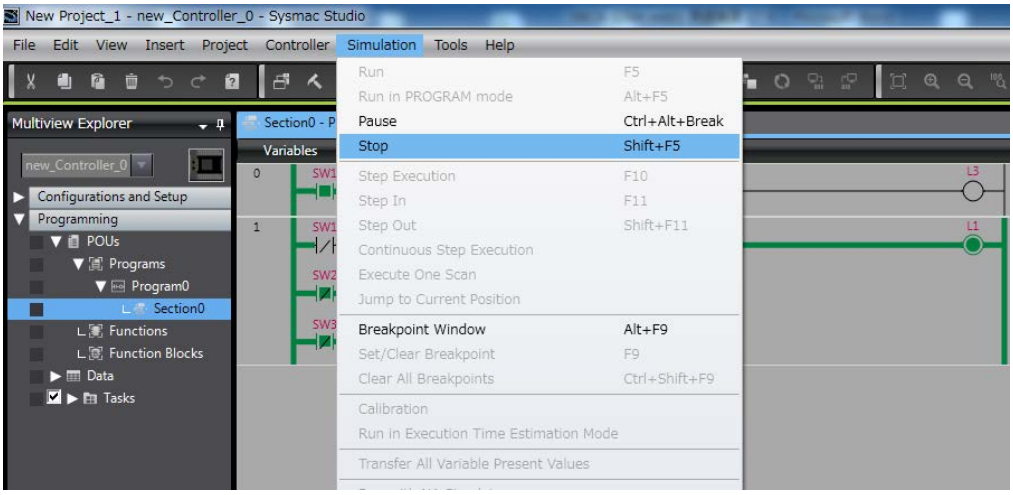
2. The Simulator is started and connected after displaying some messages.



3. Double-click the input. You can change the value between True (ON) and False (OFF) to debug the program, instead of pressing the physical switch.

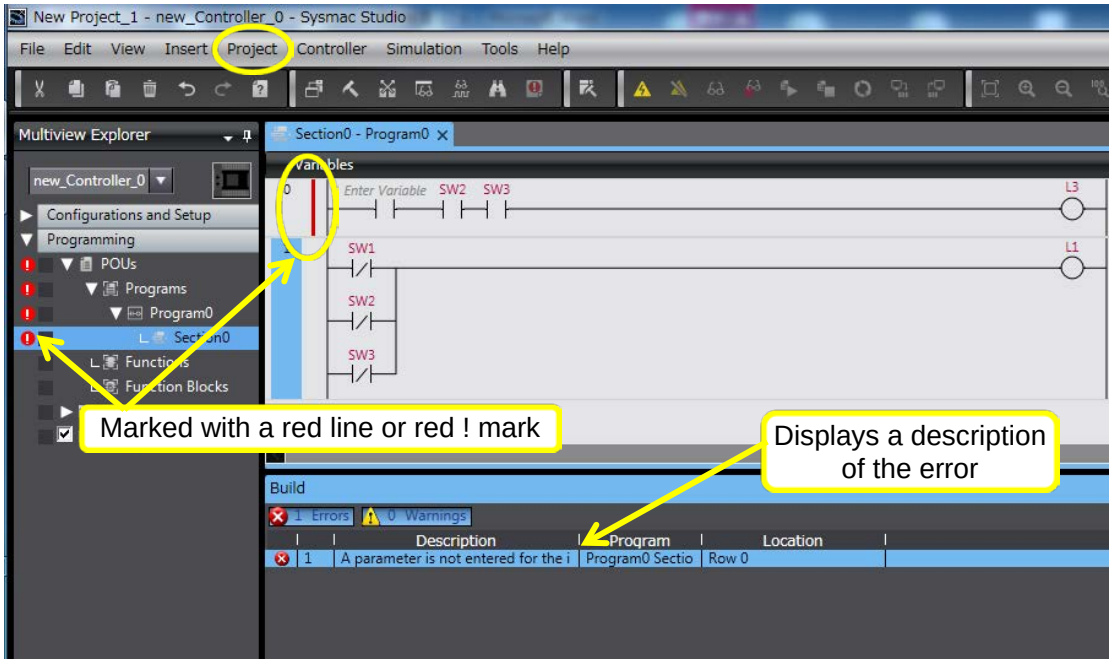


4. Select ***Stop*** from the Simulation Menu to stop the Simulator.



3-5-7 Example of a Program Error (Offline)

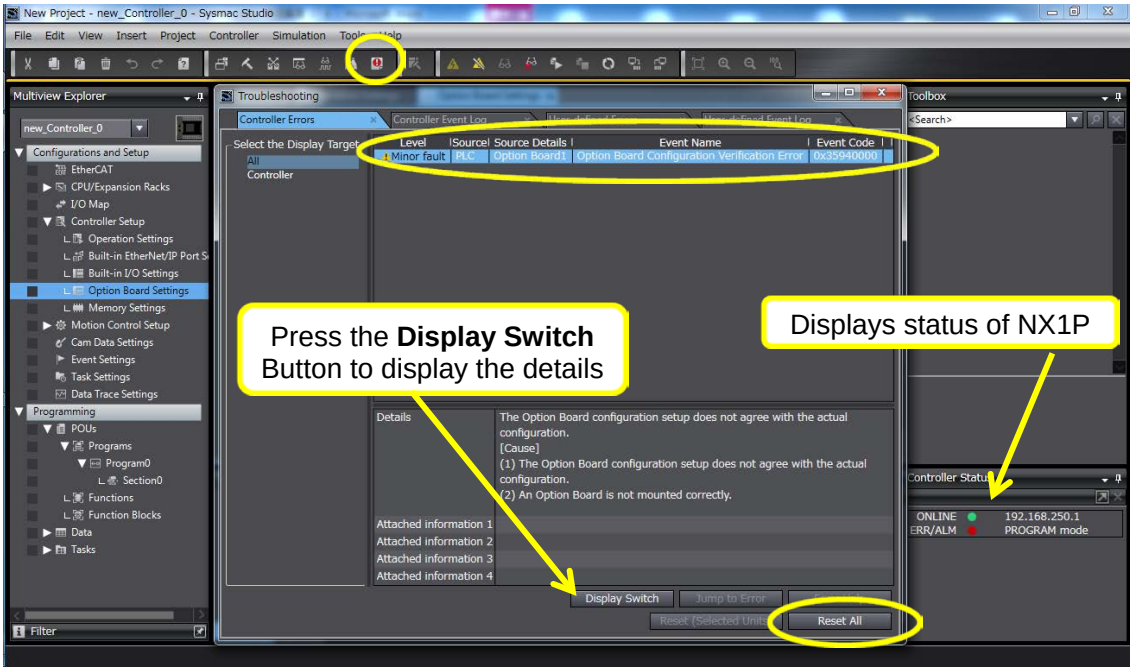
Delete the variable name “SW1” of input *SW1* offline. Select ***Check All Programs*** from the Project Menu. Errors are displayed in the Build Tab Page.



3-5-8 Example of an Error Occurred During Operation

Click the **Troubleshooting** Button (!) in the toolbar when an error occurs. The example below shows a verification error that occurs when the NX1W-CIF01 Serial Communications Option Board is not connected physically but is connected on the Sysmac Studio.

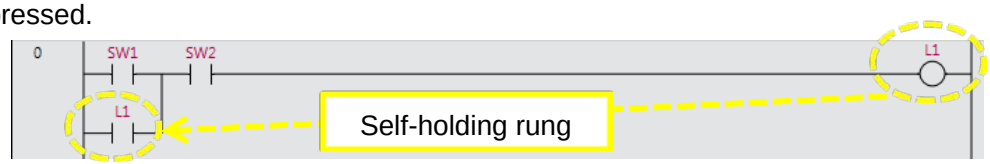
* Double-click **Option Board Settings** under **Configurations and Setup - Controller Setup** to configure the Option Board settings.



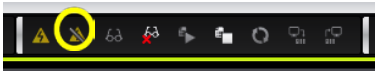
3-6 Example of a Ladder Program Using a Timer Instruction

3-6-1 Self-holding Rung

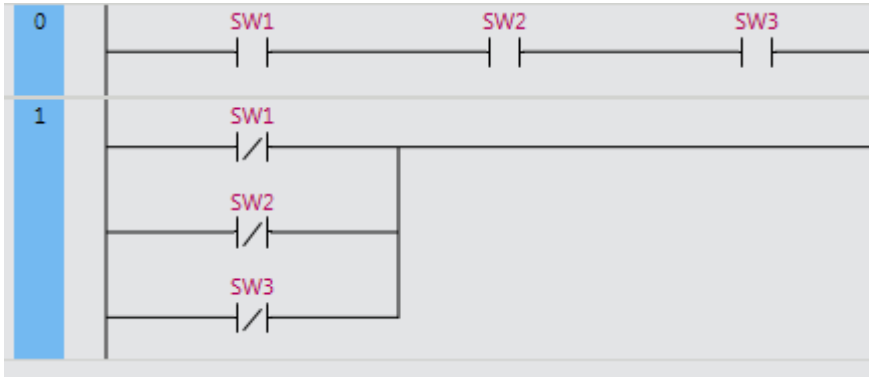
Create a self-holding rung to turn ON L1 when SW1 is pressed and stay lit until SW2 is pressed.



1. Create the program offline.



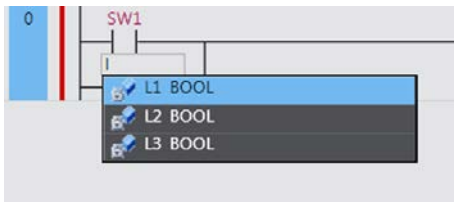
2. Delete the program created in *3-5 Example of a Basic Ladder Program*.
Right-click the rung numbers to delete while holding down the **Ctrl** Key. Press the **Delete** Key.



Although the ladder program is deleted, the I/O Map settings are not deleted and remain the same as those configured in *3-3-2 I/O Map Setting*.

3. Create the following rungs.

4. Click the *SW1* input, and then press the **W** Key to insert *L1* in an OR structure.



5. Click the connecting line to insert an N.C. Input. Press the ***/*** Key and enter “SW2”.

6. The self-holding rung is created.

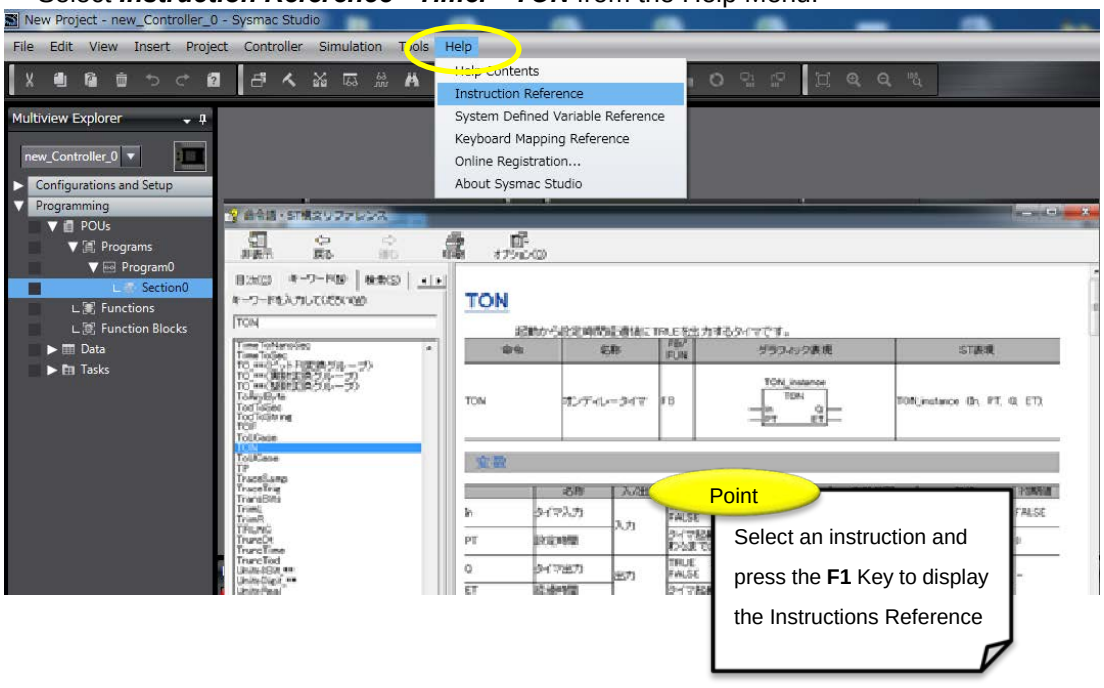
3-6-2 On-Delay Timer (TON) Instruction

Create a rung to turn ON L2 in five seconds after SW1 is pressed.

Rung to add

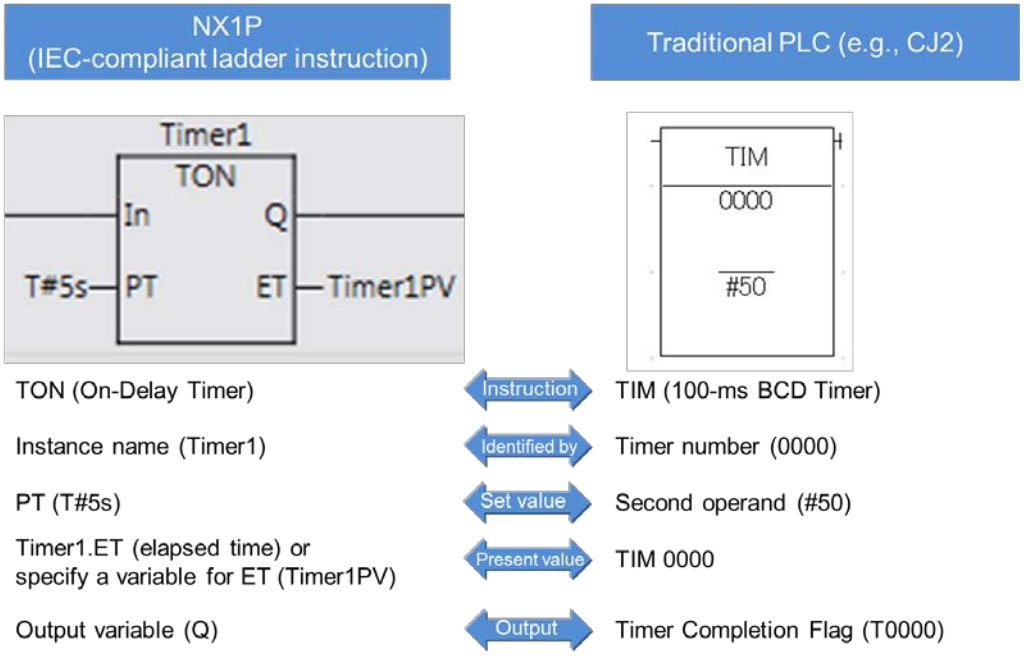
(1) Refer to the help for details of the TON instruction

- Select ***Instruction Reference - Timer - TON*** from the Help Menu.



Additional Information

Difference between the TON instruction for the NX1P and the TIM instruction for the traditional PLC



The TON instruction changes timer output *Q* to TRUE when the set time *PT* elapses after timer input *In* changes to TRUE.
The timer is reset when *In* changes to FALSE. Elapsed time *ET* changes to 0 and *Q* changes to FALSE.

(2) Input the set time

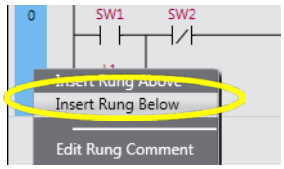
Specify a TIME data variable for the input parameter *PT* when inputting the set time.
For example, input “T#10.12s” to set to 10.12 seconds.

Classification	Data type	Data size	Alignment	Range of values	Notation
Durations	**TIME**	64 bits	8 bytes	T#-9223372036854.775808ms (T#-106751d_23h_47m_16s_854.775808ms) to T#+9223372036854.775807ms (T#+106751d_23h_47m_16s_854.775807ms)	T#12d3h3s T#3s56ms TIME#6d_10m TIME#16d_5h_3m_4s T#12d3.5h T#10.12s T#61m5s (Equivalent to T#1h1m5s) TIME#25h_3m

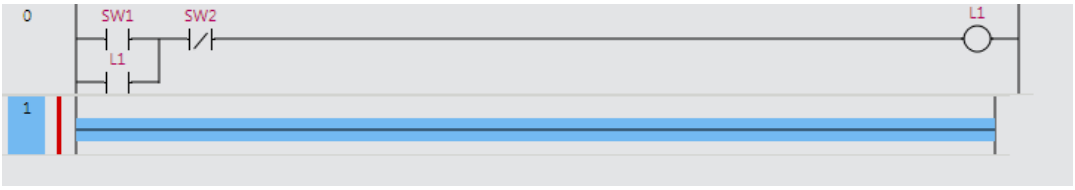
- Adding a rung using the On-Delay Timer instruction

1. Insert a rung below.

Right-click the existing rung and select ***Insert rung below***, or select the start of a rung and press the **R** Key.

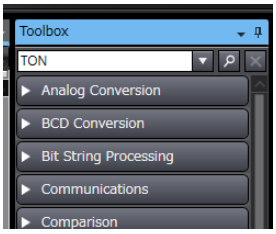


2. A rung is inserted below.



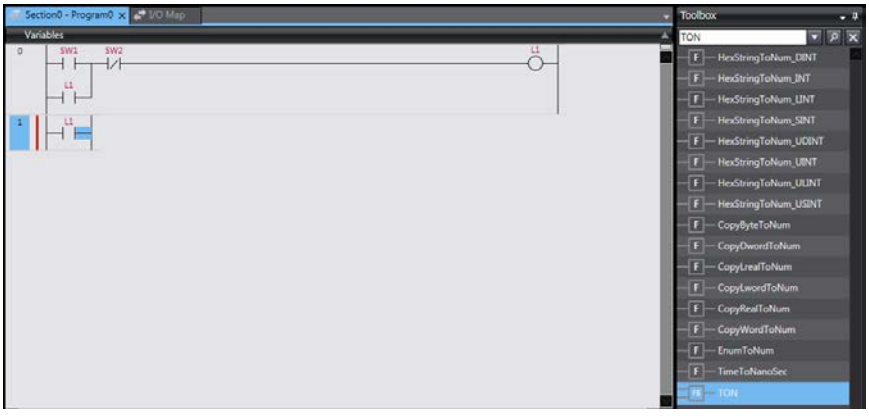
3. Insert the N.O. input *L1* as shown below.

4. Search for the TON instruction in the Toolbox on the right of the window or select **TON** in the **Timer** in the Toolbox.

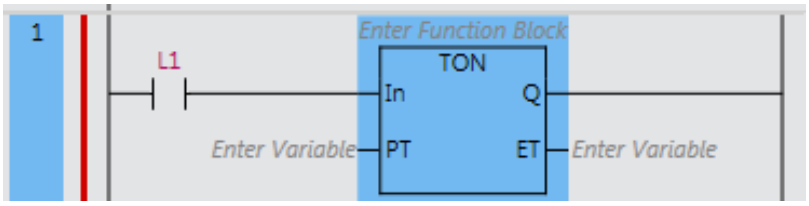


5. Add the TON instruction by dragging it from the Toolbox.

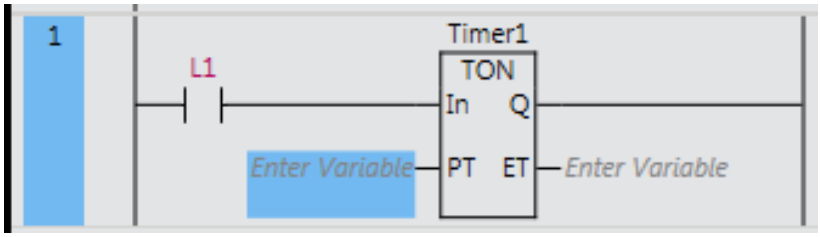
* You can also insert the TON instruction by right-clicking the desired location, selecting ***Insert Function Block*** from the menu, and entering “TON”.



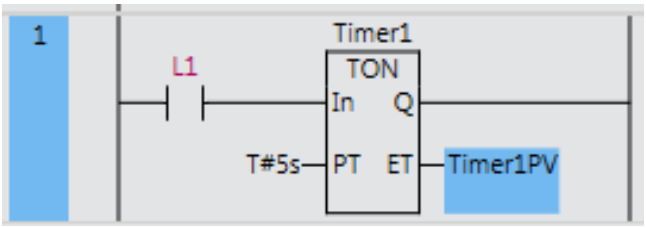
6. The TON instruction is inserted.



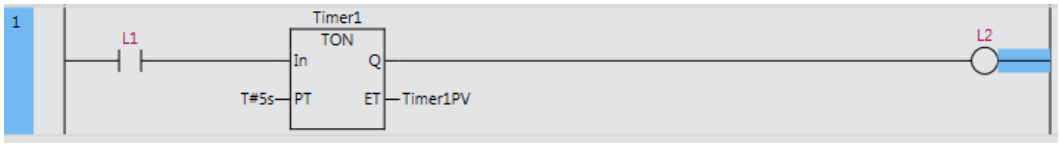
7. Enter the instance name of the TON instruction.
Click *Enter Function Block* and enter “Timer1”.



