Panasonic

Motion Controller

GM1 Controller EtherCAT

User's Manual

Operation

WUME-GM1ETCOP-03

2022.4

(MEMO)

Introduction

Thank you for purchasing a Panasonic product. Before you use the product, please carefully read through the user's manual, and understand it in detail to use the product properly.

Types of Manual

- There are different types of manuals for the GM1 series, as listed below. Refer to the appropriate manual according to your need.
  These manuals can be downloaded from our website: https://industrial.panasonic.com/ac/e/motor/motion-controller/mc/gm1/index.jsp

Manuals for GM1 series

Manual name	Manual code	Description
GM1 Controller EtherCAT User's Manual (Setup Edition)	WUME-GM1ETCSU	Explains wiring between the GM1 and its peripheral devices, installation method, and operation check method.
GM1 Controller EtherCAT User's Manual (Operation Edition)	WUME-GM1ETCOP	Explains how to use GM Programmer and PANATERM Lite for GM, set up each function, create projects, and perform other operations.
GM1 Series Reference Manual (Hardware Edition)	WUME-GM1H	Explains the functions and performance of each GM1 unit.
GM1 Series Reference Manual (Instruction Edition)	WUME-GM1PGR	Explains the specifications of each instruction that can be used with the GM1 Series.
GM1 Series Reference Manual (Analog I/O Unit)	WUME-GM1AIO	Explains the functions and performance of each GM1 Analog Expansion Unit.
GM1 Series Reference Manual (Pulse Output Unit)	WUME-GM1PG	Explains the functions and performance of each GM1 Pulse Output unit.

Copyright / Trademarks

Compliance with IEC 61131

International standard IEC 61131 is the international common standards in which the International Electrotechnical Commission (IEC) defines programming languages for PLC.

IEC defines the following five programming languages:

- Ladder Diagram (LD)
- Structured Text (ST)

- Sequential Function chart (SFC)
- Function Block Diagram (FBD)
- Instruction List (IL)

Glossary

GM Programmer

A configuration tool for the GM1 controller. Using GM Programmer makes it possible to set positioning data and various positioning parameters, and perform various monitoring. As this tool is equipped with tool operation mode that starts a motor independently without using user programs, it is convenient especially to verify operations at the time of initial startup.

PANATERM Lite for GM

A setup support tool for the MINAS series servo amplifiers manufactured by Panasonic Corporation. When GM Programmer is installed, "PANATERM Lite for GM" is also installed at the same time. By using this tool, parameter setup within servo amplifiers, control status monitoring, setup support, machine analysis, and other operations can be executed on the PC screen.

P-point control

Refers to control passing through a "Pass Point". In this manual, this control is referred to as "P-point control" for the sake of convenience.

This method is used when target multi-stage velocities are specified in a sequence of motions.

C-point control

Refers to control passing through a "Continuance Point". In this manual, this control is referred to as "C-point control" for the sake of convenience. This method is used to execute consecutive E-point controls by one-time startup.

E-point control

Refers to movement up to an "End Point". In this manual, this control is referred to as "E-point control" for the sake of convenience. This method is used for single-speed acceleration / deceleration control. It is also called "trapezoidal control".

Automatic operation

An operation that is automatically performed. It means position control.

Manual operation

An operation that is performed at initial startup or during adjustment. Home return, JOG operation, and pulser operation are manual operations.

Position control

A generic term for E-point control, P-point control, and C-point control. For each control, control for single axes and interpolation control for multiple axes can be performed. Interpolation control can be selected from 2-axis linear interpolation, 2-axis circular interpolation, and 3-axis linear interpolation.

Home return

The reference position for positioning is called a home position and an operation to travel to a home position is called home return. Each axis is moved to the preset home position and the coordinates of the home position are defined as absolute position zero. The motor rotation is

reversed automatically when the limit input (+) or the limit input (-) is input and the home position or near home position is searched to return to the home position automatically.

■ JOG operation

Refers to an operation in which the motor is rotated only while operation commands are being input. This is used to forcibly rotate the motor using inputs from external switches during startup or adjustment, for example. This can also be applied to unlimited feed.

■ Limit input (+), limit input (-)

A limit switch input that is used to limit the motor movement. Limit input (+) is the limit point on the side where the elapsed value increases and limit input (-) is the limit point on the side where the elapsed value decreases.

■ Dwell time

For E-point control, the time from the completion of a position command until the operation done contact turns ON can be specified as a dwell time. For C-point control, similarly, the time from deceleration stop until execution of the next positioning table can be specified as a dwell time.

■ Soft limit

Limits in software can be set for the absolute coordinates managed by the GM1 controller. When the range of soft limits is exceeded, an error occurs, causing the system to decelerate and stop. Deceleration time can be set individually.

■ Torque control

The output torque of the servo amplifier can be limited arbitrarily.

■ Servo ON / Servo OFF

The operation that changes the servo free state to a servo lock state is called "servo ON", and the operation that changes the servo lock state to a servo free state is called "servo OFF".

■ Linear interpolation

Interpolation control that controls straight lines as loci for the operations of 2-axis motors with grouped X-axis and Y-axis or 3-axis motors with grouped X-axis, Y-axis, and Z-axis. There are two setting methods, which are a composite speed specification and long axis speed specification.

■ Circular interpolation

Interpolation control that controls arcs as loci for the operation of 2-axis motors with grouped X-axis and Y-axis. There are two setting methods, which are a center point specification and pass point specification.

■ Edge detection

One of the methods for detecting the request signals allocated to this unit. It executes each requested process by detecting a trigger that is the rising edge when the request signal turns ON.

Therefore, the next request cannot be accepted until the current request signal turns OFF.

(MEMO)

Table of Contents

1 Before Using This Product 1-1

1.1 Safety Precautions 1-2

1.2 Handling Precautions 1-3

1.3 Software License Agreement 1-4

2 Operation Flow 2-1

2.1 System Configuration Diagram 2-2

2.2 Work Flowchart 2-4

3 Overview of the GM Programmer 3-1

3.1 System Requirements 3-2

3.1.1 Usage Environment of the GM Programmer 3-2

3.2 Installation and Uninstallation 3-3

3.2.1 Installing GM Programmer 3-3

3.2.2 Uninstalling GM Programmer 3-7

3.3 Basic Operations 3-8

3.3.1 How to start 3-8

3.3.2 How to quit 3-9

3.4 Component Names 3-10

3.4.1 Menu Bar 3-10

3.4.2 Toolbar 3-17

3.4.3 Navigator Pane 3-21

3.4.4 Main Pane 3-22

3.4.5 Status Bar 3-23

3.5 Window Operations 3-25

3.5.1 Moving the Pane Location 3-25

3.5.2 Showing / Hiding Panes 3-27

3.5.3 Switching the Tab of the Main Pane 3-28

3.5.4 Full-screen Display 3-29

3.6 Switching the Object Window 3-31

3.6.1 Operating the Object Window 3-31

3.7 Other Functions 3-32

3.7.1 Option Setting Function 3-32

3.7.2 Display Language Setting Function 3-33

3.7.3 Online Help Function 3-34

3.7.4 Version Display Function 3-35

4 Project Operations 4-1

4.1 Creating a New Project 4-2

4.2 Saving a Project 4-5

4.3 Opening a Project 4-6

4.4 Closing a Project 4-7

4.5 Device Tree Configuration 4-8

4.6 Project Configuration 4-11
4.7 Adding an Object 4-12
4.7.1 Adding Objects 4-13
4.7.2 Adding Devices 4-17
4.8 Setting up A Project 4-19
4.9 Exporting and Importing Objects 4-21
4.9.1 Exporting Objects 4-21
4.9.2 Importing Objects 4-22
4.10 Creating a Backup when a Project Is Saved 4-25
4.11 Automatically Saving Project Files 4-27
4.12 Printing a Project 4-29
4.13 Printing an Object within a Project 4-31
4.14 Comparing Projects 4-32
4.14.1 Project Comparison Method 4-32
4.14.2 Merging Differences 4-35

5 Project Creation 5-1
5.1 Project Creation Flow 5-2
5.2 Setting up the GM1 Controller 5-3
5.3 Setting up Motion Control 5-5
5.3.1 Basic Setting for GM1 Controller 5-5
5.3.2 Adding and Setting up Servo Amplifiers 5-6
5.3.3 Adding and Setting up Free Encoder and Virtual Drive 5-11
5.4 Setting up Unit Control 5-14
5.4.1 Setting up General-purpose I/O, PWM Output, and High-speed Counter for GM1 Controller 5-14
5.4.2 Adding Expansion Units 5-16
5.5 Setting up the Communication Function 5-20
5.5.1 Adding a Protocol to Be Used for the LAN Port 5-20
5.5.2 Adding a Protocol to Be Used for the COM Port 5-23

6 Program Creation 6-1
6.1 Flow of Program Creation 6-2
6.2 Program Creation Window 6-3
6.2.1 Main Pane 6-3
6.2.2 Declaration Editor 6-3
6.2.3 Auto Declaration 6-5
6.2.4 Toolbox 6-7
6.2.5 Setting up the Program Input Window 6-8
6.2.6 Window Operations for the Program Input Window 6-12
6.3 Creating a Program Object (POU Object) 6-14
6.4 Types of Programming Language 6-15
6.5 Variables 6-18
6.5.1 Standard Data Types 6-18
6.5.2 STRING type 6-19
6.5.3 WSTRING type 6-19

6.5.4 Array 6-19
6.5.5 Subrange Types 6-20
6.5.6 Structure, Enumeration, Alias, and Union Data Types 6-21
6.5.7 Constants 6-24
6.5.8 Object for Global Variable Declaration 6-25
6.5.9 Global Variables 6-25
6.5.10 Persistent Variables 6-27
6.5.11 Short Form Function 6-30
6.6 Function and Function Block 6-32
6.6.1 Function 6-32
6.6.2 Function Block 6-36

7 Entering Programs in Each Programming Language 7-1

7.1 Programming in Ladder Diagram (LD) 7-3
7.1.1 Inserting Contacts, Coils, and Function Blocks 7-3
7.1.2 Inserting Contacts in Parallel 7-8
7.1.3 Inserting a Network (Circuit) 7-9
7.1.4 Inserting a Branch 7-10
7.1.5 Input of Title and Comment (LD) 7-11
7.1.6 Commenting out a Network (Circuit) 7-13
7.2 Programming in Structured Text (ST) 7-15
7.2.1 ST Program Syntax 7-15
7.2.2 Commenting out Code in ST Program 7-17
7.3 Programming in Sequential Function Chart (SFC) 7-19
7.3.1 Inserting Elements from Menu 7-19
7.3.2 Inserting Elements from Toolbox 7-21
7.3.3 Inserting Elements from Toolbar 7-22
7.3.4 Setting up the SFC Editor 7-23
7.3.5 Setting SFC Program Execution Conditions 7-25
7.4 Programming in Function Block Diagram (FBD) 7-27
7.4.1 Entering Function Blocks 7-27
7.4.2 Inserting and Commenting out a Network (Circuit) 7-32
7.4.3 Input of Title and Comment (FBD) 7-32
7.4.4 Settings in FBD Program 7-32
7.5 Programming in Instruction List (IL) 7-34
7.5.1 Entering Instructions and Operands 7-34
7.5.2 Settings in IL Program 7-36
7.6 Programming in Continuous Function Chart (CFC) 7-38
7.6.1 Inserting and Connecting Elements 7-38
7.6.2 Connection Mark 7-43
7.7 Program Creation Support Functions 7-44
7.7.1 Bookmark 7-44
7.7.2 Call Tree View 7-44
7.7.3 Cross reference List View 7-45
7.7.4 Function Block Guidance 7-46
7.7.5 Input Assistant Function 7-50
7.7.6 Argument / Variable Input Support (Component List) 7-51
7.7.7 Global Renaming (Refactoring) 7-52
7.7.8 Displaying Programs in Multiple Languages (Project Localization) 7-55
7.8 Build 7-59

7.8.1 Build 7-59
7.8.2 Rebuild 7-59
7.8.3 Code Generation 7-59
7.8.4 Clean 7-60
7.8.5 Clean All 7-61
7.9 Tasks 7-63
7.9.1 Adding Programs 7-63
7.9.2 Adding a UserTask 7-68
7.9.3 Task Configuration Window 7-70

8 Connecting the GM1 Controller and PC 8-1
8.1 Flow of Operation Check 8-2
8.2 Connecting the GM1 Controller and PC 8-3
8.2.1 Selecting a Connection Port for GM Programmer 8-3
8.2.2 Connecting the GM1 Controller and PC with a Cable 8-3
8.2.3 Operation when Power is ON 8-3
8.3 Operation Mode Switching 8-5
8.4 Communication Setting 8-6
8.4.1 Adding USB Ports 8-6
8.4.2 Setting the LAN Port 8-8
8.5 Connecting to the GM1 Controller 8-10
8.6 Setting Time 8-12
8.7 Other Settings 8-14
8.7.1 Changing the Device Name 8-14
8.7.2 Sending Echo Services 8-15
8.7.3 Device Preference Management 8-17
8.7.4 Confirmed Online Mode 8-18
8.8 Login / Logout 8-20
8.8.1 Login 8-20
8.8.2 Logout 8-21
8.8.3 Download 8-21
8.8.4 Online Change 8-23
8.8.5 Code Analysis (Static Analysis Light) 8-23
8.9 Source Upload 8-27
8.10 Commissioning 8-29
8.10.1 Online Config Mode 8-29
8.10.2 Conducting Commissioning for Servo Amplifiers 8-30

9 Debug 9-1
9.1 Running and Stopping the GM1 Controller 9-2
9.1.1 Running and Stopping the GM1 Controller 9-2
9.1.2 Single Cycle 9-3
9.2 Breakpoint 9-5
9.2.1 Setting a Breakpoint 9-5
9.2.2 Setting an Execution Point 9-6
9.2.3 Call Stack View 9-8
9.3 Debug Operations 9-10
9.3.1 Writing Values and Forcibly Changing Values 9-10

9.3.2 Watch 9-11
9.3.3 Flow Control 9-12
9.3.4 Operation Mode 9-14
9.4 Monitoring Function 9-15
9.5 Reset 9-18
9.5.1 Reset Warm, Reset Cold, and Reset Origin 9-19
9.5.2 Executing Device Reset from GM Programmer 9-19
9.5.3 Executing Device Reset from GM1 Controller 9-20
9.6 Checking the Status of GM1 Controller 9-22
9.6.1 Checking Logs 9-22
9.6.2 Checking the Status 9-23
9.6.3 Checking the System Data History 9-24
9.6.4 Task Monitoring 9-25
9.7 Device Trace Function 9-26
9.8 Checking the Performance of GM1 Controller 9-33
9.8.1 Checking Missing EtherCAT Packets 9-33
9.8.2 Performance Check Based on Device Trace 9-34
9.9 Error Notification Function 9-36
9.9.1 Overview of Errors 9-36
9.9.2 Checking and Clearing Errors Using GM Programmer 9-36
9.9.3 Obtaining Error Information Using User Programs 9-37
9.9.4 Error Recovery Processing 9-40
9.9.5 Error Code List 9-42

10 Useful Functions of GM Programmer 10-1

10.1 Simulation Function 10-2
10.2 Security Function 10-3
10.3 Security Function: User Management 10-4
10.3.1 Project User Management 10-4
10.3.2 Creating a New User and Group 10-4
10.3.3 Setting Operation Privileges 10-8
10.3.4 Performing Operation with Privileges Set 10-10
10.3.5 Device User Management 10-11
10.4 Security Function: Encryption 10-16
10.4.1 Encrypting Project Files 10-16
10.4.2 Encrypting the Communication Path: Encrypting Communications Using the Certificate Possessed by the GM1 Controller 10-17
10.4.3 Encrypting the Communication Path: Encrypting Communications Using a Created Certificate 10-19
10.5 Security Function: Write-protection 10-24
10.5.1 Opening Files in Read-only Mode 10-24
10.5.2 Setting the "Released" Flag 10-24
10.6 User Library Function 10-26
10.6.1 Creating a Library and Adding to the Library Repository 10-26
10.6.2 Using Created Libraries 10-31
10.7 POU for implicit checks 10-34
10.7.1 Setting up POU for implicit checks 10-34
10.8 Interface 10-36

10.8.1 Setting up an Interface Object 10-36
10.8.2 Implementing in New Function Block 10-38
10.8.3 Implementing in Existing Function Block 10-40
10.8.4 Extending the Interface 10-43
10.9 External File Functions 10-46
10.9.1 Setting up an External File Object 10-46
10.10 Servo Amplifier / Motor Operation Function (PANATERM Lite for GM) 10-48
10.10.1 Starting PANATERM Lite for GM 10-48

11 Motion Control 11-1

11.1 EtherCAT Master Setting 11-3
11.1.1 EtherCAT_Master_SoftMotion 11-3
11.2 EtherCAT Slave Settings 11-6
11.2.1 EtherCAT Slave Settings 11-6
11.3 EtherCAT Axis Settings 11-12
11.3.1 Overview of EtherCAT Axis Settings 11-12
11.3.2 Basic Settings for EtherCAT Axes 11-12
11.3.3 EtherCAT Axis Extended Setting 11-16
11.4 Basic Preparations for Operation 11-23
11.4.1 Overview of Basic Preparations for Operation 11-23
11.4.2 Servo ON or OFF 11-23
11.4.3 Home Return 11-24
11.4.4 JOG Operation 11-36
11.5 Single-axis Operation 11-38
11.5.1 Overview of Single-axis Operation 11-38
11.5.2 Position Control 11-38
11.5.3 Switching the Control Mode 11-45
11.5.4 Velocity Control 11-46
11.5.5 Torque Control 11-48
11.5.6 Stop 11-50
11.6 Synchronous Operation 11-52
11.6.1 Synchronous Cam Operation 11-52
11.6.2 Synchronous Gear operation 11-58
11.7 Multi-axis Operation 11-61
11.7.1 Overview of Interpolation Control 11-61
11.7.2 Linear Interpolation and Circular Interpolation 11-62
11.7.3 How to Use Interpolation Control 11-63
11.7.4 Registering a CNC Table 11-64
11.7.5 Overview of G-code 11-75
11.7.6 G-code Editor and Coding Rules 11-77
11.7.7 Movements Executed by Each G-code and Setting Methods 11-78
11.7.8 SMC_CNC_REF and SMC_OUTQUEUE 11-90
11.7.9 Interpolation Operation Programming: How to Create a Program for Executing Operation 11-91
11.7.10 Interpolation Operation Programming: Explanation of Function Block (FB) 11-94
11.7.11 Interpolation Operation Programming: Specifying the Starting Coordinates 11-97

11.7.12 Interpolation Operation Programming: P-point Control and C-point Control 11-101
11.7.13 Interpolation Operation Programming: Settings in CNC Table for C-point Control 11-103
11.7.14 Interpolation Operation Programming: Settings in POU for P-point Control and C-point Control 11-104
11.7.15 Interpolation Operation Programming: Joining and Repeating CNC Tables 11-105
11.7.16 Interpolation Operation Programming: Changing Parameter Settings (Converting to Variables in CNC Table) 11-108
11.8 Motion Function Errors 11-110
11.8.1 Overview of Motion Function Errors 11-110
11.8.2 Error Check Method 11-111
11.8.3 Clearing Errors 11-112

12 Unit Control 12-1
12.1 Overview of Unit Control 12-2
12.2 IO Parameters for Unit Control 12-3
12.3 I/O Mapping for Unit Control 12-4
12.4 General-purpose I/O 12-5
12.4.1 Overview of General-purpose I/O Function 12-5
12.4.2 Setting Parameters with GM Programmer 12-5
12.4.3 Setting Items of IO_Configuration Parameters 12-6
12.4.4 I/O Mapping for General-purpose I/O 12-7
12.5 PWM Output 12-9
12.5.1 Overview of PWM Output 12-9
12.5.2 Setting Output Ports with GM Programmer 12-9
12.5.3 I/O Mapping for PWM Output 12-10
12.5.4 Data Update Timing (Output Frequency) 12-11
12.5.5 Data Update Timing (Duty Ratio) 12-12
12.5.6 PWM Output Setting Example 12-12
12.6 High-speed Counter Function 12-18
12.6.1 Overview of High-speed Counter Function 12-18
12.6.2 Setting Parameters with GM Programmer 12-19
12.6.3 Counter Parameter Setting Items 12-22
12.6.4 I/O Mapping for High-speed Counter Output 12-25
12.6.5 Operation Ready Request 12-28
12.6.6 Count Function 12-31
12.6.7 Comparison Function 12-41
12.6.8 External Output Function 12-47
12.6.9 Capture Function 12-48
12.6.10 Unit Error 12-60
12.7 Settings of I/O Unit 12-62
12.7.1 Parameter Settings 12-62
12.7.2 I/O Mapping for I/O Unit 12-62

13 Communication Function 13-1
13.1 Overview of Communication Function 13-2
13.1.1 Adding Network Communication Devices 13-2
13.1.2 Adding Serial Communication Devices 13-3

13.2 General-purpose Communication 13-6
13.2.1 General-purpose Communication (Ethernet) 13-6
13.2.2 General-purpose Communication (Serial) 13-20
13.3 MODBUS 13-24
13.3.1 What is Modbus TCP? 13-24
13.3.2 Modbus-TCP Master Communication 13-24
13.3.3 Modbus-TCP Slave Communication 13-28
13.3.4 Modbus-RTU Master Communication 13-30
13.3.5 Modbus-RTU Slave Communication 13-38
13.4 EtherNet/IP 13-42
13.4.1 What is EtherNet/IP? 13-42
13.4.2 Cyclic Communication Function 13-42
13.4.3 EtherNet/IP Scanner Function 13-42
13.4.4 Setting up the EtherNet/IP Scanner Function 13-42
13.4.5 EtherNet/IP Scanner Operation 13-49
13.4.6 EtherNet/IP Adapter Function 13-51
13.4.7 Setting up the EtherNet/IP Adapter Function 13-51
13.4.8 EtherNet/IP Adapter Operation 13-55

14 Other Controller Functions 14-1

14.1 SD Card Access 14-2
14.1.1 Overview of SD Card Access Function 14-2
14.1.2 File Manipulations Using the CAA File Library 14-2
14.2 Time Function 14-10
14.2.1 Overview of Time Function 14-10
14.2.2 Settings Based on GM Programmer 14-10
14.2.3 Settings Based on Function Blocks 14-10
14.3 Trace Function 14-11
14.3.1 Setting up Trace 14-11
14.3.2 Executing Trace 14-14
14.4 Recipe Manager Functions 14-16
14.4.1 Setting the Recipe Manager 14-16
14.4.2 Setting the Recipe Definition 14-17
14.4.3 Recipe Operation Using the GM Programmer 14-19

15 Overview of PANATERM Lite for GM 15-1

15.1 System Requirements 15-3
15.1.1 Operating Environment of PANATERM Lite for GM 15-3
15.2 Installation and Uninstallation 15-4
15.2.1 Installing PANATERM Lite for GM 15-4
15.2.2 Uninstalling PANATERM Lite for GM 15-4
15.3 Basic Operations 15-5
15.3.1 How to Start 15-5
15.3.2 How to Exit 15-8
15.4 Component Names 15-9
15.4.1 Menu Bar 15-9
15.4.2 Toolbar 15-10
15.4.3 Navigation Pane 15-11
15.4.4 Main Pane 15-11

15.4.5 Status Field 15-12
15.5 Window Operations 15-13
15.5.1 Moving the Pane Location 15-13
15.5.2 Switching the Tab of the Main Pane 15-14
15.6 Selecting the Device to Connect 15-16
15.6.1 Configuring Servo Amplifier Communication Settings 15-16
15.6.2 Setting up the Servo Amplifier Connected to the GM1 Controller 15-18
15.6.3 Editing Settings without Connecting to the GM1 Controller 15-23
15.7 Parameter Window 15-25
15.7.1 Configuration of Parameters Window 15-25
15.7.2 Setting Parameters 15-27
15.7.3 Copying Parameters 15-29
15.7.4 Switching the Input Format of Parameter Values 15-30
15.7.5 Setting I/O Pin Assignment 15-31
15.8 Object Window 15-33
15.8.1 Configuration of Object Window 15-33
15.8.2 Setting Objects 15-36
15.9 MINAS Parameters for the GM1 Controller 15-38
15.10 Monitor Window 15-39
15.10.1 Configuration of Monitor Window 15-39
15.10.2 Checking the Monitor Window 15-41
15.11 Alarm Window 15-43
15.11.1 Configuration of Alarm Window 15-43
15.11.2 Checking Alarms 15-45
15.12 Other Functions 15-47
15.12.1 Language Setting Function 15-47
15.12.2 Help Function 15-47
15.12.3 Version Display Function 15-47
15.13 Troubleshooting for Servo Amplifiers and Motors 15-49
15.13.1 I Cannot Set up 15-49
15.13.2 I Cannot Communicate 15-49
15.13.3 I Cannot Print 15-49
15.13.4 I Cannot Set up Axes 15-50
15.13.5 PANATERM Lite for GM Does Not Behave Normally 15-50
15.13.6 The Parameter Window Does Not Behave Normally 15-50
15.13.7 The Object Window Does Not Behave Normally 15-51
15.13.8 The Monitor Window Does Not Behave Normally 15-51
15.13.9 The Alarm Window Does Not Behave Normally 15-51

Appendix Warranty / Cautions for Proper Use App-1
Warranty App-2
Warranty Period App-2
Warranty Scope App-2
Cautions for Proper Use App-3

(MEMO)

1 Before Using This Product

1.1 Safety Precautions....1-2
1.2 Handling Precautions....1-3
1.3 Software License Agreement....1-4

1.1 Safety Precautions

This section explains important rules that must be observed to prevent personal injury and property damage.

- Injuries and damages that may occur as a result of incorrect use are classified into the following levels and safety precautions are explained according to the level.

⚠ WARNING	Indicates that there is a risk of death or serious injury
⚠ CAUTION	Indicates that there is a risk of minor injury or property damage

	Indicates an action that is prohibited
	Indicates an action that must be taken

⚠ WARNING	
	• Take safety measures outside this product to ensure the safety of the entire system even if this product fails or an error occurs due to external factors.
	• Do not use this product in atmospheres that contain flammable gases. Doing so may result in explosion.
	• Do not throw this product into the fire. Doing so may cause the batteries or other electronic parts to explode.

⚠ CAUTION	
	• To prevent abnormal heat generation or smoke generation, use this product with some leeway from the guaranteed characteristics and performance values of the product.
	• Do not disassemble or modify this product. Doing so may result in abnormal heat generation or smoke generation.
	• Do not touch any terminals while the power is on. Doing so may result in electrical shock.
	• Configure emergency stop and interlock circuits outside this product.
	• Connect wires and connectors properly. Failure to do so may result in abnormal heat generation or smoke generation.
	• Do not perform work (such as connection or removal) with the power turned on. Doing so may result in electrical shock.
	• If this product is used in any way that is not specified by Panasonic, its protection function may be impaired.
	• This product has been developed and manufactured for industrial use only.

1.2 Handling Precautions

■ **In this manual, the following symbols are used to indicate safety information that must be observed.**

Stop	Indicates an action that is prohibited or a matter that requires caution.
!	Indicates an action that must be taken.
Info.	Indicates supplemental information.
Note	Indicates details about the subject in question or information useful to remember.
1 2 Procedure	Indicates operation procedures.

1.3 Software License Agreement

Software License Agreement

Panasonic Corporation ("PANASONIC") grants to you a license to use this Software on condition that you accept this Agreement. You must read this Software License Agreement (this "Agreement") carefully before using this Software. Only in case that you accept this Agreement, you may start your use of this Software.
Your unsealing the package of this Software, or your downloading, installing or launching this Software or the like shall be deemed as your acceptance of this Agreement.
The Software includes not only proprietary computer programs owned by or licensed by PANASONIC but also open source software programs. As for the open source software programs, refer to the detailed terms and conditions thereof shown in the installation package of the Software. Should a discrepancy arise between any of the terms of this Agreement and any open source software program license statement, the open source software program license statement shall take precedence over this Agreement.

Article 1 Grant of License

PANASONIC hereby grants to you a non-exclusive license to use this Software only in combination with PANASONIC product(s) specified in the manual of this Software (the "Product") in accordance with the terms of this Agreement. You may not use this Software in connection with products of any third party other than PANASONIC.

Article 2 Restrictions

You may NOT:

(1)Modify, reverse engineer, decompile, or disassemble this Software, except where the terms and conditions of open source software program license statement (including, but not limited to, GPL and LGPL) apply.
Should a defect of the Software arise owing to your modification, reverse engineering, decompiling, or disassembling, to the extent permitted under the law, Panasonic shall not assume any responsibility for such defect,

(2)Use this Software by methods or for purposes other than those specified in the manual of this Software provided by PANASONIC, nor

(3)Distribute, rent, lease or otherwise transfer this Software to any third party; provided, however, that you may assign the rights to use this Software under this Agreement along with the Product on the condition that the assignee agrees to be bound by all the terms of this Agreement. In the case of such assignment, you must deliver any and all the copies of this Software and all the accompanying materials to the assignee and you may not retain any copies of this Software including backups.

Article 3 Disclaimer

3-1. PANASONIC HEREBY DISCLAIMS ALL OTHER WARRANTIES ON THIS SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MARCHANTABILITY, FITNESS FOR PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

3-2. UNDER NO CIRCUMSTANCES SHALL PANASONIC BE LIABLE FOR ANY DAMAGES (INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR SPECIAL OR WHATSOEVER) ARISING OUT OF THE USE OF THIS SOFTWARE, INABILITY TO USE THIS SOFTWARE, DEFECTS IN THIS SOFTWARE (e.g., BUGS, SECURITY HOLES, AND MALFUNCTION), OR OTHERWISE IN CONNECTION

WITH THIS SOFTWARE.

Article 4 Term

4-1. This Agreement shall come into effect upon your unsealing the package of this Software, or your downloading, installing or launching this Software or the like.

4-2. PANASONIC may terminate this Agreement immediately, if you breach any of the provisions of this Agreement.

4-3. You shall, at your own costs, return, delete or destroy this Software and any of its copies within four (4) weeks after termination of this Agreement.

Article 5 Export Control

You shall comply with all laws and regulations regarding export control under any competent jurisdiction, including but not limited to the Japanese Foreign Exchange & Foreign Trade Control Law, the export control regulations based on resolutions of the United Nations Security Council, etc. If any license or appropriate approval from a governmental authority is required under the applicable laws, you may not export this Software without such approval to any countries either directly or indirectly. Furthermore, you shall neither use nor sell this Software for military purposes either directly or indirectly.

Article 6 Intellectual Property Rights

Except the open source software program (including, but not limited to, GPL and LGPL), all intellectual property rights in this Software, including the copyright, belong to PANASONIC and/or the licensors of PANASONIC.

Article 7 Upgrade of this Software

7-1 Release of future upgrades or updates of this Software is not guaranteed and left to the sole discretion of PANASONIC. Furthermore, PANASONIC may charge fees for upgrading or updating of this Software.

7-2. If any upgrades or updates are provided to you either for fees or for free, such upgrades or updates shall be deemed as a part of this Software and shall be governed by this Agreement, unless PANASONIC designates otherwise at the time of provision of such upgrades or updates.

Article 8 Limitation on Liability

AGGREGATE LIABILITIES OF PANASONIC IN CONNECTION WITH THIS AGREEMENT OR THIS SOFTWARE SHALL IN NO EVENT EXCEED TEN THOUSAND (10,000) YEN.

Article 9 Governing Law and Jurisdiction

9-1. This Agreement shall be governed by the laws of Japan.

9-2. Should any dispute arise from or in connection with this Agreement, Osaka District Court, Japan shall exclusively have the jurisdiction over such dispute.

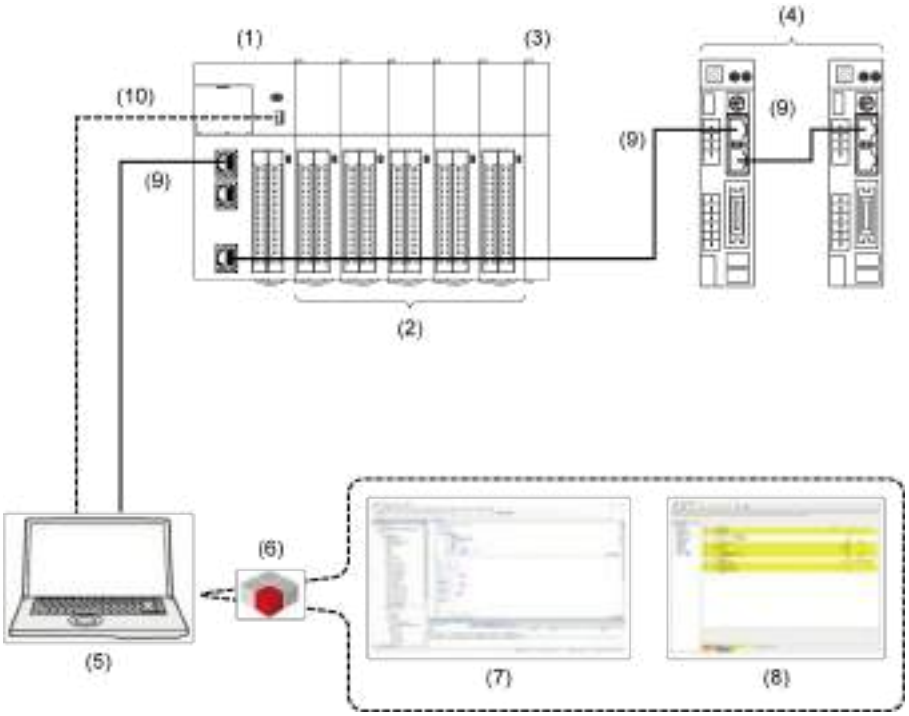
This Software consists of the following types of software.
(1) Software developed independently by PANASONIC
(2) Software owned by and licensed by the third party
(3) Software licensed under GNU General Public License Version 2.0 (GPL V2.0)
(4) Software licensed under GNU Lesser General Public License Version 2.0 (LGPL V2.0) or Version 2.1 (LGPL V2.1)
(5)Open source software licensed on conditions other than those of GPL V2.0, LGPL V2.0, or LGPL V2.1
Software in categories (3) – (5) above is distributed with the expectation of effectiveness as a single piece of software, but there is no guarantee provided, including implied guarantees regarding viability as a product and/or suitability for specific purposes. For details, Please refer to the detailed terms and conditions thereof shown in the installation package of the Software.
For at least three years following the release of the Product, PANASONIC will provide, at customer's expense, complete machine-readable source code for software licensed under GPL V2.0, LGPL V2.0, LGPL V2.1, or a license based on other conditions that meet source code disclosure requirements, along with information on the respective copyright holders, to customers who contact us at the following e-mail address.
【Contact e-mail address: oss-cd-request@gg.jp.panasonic.com】

2 Operation Flow

2.1 System Configuration Diagram 2-2
2.2 Work Flowchart 2-4

2.1 System Configuration Diagram

The figure below shows the configuration of the GM1 series motion controller (controller and expansion units), servo amplifiers, and PC. GM Programmer and PANATERM Lite for GM communicate with the GM1 Controller via Gateway.



No.	Name
(1)	GM1 Controller
(2)	Expansion unit
(3)	End unit
(4)	Servo amplifier
(5)	PC (on which GM Programmer and PANATERM Lite for GM are installed)
(6)	Gateway, CodeMeter
(7)	GM Programmer
(8)	PANATERM Lite for GM
(9)	Ethernet cable(Note 1)
(10)	USB cable(Note 1)

(Note 1) Use either Ethernet cables or USB cables.

Info.

- To operate the system, you must install GM Programmer and PANATERM Lite for GM on the PC.
- When GM Programmer is installed, MINAS setup support software "PANATERM Lite for GM", Gateway (the application that connects GM Programmer and the GM1 Controller), and CodeMeter are installed at the same time.

2.2 Work Flowchart

The following table explains the workflow from installation of the GM1 controller through to its operation.

Step	Description		Reference
1	Install GM Programmer and PANATERM Lite for GM.		"GM1 Controller EtherCAT User's Manual (Setup Edition)"
2	Make preparations for the servo amplifiers.		
	2-1	Connect the servo amplifiers and the PC.	
	2-2	Install the USB driver on the PC.	
	2-3	Configure initial settings for the servo amplifiers.	
	2-4	Disconnect the servo amplifiers from the PC.	
3	Connect the GM1 controller and EtherCAT slaves.		
4	Connect the GM1 controller and the GM Programmer.		
	4-1	Connect the GM1 controller and the PC with a cable.	
	4-2	Create a new project.	
	4-3	Make communication settings.	
	4-4	Add and set up EtherCAT slaves.	
	4-5	Make basic settings of the EtherCAT axis	
	4-6	Connect the GM1 controller, PC, and EtherCAT slaves and perform an operation check.	
	4-7	Log in to the GM1 controller.	
	4-8	Log out from the GM1 controller.	
5	Connect the GM1 controller and PANATERM Lite for GM.		
	5-1	Set up the servo amplifier connected to the GM1 controller.	
	5-2	Write parameters to the servo amplifier.	
	5-3	Write objects to the servo amplifier.	
6	Prepare for operation.		
	6-1	Check if safety circuit design is implemented.	
	6-2	Check wiring for each device.	
	6-3	Perform an operation check.	
7	Using the GM Programmer, make settings for GM1 parameters, motion control, unit control, and communication function.		
	7-1	Make settings for the GM1 controller.	"P.5-3"
	7-2	Make settings for the motion control.	"P.5-5"
	7-3	Make settings for the unit control.	"P.5-14"
	7-4	Make settings for the communication function	"P.5-20"
8	Create programs with GM Programmer.		"P.6-1"
	8-1	Create objects (POU objects) for a program.	"P.6-14"
	8-2	Select a programming language (LD, ST, SFC, FBD, IL, or CFC program) and enter a program.	"P.6-15"

Step	Description		Reference
	8-3	Set variables.	"P.6-18"
9	Set up the GM1 controller in GM Programmer.		
	9-1	Make time setting.	"P.8-12"
	9-2	Log in to the GM1 controller.	"P.8-20"
	9-3	Log out from the GM1 controller.	"P.8-21"
	9-4	Upload the source.	"P.8-27"
10	Configure security settings with GM Programmer.		"P.10-3"
	10-1	Configure user management settings.	"P.10-4"
	10-2	Configure encryption and signature settings.	"P.10-16"
	10-3	Configure write-protection settings.	"P.10-24"

(MEMO)

3 Overview of the GM Programmer

3.1 System Requirements....3-2
3.1.1 Usage Environment of the GM Programmer....3-2
3.2 Installation and Uninstallation....3-3
3.2.1 Installing GM Programmer....3-3
3.2.2 Uninstalling GM Programmer....3-7
3.3 Basic Operations....3-8
3.3.1 How to start....3-8
3.3.2 How to quit....3-9
3.4 Component Names....3-10
3.4.1 Menu Bar....3-10
3.4.2 Toolbar....3-17
3.4.3 Navigator Pane....3-21
3.4.4 Main Pane....3-22
3.4.5 Status Bar....3-23
3.5 Window Operations....3-25
3.5.1 Moving the Pane Location....3-25
3.5.2 Showing / Hiding Panes....3-27
3.5.3 Switching the Tab of the Main Pane....3-28
3.5.4 Full-screen Display....3-29
3.6 Switching the Object Window....3-31
3.6.1 Operating the Object Window....3-31
3.7 Other Functions....3-32
3.7.1 Option Setting Function....3-32
3.7.2 Display Language Setting Function....3-33
3.7.3 Online Help Function....3-34
3.7.4 Version Display Function....3-35

3.1 System Requirements

3.1.1 Usage Environment of the GM Programmer

Programming software

Product name	Version	Applicable language
GM Programmer	Ver.1.1	Japanese / English / Chinese

(Note 1) When GM Programmer is installed, MINAS setup support software "PANATERM Lite for GM" is installed at the same time.

Software operating environment

Item	Description
OS	Microsoft(R) Windows(R) 10: 32 bit / 64 bit
PC	PC with the following installed: ● Microsoft.NET Framework 4.6.1 or higher ● Microsoft Visual C++ 2010 SP1 Redistributable Package (x86) ● Microsoft Visual C++ 2010 SP1 Redistributable Package (x64) ● Microsoft Visual C++ 2013 Redistributable Package (x86) ● Microsoft Visual C++ 2013 Redistributable Package (x64) ● Microsoft Visual C++ 2015 Update 3 Redistributable Package (x86) ● Microsoft Visual C++ 2015 Update 3 Redistributable Package (x64)
HDD	At least 4 GB of free space
Memory	At least 8 GB
Communication port	LAN port (for Ethernet connection) USB 2.0 port (for USB connection)

3.2 Installation and Uninstallation

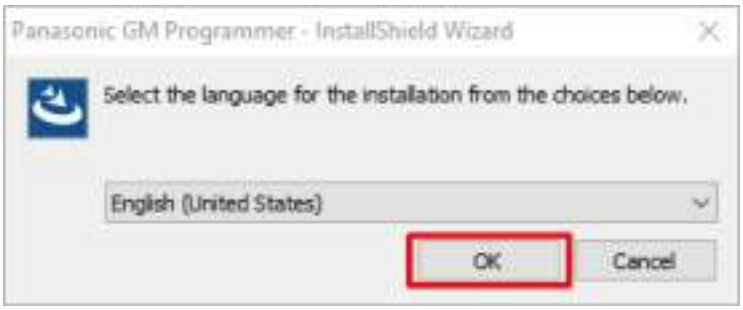
3.2.1 Installing GM Programmer

Before installing the GM Programmer on a PC, log on to the PC as an account with Administrator privileges.

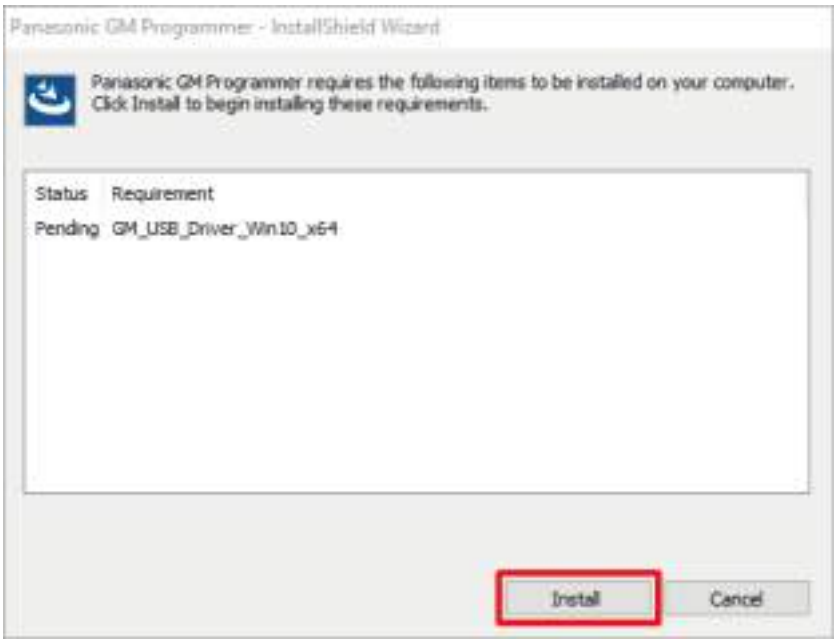
If other applications are running, be sure to close all the applications before installing GM Programmer.

1 2 Procedure

1. Double-click "setup.exe".
   The following window will be displayed. Click [OK].



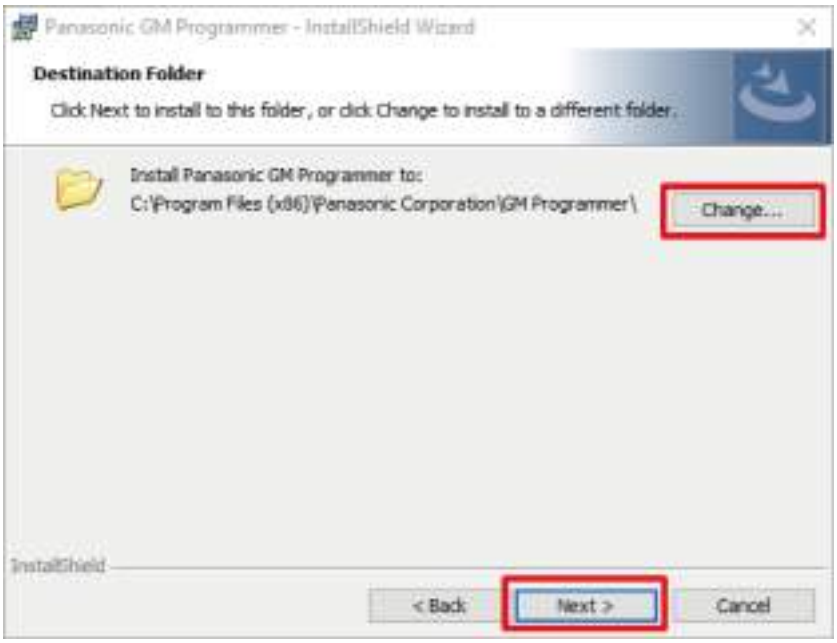
2. The following window will be displayed. Click [Install].
   The display content differs according to the PC environment that you use. (This window may not be displayed at all, depending on the situation.)



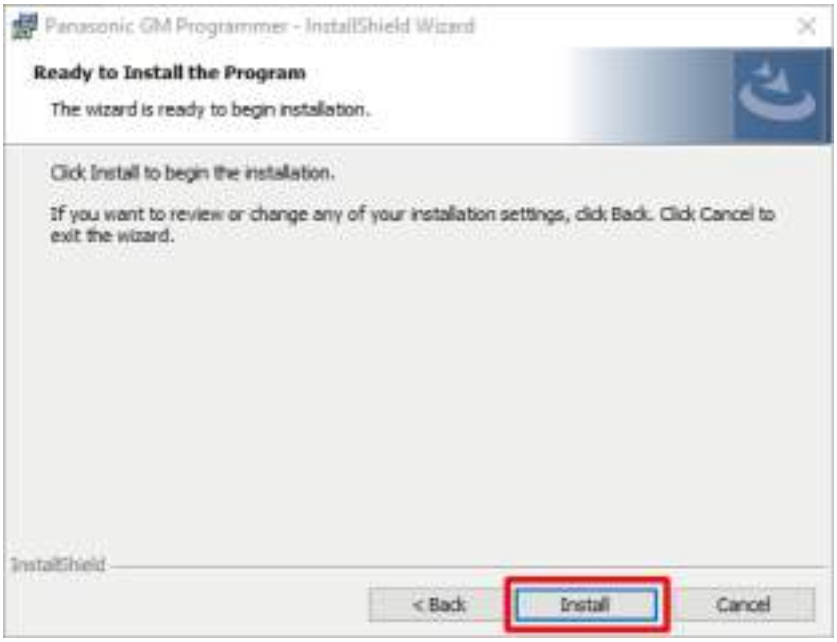
3. The following window will be displayed. Click [Next].

4. The following window will be displayed. Select [I accept the terms in the license agreement] and click [Next].

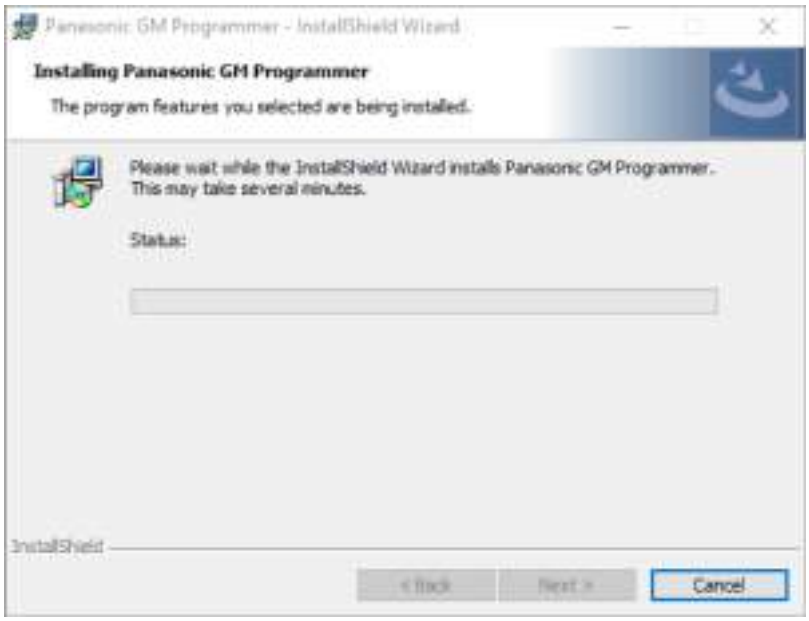
5. The following window will be displayed. If you change the installation destination folder, click [Change] and specify a desired installation destination. If you do not change the installation destination folder, click [Next].



6. The window below will be displayed. Click [Install] to start the installation.



7. The following window will be displayed while the installation is in progress.

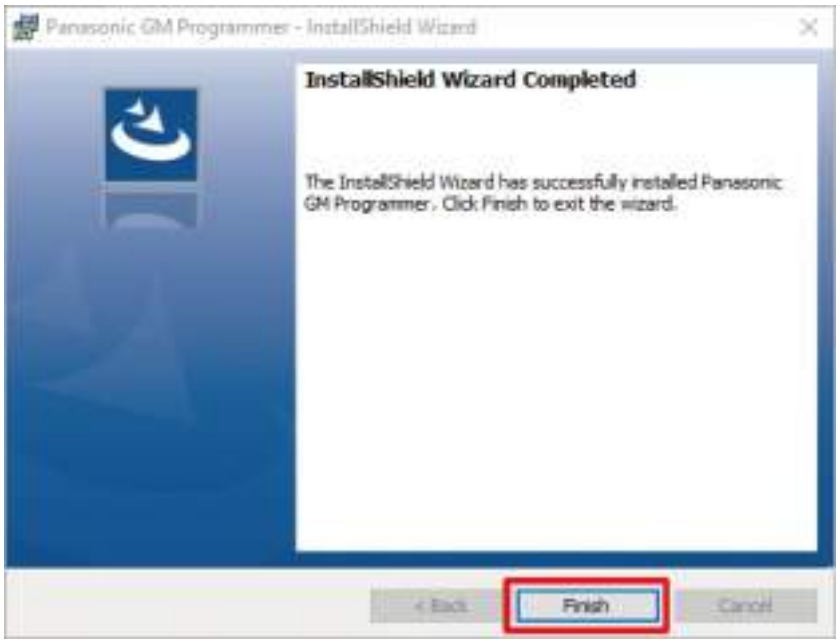


Following this installation, the three packages below will be installed. (The segments indicated by * differ according to the version of the software.)

- CODESYS SoftMotion*.*.*.*_P
- GMPLibrary (*.*.*.*)
- PANATERM-Lite for GM V*.*

These packages take a long time to install. Take care not to click [Cancel] while the installation is in progress.

8. When the installation of all the packages is completed, the following window will be displayed. Click [Finish].



This completes the installation procedure.

Info.

- When the GM Programmer is installed, PANATERM Lite for GM, Gateway (CODESYS Gateway), and CodeMeter applications are installed at the same time.

3.2.2 Uninstalling GM Programmer

Procedure

1. From the Start menu, select **Windows System**>**Control Panel**, and then click "Uninstall a program".
   A list of installed programs will be displayed.
2. Double-click "Panasonic GM Programmer".
   The following window will be displayed. [Yes]

3. Click the [Yes] button.
   The GM Programmer will be uninstalled.

Info.

- When the GM Programmer is uninstalled, PANATERM Lite for GM and Gateway are also uninstalled at the same time.
- CodeMeter will not be uninstalled at this time. Uninstall it separately.

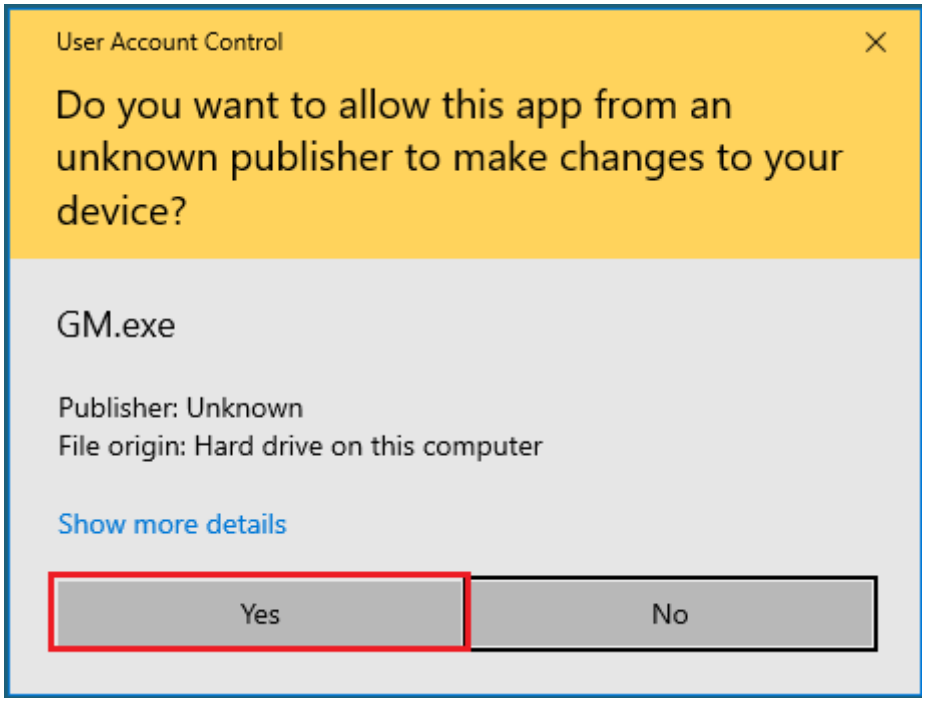
3.3 Basic Operations

This section explains how to start and quit GM Programmer.

3.3.1 How to start

Procedure

1. Click the [Start] button and select **Panasonic Corpration>GM Programmer**.
   The "User Account Control" dialog box will be displayed. Click [Yes].



   GM Programmer will be started.

3.3.2 How to quit

- Before closing GM Programmer, be sure to save any project files that you are editing and must save.

Procedure

1. From the menu bar, select **File**>**Exit**.
   If changes have not been saved, the following window will be displayed.
   If exiting without saving, select [No].
   If changes need to be saved, select [Yes] to perform the save process.

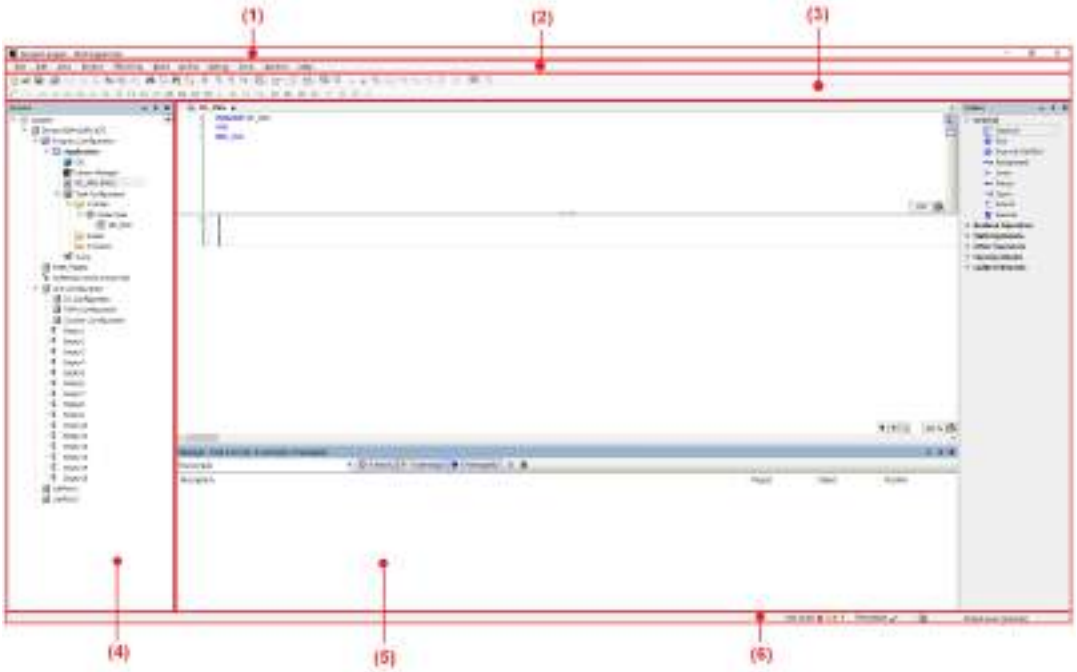
2. Click the [Yes] button.
   GM Programmer will be closed.