8. Set the parameters.
PT: T#5s
ET: Timer1PV

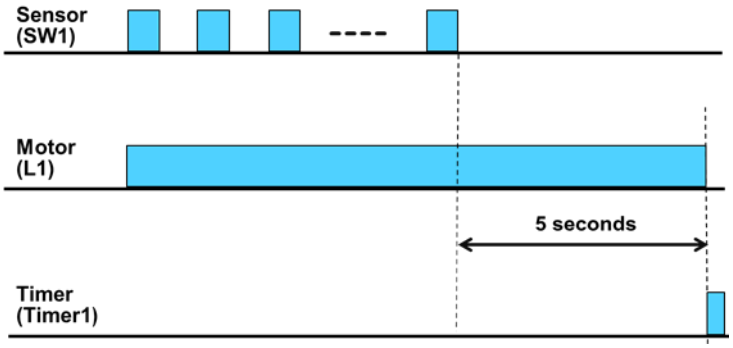


9. Insert an output that changes to TRUE when *Timer1* times out.
Enter “L2” for the variable name.



3-6-3 Exercise: Energy Saving Escalator

This section explains the operation of the TON instruction.
This escalator does not move until someone approaches it. When a person passes in front of the sensor (SW1), the motor (L1) starts. In order to save energy, the motor stops in five seconds after the last person passes.



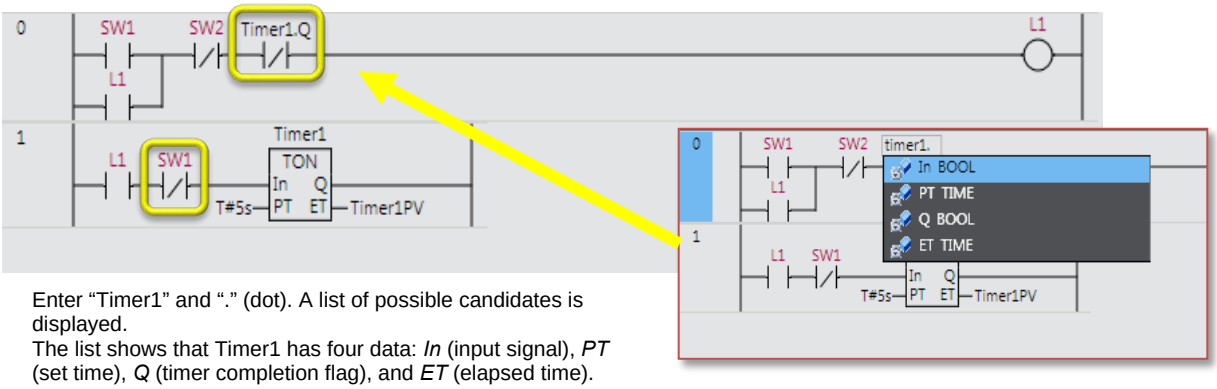
Tips

(1) Modify the created program.
(2) Insert an N.C. Input for timer output in the rung to stop the motor. There are two N.C. input methods
① Specify timer output as a work bit (e.g., Timer1UP).
② Use Timer1.Q that represents the output status of the timer instruction.
(3) Modify so as to reset the timer when a person passes.

Add an N.C. Input (*Timer1UP*) in the first rung to stop *L1* (the motor of the escalator in this example) when *Timer1UP* changes to TRUE. Add another N.C. Input (*SW1*) to reset the present value of the TON instruction. The program is completed.

■ Example (Output *Timer1UP* for when *Timer1* times out)

■ Example (Using Timer1.Q)



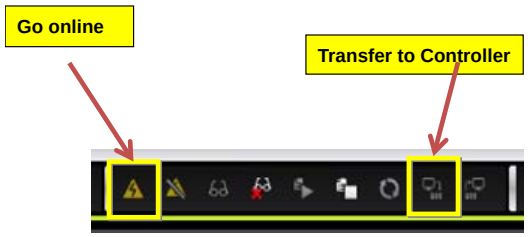
Enter “Timer1” and “.” (dot). A list of possible candidates is displayed.
The list shows that Timer1 has four data: *In* (input signal), *PT* (set time), *Q* (timer completion flag), and *ET* (elapsed time).

3-6-4 Checking the Operation of the Program

Check the operation of the program.

1. Connect the NX1P to the computer (Sysmac Studio) via an Ethernet cable.

2. Go online, and then transfer the program to the NX1P.

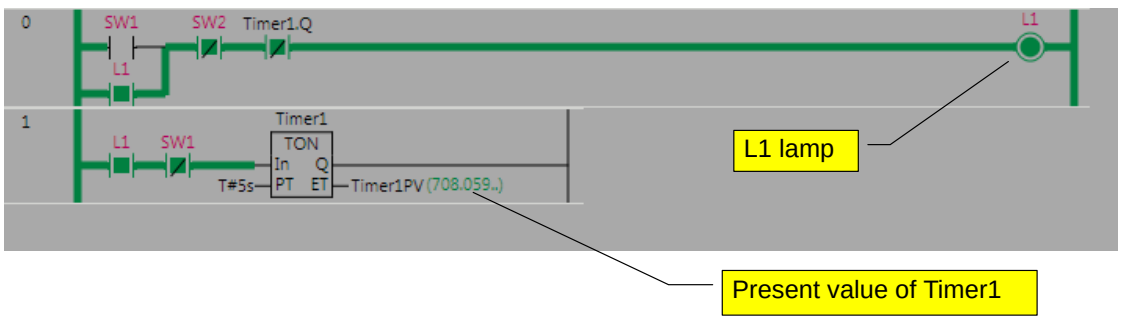


3. Change input *SW1* (passing person) to TRUE. Output *L1* (lamp 1) changes to TRUE and variable *Timer1PV* (present value of Timer1) is incremented as time elapses.

4. Change input *SW1* to FALSE, and check that variable *Timer1PV* is reset.

5. Change input *SW2* (stop button of the escalator) to TRUE, and check that output *L1* changes to FALSE and variable *Timer1PV* is reset.

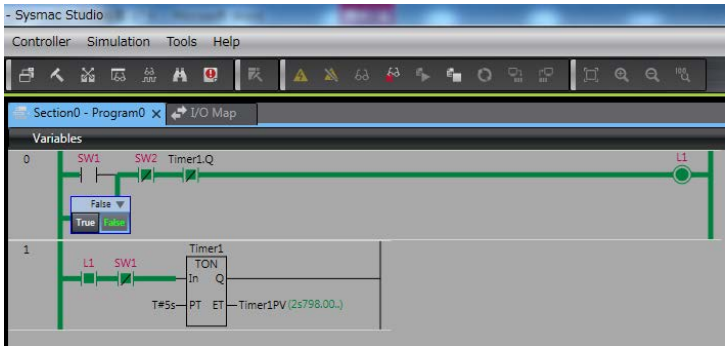
6. Change input *SW1* to TRUE. Check that output *L1* automatically changes to FALSE and variable *Timer1PV* is reset in five seconds after nothing is done.



● Checking the operation using the Simulator

1. Select ***Run*** from the Simulation Menu to start the Simulator.

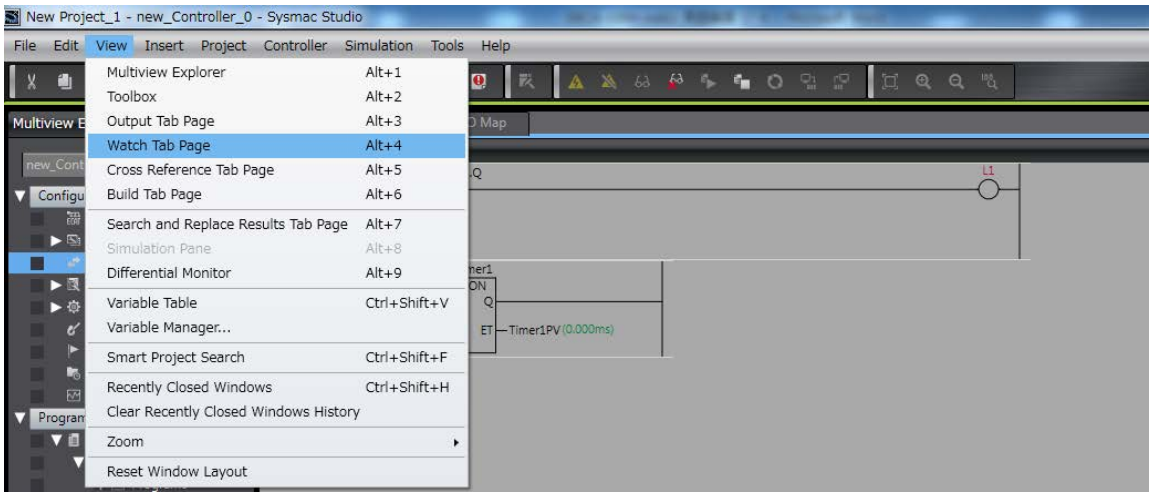
2. Double-click the input. You can change the value between True (ON) and False (OFF) to debug the programs, instead of pressing the physical switch.



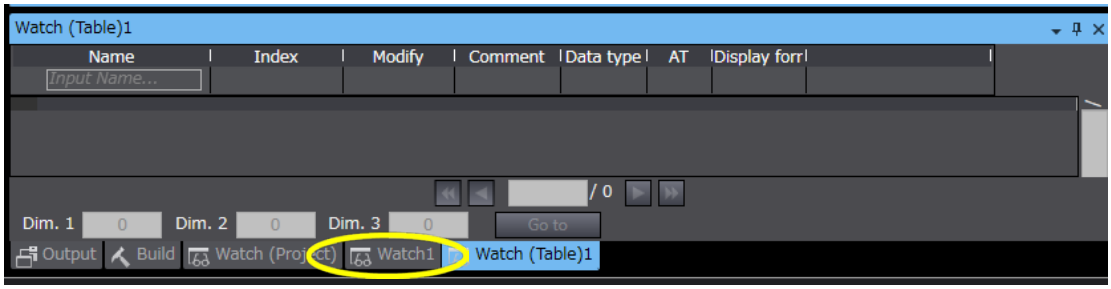
3-6-5 Checking the Operation of the Program (Watch Tab Page)

You can also check the operation of the program in the Watch Tab Page.
Monitoring can be performed online on the NX1P or offline with the Simulator in the same way.

1. Select ***Watch Tab Page*** from the View Menu. The Watch Tab Page is displayed at the bottom of the window.

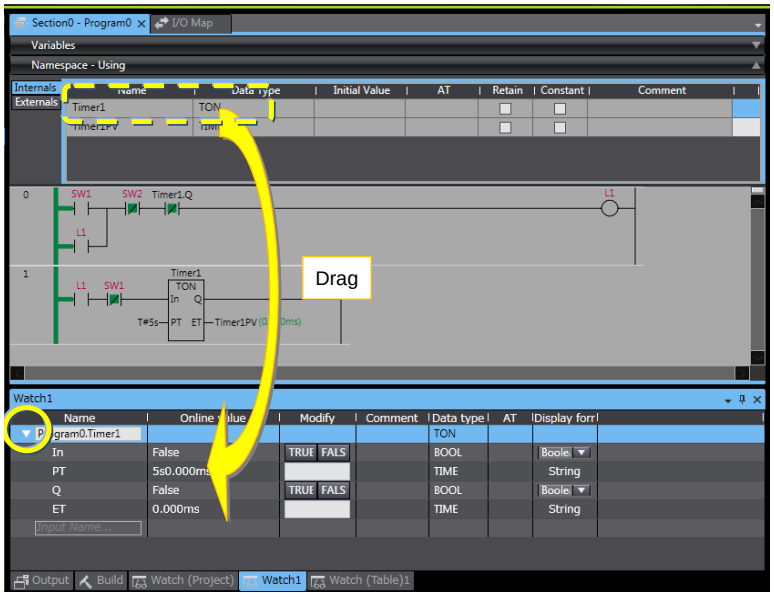


2. Click the **Watch Tab Page 1** Tab.



3. Click the **Variables** Bar at the top of the window. The variable table appears. Select the variable to monitor from the variables (external variables and internal variables) used in the program and drag it to the Watch Tab Page.

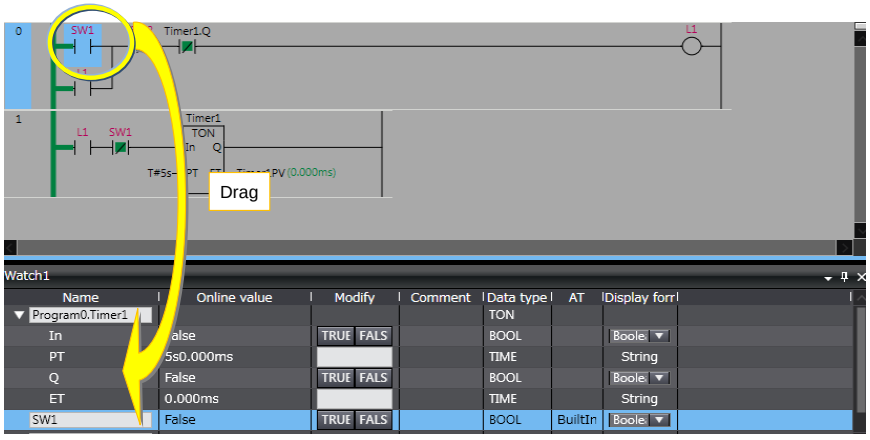
Register *Timer1* by dragging it to the Watch Tab Page.



Click the ▼ mark, and check that data (timer start flag (*In*), timer set value (*PT*), timer present value (*ET*), and timer completion flag (*Q*)) contained in *Timer1* can be monitored.

4. Select the variable to monitor in the Ladder Editor and drag it to the Watch Tab Page.

Register *SW1* by dragging it to the Watch Tab Page.



5. Execute the program. You can monitor the values.

6. Go offline before taking the next step.
Save and export the project file.

- Select ***Save*** from the File Menu.
- Select ***Export*** from the File Menu to export the file to the desktop.

3-7 Example of a Ladder Program Using Date and Time

3-7-1 Programming the NX1P Using Date and Time

For example, best before date and time that is 30 hours from production is printed on boxed lunch.

A program is required to
acquire date and time of production and
calculate the best before date and time by adding 30 hours to the acquired date and time

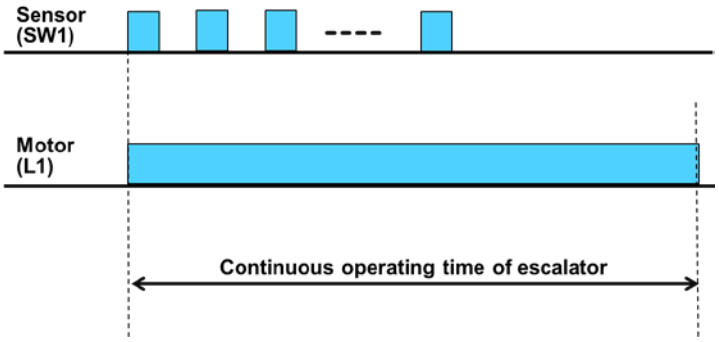
Programming with variables uses DATE_AND_TIME data (year, month, day, hour, minute, and second), TIME data, and instructions to perform calculations easily.

3-7-2 Exercise: Continuous Operating Time of Escalator

Create a program to measure time by using the program created in *3-6-3 Exercise: Energy Saving Escalator*.

[Exercise] Acquire current time and calculate elapsed time

Measure continuous operating time of the escalator (continuous ON time of L1). Create a program to subtract time of day when L1 turns OFF from time of day when L1 turns ON.



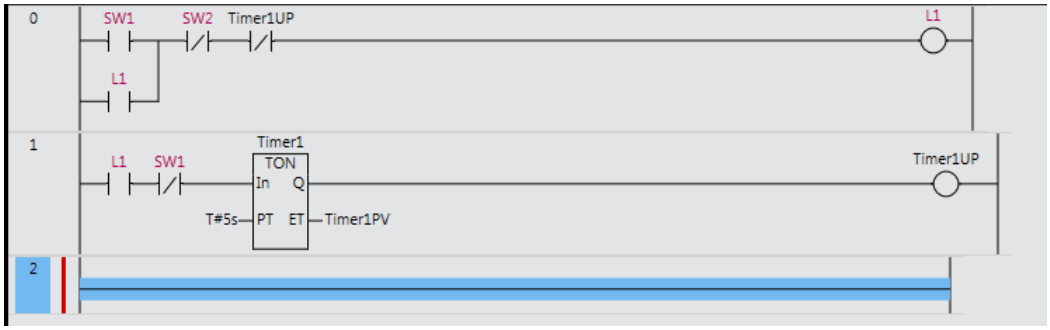
Tips

(1) Use the GetTime function to acquire current time.
(2) Use the SUB_DT_DT function to subtract date and time.
(3) The SUB_DT_DT function returns TIME data.

Add code to the program created in *3-6-3 Exercise: Energy Saving Escalator*.

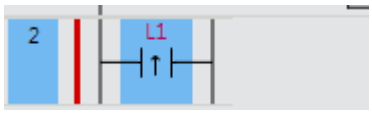
1. Insert a rung below.

Right-click the rung 1 and select ***Insert rung below***, or select the start of a rung and press the **R** Key.



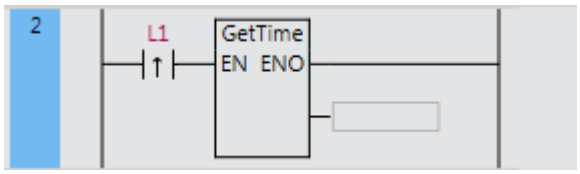
2. Set upward differentiation for *L1*.

Press the **C** Key, or right-click a connecting line and select ***Insert Input*** from the Menu to insert an input. Press the @ Key, or right-click the input and select ***Diff Up*** from the menu.



3. Insert the GetTime function to acquire date and time when *L1* changes to TRUE.

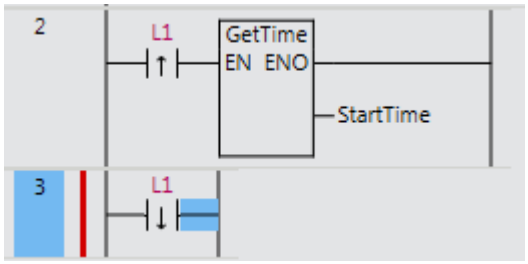
Press the **I** Key and enter “GetTime” as the function name



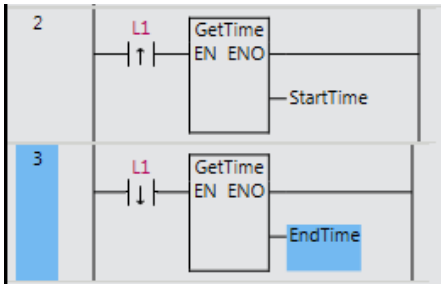
Enter “StartTime” as the output variable name.

4. In the same way create another rung to execute the GetTime function when *L1* changes to FALSE.

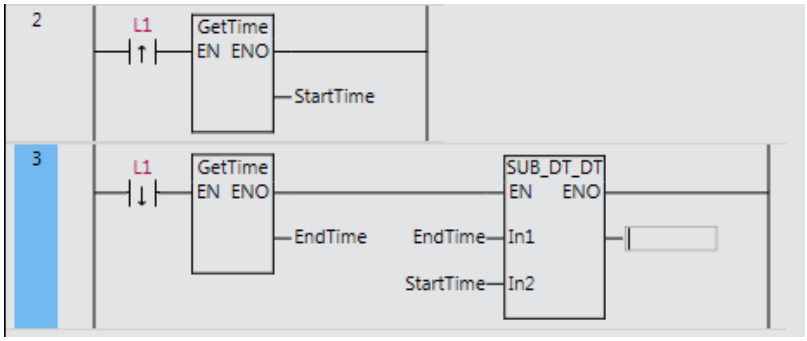
To set downward differentiation, press the **%** Key, or right-click the input and select ***Diff Down*** from the menu.



5. Insert the GetTime function and enter “EndTime” as the output variable name.

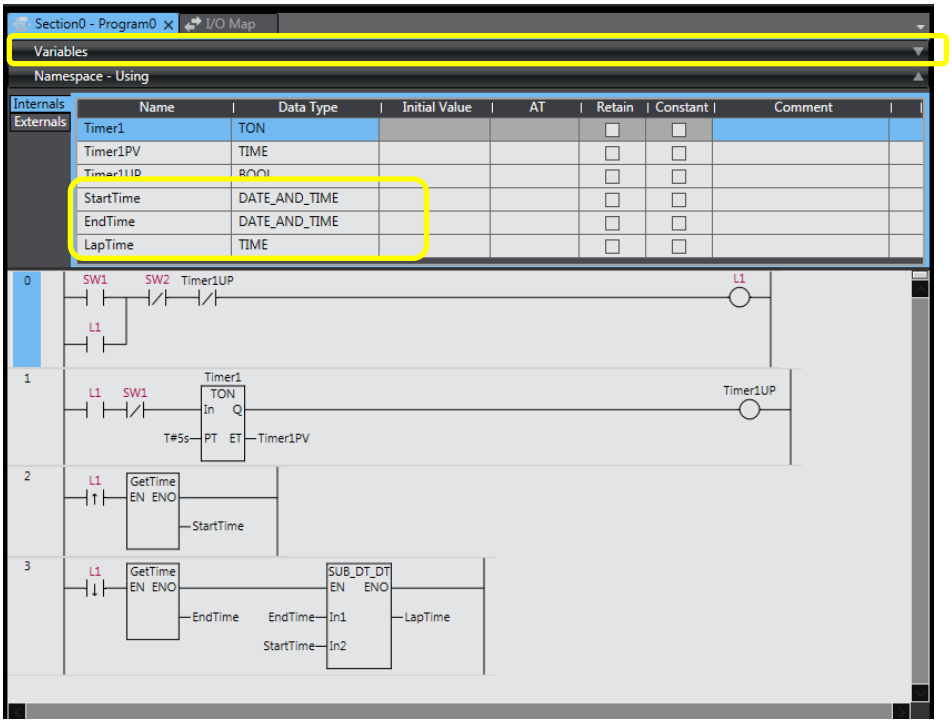


6. Insert the SUB_DT_DT (Subtract Date and Time) instruction to subtract *StartTime* from *EndTime*.



Enter “LapTime” as the output variable name.

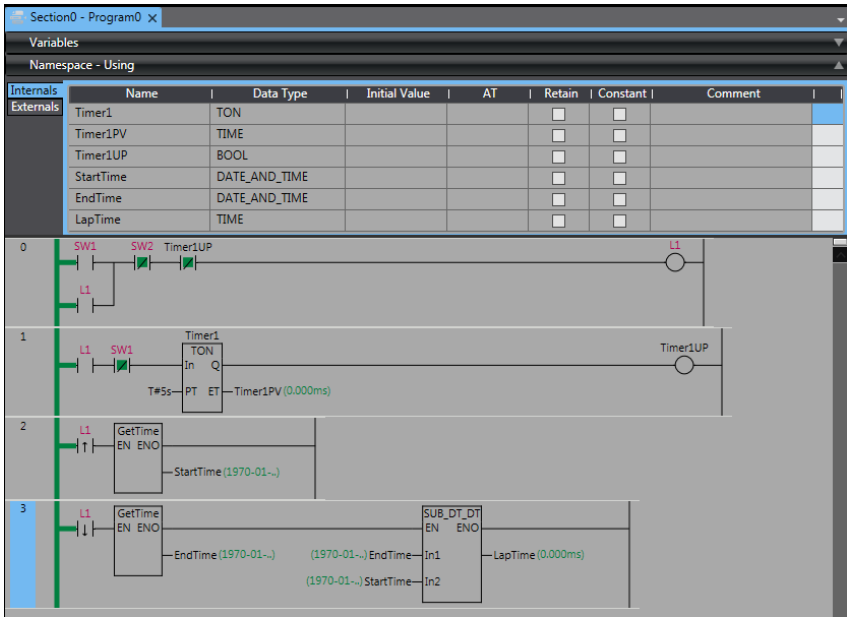
7. Click the **Variables** Bar at the top of the window to check the variable table.



StartTime and *EndTime* are registered as DATE_AND_TIME (date and time) data and *LapTime* as TIME (durations) data.
The data types are automatically set according to the used instructions.

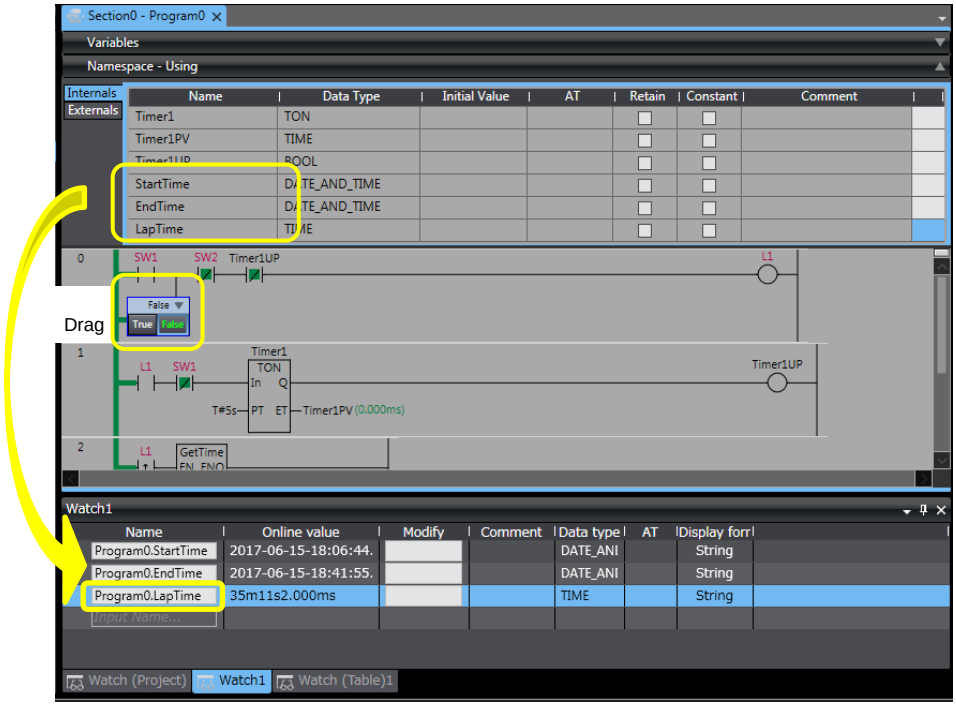
8. Execute the program.

Transfer the program to the NX1P or change the operating mode to RUN mode to use the Simulator in the Sysmac Studio.



9. Click the **Watch Tab Page 1** Tab.

Select *StartTime*, *EndTime*, and *LapTime* in the variable table and drag them to the Watch Tab Page. Change the value of input *SW1* from False to True and check the value of variable *LapTime*. The value of variable *LapTime* shows the time from when output *L1* changes to TRUE to when output *L1* changes to FALSE.

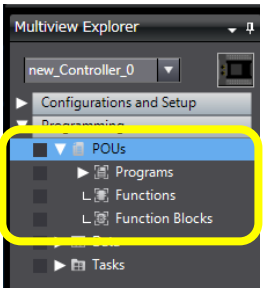


10. Go offline before taking the next step. Save and export the project file.

3-8 Fundamentals of Programming to Reduce Development Time

3-8-1 POUs (Program Organization Units)

- POUs

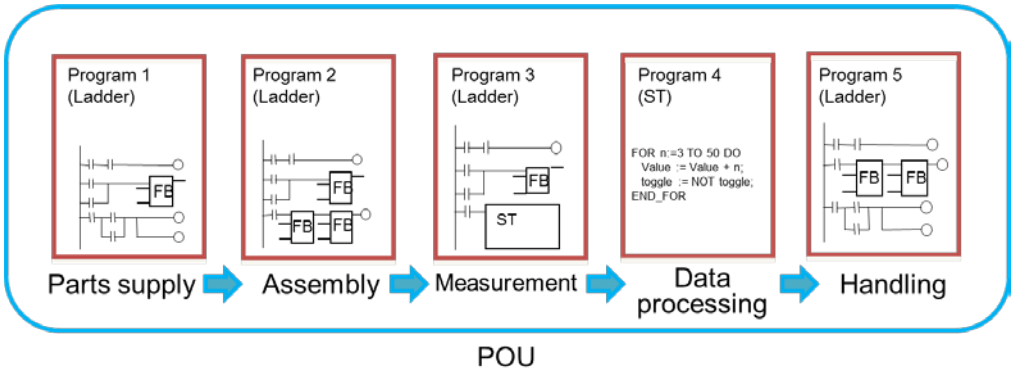


A POU (program organization unit) is a unit that is defined in the IEC 61131-3 and used to build the user program. There are three types of POUs: programs, functions (FUNs), and function blocks (FBs). FUNs and FBs that are reusable software components make programming easier.

3-8-2 Programs and Execution Priorities (Tasks)

- Programs

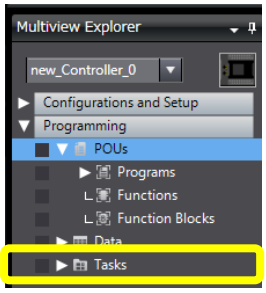
Separate programs for different processes make programs easy to read and reuse.



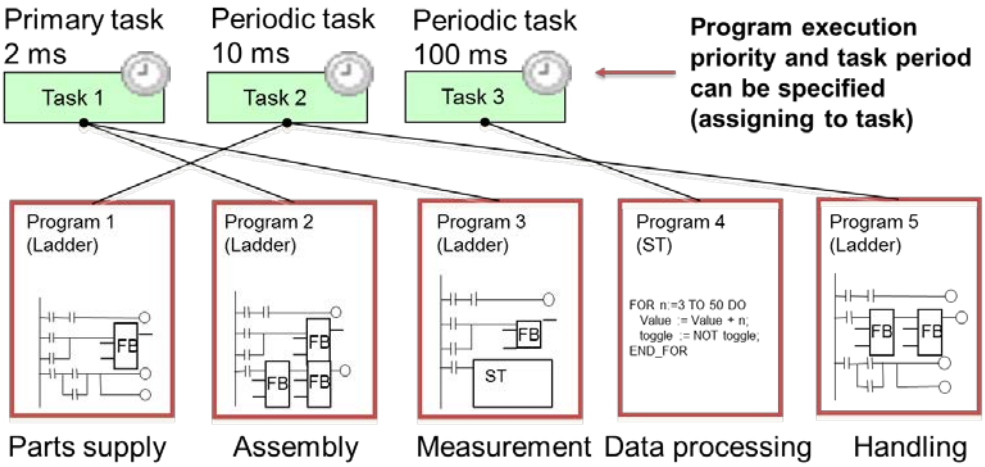
POU

Two different programming languages, ladder diagram and ST, can be used. You can choose the appropriate language for each process and also program in ST within a ladder diagram program (inline ST).

- Tasks

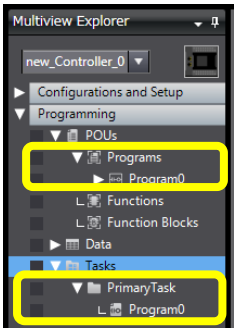


A task is an attribute that defines when a program is executed. You can set a task period for each program.
To execute processes with high speeds, assign the program to the primary periodic task that has the highest execution priority.
By assigning processes that do not require high-speed processing to the task that has the lower execution priority, you can reduce the load on the NX1P. One or more programs can be assigned to one task.

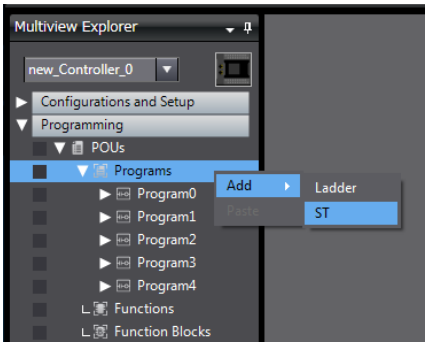


- Default task setting and addition of a program

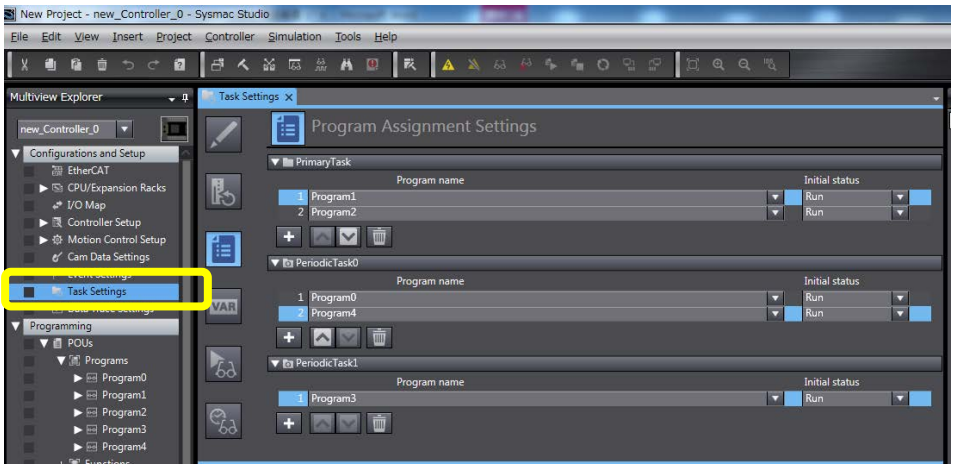
1. When a project is created in the Sysmac Studio, *Program0* (ladder program) is registered in advance and assigned to the primary periodic task by default.
Create a program in *Program0* because there is no need to worry about task setting.



2. When adding a program, right-click **Programs** under **Programming - POUs** and select ***Add - Ladder*** or ***ST*** from the menu.
Program1 is added. *Program2* will be added when you add another program. When changing the name of the program, right-click *Program** and select ***Rename*** from the menu.

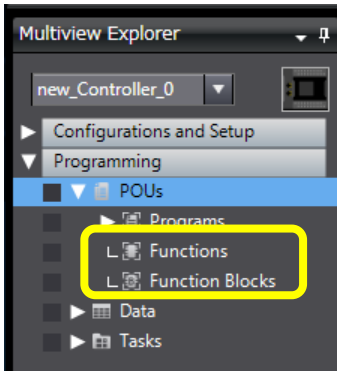


3. When assigning the added program to a task, double-click **Task Settings** under **Configurations and Setup** and click the **Task Settings** Button and the **Program Assignment Settings** Button.

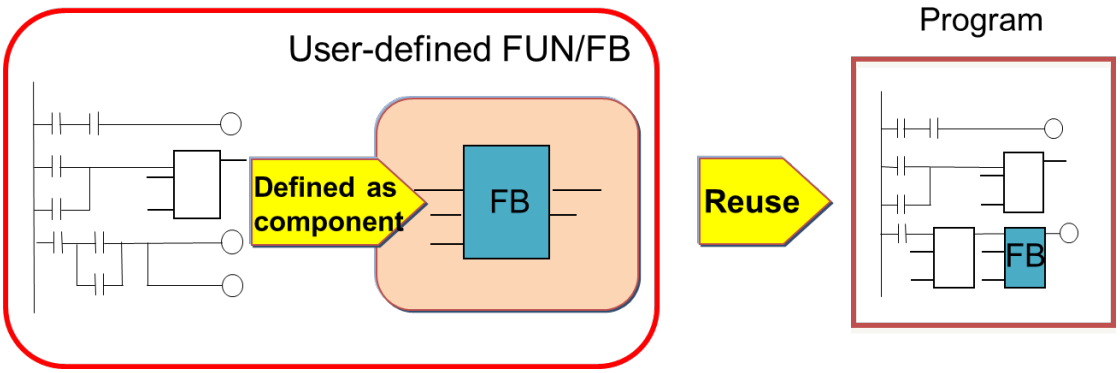


3-8-3 Functions (FUNs) and Function Blocks (FBs)

Functions (FUNs) and function blocks (FBs) are instructions used in programs.
In traditional PLCs like the CJ2, they are called special instructions (e.g., MOV instruction = FUN, TIM instruction = FB).
In addition to system-defined FUN/FBs, the uses can define their own FUN/FBs (user-defined FUN/FBs).

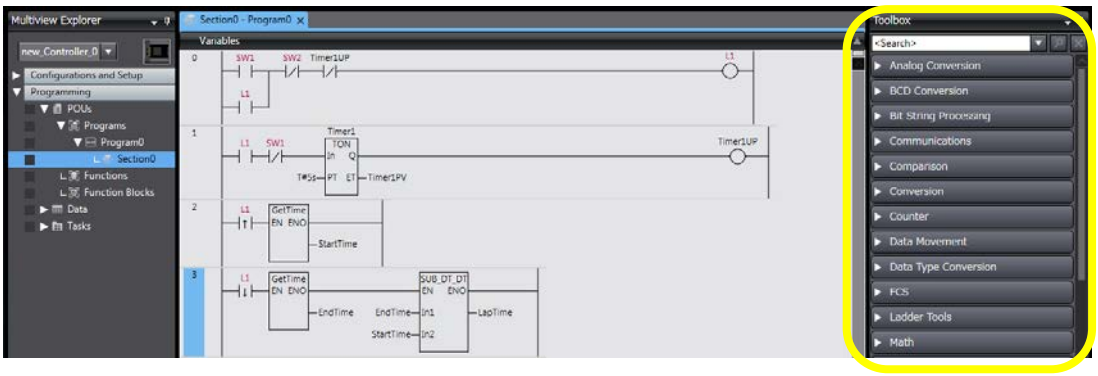


- User-defined FUNs/FBs
  You can define the existing programs that will be used in other programs as FUNs or FBs.
  You can program using the user-defined FUNs/FBs, which makes programming easier and faster.

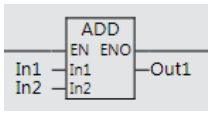
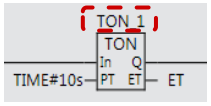


- System-defined FUNs/FBs

  The FUNs/FBs available for the NX1P are listed in the Toolbox on the right of the window. Drag an instruction to use in the program.

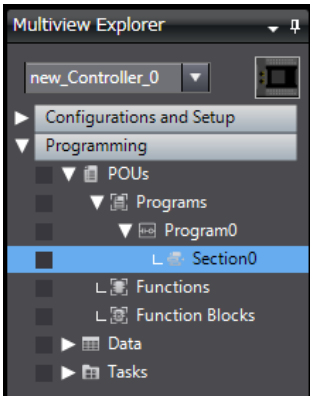


- Difference between FUN and FB

	Description	Figure that represents instruction	Example
Function (FUN)	An instruction that performs a single function. The values of internal variables are not retained.		Bit string processing (AND, OR, XOR, NOT) Math (ADD, SUB, MUL, DIV, SQRT, LN, LOG, EXP, SIN, COS, TAN) Comparison (GT, GE, EQ, LT, LE, NE) etc.
Function block (FB)	The values of internal variables are retained until the conditions are completed, such as for timers.	 The name (instance name) is required.	Set, reset (SR, RS) Trigger (R_TRIG, F_TRIG) Counter (CTU, CTD, CTUD) Timer (TP, TON, TOF) Motion control (MC_HOME, MC_MOVE) etc.

3-8-4 Sections

- Sections

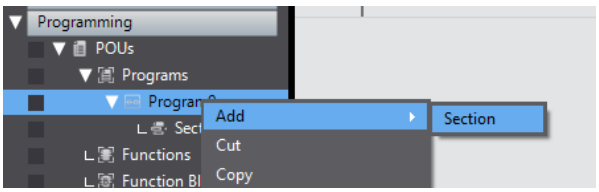


You can divide a ladder diagram into smaller units and set a name for each unit. This makes the program easy to understand and manage. The section can be moved and deleted.
Programs are executed from top to bottom in the order that the sections are displayed in the Multiview Explorer. To change the order of execution, you must change the order of the sections.

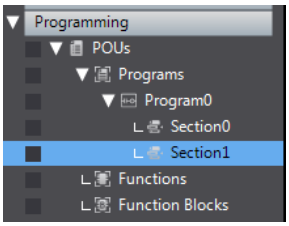
Section0 is registered in *Program0* by default.

- Adding a section

  1. Right-click **Program0** under **Programming - POUs - Programs** in the Multiview Explorer. Select ***Add - Section*** from the menu.



  2. A section with the name *Section1* is added under *Program0*.



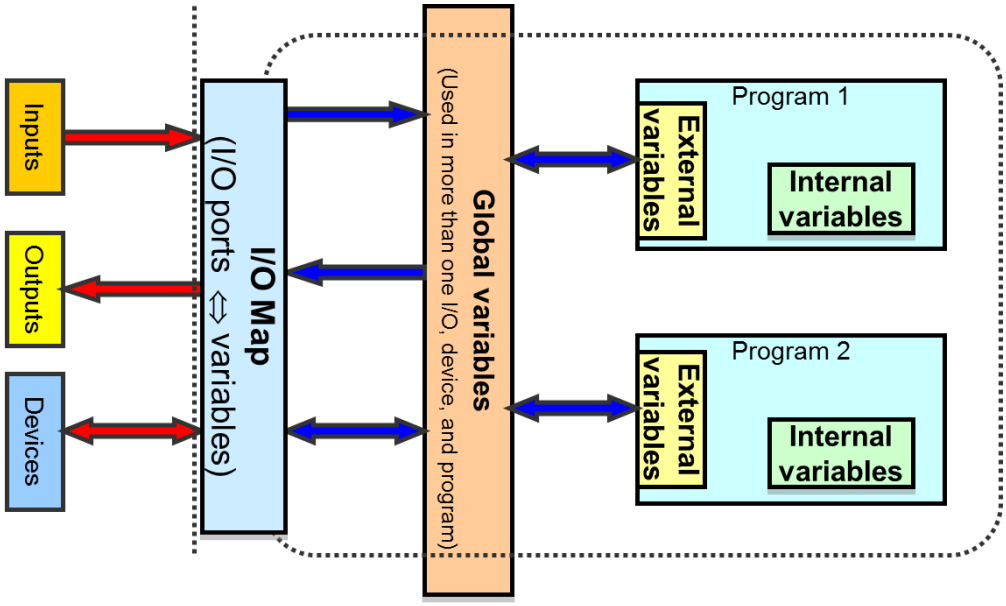
3-8-5 Types of Variables

- Global variables

  Global variables are variables registered in the I/O Map and used for more than one program. They can be accessed from any program.