Info.

- You can also close GM Programmer by clicking the [×] button on the title bar.

3.4 Component Names

This section presents the name and display content of each component of GM Programmer.



No.	Name	Description
(1)	Title bar	The title bar displays the project file name, [minimize] button, [maximize] button, and [close] button.
(2)	Menu bar	The menu bar displays the menu commands for each purpose in list format.
(3)	Toolbar	The toolbar displays each command as an icon.
(4)	Navigator pane	The navigator pane displays the objects (such as devices, applications, and programs) added to the project in a tree structure.
(5)	Main pane	The main pane displays a program, function settings, messages, and other data. The window can be switched by selecting a desired tab.
(6)	Status field	The status bar displays the build status, logged-in users, and other information.

3.4.1 Menu Bar

The menu bar displays the following menus:

File Edit View Project FBD/LD/IL Build Online Debug Tools Window Help

File

Item	Function
New Project	Creates a new project.
Open Project	Opens a project that is stored.

Item	Function
Close Project	Close the project that is currently viewed.
Save Project	Saves the project that is currently viewed, in overwrite mode.
Save Project As	Saves the project that is currently viewed, as a different file name.
Source Upload	Loads the project source code as a project archive.
Print	Prints the active editor screen.
Print Preview	Displays the active editor screen in print preview mode.
Page Setup	Opens the Page Setup dialog box to configure a print layout.
Recent Projects	Displays the recently used projects.
Exit	Closes GM Programmer.

Edit

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Undo</td><td>Reverses the results of a previous editing action.</td></tr>
<tr><td>Redo</td><td>Allows the user to redo the last editing action after Undo.</td></tr>
<tr><td>Cut</td><td>Cuts data.</td></tr>
<tr><td>Copy</td><td>Copies data.</td></tr>
<tr><td>Paste</td><td>Pastes data.</td></tr>
<tr><td>Delete</td><td>Deletes data.</td></tr>
<tr><td>Select All</td><td>Selects all text.</td></tr>
<tr><td>Find Replace</td><td>Used to find and replace a string.

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Find</td><td>Opens the "Find" dialog box.</td></tr>
<tr><td>Replace</td><td>Opens the "Replace" dialog box.</td></tr>
<tr><td>Find in Project</td><td>Opens the "Find" dialog box to find the target within the entire project.</td></tr>
<tr><td>Replace in Project</td><td>Opens the "Replace" dialog box to replace the target within the entire project.</td></tr>
<tr><td>Find Next</td><td>Finds the next match from the selected cursor position within the project.</td></tr>
<tr><td>Find Next (Selected)</td><td>Finds the next match from the selected cursor position within the editor.</td></tr>
<tr><td>Find Previous</td><td>Finds the previous match from the selected cursor position within the project.</td></tr>
<tr><td>Find Previous (Selected)</td><td>Finds the previous match from the selected cursor position within the editor.</td></tr>
<tr><td>Toggle Field for Incremental Search</td><td>Searches for the character string within the POU editor each time a single character is entered.</td></tr>
</table>
</td></tr>
<tr><td>Browse</td><td>Used to browse the positions where the declaration part of a defined variable is referenced or used.</td></tr>
</table>

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Go To Definition</td><td>Allows the cursor to move to the position where the variable or function specified by the cursor is defined within the editor.</td></tr>
<tr><td>Browse Cross References</td><td>Allows the positions where the variable specified by the cursor is used to be displayed in the "Cross reference List" view.</td></tr>
<tr><td>Browse Call Tree</td><td>Allows the callee and caller of the variable specified by the cursor to be displayed in the "Call Tree" view.</td></tr>
<tr><td>Go To Reference</td><td>Displays the declaration position of the variable to which the pointer variable specified by the cursor refers.</td></tr>
<tr><td>Go To Instance</td><td>Displays the instance of the function block specified by the cursor in the new editor.</td></tr>
</table>
</td></tr>
<tr><td>Insert File as Text</td><td>Inserts the contents of the specified text file in the cursor position.</td></tr>
<tr><td>Advanced</td><td>Executes functions related to the text editor.

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Overwrite Mode</td><td>Switches the text input mode from insert mode to overwrite mode.</td></tr>
<tr><td>View Whitespace</td><td>Displays the control characters of spaces and tabs.</td></tr>
<tr><td>View Indentation Guides</td><td>Inserts a broken line between indents when an indent is inserted in the program code.</td></tr>
<tr><td>Go To Line</td><td>Displays a line number dialog box and moves the cursor to the specified line.</td></tr>
<tr><td>Make Uppercase</td><td>Converts the selected character string in the text editor to uppercase letters.</td></tr>
<tr><td>Make Lowercase</td><td>Converts the selected character string in the text editor to lowercase letters.</td></tr>
<tr><td>Go to Matching Bracket</td><td>Moves the cursor to the corresponding bracket when the cursor is positioned in a bracket in the code.</td></tr>
<tr><td>Select to Matching Bracket</td><td>Selects the entire code in brackets when the cursor is positioned in either of the brackets in the code.</td></tr>
<tr><td>Expand All Folds</td><td>Unfolds the indented code segment.</td></tr>
<tr><td>Collapse All Folds</td><td>Folds the indented code segment.</td></tr>
<tr><td>Comment out selected lines</td><td>Comments out the selected line.</td></tr>
<tr><td>Uncomment selected lines</td><td>Uncomments the selected line.</td></tr>
<tr><td>Enable inline monitoring</td><td>Sets whether to enable or disable the function that displays the value of each variable on the code during online mode.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Bookmarks</td><td>Allows the cursor to move to bookmarked locations.
Used to browse the positions where the declaration part of a defined variable is referenced or used.

<table><tr><th>Item</th><th>Function</th></tr><tr><td>Toggle Bookmark</td><td>Saves the position selected in the active editor as a bookmark.</td></tr><tr><td>Next Bookmark (active editor)</td><td>Moves to the previous bookmark in the active editor.</td></tr><tr><td>Previous Bookmark (active editor)</td><td>Moves to the next bookmark in the active editor.</td></tr><tr><td>Clear All Bookmarks (active editor)</td><td>Removes all bookmarks in the active editor.</td></tr></table></td></tr>
<tr><td>Input Assistant</td><td>Allows the user to select variables, function blocks, operators, types, or other data that can be inserted in the cursor position from a category and insert them in the cursor position.</td></tr>
<tr><td>Function Block Guidance</td><td>Invokes the Function Block Guidance.</td></tr>
<tr><td>Auto Declare</td><td>Opens the Auto Declare dialog box to support variable declaration.</td></tr>
<tr><td>Next Message</td><td>Selects the next message in the message view.</td></tr>
<tr><td>Previous Message</td><td>Selects the previous message in the message view.</td></tr>
<tr><td>Go To Source Position</td><td>Moves to the position of the source code applicable to the message selected in the message view.</td></tr>
<tr><td>Refactoring</td><td>Displays the positions where the changed variable name is used and allows changes to be made collectively.</td></tr>
</table>

View

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Devices</td><td>Displays the device view.</td></tr>
<tr><td>POUs</td><td>Displays the POU view.</td></tr>
<tr><td>Messages</td><td>Displays the message window.</td></tr>
<tr><td>Element properties</td><td>Displays element properties.</td></tr>
<tr><td>ToolBox</td><td>Displays the toolbox.</td></tr>
<tr><td>Watch</td><td>Displays the watch window.

<table><tr><th>Item</th><th>Function</th></tr><tr><td>Watch 1 to Watch 4</td><td>Displays a list of user-defined variables for the purpose of value monitoring.</td></tr><tr><td>Watch all Forces</td><td>Displays a list of value-forced variables.</td></tr></table></td></tr>
<tr><td>Cross Reference List</td><td>Displays the cross reference list window.</td></tr>
<tr><td>Call Tree</td><td>Displays the call tree window.</td></tr>
<tr><td>Bookmarks</td><td>Displays the bookmark window.</td></tr>
</table>

Item	Function
Breakpoints	Displays the breakpoint window.
Call Stack	Displays the call stack window.
Start Page	Displays the start page.
Full Screen	Displays the window in full-screen mode.
Properties	Displays the properties dialog box.

Project

Item	Function
Add Object	Adds an object.
Add Folder	Adds a folder.
Edit Object	Allows the user to edit an object.
Online Config Mode	Removes the applications downloaded to the GM1 controller and allows connection to the GM1 controller.
Project Information	Allows the user to set project author information or check project file information.
Project Settings	Allows the user to configure project-related settings.
Localization	Allows the user to translate and register comments, titles, and other information in the program to display the translated content in the program window. **Item** / **Function** Create Localization Template — Creates and saves a localization template. Manage Localizations — Imports a localization template that has been created. Toggle Localization — Switches the language in the project.
Document	Allows the user to print the entire project.
Compare	Compares the displayed project with the stored project.
Commit accepted changes	Commits the difference between the objects compared by selecting **Project**>**Compare** from the menu bar.
Export	Outputs an object from the displayed project as an XML file.
Import	Imports an object into the displayed project.
User Management	Allows execution permissions for operations (such as executing menu commands and adding, editing, and deleting objects) to be assigned to each group in which users are registered. **Item** / **Function** User Logon — Logs in to the displayed project. User Logoff — Logs off from the displayed project. Permissions — Logs off from the displayed project.

Build

Item	Function
Build	Verifies the syntax of objects.
Rebuild	Verifies the syntax of all objects again.
Generate code	Generates application codes.
Clean	Deletes application build information.
Clean all	Deletes all application build information in the same way as "Clean".

Online

<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Add USB Port</td><td>Adds a USB port as a communication interface.</td></tr>
<tr><td>Login</td><td>Downloads the applications generated by code generation to the GM1 controller at the time of login.</td></tr>
<tr><td>Logout</td><td>Logs out from the device to which the user logged in.</td></tr>
<tr><td>Download</td><td>Downloads a program while the user is logged in.</td></tr>
<tr><td>Online Change</td><td>Allows the user to change applications without having to stop the GM1 controller during operation.</td></tr>
<tr><td>Status</td><td>Allows the user to check any errors that are currently occurring in the GM1 controller.</td></tr>
<tr><td>System Data History</td><td>Allows the user to check any errors that occurred in the GM1 controller.</td></tr>
<tr><td>Reset Warm</td><td>Initializes variables other than the RETAIN and PERSISTENT variables.</td></tr>
<tr><td>Reset Cold</td><td>Initializes variables other than the PERSISTENT variable.</td></tr>
<tr><td>Reset Origin</td><td>Initializes all variables. Removes active applications from the GM1 controller.</td></tr>
<tr><td>Simulation</td><td>Allows the user to perform a login operation without connecting to the GM1 controller and check behaviors in the same way as if the user logged in.</td></tr>
<tr><td>Security</td><td>Allows the user to configure user management, project encryption, and other settings.
<table>
<tr><th>Item</th><th>Function</th></tr>
<tr><td>Logoff Current Device User</td><td>Logs off the users who are logged in to the device.</td></tr>
<tr><td>Add Device User</td><td>Adds users who can log in to the device.</td></tr>
<tr><td>Change Password Device User</td><td>Changes the passwords of users who are logged in to the device.</td></tr>
<tr><td>Remove Device User</td><td>Removes users who can log in to the device.</td></tr>
</table></td></tr>
<tr><td>Operation Mode</td><td>Allows the user to prevent some debug operations from being executed.</td></tr>
</table>

Item	Function
	<table><tr><th>Item</th><th>Function</th></tr><tr><td>Debug</td><td>Allows all debug operations to be executed.</td></tr><tr><td>Locked</td><td>Prohibits some operations such as adding new breakpoints or forcing variable values.</td></tr><tr><td>Operational</td><td>Prohibits any changes other than writing variables.</td></tr></table>

Debug

Item	Function
Start	Starts the application.
Stop	Stops the application.
Single Cycle	Executes the application in every single cycle.
New Breakpoint	Creates a new breakpoint.
Edit Breakpoint	Allows the user to edit breakpoints.
Toggle Breakpoint	Allows the user to set or delete breakpoints.
Disable Breakpoint	Disables invalid breakpoints.
Enable Breakpoint	Enables valid breakpoints.
Step Over	Executes the program line by line. When a block (function or function block) is executed in the block invocation location, the cursor moves to the next line.
Step Into	Executes the program line by line. When a block (function or function block) is executed in the block invocation location, the cursor moves to the first line of the called block.
Step Out	When the program is executed within the called block, the execution continues until control returns to the calling block. When the program is executed outside the called block, the execution continues until control returns to the beginning of the program.
Run to Cursor	Executes the program up to the line specified by the cursor.
Set next Statement	Regards the line specified by the cursor as the next statement to be executed and skips processes over to that line.
Show next Statement	Jumps the cursor to the program line to be executed as the next step.
Write Values	Sets a value (to be changed later) only once. This value can then be changed by the program.
Force Values	Sets a value to be changed in every cycle and maintains the value.
Unforce Values	Cancels forced value change
Toggle Flow Control Mode	Performs monitoring by using different colors in positions where the program is executed and in positions where the program is not executed.
Display Mode	Allows the user to select binary, decimal, or hexadecimal as the display format of the variable value to be displayed.

Tools

Item	Function
PANATERM Lite for GM	Allows the user to select a device to which PANATERM Lite for GM is to connect.
Library Repository	Allows the user to install a created library in the library repository in order to use the functions or function blocks in the library.
Options	Allows the user to set up each function of GM Programmer.

Window

Item	Function
Next Editor	Displays the next window.
Previous Editor	Displays the previous window.
Close All Editors	Closed all windows.
Reset Window Layout	Resets the layout of the window to its initial state.
New Horizontal Tab Group	Moves the selected window downward.
New Vertical Tab Group	Moves the selected window to the right.
Float	Sets the selected window in a floating state.
Dock	Sets the selected window in a docking state.
Auto Hide	Minimizes the window.
Next Pane	Switches the pane between the declaration section (first pane) and the implementation section (second pane).
Previous Pane	Switches the pane between the declaration section (first pane) and the implementation section (second pane).
Window	Displays a list of open windows.

Help

Item	Function
Manual	Displays the CODESYS manual.
About	Displays version information.

3.4.2 Toolbar

The toolbar displays the following icons:

Name	Icon	Function
New Project		Creates a new project.

Name	Icon	Function
Open Project		Opens a project that is stored.
Save Project		Saves the project that is currently viewed, in overwrite mode.
Print		Prints the active editor screen.
Undo		Reverses the results of a previous editing action.
Redo		Allows the user to redo the last editing action after Undo.
Cut		Cuts data.
Copy		Copies data.
Paste		Pastes data.
Delete		Deletes data.
Find		Searches for a particular character string that appears in the active editor.
Find Replace		Searches for a particular character string that appears in the active editor and replaces it with another character string.
Find in Project		Searches for a specified character string within the current project.
Replace in Project		Searches for a specified character string within the current project and replaces it with another character string.
Toggle Bookmark		Saves the position selected in the active editor as a bookmark.
Previous Bookmark (active editor)		Moves to the previous bookmark in the active editor.
Next Bookmark (active editor)		Moves to the next bookmark in the active editor.
Clear All Bookmarks (active editor)		Removes all bookmarks in the active editor.
Function Block Guidance		Displays the Function Block Guidance.

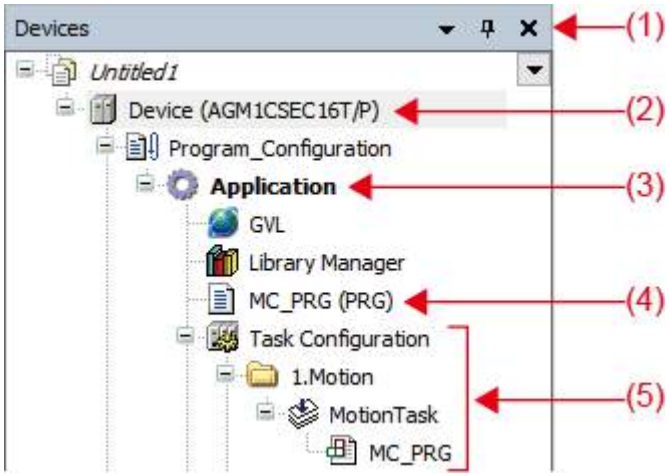
Name	Icon	Function
Properties		Displays the properties.
Add Object		Adds an object.
Edit Object		Opens an object.
Build		Compiles an object in the application.
Login		Downloads the applications generated by code generation to the GM1 controller at the time of login.
Logout		Logs out from the device to which the user logged in.
Start		Starts the application.
Stop		Stops the application.
Online Config Mode		Removes the applications downloaded to the GM1 controller and allows connection to the GM1 controller.
Step Over		Executes the program line by line. When a block (function or function block) is executed in the block invocation location, the cursor moves to the next line.
Step Into		Executes the program line by line. When a block (function or function block) is executed in the block invocation location, the cursor moves to the first line of the called block.
Step Out		When the program is executed within the called block, the execution continues until control returns to the calling block. When the program is executed outside the called block, the execution continues until control returns to the beginning of the program.
Run to Cursor		Executes the program up to the line specified by the cursor.
Set next Statement		Regards the line specified by the cursor as the next statement to be executed and skips processes over to that line.
Show next Statement		Jumps the cursor to the program line to be executed as the next step.
Toggle Localization		Switches the language to the one enabled in [Default Localization] in the window displayed by selecting [Project

Name	Icon	Function
		Localization] and then [Manage Localizations] from the [Project] menu.
Commit accepted changes		Commits the difference between the objects compared by selecting the [Project] > [Compare] from the menu bar.
Insert Network		Inserts an empty network.
Toggle network comment state		Changes the comment status of the selected network.
Insert Assignment		Inserts a new assignment in the specified position.
Insert Coil		Inserts a coil in the specified position.
Insert Set Coil		Inserts a set coil in the specified position.
Insert Reset Coil		Inserts a reset coil in the specified position.
Insert Contact		Inserts a normally open contact in the specified position.
Insert Negated Contact		Inserts a normally closed contact in the specified position.
Insert Contact (right)		Inserts a normally open contact on the right side of the specified position.
Insert Contact Parallel (below)		Inserts a normally open contact below and in parallel with the contact at the specified position.
Insert Negated Contact Parallel (below)		Inserts a normally closed contact below and in parallel with the contact at the specified position.
Insert Contact Parallel (above)		Inserts a normally open contact above and in parallel with the contact at the specified position.
Insert Box		Opens the Input Assistant to insert a box in the specified position.
Insert Empty Box		Inserts an empty box in the specified position.
Insert Box with EN/ENO		Opens the Input Assistant to insert a box with EN/ENO in the specified position.
Insert Empty Box with EN/ENO		Inserts a box with EN/ENO in the specified position.

Name	Icon	Function
Insert Jump		Inserts a jump in the specified position.
Insert label		Inserts a label in the selected network.
Insert Return		Inserts a return value in the specified position.
Insert Input		Adds an input to the specified box.
Negation		Adds a negation to the selected element.
Edge Detection		Adds an edge detection (rising edge detection) to the selected element.
Set/Reset		Converts the selected coil to a set coil or reset coil.
Set output connection		Converts box output to forwarding box output.
Insert Branch		Inserts a branch on the right side of the selected contact.
Insert Branch below		Inserts a new branch below the selected branch.
Insert Branch above		Inserts a new branch above the selected branch.
Set Branch Start/End Point		Sets the selected line as the branch starting point.

3.4.3 Navigator Pane

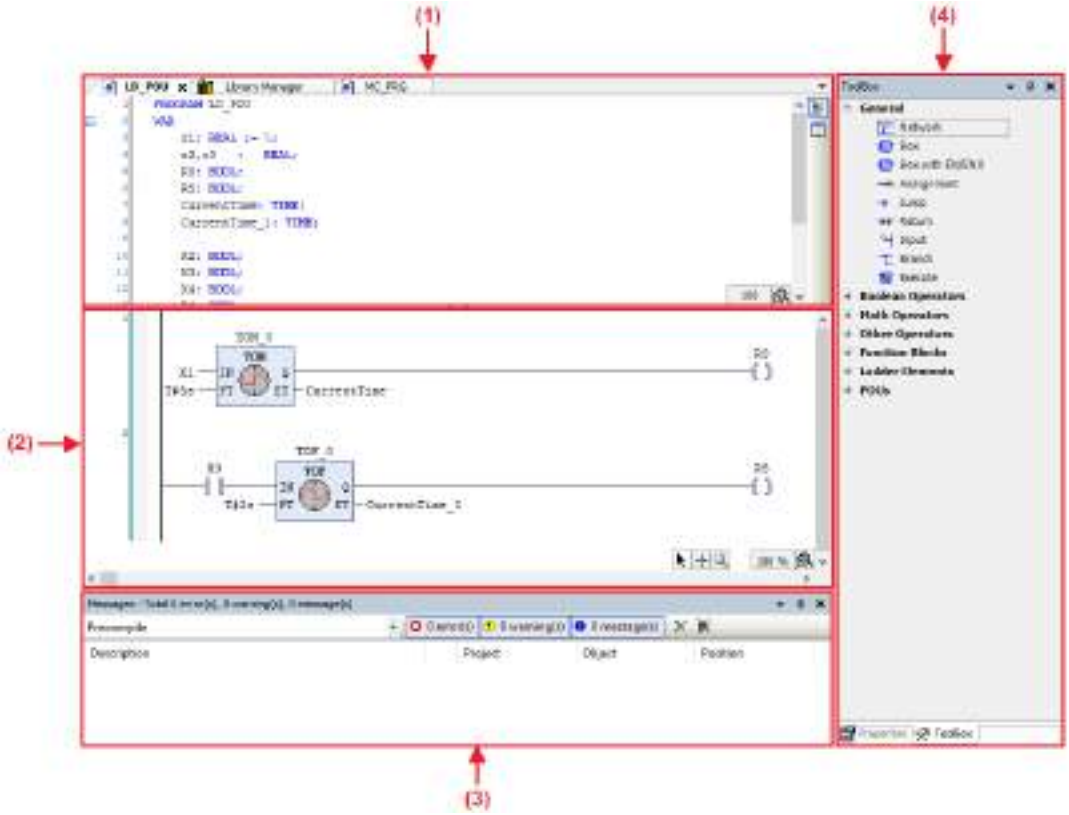
The navigator pane displays the following tree:



No.	Name	Icon	Function
(1)	Window Position		• New Horizontal Tag Group Moves the selected window to the right. • New Vertical Tag Group Moves the selected window downward. • Float Sets the selected window in a floating state. • Dock Sets the selected window in a docking state. • Auto Hide Minimizes and hides the navigator pane.
	Auto Hide		Always shows the navigator pane.
			Minimizes and hides the navigator pane.
	Close		Closes the navigator pane.
(2)	Device object		Sets up device objects.
(3)	Application object		Sets up application objects.
(4)	Program object (POU object)		Sets up program objects (POU objects).
(5)	Task object		Sets up task objects.

3.4.4 Main Pane

The main pane displays the following sub-panes:



No.	Name	Function
(1)	Declaration section (first pane)	Allows the user to declare variables.
(2)	Implement section (second pane)	Allows the user to enter a program.
(3)	Message view	Displays any error or warning messages.
(4)	ToolBox	Allows the user to place elements in the implementation section by selecting them and then dragging and dropping them in the implementation section.

3.4.5 Status Bar

The status bar displays the following icons:

Name	Icon	Function
Last Build		Displays the number of errors in the results of the build process.
		Displays the number of warnings in the results of the build process.
Precompile	—	Displays the results of the precompile process.

Name	Icon	Function
Application Information		Compares the application information of the displayed project with the application information downloaded to the GM1 controller.
Project user	—	Displays the users who are logged in to the displayed project.

3.5 Window Operations

This section explains operations related to common windows for GM Programmer.

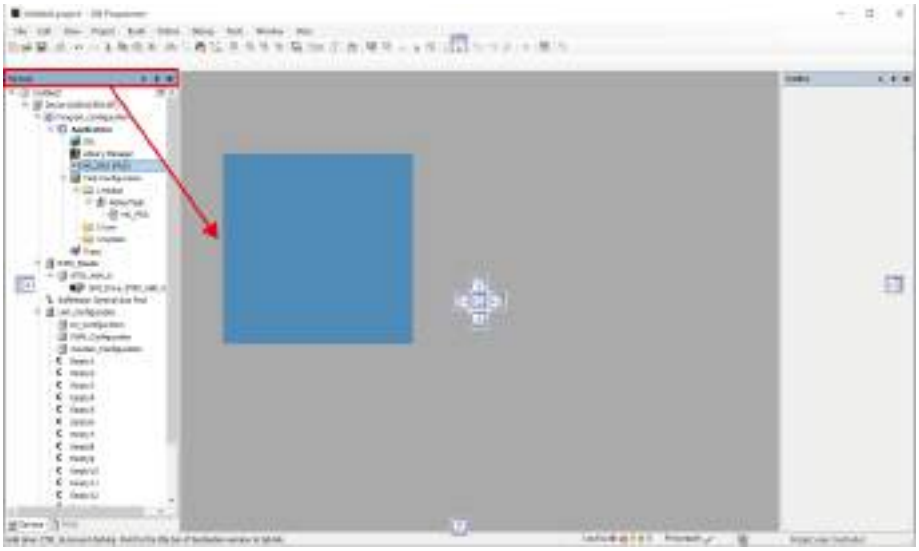
3.5.1 Moving the Pane Location

You can freely change the layout of each window for GM Programmer.

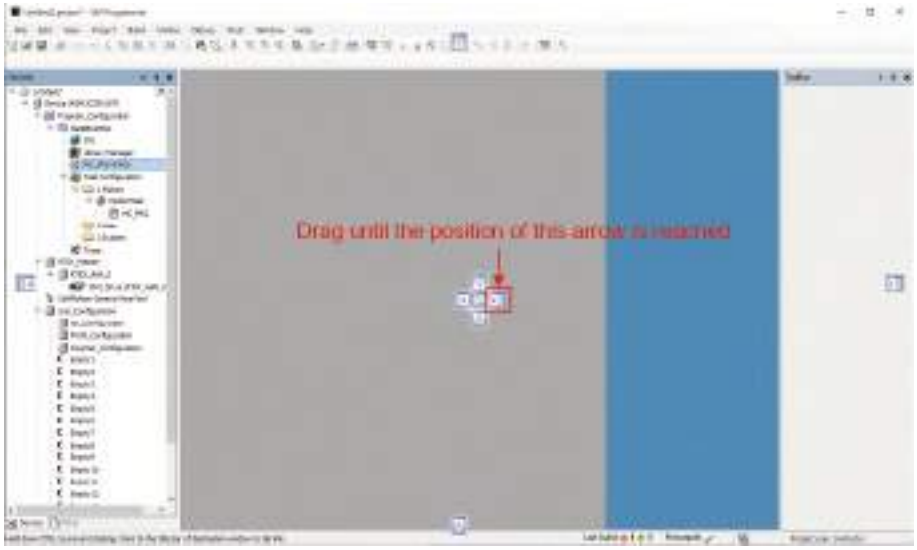
For example, use the following procedure to move the navigator pane from the left edge to the right edge of the window.

1 2 Procedure

1. Click the title bar of the navigator pane and then drag it to the main pane.
   The navigator pane will stay in a floating state and arrows indicating movable directions will be displayed.

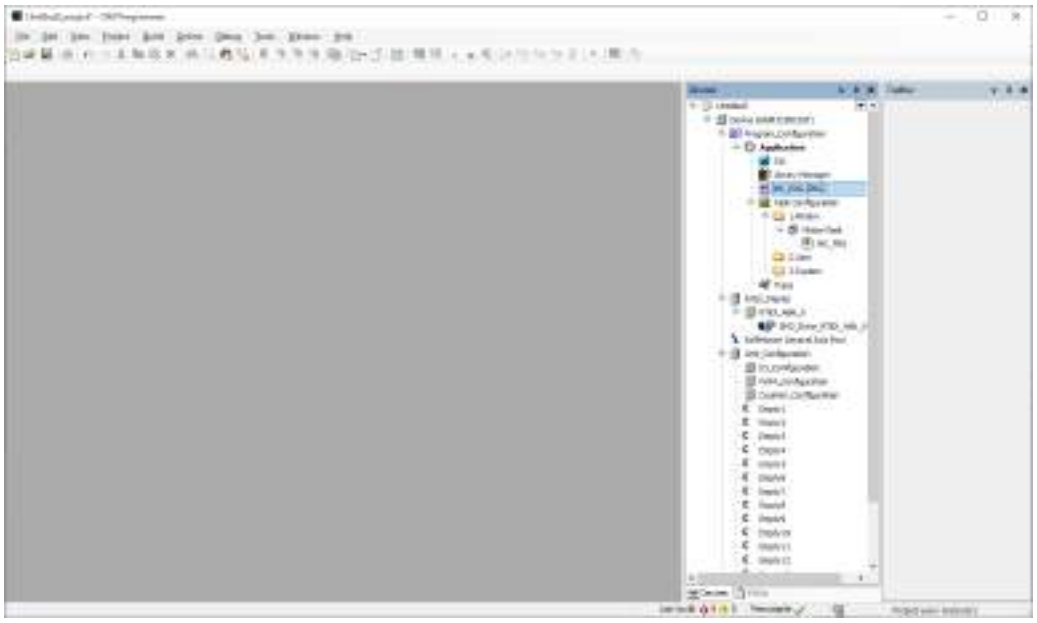


2. Drag the navigator pane in the direction in which you want to move it.
   The relocation destination will be displayed in light blue.



3. Release the left mouse button.

The navigator pane will be docked into the existing pane and the relocation will be completed.



Info.

- You can return the changed layout of the window to its initial state.

  From the menu bar, select **Window>Reset Window Layout**.
- You can put a pane in a floating or docking state. To put a pane in a floating state, select **Window>Float** from the menu bar. To put a pane in a docking state, right-click the title bar in the window in a floating state and then select Dock from the context-sensitive menu that is displayed.

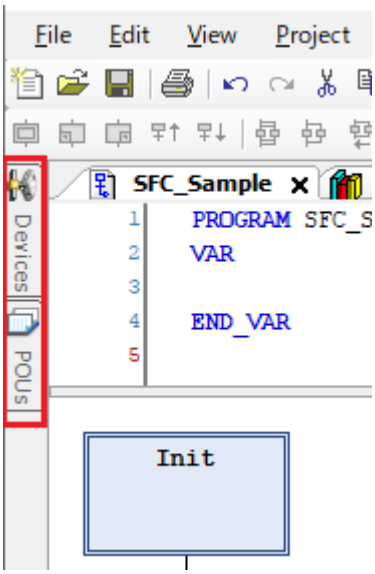
3.5.2 Showing / Hiding Panes

You can normally hide the navigator pane and some sub-panes in the main pane and show them only when you use them.

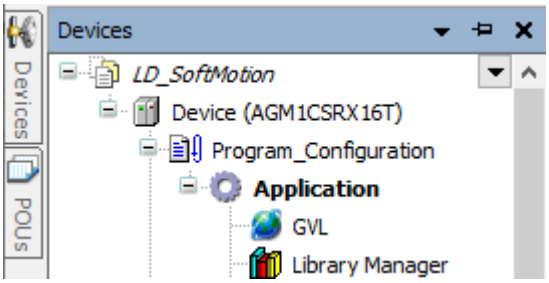
■ **Showing / hiding the navigator pane**

Procedure

1. Click on the title bar of the navigator pane.
   The navigator pane will be minimized and hidden.



2. Click the minimized pane.
   The navigator pane will be displayed. Clicking in another pane automatically hides the navigator pane again.



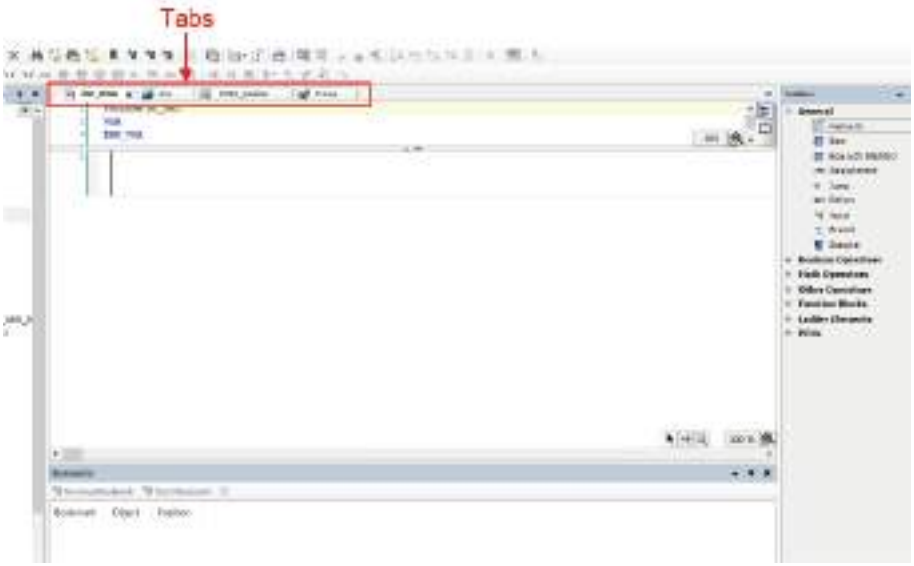
3. Click on the title bar of the navigator pane.
   The navigator pane will always be displayed.

Info.

- You can also hide the navigator pane from the menu bar. From the menu bar, select **Window>Auto Hide**. To always display the navigator pane again, select **Window>Auto Hide** again from the menu bar.

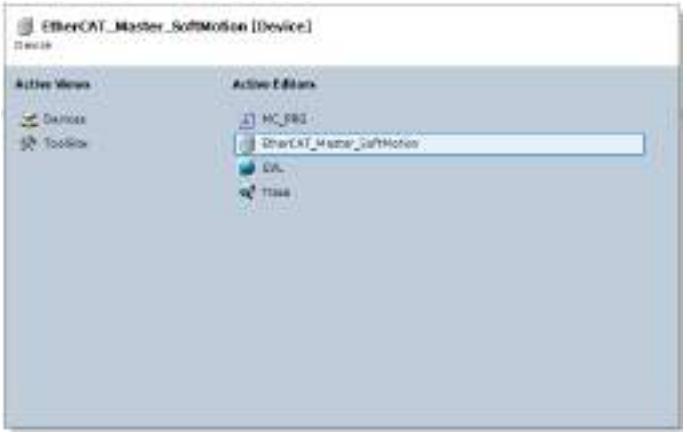
3.5.3 Switching the Tab of the Main Pane

You can switch the tab of the main pane.

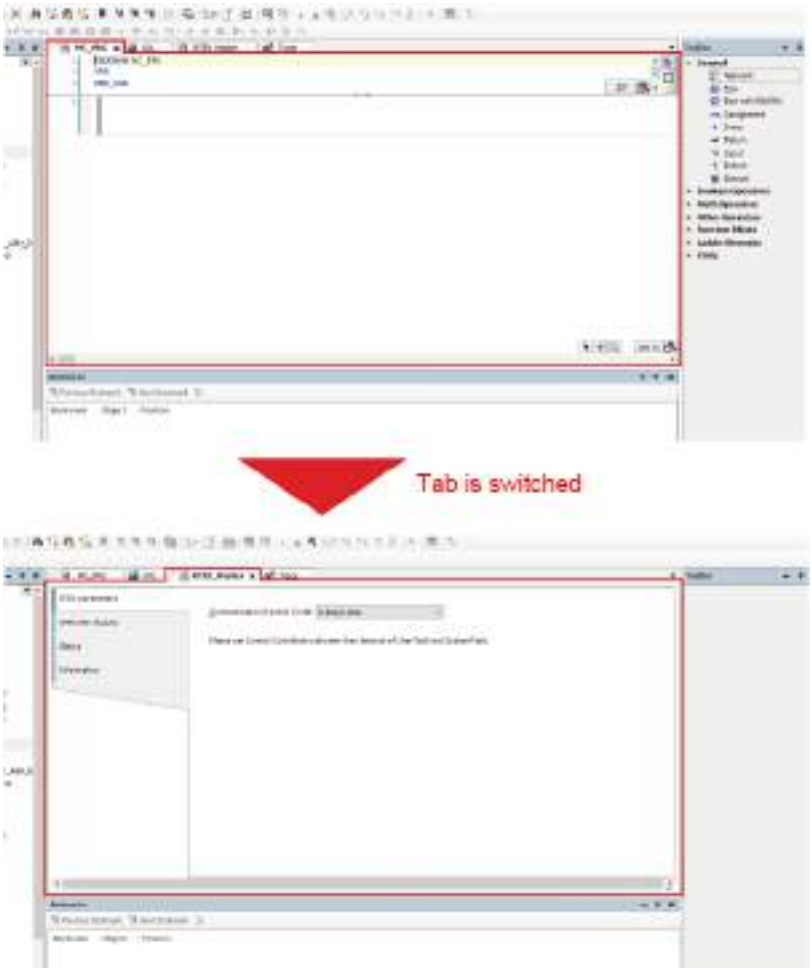


Procedure

1. Press the shortcut keys "Ctrl+Tab" simultaneously.
   The window for switching the tab of the main pane will be displayed.



2. While holding down the "Ctrl" key, press the "Tab" key until the desired tab is selected.
3. Release the "Ctrl" key.
   The current tab will be switched to the selected tab.



3.5.4 Full-screen Display

You can display each window of GM Programmer in full-screen mode.

Procedure

1. From the menu bar, select **View**>**Full Screen**.
   Then GM Programmer window will be displayed in full-screen mode.
2. From the menu bar, select **View**>**Full Screen** again.
   Then GM Programmer window will return from full-screen mode to normal display mode.

Info.

- You can also switch to full-screen mode by pressing shortcut keys "Ctrl+Shift+F12" simultaneously.

3.6 Switching the Object Window

Double-clicking an object added to the navigator pane displays its window in the main pane.
You can open multiple objects in the main pane and switch to each of their windows using the tab.

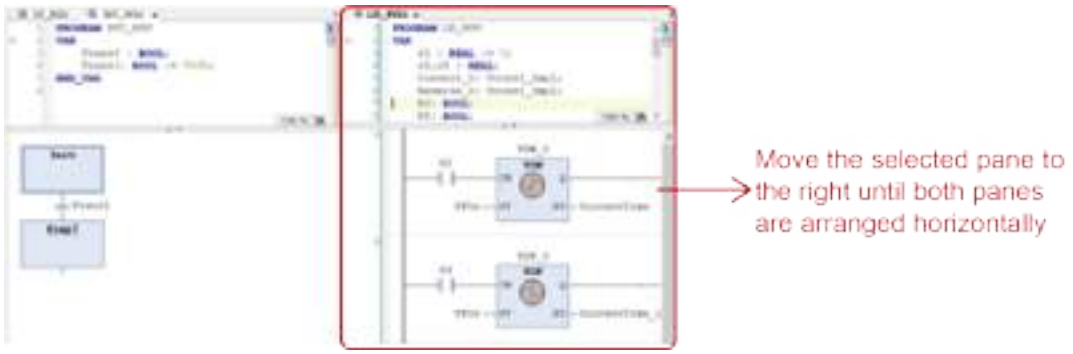
3.6.1 Operating the Object Window

The following operations can be performed on the object window displayed in the main pane.

Operation	Menu	Shortcut keys
Displaying the next window	**Window>Next Editor**	<Ctrl> + <F6>
Displaying the previous window	**Window>Previous Editor**	<Ctrl> + <Shift> + <F6>
Closing all windows	**Window>Close All Editors**	None
Moving the selected window downward	**Window>New Horizontal Tab Group**	None
Moving the selected window to the right	**Window>New Vertical Tab Group**	None

<Moving the selected window downward (New Vertical Tab Group)>

<Moving the selected window to the right (New Horizontal Tab Group)>



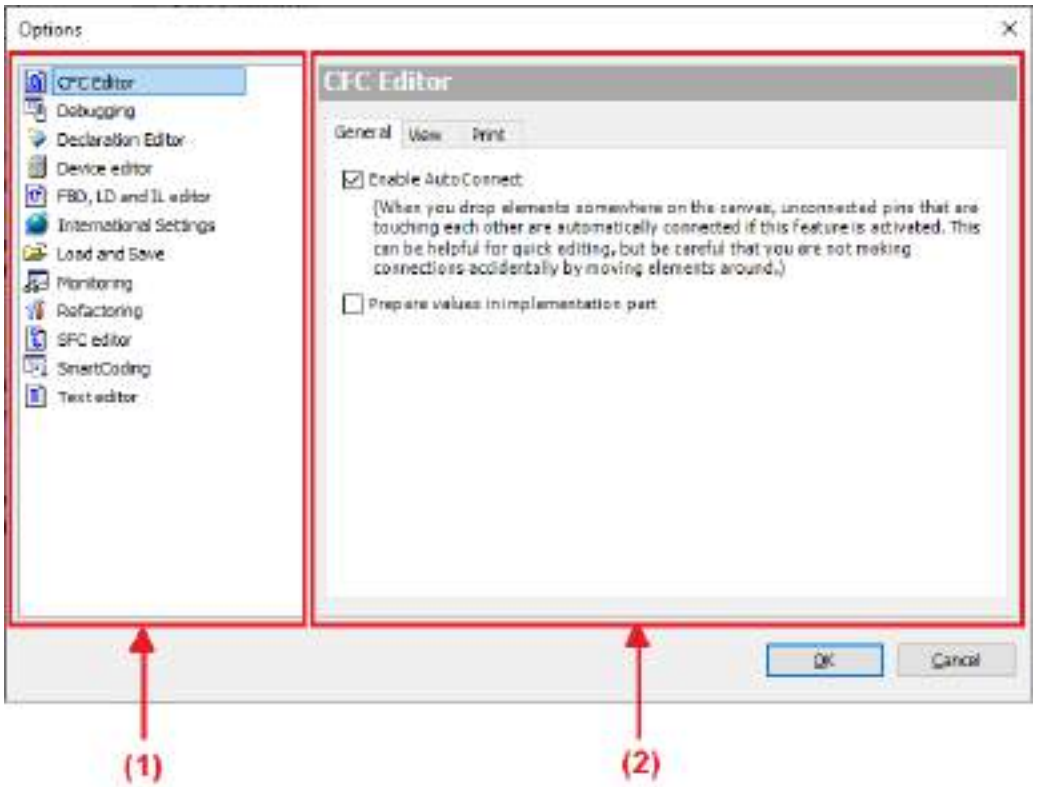
3.7 Other Functions

3.7.1 Option Setting Function

GM Programmer allows the user to set up each function from the "Options" dialog box. The settings will be applied to all projects created with GM Programmer.

Procedure

1. From the menu bar, select **Tools**>**Options**.
   The "Options" dialog box will be displayed.



No.	Name	Function
(1)	Categories pane	Displays option categories.
(2)	Setting pane	Displays the settings of the selected category and allows the user to configure settings.

Option categories

Category name	Function	Reference page
CFC Editor	Allows the user to configure settings related to editing and printing CFC programs.	-

Category name	Function	Reference page
FBD, LD and IL editor	Allows the user to configure settings related to editing, commenting, and printing FBD, LD, and IL programs.	"P.7-11" "P.7-32"
SFC editor	Allows the user to configure settings related to the sizes and fonts of SFC editor elements, the behavior at the time of action element insertion, the display of embedded objects in the navigator pane, the display of properties, stepwise execution time during online operation, and other items.	"P.7-23"
SmartCoding	Allows the user to configure settings related to the functions for supporting program creation, such as Input Assistant.	"P.7-15" "P.6-5" "P.7-51"
Text editor	Allows the user to configure settings related to program editing and inline monitoring.	"P.9-15"
Device editor	Allows the user to configure settings related to displays for the device editor.	-
Debugging	Allows the user to configure settings regarding whether to restore breakpoints after resetting.	-
Monitoring	Allows the user to configure settings related to displays for monitoring.	-
Refactoring	Allows the user to configure settings for the valid range of refactoring.	"P.7-52"
Load and save	Allows the user to configure settings regarding whether to enable backup and auto saving of project files.	"P.4-25" "P.4-27"
International Settings	Allows the user to set a display language for GM Programmer and PANATERM Lite for GM, as well as a display language for the manual.	"P.3-33"
Declaration Editor	Allows the user to configure settings related to the display format (text format or table format) for the declaration section.	"P.6-3"

2. Select a desired category from the Categories pane.
   The setting items for the selected category will be displayed in the setting pane.
3. Change the setting items as appropriate and click the [OK] button.
   The setting items will be applied.

3.7.2 Display Language Setting Function

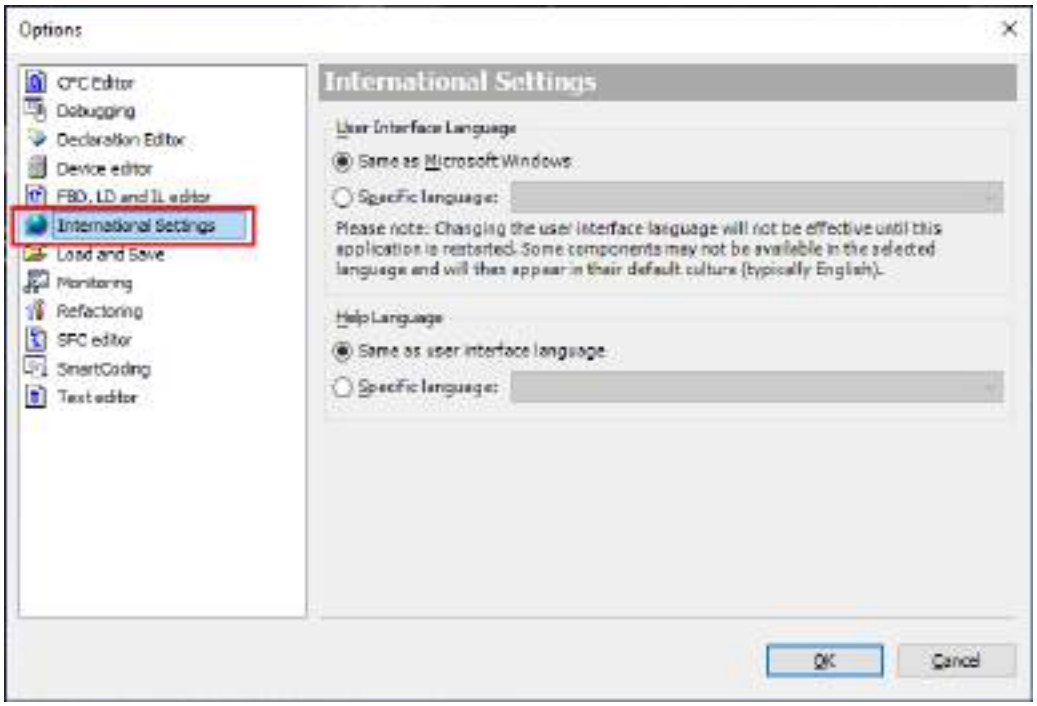
This function allows the user to change the display language setting for GM Programmer.
The default setting is the same language as the one used in the operating system. If you want to use a different language from the one used in the operating system, change the display language setting. After you change the language setting, you must restart GM Programmer.

Procedure

1. From the menu bar, select **Tools**>**Options**.

The "Options" dialog box will be displayed.

2. Select "International Settings" from the Categories pane.
The "International Settings" pane will be displayed.



3. Select **User Interface Language**>**Specific language** option and specify a desired language in the field.

4. Click [OK].
The "Options" dialog box will be closed.
At this stage, the language has not been changed yet.

5. Close GM Programmer and then start GM Programmer again.
After GM Programmer is started, the selected language takes effect.

Info.

- The display language setting of GM Programmer is linked with that of PANATERM Lite for GM. Therefore, if the display language setting of PANATERM Lite for GM is changed, the display language setting of GM Programmer will also be changed automatically.

3.7.3 Online Help Function

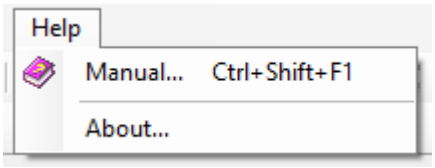
This function allows the user to open the manual and check information such as operating methods.

Procedure

1. Press the [F1] key.
   Online help will be started and the page corresponding to the displayed window will be displayed.

Info.

- You can also start online help by selecting **Help**>**Manual** from the menu bar.



3.7.4 Version Display Function

This function allows the user to check the version, license, and other information for GM Programmer.

1 2 Procedure

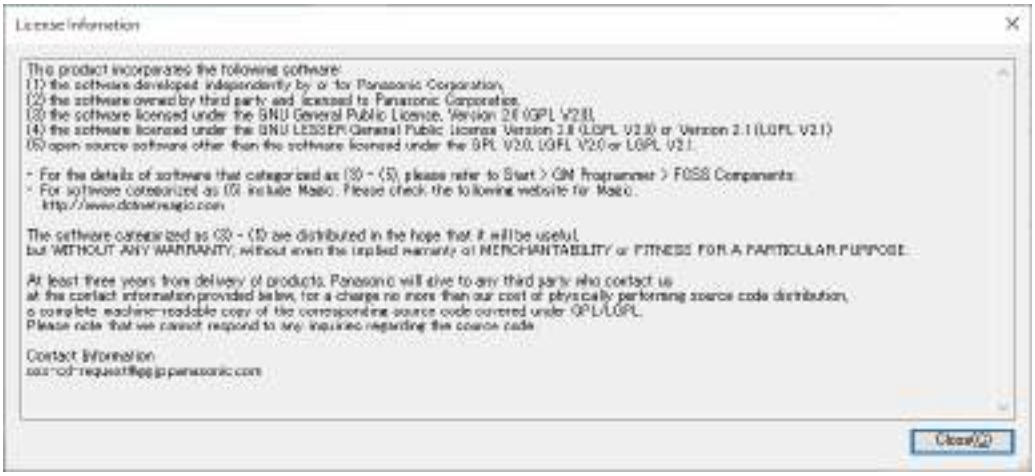
1. From the menu bar, select **Help**>**About**.

2. Click a desired button at the bottom of the window.

Button	Description
Version Info	Displays information about the plug-ins that have been applied and the operating system of the PC that is used.
License Info	Displays license information for the software used by GM Programmer.

Clicking the [Version Info] button displays the "Version Info" dialog box.

Clicking the [License Info] button displays the "License Information" dialog box.



(MEMO)

4 Project Operations

4.1 Creating a New Project 4-2
4.2 Saving a Project 4-5
4.3 Opening a Project 4-6
4.4 Closing a Project 4-7
4.5 Device Tree Configuration 4-8
4.6 Project Configuration 4-11
4.7 Adding an Object 4-12
4.7.1 Adding Objects 4-13
4.7.2 Adding Devices 4-17
4.8 Setting up A Project 4-19
4.9 Exporting and Importing Objects 4-21
4.9.1 Exporting Objects 4-21
4.9.2 Importing Objects 4-22
4.10 Creating a Backup when a Project Is Saved 4-25
4.11 Automatically Saving Project Files 4-27
4.12 Printing a Project 4-29
4.13 Printing an Object within a Project 4-31
4.14 Comparing Projects 4-32
4.14.1 Project Comparison Method 4-32
4.14.2 Merging Differences 4-35

4.1 Creating a New Project

When creating a program using GM Programmer for the first time, create a new project. For the new project, set a device and a programming language to be used.

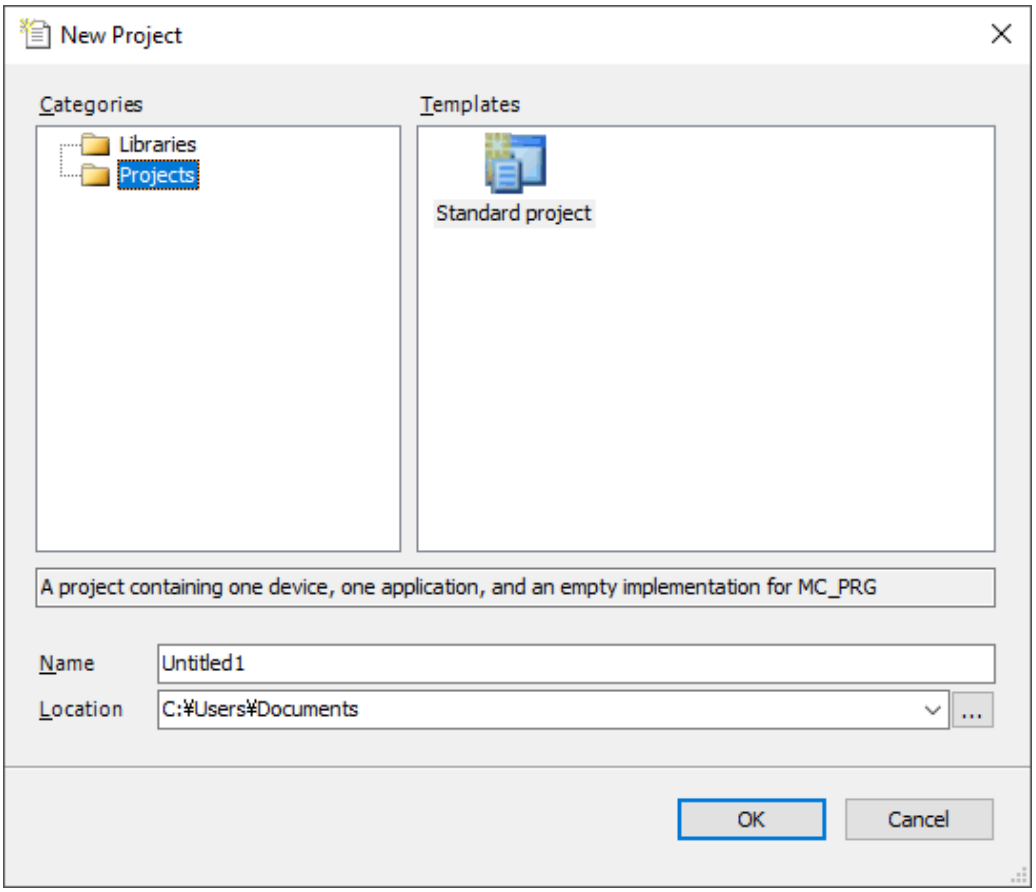
This section describes how to create a new project.

Given below is an example that explains the procedure to create a project for the GM1 Controller (product number: AGM1CSEC16T/P) in Structured Text (ST) format.

Procedure

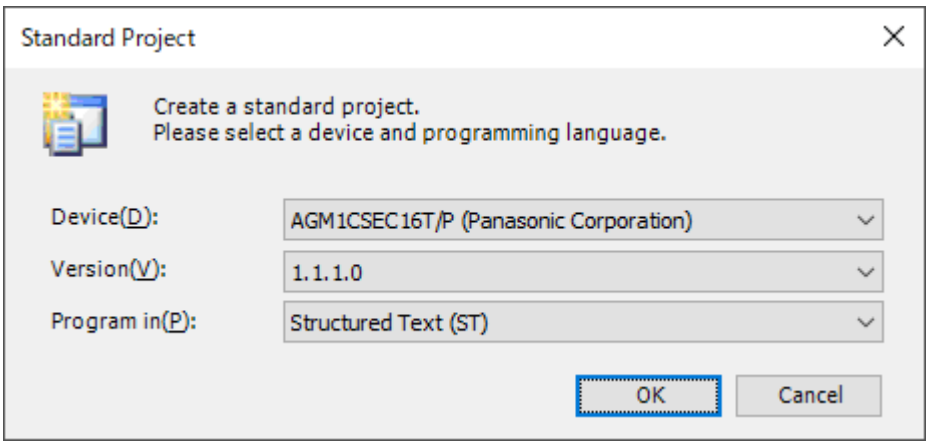
1. Start GM Programmer.
   For details on how to start GM Programmer, refer to "3.3.1 How to start".
   When GM Programmer is started, the Start Page will be displayed.