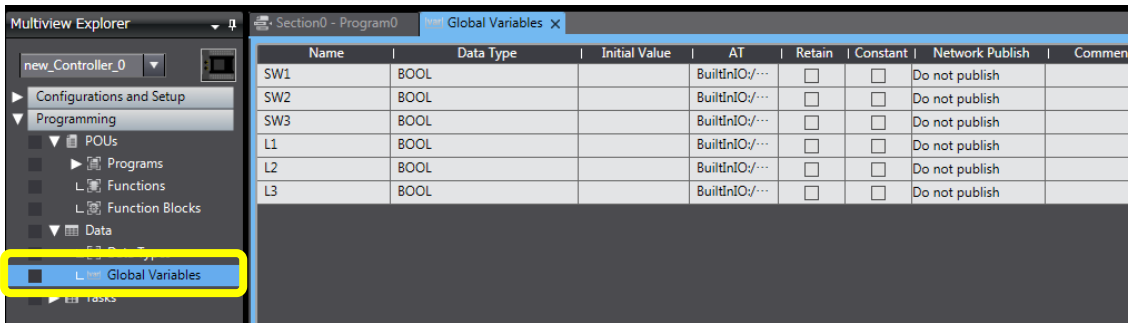
- External variables and internal variables

  External variables and internal variables are used only within one program.
  When global variables (*SW1* and *L1*) are used in programs, the global variables are registered as external variables. Variables that are registered in programs (*Timer1* and *Timer1PV*) are registered as internal variables.



- Checking global variable

  Check the global variables registered in *3-7 Example of a Ladder Program Using Date and* Time. Double-click **Global Variables** under **Programming - Data** in the Multiview Explorer.



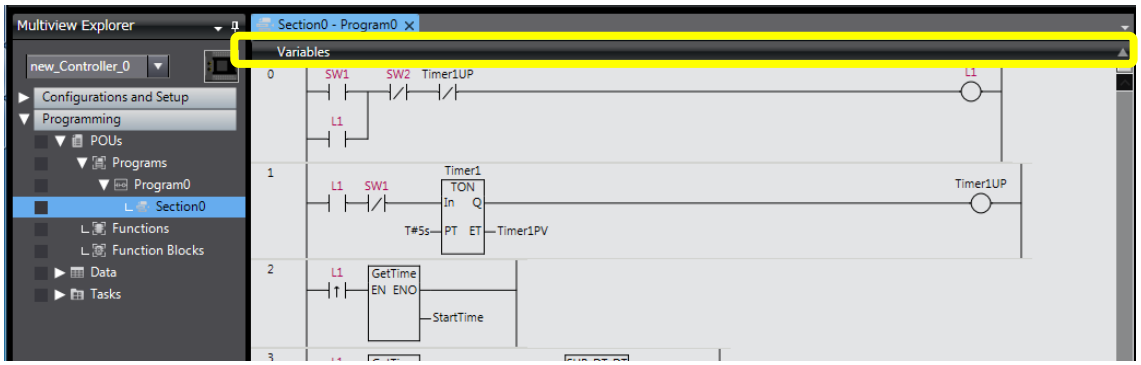
Check that the variables such as *SW1* and *L1* registered in the I/O Map are automatically registered as global variables.

Additional Information

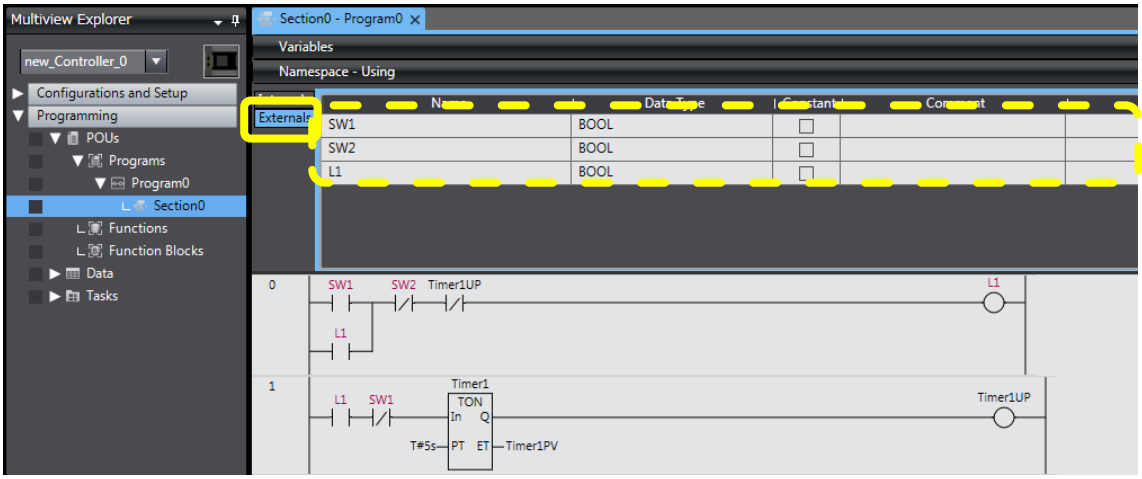
Global variables are displayed in purple.

- Checking the variable table

  Click the **Variables** Bar at the top of the Edit Pane.

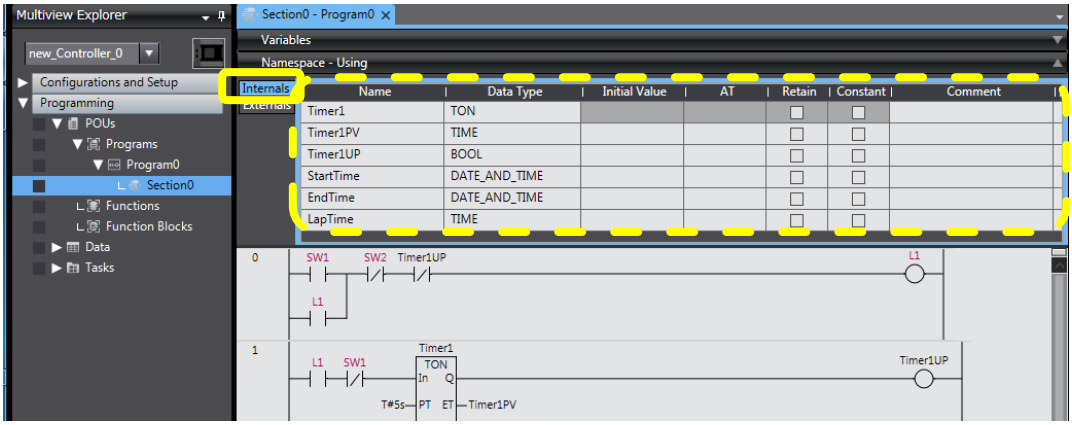


The local variable table is displayed. Click the **Externals** Tab.



When variables (global variables) registered in the I/O Map are used in this program, they

are automatically registered as external variables.
Click the **Internals** Tab.



Variables registered in this program are automatically registered as internal variables.

4

4 Creating Programs to Handle Data

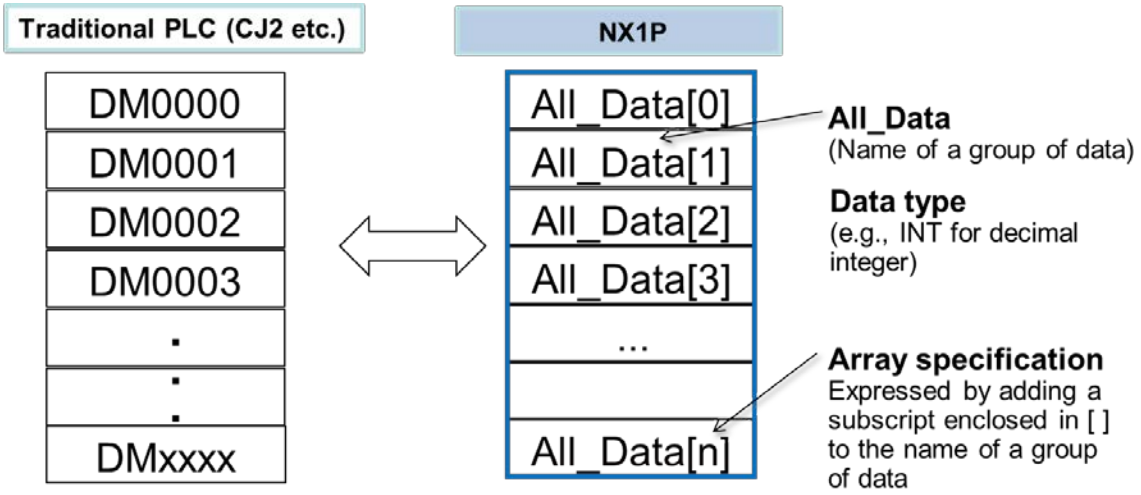
This section describes how to create programs to handle data.

4-1 Variables Used for Data Processing 4-74
4-1-1 Arrays 4-74
4-2 Programming Exercise 4-75
4-2-1 Application Example 4-75
4-2-2 Programming 4-75
4-2-3 Creating a Project 4-76
4-2-4 Configuring Analog Option Board Settings 4-77
4-2-5 Assigning Variables to the Option Board and Input Terminal 4-77
4-2-6 Program Example 4-78
4-2-7 Creating an Array 4-79
4-2-8 Entering Programming Code 4-80
4-2-9 Checking the Operation of the Program 4-81
4-2-10 Referring Values of Array Variables 4-83

4-1 Variables Used for Data Processing

4-1-1 Arrays

The CJ2 and other traditional PLCs use Data Memory Area as a memory area for data processing and storage.
The NX1P does not have Data Memory Area and uses variables as memory used for data processing.

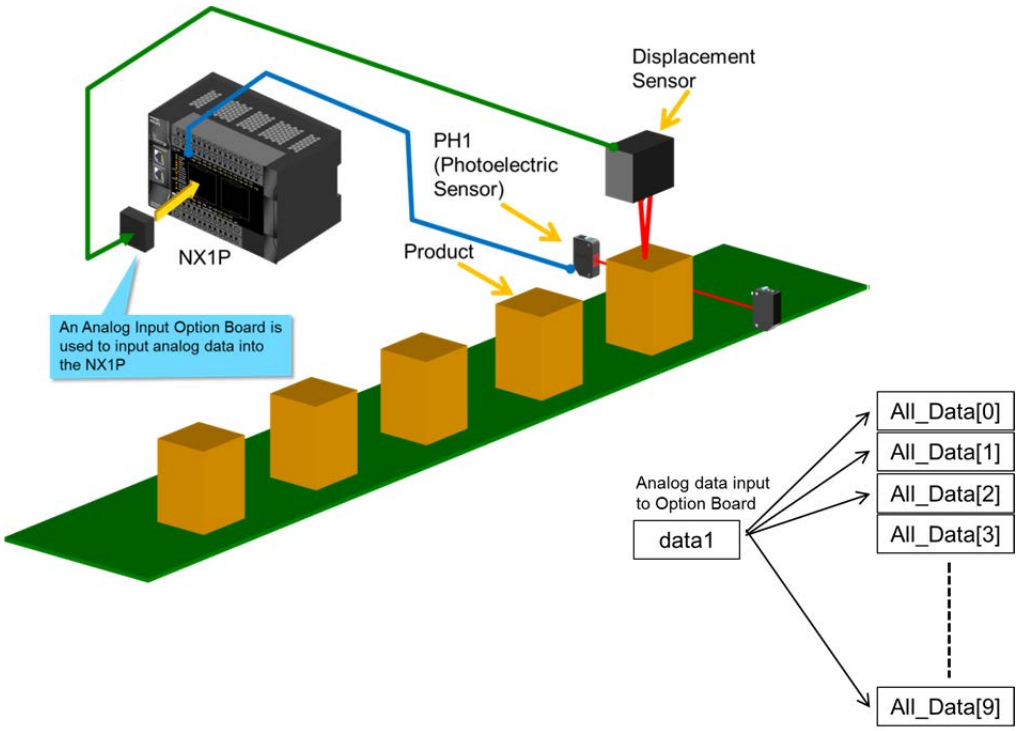


- Arrays
  All_Data[n] shown above is called an “array”.
  The elements of an array are expressed by adding a [subscript] to the name of the variable that represents the entire array.
  An element expressed by “variable name [subscript]” (e.g., All_Data[3]) is used as a variable in programs.
  Only one data type can be set for an array variable.
  One name can be used for multiple variables, making the program easy to understand and read.

4-2 Programming Exercise

4-2-1 Application Example

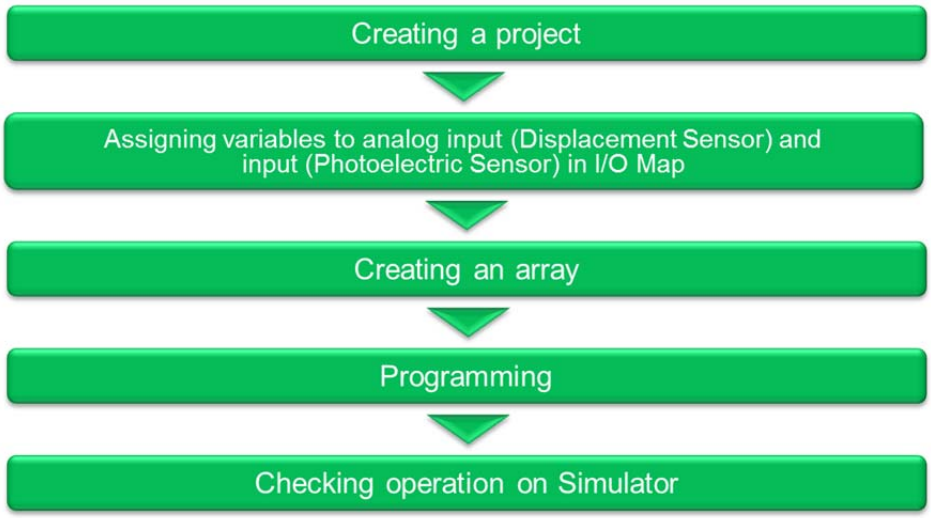
Create a program to store the first 10 values measured by the Displacement Sensor when the Photoelectric Sensor (PH1) turns ON.



The value measured by the Displacement Sensor is stored in variable *data1* as analog data and then stored as an element of array variable *All_Data*.

4-2-2 Programming

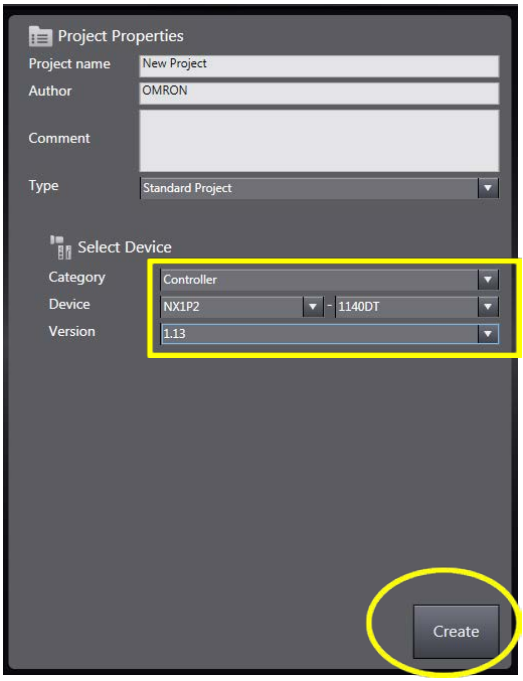
This section describes the procedure to check operation using the Simulator in the Sysmac Studio, without using physical devices.



4-2-3 Creating a Project

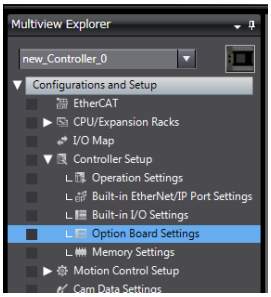
1. Start the Sysmac Studio.

2. Enter the project name. Select *NX1P2, 9024DT/1140DT* for the *device* parameter and *1.13* (version indicated on the NX1P) for the *version* parameter, and then click the **Create** Button.

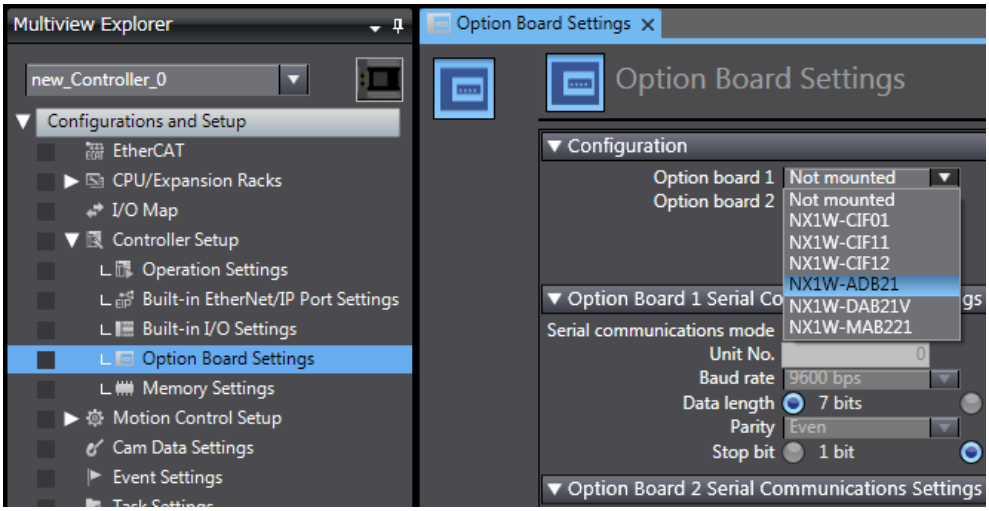


4-2-4 Configuring Analog Option Board Settings

1. Double-click **Option Board Settings** under **Configurations and Setup - Controller Setup** in the Multiview Explorer.

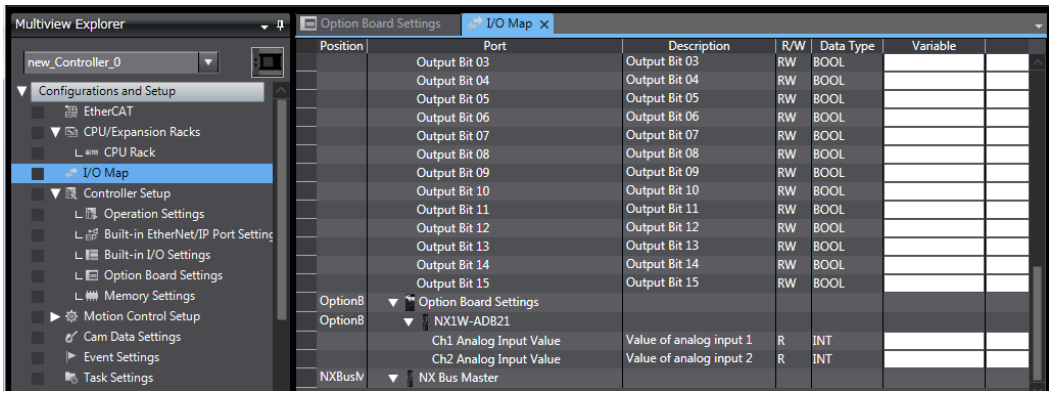


2. Select *NX1W-ADB21* (Analog Input Option Board) for the *Option board 1* parameter.

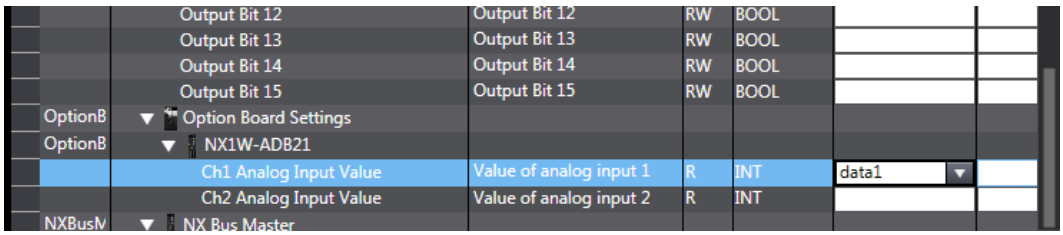


4-2-5 Assigning Variables to the Option Board and Input Terminal

1. Select **Configurations and Setup - I/O Map**.
NX1W-ADB21 is displayed at the bottom of the I/O Map.

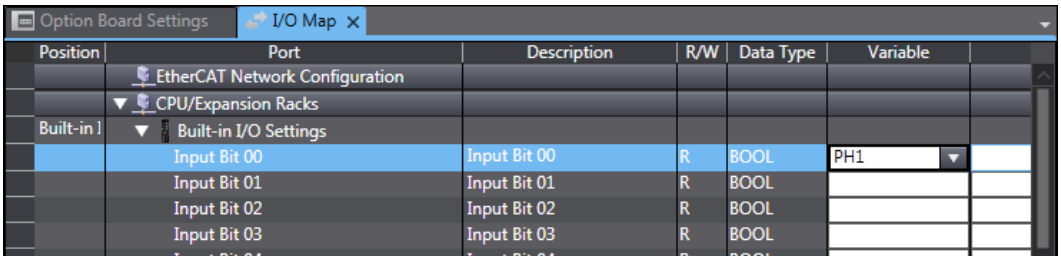


2. Select Ch1 Analog Input Value and enter “data1” in the *Variable* Column.



INT data from 0 to 4000 is stored in variable *data1* according to the analog input value of the Displacement Sensor (0 to 10 V).