2. Select "New Project" under "Basic Operations".
   The "New Project" dialog box will be displayed.



3. Select **Projects>Standard project**, and specify a project file name in the "Name" field and a project storage location in the "Location" field.

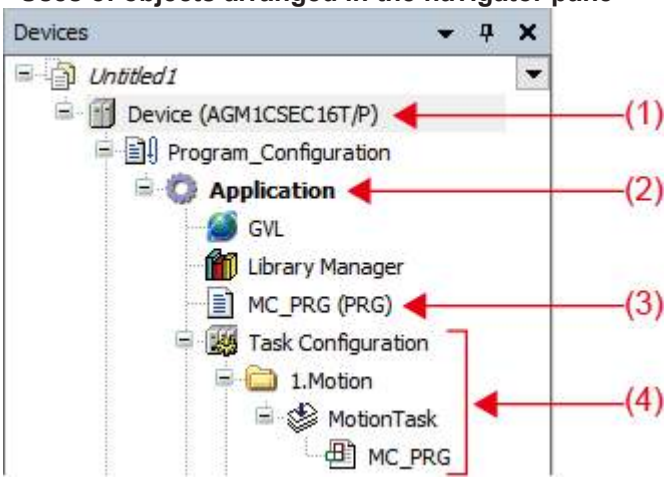
4. Click the [OK] button.
   The "Standard project" dialog box will be displayed.



5. Select "AGM1CSEC16T/P(Panasonic Corporation)" in the "Device" field and "Structured" in the "Program in" field, and click the [OK] button.

A new project will be created. Device and other objects including objects for ST programs are arranged in the navigator pane.

<Uses of objects arranged in the navigator pane>



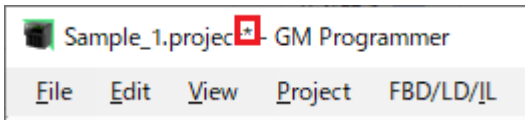
No.	Name	Function
(1)	Device object	Sets up device objects.
(2)	Application object	Sets up application objects.
(3)	Program object (POU object)	Sets up program objects (POU objects).
(4)	Task object	Sets up task objects.

Info.

- A new project can also be created from the menu bar by selecting **File**>**New Project**.

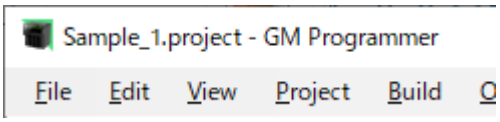
4.2 Saving a Project

Save a project that is created. The project will be saved as a file with extension ""*.project"". Unsaved projects are indicated by "*" on the right side of their project file names on the title bar.



Procedure

1. From the menu bar, select **File>Save Project**, or press the shortcut keys "Ctrl+s".
   A project that has been created will be saved.
   ""*"" displayed on the right side of the project file name will disappear.



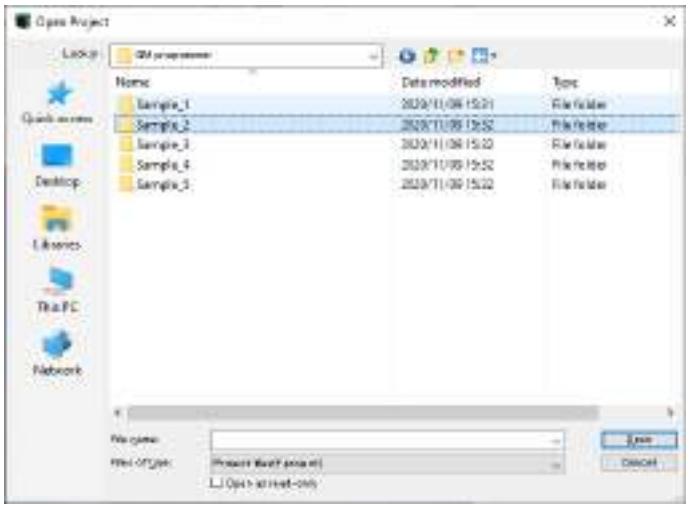
Info.

- Before saving a project, you can change its project name. From the menu bar, select **File>Save Project As**.
- Project files can be saved automatically. For details, refer to "4.11 Automatically Saving Project Files".
- Before updating a file, you can save it as a backup file. For details, refer to "4.10 Creating a Backup when a Project Is Saved".

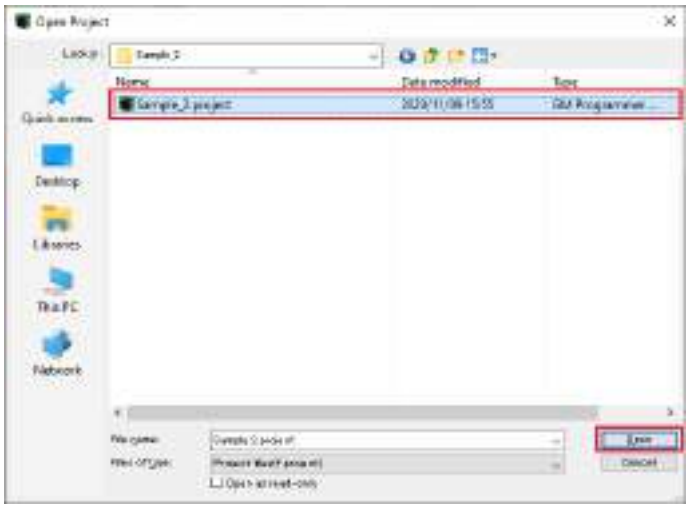
4.3 Opening a Project

Procedure

1. From the menu bar, select **File**>**Open Project**.
   The "Open Project" dialog box will be displayed.



2. Select a project file and click the [Open] button.
   The selected project file will be opened.



4.4 Closing a Project

Procedure

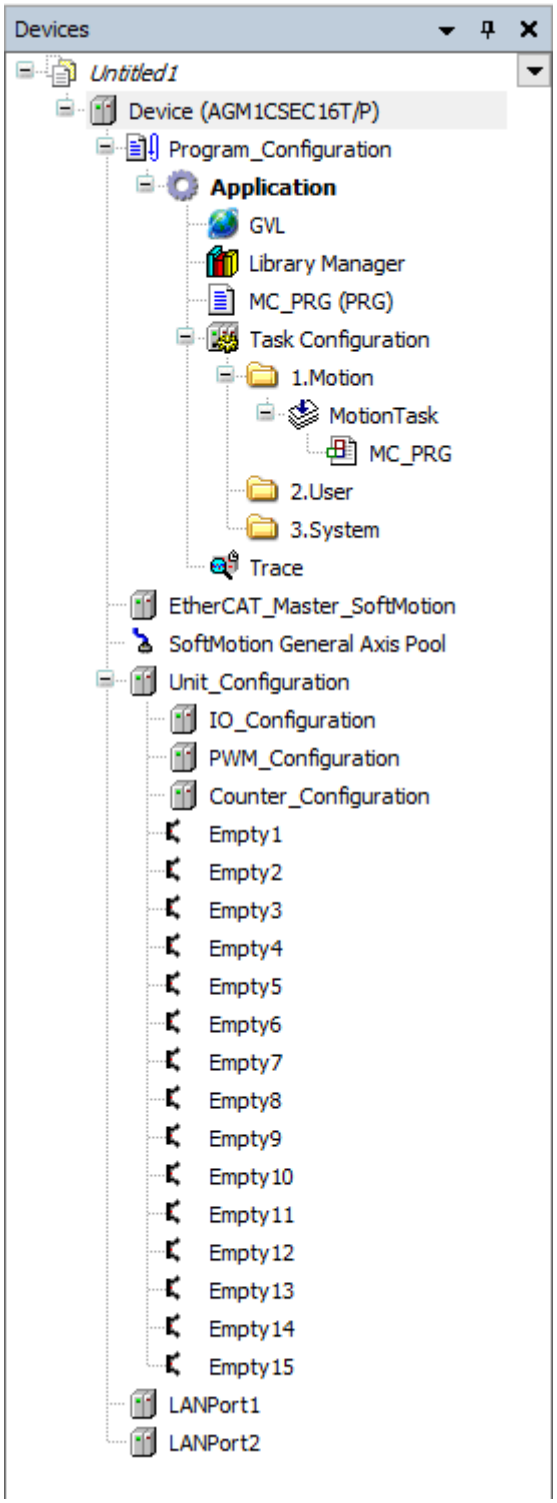
1. From the menu bar, select **File**>**Close Project**.
   The project that has been created will be closed.

Info.

- If you select "Close Project" without saving a project file that has been updated, a confirmation dialog box will be displayed, asking whether to save the project. Click the [Yes] button to save the project.

4.5 Device Tree Configuration

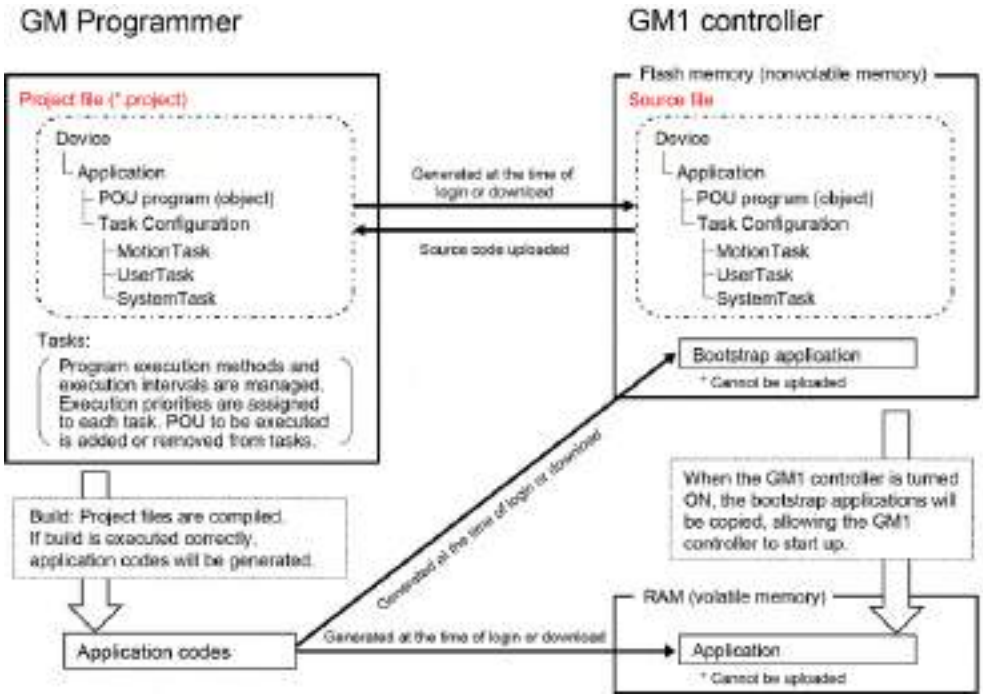
When a new project is created, it is started in the device tree configuration shown below.



Sample1	Project name

Device	GM1 controller device
Program Configuration	Object that defines applications including source codes
Application	Application object
GVL	Global variable object
Library_Manager	List of all libraries linked to the project
MC_PRG(PRG)	Main program
Task Configuration	Configuration of tasks invoking application programs
1.Motion	User program tasks for motion control
MotionTask	
MC_PRG	
2.User	User program task for any control other than motion control
3.System	Task used by the system
Trace	Object that monitors variable data graphically
EtherCAT_Master_SoftMotion	Object that is the parent node of servo amplifiers for EtherCAT
SoftMotion General Axis Pool	Object that is an interface for inserting a free drive unit
Unit_Configuration	Object that is the parent node of I/O related devices
IO_Configuration	General-purpose I/O incorporated in GM1 controller
PWM_Configuration	PWM output incorporated in GM1 controller
Counter_Configuration	High-speed counter incorporated in GM1 controller
Empty1	Objects for adding I/O for expansion unit
Empty2	
Empty3	
Empty4	
Empty5	
Empty6	
Empty7	
Empty8	
Empty9	
Empty10	
Empty11	
Empty12	
Empty13	
Empty14	
Empty15	
LAN Port1	Objects that are the parent node of devices that use the Ethernet protocol
LAN Port2	

4.6 Project Configuration



4.7 Adding an Object

Check the following list for objects that can be added.

Addition source	Added object
Device	DeviceTrace
Application	CNC program
POU	CNC setting DUT POU Interface Cam table Global variable list Trace External file Persistent variable POU for implicit checks
POU	Action Transition Property Methods
Interface	Interface property Interface method
Global Variables	Property
Persistent variable	Property
MotionTask	Program call
"2.User" folder	User task

Check the following list for devices that can be added.

Addition source	Added device
Device	Modbus COM
EtherCAT_Master_SoftMotion	Various EtherCAT slaves can be selected.
SoftMotion General Axis Pool	SM_FreeEncoder SM_Drive_Virtual
Unit_Configuration or Empty1 to Empty15	AGM1X64D2 AGM1Y64T AGM1Y64P AGM1XY64D2T AGM1XY64D2P AGM1AD8 AGM1DA4 AGM1PG04T AGM1PG04L
LAN Port1	Modbus TCP Master ModbusTCP Slave Drive

Addition source	Added device
LAN Port2	Modbus TCP Master ModbusTCP Slave Drive EtherNet/IP Scanner EtherNet/IP Adapter
Modbus TCP Master	Modbus TCP Slave
EtherNet/IP Scanner	Remote adapter for each device can be selected
EtherNet/IP Adapter	EtherNet/IP Module
Modbus COM	Modbus Serial Device Modbus Master, COM Port
Modbus Master, COM Port	Modbus Slave COM Port

4.7.1 Adding Objects

Program creation objects (POU objects) and objects with various functions can be added to applications within a project.

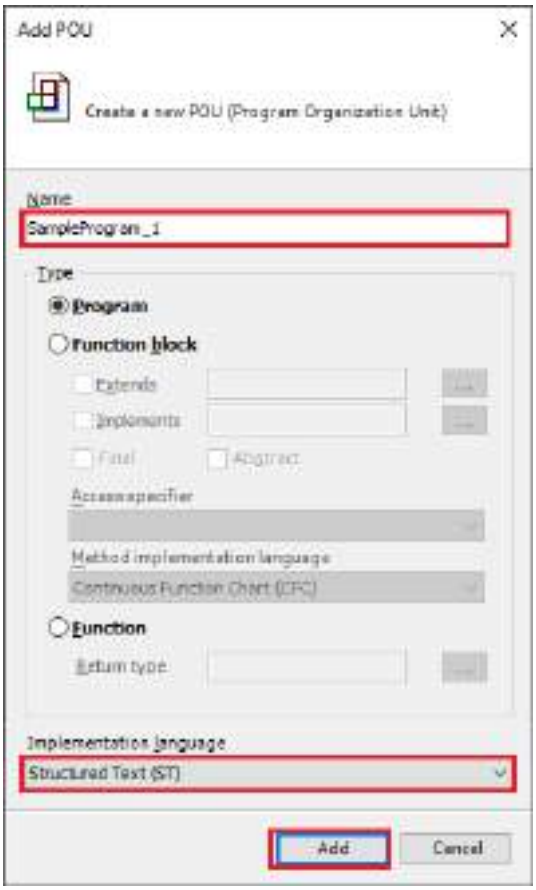
For example, use the following procedure to add POU objects for ST programs.

Procedure

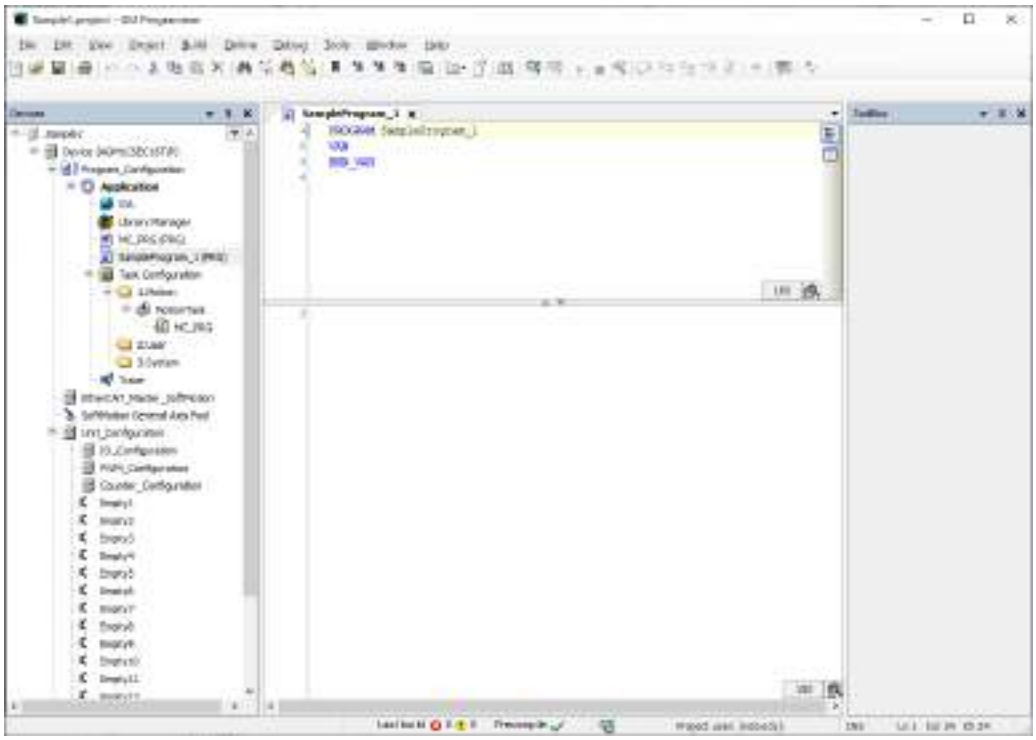
1. Right-click the [Application] object in the navigator pane and then select **Add Object**>**POU** from the context-sensitive menu that is displayed.

The "Add POU" dialog box will be displayed.

2. Enter a program name in the "Name" field, select a programming language from the "Implementation Language" drop-down list, and click the [Add] button.



A [POU] object for the programming language selected in the "Implementation Language" drop-down list will be added to the navigator pane.

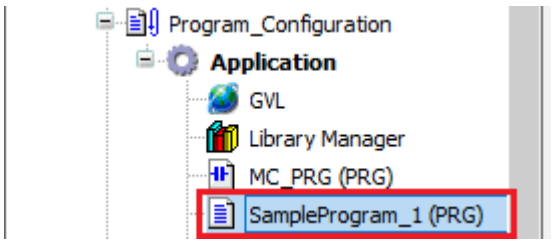


Info.

- You can also add objects by selecting the [Application] object in the navigator pane and then selecting **Project>Add Object** from the menu bar.
- To remove an object, select the object in the navigation pane and then press the "Delete" key or right-click the object and then select "Delete" from the context-sensitive menu that is displayed.
- You can add folders under the [Application] object in the navigation pane. By adding objects under each folder, you can create a hierarchical structure.

  Right-click the [Application] object in the navigator pane and then select "Add Folder" from the context-sensitive menu that is displayed. The "Add Folder" dialog box will be displayed. Enter a folder name and click the [OK] button.

  Example: When a [POU] object is added under an added folder (program for project A)



- You can also add an object for creating functions and function blocks. For details, refer to "6.6 Function and Function Block".

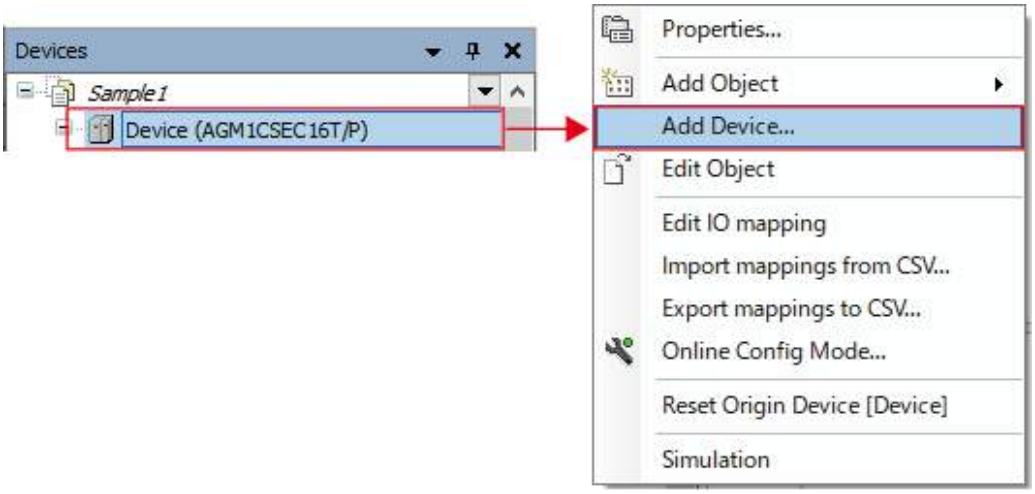
4.7.2 Adding Devices

You can add a device to the devices within the project.

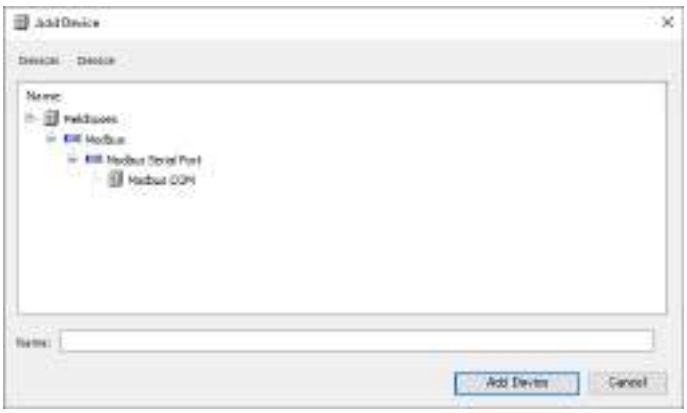
For example, use the following procedure to add a Modbus COM device to Device (AGM1CSEC16T).

Procedure

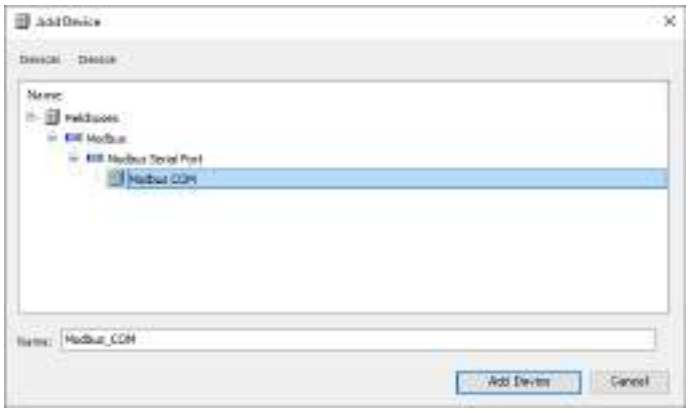
1. Right-click "Device (AGM1CSEC16T)" in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.



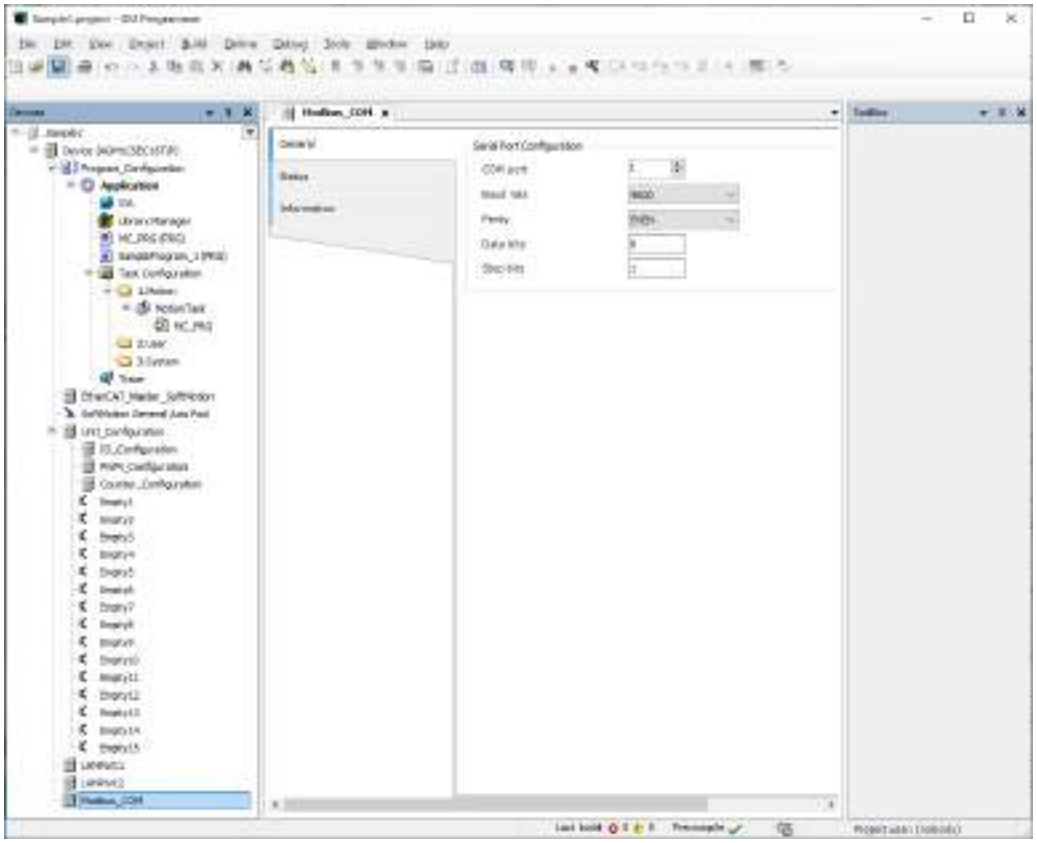
The "Add Device" dialog box will be displayed.



2. Select "Modbus COM" and click the [Add Device] button.



The "Modbus COM" object will be added to the navigator pane.

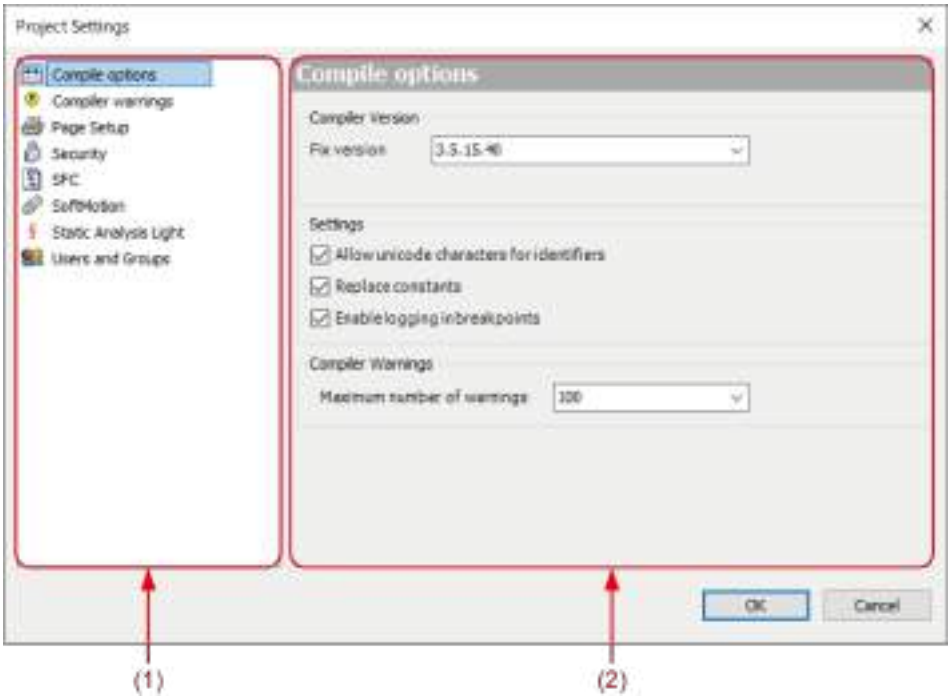


4.8 Setting up A Project

Projects can be set up using the "Project Settings" dialog box. The settings will be applied to only the project that is set up.

Procedure

1. From the menu bar, select **Project**>**Project Settings**.
   The "Project Settings" dialog box will be displayed.



No.	Name	Function
(1)	Categories pane	Displays project setting categories.
(2)	Setting pane	Displays the settings of the selected category and allows the user to configure settings.

Project setting categories

Category name	Function	Reference page
SFC	Allows the user to configure settings related to variables used in SFC and code generation.	"P.7-23"
SoftMotion	Displays the version of the SoftMotion package.	"P.5-11"
Sraric Analysis Light	Allows the user to configure settings regarding whether to enable code analysis during code generation.	-
Compiler warnings	Allows the user to configure settings regarding whether to enable warnings output during build.	"P.10-40"

Category name	Function	Reference page
Compile options	Allows the user to configure build-related settings such as whether to enable Unicode for the program objects (POU objects) and the number of compiler warnings to be output.	"P.9-6"
Security	Allows the user to configure settings related to project file encryption using passwords.	"P.10-16"
Page Setup	Allows the user to configure printing-related settings.	-
Users and groups	Allows the user to add, edit, and remove users and groups for project user management.	"P.10-4"

2. Select a desired category from the category pane.
   The setting items for the selected category will be displayed in the setting pane.
3. Change the setting items as appropriate and click the [OK] button.
   The setting items will be applied.

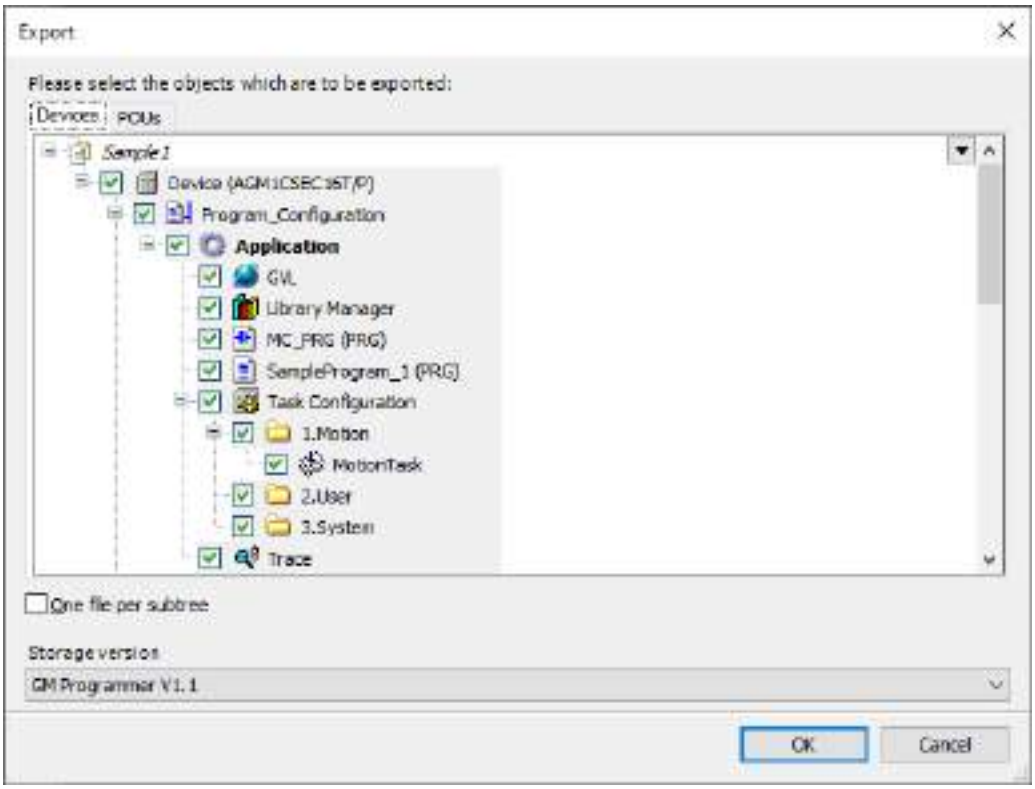
4.9 Exporting and Importing Objects

Objects in a project can be exported as XML files. The extension of exported files is ".export". Files that are being exported can be imported to GM Programmer.

4.9.1 Exporting Objects

Procedure

1. From the menu bar, select **Project**>**Export**.
   The "Export" dialog box will be displayed.

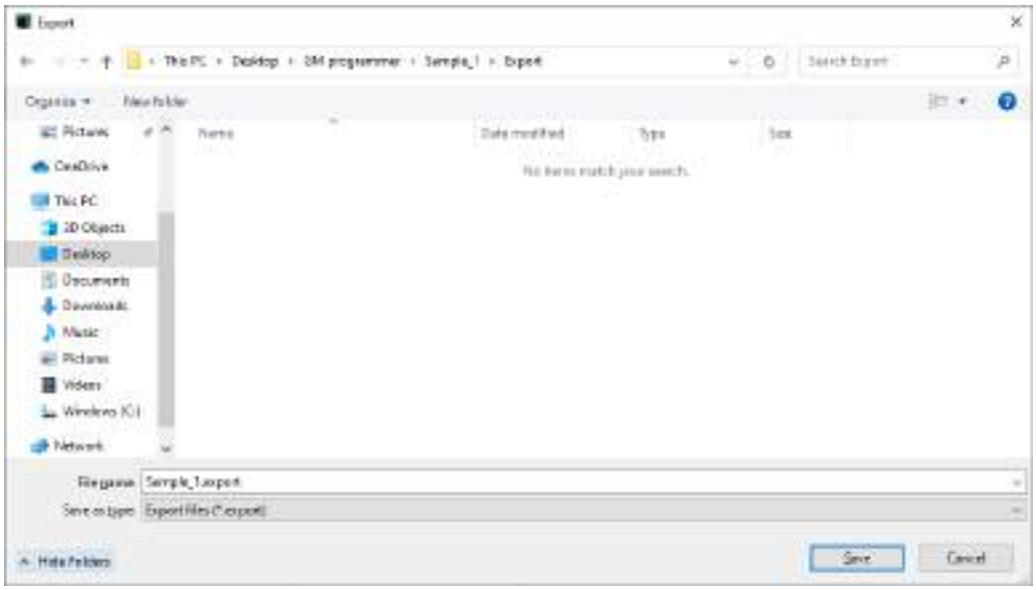


Info.

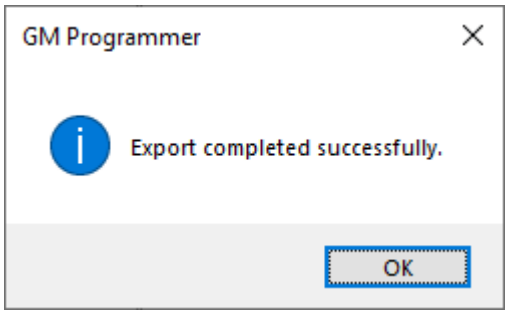
- To import an object into GM Programmer, select only one object below the [Application] object and export it.

2. Select objects to be exported.
   Normally, there is no need to make changes.

3. Click the [OK] button.
   The "Export" dialog box will be displayed.
   If necessary, change the file name and save destination.



4. Click the [Save] button.
Export will be executed.

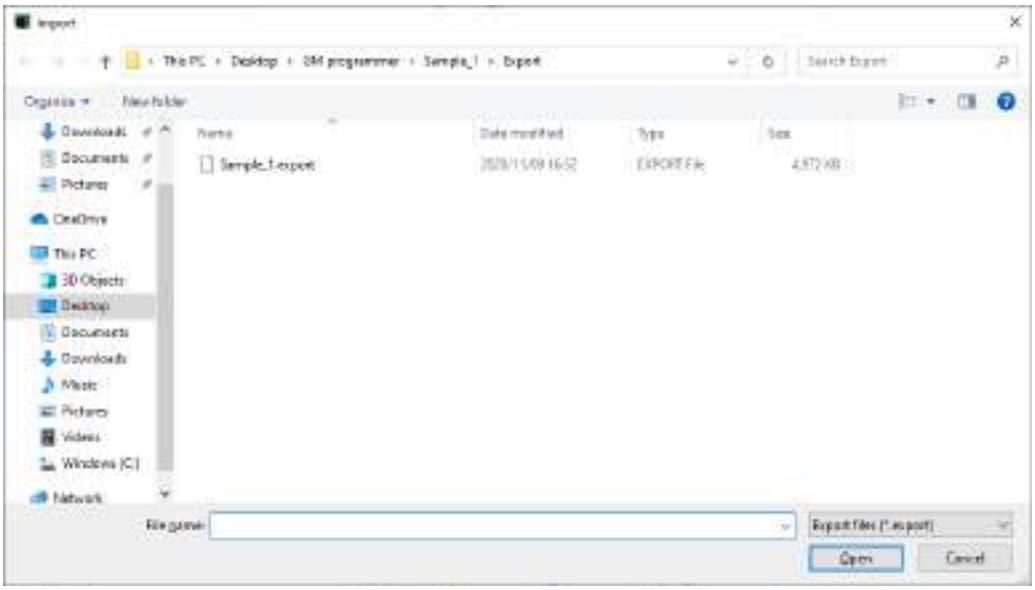


4.9.2 Importing Objects

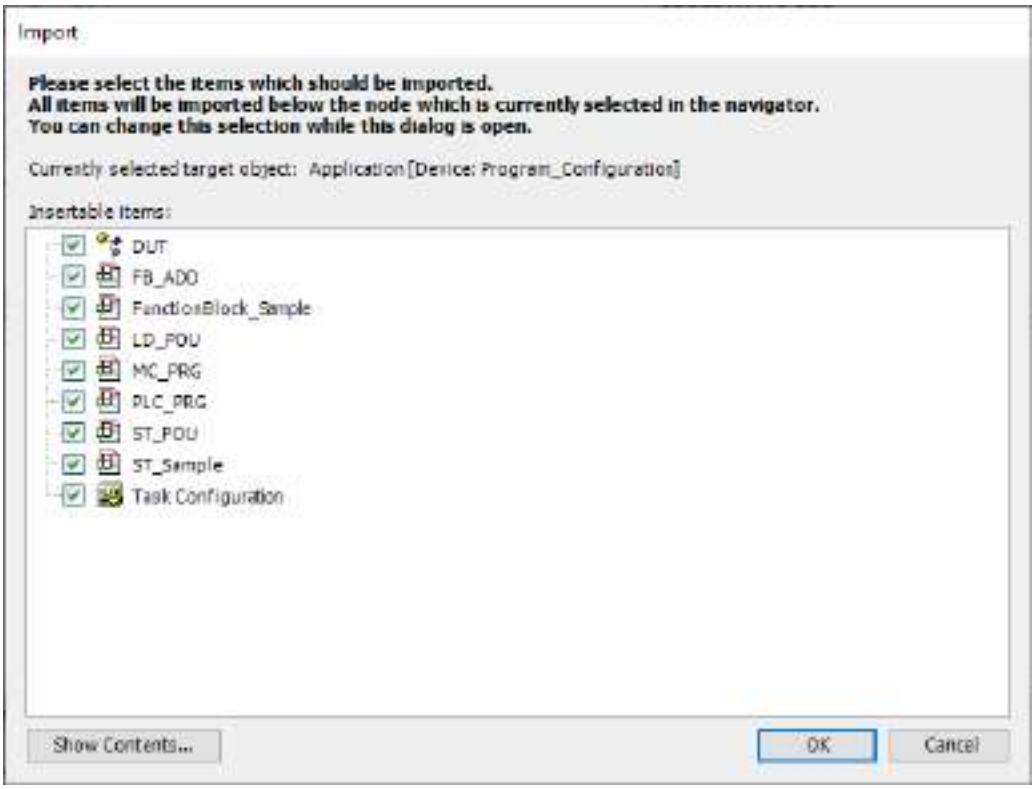
This section explains the procedure for importing objects exported to a project into GM Programmer.
For example, use the following procedure to import objects below the "Application" object.

Procedure

1. Select the [Application] object in the navigator pane and then select **Project>Import** from the menu bar.
The "Import" dialog box will be displayed.



2. Select a file with extension ".export" and click the [Open] button.
   The "Import" dialog box will be displayed.
   Objects that can be imported will be displayed in the "Insertable items" area.



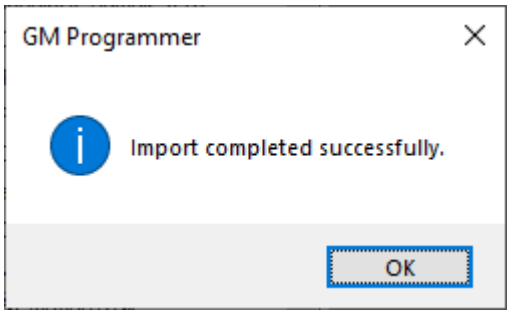
Info.

- To import an object into GM Programmer, select an export file where only one object below the [Application] object is selected.

3. Clear the check boxes of the objects that do not need to be imported and click the [OK] button.

Import will be executed.

The objects that have been imported will be displayed below the [Application] object in the navigator pane.

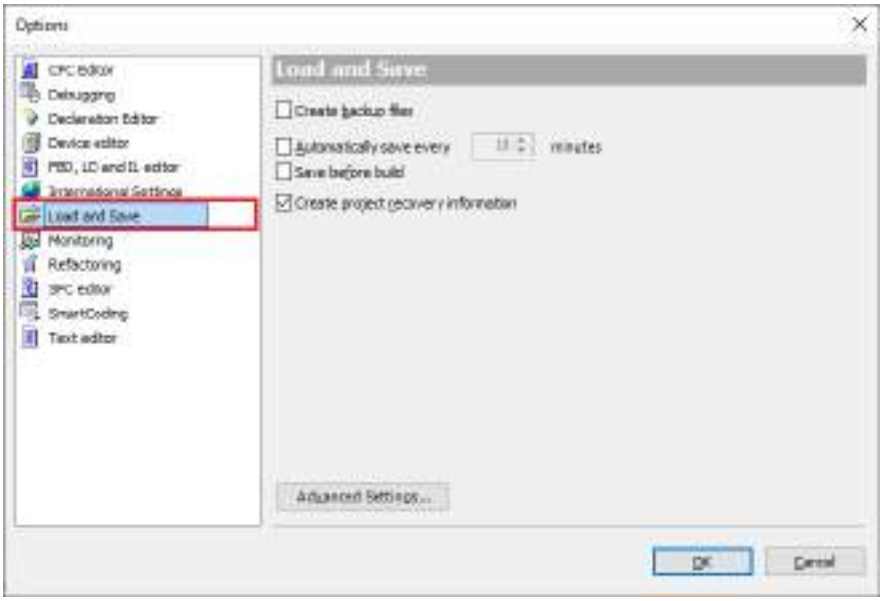


4.10 Creating a Backup when a Project Is Saved

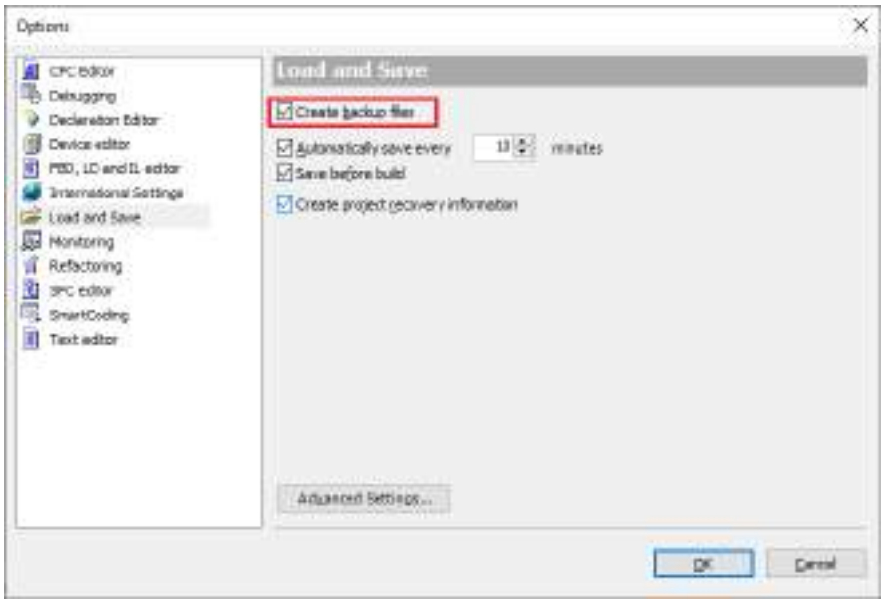
When saving a project, you can create a backup file of the project file to be updated. The extension of backup files is ".backup".

Procedure

1. From the menu bar, select **Tools**>**Options**.
   The "Options" dialog box will be displayed.
2. Select the "Load and Save" category.
   The "Load and Save" pane will be displayed.



3. Select the "Create backup files" check box and click the [OK] button.
   Then, whenever a project is saved, the project file to be updated will be automatically saved as a backup file (with extension ".backup").



<Restoring backup files>

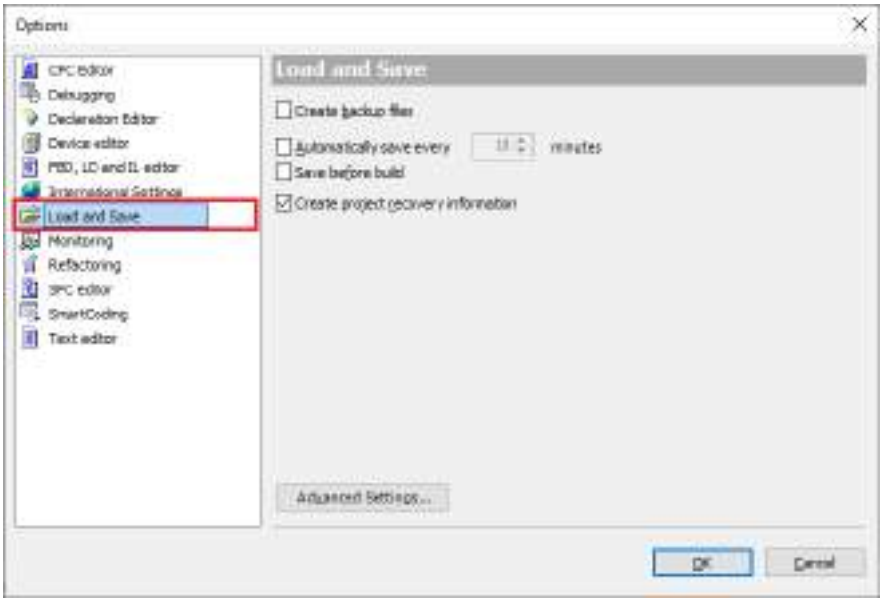
To restore a project file that has been backed up, manually change the extension of the file from ".backup" to ".project" and then open the project file in GM Programmer.

4.11 Automatically Saving Project Files

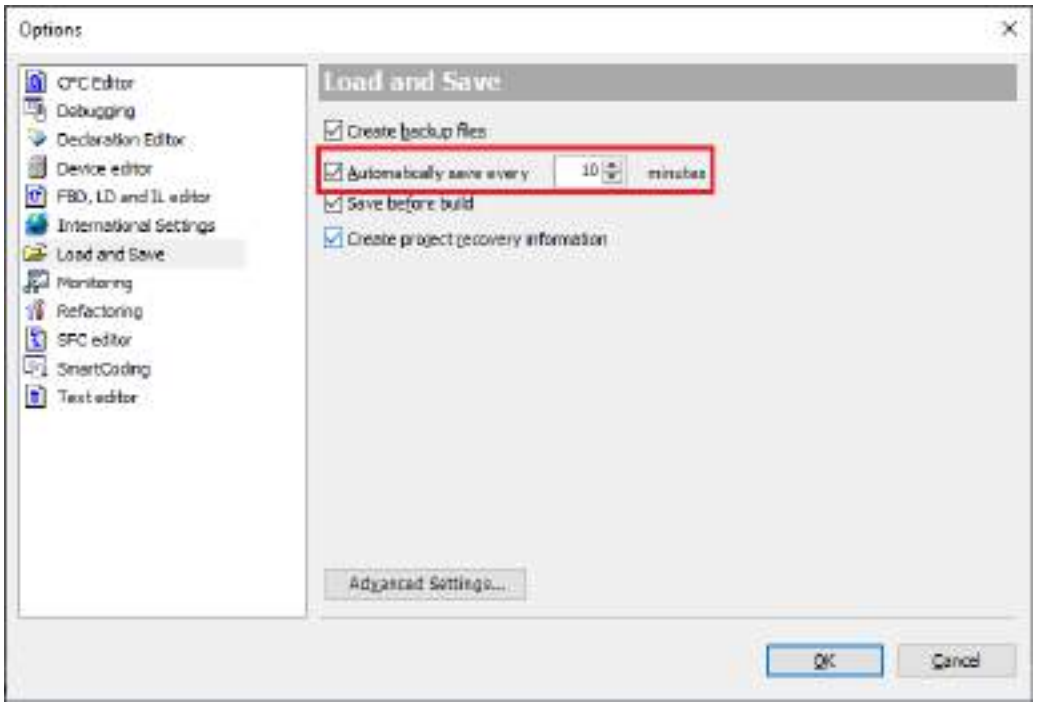
Project files can be saved automatically during editing. Even if data disappears when GM Programmer terminates abnormally, a file up to the point in time when it was saved automatically can be restored. The extension of backup files is ".autosave".

Procedure

1. From the menu bar, select **Tools**>**Options**.
   The "Options" dialog box will be displayed.
2. Select the "Load and Save" category.
   The "Load and Save" pane will be displayed.

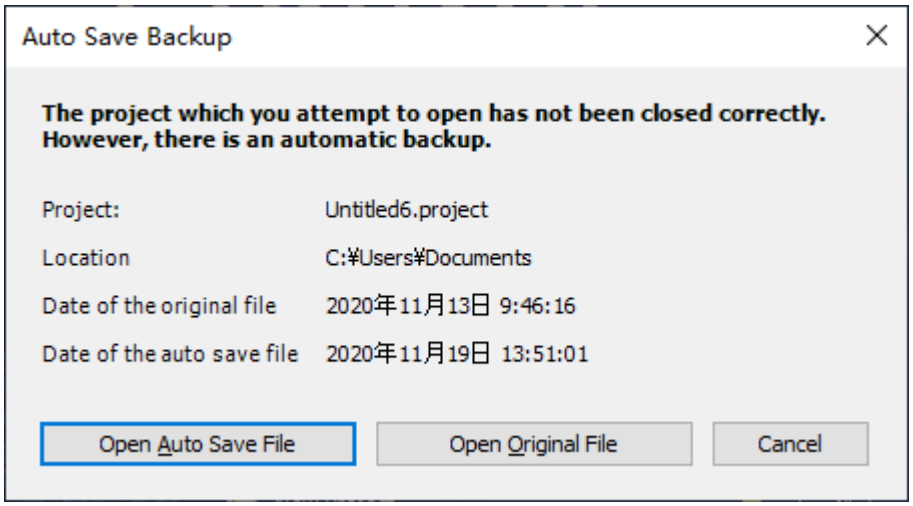


3. Select the "Automatically save evey" check box and click the [OK] button.
   The automatic save interval can be changed in minutes (default value: 10 minutes).



After auto save is set, project files will be automatically saved as files with extension ".autosave" at the specified interval during editing.

After a project file is closed due to abnormal termination of GM Programmer, when you open the project file again, you can select either the original project file with extension ".project" or the automatically saved project file with extension ".autosave". To open the automatically saved project file, click the [Open Auto Save File] button.

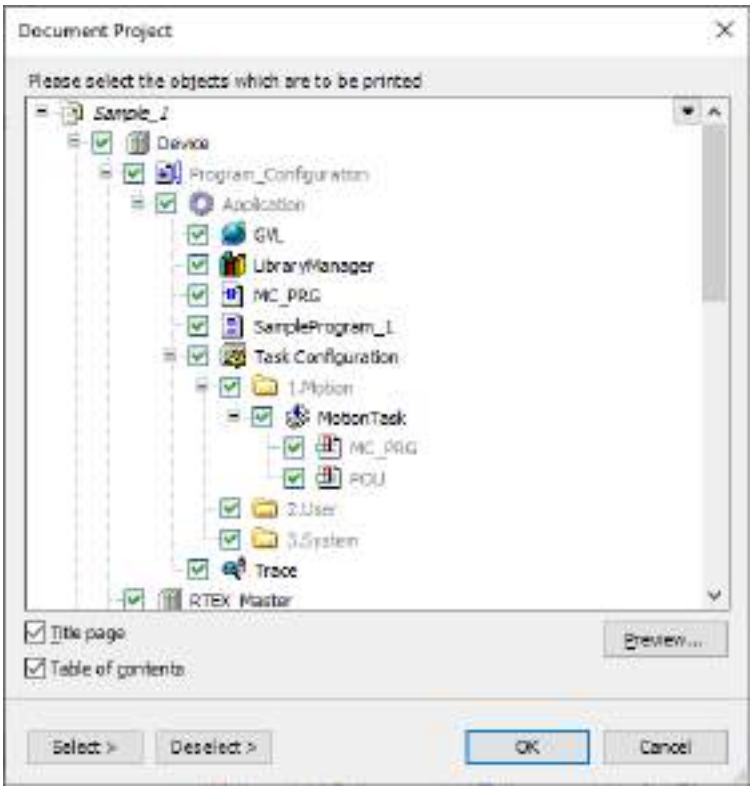


4.12 Printing a Project

You can print the entire project.

Procedure

1. From the menu bar, select **Project**>**Document**.
   The "Document Project" dialog box will be displayed.



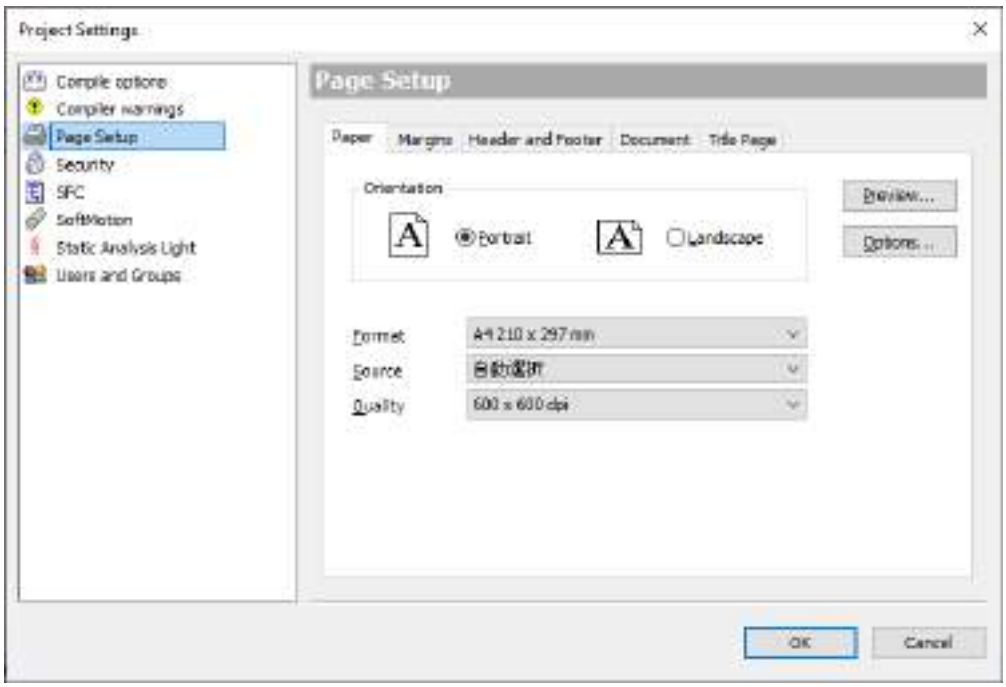
2. Select the check boxes of the devices and objects for which documents are to be printed, and click the [OK] button.
   The "Print" dialog box will be displayed. Select a printer to be used and click the [OK] button.
   Documents will be printed.

Info.

- By clicking the [Preview] button in the "Document Project" dialog box, a print preview window is displayed, so that you can check what a hard copy would look like when printed.
- You can edit page orientation, margin, header, footer, table of contents, and title page details.

  From the menu bar, select **File>Page Setup**. The "Project Settings" dialog box will appear with the Page Setup pane displayed.

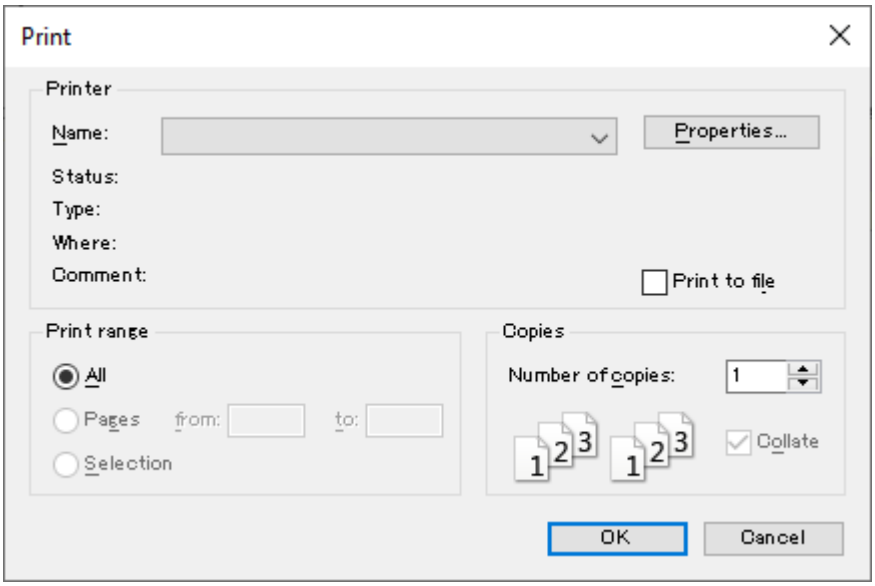
  Select the tab of the item to be changed and change the settings.



4.13 Printing an Object within a Project

Procedure

1. With the object editor opened, from the menu bar, select **File**>**Print**.
   The "Print" dialog box will be displayed. Execute printing.



4.14 Comparing Projects

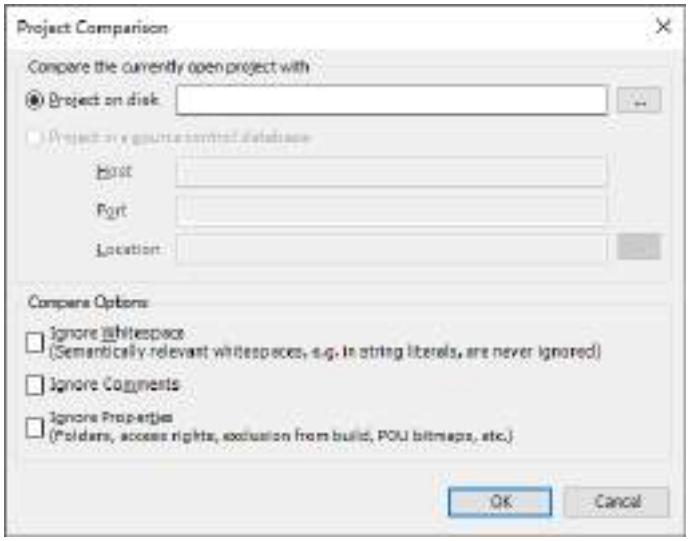
You can compare the opened project file with another project file to display and merge the differences between them.

4.14.1 Project Comparison Method

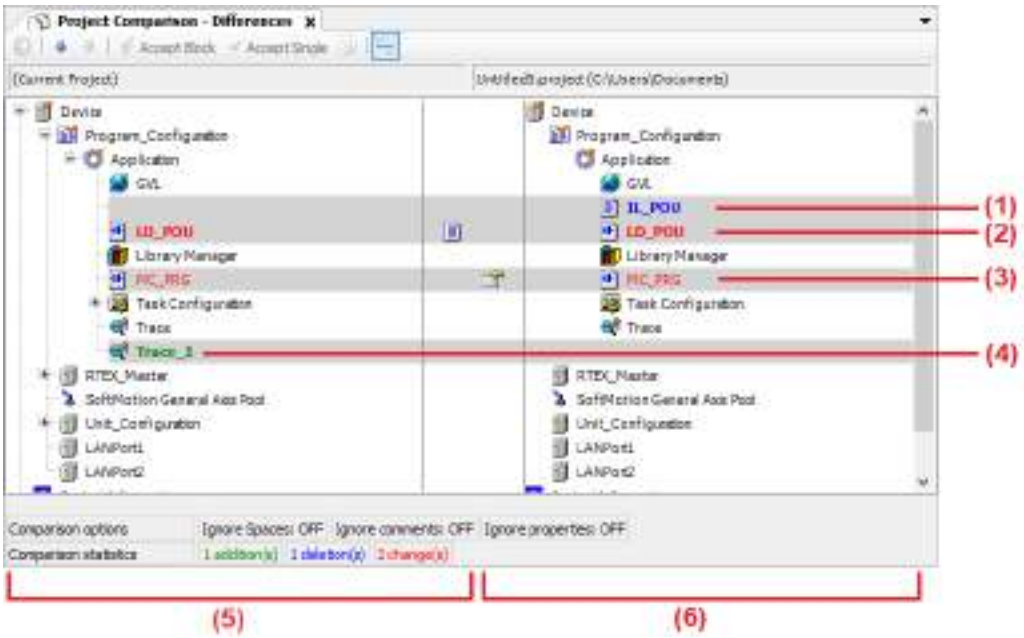
Compare the opened project file with another project file.

Procedure

1. From the menu bar, select **Project**>**Compare**.
   The "Project Comparison" dialog box will be displayed.

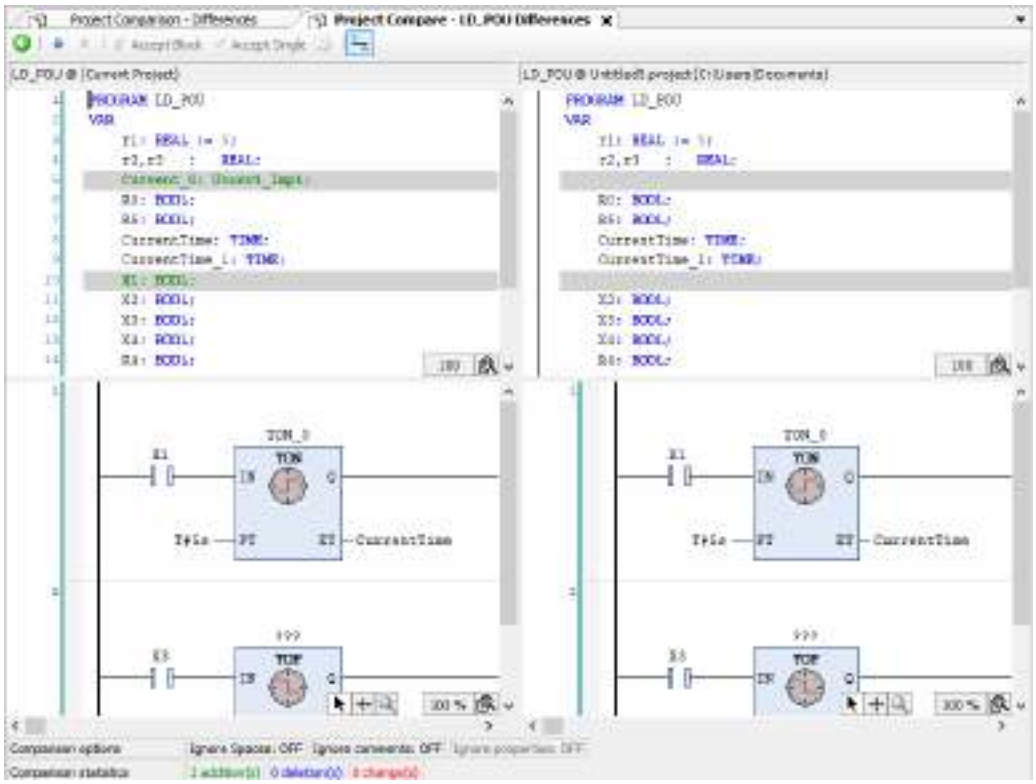


2. In the "Project on disk" field, specify a project file to be compared with and click the [OK] button.
   The comparison results will be displayed in the main pane.
   The backgrounds of object lines with differences are displayed in gray. The text color of each object indicates the type of difference.

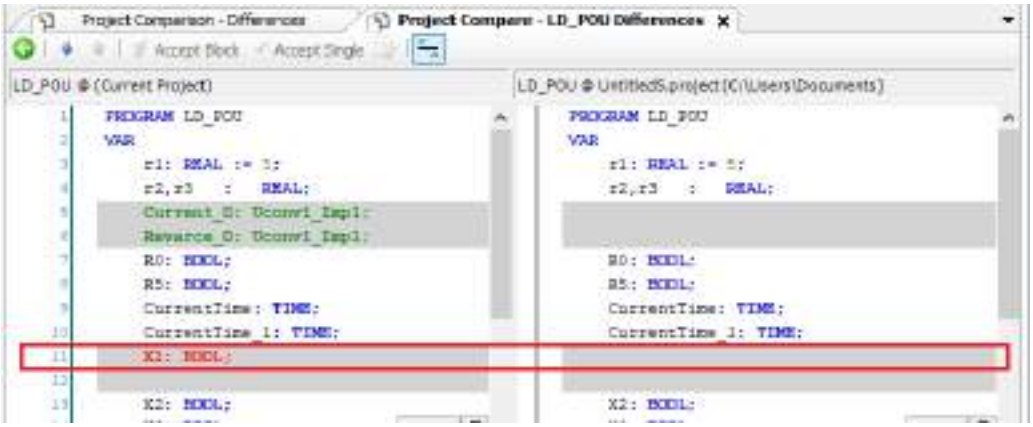


No.	Description
(1)	Removed object
(2)	Object with a difference in contents
(3)	Object with a difference in properties
(4)	Added object
(5)	Opened project
(6)	Specified project

Double-clicking an object line with a difference in contents displays the object comparison results.



Clicking the icon returns the display to the comparison view in the navigator pane.
Clicking the icon switches to the mode in which comparison results including removed object lines are displayed. The above window in which comparison results are displayed will be switched to the window shown below. If the differences displayed as variable bVar0 and empty lines that have been added differ from the specifications of variable bVar0, differences will be displayed as changes in variable bVar0.



4.14.2 Merging Differences

Differences displayed by comparing projects can be merged into the opened project.

When there are differences in the contents of objects, the differences can be merged by using the [Accept Block] button or [Accept Single] button.

For example, use the following procedure to merge the differences in the contents of programs.

Procedure

1. Perform project comparison.

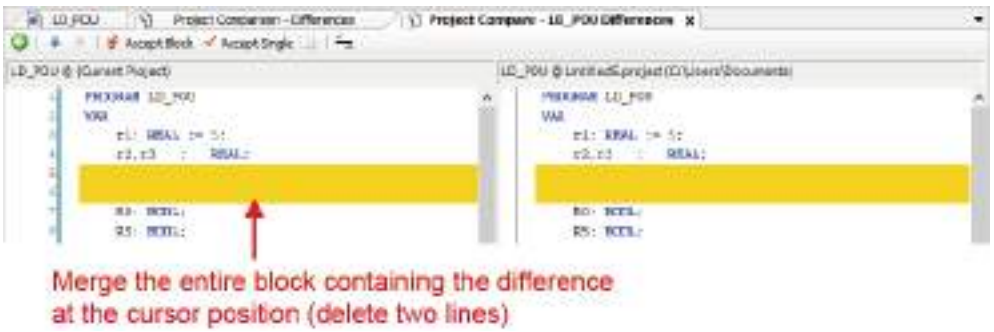
   Project comparison results will be displayed.

   For details on how to perform project comparison, refer to "4.14.1 Project Comparison Method".

2. Move the cursor to the difference location and click the [Accept Block] button or [Accept Single] button.

   The difference will be merged into the opened project. The background of the merged section will be displayed in yellow. At this stage, the merged content has not been reflected in the project yet. To reflect the merged content, approval operation is required as below.

   [Accept Block] button: Merges the entire block containing the difference at the cursor position

   [Accept Single] button: Merges the difference at the cursor position

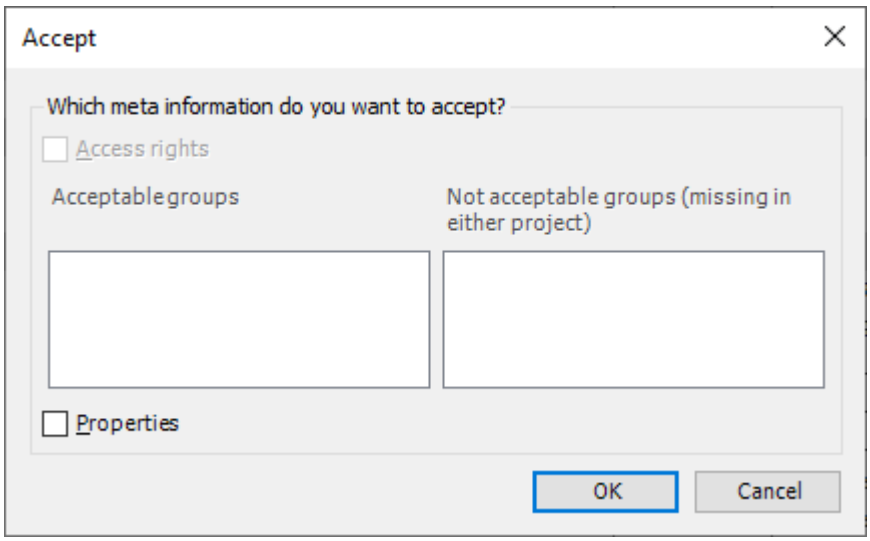
   

3. From the menu bar, select **Project**>**Commit accepted changes**.

   The merged content will be approved and reflected in the opened project. Because the reflected content has not been saved yet, save the project as necessary.

Info.

- To merge between objects whose properties are different, select an object with differences and click the icon. The "Accept" dialog box will be displayed. Select the "Properties" check box and click the [OK] button.



5 Project Creation

5.1 Project Creation Flow....5-2

5.2 Setting up the GM1 Controller....5-3

5.3 Setting up Motion Control....5-5
- 5.3.1 Basic Setting for GM1 Controller....5-5
- 5.3.2 Adding and Setting up Servo Amplifiers....5-6
- 5.3.3 Adding and Setting up Free Encoder and Virtual Drive....5-11

5.4 Setting up Unit Control....5-14
- 5.4.1 Setting up General-purpose I/O, PWM Output, and High-speed Counter for GM1 Controller....5-14
- 5.4.2 Adding Expansion Units....5-16

5.5 Setting up the Communication Function....5-20
- 5.5.1 Adding a Protocol to Be Used for the LAN Port....5-20
- 5.5.2 Adding a Protocol to Be Used for the COM Port....5-23

5.1 Project Creation Flow

To create a program, you must first create a project.

This section explains operations for projects, operations for adding objects to projects, and other related operations.

First, this section explains the flow of project creation.

1. Setting up the GM1 controller

Set up parameters for the GM1 controller.
Refer to "5.2 Setting up the GM1 Controller".

2. Setting up motion control

- Add device objects for servo amplifiers to a project and set them up.
- Add device objects for free encoders and virtual drives to a project and set them up.

Refer to "5.3 Setting up Motion Control".

3. Setting up unit control

- Set up general-purpose I/O, PWM output, and high-speed counter for the GM1 controller.
- Add device objects for expansion units to a project and set them up.

Refer to "5.4 Setting up Unit Control".

4. Setting up the communication function

- Add an object of the protocol to be used for the LAN port to a project and set it up.
- Add an object of the protocol to be used for the COM port to a project and set it up.

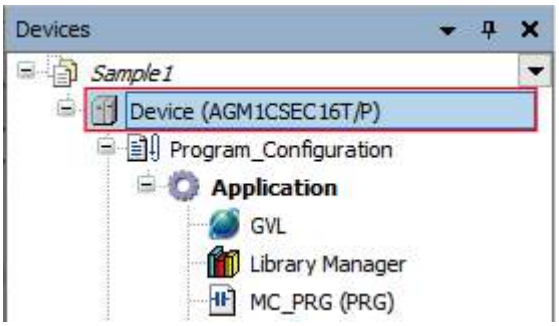
Refer to "5.5 Setting up the Communication Function".

5.2 Setting up the GM1 Controller

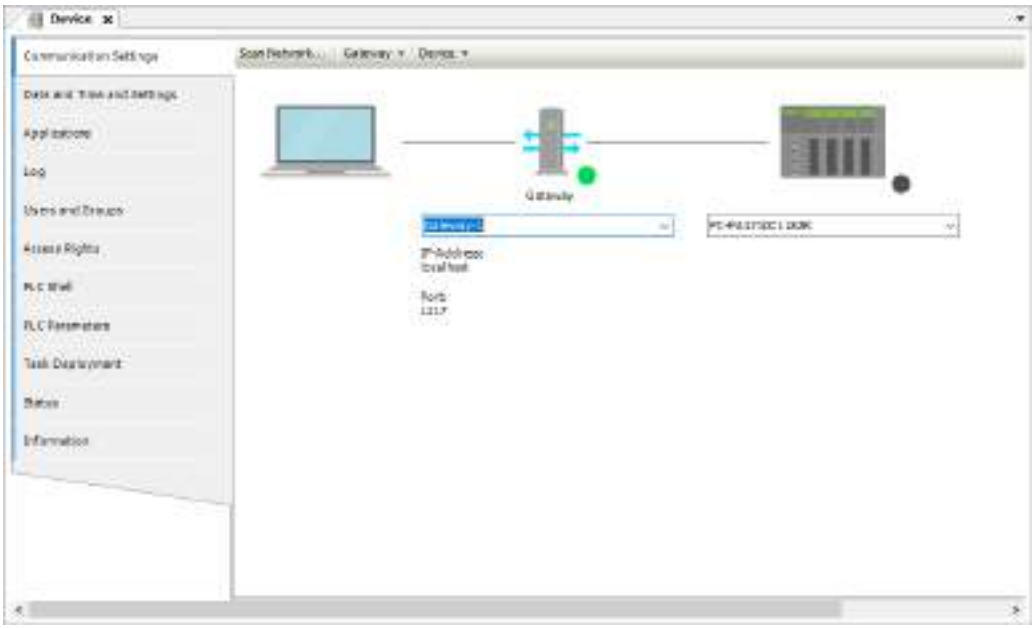
Set up parameters for the GM1 controller.

Procedure

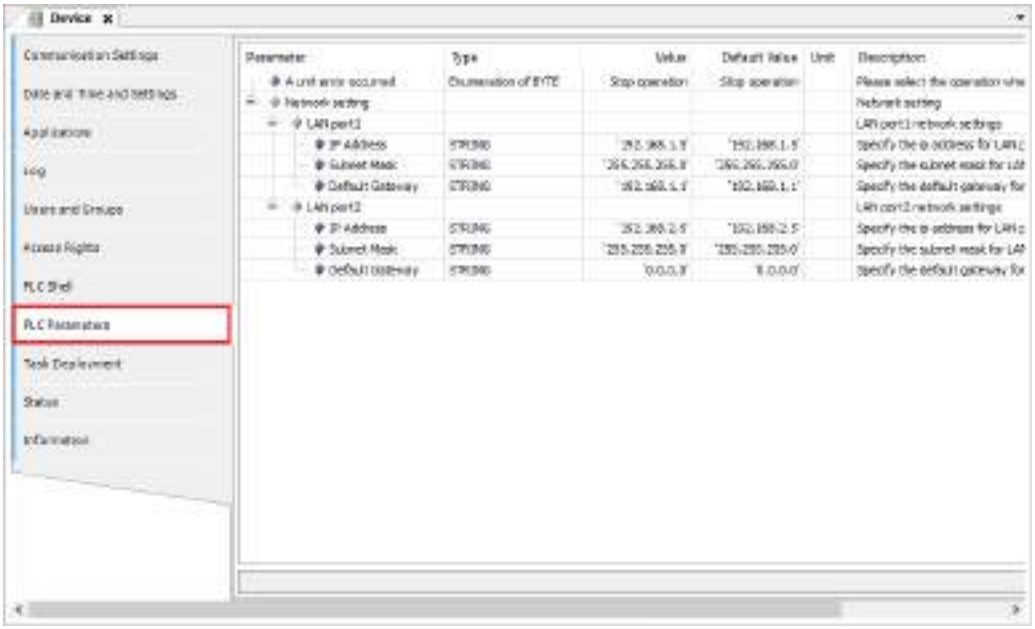
1. Double-click the [Device] object in the navigator pane.



The Communication Settings window for the device will be displayed.



2. Click the "PLC Parameters" tab.
   The parameter settings sub-pane will be displayed.



3. Change the values of the parameters that you want to update.
"A unit error occurred": Select whether to stop or continue operation when an error occurs.
"LAN port1": Change the IP address of LAN port1.
"LAN port2": Change the IP address of LAN port2.

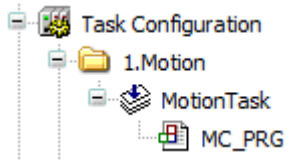
5.3 Setting up Motion Control

5.3.1 Basic Setting for GM1 Controller

Set a control cycle for the GM1 Controller.

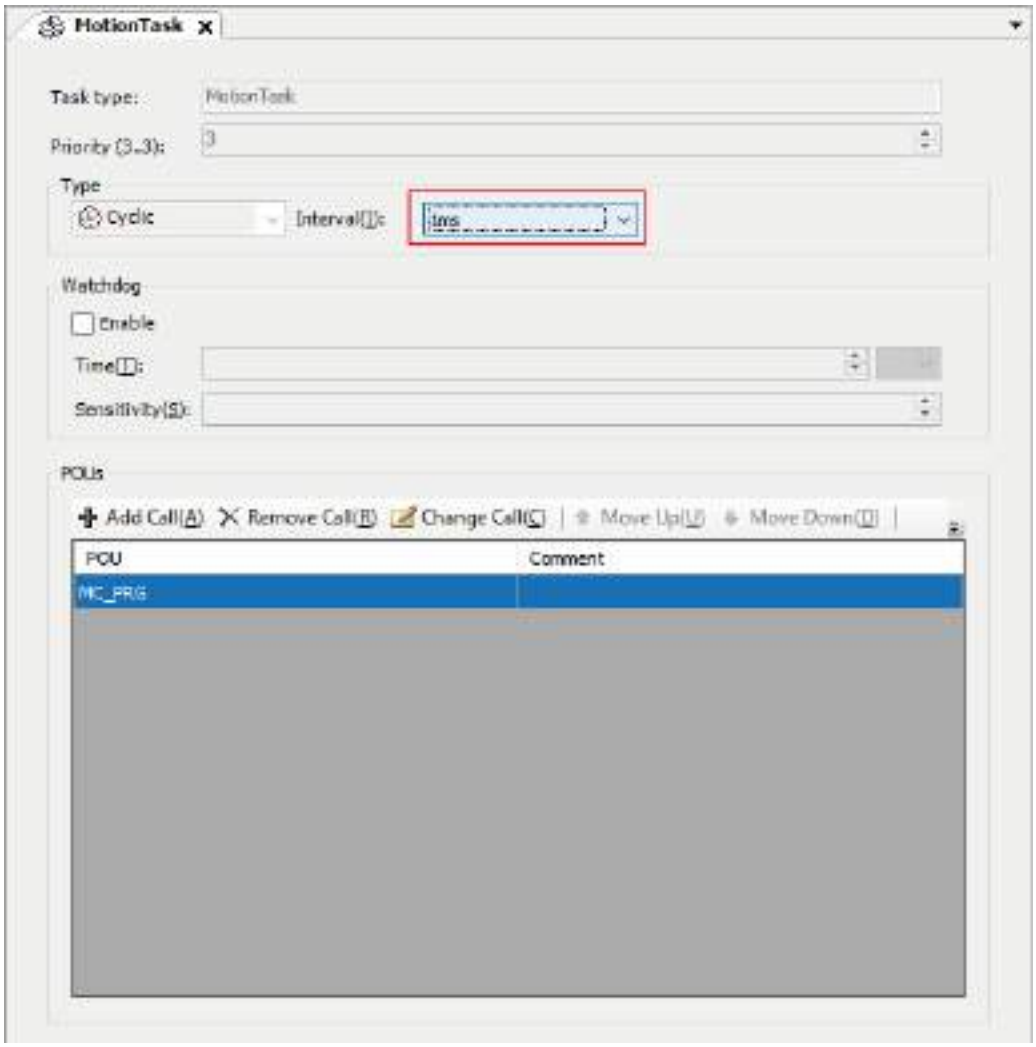
Procedure

1. Double-click the **MotionTask** object in the navigator pane.



The "MotionTask" editor will open in the main pane.

2. Set a control cycle.



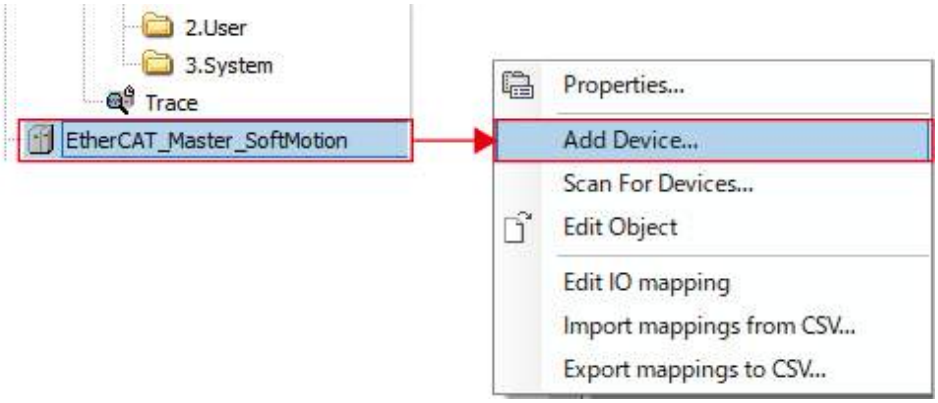
5.3.2 Adding and Setting up Servo Amplifiers

This section explains how to add device objects for servo amplifiers to a project and set them up.

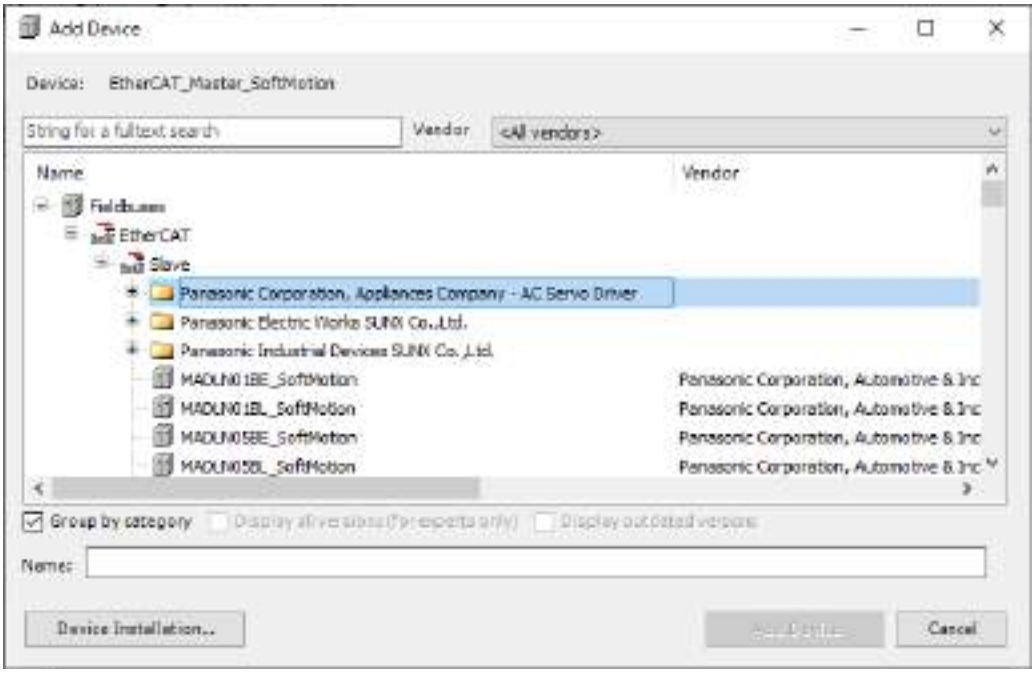
The description below explains how to add device objects for A5B servo amplifiers to a project and how to set them up.

Procedure

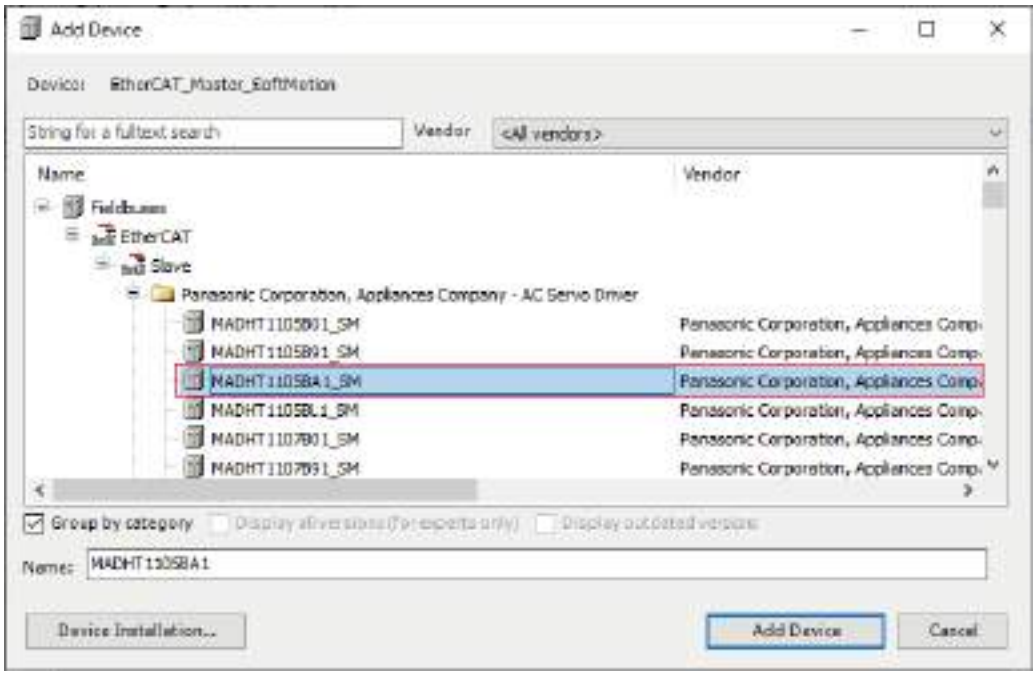
1. Right-click the **EtherCAT_Master_SoftMotion** object in the navigator pane and then select **Device addition** from the context-sensitive menu that is displayed.



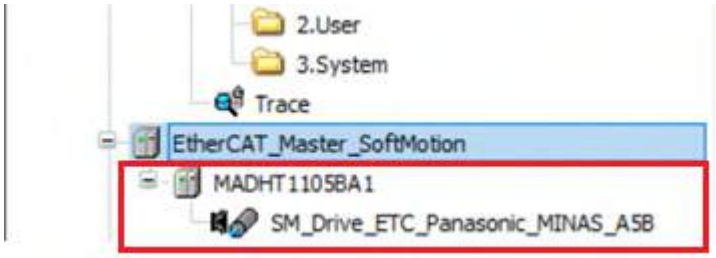
The "Add Device" dialog box will be displayed.



2. Select a device object for the servo amplifier.
The selected device object of the servo amplifier will be added.

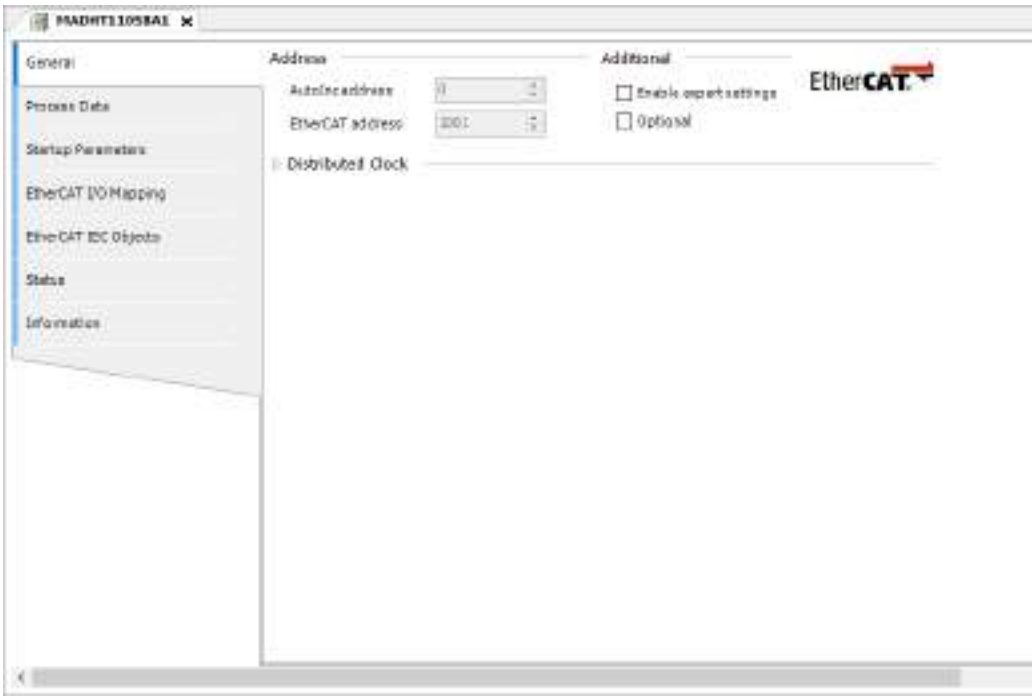


3. Click the [Add Device] button.
The selected device object of the servo amplifier will be added to the navigator pane.



Click the [Cancel] button to close the “Select Device” dialog box.

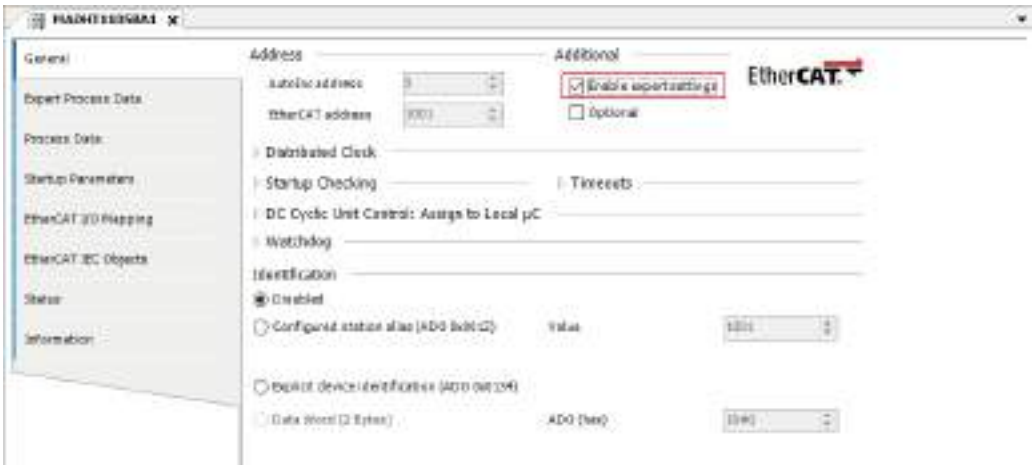
4. Double-click the added object.
The setting pane will be displayed in the main pane.



Info.

- To remove a device object that has been added, select the device object in the navigation pane and press the <Delete> key.

5. Enable the “Enable expert setting” item.



6. Set “Station alias”.

Depending on the setting of Pr7.41 as the initial setup for the servo amplifier, the method for setting “Station alias” is different.

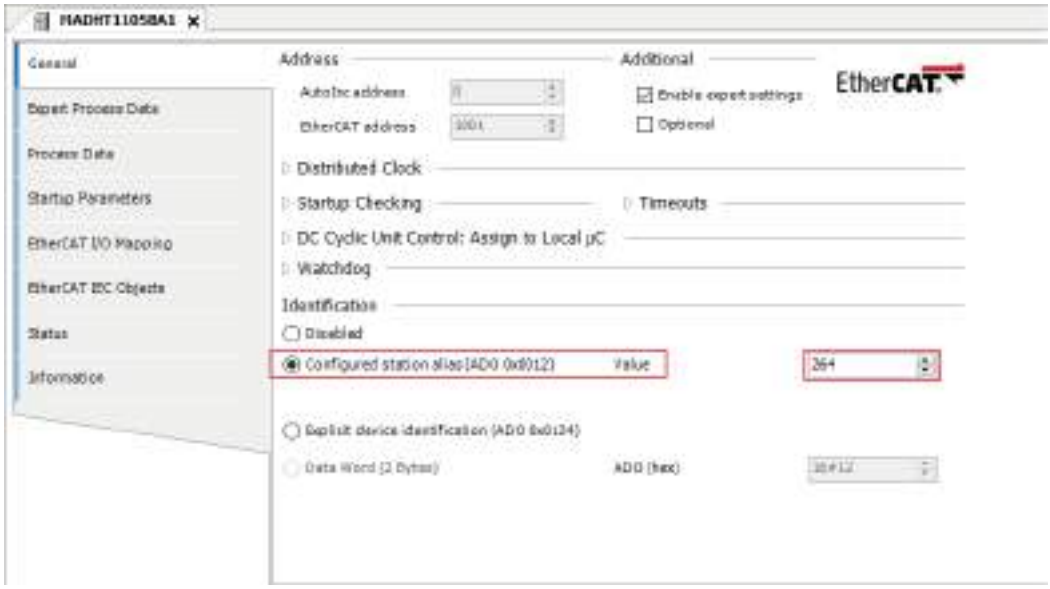
- When Pr.7.41 is set to 0

Select the configured station alias and input in the input field the set value of the rotary switch on the front panel of the servo amplifier and the set value of Pr7.40.

Example) Set value of the rotary switch on the front panel of the servo amplifier:8, set value of Pr7.40: 1

The higher 8 bits are 1 and lower 8 bits are 8. Therefore, input 264 in the input field.

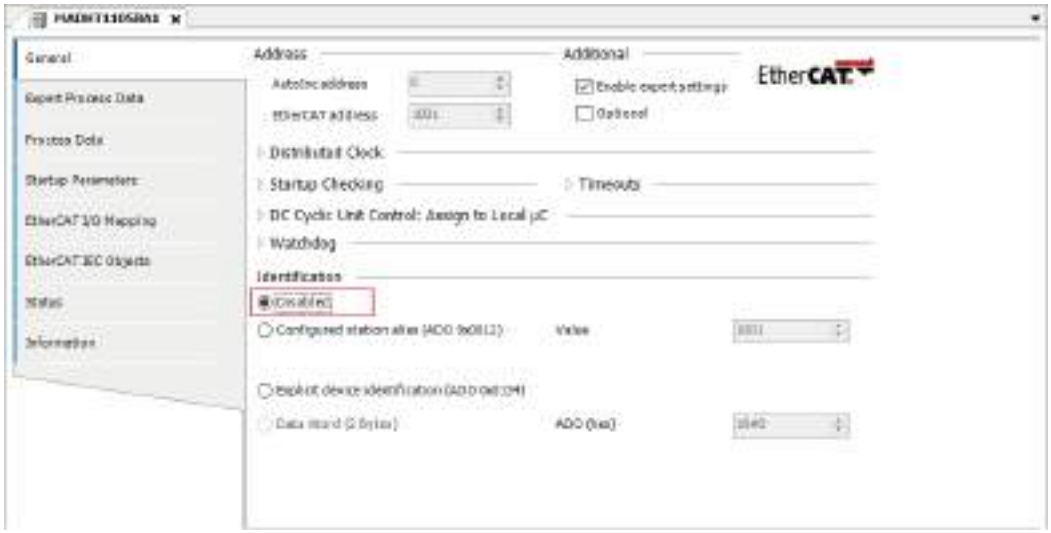
Station alias	
Higher 8 bits	**Lower 8 bits**
Set value of 3740h	Set value of the rotary switch



- When Pr.7.41 is set to 1

  Select “Disable”.

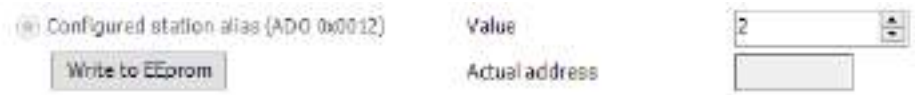
  The value displayed on the EtherCAT address is set as the station alias value.



Info.

- Select the configured station alias value and log in, and then you can also set a desired value.

  Input a desired value in the input field and click the [Write to EEPROM] button.



  Restart the power supply to the servo amplifier and then log in again, and the desired value input in the input field will be set as the station alias value.

  Select the configured station alias value and log in for the first time, and the device name of the servo amplifier in the navigation pane will be grayed out due to a mismatch of the station alias value .

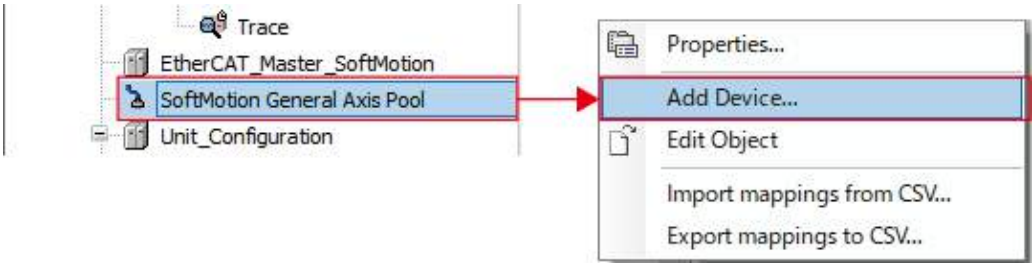
5.3.3 Adding and Setting up Free Encoder and Virtual Drive

This section explains how to add device objects for free encoders and virtual drives to a project and set them up.

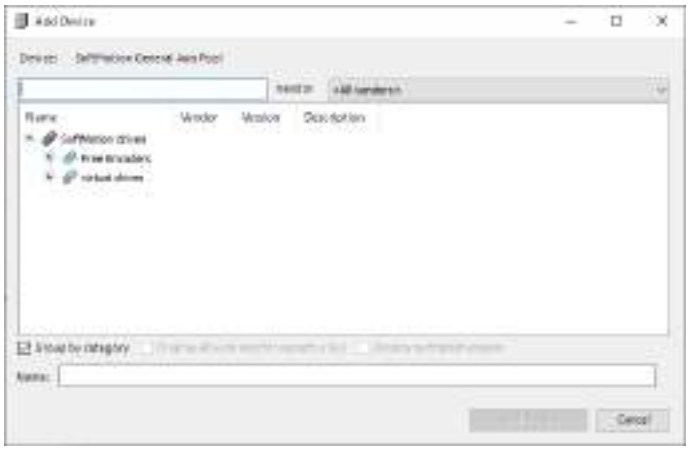
For example, use the following procedure to add a device object for a virtual drive to a project and set it up.

Procedure

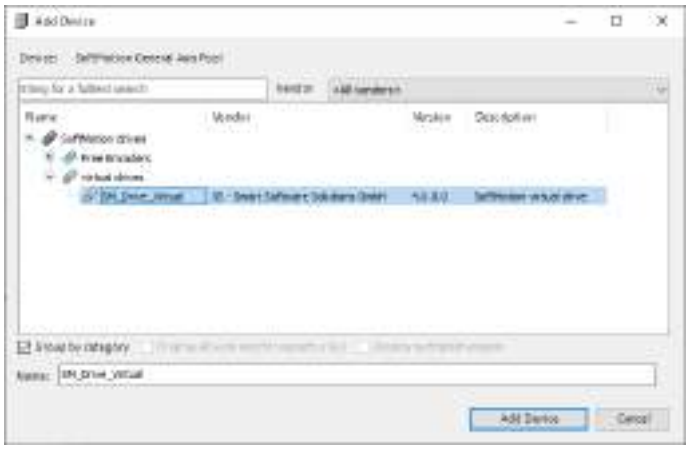
1. Right-click the [SoftMotion General Axis Pool] object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.



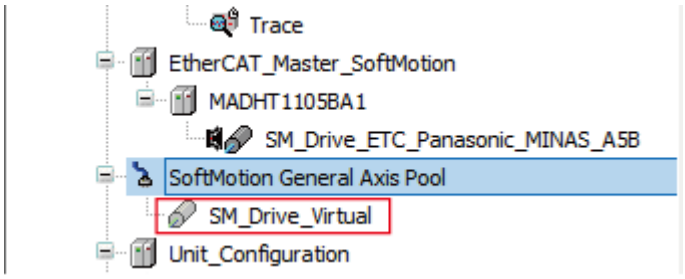
   The "Device Management" dialog box will be displayed.



2. Select a device object for the virtual drive.



3. Click the [Add Device] button.
   The selected device object of the virtual drive will be added to the navigator pane.



4. Double-click the added object.
   The setting pane will be displayed in the main pane. Specify settings related to the virtual drive.

Info.

- To remove a device object that has been added, select the device object in the navigation pane and press the "Delete" key.

5.4 Setting up Unit Control

5.4.1 Setting up General-purpose I/O, PWM Output, and High-speed Counter for GM1 Controller

This section explains how to set up general-purpose I/O, PWM output, and high-speed counter for the GM1 Controller.

The object of each unit is shown below.

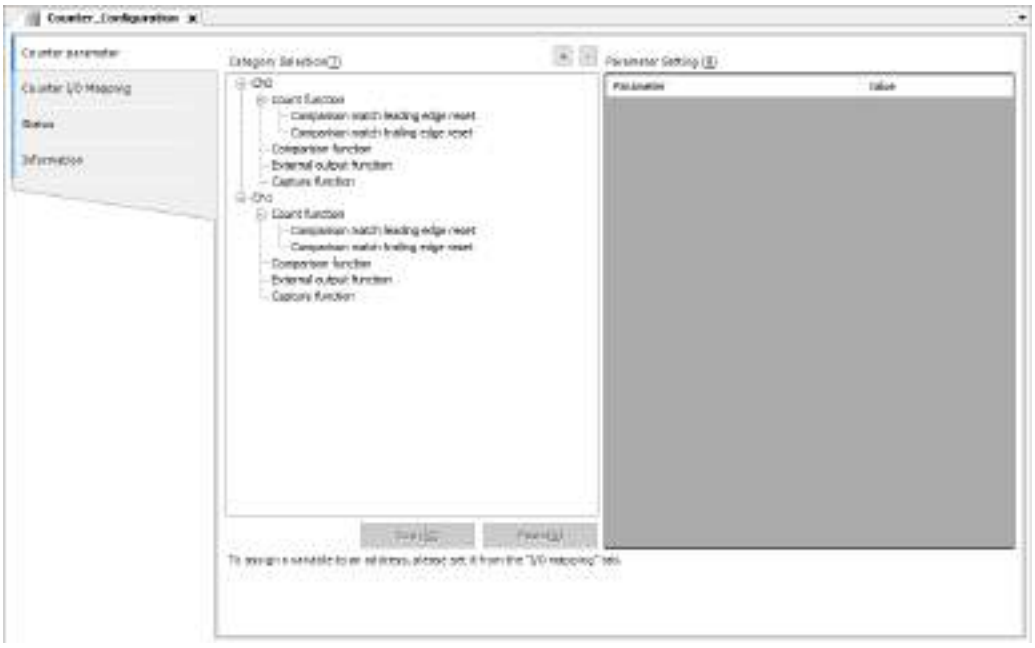
Info.

- For details on general-purpose I/O, refer to "12.4 General-purpose I/O".
- For details on PMW output, refer to "12.5 PWM Output".
- For details on the high-speed counter function, refer to "12.6 High-speed Counter Function".

For example, use the following procedure to set up a high-speed counter.

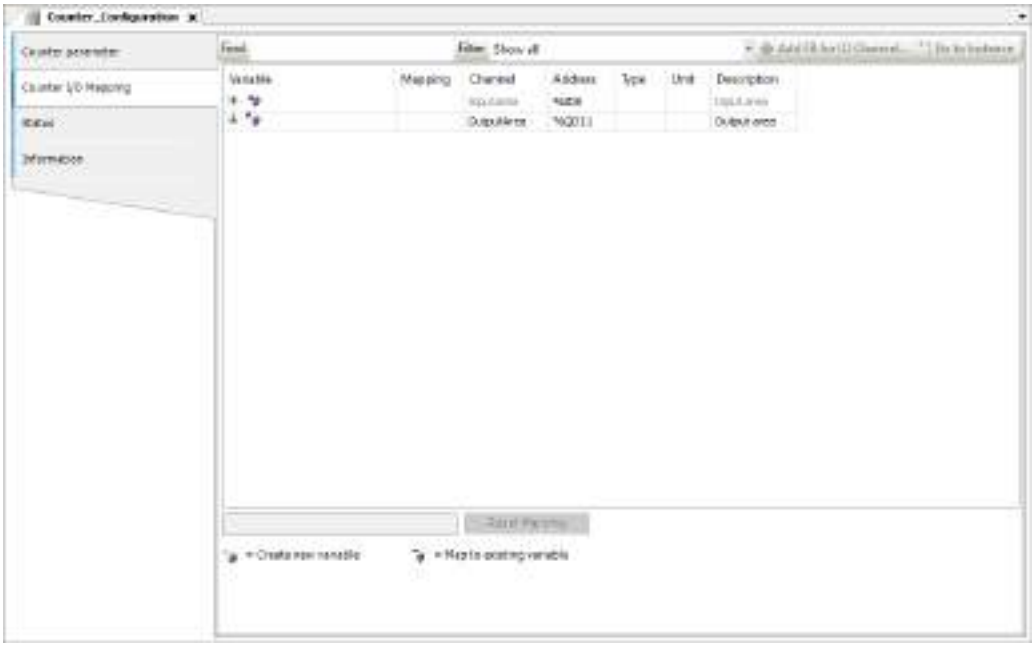
Procedure

1. In the navigator pane, double-click the [Counter_Configuration] object.
   The high-speed counter setting window will be displayed.



2. Set up parameters for the high-speed counter.

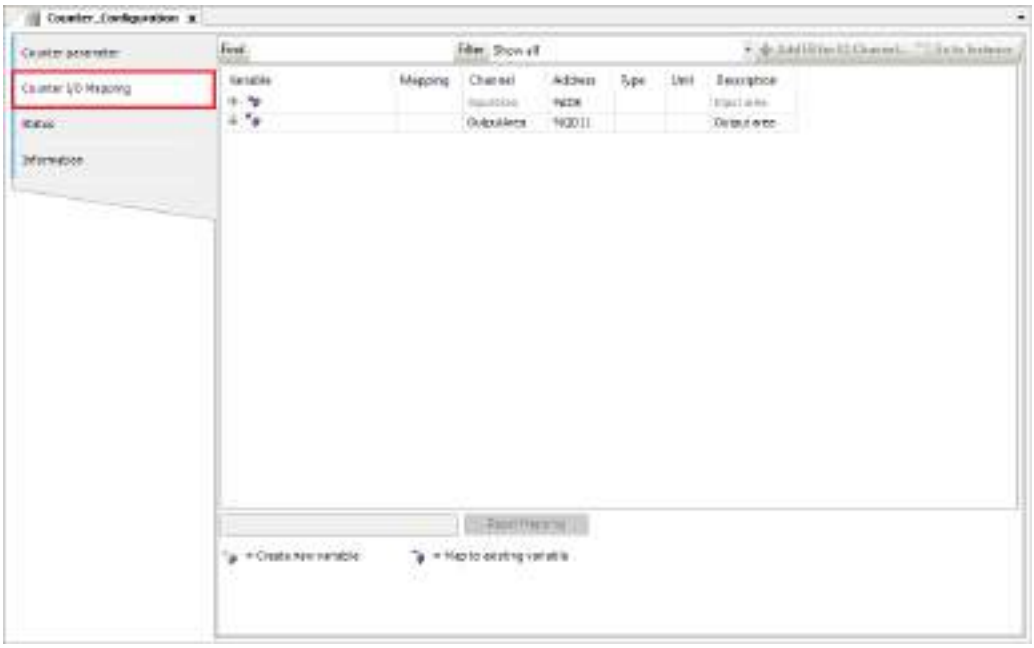
Select the category of a channel to be set up in the "Category Selection" section and enter values in the "Parameter Settings" section.



3. Select the "Counter I/O Mapping" tab and set the correspondence (mapping) between the channel and variable in the mapping setting pane.

Click the "Variable" column corresponding to the channel to be used by the program and enter a variable name.

Clicking the mark in the "Mapping" column allows you to change the type of mapping.



Info.

- You can copy the parameter set in a channel. To do so, select a channel (CH0 or CH1) in the "Category Selection" column and click the [Copy] button. Next, select another channel and click the [Paste] button.

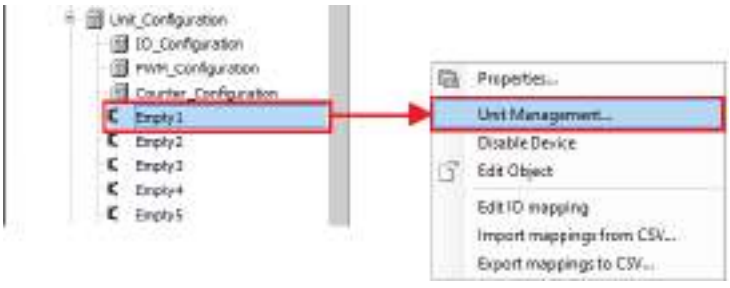
5.4.2 Adding Expansion Units

This section explains how to add device objects for expansion units to a project and set them up.

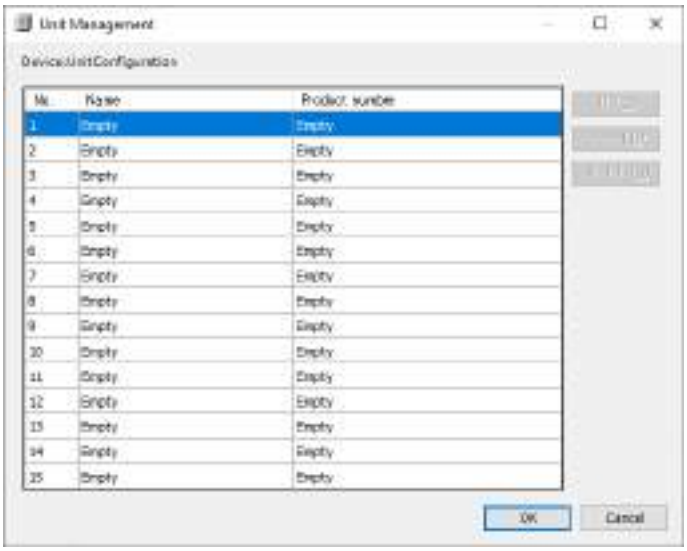
For example, use the following procedure to add a digital input unit (product number: AGM1X64D2) to Empty1.

Procedure

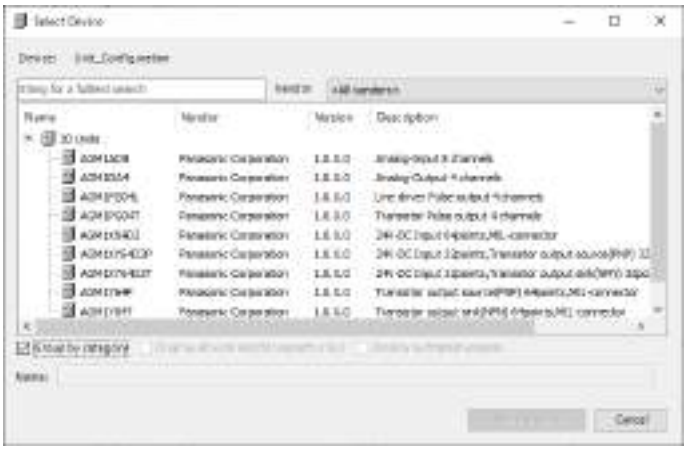
1. Right-click the [Empty1] object in the navigator pane and then select "Unit Management" from the context-sensitive menu that is displayed.