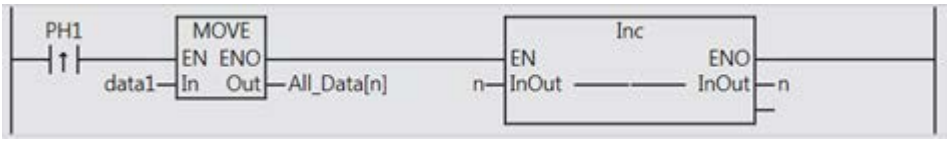
3. Enter “PH1” in the *Variable* Column of Input Bit 00 to which the Photoelectric Sensor is connected.



Variable *PH1* is changed between True (ON) and False (OFF) by changing the ON/OFF state of the Photoelectric Sensor.
The data type is Boolean.

4-2-6 Program Example

When *PH1* changes to TRUE, the MOVE (data movement) instruction stores the value of variable *data1* in the *n*th element of array variable *All_Data* and the Inc (increment) instruction adds 1 to *n* (n = n + 1).
The first time *PH1* changes to TRUE, *data1* (analog input value) is stored in *All_Data[0]*. The next time *PH1* changes to TRUE, it is stored in *All_Data[1]*, and then *All_Data[2]*.

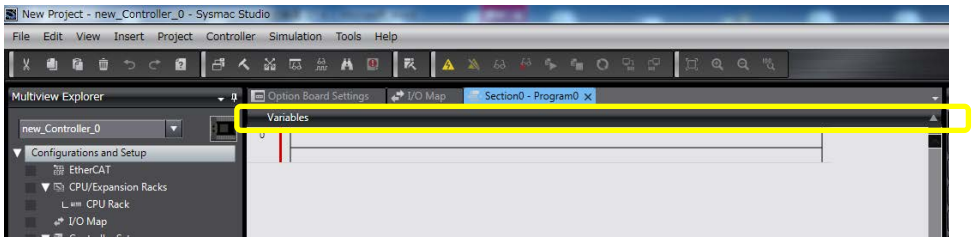


4-2-7 Creating an Array

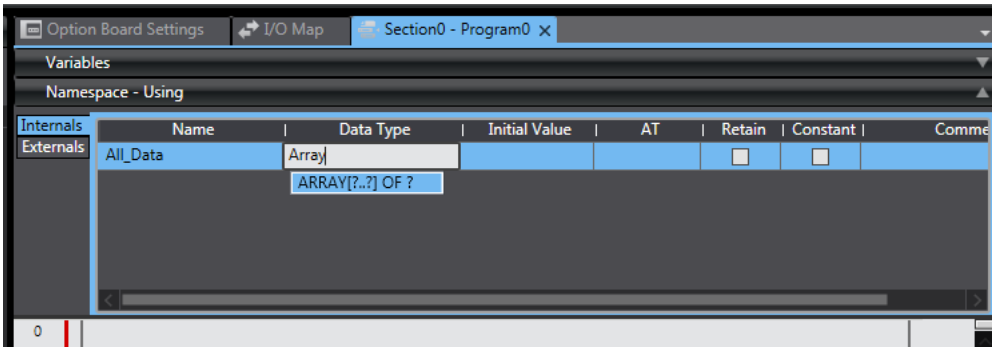
Create array variable *All_Data[n]*.

All_Data[0]
All_Data[1]
All_Data[2]
All_Data[3]
All_Data[n]

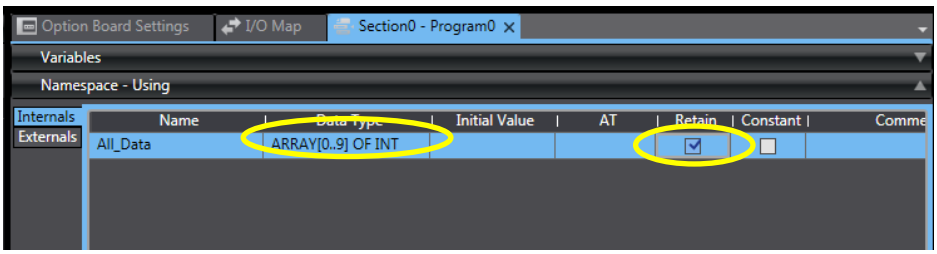
1. Double-click **Program0 - Section0** and then click the **Variables** Bar at the top of the Edit Pane to display the variable table.



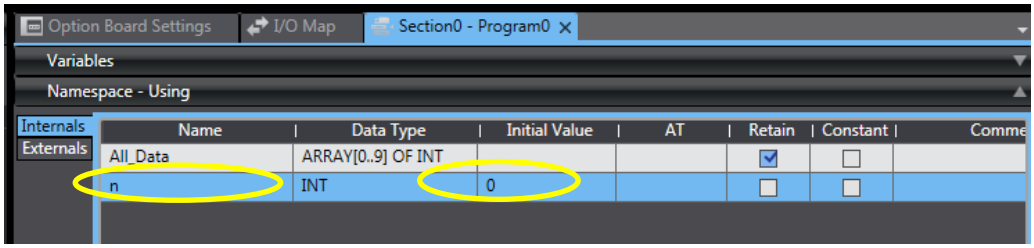
2. Click the **Internals** Tab to create internal variables.
Use an array specification for a data type.
Enter “All_Data” into the *Name* Column and then enter “Array” into the *Data type* Column.
The data type name candidate *ARRAY[?..?] OF ?* appears.



3. Enter “0” for the left question mark and “9” for the right question mark in the [?..?] section. Next, enter “INT” for the question mark in the OF ? section. Array variables *All_Data[0]* to *All_Data[9]* with the INT data type are registered.
Like with Data Memory Area of a traditional PLC (e.g., CJ2), values of variables can be retained when power is turned OFF. Selecting the *Retain* Check Box.

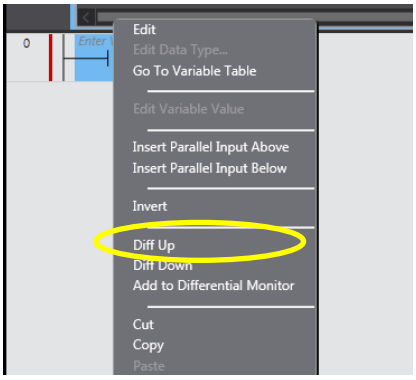


4. Right-click in the internal variable table and select ***Create New*** from the menu.
Register INT variable *n* that is the element number of *All_Data[**n**]*.
Enter “0” into the *Initial Value* Column in the variable table.

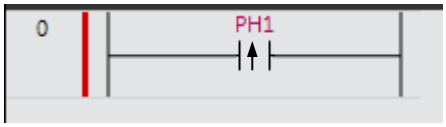


4-2-8 Entering Programming Code

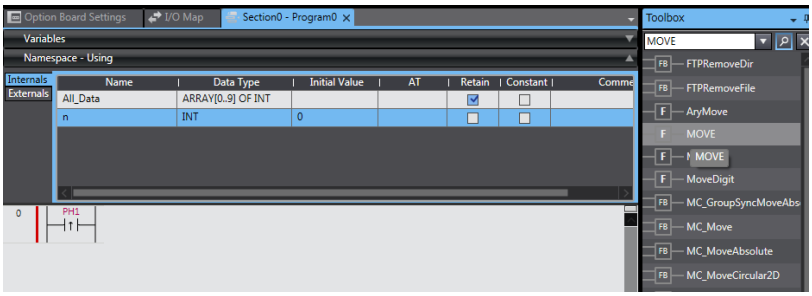
1. Insert input *PH1* and set upward differentiation. Select a connecting line and press the **C** Key. Right-click the input and select ***Diff Up*** from the menu or press the @ Key.



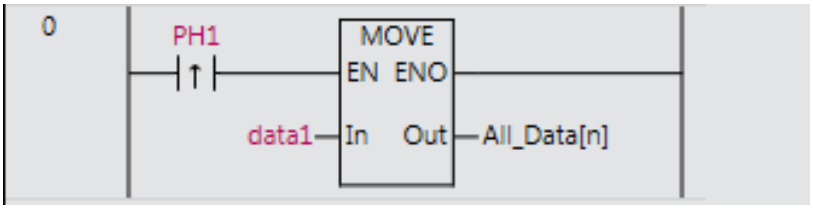
Click *Enter Variable* and enter “PH1”.



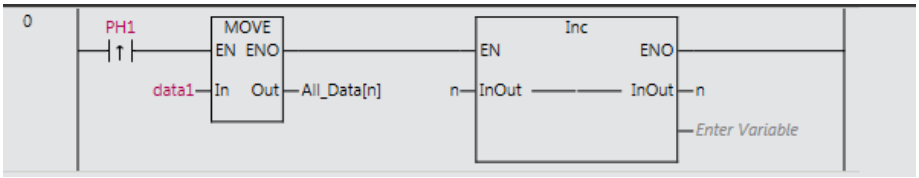
2. Add the MOVE instruction. Search for “MOVE” in the Toolbox. Add the MOVE function by dragging it from the Toolbox. (Or press the **I** Key and enter “MOVE”.)



3. Enter variables “data1” and “All_Data[n]” into the MOVE instruction.



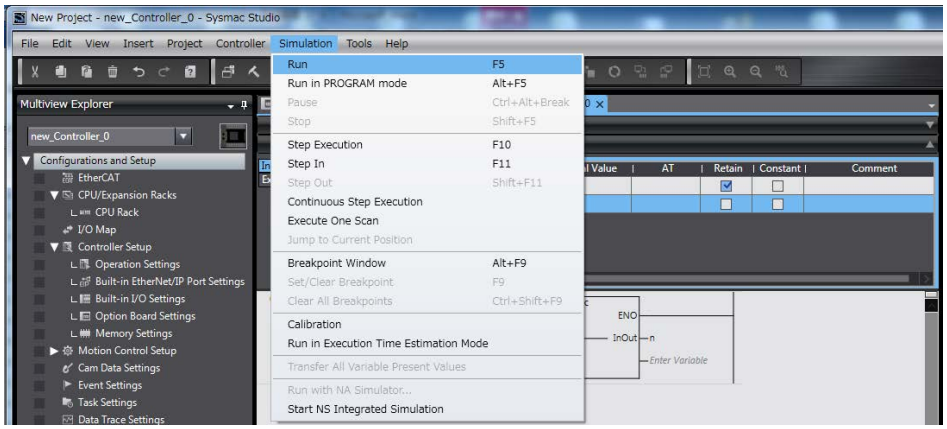
4. In the same way as the MOVE instruction, enter variable “n” into the Inc instruction.



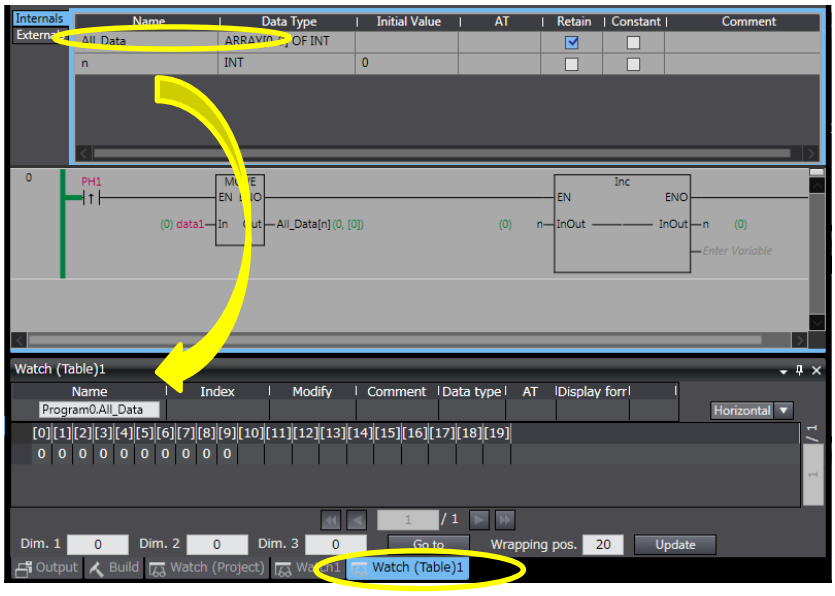
4-2-9 Checking the Operation of the Program

This section explains how to check the operation of the program on the Simulator.

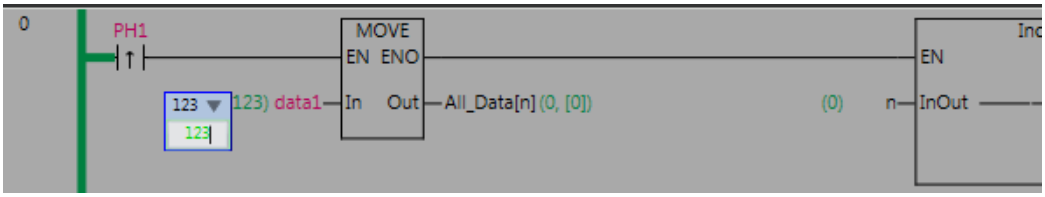
1. Select ***Run*** from the Simulation Menu to start the Simulator.



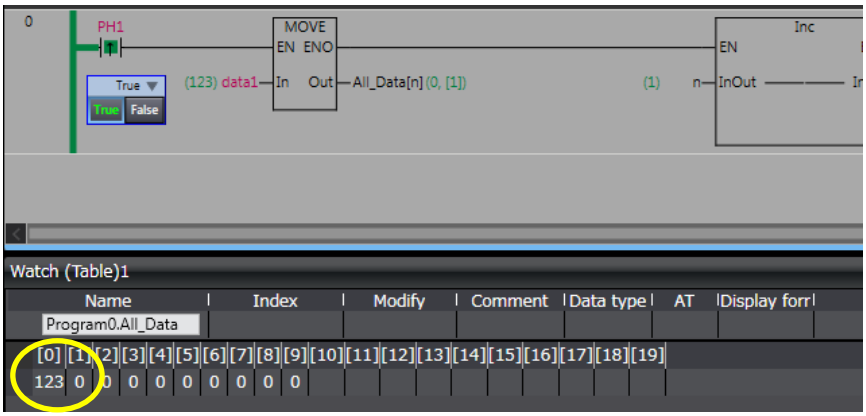
2. Select ***Watch Tab Page*** from the View Menu to display the Watch Tab Page (Table). Drag *All_Data* in the variable table to the Watch Tab Page.



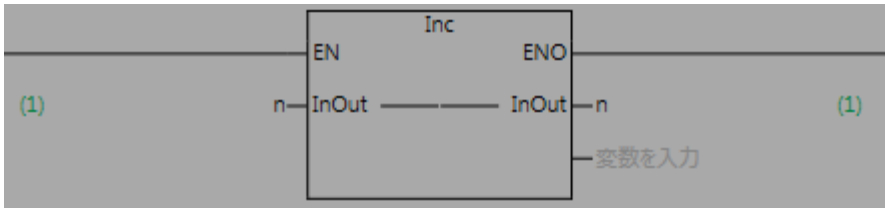
3. Double-click *data1* in the MOVE instruction to set the value.
Enter “123” and press the **Enter** Key.



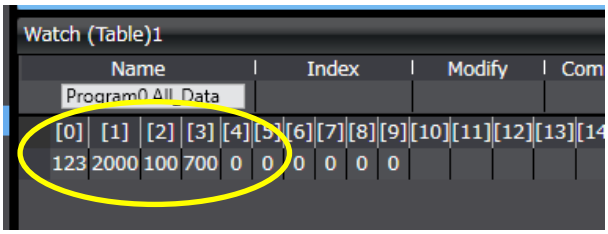
4. Press the **Enter** Key on N.O. input *PH1* to change the value between True (ON) and False (OFF).
Change the value of *PH1* to True (ON). “123” is stored in the value of *All_Data[0]* in the Watch Tab Page.



Check that *n* of the Inc instruction is incremented by 1 and now it is 1.



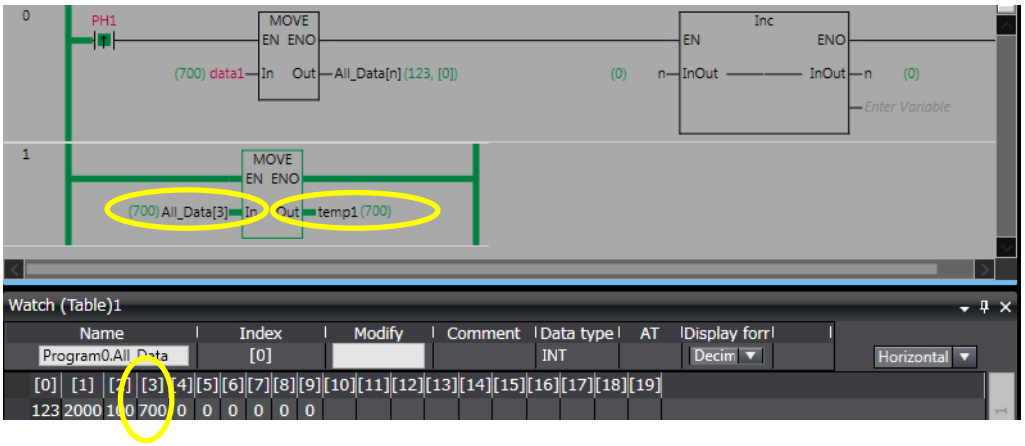
5. Enter any value in *data1* several times.



The values are set to array variable *All_Data* in order as shown above.

4-2-10 Referring Values of Array Variables

1. Use the following procedure to refer values of the registered array variables.
The figure below shows an example of the program to assign the value of All_Data[3] (the 4th value) to INT variable *temp1*.



2. Go offline before taking the next step. Save and export the project file.

5

5 Motion FB Programming

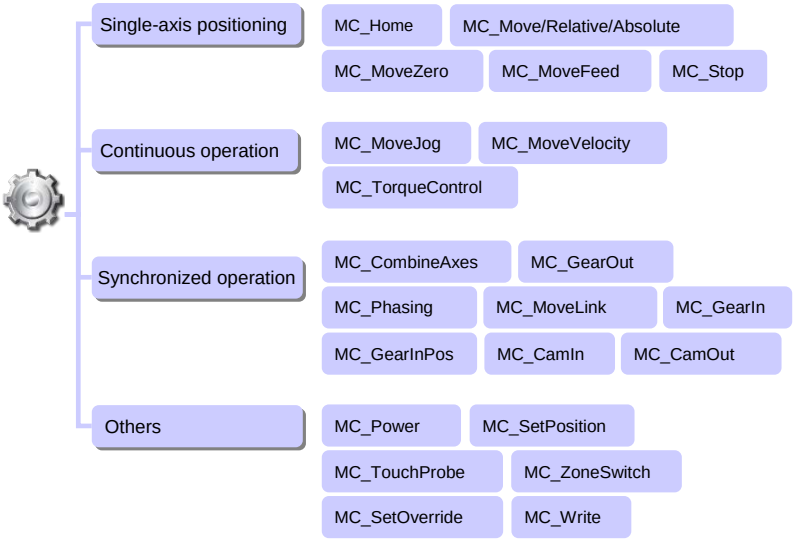
This section describes how to write programs using motion FBs.

5-1 Motion FB Programming 5-85
5-1-1 Motion FB Programming 5-85
5-1-2 Programming Procedure 5-85
5-2 Adding a Servo Drive and Setting the Parameters 5-86
5-2-1 Registering a Servo Drive 5-86
5-2-2 Registering the Axis 5-87
5-2-3 Setting the Axis Parameters 5-87
5-3 Creating a Program 5-89
5-3-1 Overview of the Ladder Program 5-89
5-3-2 Motion FBs to Use 5-89
5-3-3 Writing the Ladder Program 5-90
5-4 Data Tracing 5-93
5-4-1 Checking the Operation with Data Traces 5-93
5-5 3D Simulation 5-95
5-5-1 Starting 3D Simulation 5-95

5-1 Motion FB Programming

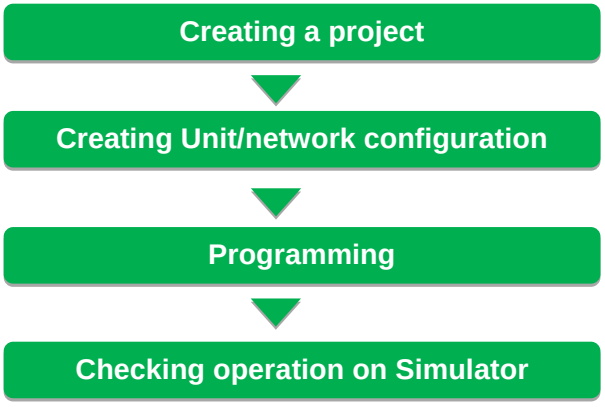
5-1-1 Motion FB Programming

This section explains how to create a program using PLCopen®-defined function blocks for motion control (hereafter called motion FBs).
The motion FBs listed below can be used for the NX1P. You can implement your desired motion control by combining the motion FBs.