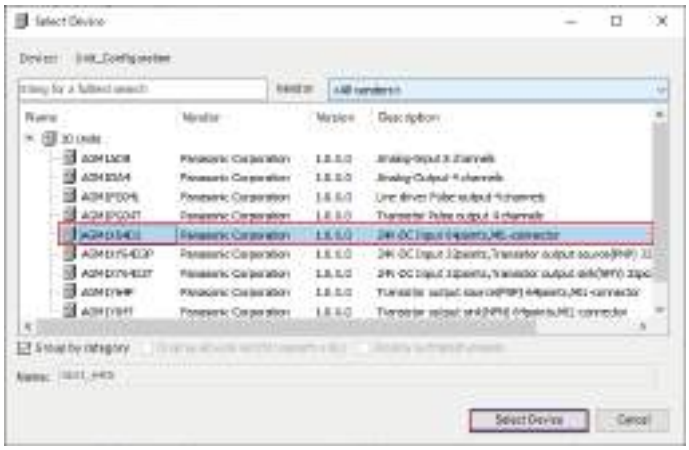
The "Unit Management" dialog box will be displayed.



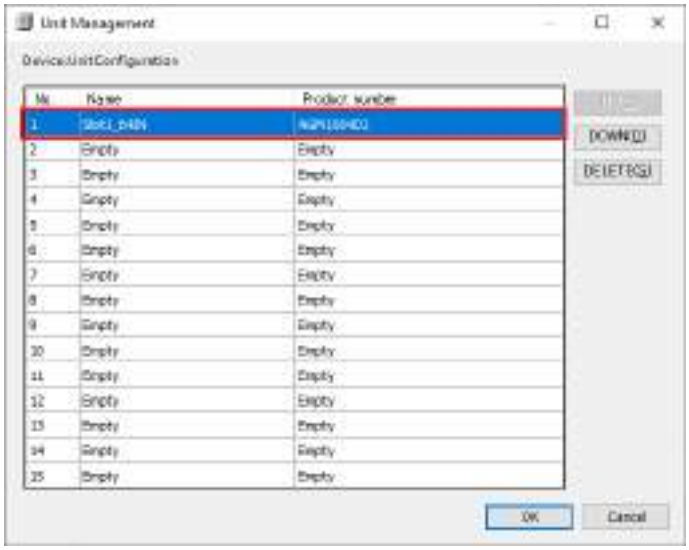
2. Double-click the first row in the "Device: Unit_Configuration" table.
The "Select Device" dialog box will be displayed.



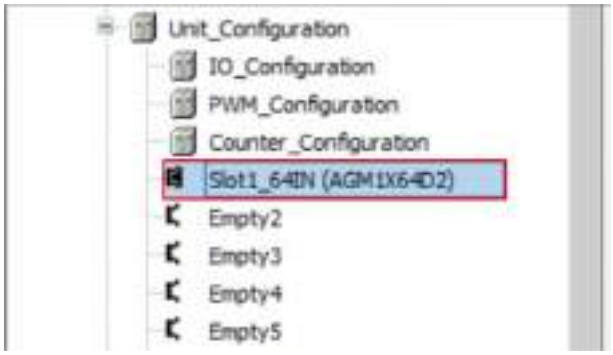
3. Select a device object for the expansion unit to be added.



4. Click the [Select Device] button.
   The selected device object of the expansion unit will be added.



5. Click the [OK] button.
   The selected device object of the expansion unit will be added to the navigator pane.



6. Double-click the added object.

The setting pane will be displayed in the main pane. Specify settings related to the expansion unit.

Info.

- To remove the device object of an expansion unit that has been added, select the expansion unit to be removed in the "Unit Management" dialog box and press the "Delete" key or click the [Delete] button.

5.5 Setting up the Communication Function

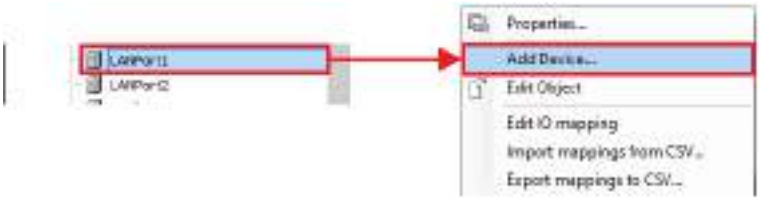
5.5.1 Adding a Protocol to Be Used for the LAN Port

This section explains how to add an object of the protocol to be used for the LAN port to a project and set it up.

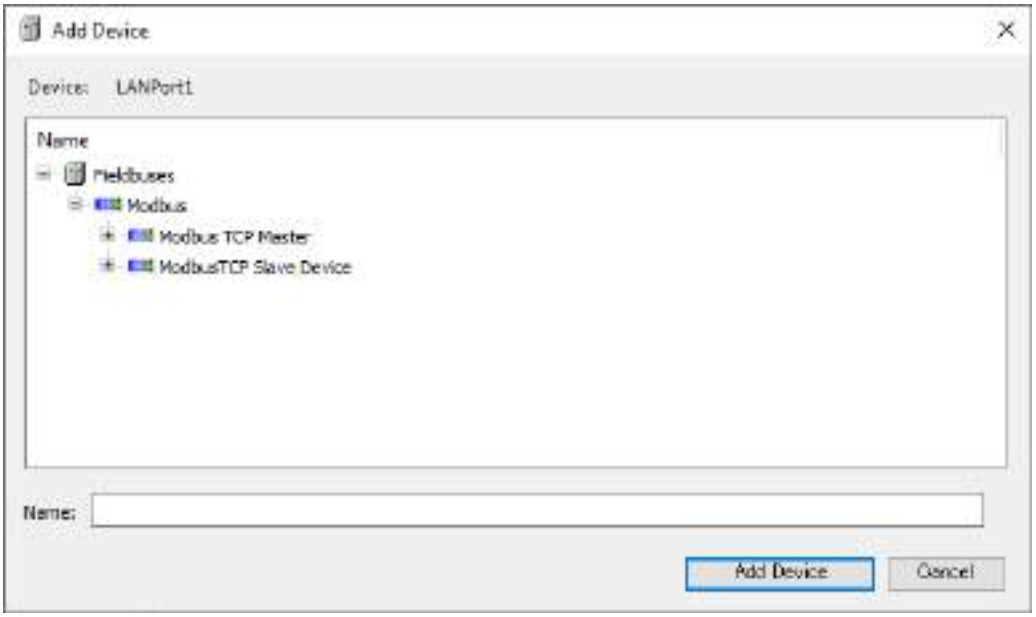
For example, use the following procedure to add an object of Modbus TCP to LANPort1 and set it up.

Procedure

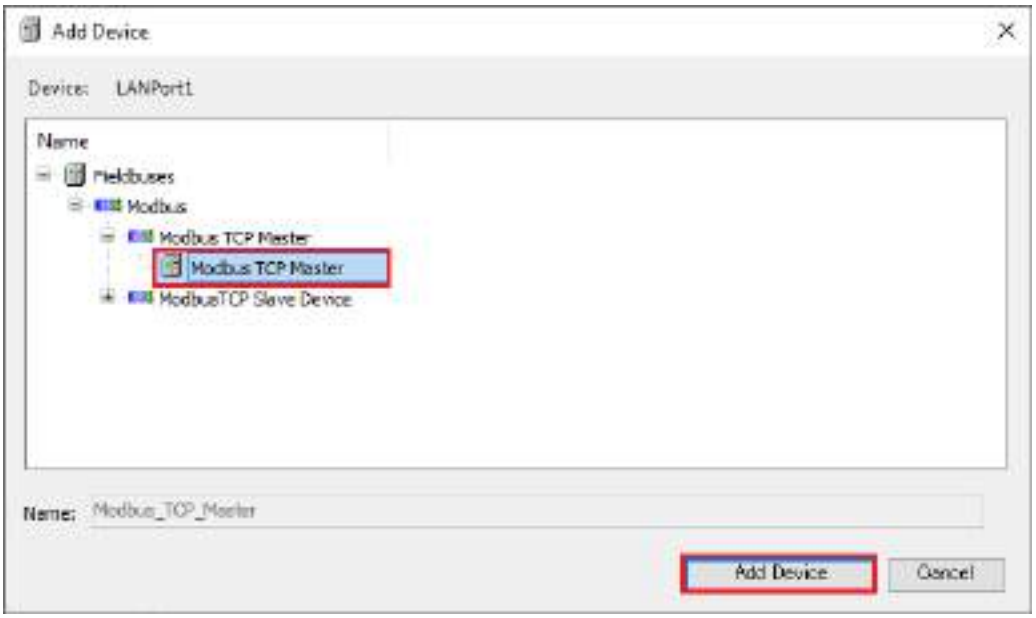
1. Right-click the [LANPort1] object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.



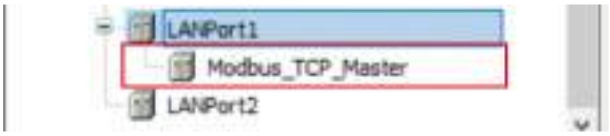
The "Add Device" dialog box will be displayed.



2. Select device "Modbus TCP Master".

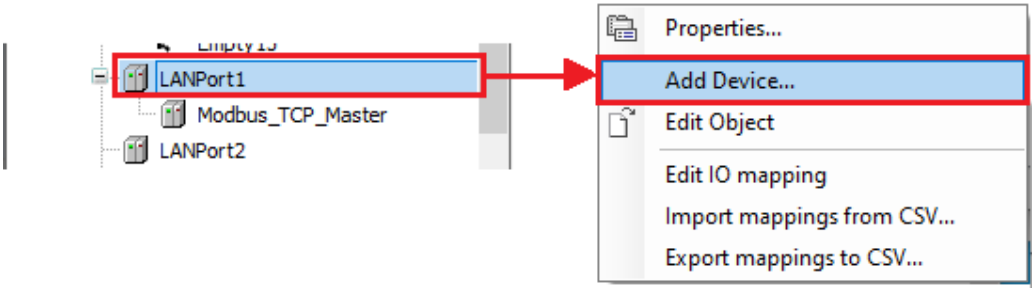


3. Click the [Add Device] button.
Object [Modbus_TCP_Master] will be added to the navigator pane.

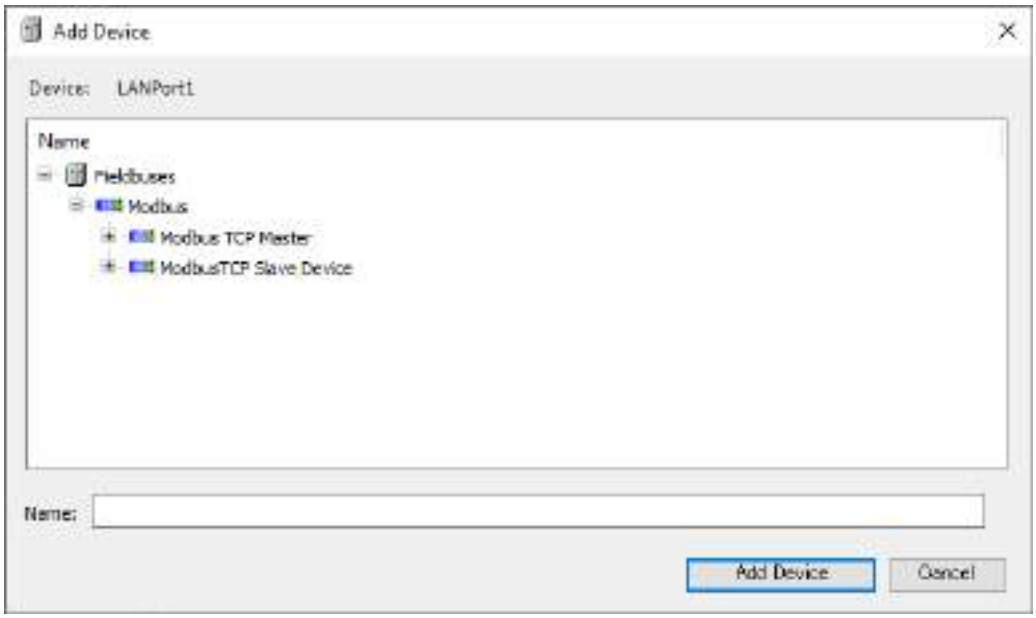


Then, add object [ModbusTCP_Slave_Device] below object [Modbus_TCP_Master].

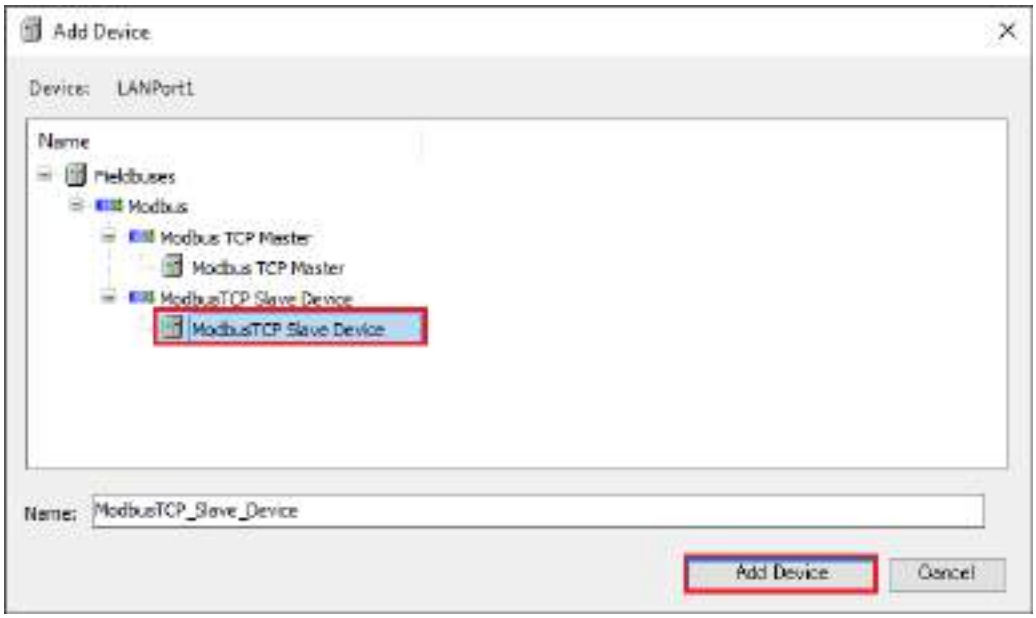
4. Right-click the [LANPort1] object and then select "Add Device" from the context-sensitive menu that is displayed.



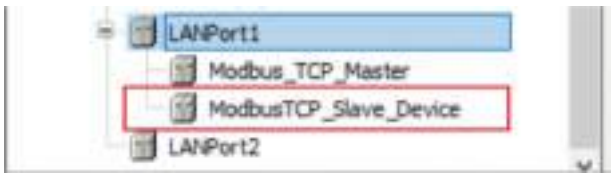
The "Add Device" dialog box will be displayed.



5. Select device "Modbus TCP Slave Device".



6. Click the [Add Device] button.
The [ModbusTCP_Slave_Device] object will be added.



7. Double-click the added object.

The setting pane will be displayed in the main pane. Specify settings related to Modbus TCP.

Info.

- To remove a device that has been added, select the device in the navigator pane and press the "Delete" key.

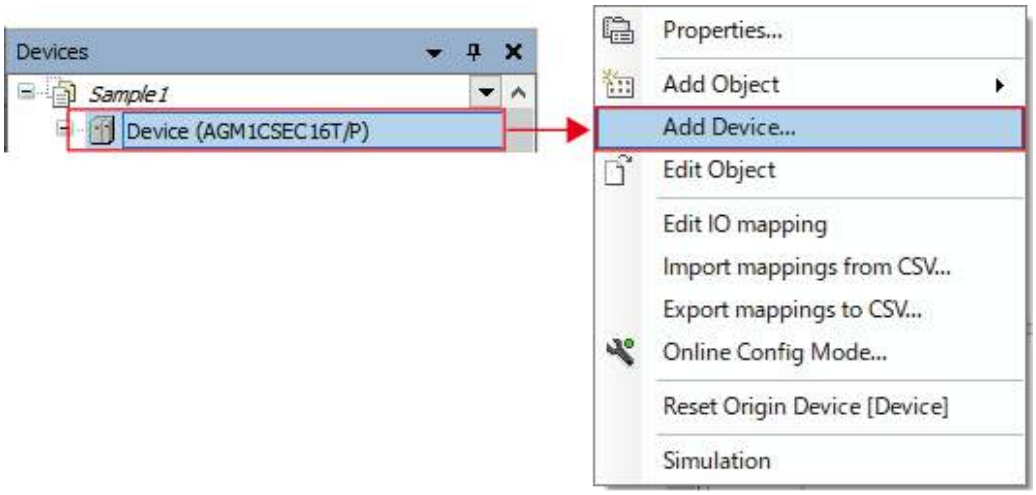
5.5.2 Adding a Protocol to Be Used for the COM Port

This section explains how to add an object of the protocol to be used for the COM port to a project and set it up.

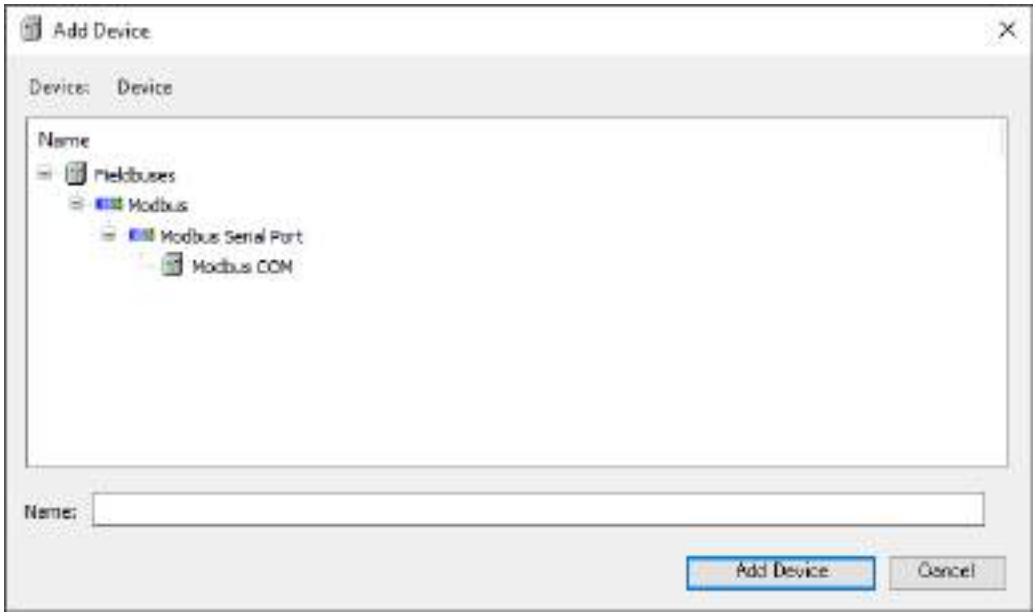
For example, use the following procedure to add an object of Modbus RTU to the COM port and set it up.

Procedure

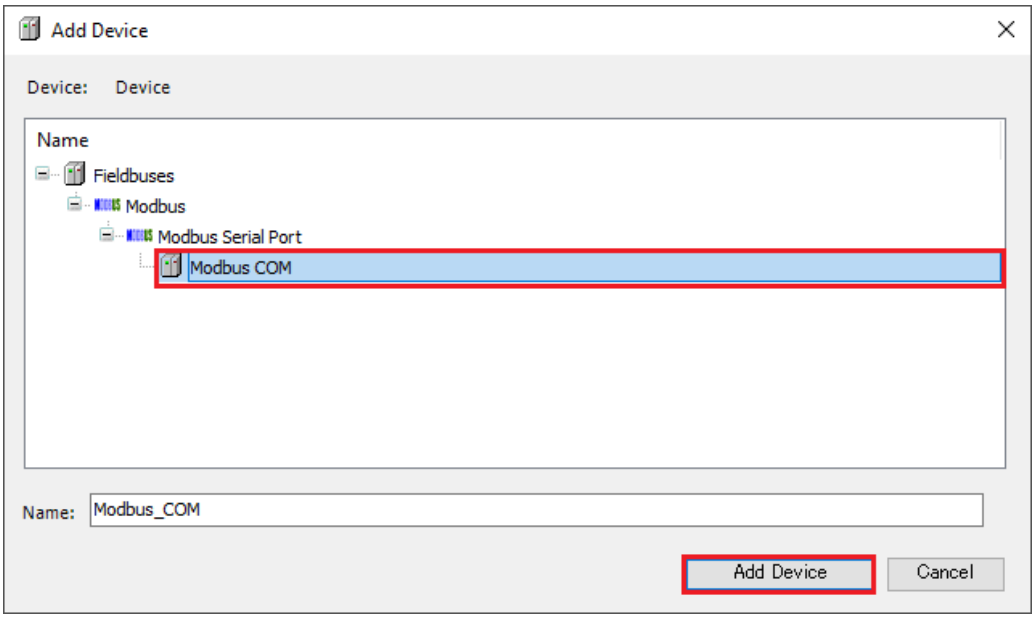
1. Right-click the [Device] object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.



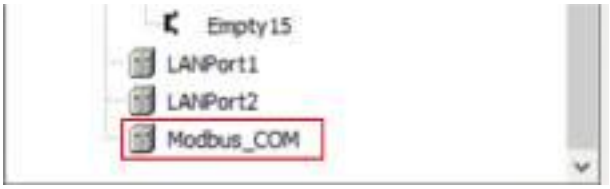
The "Add Device" dialog box will be displayed.



2. Select device "Modbus COM".



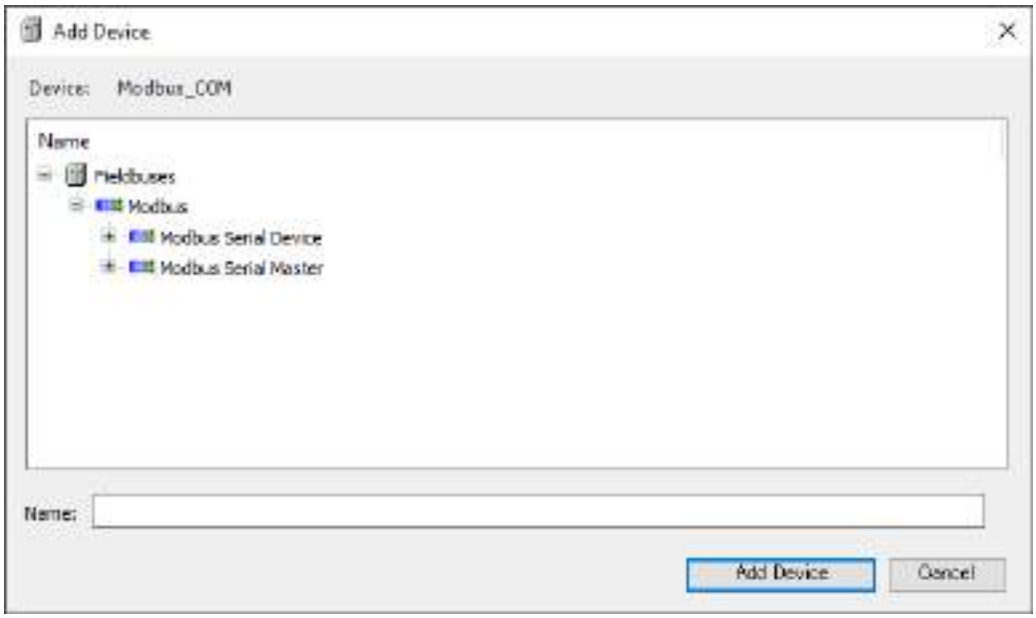
3. Click the [Add Device] button.
Object [Modbus_COM] will be added to the navigator pane.



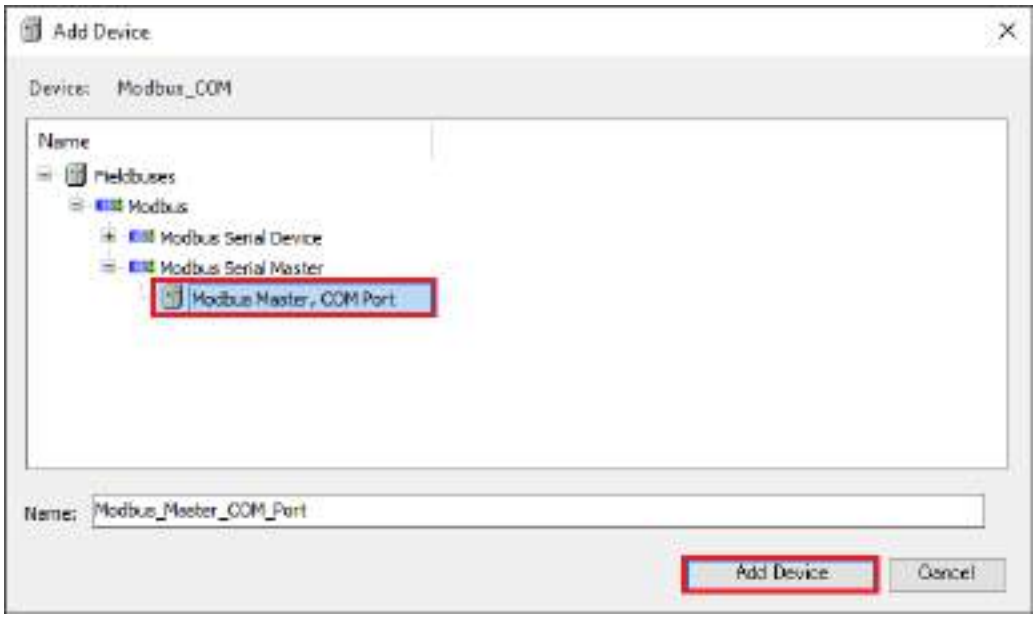
Then, add object [Modbus_COM_Master] below object [Modbus_COM].

4. Right-click the [Modbus_COM] object in the navigation pane and then select "Add Device" from the context-sensitive menu that is displayed.

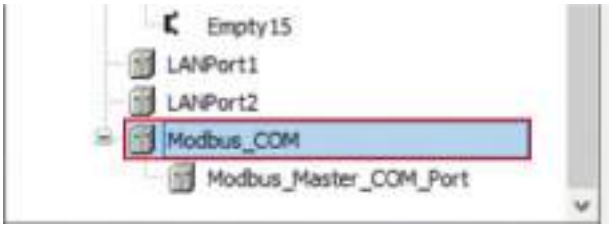
The "Add Device" dialog box will be displayed.



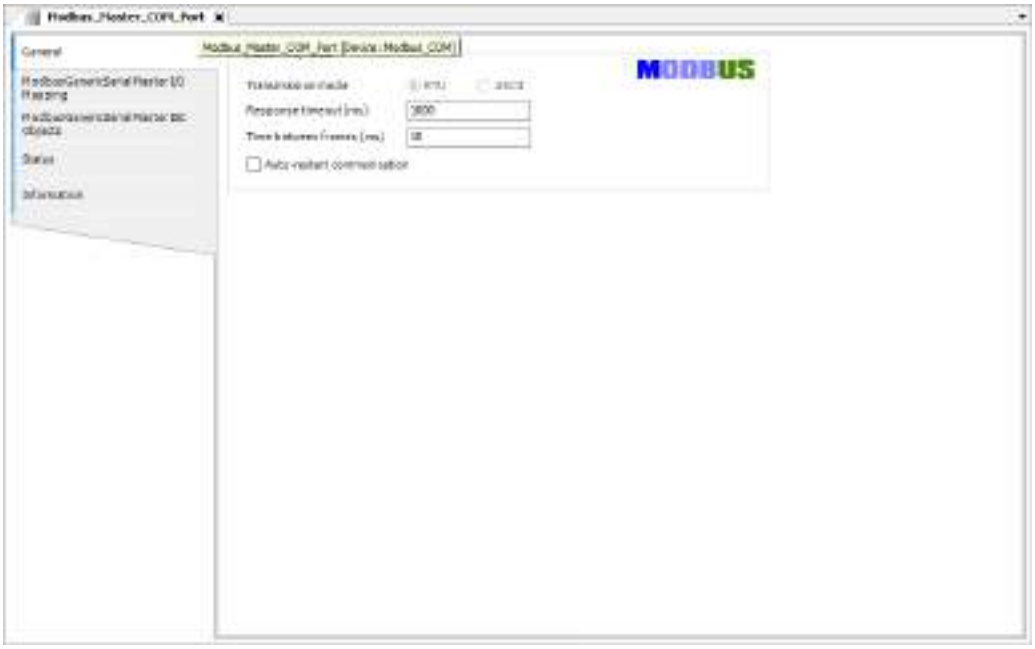
5. Select device "Modbus Master, COM Port".



6. Click the [Add Device] button.
The [Modbus_Master_COM_Port] object will be added.



7. Double-click the added [Modbus_Master_COM_Port] object.
The setting pane will be displayed in the main pane. Specify settings related to Modbus RTU.



(MEMO)

6 Program Creation

6.1 Flow of Program Creation 6-2
6.2 Program Creation Window 6-3
6.2.1 Main Pane 6-3
6.2.2 Declaration Editor 6-3
6.2.3 Auto Declaration 6-5
6.2.4 Toolbox 6-7
6.2.5 Setting up the Program Input Window 6-8
6.2.6 Window Operations for the Program Input Window 6-12
6.3 Creating a Program Object (POU Object) 6-14
6.4 Types of Programming Language 6-15
6.5 Variables 6-18
6.5.1 Standard Data Types 6-18
6.5.2 STRING type 6-19
6.5.3 WSTRING type 6-19
6.5.4 Array 6-19
6.5.5 Subrange Types 6-20
6.5.6 Structure, Enumeration, Alias, and Union Data Types 6-21
6.5.7 Constants 6-24
6.5.8 Object for Global Variable Declaration 6-25
6.5.9 Global Variables 6-25
6.5.10 Persistent Variables 6-27
6.5.11 Short Form Function 6-30
6.6 Function and Function Block 6-32
6.6.1 Function 6-32
6.6.2 Function Block 6-36

6.1 Flow of Program Creation

1. Creating a POU object

Create an object (POU object) for the program.

2. Entering program data (refer to "7.1 Programming in Ladder Diagram (LD)" through to "6.5 Variables")

Open the POU object. Enter program data, declare variables, and perform other necessary work.

3. Executing build (refer to "7.8 Build")

- Execute build and check the program.
- If there are any errors, return to "Step 2" and correct the program.

4. Registering for a task (refer to "7.9 Tasks")

Register the POU objects (to be executed on the GM1 controller) for a task.

Info.

- GM Programmer provides support functions that can be used to create programs efficiently. For details on support functions, refer to "7.7 Program Creation Support Functions".
- Programs consisting of functions and function blocks can be created. For details on functions and function blocks, refer to "6.6 Function and Function Block".

6.2 Program Creation Window

This section explains the window for creating programs in GM Programmer.

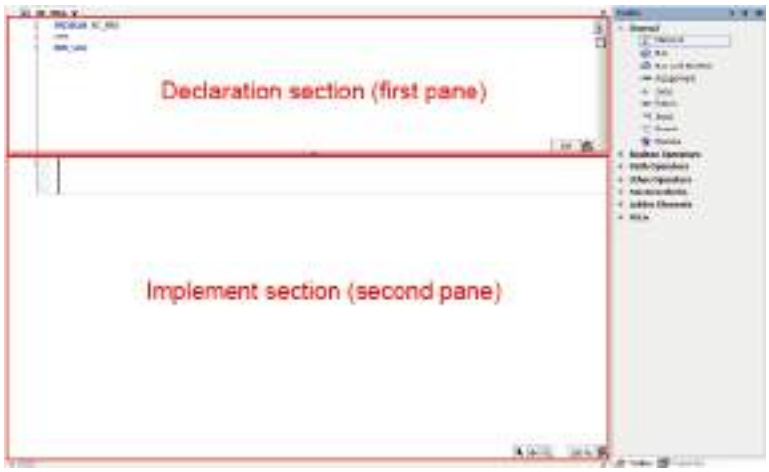
6.2.1 Main Pane

The upper section of the main pane is the declaration section where variables are declared.

The lower section of the main pane is the implementation section where program processes are coded. The declaration section and implementation section may be called the first and second panes, respectively.

The editing method in the implementation section differs according to the program. Refer to the section related to each program creation.

Example: Main pane for LD programs



Info.

- The selected pane can be switched between the declaration section (first pane) and the implementation section (second pane). To switch the selected pane, from the menu bar, select **Window**>**Next Pane** or "Previous Pane".
- You can also hide the declaration section (first pane) or the implementation section (second pane).

  To hide the declaration section, from the menu bar, select **Window**>**Switch First Pane**.

  To hide the implementation section, from the menu bar, select **Window**>**Switch Second Pane**.
- When the cursor stays in a variable position in the implementation section, the cursor can be moved to the declaration position of the variable by selecting **Edit**>**Browse**>**Go To Definition**.
- You can also declare variables of user-defined types such as structure. User-defined types must be defined in DUT objects beforehand. For details, refer to "6.5.6 Structure, Enumeration, Alias, and Union Data Types".

6.2.2 Declaration Editor

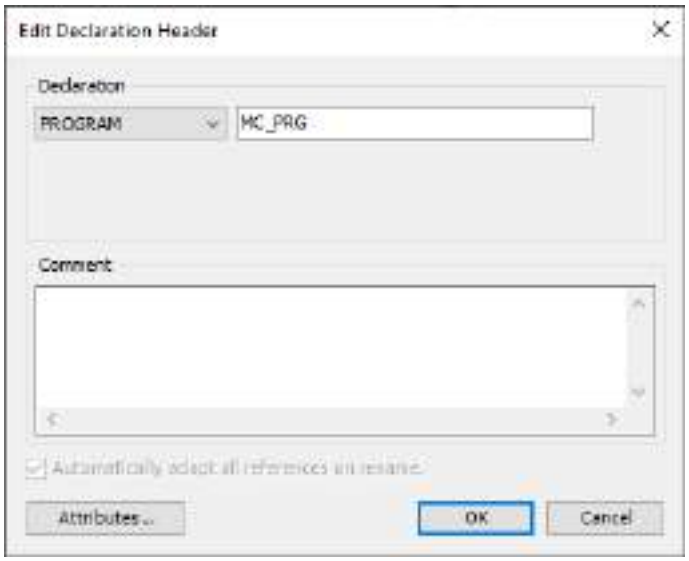
The declaration editor is used to declare variables.

The display format of declarations is divided into table format and text format. The table format and text format can be switched using the switching button on the top right of the declaration editor.

■ Table format

- To add a new declaration, click the [ ] icon ("Insert") to add a new row. Enter a variable name in the "Name" column. For other items, double-click each cell to set the cell in an input-enabled state, and enter values as necessary.
- To sort variables, use the [ ] icon ("Move up" or "Move down").
- To delete variables, use the [ ] icon ("Delete").
- When adding a program name or program name comment, click the declaration header section.

  The "Edit Declaration Header" dialog box will be displayed.



■ Text format

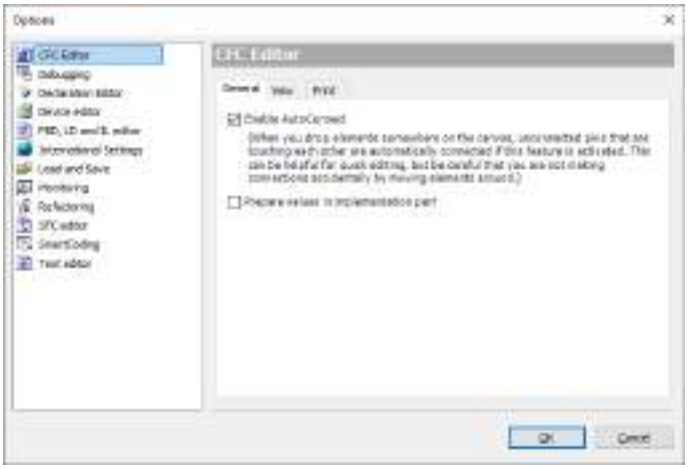
- As is the case with the text editor, enter variables to be declared.

- You can use single-line comments (prefixed with //) and multiple-line comments (enclosed with *).
- Pressing the "F2" key starts Input Assistant, which allows the user to enter variable types and other items by selecting them. For details on Input Assistant, refer to "7.7.5 Input Assistant Function".

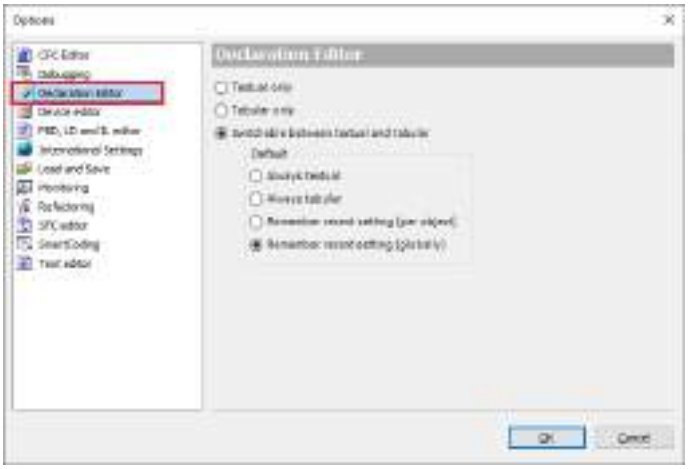
Info.

- You can set the display format to be used, as below.

1. From the menu bar, select **Tools>Options**.
   The "Options" dialog box will be displayed.



2. In the Options dialog box, select the "Declaration Editor" category.



3. Select a desired display format.

6.2.3 Auto Declaration

If a variable that has not been declared in the declaration section is entered in the implementation section, the "Auto Declare" dialog box will be displayed.

If you change any necessary items and click the [OK] button, the variable will be declared in the declaration section.

Example: When the variable name of a contact is entered as bVar0 in an LD program

■ Address

In the Address field, you can specify the address of input data or output data for the GM1 controller or expansion unit. In such a case, the variable assigned to the input data or output data corresponding to the entered address is declared.

■ Flag

If you select the CONSTANT, RETAIN, and PERSISTENT check boxes in the Flag section, you can set variable attributes.

CONSTANT

Declares the variable as a constant. Enter a default value.

RETAIN

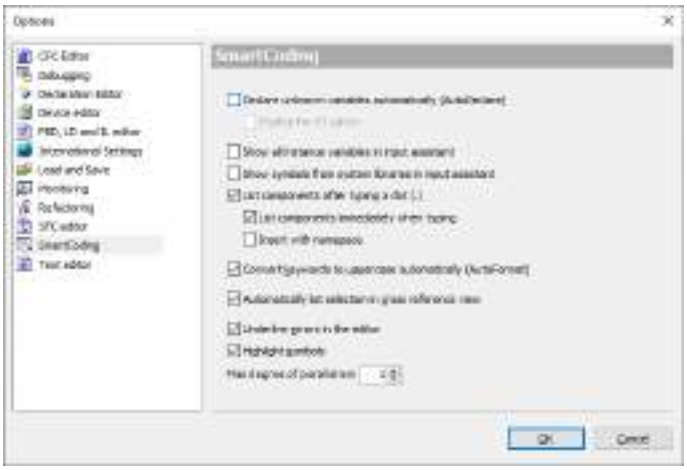
Declares the variable as a retain variable. For retain variables, values are not reset even if warm reset is performed. For details on warm reset, refer to "9.5.1 Reset Warm, Reset Cold, and Reset Origin".

PERSISTENT

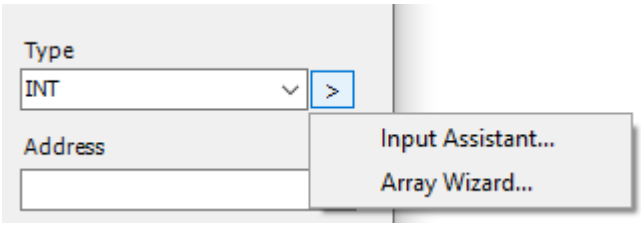
Declares the variable as a persistent variable. To declare a persistent variable, you must also select the RETAIN check box. For persistent variables, values are not reset even if cold reset or warm reset is performed.For details on cold reset or warm reset, refer to "9.5.1 Reset Warm, Reset Cold, and Reset Origin".

Info.

- You can also prevent the "Auto Declare" dialog box from being displayed when a variable that has not been declared is entered in the implementation section. From the menu bar, select **Tools**>**Options**>**SmartCoding** category and clear the "Delcare unknown variables automatically (AutoDeclare)" check box.



- Using the Array Wizard, you can declare an array only by entering an index and base type. Click > on the right side of the Type field and select "Array Wizard".

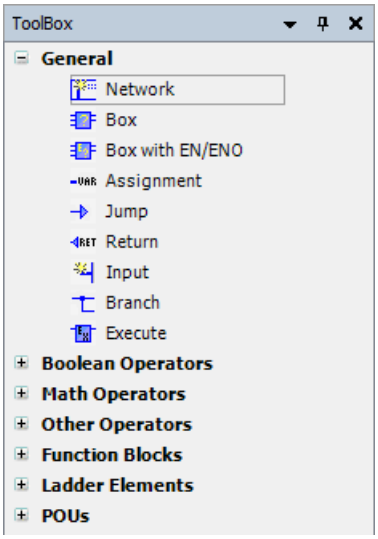


6.2.4 Toolbox

Programs can be created by dragging the programming elements displayed in the toolbox. For programs other than ST programs, programming elements are displayed in the toolbox.

For details on how to create programs, refer to the section related to each program creation.

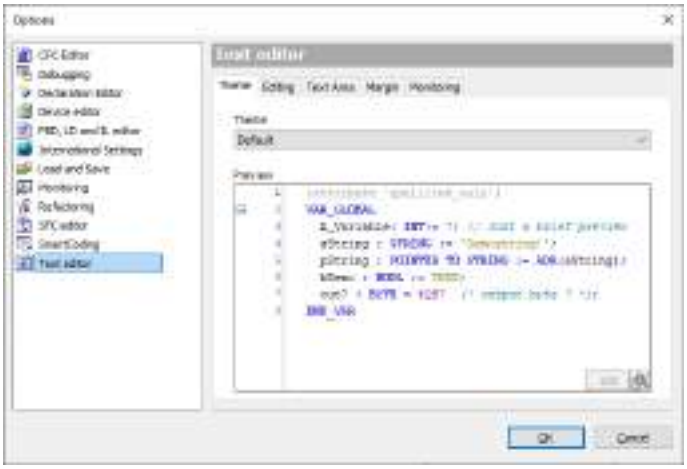
Example: Toolbox for LD programs



6.2.5 Setting up the Program Input Window

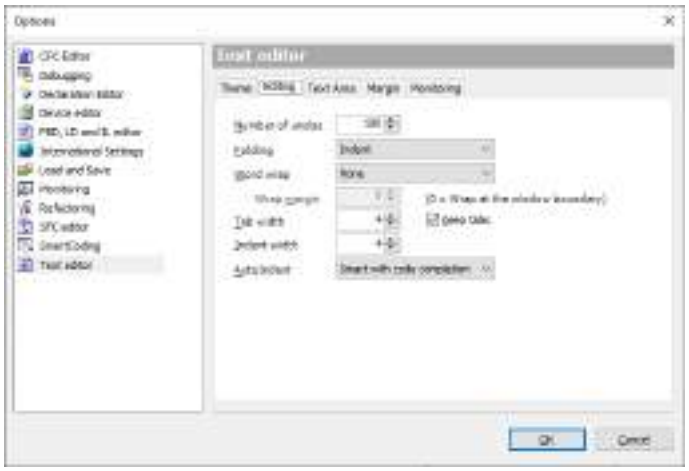
You can change settings related to the text editor.
From the menu bar, select **Tools**>**Options** to open the "Options" dialog box.
In the "Options" dialog box, select the "Text editor" category and change the settings.

Theme



Item name	Default value	Settings
Theme	Default	Sets a color scheme theme for the text editor Default / Dark

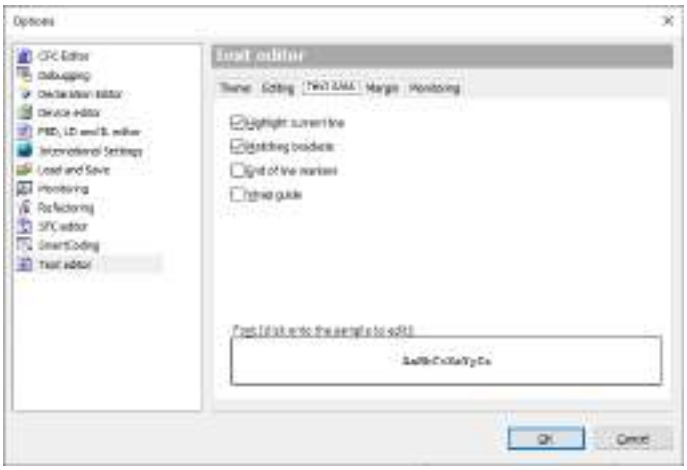
Edit



Item name	Default value	Settings
Number of undos	100	Sets the maximum number of times [Edit]-[Undo] can be executed from the menu bar Setting range: 1 to 1000 (times)
Folding	Indent	Specifies the method for defining the code structure **None:** Does not collapse code **Indent:** Combines all lines indented from the previous line as a unit **Explicit:** Explicitly marks a unit of code sections as a comment
Word wrap	None	Sets a rule for wrapping entered text automatically **None:** Does not collapse code **Soft:** If the number of characters entered in a single line exceeds the value specified in “Wrap margin”, a code continuation mark (:) will be added and a line break will be inserted automatically. If "0" is selected in the "Wrap margin" drop-down list, a line break will be inserted at the right edge of the editor window. **Hard:** If the number of characters entered in a single line exceeds the value specified in "Wrap margin", a line break will be inserted automatically. However, a code continuation mark (":") will not be added. If the number of initially entered word characters exceeds the value specified in "Wrap margin", a line break will not be inserted.
Wrap margin	0	Specifies the number of characters per line that triggers a line feed Setting range: 0 to 240
Tab width	4	Specifies the number of space characters equivalent to the code to be inserted when the Tab key is pressed Setting range: 1 to 16

Item name	Default value	Settings
Keep tabs	Selected	Specifies whether to insert space characters or a tab character when the Tab key is pressed Selected: Inserts a tab character when the Tab key is pressed Cleared: Inserts space characters when the Tab key is pressed
Indent width	4	Inserts tab spaces with the specified width when "Auto" or "Auto coding" is selected from the "Auto indent" drop-down list. However, if the "Keep tabs" check box is cleared, space characters will be inserted. Setting range: 1 to 16
Auto Indent	Smart with code completion	Specifies the behavior to be performed when auto indentation is performed **None:** Does not insert indentation automatically **Block:** Inserts indentation with the same width as that of the previous line at the time of line feed **Auto:** Inserts indentation automatically for lines following a line containing keywords (such as VAR) according to the setting of "Indent width" **Smart with code completion:** Inserts applicable keywords such as "END_IF" and "END_VAR" automatically, in addition to the behavior performed by "Auto"

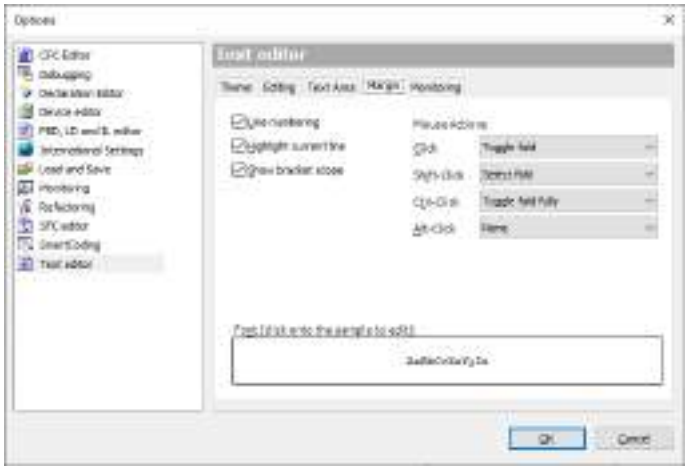
Text Area



Item name	Default value	Settings
Highlight current line	Selected	Highlights the line selected by the cursor Selected / Cleared
Matching brackets	Selected	Highlights the corresponding bracket when the cursor is positioned at a bracket within code Selected / Cleared
End of line markers	Cleared	Indicates the end of a line as a small dash mark (".") with the color specified for the theme

Item name	Default value	Settings
		Selected / Cleared
Wrap guide	Cleared	Displays a guide as the vertical line specified for the theme, in the column used as the base of wrapping. If any value other than "0" is specified in "Wrap margin", a guide will be displayed. Selected / Cleared
Font	-	Displays a font dialog box for setting fonts.

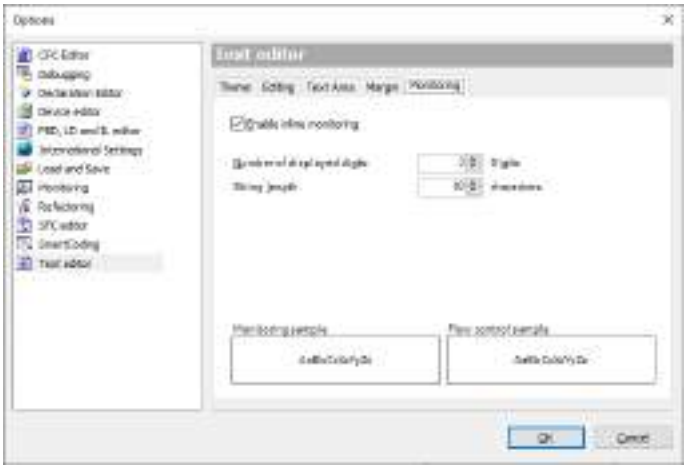
Margin



Item name	Default value	Settings
Line numbering	Selected	Displays line numbers in the variable declaration section and program implementation section
Highlight current line	Selected	Selected: Displays line numbers Cleared: Does not display line numbers Highlights the line selected by the cursor by changing the color of the line number. If the "Line numbering" check box is cleared, the current line will not be highlighted. Selected: Highlights the current line by changing the color of the line number Cleared: Does not change the color of the line number
Show bracket scope	Selected	Displays a scope in the space on the left side of the line number to indicate the beginning to the end of a keyword (such as "IF" to "END_IF") Selected: Displays a scope Cleared: Does not display a scope
Mouse Actions	-	Assigns a mouse action to be performed when "+" or "-" in a space is clicked None: Assigns no mouse action Select fold: Selects all lines within the area enclosed in brackets Toggle fold: Expands or folds the area enclosed in brackets

Item name	Default value	Settings
		Toggle fold fully: Expands or folds all nested areas when areas are nested
Font	-	Displays a "Font" dialog box for setting fonts

Monitoring

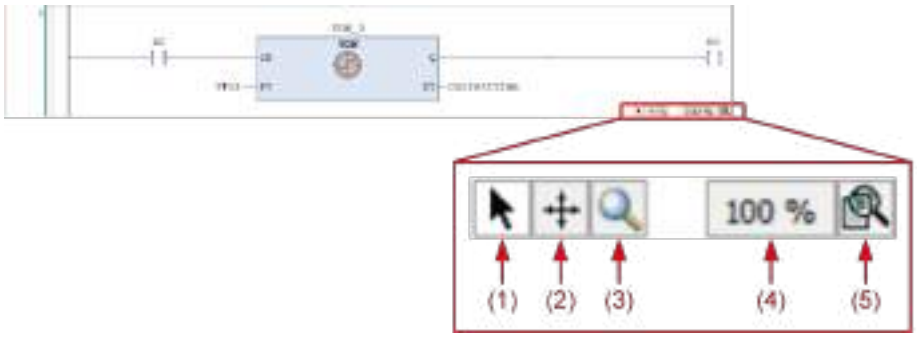


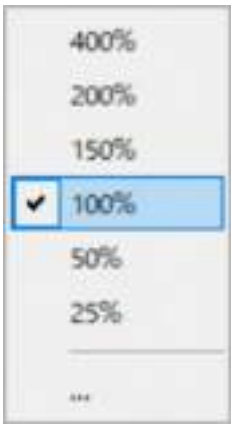
Item name	Default value	Settings
Enable inline monitoring	Selected	Displays a monitoring field in the program implementation section in online mode Selected: Displays a monitoring field Cleared: Does not display a monitoring field
Number of displayed digits	3	Sets the number of digits after the decimal point that are displayed in the monitoring field Setting range: 1 to 20
String length	10	Sets the maximum length of string variables in the monitoring field Setting range: 1 to 80

6.2.6 Window Operations for the Program Input Window

You can perform window operations such as increasing the display size in the program input window.

The window operation icons are displayed in the bottom right corner of the window.



Number	Item	Description
(1)	Normal mode	This mode allows the user to select a component by clicking it.
(2)	Move mode	This mode allows the user to moves the window by clicking in the window and then dragging the mouse.
(3)	Enlarged view tool	Clicking the icon opens the enlarged view tool window. The display at the cursor position is enlarged in the window.
(4)	Current display size	This section indicates the current display size of the program input window
(5)	Change display size	This icon is used to change the display size. Clicking the icon displays a menu. Select a desired size.  Selecting ... displays the "Enlarge" dialog box where you can enter a magnification rate.

6.3 Creating a Program Object (POU Object)

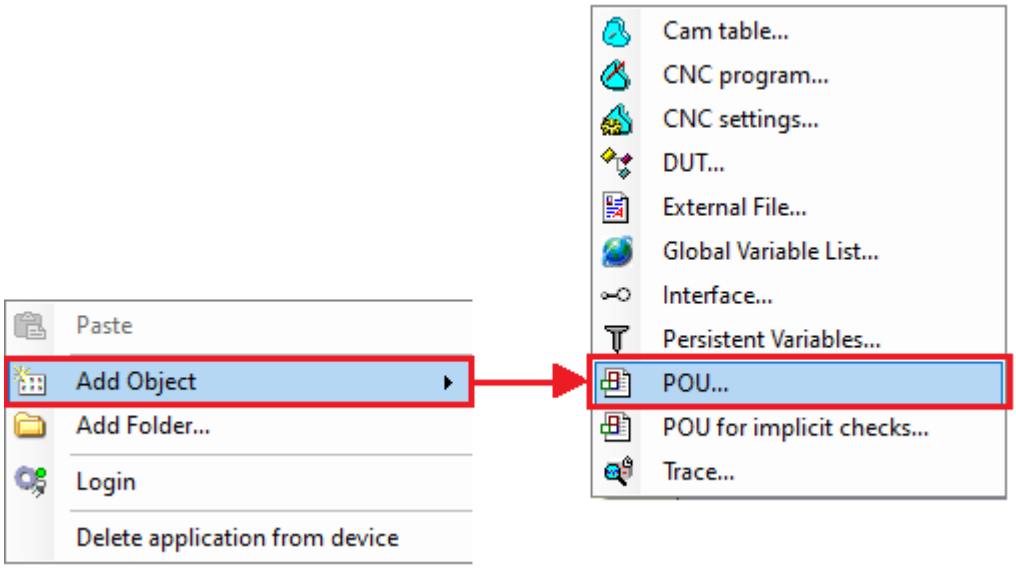
■ Creating programs

Programs are created within POU objects. A single POU object allows the use of only one program. If you want to use different programs within a project, you must add POU objects.

■ Adding POU objects

To add POU objects, right-click the [Application] object in the navigator pane and then select **Add Object**>**POU** from the context-sensitive menu that is displayed.

For details, refer to "4.7.1 Adding Objects".



6.4 Types of Programming Language

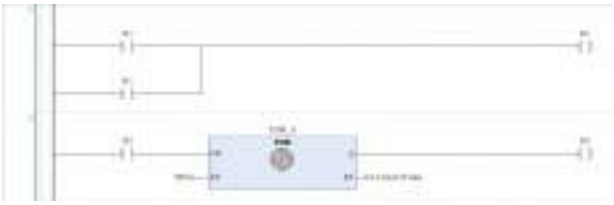
GM Programmer supports five programming languages that comply with IEC 61131-3, the international standard for PLC programming languages.

Info.

- Continuous Function Chart (CFC) and Page-Oriented CFC are not included in the five programming languages compliant with IEC 61131-3. However, the third edition of IEC 61131-3 defines them as object-oriented programming languages.

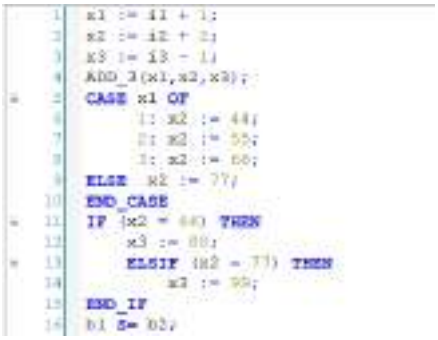
■ Ladder Diagram program (LD program)

Ladder Diagram is a graphical programming language used to create a program by arranging ladder logic elements such as contacts and coils on a network (circuit). It also allows the use of functions and function blocks with various functions.



■ Structured Text program (ST program)

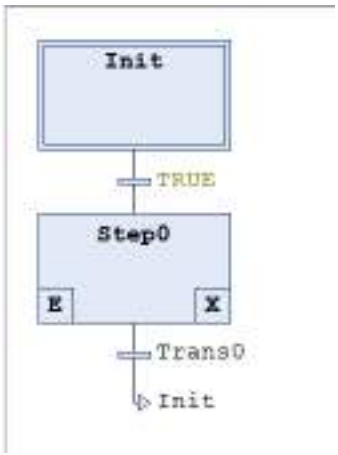
Structured Text is a programming language that creates expressions, conditional statements, and other program elements in text format. It is based on the Pascal programming language and suitable for numerical calculation, data processing, and processing such as conditional branch and repetitive processing.



```
x1 := i1 + 1;
x2 := i2 + 2;
x3 := i3 - 1;
ADD_3(x1,x2,x3);
CASE x1 OF
     1: x2 := 44;
     2: x2 := 55;
     3: x2 := 66;
ELSE  x2 := 77;
END_CASE
IF (x2 = 66) THEN
    x3 := 88;
    ELSIF (x2 = 77) THEN
        x3 := 99;
END_IF
b1 := 83;
```

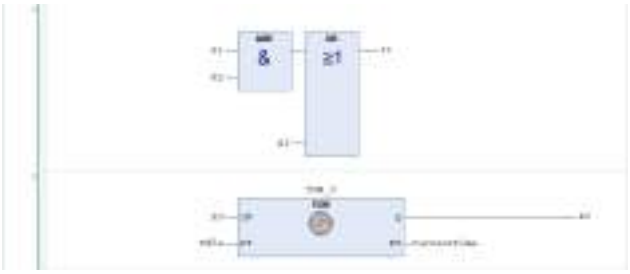
■ Sequential Function Chart program (SFC program)

Sequential Function Chart is a graphical programming language used to create a program by arranging steps, transitions, actions, and other elements sequentially from top to bottom. It is suitable for processing that describes state transitions.



■ Function Block Diagram program (FBD program)

Function Block Diagram is a graphical programming language used to create a program by arranging functions and function blocks on a network (circuit). Unlike Ladder Diagram programs, contacts, coils and other ladder logic elements cannot be arranged in Function Block Diagram programs.



In addition to the five programming languages compliant with IEC 61131-3, Continuous Function Chart (CFC) and Page-Oriented CFC can also be used.

■ Instruction List program (IL program)

Instruction List is a programming language that creates assembler-like instructions sequentially in text format. This language is suitable when you want to perform high-speed processing, restrict the memory usage, and perform other similar operations.

```
1   LD      iVar
    ADD     3
    ST      iResult
    LD      bVar1
    JMPC    mark1
2   mark1:
    LD      bVar2
    S       bVar3
```

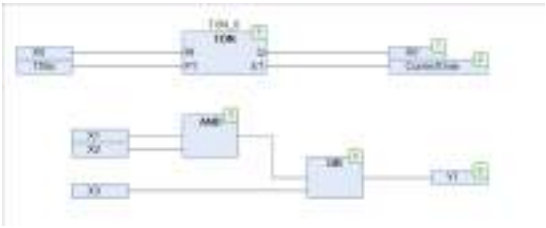
In addition to the five programming languages compliant with IEC 61131-3, Continuous Function Chart (CFC) and Page-Oriented CFC can also be used.

■ Continuous Function Chart (CFC) and Page-Oriented CFC programs

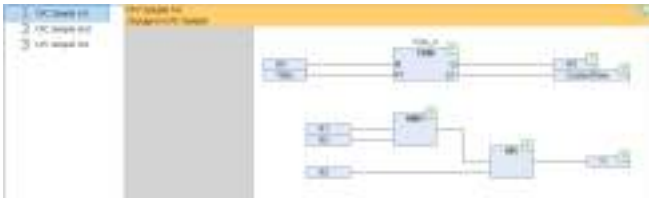
Continuous Function Chart (CFC) and Page-Oriented CFC are graphical programming languages used to create a program by arranging function blocks and other elements on the screen. Elements can be freely arranged on the screen and the order of execution can be specified.

CFC allows the user to create a program on a single screen, while Page-Oriented CFC allows the user to create a program while switching the screen called a page.

<CFC program>



<Page-Oriented CFC program>



6.5 Variables

Variables are declared in the main pane of POU objects for programs.

6.5.1 Standard Data Types

GM Programmer allows the following data types to be used as standard data types.

Type	Type	Range	Size (in bits)
Truth	BOOL	TRUE (1) and FALSE (0)	8
Integer	BYTE	0 to 255	8
Integer	WORD	0 to 65535	16
Integer	DWORD	0 to 4294967295	32
Integer	LWORD	0 to 2^{64}-1	64
Integer	SINT	-128 to 127	8
Integer	USINT	0 to 255	8
Integer	INT	-32768 to 32767	16
Integer	UINT	0 to 65535	16
Integer	DINT	-2147483648 to 2147483647	32
Integer	UDINT	0 to 4294967295	32
Integer	LINT	-2^{63} to 2^{63}-1	64
Integer	ULINT	0 to 2^{64}-1	64
Floating-point number	REAL	-3.402823e+38 to 3.402823e+38	32
Floating-point number	LREAL	-1.7976931348623158e+308 to 1.7976931348623158e+308	64
Character string	STRING		(Number of characters + 1) × 8
Character string	WSTRING		(Number of characters + 1) × 16
Time	TIME	0 to 4294967295	32
Time	LTIME	0 to 213503d23h34m33s709ms551us615ns	64
Time	TIME_OF_DAY(TOD)	0 (00:00:00:000) to 4294967295 (11:59:59 PM:999)	32
Date	DATE	0 (1970-01-01) to 4294967295 (2106-02-07)	32
Date and time	DATE_AND_TIME(DT)	0 (1970-01-01,00:00:00) to 4294967295 (2106-02-07,06:28:15)	32

Info.

- You can also use user-defined data types such as structure, enumeration, alias, and union. For details, refer to "6.5.6 Structure, Enumeration, Alias, and Union Data Types".

6.5.2 STRING type

The STRING type data can be used by enclosing the data with single quotation marks.

Usable characters are half-width English letters (a to z and A to Z), Arabic numerals (0 to 9), symbols, and space characters in the ASCII code.

Possible to set the memory size when declaring variables.

Declaration example of a 35-letter character string:

str : STRING(35):= 'This is a String';

Info.

- In principle, there are no restrictions on the length of a character string. However, when using character string functions described in Section 3.9 of Instruction Edition, only the length of 1 to 255 characters are processed.
  Any characters exceeding 255 characters will be truncated from the right.

6.5.3 WSTRING type

The WSTRING type data can be used by enclosing the data with double quotation marks.

Usable characters are Unicode characters.

Typical example: ASCII characters, hiragana characters, katakana characters, kanji characters, symbols, ancient characters, Korean characters, etc.

Some of them may not be used correctly.

Possible to set the memory size when declaring variables.

Declaration example of a 35-letter character string:

wstr : WSTRING(35):= "This is a WString";

6.5.4 Array

GM Programmer allows the use of arrays.

Using arrays enables multiple data items to be used as a single variable.

This feature is useful when variables of the same type are handled collectively.

Example: When one-dimensional array a1 with eight INT type data items is declared and used in an ST program

```
ST_POU ×
1  PROGRAM ST_POU
2  VAR
3      a1: ARRAY [0..7] OF INT;
4      i: INT;
5  END_VAR
6

1  i := 0;
2  FOR i := 0 TO 7 DO
3      a1[i] := i;
4  END_FOR;
5
```

Info.

- Whether indexes are within the declared range can be automatically checked when variables in an array are accessed. Use a POU for implicit checks that checks boundaries. For details, refer to "10.7 POU for implicit checks".
- Using the array wizard for auto declaration enables array variables to be declared only by entering index and base types. For details, refer to "6.2.3 Auto Declaration".

6.5.5 Subrange Types

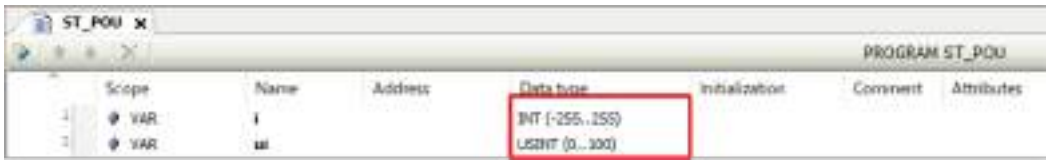
GM Programmer allows the use of subrange types. Subrange types allow the user to specify the range of values for standard data types.

The following is a declaration example of subrange type variables in character string format.

If an attempt is made to substitute an out-of-range value for a variable, an error will be displayed during build.

```
ST_POU ×
1  PROGRAM ST_POU
2  VAR
3      i : INT (-255..255);   → INT type value within the range between -255 and 255
4      ui : USINT (0..100);   → USINT type value within the range between 0 and 255
5  END_VAR
6

1  i  := 300;
2  ui := 105;                 → Out-of-range error
3
```

Subrange types can also be declared in table format.



Info.

- Whether values are within the declared range can be automatically checked when subrange type variables of the DINT, UDINT, LINT, or ULINT data type are accessed. Use a POU for implicit checks that checks ranges. For details, refer to "10.7 POU for implicit checks".

6.5.6 Structure, Enumeration, Alias, and Union Data Types

User-defined structure, enumeration, alias, and union data types can be declared using DUT objects.

When using these data types, add DUT objects to the project.

Procedure

1. Right-click the [Application] object in the navigator pane and then select **Add Object**>**DUT** from the context-sensitive menu that is displayed.

The "Add DUT" dialog box will be displayed.

The name in the "Name" field will be used as the name when the data unit type is accessed by programs.

2. Select a data type to be defined, enter the necessary information, and then click the [Add] button.

   A DUT object for defining the selected data type will be added to the navigator pane.

Example: Adding a structure data type

3. Select an object to be added and enter definitions in the main pane.
How to define and use each data type is described below.

Structure

The following is an example of declaring structures struct1 and struct2. struct2 is an extended structure of struct1.

To extend a structure before declaration, select the "Extends" check box in "Step 2" and enter an extension declaration.

The variable declared as a variable for structure struct2 can access the members of struct1 and struct2.

Example: An ST program that accesses members of structure struct2

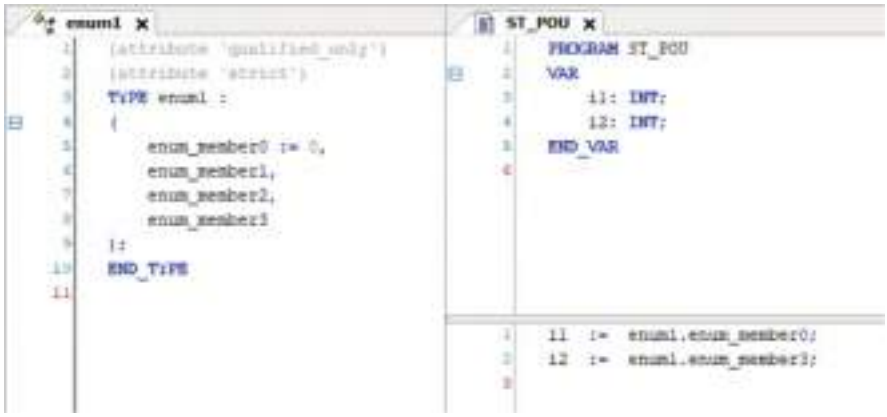
Info.

- The BIT data type can be used as members of a structure. TRUE (1) or FALSE (0) can be used as a value. The size of the BIT data type is one bit.

Enumeration

The following are a declaration that defines enumeration enum1 and an example of an ST program that accesses members of enum1.

"0" and "3" are substituted for variables iVar0 and iVar1, respectively.

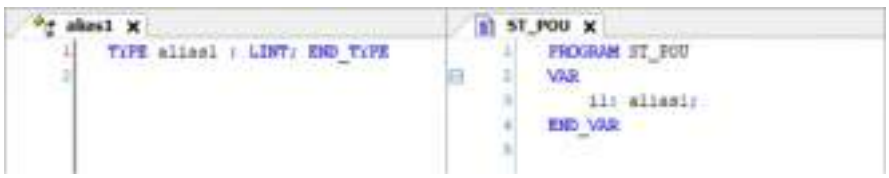


Alias

An alias can be used to assign a user-defined name as the name of a data type. Declare a variable by using an alias defined in the declaration section.

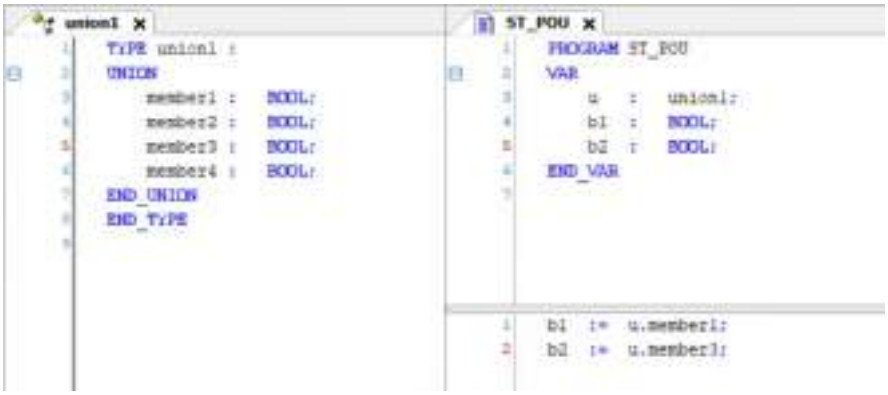
The following are an alias declaration that defines alias "alias1" for LINT and an example of a declaration section that declares variable iVar0 of the alias1 data type.

Variable iVar0 declared as the alias1 data type is handled as a variable for the LINT data type.



Union

The following are a union declaration that defines union "union1" and an example of an ST program that accesses members of union1.



6.5.7 Constants

GM Programmer allows the use of constants.

Constants are declared according to the following syntax.

VAR CONSTANT

Constant name:Type:=Default value;

END_VAR

Type	Constant type	Description
BOOL	BOOL	TRUE (1), FALSE (0)
Integer	Types that can be used as numerical values	Binary, octal, decimal, and hexadecimal numbers For numbers other than decimal numbers, integer constants are entered after number base and #. Examples: 14, 2#0101, 8#27, 16#34AB
Decimals and exponents	REAL / LREAL	Decimals and exponents Examples: 1.4, 2.34e+008
Time	TIME	32-bit time constants compliant with IEC 61131-3 Syntax: t#, T#, time#, TIME# Examples: T#12ms, T#12h32m24s
Time	LTIME	64-bit time constants In addition to TIME constants, the following units can be used. Microsecond: m Nanosecond: ns Syntax: LTIME# Example: LTIME#123m456ns
Time	TIME_OF_DAY	Time Syntax: tod#, TOD#, time_of_day#, TIME_OF_DAY# Example: tod#12:24:20.123
Date	DATE	Date

Type	Constant type	Description
		Syntax: d#, D#, date#, DATE# Example: d#2018-01-01
Date and time	DATE_AND_TIME	Date and time Syntax: dt#, DT#, date_and_time#, DATE_AND_TIME# Example: dt#2018-01-01-07:04:13
Character string	STRING	Enclosed with single quotation marks Example: 'Hello World'
	WSTRING	Enclosed with double quotation marks Example : "Hello World"

6.5.8 Object for Global Variable Declaration

GM Programmer allows the use of global variables that can be used within the entire project.

■ Global variable list

This is an object for declaring global variables.

Variables declared in the global variable list can be accessed by using "object-name.global-variable-name" in the global variable list.

Example: When variables in global variable list "Object GVL" are accessed by an ST program

For details, refer to "6.5.9 Global Variables".

■ Persistent variable list

This is an object for declaring global variables that are persistent variables.

From "Add Object", select "Persistent Variables".

For details, refer to "6.5.10 Persistent Variables".

6.5.9 Global Variables

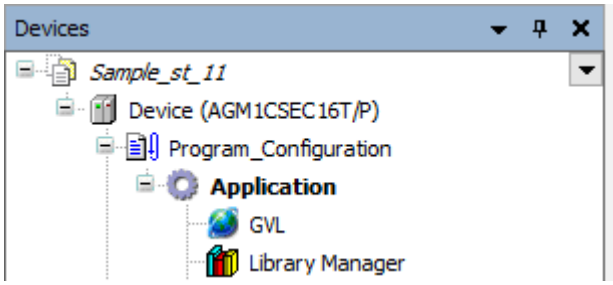
GM Programmer allows the use of global variables that can be used with all projects.

Global variables are declared within the global variable list (GVL) object.

This section explains how to declare global variables and access the declared variables.

1 2 Procedure

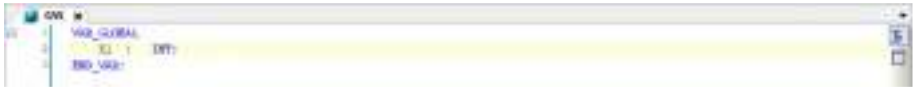
1. Double-click the GVL object in the navigator pane.



The GVL window will be displayed in the main pane.

2. Declare variables in the global variable list (GVL).

Example: Declaring global variable g_iVar0 of INT data type



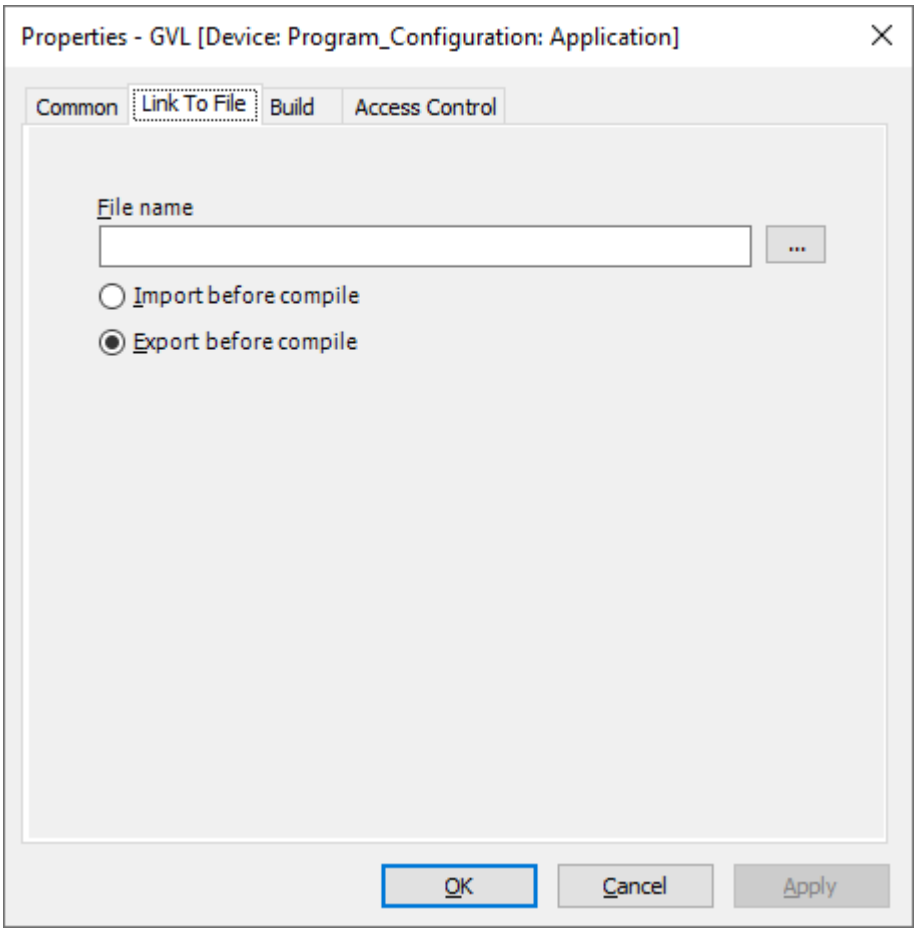
The declared variable can be accessed from the program by using "name.variable name".

Example: Substituting value "5" for global variable g_iVar0

Info.

- Variables declared before the build process can be imported and exported in XML format.

  Right-click the object in the global variable list and then select "Properties" from the context-sensitive menu that is displayed. The "Properties" dialog box will be displayed. Open the "Link To File" tab window, select either the check box for import or the check box for export, and enter the path to the file to be imported or exported in the File name field.



6.5.10 Persistent Variables

GM Programmer allows the use of global variables that are persistent variables which hold values without initializing them at the time of reset.

Persistent variables that can be used as global variables are declared within the persistent variable list object.

Only one persistent variable list object can be registered.

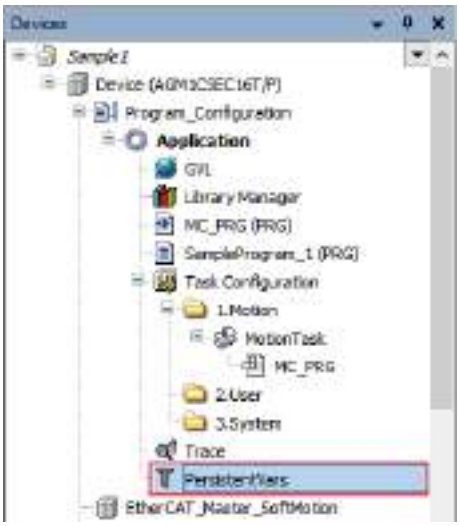
Procedure

1. Right-click the [Application] object in the navigator pane and then select **Add Object>Persistent Variable** from the context-sensitive menu that is displayed.

The "Add Persistent Variable" dialog box will be displayed.

2. Enter a name for the persistent variable list and click the [Add] button.
A "persistent variable list" object will be added to the navigator pane.

Example: Persistent variable list object with name "PersistentVars"



3. Declare variables in the persistent variable list.

Example: Declaring global variable g_iVar0 that is a persistent variable of INT type

The declared variable can be accessed from the program by using "name.variable name".

Example: Substituting value "6" for global variable g_iVar0 that is a persistent variable

Info.

- Persistent variables used as local variables can be declared (as VAR PERSISTENT RETAIN) in the declaration section for each POU object.
- Instance paths of persistent variables declared in each POU object can be added to the persistent variable list.

  With the persistent variable list declaration section selected, from the menu bar, select **Declare**>**Add All Instance Paths**.

6.5.11 Short Form Function

Using the short form function in the declaration section in character string format enables variables to be declared by entering fewer characters.

■ **Example: Declaring Boolean variables bVar0 and bVar1 with the short form function**

Procedure

1. Enter variables bVar0 and bVar1, and press the <Ctrl> key + <Enter> key simultaneously. "bVar0, bVar1:BOOL;" will be entered automatically.

<Ctrl> key + <Enter> key

The table below shows examples of input using the short form function.

Strings entered following a semicolon (;) are treated as comments.

Input in short form	Result after the <Ctrl> key + <Enter> key are pressed simultaneously
bVar0	bVar0:BOOL;
iVar0 iVar1 I 6	iVar0, iVar1: INT := 6;

Input in short form	Result after the <Ctrl> key + <Enter> key are pressed simultaneously
strVar S 8	strVar: STRING(8)
wVar w; wVar comment	wVar: WORD; // wVar comment

6.6 Function and Function Block

Functions and function blocks can be invoked from programs. Functions and function blocks can be created with POU objects.

The differences between functions and function blocks are as below.

■ Function (FUN)

- Functions can be used without being declared in the declaration section.
- Only one output is generated. However, additional outputs can be defined.
- Output variable and internal variable values are not saved.

■ Function block (FB)

- Function blocks can be used by declaring instances in the declaration section.
- Multiple outputs can be generated.
- Output variable and internal variable values are saved.
- Object-oriented definitions can be made by using EXTENDS (inheritance), IMPLEMENTS (interface implementation), or access qualifiers.

6.6.1 Function

A function generates one output in response to one or more inputs. Functions can be used without declaring variables.

For example, use the following procedure to create and invoke function "ADD_SUB", which uses three INT type arguments as inputs, calculates (first argument) + (second argument) - (third argument), and output the result.

1 2 Procedure

1. Right-click the [Application] object in the navigator pane and then select **Add Object>POU** from the context-sensitive menu that is displayed.

The "Add POU" dialog box will be displayed.

2. Select the "Function" radio button, enter appropriate values in the Name and Return type fields, and select an appropriate programming language from the Implementation Language drop-down list.

   In the Name field, specify a function name. In the Return type field, select a return value to be output when the function is executed. From the Implementation Language drop-down list, select a programming language that is used to code function processing.

3. Click the [Add] button.

   A POU object of the function will be added.

The POU object is displayed as "function-name (FUN)" in the navigator pane.

4. Enter function processing.

Open the POU object of the function and create a function.

In "VAR_INPUT", declare input variables for the function.

Substitute function output for the variable of the function name.

This completes the function creation procedure.

Next, the procedure for invoking the created function is explained below.

5. Open the POU object from which the function is to be invoked, and invoke the function.

The function can be invoked by using its name. To invoke the function, there is no need to declare variables.

Example: Invoking the function from LD program

Example: Invoking the function from ST program

Info.

- Additional outputs can be defined for the function. Declare an additional output as variable "VAR_OUTPUT" in the declaration section of the POU object that defines the function.

Example: Definition of function "ADD_SUB" to which variable iOut that outputs the sum of three input variables is added

Invoking function "ADD_SUB" from LD program

Invoking function "ADD_SUB" from ST program

6.6.2 Function Block

A function block generates one or more outputs in response to one or more inputs. To use function blocks, variables (instances) must be declared.

For example, use the following procedure to create function block "FB_ADD", which uses three INT type variables as inputs and outputs the sum of three arguments, and to invoke an instance.

Procedure

1. Right-click the [Application] object in the navigator pane and then select **Add Object>POU** from the context-sensitive menu that is displayed.

The "Add POU" dialog box will be displayed.

2. Select the "Function block" radio button, enter a name in the Name field, and select a programming language from the Implementation Language drop-down list.

   In the Name field, specify a function block name. In the Implementation Language drop-down list, select a programming language that is used to code function block processing.

3. Click the [Add] button.
A POU object of the function block will be added.
The POU object is displayed as "function-block-name (FB)" in the navigator pane.

4. Enter function block processing.
Open the POU object of the function block and create a function block.
In "VAR_INPUT", declare input variables to the function block.
In "VAR_OUTPUT", declare output variables from the function block.

This completes the function block creation procedure.
Next, the procedure for invoking the created function block is explained below.

5. Open the POU object from which the function block is to be invoked, and declare an instance of the function block in the declaration section.

Declare an instance that is a copy of the function block.

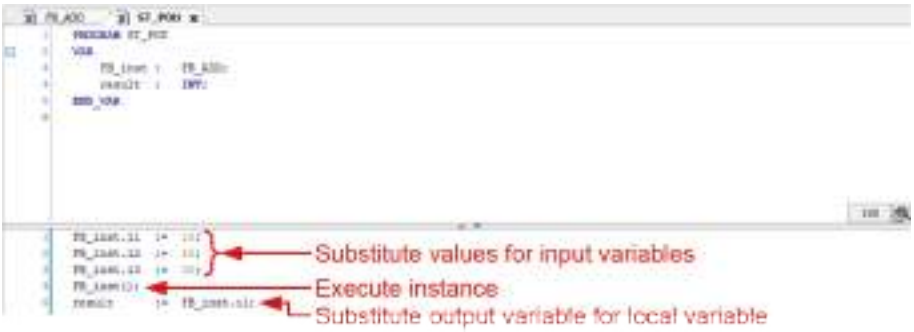
Declare an instance name in the form of "instance-name: function-block-name".

6. Invoke the instance of the function block.

Invoking the instance of the function block executes the processing defined with the function block. Input variables and output variables can be accessed by specifying "instance.variable-name".

Example: Invoking the instance from LD program

Example: Invoking the instance from ST program



7 Entering Programs in Each Programming Language

7.1 Programming in Ladder Diagram (LD) 7-3
7.1.1 Inserting Contacts, Coils, and Function Blocks 7-3
7.1.2 Inserting Contacts in Parallel 7-8
7.1.3 Inserting a Network (Circuit) 7-9
7.1.4 Inserting a Branch 7-10
7.1.5 Input of Title and Comment (LD) 7-11
7.1.6 Commenting out a Network (Circuit) 7-13
7.2 Programming in Structured Text (ST) 7-15
7.2.1 ST Program Syntax 7-15
7.2.2 Commenting out Code in ST Program 7-17
7.3 Programming in Sequential Function Chart (SFC) 7-19
7.3.1 Inserting Elements from Menu 7-19
7.3.2 Inserting Elements from Toolbox 7-21
7.3.3 Inserting Elements from Toolbar 7-22
7.3.4 Setting up the SFC Editor 7-23
7.3.5 Setting SFC Program Execution Conditions 7-25
7.4 Programming in Function Block Diagram (FBD) 7-27
7.4.1 Entering Function Blocks 7-27
7.4.2 Inserting and Commenting out a Network (Circuit) 7-32
7.4.3 Input of Title and Comment (FBD) 7-32
7.4.4 Settings in FBD Program 7-32
7.5 Programming in Instruction List (IL) 7-34
7.5.1 Entering Instructions and Operands 7-34
7.5.2 Settings in IL Program 7-36
7.6 Programming in Continuous Function Chart (CFC) 7-38
7.6.1 Inserting and Connecting Elements 7-38
7.6.2 Connection Mark 7-43
7.7 Program Creation Support Functions 7-44
7.7.1 Bookmark 7-44
7.7.2 Call Tree View 7-44
7.7.3 Cross reference List View 7-45
7.7.4 Function Block Guidance 7-46
7.7.5 Input Assistant Function 7-50
7.7.6 Argument / Variable Input Support (Component List) 7-51
7.7.7 Global Renaming (Refactoring) 7-52
7.7.8 Displaying Programs in Multiple Languages (Project Localization) . 7-55
7.8 Build 7-59

7.8.1 Build 7-59
7.8.2 Rebuild 7-59
7.8.3 Code Generation 7-59
7.8.4 Clean 7-60
7.8.5 Clean All 7-61
7.9 Tasks 7-63
7.9.1 Adding Programs 7-63
7.9.2 Adding a UserTask 7-68
7.9.3 Task Configuration Window 7-70

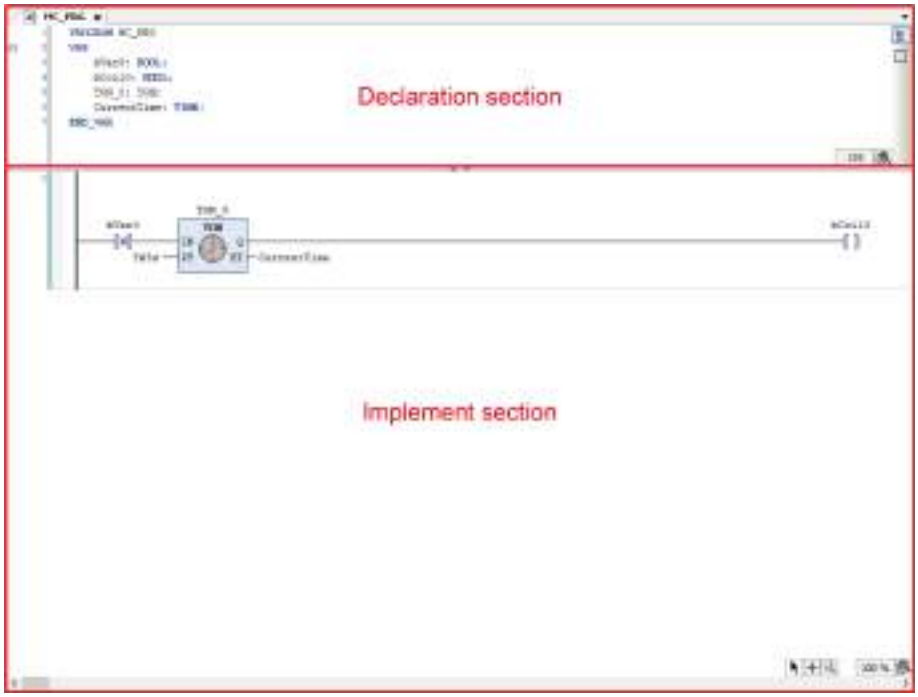
7.1 Programming in Ladder Diagram (LD)

This section explains how to create programs (LD programs) in Ladder Diagram compliant with IEC 61131-3, the international standard for PLC programming languages.

To create LD programs, POU objects for LD programs are required. Set the object setup language to Ladder Diagram (LD).

7.1.1 Inserting Contacts, Coils, and Function Blocks

This section explains how to create an LD program that consists of the normally open contact, coil, and function block TON shown below.



1 2 Procedure

1. In Toolbox, select **Ladder elements**>**Contact** and drag the mouse until "Start here" is displayed in the main pane.

   "Start here" will be displayed in the implementation section.

When you drag the mouse until the position of "Start here" is reached, the display of "Start here" turns green.

When you stop dragging the mouse at the position of "Start here", a normally open contact will be placed in the network (circuit).

Info.

- You can also insert a normally open contact in the network (circuit) using the following procedure:
  - Right-click in the network (circuit) and select "Insert Contact" from the context-sensitive menu that is displayed.
  - Click the icon on the toolbar.
  - From the menu bar, select **FBD/LD/IL>Insert Contact**.
  - Press the <Ctrl> key + <k> key simultaneously.

2. Select "???" of the normally open contact and enter variable bVar0, and then press the <Enter> key.

The "Auto Declare" dialog box will be displayed.

3. Click the [OK] button.

Variable bVar0 will be declared in the declaration section.

4. In Toolbox, select **Ladder elements>Coil** and drag the mouse until "Add output or jump here" is displayed in the main pane.

   When you drag the mouse until the position of "Add output or jump here" is reached, the display of "Add output or jump here" turns green.

   When you stop dragging the mouse at the position of Add output or jump here, a coil will be placed in the network (circuit).

5. Select "???" of the coil and enter variable bCoil0, and then press the <Enter> key.

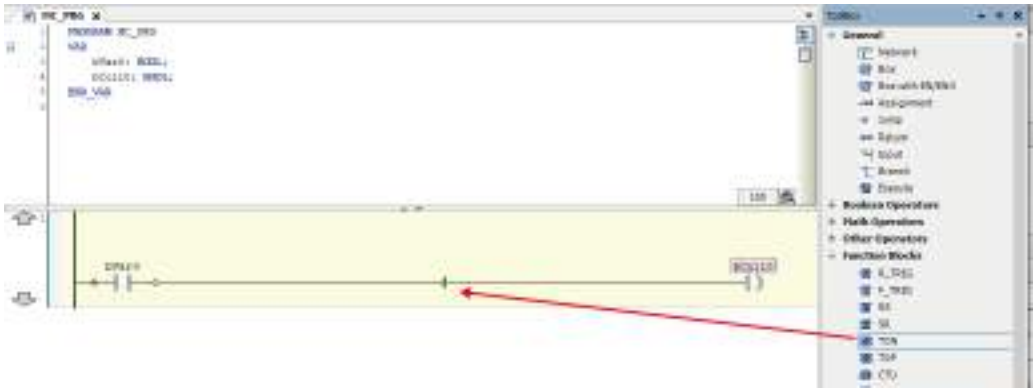
   The "Auto Declare" dialog box will be displayed.

6. Click the [OK] button.

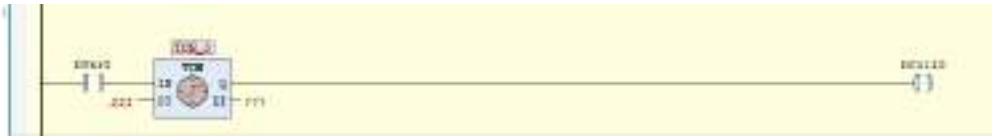
   Variable bCoil0 will be declared in the declaration section.

7. In Toolbox, select **Function block>TON** and drag the mouse until is displayed in the main pane.

Function block TON will be displayed.



TON will be placed in the implementation section.

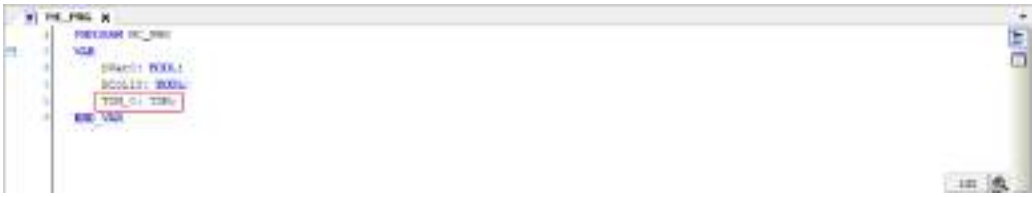


8. Select TON and enter a variable name, and then press the <Enter> key.

The "Auto Declare" dialog box for TON will be displayed.

9. Click the [OK] button.

The name of TON will be declared as variable TON_0.



10. Enter "T#5s" for input PT ("IN PT") and "CurrentTime" for output ET ("O ET"), as shown below.

In the declaration section, declare "CurrentTime" as a TIME type variable.

This completes insertion of a normally open contact, coil, and function block TON.

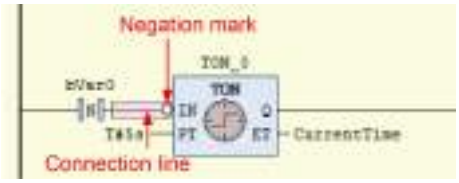
The network (circuit) is now complete.

Info.

- To remove an element that has been placed, select the element and press the <Delete> key.
  You can also remove an element by right-clicking the element and selecting "Delete" from the context-sensitive menu that is displayed.
- The normally open contact that is placed can be set as a normally open contact for detecting a rising edge ( ). To do so, perform the following procedure with the normally open contact selected.
  - Right-click the normally open contact and select "Edge Detection" from the context-sensitive menu that is displayed.
  - From the menu bar, select **FBD/LD/IL>Edge Detection**.
  - Press the <Ctrl> key + <e> key simultaneously.
  - Click the icon on the toolbar.

  If the above procedure is performed one more time, the normally open contact that is placed can be set as a normally open contact for detecting a falling edge ( ).
- Input to the function block can be negated. To do so, perform the following procedure with the connection line to the input selected.
  - Right-click the function block and select "Negation" from the context-sensitive menu that is displayed.
  - From the menu bar, select **FBD/LD/IL>Negation**.
  - Press the <Ctrl> key + <n> key simultaneously.
  - Click the icon on the toolbar.

  A negation mark will be displayed on the left side of the input ("IN").



- The LD program can be converted and displayed as an FBD program.
  From the menu bar, select **FBD/LD/IL**>**View** and select a post-conversion programming language.

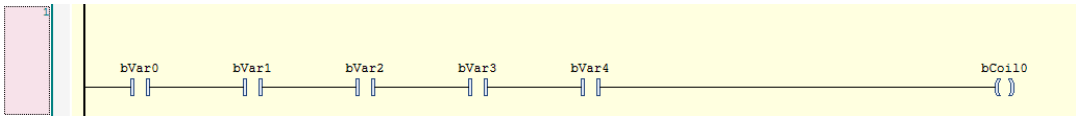
7.1.2 Inserting Contacts in Parallel

This section explains how to place a contact in parallel with a normally open contact.
In Toolbox, select **Ladder elements**>**Parallel contact** and drag the mouse until the position of displayed on the right side of the normally open contact is reached.

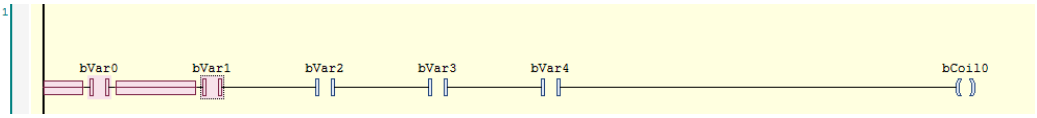
The normally open contacts will be placed in parallel with the other one.

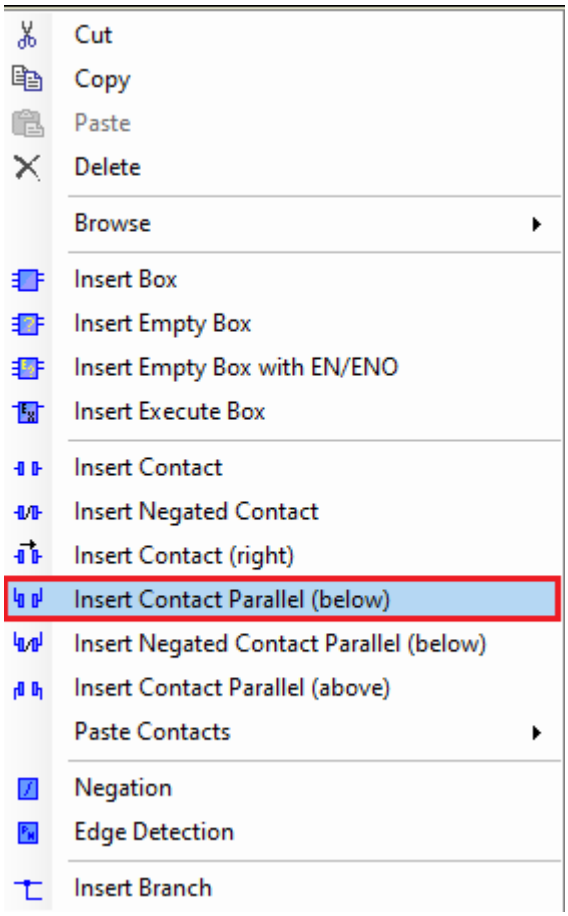
■ Placing a contact in parallel with multiple contacts

The following explains how to place a contact in parallel with multiple contacts.

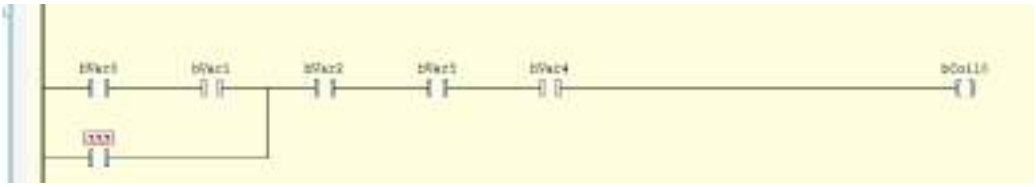


While holding down the <Ctrl> key, select relevant contacts and then right-click.





Select "Insert Contact Parallel (below)" from the context-sensitive menu that is displayed. A contact will be placed in parallel with the selected contacts.



7.1.3 Inserting a Network (Circuit)

This section explains how to insert a new network (circuit).

Procedure

1. In Toolbox, select **General**>**Network** and drag the mouse until displayed in the network (circuit) in the main pane is reached.

A network (circuit) will be inserted underneath.

Info.

- You can also insert a network (circuit) underneath using the following procedure:
  - Right-click in the network (circuit) and select "Insert Network (Below)" from the context-sensitive menu that is displayed.
  - From the menu bar, select **FBD/LD/IL**>**Insert Network (below)**.
  - Press the <Ctrl> key + <t> key simultaneously.
- To add a network (circuit) above the existing network, in Toolbox, select **General**>**Network** and drag the mouse until the position of  is reached.

- To remove a network (circuit), select the network (circuit) and press the <Delete> key.

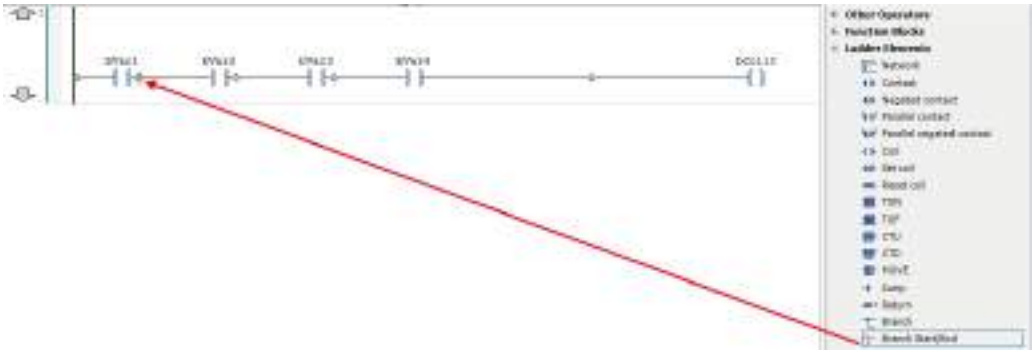
7.1.4 Inserting a Branch

This section explains how to add a branch by specifying the starting point and end point of the branch.

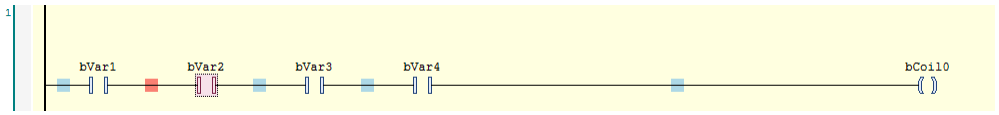
Example: Creating a branch in the following network (circuit) that extends from a point between contacts bVar1 and bVar2 to a point between contacts bVar3 and bVar4

Procedure

1. In Toolbox, select **Ladder elements**>**Branch Start/End** and drag the mouse until displayed on the right side of contact bVar1 in the main pane is reached.

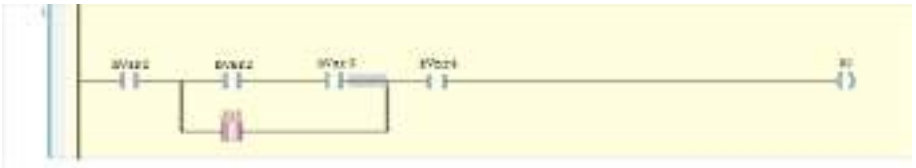


When you stop dragging the mouse, a red square mark indicating the starting point of a branch is displayed between contacts bVar1 and bVar2. Blue square marks indicate candidates for the end point of the branch.



2. Click the blue square mark between contacts bVar3 and bVar4.

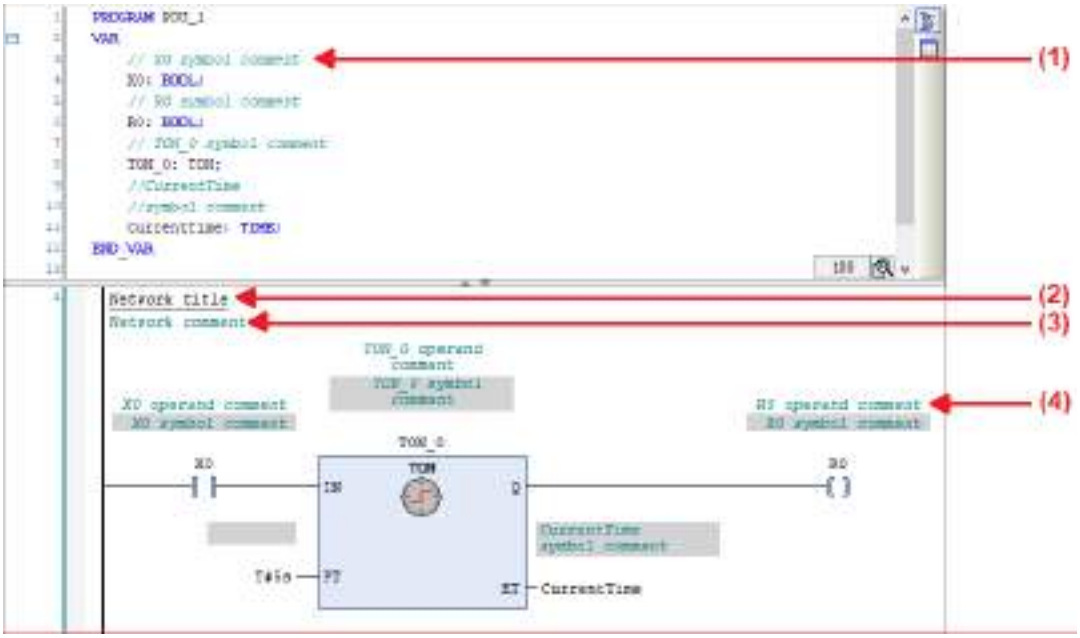
   A branch that extends from the point between contacts bVar1 and bVar2 to the point between contacts bVar3 and bVar4 will be inserted.



7.1.5 Input of Title and Comment (LD)

The Ladder Diagram programming language allows the user to enter the following four types of titles and comments.

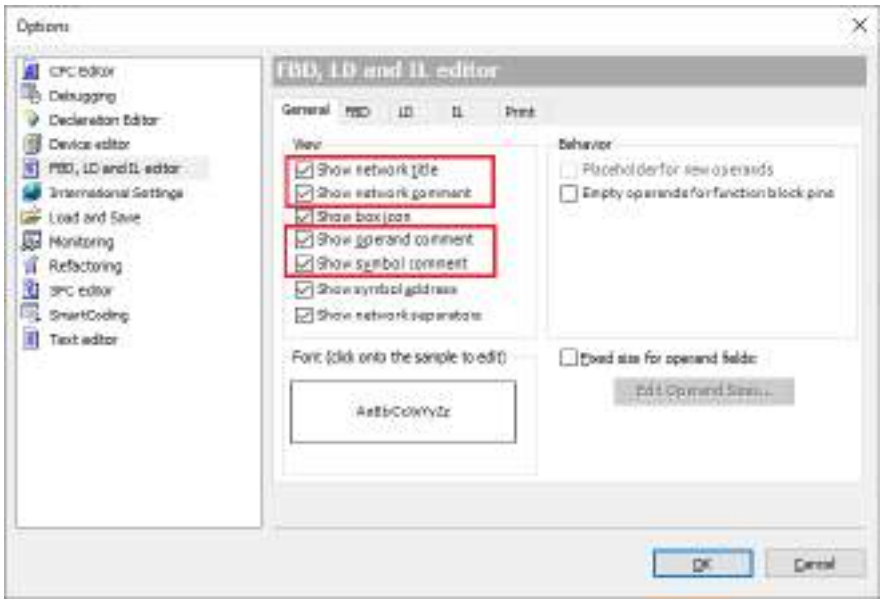
Display examples of titles and comments are shown below.



No.	Item	Description
(1)	Symbol comment	This is a comment on a declared variable. The same comment is displayed for the same variable. Enter a comment on a variable in the declaration section. The comment will be displayed in a cell with a black background.
(2)	Network title (circuit title)	A title can be assigned to each network (circuit). Click the top left of the network (circuit) and enter a title.
(3)	Network comment (circuit comment)	A comment can be assigned to each network (circuit). Click the top left of the network (circuit) and enter a comment.
(4)	Operand comment	This is a comment on a variable. Different comments can be assigned to the same variable. Click a position above each variable in the implementation section and enter a comment.

To display titles and comments, you must configure settings.

Open the Options window (by selecting **Tools>Options**), select the "FBD, LD, and IL editors" category and then the General tag, and select the items to be displayed in the Display section.



Info.

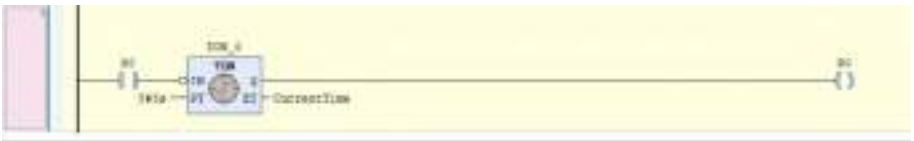
- Titles and comments can be displayed as those translated in a particular language beforehand. For details, refer to "7.7.8 Displaying Programs in Multiple Languages (Project Localization)".

7.1.6 Commenting out a Network (Circuit)

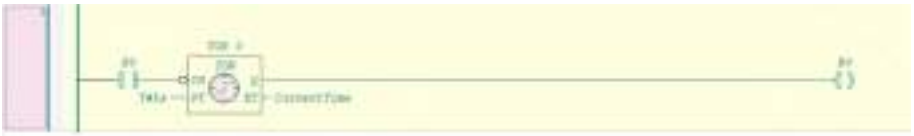
Networks (circuits) can be commented out. A network (circuit) that is commented out cannot be executed.

Procedure

1. Select a network (circuit) to be commented out.



2. Click the icon (Toggle Network Comment State) on the toolbar.
   The network (circuit) will be commented out. To cancel the comment-out state, perform the same operation again.



Info.

- You can also comment out the selected network (circuit) using the following procedure:
  - Right-click in the network (circuit) and select "Toggle network comment state" from the context-sensitive menu that is displayed.
  - From the menu bar, select **FBD/LD/IL**>**Toggle network comment state**.
  - Press the <Ctrl> key + <o> key simultaneously.

7.2 Programming in Structured Text (ST)

This section explains how to create programs (ST programs) in Structured Text compliant with IEC 61131-3, the international standard for PLC programming languages.

- To create ST programs, POU objects for ST programs are required. Set the object setup language to Structured Text (ST).
- An ST program is made up by combining expressions and instructions. Expressions and instructions can also be executed under certain conditions or within a loop. Each instruction must end with a semicolon (;).

7.2.1 ST Program Syntax

For ST programs, the following syntax can be used.

Item	Example
Assignment statement	The value of the right side is set on the left side. Example: iVar1 : = 4;
Set assignment statement	If the value of the right side is judged to be TRUE, TRUE will be set on the left side. Once the value of the left side is judged to be TRUE, the left side will maintain TRUE even if the value of the right side is judged to be FALSE. Do not leave any spaces between "S" and "=". Example: bVar0S=bVar1;
Reset assignment statement	If the value of the right side is judged to be TRUE, FALSE will be set on the left side. Once the value of the left side is judged to be FALSE, the left side will maintain FALSE even if the value of the right side is judged to be FALSE. Do not leave any spaces between "R" and "=".

Item	Example
	Example: bVar0R=bVar1;

IF instruction	Conditions are judged and subordinate instructions are executed according to the judgment result.
	Example:
	IF (iVar0 = 4) THEN
	iVar1 : = 5;
	ELSIF (iVar0 = 7) THEN
	iVar1 : = 10;
	ELSE
	iVar1 : = 15;
	END_IF;

FOR instruction	Subordinate instructions are executed repeatedly the specified number of times.
	Example:
	FOR iVar0 : = 1 TO 10 BY 1 DO
	iVar1 : = iVar1 + 1;
	END_FOR;

WHILE instruction	Conditions are judged and subordinate instructions are executed repeatedly as long as the conditions are satisfied.
	Example:
	WHILE (iVar0 <> 0) DO
	iVar1 : = iVar1 * 2;
	END_WHILE;

CASE instruction	Conditions are judged and subordinate instructions are executed according to the judgment result.
	Example:
	CASE iVar0 OF
	1 : iVar1 : = iVar1 / 2;
	2 : iVar1 : = iVar1 / 4;
	ELSE
	iVar1 : = iVar1 / 8;
	END_CASE;

REPEAT instruction	Conditions are judged and subordinate instructions are executed repeatedly as long as the conditions are satisfied.
	Example:
	REPEAT
	iVar0 : = iVar0 + 1;
	UNTIL
	iVar0 = 100
	END_REPEAT;

EXIT instruction	The EXIT instruction is used to terminate a loop within the FOR, WHILE, or REPEAT instruction.
RETURN instruction	The RETURN instruction is used to terminate a program organization unit (POU). Instructions within POUs following the RETURN instruction will not be executed.
JMP instruction	The JMP instruction is used to unconditionally move control to the line indicated by the JMP label. Example:

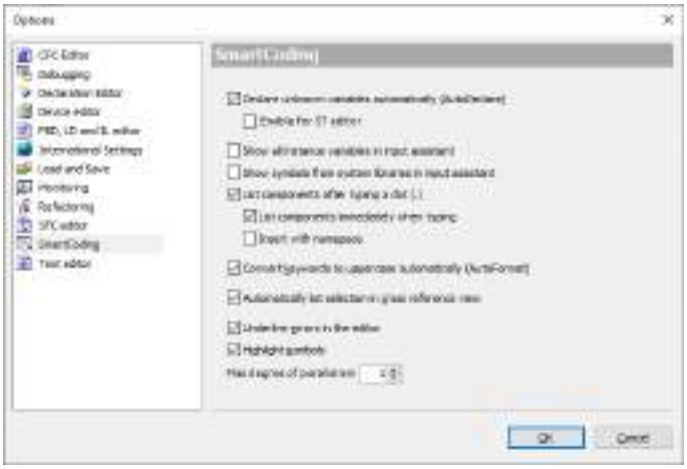
Item	Example

	iVar0 : = 0;
	Label1 : iVar0 : = iVar0 + 1;
	IF (iVar1 = 5) THEN
	JMP Label1;
	END_IF;

CONTINUE instruction	The CONTINUE instruction is used to move control to the beginning of the loop within the FOR, WHILE, or REPEAT instruction.