5-1-2 Programming Procedure

Create a program to perform simple positioning by using motion FBs.
The Sysmac Studio allows you to debug programs and check motion in 3D on the Simulator, without using physical devices such as NX1P and Servomotors.



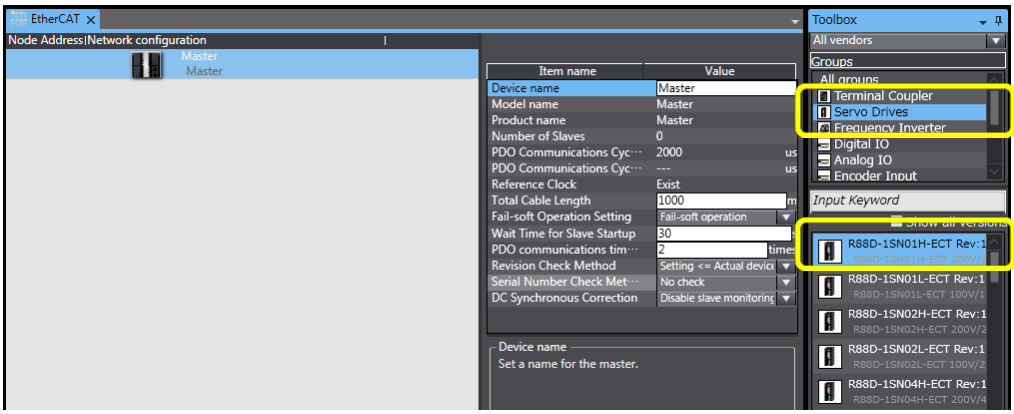
This Guide mainly explains how to use motion FBs. Although it is required to set up the Servomotor and Absolute Encoder in real applications, this Guide does not explain it.

5-2 Adding a Servo Drive and Setting the Parameters

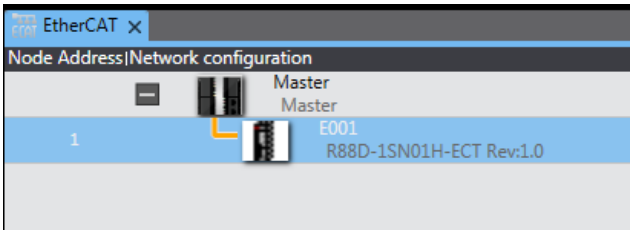
5-2-1 Registering a Servo Drive

1. After creating a project, double-click **EtherCAT** under **Configurations and Setup** in the Multiview Explorer to display the EtherCAT Tab Page.

2. Add a Servo Drive as the EtherCAT slave.
Click the **Servo Drives** in the Toolbox. The list of Servo Drives is displayed.
Double-click the Servo Drive to use. (Select *R88D-1S□* in this example.)

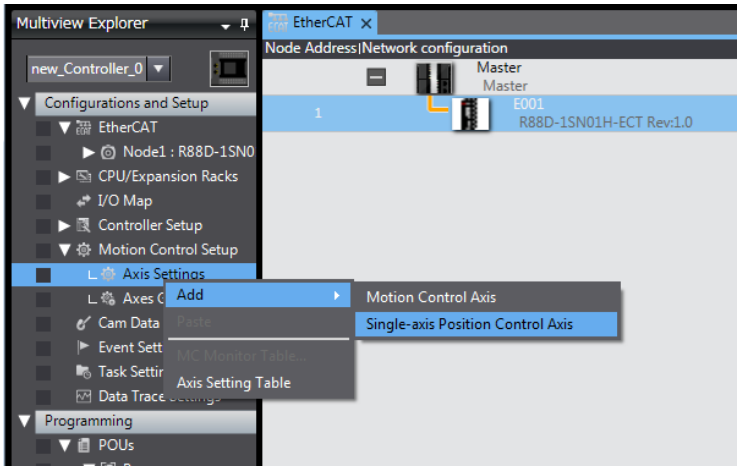


3. The Servo Drive is added under the master.

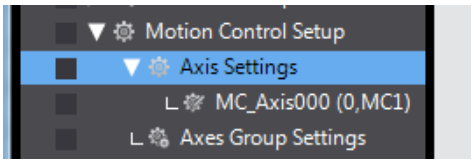


5-2-2 Registering the Axis

1. Register the axis to perform motion control. Right-click **Axis Settings** under **Configurations and Setups − Motion Control Setup** and select ***Add − Single-axis Position Control Axis*** from the menu.



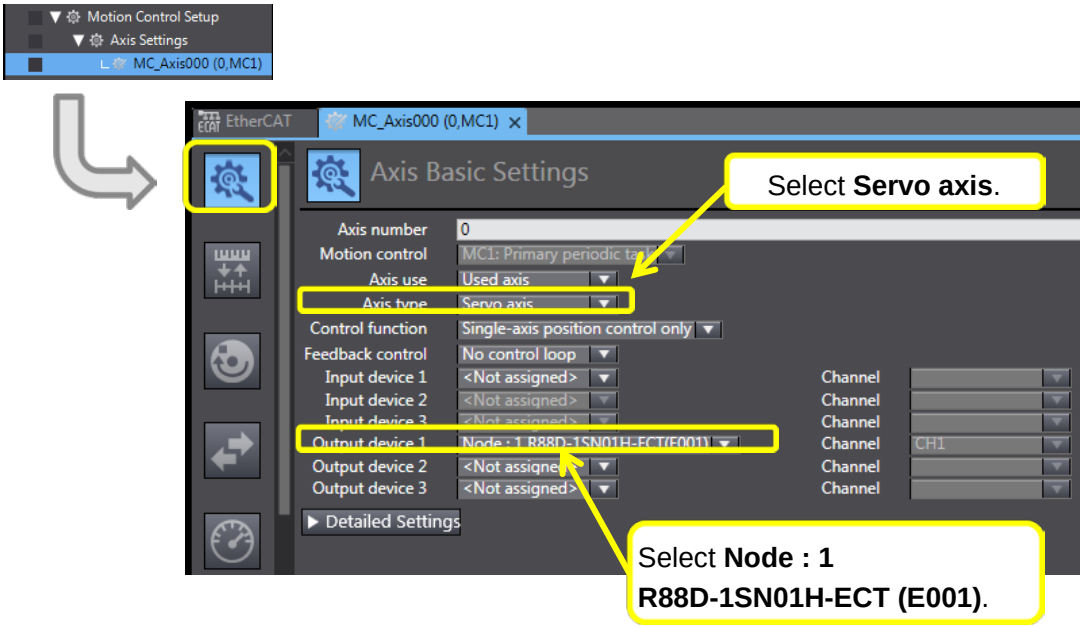
2. The axis *MC_Axis000(0)* is added as shown below.



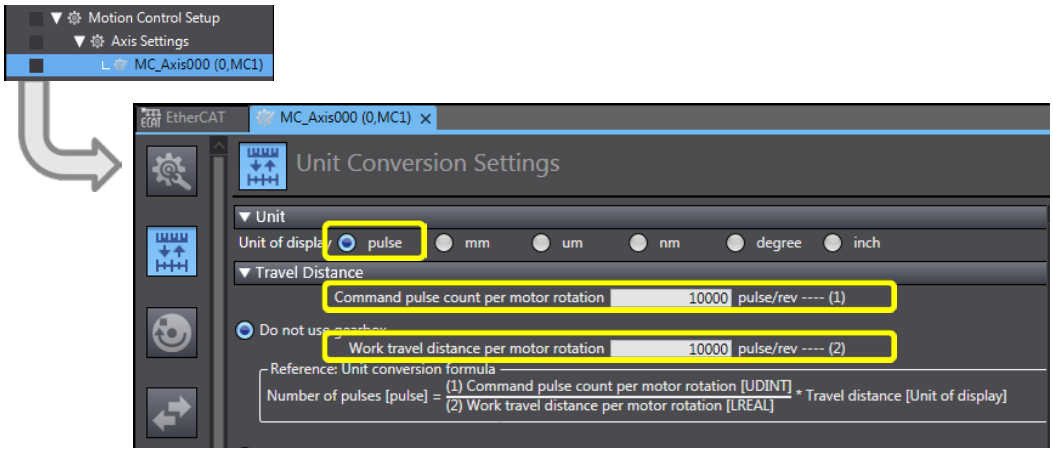
5-2-3 Setting the Axis Parameters

Double-click **MC_Axis000(0)**. The Axis Parameter Settings Tab Page is displayed. Set the parameters used in this exercise.

1. Set the parameters in the Axis Parameter Settings Tab Page.

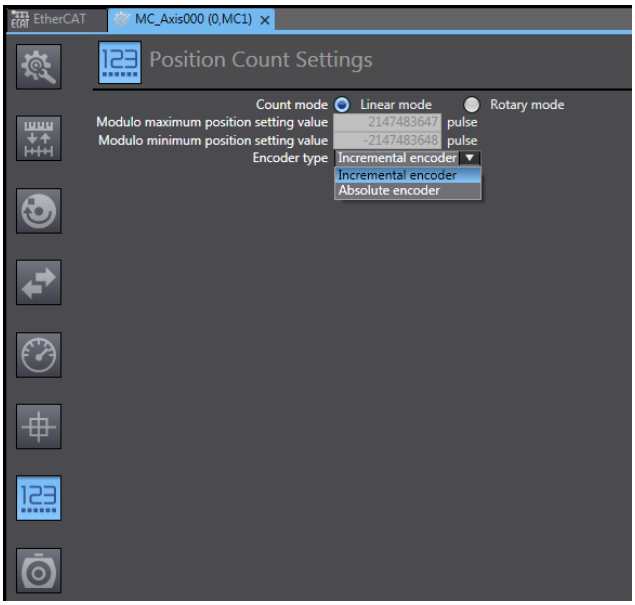


2. Click the **Unit Conversion Settings** Button and check that settings are the same as those shown below (default settings).



Additional Information

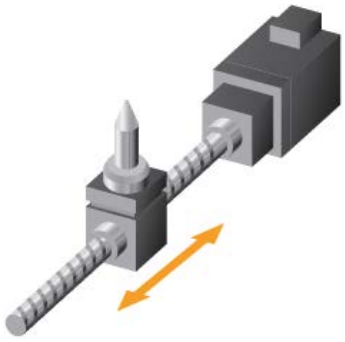
Although the 1S-series AC Servo System has a built-in absolute encoder, default incremental encoder settings are used in this exercise.
When using as an absolute encoder, select absolute encoder in the Position Count Settings Tab Page.



5-3 Creating a Program

5-3-1 Overview of the Ladder Program

Create a program using single-axis motion FBs to move a ball screw forward and backward as shown below.



5-3-2 Motion FBs to Use

Three motion FBs are used for this exercise.

Instruction	Name	FB/FUN	Graphic expression
MC_Power	Power Servo	FB	MC_Power_instance MC_Power Axis — — Axis Enable — Status — Busy — Error — ErrorID

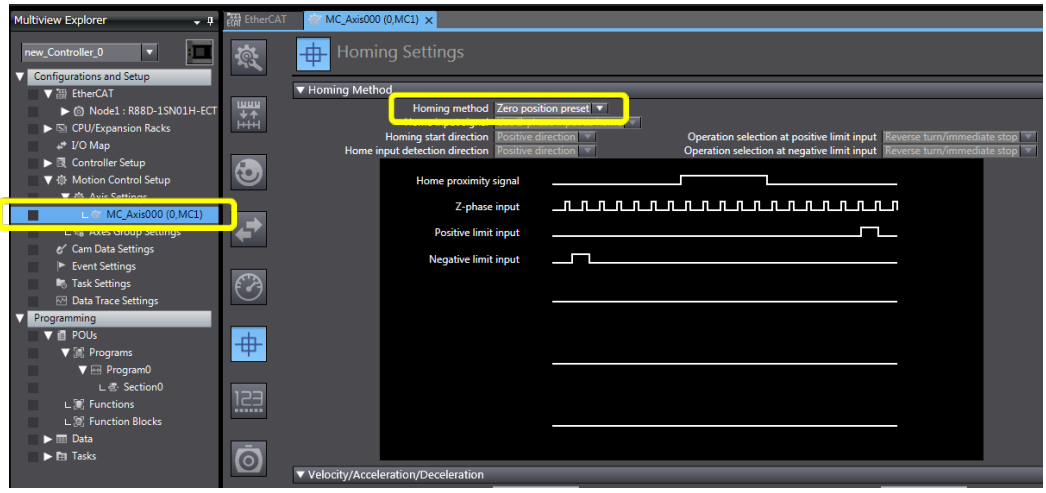
Instruction	Name	FB/FUN	Graphic expression
MC_Home	Home	FB	MC_Home_instance MC_Home Axis — — Axis Execute — Done — Busy — CommandAborted — Error — ErrorID

Instruction	Name	FB/FUN	Graphic expression
MC_MoveRelative	Relative Positioning	FB	MC_MoveRelative_instance MC_MoveRelative Axis — — Axis Execute — Done Distance — Busy Velocity — Active Acceleration — CommandAborted Deceleration — Error Jerk — ErrorID BufferMode

Additional Information

MC_Home is the Home instruction. Set the homing operation in the Homing Settings Tab Page that is displayed by double-clicking **Axis Settings - MC_Axis000**. Zero position preset (default) is used for this exercise.

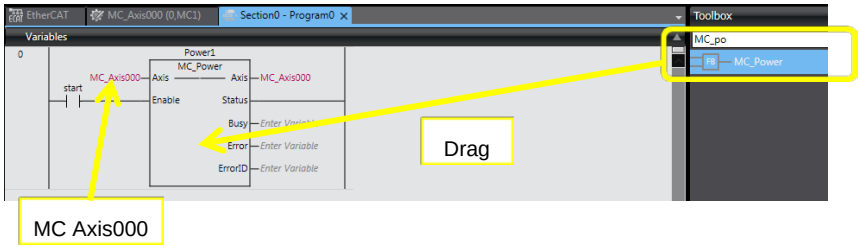
Zero position preset: The present value becomes 0 when the MC_Home instruction is executed



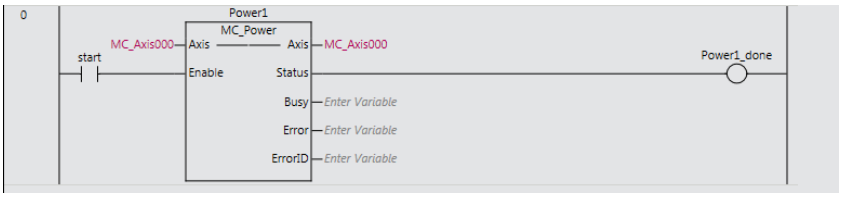
5-3-3 Writing the Ladder Program

1. Start the Sysmac Studio and create a project.
Double-click **Section0** under **Programming - POUs - Programs - Program0** in the Multiview Explorer to open the Ladder Editor.

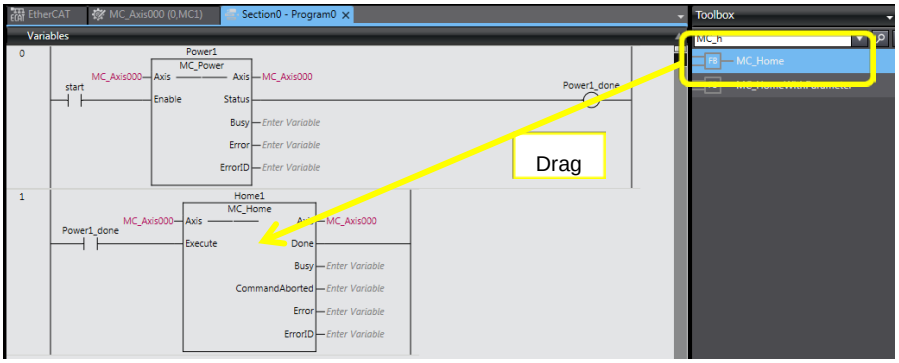
2. Insert N.O. Input *start* in the first rung.
Insert the MC_Power motion FB and enter “Power1” as the instance name.
(Search for “MC_Power” and drag the motion FB from the Toolbox.)
Enter “MC_Axis000” into the parameter for *Axis*.



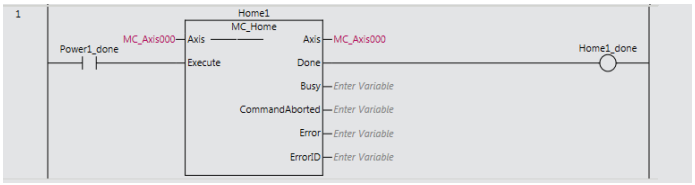
3. Insert output *power1_done* (or any other name you prefer).



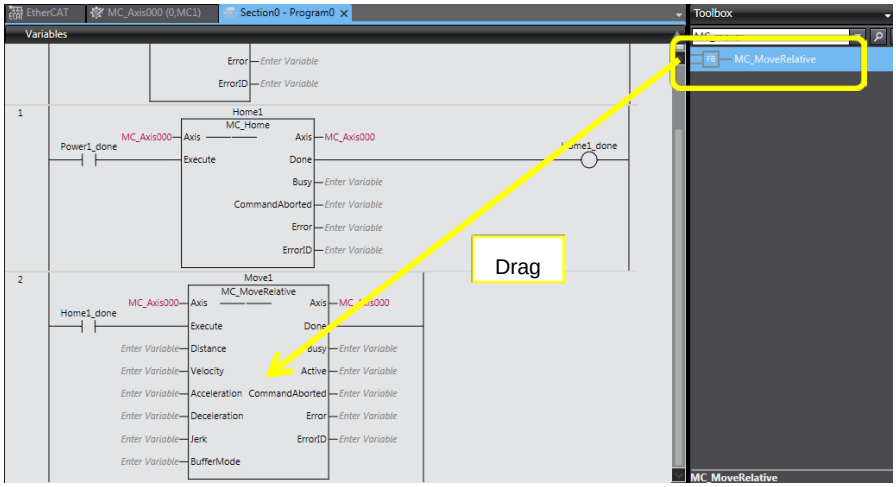
4. Press the **R** Key to insert a rung below the first rung.
Insert N.O. Input *power1_done* and then insert the MC_Home motion FB.
Enter “Home1” as the instance name.



5. Insert output *home1_done* (or any other name you prefer).



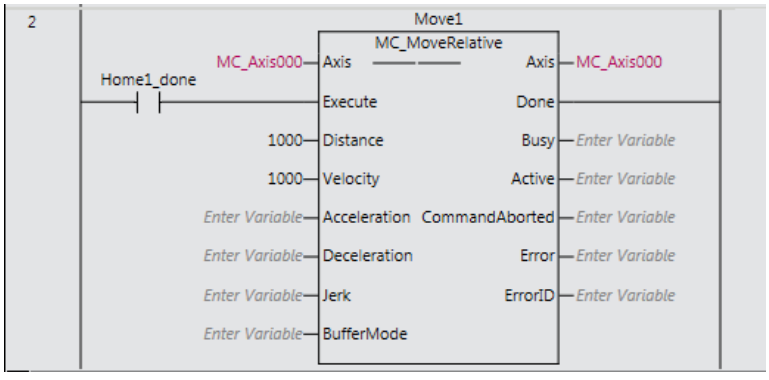
6. Press the **R** Key to insert a rung below the second rung.
Insert N.O. Input *home1_done* and then insert the MC_MoveRelative motion FB.
Enter “Move1” as the instance name.



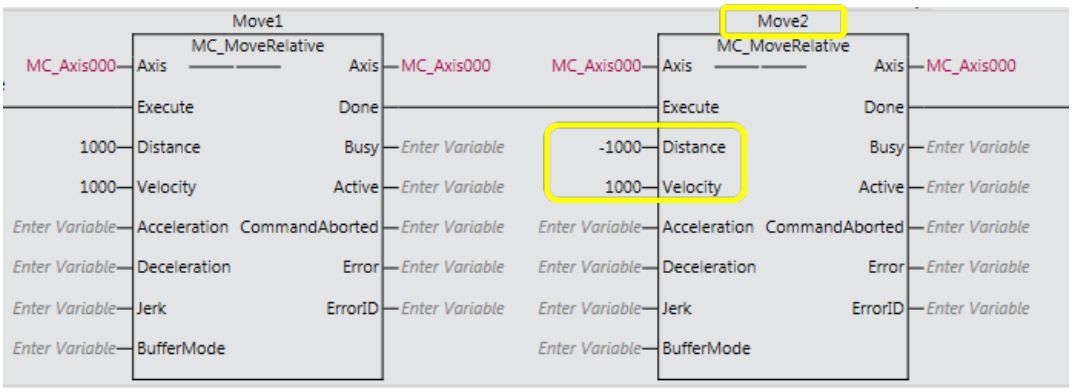
7. Set the parameters as follows. (You do not need to set other parameters in this exercise.)