Info.

- Entered keywords are converted to uppercase letters automatically (AutoFormat). To disable this function, clear the "Convert keywords to uppercase automatically (AutoFormat)" check box in the Options window.



7.2.2 Commenting out Code in ST Program

In ST programs, code can be commented out. Program code that is commented out cannot be executed.

Comment type	Description
Single line	Program code from // to the end of the line is treated as a comment. Example: bVar1 := 2; // Single-line comment
Multiple lines	Program code from (* to *) is treated as a comment. (* *) can also be inserted into another comment enclosed between (* and *). Example: (* Multiple-line comment 1 Multiple-line comment 2 *)

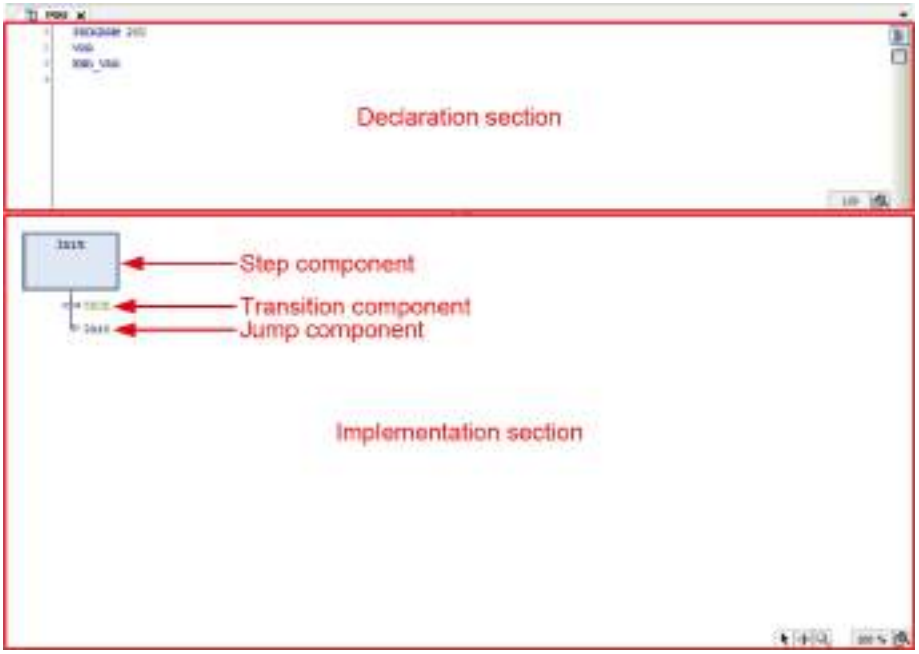
Example: Single-line and multiple-line comments

```
1   X1  := 2;   //Single line comment
2   (*
3       Multi-line comment 1
4       Multi-line comment 2
5   *)
6
```

7.3 Programming in Sequential Function Chart (SFC)

This section explains how to create programs (SFC programs) in Sequential Function Chart compliant with IEC 61131-3, the international standard for PLC programming languages.

- To create SFC programs, POU objects for SFC programs are required. Set the object setup language to Sequential Function Chart (SFC).
- The SFC program editor is divided into the declaration section and implementation section. Three elements are originally coded in the implementation section.

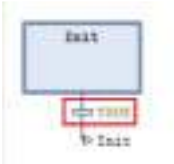


7.3.1 Inserting Elements from Menu

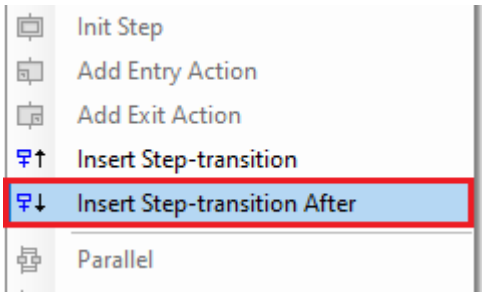
For example, step elements and transition elements can be inserted from the menu, as below.

Procedure

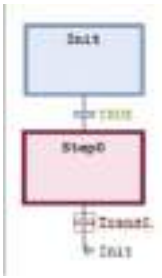
1. Select the TRUE transition element in the implementation section.
   The selected transition element will turn red.



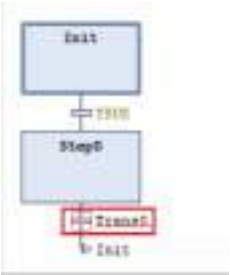
2. Click the right mouse button and select "Insert Step-transition After" from the context-sensitive menu that is displayed.



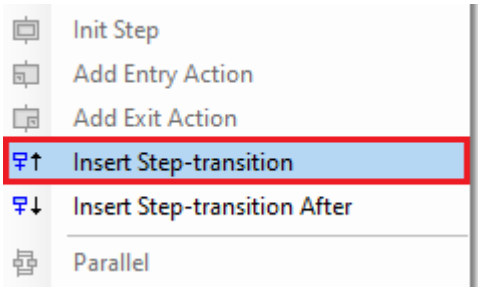
"Step0" step element and "Trans0" transition element will be inserted below the TRUE element.



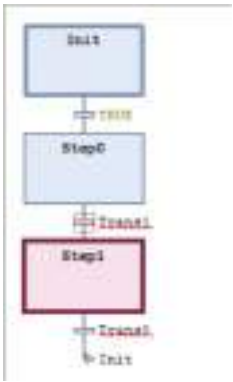
3. Select the "Trans0" element.



4. Click the right mouse button and select "Insert Step-transition" from the context-sensitive menu that is displayed.



"Step1" step element and "Trans1" transition element will be inserted above the "Trans0" transition element.



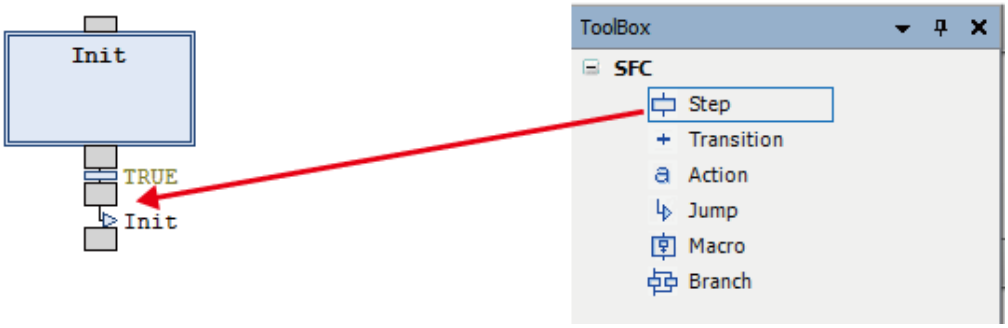
7.3.2 Inserting Elements from Toolbox

Each element can be inserted from Toolbox.

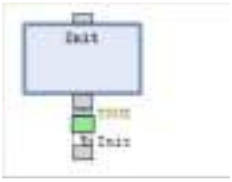
This section explains the procedure for inserting elements from Toolbox, using a step element as an example.

Procedure

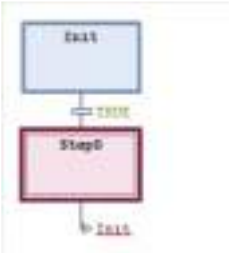
1. In Toolbox, select "Step" and then drag the step element to the position where you want to insert it.



When you drag the step element to the insertion position, the step element is transformed into .



2. Stop dragging the mouse.
   The step element will be inserted.

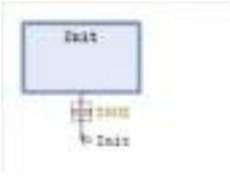


7.3.3 Inserting Elements from Toolbar

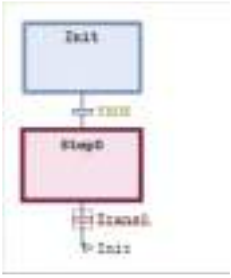
For example, step elements and transition elements can be inserted from the toolbar, as below.

Procedure

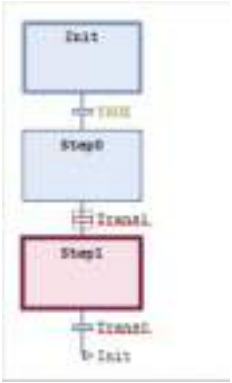
1. Select the TRUE transition element in the implementation section.
   The selected transition element will turn red.



2. Click the icon ("Insert Step-transition After") on the toolbar.
   "Step0" step element and "Trans0" transition element will be inserted below the TRUE element.



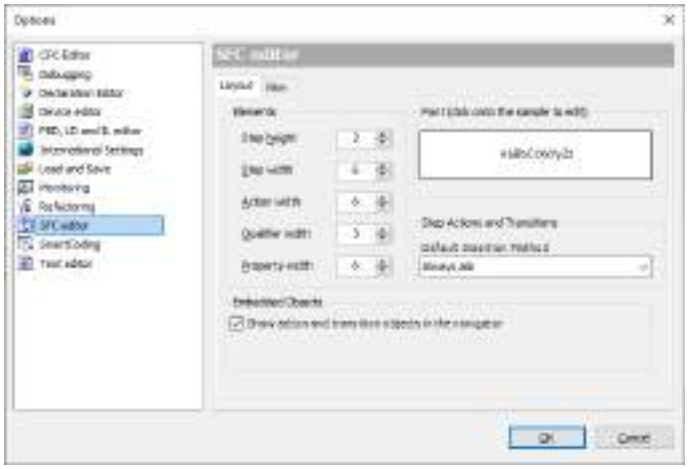
3. Select the "Trans0" transition element and click the icon ("Insert Step-transition") on the toolbar.
   "Step1" step element and "Trans1" transition element will be inserted above the "Trans0" transition element.



7.3.4 Setting up the SFC Editor

For SFC editor elements, you can change step specifications, fonts, and other settings.
From the menu bar, select **Tools>Options** to open the "Options" dialog box.
In the "Options" dialog box, select the "SFC editor" category and change the settings.

Layout

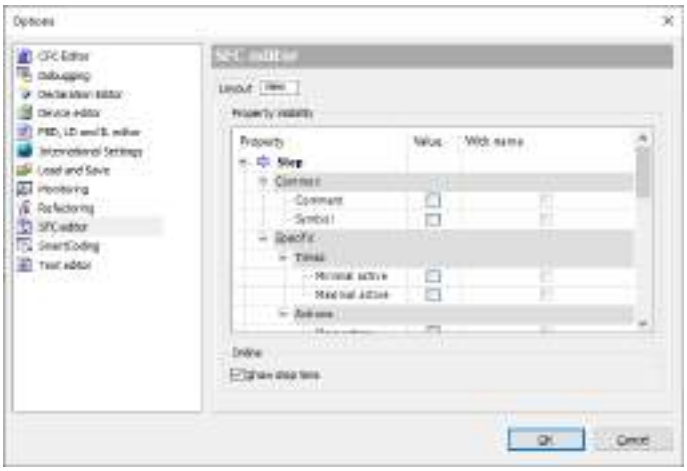


Item name		Default value	Function
Elements	Step height	2	Sets the height of a step. Setting range: 1 to 100 matrix units(Note 1)
	Step width	6	Sets the width of a step. Setting range: 2 to 100 matrix units(Note 1)
	Action width	6	Sets the width of an action. Setting range: 2 to 100 matrix units(Note 1)
	Qualifier width	3	Sets the width of a qualifier. Setting range: 2 to 100 matrix units(Note 1)
	Property width	6	Sets the width of a property.

Item name		Default value	Function
			Setting range: 2 to 100 matrix units(Note 1)
Font		-	Sets a font to be displayed on the SFC editor.
Step Actions and Transitions	Default insertion method	Always check	Sets the operation to be performed when an action is added to a step. **Copy reference:** When a step is copied, a link to the step action is also copied. The step that is copied invokes the same action. **Copy implementation:** The step action of a step that is copied is embedded. A new action object is copied to a new step. **Always ask:** Which of the above operations is to be executed is checked each time an action is initially added to a step.
Embedded Objects	Show actions and transition objects in navigator pane	Selected	This check box is used to specify whether to display the actions embedded in steps in the navigator pane when an action is added to a step by "Copy implementation". **Selected:** An action that is embedded in a step by "Copy implementation" is displayed in the Device view or POU view. **Cleared:** An action that is embedded in a step by "Copy implementation" is not displayed in the Device view or POU view.

(Note 1) "1 matrix unit" is equal to the font size specified in Font in **Options**>**Text editor**>**Text area** tab.

View



Item name		Default value	Settings
Property Visibility		Cleared	Specifies whether to display property values and names beside steps, etc. Selected: Displays property values and names Canceled: Does not display property values or names
Online	Display step time	Selected	Specifies whether to display step active time beside the step in online mode Selected: Displays active time Cleared: Does not display active time

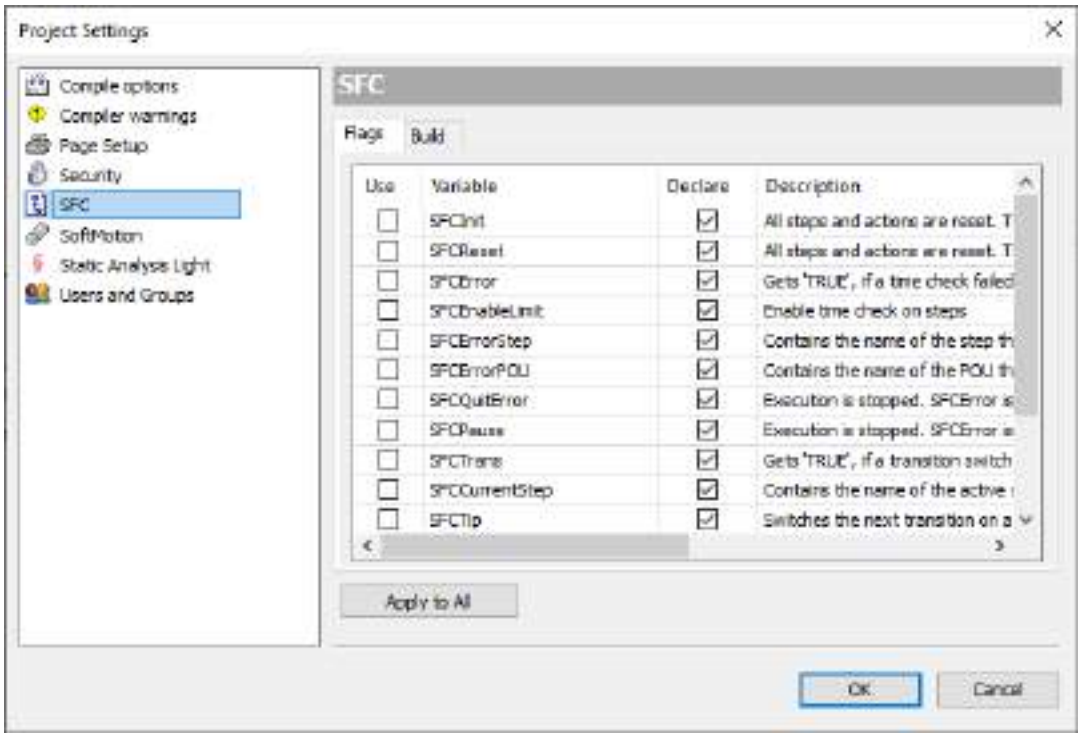
7.3.5 Setting SFC Program Execution Conditions

■ Setting SFC program execution conditions

For SFC programs within a project, you can specify whether to generate code for variables used to check processing or for active transitions during build.

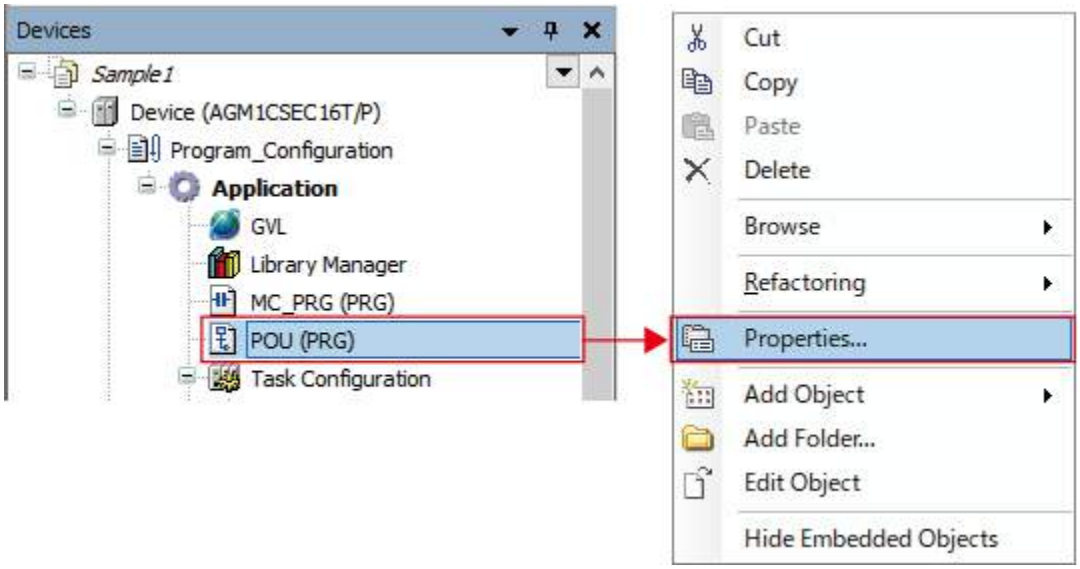
From the menu bar, select **Project**>**Project Settings**. In the Project Settings dialog box, select the "SFC" category.

The settings will be applied to all SFC objects.

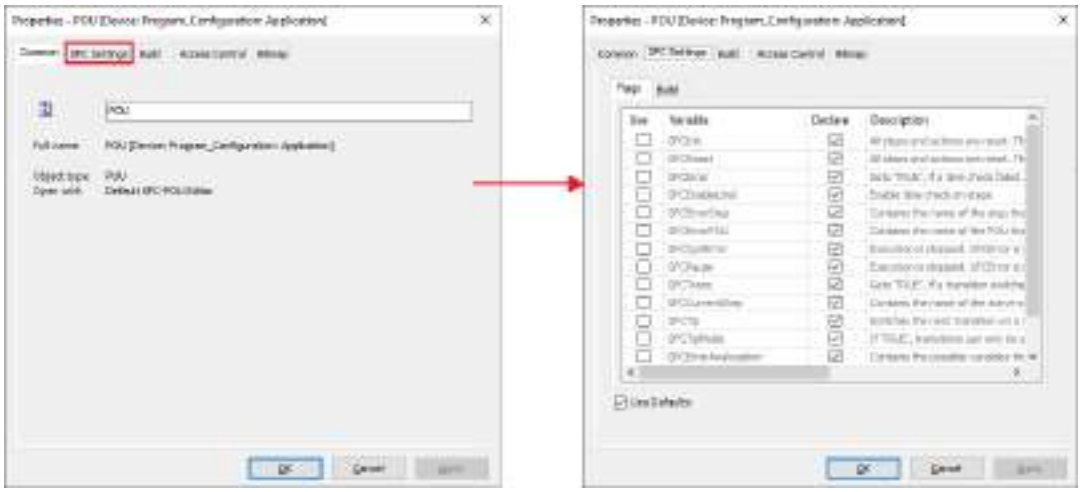


The "Properties" dialog box for SFC objects can be enabled for only particular SFC objects.

Right-click the POU object in the navigator pane and then select "Properties" from the context-sensitive menu that is displayed.



In the "Properties" dialog box, select the "SFC Settings" tab and clear the "Use default" check box.

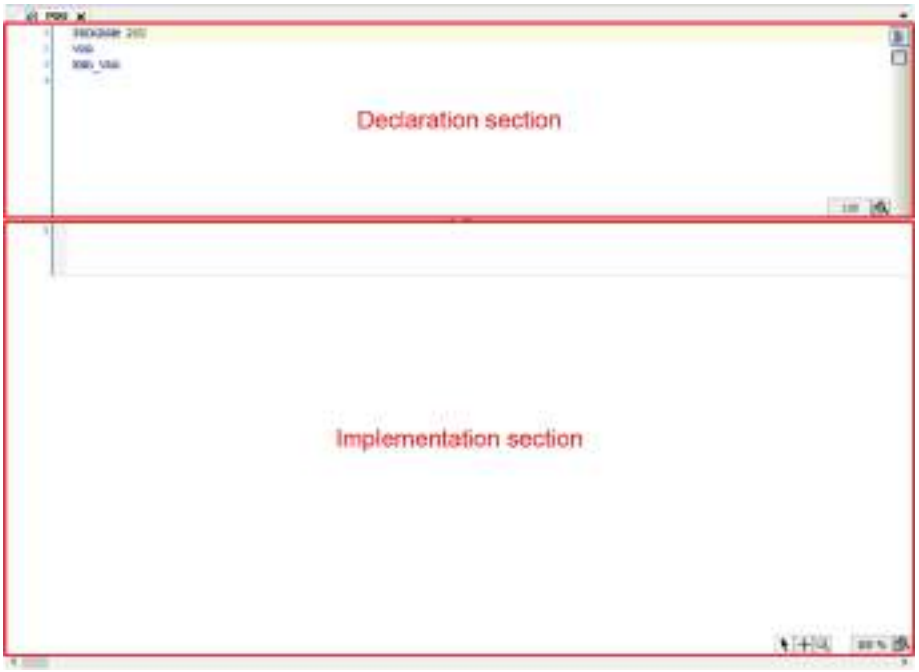


Clearing the check box allows settings to be changed and the Properties dialog box to be enabled for only particular SFC objects.

7.4 Programming in Function Block Diagram (FBD)

This section explains how to create programs (FBD programs) in Function Block Diagram compliant with IEC 61131-3, the international standard for PLC programming languages.

- To create FBD programs, POU objects for FBD programs are required. Set the object setup language to Function Block Diagram (FBD).
- The editor for FBD programs is a window like the one shown below. It consists of the declaration section used to declare variables and the implementation section used to enter program data. Elements can be arranged in the implementation section by selecting them from Toolbox and then dragging and dropping them in the implementation section.



7.4.1 Entering Function Blocks

This section explains the procedure for entering function blocks, using an FBD program consisting of the following variables and function block TON as an example.

12 Procedure

1. In Toolbox, select **Function block>TON** and drag the mouse until the mouse pointer reaches the position where "Start here" is displayed in the implementation section.
   "Start here" will be displayed in the implementation section.

When you drag the mouse until the position of "Start here" is reached, the display of "Start here" turns green.

When you stop dragging the mouse at the position of "Start here", function block TON will be placed in the network (circuit).

2. Select TON and enter variable TON_0, and then press the <Enter> key.
   The "Auto Declare" dialog box will be displayed.

3. Click the [OK] button.
Instance variable TON_0 for function block TON will be declared in the declaration section.

4. Select "???" beside IN of function block TON and enter variable bVar0, and then press the <Enter> key.
The "Auto Declare" dialog box will be displayed.

5. Click the [OK] button.
Variable bVar0 will be declared in the declaration section.

6. In function block TON, enter "T#5s" for input PT ("IN PT") and variable "CurrentTime" for output ET ("O ET").

When "CurrentTime" is entered, the "Auto Declare" dialog box is displayed.

When making a declaration, check that the type is "Time".

7. In Toolbox, select **General**>**Assignment** and drag the mouse until a diamond-shaped icon appears on the right side of output "Q" in function block TON.

When you stop dragging the mouse, "???" appears on the right side of output "Q".

8. Select "???" on the right side of output "Q" and enter variable bVar1, and then press the <Enter> key.

The "Auto Declare" dialog box will be displayed.

Declare variable bVar1.

Info.

- The FBD program can be converted and displayed as an LD program.
  From the menu bar, select **FBD/LD/IL>View** and select a post-conversion programming language.

7.4.2 Inserting and Commenting out a Network (Circuit)

The procedure for inserting a new network (circuit) is the same as for LD programs.

Refer to "7.1.3 Inserting a Network (Circuit)".

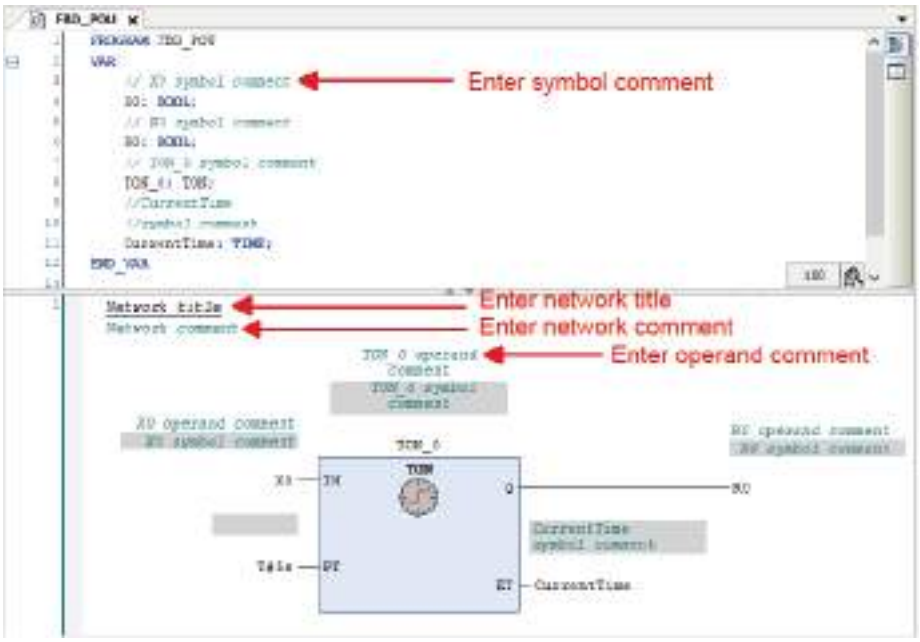
Networks (circuits) can be commented out. The procedure for commenting out a network (circuit) is the same as for LD programs.

Refer to "7.1.6 Commenting out a Network (Circuit)".

7.4.3 Input of Title and Comment (FBD)

In FBD programs, titles and comments can be entered in the same way as for LD programs.

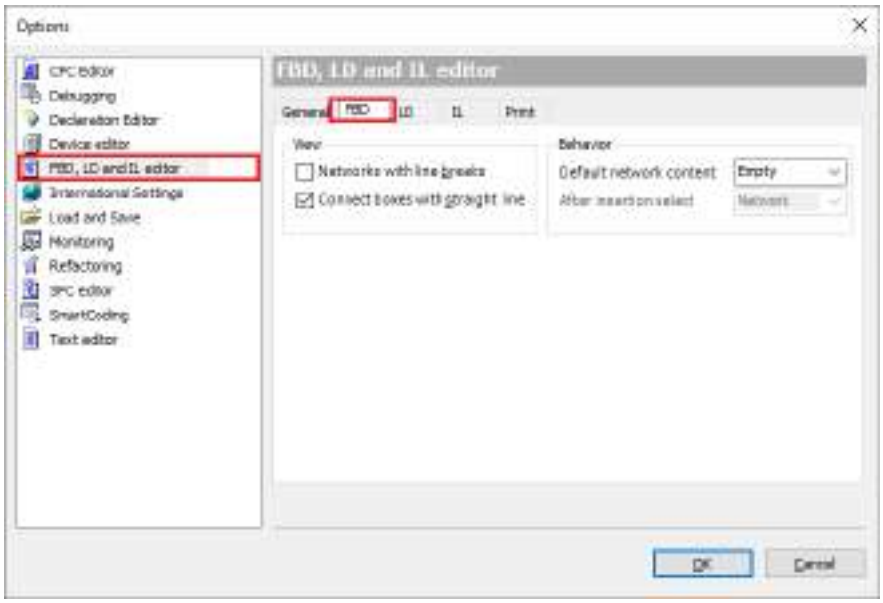
Refer to "7.1.5 Input of Title and Comment (LD)".



7.4.4 Settings in FBD Program

For FBD programs, the following settings can be configured.

Open the Options window (by selecting **Tools**>**Options**), select the "FBD, LD and IL editors" category and specify settings in the "FBD" tag window.



Type	Item	Description
View	Networks with line breaks	Selects whether to arrange elements by inserting line breaks automatically so that the display fits in the lateral width of the main pane.
	Connect boxes with straight line	Selects whether to fix the shortest length of a line connecting boxes.
Behavior	Default network content	Selects whether to arrange elements and variables automatically or arrange nothing when a network (circuit) is inserted.
	After insertion select	Selects whether to select a circuit or element after a network (circuit) is inserted.

7.5 Programming in Instruction List (IL)

This section explains how to create programs (IL programs) in Instruction List compliant with IEC 61131-3, the international standard for PLC programming languages.

- IL programs are text-based programs that consist of instructions and operands.
- To create IL programs, POU objects for IL programs are required. Set the object setup language to Instruction List (IL).
- The editor for IL programs is a window like the one shown below. It consists of the declaration section used to declare variables and the implementation section used to enter program data.

- To use POU objects for IL programs, the "Enable IL" check box must be selected as below.
- From the menu bar, select **Tools**>**Options**>**FBD, LD and IL editors**> and select the "Enable IL" check box in the "IL" tab.

7.5.1 Entering Instructions and Operands

This section explains how to create an IL program that consists of the following instructions.

```
IL_POU ×
1  PROGRAM IL_POU
2  VAR
3      iVar: INT := 5;
4      iResult: INT := 0;
5      bVar1: BOOL := TRUE;
6      bVar2: BOOL := TRUE;
7      bVar3: BOOL;
8  END_VAR
9

1      LD      iVar
       ADD     3
       ST      iResult
       LD      bVar1
       JMPC    mark1
2      mark1:
       LD      bVar2
       S       bVar3
```

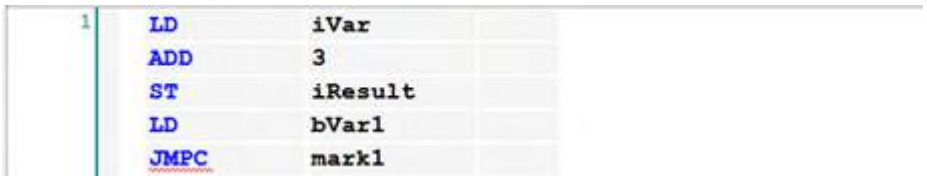
1 2 Procedure

1. Select the cell in the first row and the first column, and enter instruction "LD".
   The LD instruction will be entered.

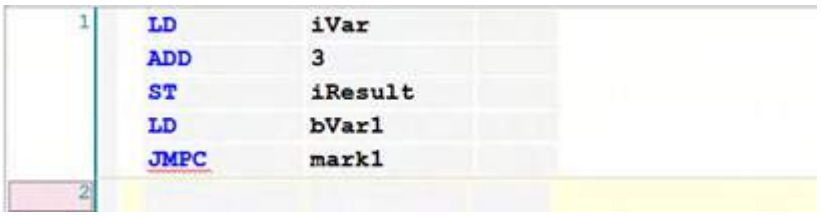
2. Press the <Tab> key to move to another cell and enter operand "iVar".
   The operand will be entered and the "Auto Declare" dialog box will be displayed.
   In the "Auto Declare" dialog box, declare a variable.

3. Press the <Ctrl> key + <Enter> key simultaneously.
   The cursor will move to the next row.

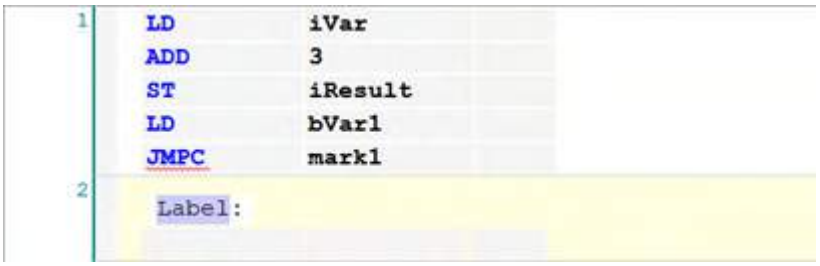
   In the second and subsequent rows, enter instructions and operands in the same way as above.



4. In Toolbox, select **General**>**Network** and drag the mouse until displayed in the network (circuit) in the main pane is reached.
   A new network (circuit) will be inserted.



5. From the menu bar, select **FBD/LD/IL**>**Insert label**.
   "Label" will be inserted.



6. Enter label name "mark1" and add instructions and operands to the network (circuit) that has been inserted.

```
1   LD      iVar
    ADD     3
    ST      iResult
    LD      bVar1
    JMPC    mark1
2
    mark1:
    LD      bVar2
    S       bVar3
```

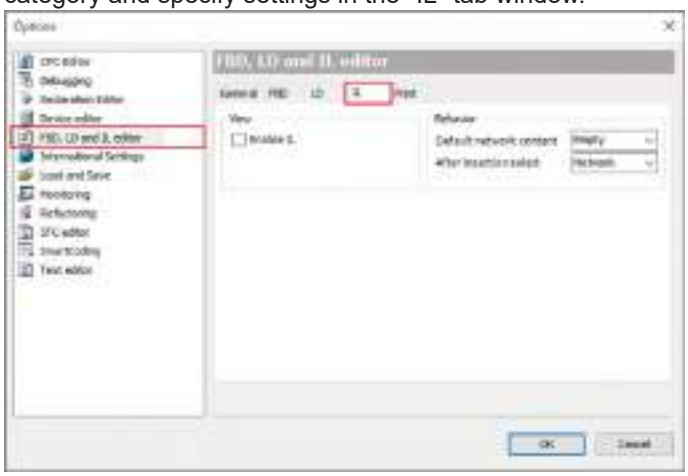
Info.

- To create IL programs, the "Enable IL" check box in the Options dialog box must be selected. Refer to "7.5.2 Settings in IL Program".
- To delete a row, select the row and press the <Ctrl > key + <Delete> key simultaneously.
- Networks (circuits) can be commented out. The procedure for commenting out a network (circuit) is the same as for LD programs.
  Refer to "7.1.6 Commenting out a Network (Circuit)".
- The IL program can be converted and displayed as an LD or FBD program.
  From the menu bar, select **FBD/LD/IL**>**View** and select a post-conversion programming language.

7.5.2 Settings in IL Program

For IL programs, the following settings can be configured.

Open the Options window (by selecting **Tools**>**Options**), select the "FBD, LD and IL editors" category and specify settings in the "IL" tab window.



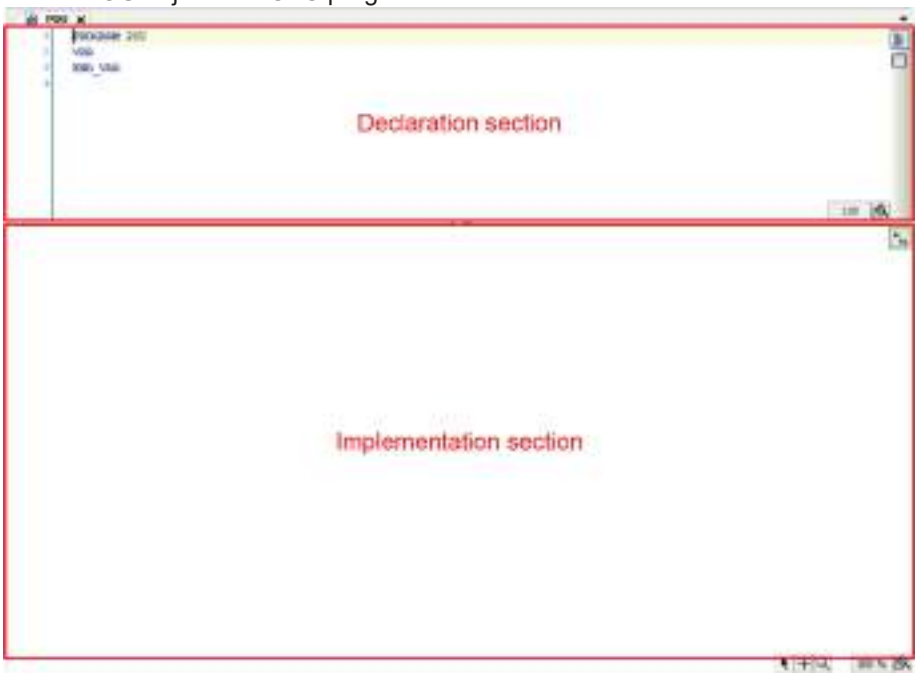
Type	Item	Description
View	Enable IL	Enables the use of IL programming language. If this check box is cleared, IL will not be displayed in the list of programming languages for setting objects when a new project is created.

Type	Item	Description
Behavior	Default network content	Selects whether to arrange elements and variables automatically or arrange nothing when a network (circuit) is inserted.
	After insertion select	Selects whether to select a circuit or element after a network (circuit) is inserted.

7.6 Programming in Continuous Function Chart (CFC)

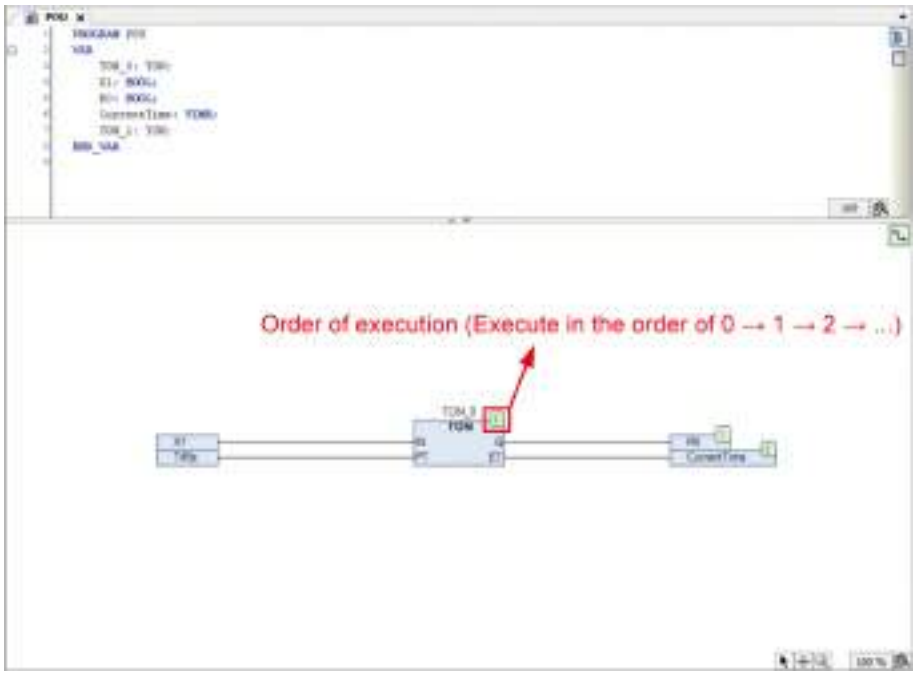
Continuous Function Chart is a graphical programming language that allows programs to be created by arranging elements within the CFC editor. Elements can be freely arranged within the editor and the order of execution is determined according to the list of elements inserted.

- CFC programs are classified into standard CFC programs and page-oriented CFC programs. Page-oriented CFC programs allow page-based switchover.
- To create CFC programs, POU objects for CFC programs are required. Use "Add Object" to add POU objects for CFC programs.



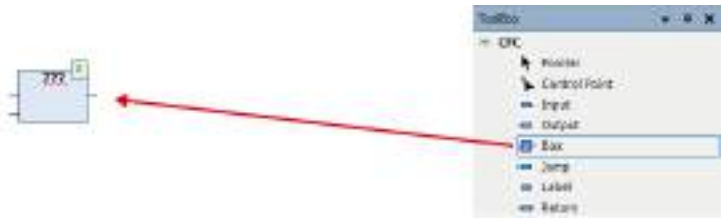
7.6.1 Inserting and Connecting Elements

This section explains how to create a CFC program that consists of the variables and function block TON shown below.

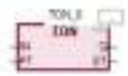


Procedure

1. In Toolbox, select **CFC**>**Box** and drag the box element and drop it in the implementation section.
   The box element will be placed in the implementation section.



2. Enter an instance name in "???".
   The box element will be transformed into function block TON and an instance name can be entered for function block TON.

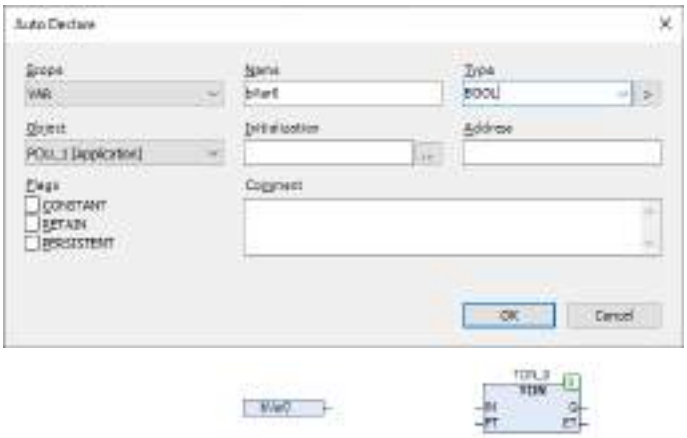


3. Select TON and enter variable TON_0, and then press the <Enter> key.
   The "Auto Declare" dialog box will be displayed.

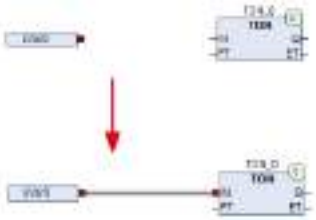
4. Click the [OK] button.
Variable TON_0 for function block TON will be declared in the declaration section.

5. In Toolbox, select **CFC**>**Input** and drag the input element and drop it in the implementation section.
The input element will be placed in the implementation section.

6. Enter variable bVar0 in input element "???" and press the "Enter" key.
The "Auto Declare" dialog box will be displayed. Declare Boolean variable bVar0.



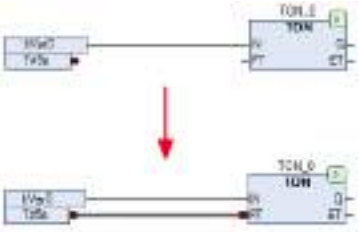
7. Select the pin on input element "variable bVar0" and drag it to "IN" on TON.
Input element "variable bVar0" and "IN" on TON will be connected with a line.



8. In Toolbox, select **CFC**>**Input** and drag the input element and drop it in the implementation section.
The input element will be placed in the implementation section.

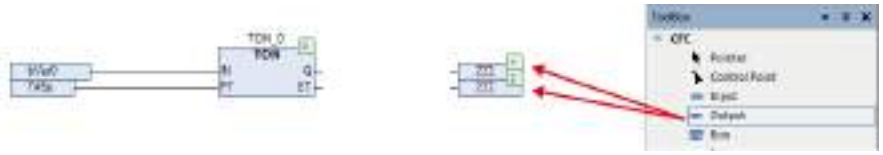
9. Enter "T#5s" in "???" and press the "Enter" key.

10. Select the pin on input element "T#5s" and connect it to "PT" on TON.
Input element "T#5S" and "PT" on TON will be connected with a line.



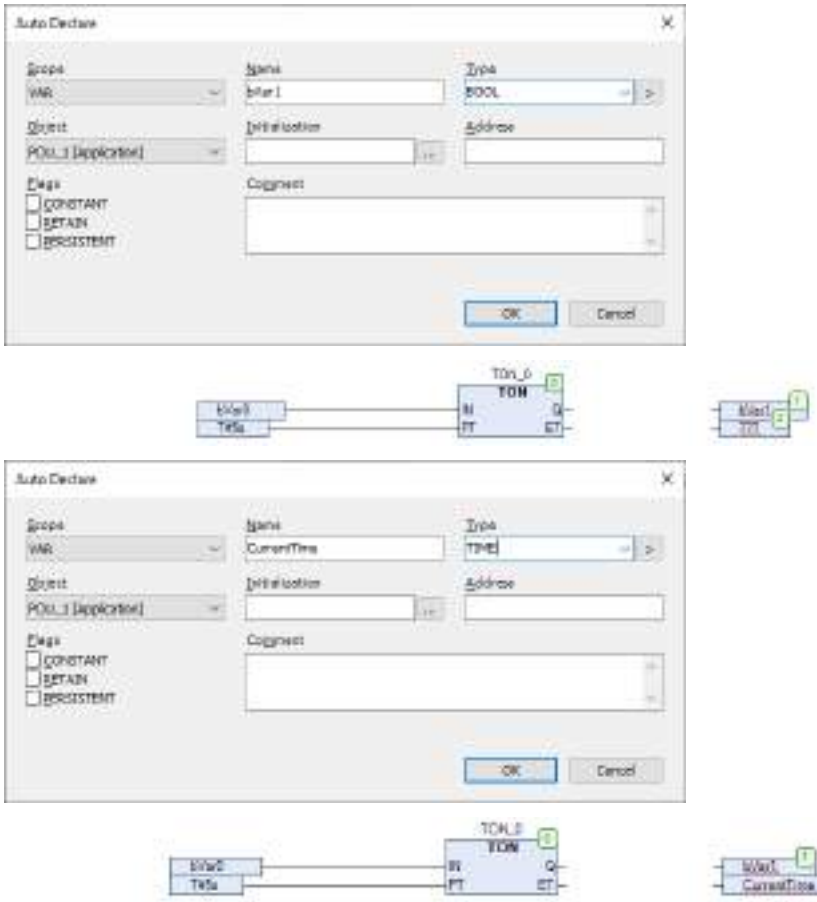
11. In Toolbox, select **CFC**>**Output** and drag the output element and drop it in the implementation section.

The output element will be placed in the implementation section.

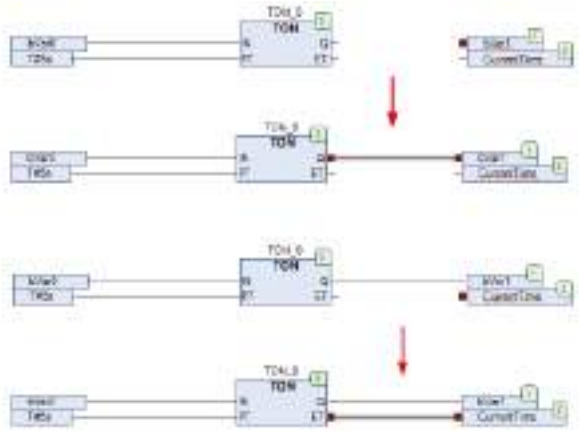


12. Enter variable bVar1 and "CurrentTime" in output element "???" and press the "Enter" key.

The "Auto Declare" dialog box will be displayed. Declare a Boolean variable for variable bVar1 and a Time variable for variable "CurrentTime".



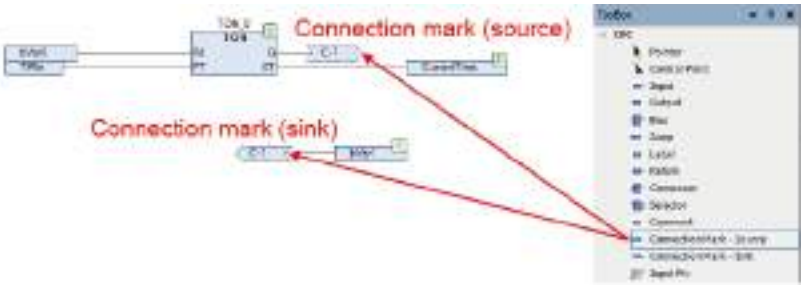
13. Connect output element "variable bVar1" and "Q" on TON and connect output element "CurrentTime" and "ET" on TON.



7.6.2 Connection Mark

Connection marks can be used to separate connection lines.

In Toolbox, select **CFC>Connection Mark - Source** and **CFC>Connection Mark - Sink**, arrange them in the implementation section, and enter the same name for them.



Info.

- You can also separate a connection line as a connection mark by selecting **CFC>Connection Mark** from Toolbox or clicking the icon on the toolbar, with the connection line selected . Conversely, you can restore the connection mark to a connection line by selecting **CFC>Connection Mark** from Toolbox or clicking the icon on the toolbar, with the connection mark selected.

7.7 Program Creation Support Functions

This section explains the program creation support functions used to create programs.

7.7.1 Bookmark

The bookmark function allows the cursor to move to bookmarked locations.
The bookmark function can be used in all programs other than SFC programs.
This section explains the procedures for setting bookmarks and moving the cursor between bookmarks, using an LD program as an example.

Procedure

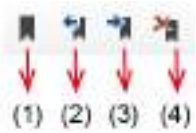
1. Select the network (circuit) where you want to set a bookmark. From the menu bar, select **Edit**>**Bookmarks**>**Toggle Bookmark**, or click the <Ctrl> key + <F12> key simultaneously.

   A bookmark ( ) will be set in the selected network (circuit).
   Performing the above operation again clears (deletes) the set bookmark.

2. To move to the next bookmark, press the <F12> key. To move to the previous bookmark, press the <Shift> key + <F12> key simultaneously.

Info.

- To clear (delete) all the set bookmarks, from the menu bar, select **Edit**>**Bookmarks**>**Clear All Bookmarks**.
- You can also perform bookmark operations by clicking appropriate icons on the toolbar.



No.	Item
(1)	Toggle Bookmark
(2)	Previous Bookmark
(3)	Next Bookmark
(4)	Clear All Bookmarks

7.7.2 Call Tree View

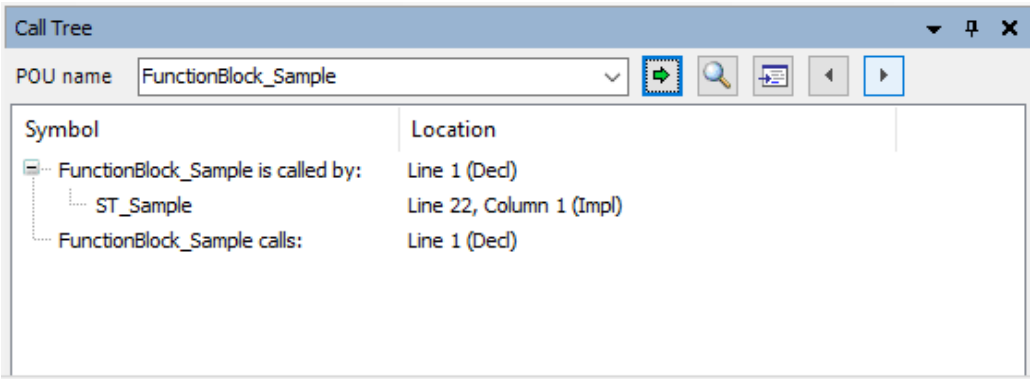
Opening the Call Tree view enables the user to search the callers and callees of blocks such as functions, function blocks, or POU.

Procedure

1. From the menu bar, select **View**>**Call Tree**.
   The Call Tree view will be displayed.

2. Enter a block to be searched in the POU Name field and press the <Enter> key.
   The caller and callee of the block will be displayed in tree structure.
   Double-clicking in any search result line displays the corresponding window in the main pane.

   Example: Searching the caller and callee of FunctionBlock_Sample



Info.

- Pressing the <F4> key moves the cursor to the next search result line. Pressing the <Shift> key + <F4> key simultaneously moves the cursor to the previous search result line.
- Position the cursor on a block in the implementation section. From the menu bar, select **Edit**>**Browse**>**Browse Call Tree**. The Call Tree view will be displayed with a search conducted for the block at the cursor position.

7.7.3 Cross reference List View

Opening the Cross reference List view allows the user to search the locations of variables and other elements used within the entire project.

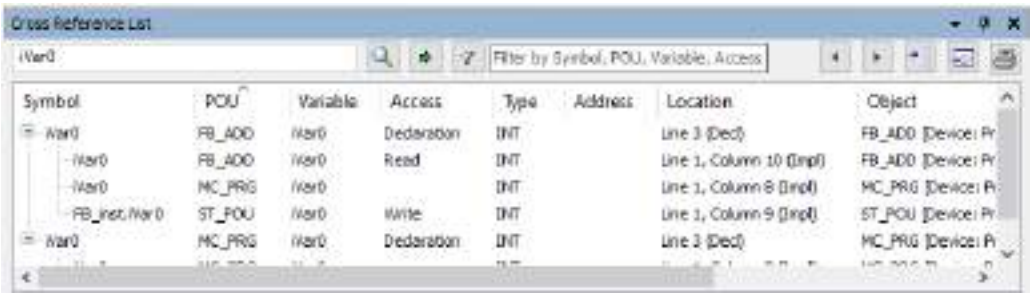
Procedure

1. From the menu bar, select **View**>**Cross Reference List**.

The Cross reference List view will be displayed.

2. Enter a variable name (or another element name) to be searched in the search field and press the <Enter> key.
   The locations of the variable (or element) used will be displayed in list form.
   Double-clicking in any search result line displays the corresponding section in the main pane.

Example: When variable iVar0 is entered



Info.

- Pressing the <F4> key moves the cursor to the next search result line. Pressing the <Shift> key + <F4> key simultaneously moves the cursor to the previous search result line.
- For searches, you can use an asterisk (*) that represents any character string or a question mark (?) that represents a single character.
- Position the cursor on a variable in the implementation section. From the menu bar, select **Edit**>**Browse**>**Browse Cross Reference**. The Cross-reference view will be displayed with a search conducted for the variable at the cursor position.

7.7.4 Function Block Guidance

The Function Block Guidance allows the user to enter motion function blocks into a program.
The Function Block Guidance can be used in LD programs, ST programs, FBD programs, and CFC programs.
The following procedure is explained, using an example in which "Power" is searched and function block "MC_Power" is inserted into an LD program.

Procedure

1. Double-click the [MC_PRG(PRG)] object in the navigator pane.

The "MC_PRG" window will be displayed.

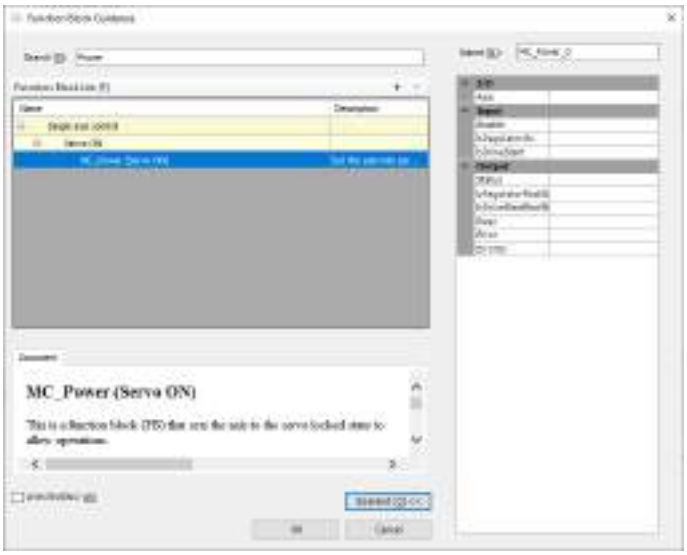
2. Select the network in the implementation section and, from the menu bar, select **Edit**>**Function Block Guidance**.

The "Function Block Guidance" dialog box will be displayed.