- *Distance*: 1000 (pulses)
- *Velocity* : 1000 (pulses/s)

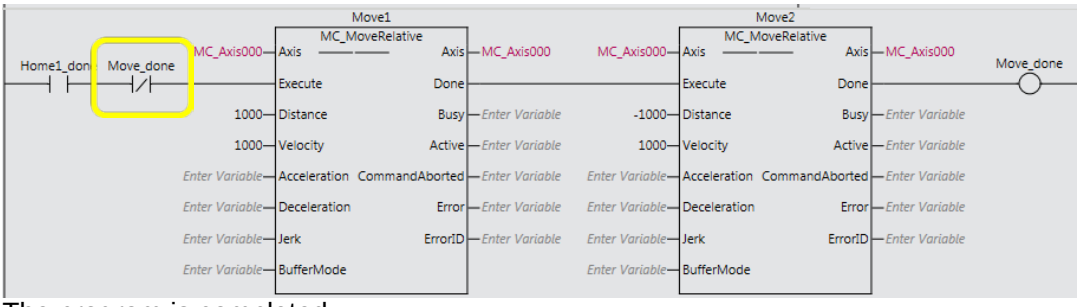
These settings move the ball screw the set distance in a second.



8. In the same way as step 6 and 7, insert another MC_MoveRelative motion FB to move the ball screw backward at the same velocity. Enter “Move2” as the instance name. Set the parameter for *Distance* to -1000 to move backward.

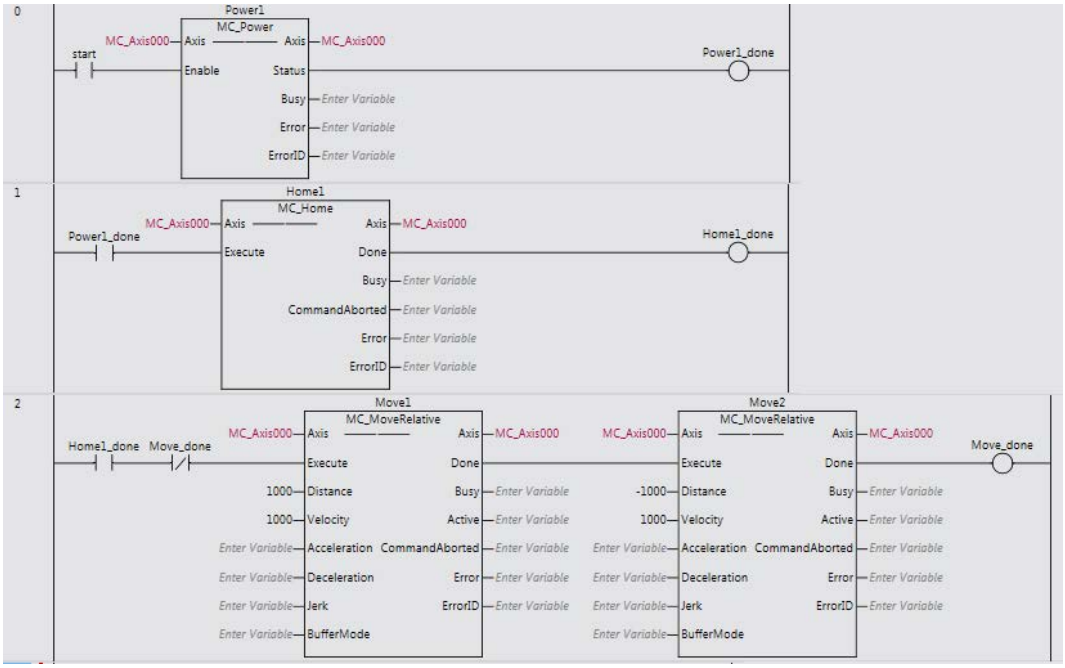


9. Insert output *move_done* as shown below.



The program is completed.

■ Completed program

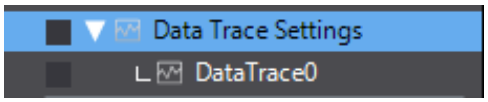


5-4 Data Tracing

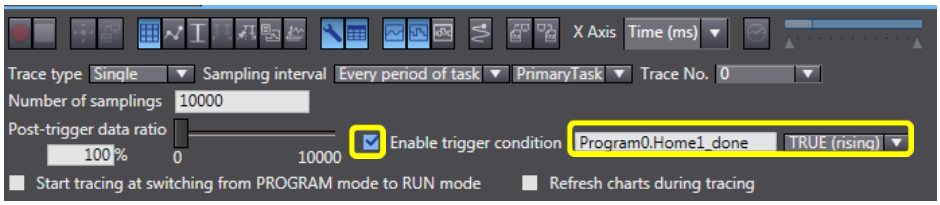
5-4-1 Checking the Operation with Data Traces

Use the traced data to check the positions during single-axis motion control.

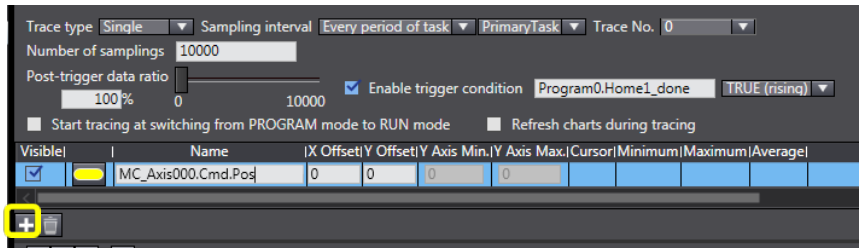
1. Right-click **Data Trace Settings** under **Configurations and Setup** in the Multiview Explorer and select ***Add − Data Trace*** from the menu.
DataTrace0 is added.



2. Double-click **DataTrace0** to make settings. Select the *Enable trigger condition* Check Box and enter “Program0” and “.” (dot). A list of possible candidates is displayed. Select *home1_done* and *TRUE (rising)* to set the execution condition of the motion FB.

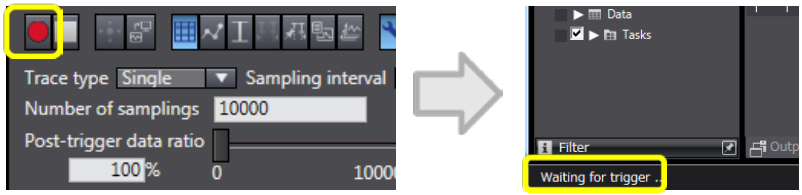


3. Set the variable to trace.
Click the **Add Target** Button (+) and specify MC_Axis000.Cmd.Pos.
Enter “MC_Axis000” and “.“ (dot). A list of possible candidates is displayed. Select *Cmd* (command value) and *Pos* (position).

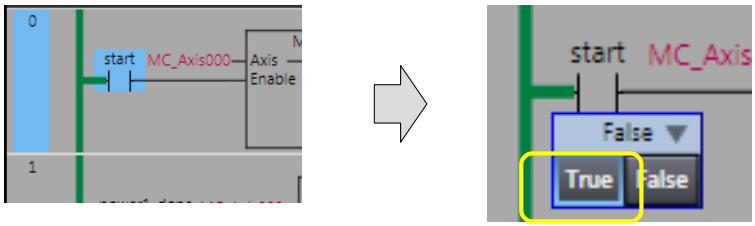


4. Select ***Run*** from the Simulation Menu. The program can be debugged without connecting the NX1P physically.
Simulation starts and the color of the top of the Edit Pane changes to yellow green.

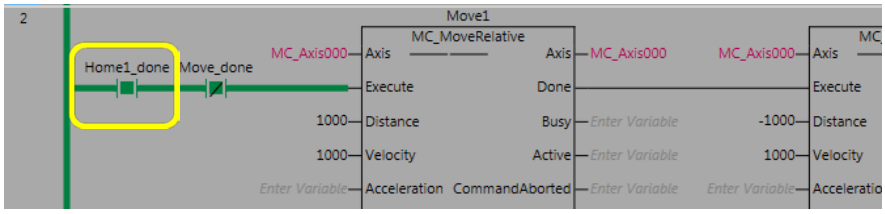
5. Click the **Execute** Button (red button) to start a trace.
A “Waiting for trigger ...” message appears on the status bar at the lower left of the window.



6. Double-click **Section0** to open the Ladder Editor.
Use the ***Set*** and ***Reset*** menu commands to change program inputs and outputs in the Ladder Editor to TRUE or FALSE.
Double-click *start* in the first rung and select **True**.

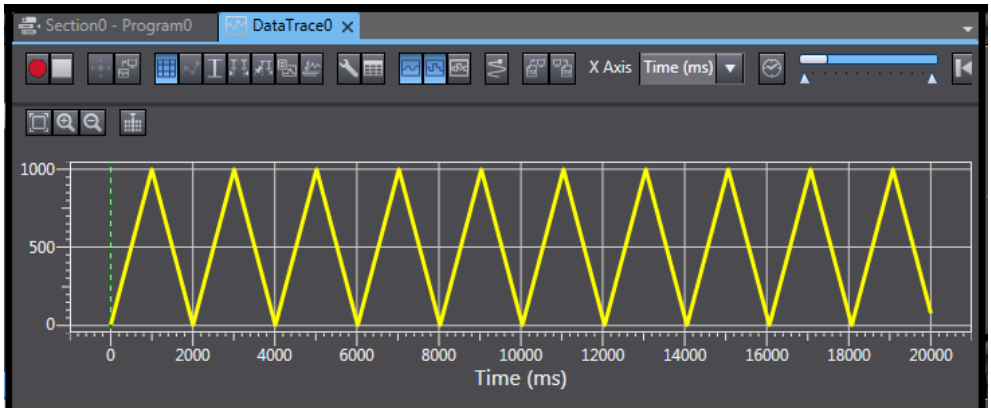


7. When *start* changes to True, the trigger condition *home1_done* changes to TRUE and the tracing starts.



The progress of tracing is displayed in a light blue progress bar at the lower right of the window.
When the bar disappears, the tracing is completed.

8. Double-click **DataTrace0** under **Configurations and Setup - Data Trace Setting**.
Check that position data is traced as shown below.
This graph shows that the Servomotor moves forward and backward every second. (The X axis represents time and the Y axis represents travel distance (pulse).)



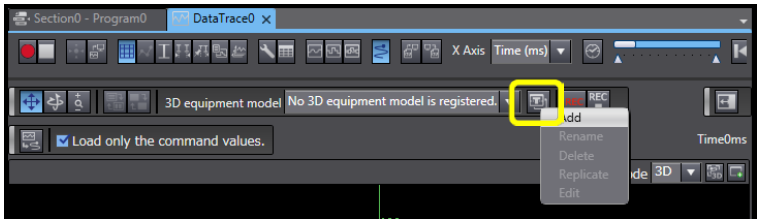
You can adjust the screen layout by using the icons shown below to display and hide items.

5-5 3D Simulation

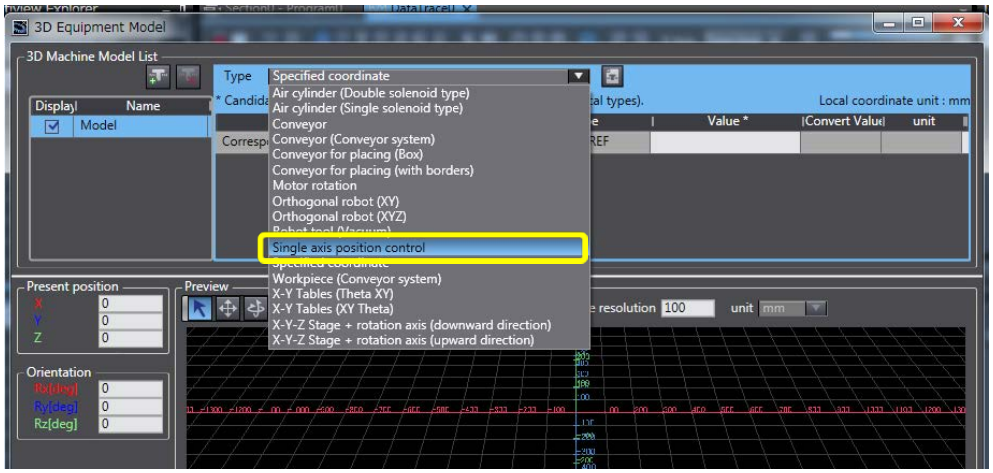
5-5-1 Starting 3D Simulation

1. Click the **Display 3D Motion Monitor** Button shown below. Close unnecessary windows.

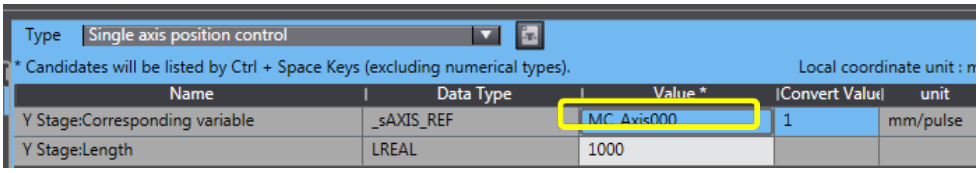
2. Click the **Settings** Button shown below and select **Add** from the menu.



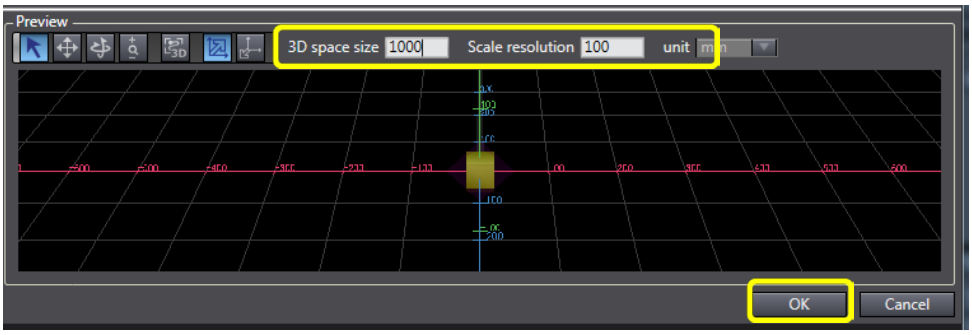
3. Select *Single axis position control* for the *Type* parameter in the 3D Machine Model List.



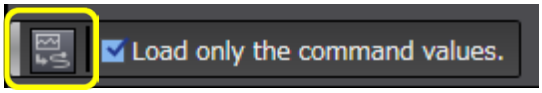
4. Enter “MC_Axis000” into the *Value* Column of Y Stage: Corresponding variable.



5. Enter “1000” into the *3D space size* Box and “100” into the *Scale resolution* Box, and then click the **OK** Button.



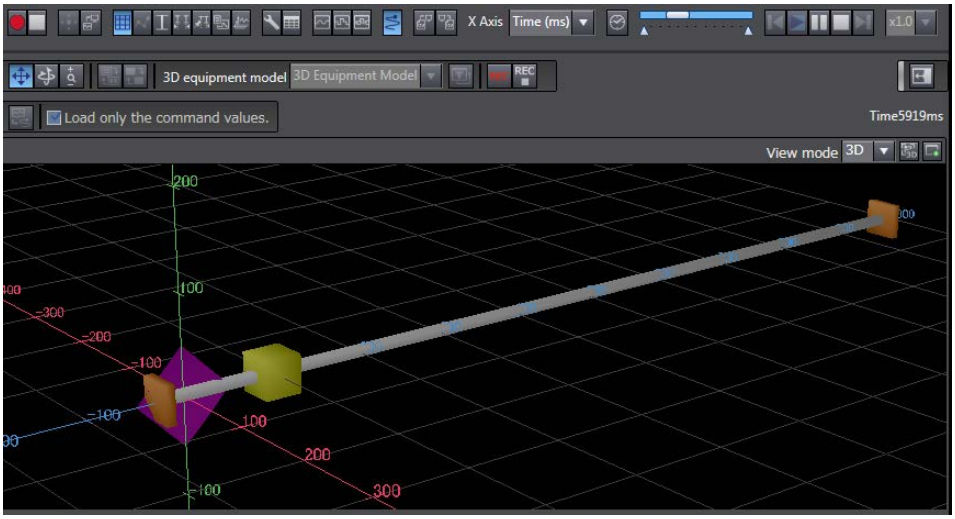
6. Click the **Trace Data Loading** Button shown below to load the traced data.



7. Check the 3D equipment motion by using the buttons shown below.

8. Use the three viewpoint operation buttons shown below to change and rotate your viewpoint and zoom in and out of the 3D display area.

9. 3D simulation debugging is completed.



10. Go offline before taking the next step. Save and export the project file.

6

6 ST Programming

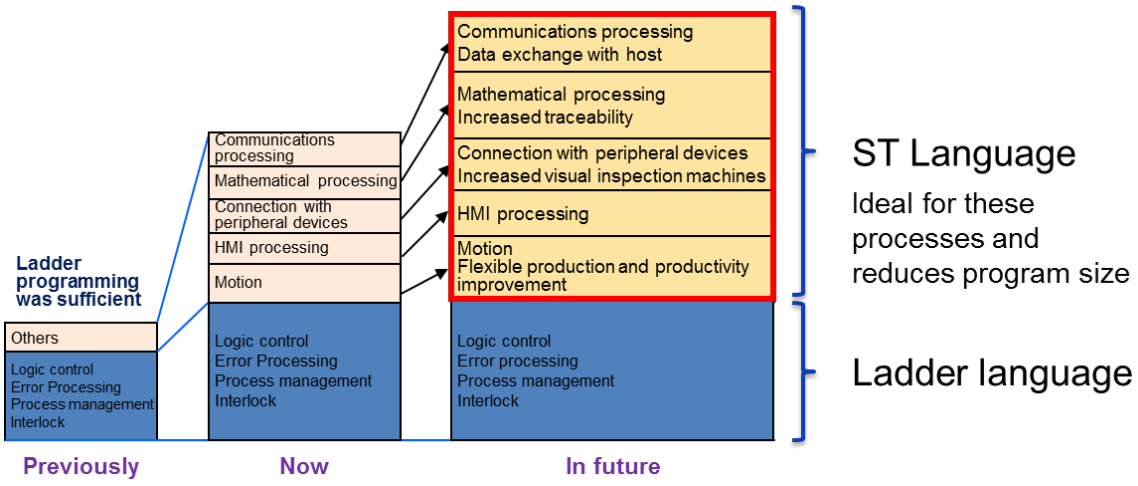
This section describes how to create ST programs.

6-1 Overview of ST Programming 6-98
6-1-1 Advantages of ST Language 6-98
6-1-2 ST Programs Including Constructs 6-98
6-1-3 Structure of ST and Example 6-99
6-1-4 Operators 6-99
6-2 NX1P Programming in ST 6-100
6-2-1 Writing an ST Program for NX1P 6-100
6-3 ST Programming Exercise 6-101
6-3-1 Exercise of Numerical Calculation Programming 6-101
6-3-2 Programming Procedures 6-102
6-3-3 Checking the Program 6-104
6-3-4 Checking the Operation of the ST Program 6-104

6-1 Overview of ST Programming

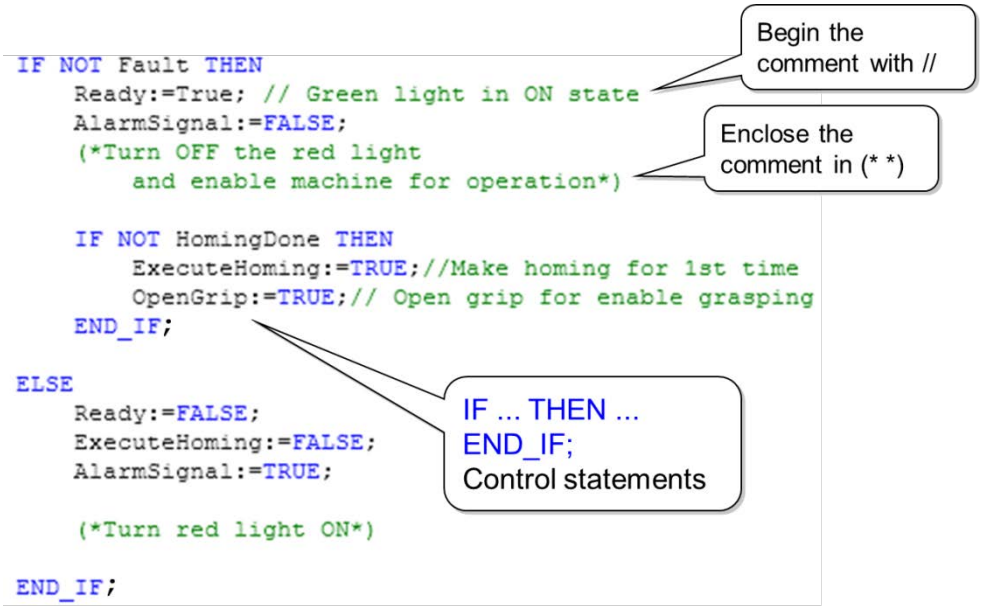
6-1-1 Advantages of ST Language

Machine control programs are becoming larger in size and more complicated. The percentage of status control and interlock control that can be programmed easily in ladder diagrams is decreasing. On the other hand, complex mathematical processing and data storage that are difficult to program in ladder diagrams account for about 70% of an entire program. The use of ST for this part makes programming easier and reduces program size.



6-1-2 ST Programs Including Constructs

Branch and loop statements such as “IF” and “FOR” can be used in ST programming, like BASIC and C programming.
Programs including mathematical processing and control statements, which are difficult to write in ladder diagrams, can be created easily.



6-1-3 Structure of ST and Example

Using general expressions, ST programming requires no special knowledge.
Remember the following two rules:

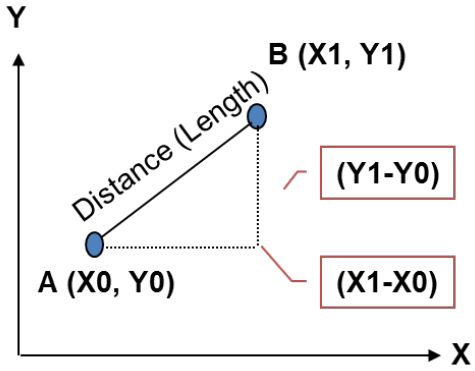
(1) Use a colon and an equals sign (:=) to assign a value to a variable.
(2) Statements must end with a semicolon (;).

Example

An example of the statement to calculate the distance between two points using Pythagoras' theorem is shown below.
Just apply the formula.

■ Formula

$$\mathbf{Length} = \sqrt{(\mathbf{X1} - \mathbf{X0})^2 + (\mathbf{Y1} - \mathbf{Y0})^2}$$



■ ST program

```
Length := SQRT((X1-X0)**2+(Y1-Y0)**2);
```

* SQRT: Square Root

6-1-4 Operators

Operator	Operation	Notation example		Priority
()	First priority	Value := (1+2)*(3+4);	// Value is 21	1
-, +	Sign	+100, -100		2
NOT	Logical NOT	Value := NOT TRUE;	// Value is FALSE	
**	Exponent	Value := 2**8;	// Value is 256	3
*	Multiplication	Value := 8*100;	// Value is 800	4
/	Division	Value := 200/25;	// Value is 8	
MOD	Remainder	Value := 10 MOD 6;	// Value is 4	
+	Addition	Value := 200+25;	// Value is 225	5
-	Subtraction	Value := 200-25;	// Value is 175	
<, >, <=, >=	Comparison	Value := 60>10;	// Value is TRUE	6
=	Matches	Value := 8=7;	// Value is FALSE	7
<>	Does not match	Value := 8<>7;	// Value is TRUE	
&, AND	Logical AND	Value := 2#1001 AND 2#1100;	// Value is 2#1000	8
XOR	Logical exclusive OR	Value := 2#1001 XOR 2#1100;	// Value is 2#0101	9
OR	Logical OR	Value := 2#1001 XOR 2#1100;	// Value is 2#1101	10

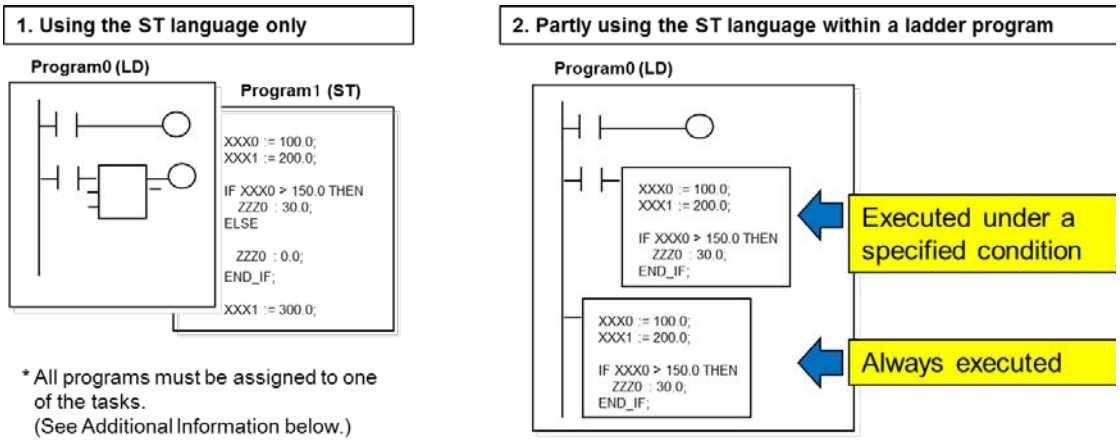
6-2 NX1P Programming in ST

6-2-1 Writing an ST Program for NX1P

You can create an ST program for the NX1P in two ways:

1. Using the ST language only
2. Partly using the ST language within a ladder program (inline ST)

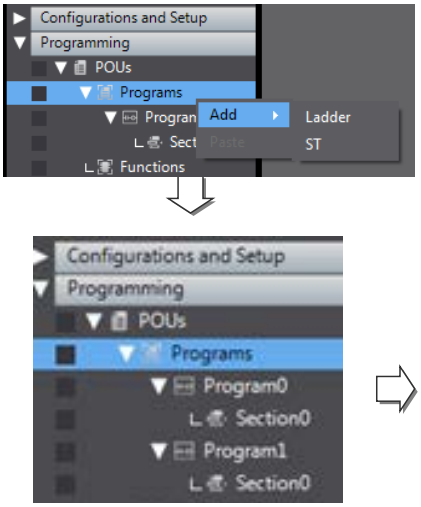
In the second method, the ST program can be executed under a specified condition (e.g., when an input changes to TRUE) or can always be executed.



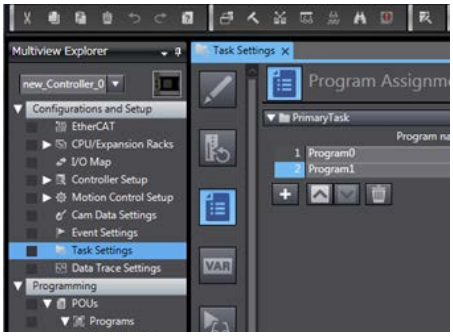
Additional Information

Adding a program and assigning the program to a task

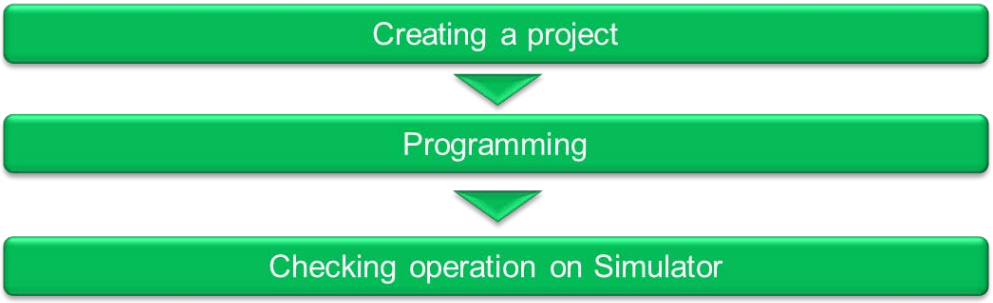
(1) Right-click **Programs** and select ***Add − Ladder*** or ***Add − ST*** from the menu.



(2) Double-click **Task Settings** under **Configurations and Setup** in the Multiview Explorer and then click the **Program Assignment Settings** Button. Select the program to assign to Primary Task and set the initial status.



6-3 ST Programming Exercise



6-3-1 Exercise of Numerical Calculation Programming

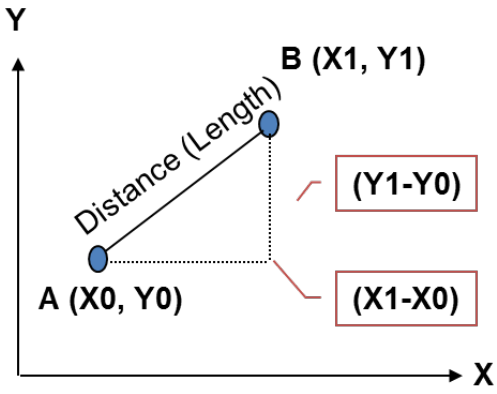
Create a ladder program including inline ST to calculate the distance between two points.

Exercise

Create a program to execute the ST code to calculate the distance between two points when the Switch SW3 is pressed.

[Pythagoras' theorem]

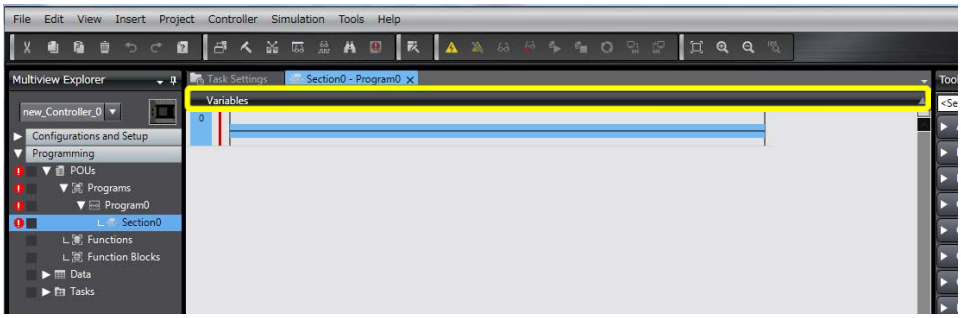
$$\mathbf{Length} = \sqrt{(\mathbf{X1} - \mathbf{X0})^2 + (\mathbf{Y1} - \mathbf{Y0})^2}$$



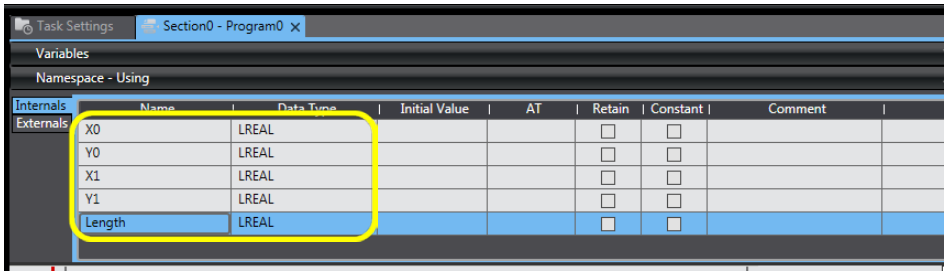
6-3-2 Programming Procedures

1. Create a project. Double-click **Section0** under **Programming - POUs - Programs - Program0** in the Multiview Explorer to open the Ladder Editor.

2. Click the **Variables** Bar to display the variable table.



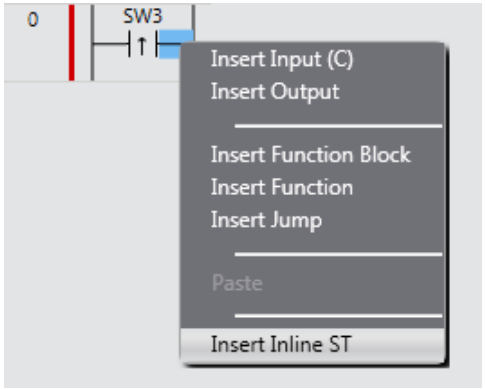
3. Register the variables used for inline ST as internal variables in the variable table.



Press the **Insert** Key to insert a row.
LREAL (double-precision floating-point) data enables precise decimal point calculation.

4. Insert a rung and N.O. Input *SW3*. Set upward differentiation. (Press the @ Key on the N.O. input.)

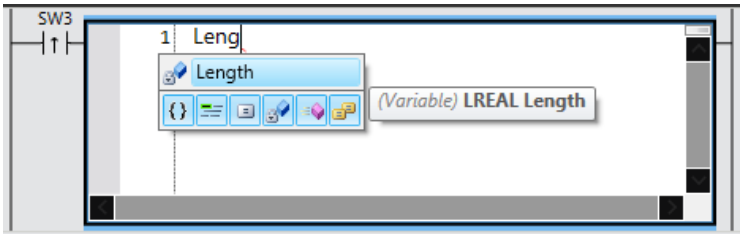
5. Right-click the connecting line as shown below and select ***Insert Inline ST*** from the menu.



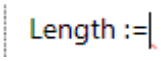
6. An inline ST box is inserted. Write the following ST code in this box:

Length := SQRT((X1-X0)**2.0+(Y1-Y0)**2.0);

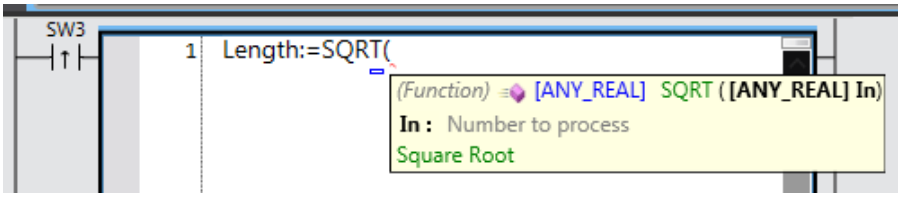
7. Enter “Leng” of variable *Length*. A list of possible candidates is displayed. Select *Length* and press the **Enter** Key.



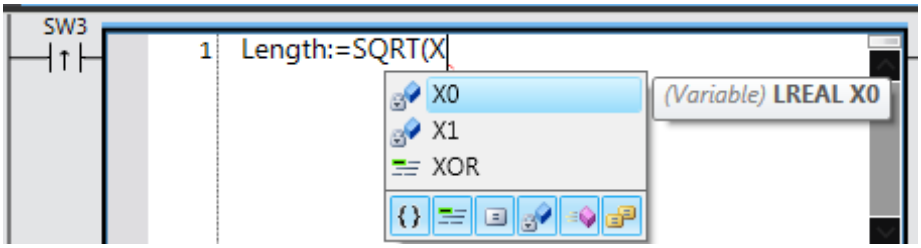
8. Enter a space and “:” (colon). The assignment keyword *:=* is entered automatically.



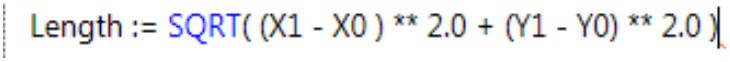
9. Enter “SQRT” (Square Root) and “(“ (left parenthesis). A description of the parameters is displayed.



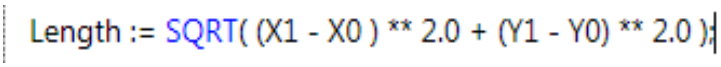
10. Enter “x”. A list of possible candidates is displayed. Select *X1* and press the **Enter** Key.



Enter the statement to the end.

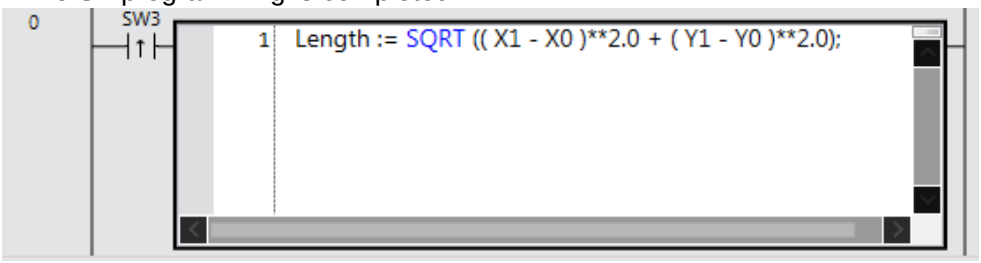


Enter “;” (semicolon).

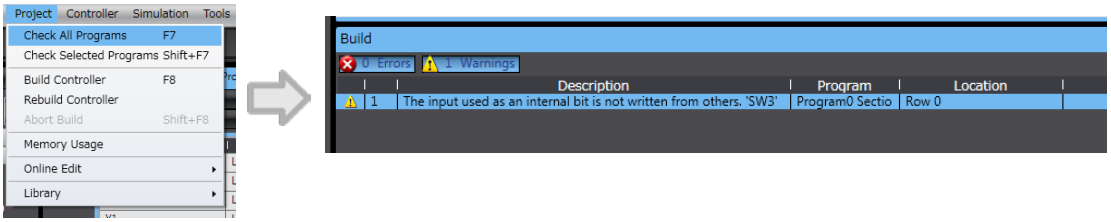


6-3-3 Checking the Program

Inline ST programming is completed.

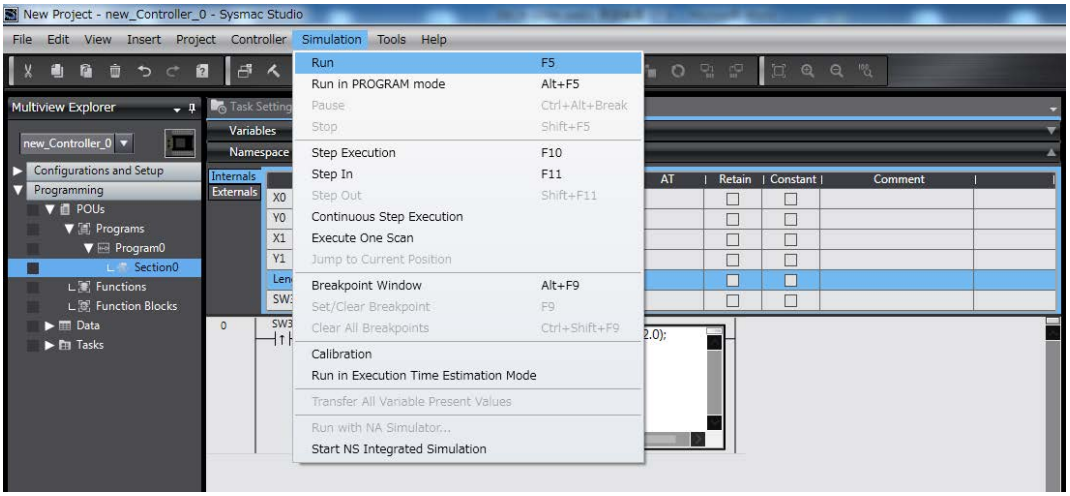


Select ***Check All Programs*** from the Project Menu to confirm that there is no error. (Ignore a warning.)



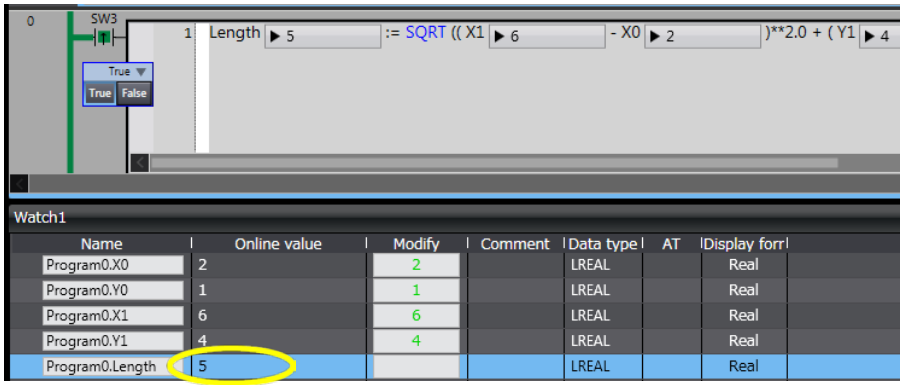
6-3-4 Checking the Operation of the ST Program

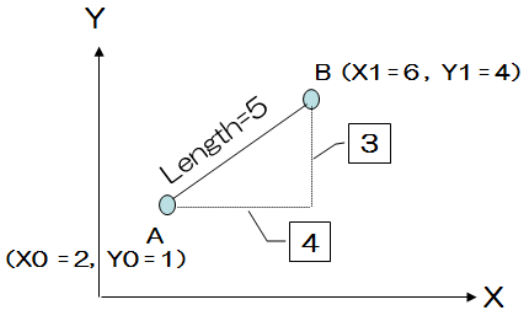
1. Start the Simulator to check the operation.
Select ***Run*** from the Simulation Menu.



2. Check the operation.

Synchronize (download) the program. Assign any value to variables of two points (X0, Y0, X1, Y1), and then press SW3. Check that the calculation result is displayed in the *Online value* Column of *Length*.



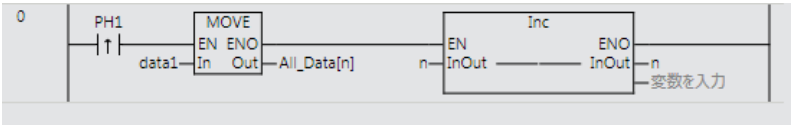


3. Go offline. Save and export the project file.

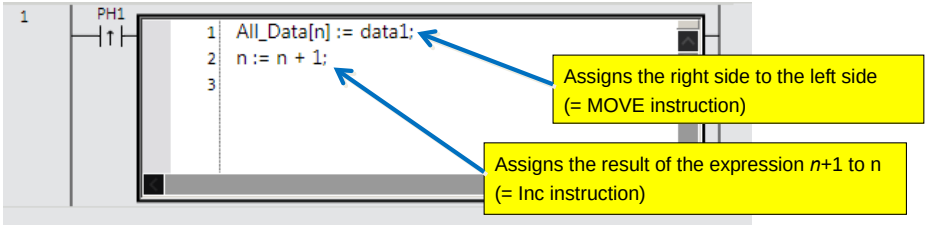
Additional Information

Write the code created in *4 Creating Programs to Handle Data* in ST.
Instead of using the MOVE and Inc instructions, you can use ST language that is a BASIC or C-like language.

- Ladder programming



- ST programming (inline ST)



OMRON Corporation **Industrial Automation Company**
Tokyo, JAPAN
Contact: www.ia.omron.com

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

© OMRON Corporation 2018 All Rights Reserved.
In the interest of product improvement,
specifications are subject to change without notice.

Cat. No. P122-E1-02 1218 (0917)