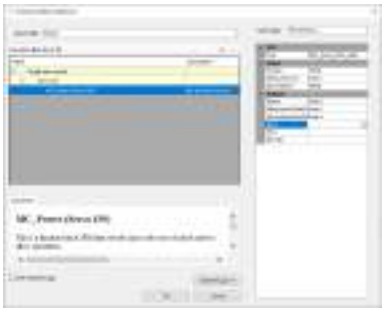
3. Enter a character string in the Search field.
   Function blocks related to the entered character string will be displayed in the Function Block List table.

4. Select a function block that you want to insert into the program.
   A description of the selected instruction will be displayed in the Document tab pane.

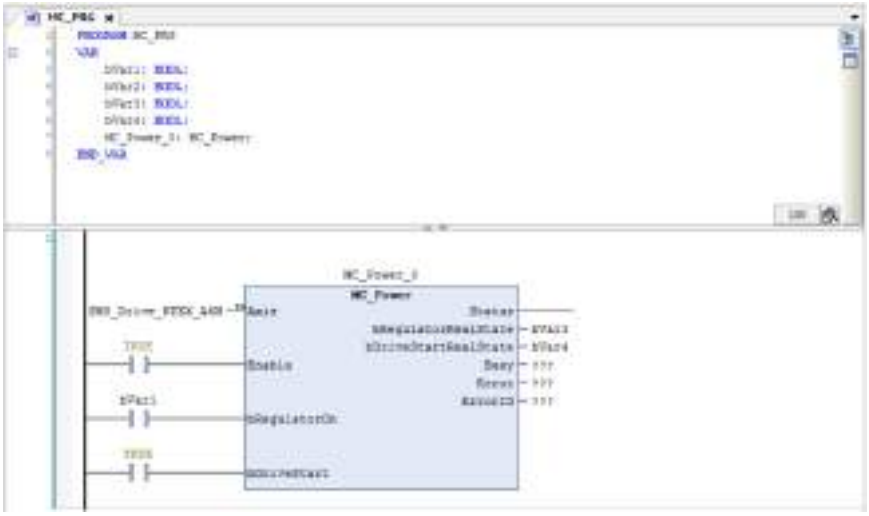
5. Click the [Operand] button.
The instance and operand input fields will be displayed.



6. Enter an instance name in the Name field and values in each operand field.
If the operand for which a value has been entered is a variable that has not been declared, the "Auto Declare" dialog box will be displayed, so that the variable can be declared.



7. Click the [OK] button.
   If an instance name has not been declared, the "Auto Declare" dialog box will be displayed, so that an instance can be declared.
   The function block will be inserted into the program.



Info.

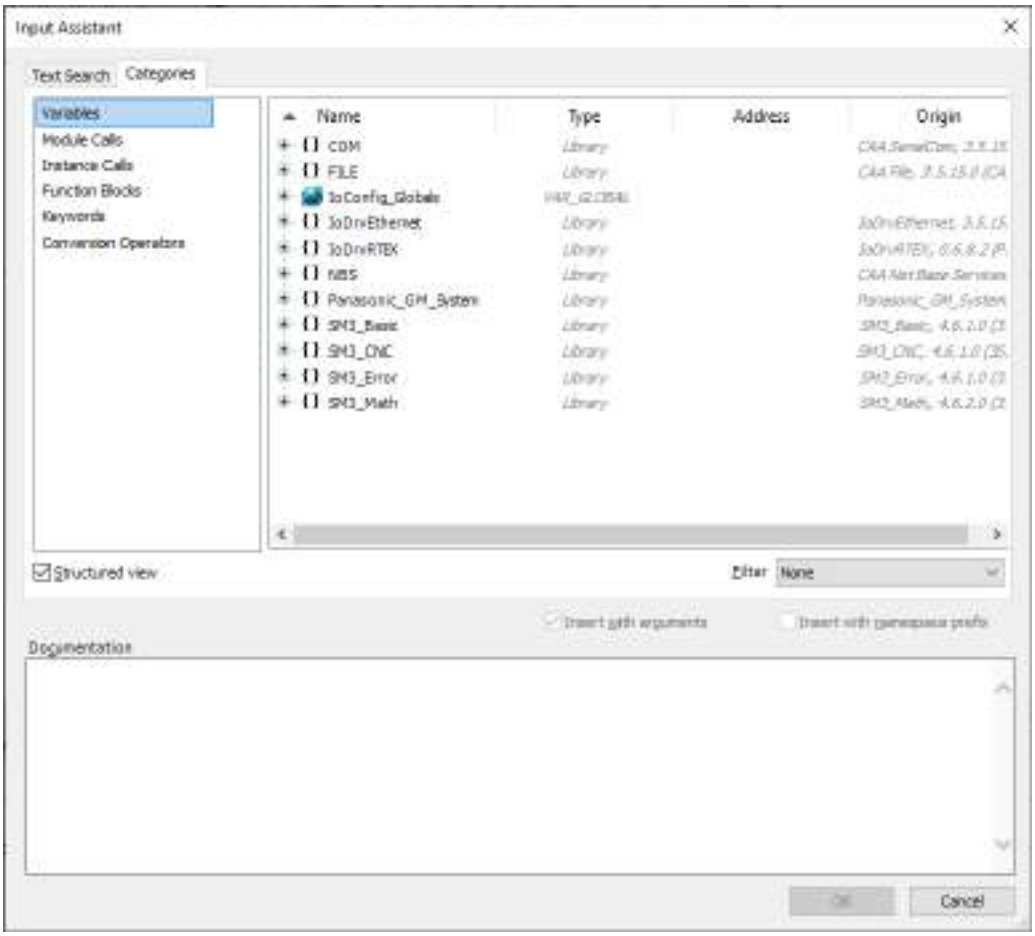
- To insert a function block into a CFC program, insert a box first and, with the box selected, start the Function Block Guidance.
- If the "With EN / ENO" check box is selected, a function block with EN input and ENO output will be inserted. When the value of EN input is TRUE, the function block is executed. Similarly, when the value of EN input is FALSE, the function block is not executed. The same value as EN input is output to ENO output.
- The Function Block Guidance can also be started using the following operations:
  - Click the "Function Block Guidance" icon on the toolbar.
  - Press the <Alt> key + <F2> key simultaneously.
  - Select the network in the implementation section, and then right-click and select "Function Block Guidance" from the context-sensitive menu that is displayed.

7.7.5 Input Assistant Function

By using the Input Assistant Function, variables, function blocks, operators, types, and other elements that can be inserted in the cursor position can be selected from categories.

Procedure

1. Move the cursor to the position where you want to insert a desired element and press the <F2> key.
   The "Input Assistant" dialog box will be displayed.
   Select a desired element from the "Categories" tab pane or the "Text Search" tab pane.



2. Click the [OK] button.
   The selected element will be inserted.

Info.

- You can also open the "Input Assistant" dialog box by selecting **Edit>Input Assistant** from the menu bar.

7.7.6 Argument / Variable Input Support (Component List)

This section explains the functions (component list) that support input of arguments and other data for function blocks during program creation.

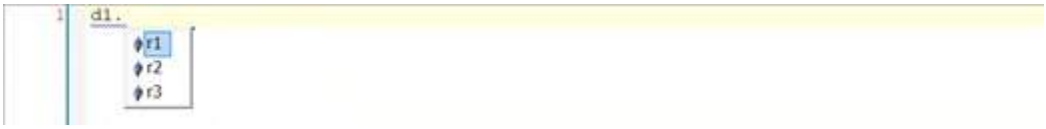
The following three input support functions are available.

■ Displaying candidates for arguments that can be entered

Entering a dot (.) after a name such as a function block name or structure name displays a list of candidates for arguments that can be entered.

To enable this function, in the Options window, select **SmartCoding**>**List components after typing a dot(.)**.

Example: Displaying a list of members of structure variable stVar after structure variable stVar and a dot are entered



- **Displaying candidates for variables or other components starting with the entered character string**

Entering any character string and then pressing the <Ctrl> key + <Space> key simultaneously displays a list of elements that can be inserted.

Example: Entering "TI" and pressing the <Ctrl> key + <Space> key simultaneously selects and displays the positions of variables or other components starting with "TI"

- **Displaying a description of function block or function**

Entering a function block name (or some other name) followed by a left parenthesis displays a description of the function block in a pop-up window.

To open the closed pop-up window again, press the <Ctrl> key + <Shift> key + <Space> key simultaneously.

7.7.7 Global Renaming (Refactoring)

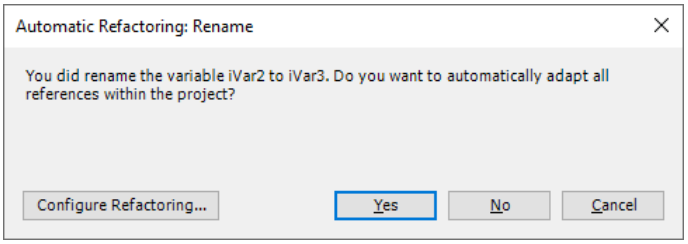
When a POU object name in the navigator pane or a variable name in the declaration section is changed, the sections where the changed name is used are displayed, so that the name can be changed collectively (this function is called "refactoring").

This section explains the procedure for changing the variable name of input variable " iVar2" in function block "FB_ADD" and using the refactoring function to change the variable name in the sections where the variable is called.

1 2 Procedure

1. Change the name of variable "iVar2" in the declaration section of the function block to "iVar3".

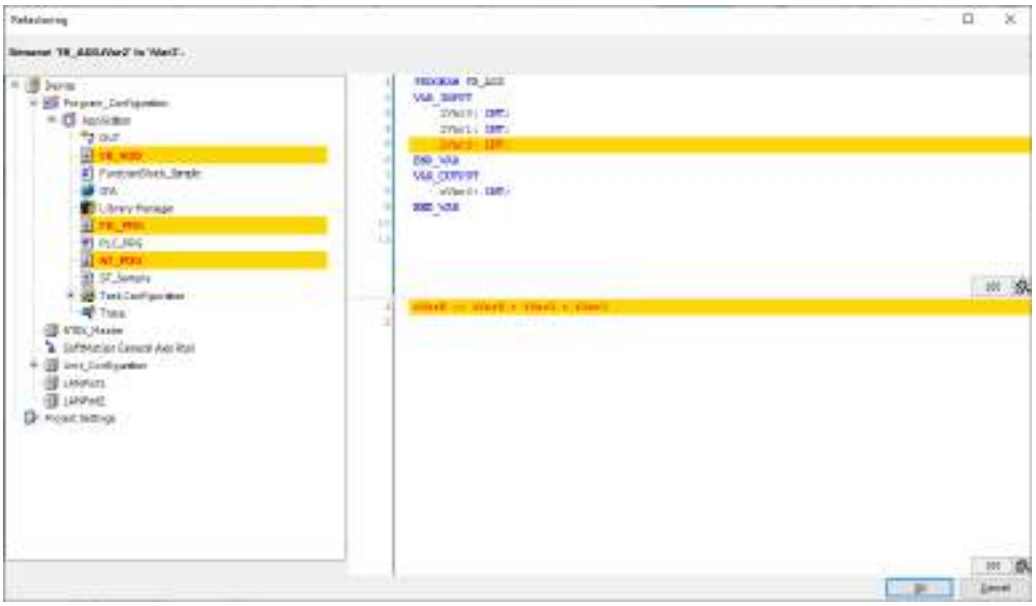
   The "Automatic Refactoring" dialog box will be displayed.



2. Click [Yes].

   The "Refactoring" dialog box will be displayed.

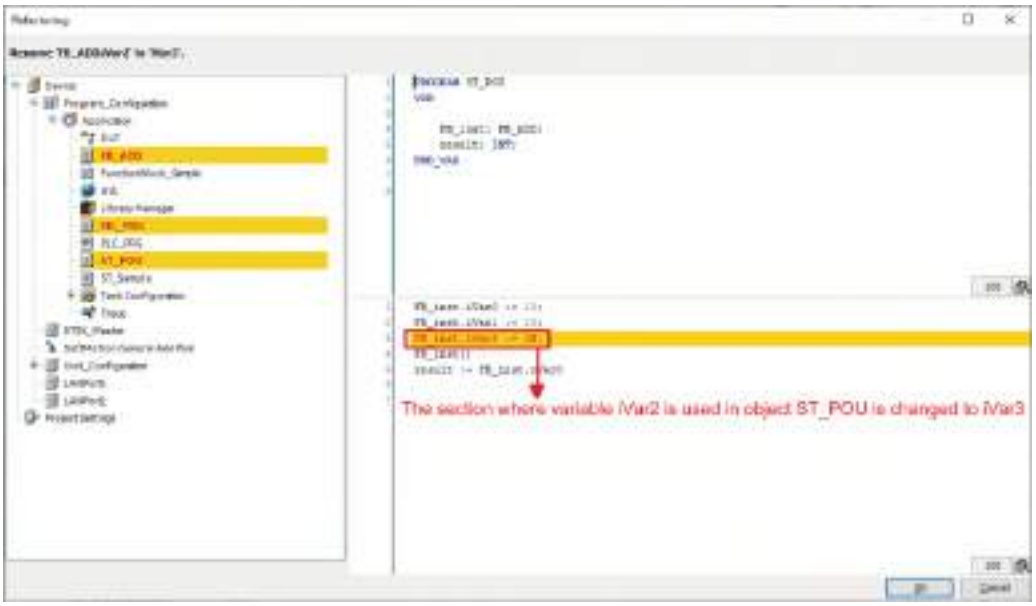
   Each section where the changed variable is used is displayed in red and the background of the section is displayed in yellow.



   By clicking an object in the navigator pane, you can check the change details of the object.

   At this stage, the changes have not been reflected yet.

When ST_POU object is selected:



3. Click the [OK] button.
All changes will be reflected.

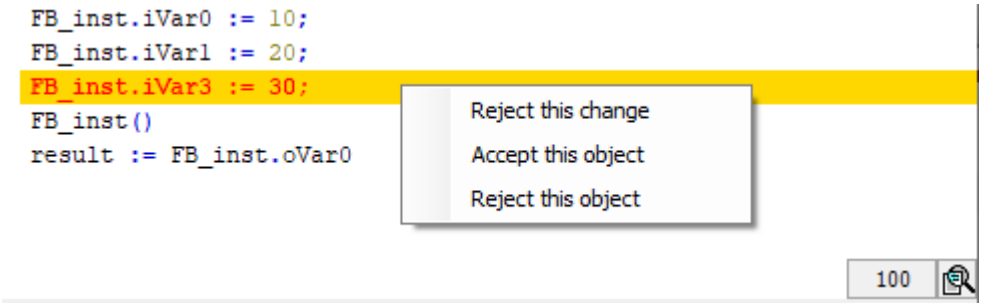
Info.

- When the sections where the changed variable is used are displayed by the refactoring function, you can individually select whether to reflect the change. Right-click in the sections where the changed variable is used and select whether to reflect the change, from the context-sensitive menu that is displayed.

  "Reject this change": Does not reflect the change in the selected section

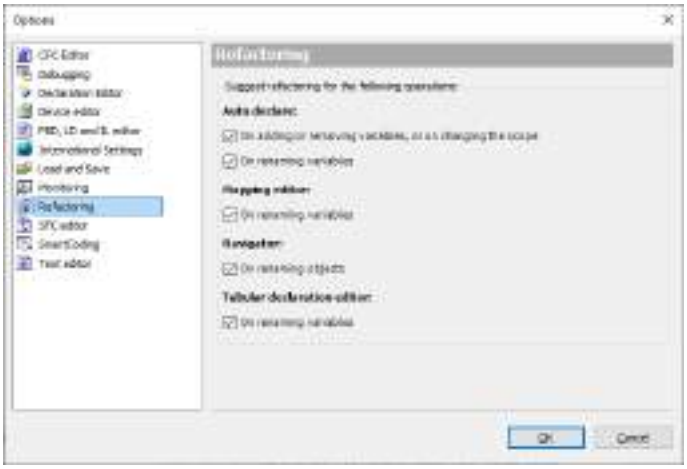
  "Accept this object": Reflects the change in the object

  "Reject this object": Does not reflect the change in the object



- In the Options window, you can specify the situations where the refactoring function is enabled.

  Open the Options window (by selecting **Tools>Options**), select the "Refactoring" category, and specify the situations where the refactoring function is enabled.



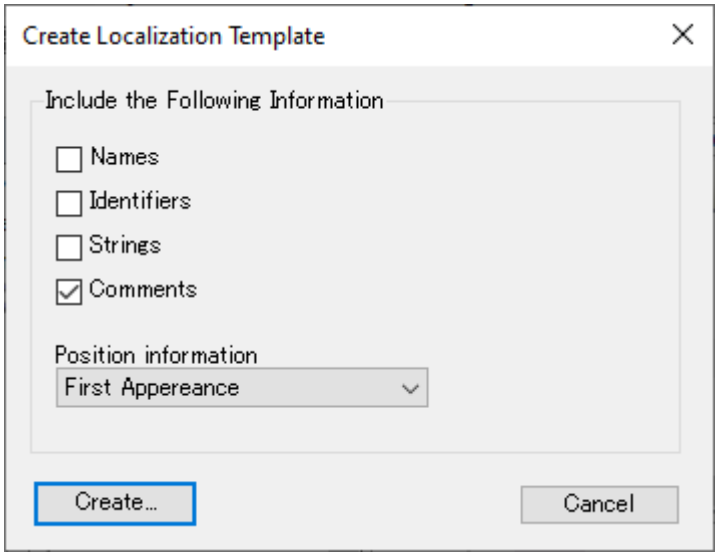
7.7.8 Displaying Programs in Multiple Languages (Project Localization)

The project localization function allows the user to translate and register comments, titles, and other information in the program to display the translated content in the program window.

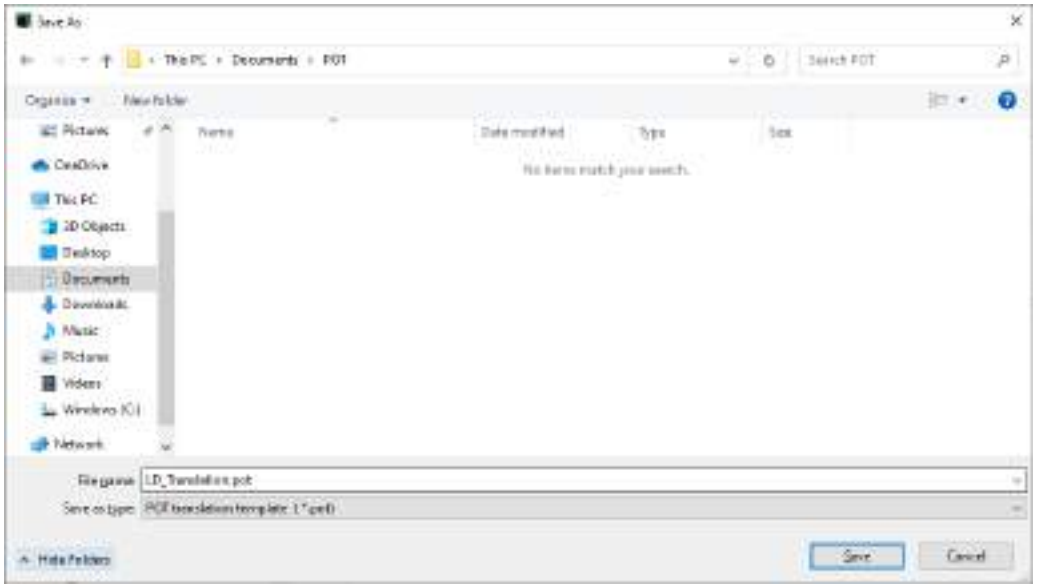
Procedure

1. From the menu bar, select **Project>Project Localization>Create Localization Template**. The "Create Localization Template" dialog box will be displayed.

2. Select information to be translated.
To translate comments and titles in a program, select the "Comments" check box.
To add location information to a template, select "First Appearance" or "All" in the Location information drop-down list.



3. Click the [Create] button.
The "Save As" dialog box will be displayed.



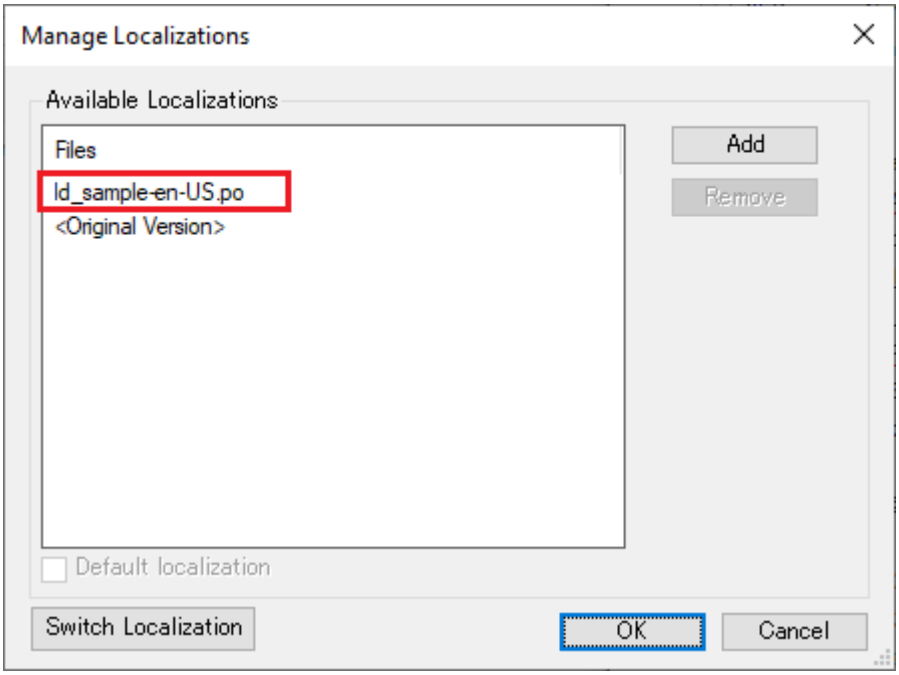
4. Enter a file name and click the Save button.
A POT translation template file (".pot") will be created.

5. For localization, use an editor such as PoEditor to enter translations.
Create a localization file (".po").

Source text — Japanese	Translation — English
// TONの現在時間	// TON Current Time
// 入力X1	// Input X1
自己保持回路	Self-holding circuit
TONファンクションブロックの実行	Execute TON FunctionBlock

6. From the menu bar, select **Project**>**Project Localization**>**Manage Localizations**.
   The "Manage Localizations" dialog box will be displayed.

7. Click the [Add] button.
   The "Open Localization File" dialog box will be displayed. Select a localization file (".po") that has been created.
   The added localization file will be displayed in the "Available Localizations" area.

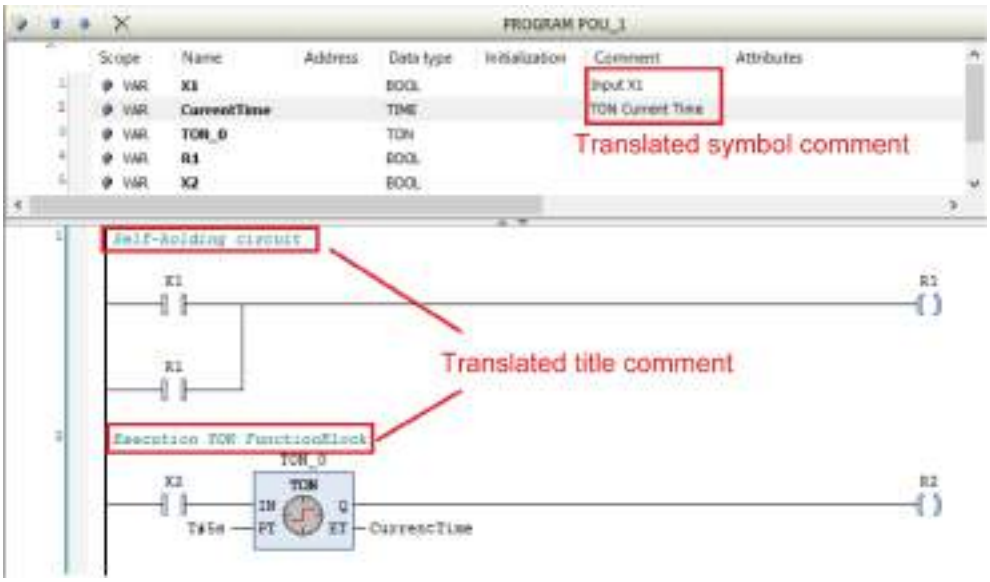


8. Click the [OK] button.
   The "Manage Localizations" dialog box will be closed.

This completes the localization file creation procedure.

Next, the procedure for switching the display is explained below.

9. From the menu bar, select **Project**>**Project Localization**>**Switch Localization**.
   Symbol comments and title comments will be displayed according to the translations in the added localization file.
   To return the display to its original state, select "Switch Localization" again.



Info.

- You can also switch the localization file by clicking on the toolbar.

7.8 Build

When a created program is subject to a build process, objects in the application are compiled. If code generation is executed after the build process is executed, an application to be downloaded to the GM1 controller will be generated.

7.8.1 Build

The syntax of all objects is verified when the build process is executed for the first time.

The syntax of only differences is verified when the build process is executed a second time and thereafter. No application code will be generated.

7.8.2 Rebuild

Verifies the syntax of all objects again.

As is the case with build, no application code will be generated.

Procedure

1. From the menu bar, select **Build**>**Rebuild**.
   The syntax of all objects will be verified. If an error or warning occurs, an error or warning message will be displayed in the message view.
   Check the message displayed in the message view and correct the program as necessary.
   After correcting the program, execute rebuild again.

7.8.3 Code Generation

The GM1 controller generates codes (application codes) to be executed when the application starts.

Displays the remaining program capacity and variable capacity during code generation.

Procedure

1. From the menu bar, select **Build**>**Code Generation**.
   Tests will be executed to check memory allocations, data types, and library availability and code size (in bytes), data size (in bytes), allocated memory contents, and most frequently used address (in bytes) will be displayed in the Messages view.
   Example: Messages view displayed when code generation is completed correctly

Memory area 0	Program capacity	For the maximum capacity, refer to the GM1 Series Reference Manual (Hardware).
Memory area 1	Variable capacity (non-hold)	For the maximum capacity, refer to the GM1 Series Reference Manual (Hardware).
Memory area 2	Input variable	Automatic assignment during code generation
Memory area 3	Output variable	Automatic assignment during code generation
Memory area 4	Internal memory	Automatic assignment during code generation
Memory area 5	Variable capacity (hold)	For the maximum capacity, refer to the GM1 Series Reference Manual (Hardware).

Info.

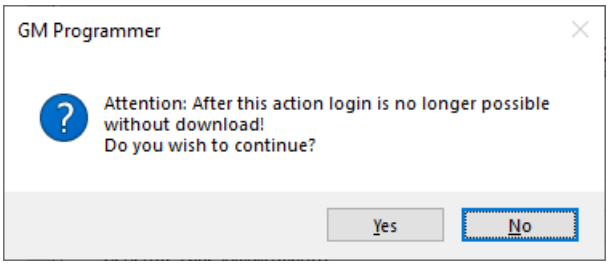
- After application codes are generated, if some codes are changed and code generation is executed again, differential compilation will reallocate memory to only newly added and changed blocks and variables. If memory fragmentation occurs as a result of memory reallocation, the amount of memory that can be actually used will be reduced. To eliminate memory fragmentation, you must perform the procedure described in "7.8.4 Clean".

7.8.4 Clean

Deletes application build information.

Procedure

1. From the menu bar, select **Build**>**Clean**.
   If "Clean" is executed, the following confirmation message will be displayed.



Clicking the [Yes] button executes "Clean".

Info.

- If "Clean" is executed, online change can no longer be performed. Therefore, to log in to the GM1 controller again, you must download the applications.
- If you copy a program object (POU object), execute “Clean” for the copied POU object.

Note

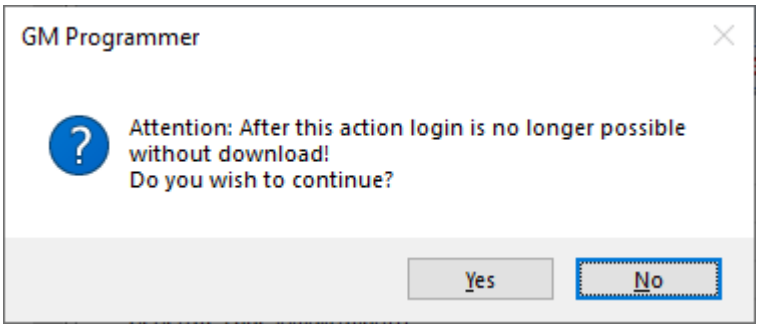
- Even if "Clean" is executed, the variables registered in the global persistent variable list will not be initialized. Other variables and persistent variables will be initialized.

7.8.5 Clean All

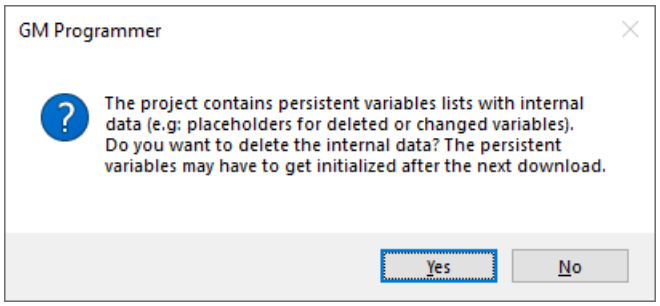
Deletes all application build information in the same way as "Clean".

Procedure

1. From the menu bar, select **Build>Clean All**.
   If "Clean All" is executed, the following confirmation message will be displayed.



2. Clicking the [Yes] button executes "Clean All".
   If variables have been registered in the persistent variable list, the following confirmation message will be displayed.



3. If you delete internal data, click the [Yes] button.
If you do not delete it, click the [No] button.

Note

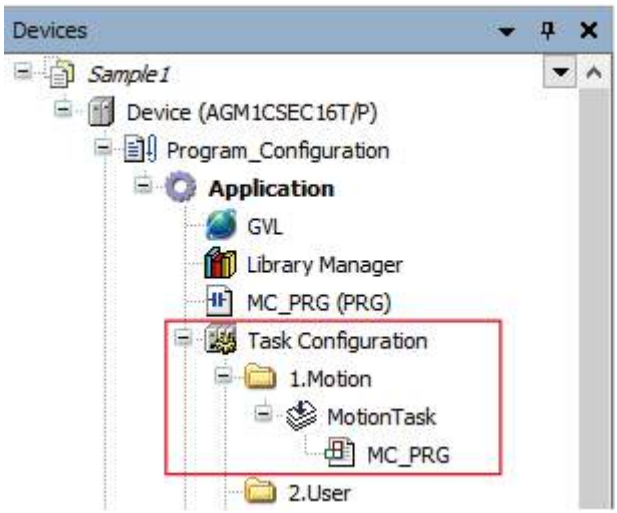
- If you have made changes such as "Add new variables to the top of persistent variable list", note that actual variable values could become different values because the memory areas for retaining variables have become misaligned.

7.9 Tasks

The GM1 series motion controller executes the following three tasks.

Task	Description
MotionTask	This is a user program task to perform motion control. It is given the highest priority. Only one MotionTask is allowed for each project.
UserTask	This is a user program task to perform control other than motion control. The user can set the level of priority. Up to 50 tasks can be registered in a single project.
SystemTask	This is a task that is used by the system and cannot be added by user programs. It is processed while other tasks are inactive.

To execute a user program, the program (POU object) must be added to tasks.
Add motion control programs to MotionTask. Add other control programs to UserTask.

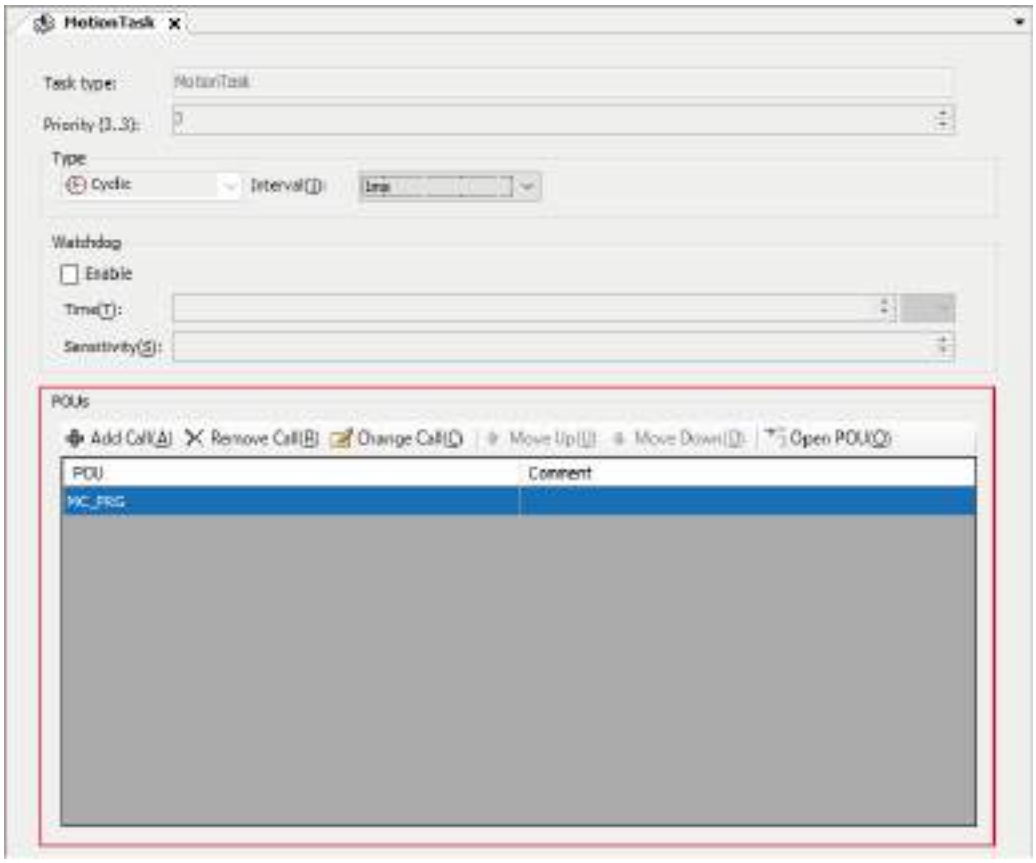


7.9.1 Adding Programs

POU objects of programs can be added to tasks.
For example, use the following procedure to add program "LD_POU" to task "MotionTask".

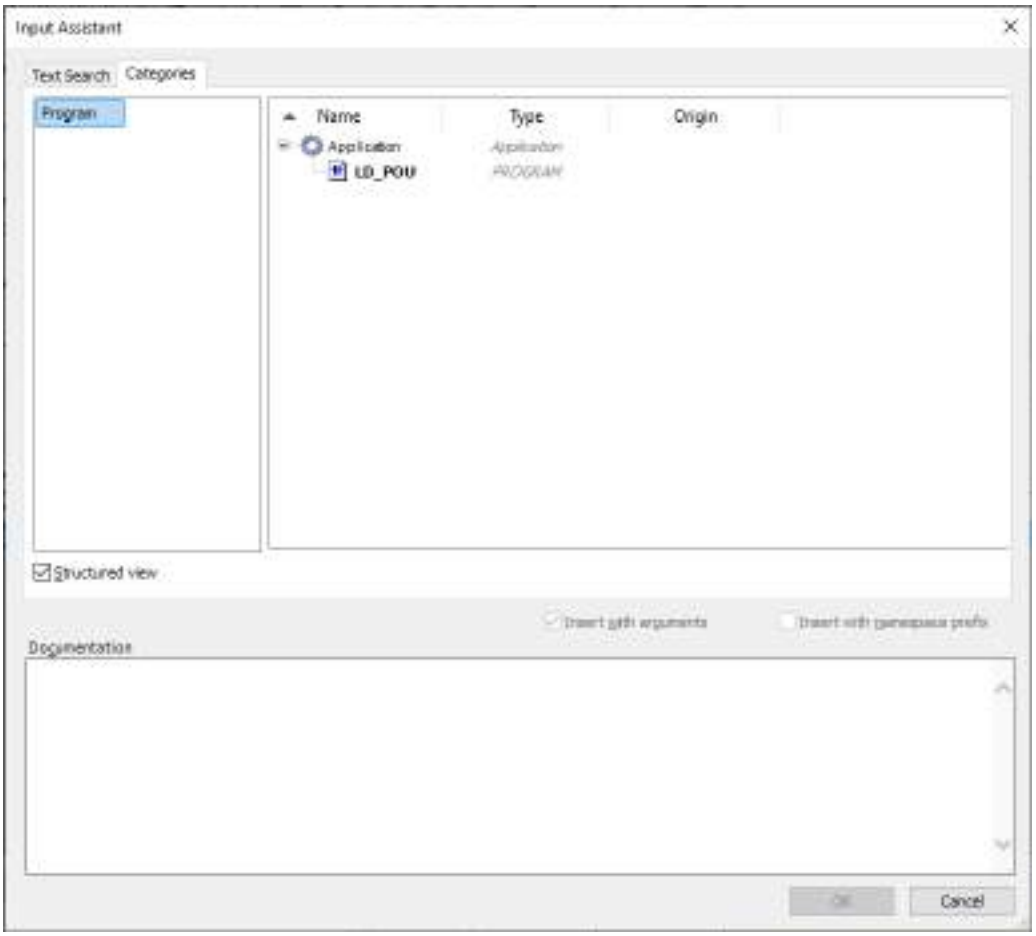
Procedure

1. Double-click the "MotionTask" object in the navigator pane.
   The "MotionTask" task configuration window will be displayed in the main pane.

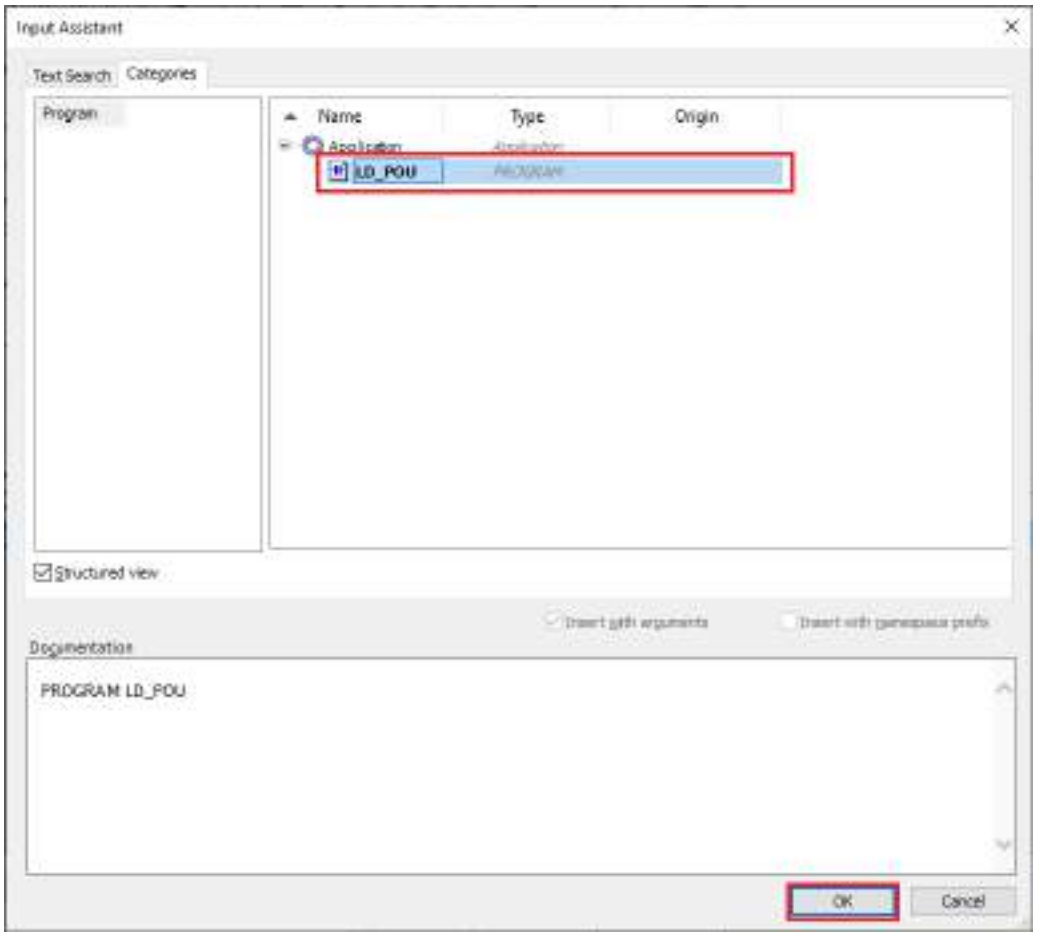


2. In the "MotionTask" window, click the [Add Call] button.

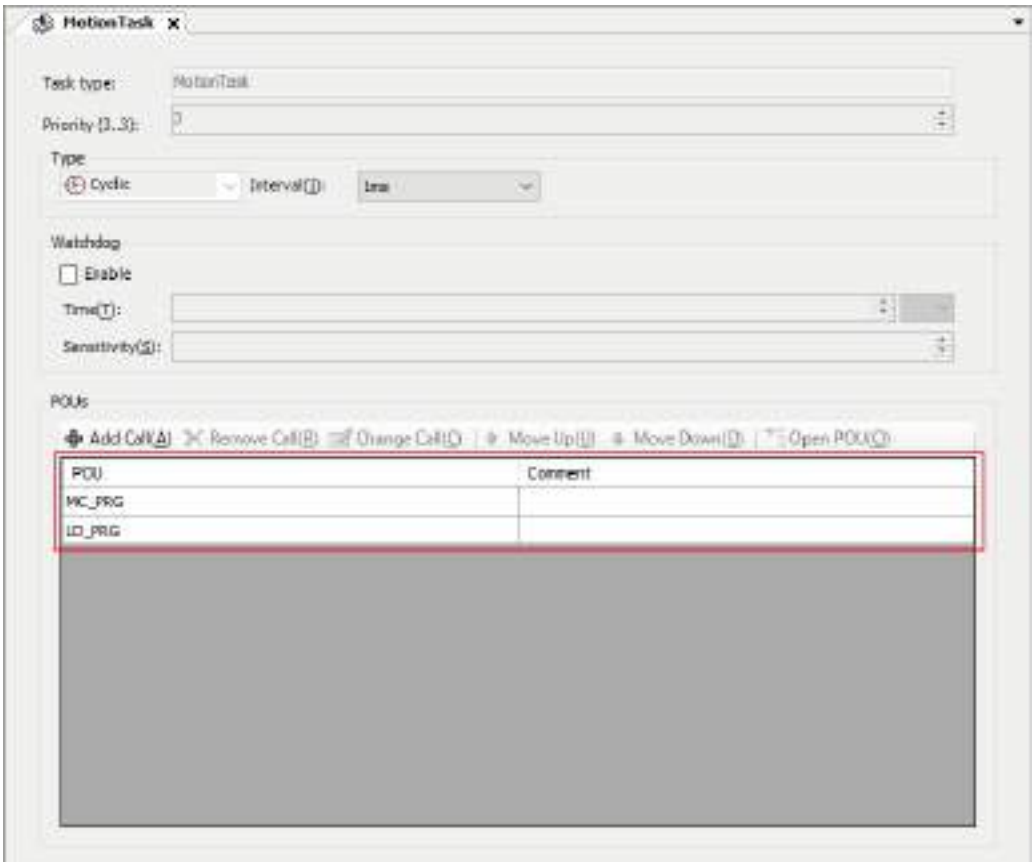
The "Input Assistant" dialog box will be displayed.



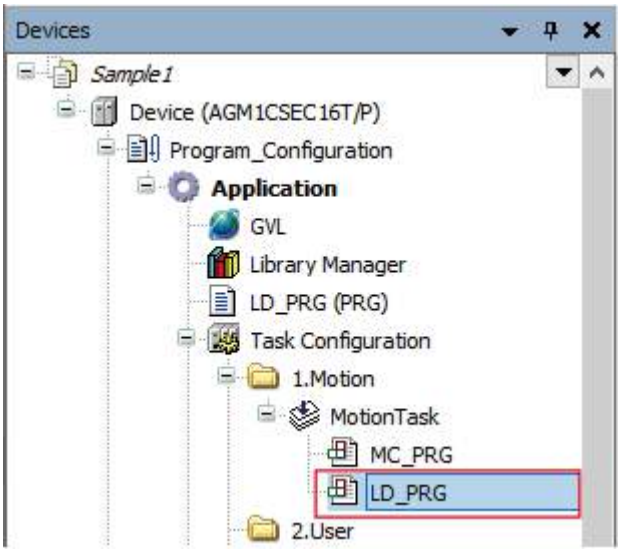
3. Select the POU object (LD_POU) of the program to be added to the task and click the [OK] button.



The POU object of the program will be added to the task.



Selecting a cell in the "Comment" column allows you to edit the comment.
The task will also be added to the navigator pane.



Info.

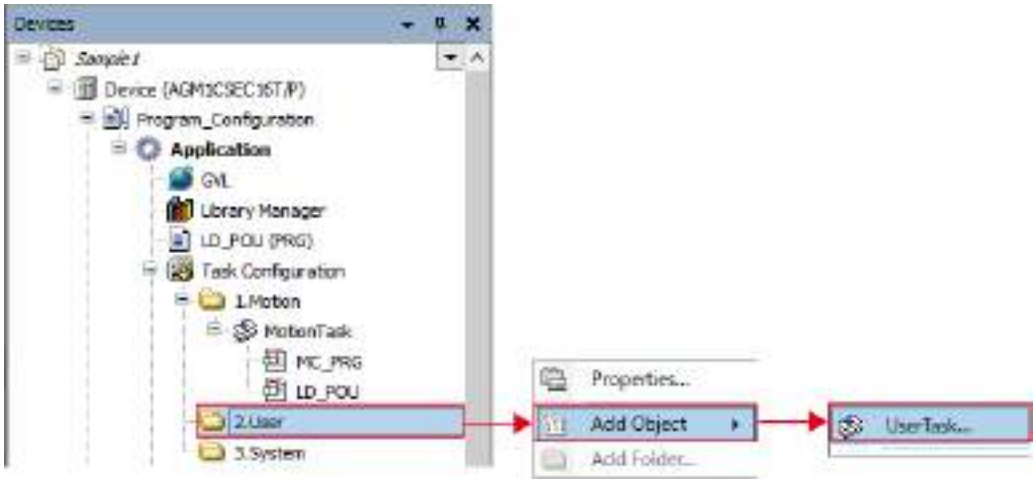
- You can also add a POU object to a task by dragging the POU object in the navigator pane and dropping it onto the task object.

7.9.2 Adding a UserTask

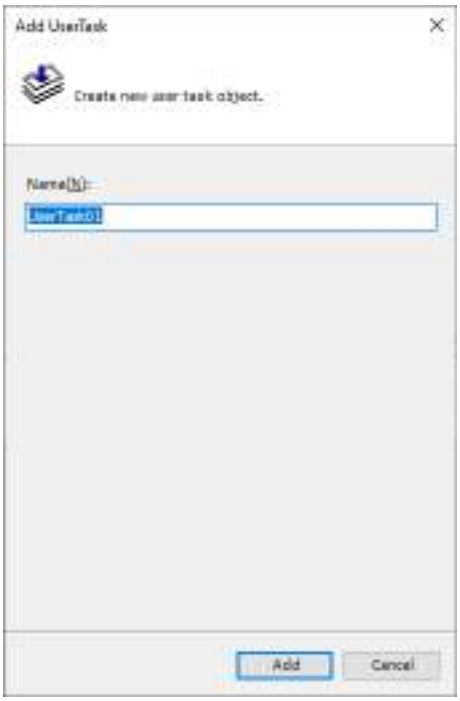
UserTask can be added to a project.

Procedure

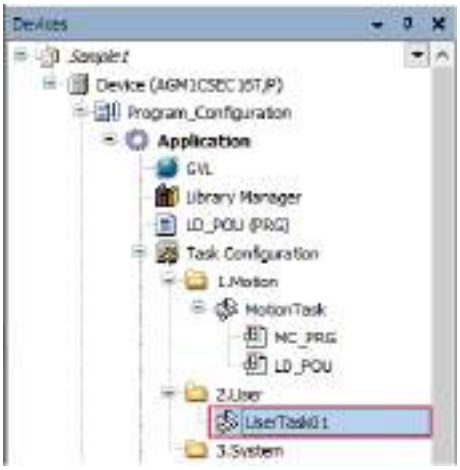
1. Right-click "2.User" in the navigator pane and then select **Add Object**>**UserTask** from the context-sensitive menu that is displayed.



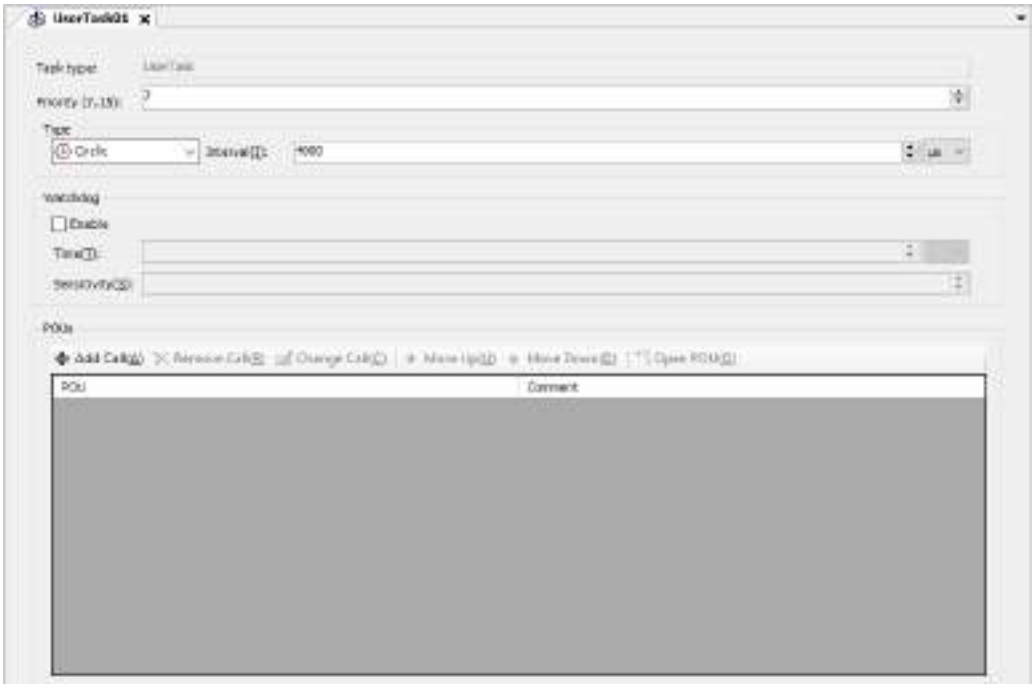
The "Add UserTask" dialog box will be displayed.



2. Enter a task name in the Name field and click the [Add] button.
   The task object will be added.



3. Add a POU object to the UserTask.
   The added POU object will be executed as a UserTask.
   Double-clicking the object of the added UserTask displays a task configuration window where task settings can be configured.

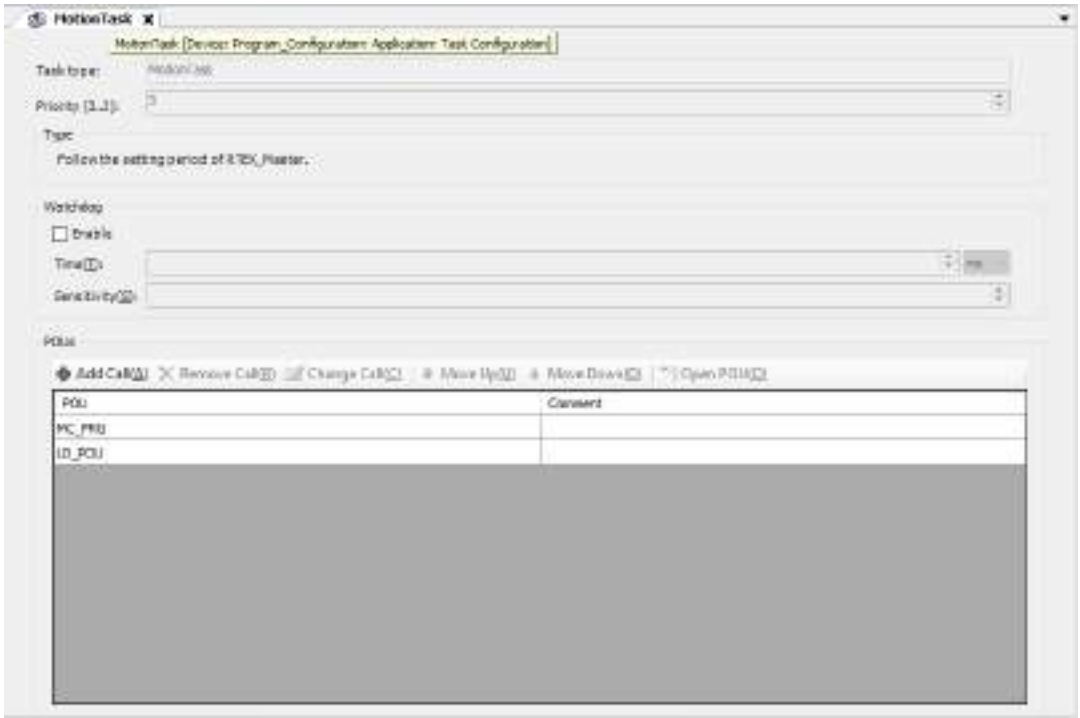


7.9.3 Task Configuration Window

The task configuration window allows the user to configure settings related to task execution, such as execution priorities, execution methods, execution intervals, and watchdog timer.

For the procedure for displaying the task configuration window, refer to "7.9.1 Adding Programs".

Example: Task configuration window for task "MotionTask"



Item	Description
Task type	This field displays the type of the task. It displays MotionTask, UserTask, or SystemTask.
Priority	This field displays the priority of the task. The smaller the value, the higher the priority. For UserTask, the priority can be set between 7 and 15 inclusive.
Type	This field specifies the processing method for the task. For UserTask, one of the following two types can be selected. ● Cyclic: Processes the task at intervals. Specify a task interval in the "Period" field. ● Event: Starts task processing as soon as a rising edge of the global variable specified in the "Event" field is detected
Watchdog	If the "Enable" check box is selected, when the program execution time exceeds the preset time, the task will enter an error state and comes to a halt. The stop conditions are divided into the following two cases: ● Case where the program execution time exceeds the number of times specified in "Sensitivity" or the time specified in "Time" ● Case where the program execution time exceeds "Sensitivity"×"Time" during a single cycle (Example: If "Sensitivity" is set to "3" and "Time" is set to "t#20ms", when the execution time exceeds 60 ms during a single cycle, the task will stop.) If the watchdog timer causes the task to stop, the event will be recorded in the "Log" tab of the device editor.

Item	Description
	

8 Connecting the GM1 Controller and PC

8.1 Flow of Operation Check 8-2

8.2 Connecting the GM1 Controller and PC 8-3
- 8.2.1 Selecting a Connection Port for GM Programmer 8-3
- 8.2.2 Connecting the GM1 Controller and PC with a Cable 8-3
- 8.2.3 Operation when Power is ON 8-3

8.3 Operation Mode Switching 8-5

8.4 Communication Setting 8-6
- 8.4.1 Adding USB Ports 8-6
- 8.4.2 Setting the LAN Port 8-8

8.5 Connecting to the GM1 Controller 8-10

8.6 Setting Time 8-12

8.7 Other Settings 8-14
- 8.7.1 Changing the Device Name 8-14
- 8.7.2 Sending Echo Services 8-15
- 8.7.3 Device Preference Management 8-17
- 8.7.4 Confirmed Online Mode 8-18

8.8 Login / Logout 8-20
- 8.8.1 Login 8-20
- 8.8.2 Logout 8-21
- 8.8.3 Download 8-21
- 8.8.4 Online Change 8-23
- 8.8.5 Code Analysis (Static Analysis Light) 8-23

8.9 Source Upload 8-27

8.10 Commissioning 8-29
- 8.10.1 Online Config Mode 8-29
- 8.10.2 Conducting Commissioning for Servo Amplifiers 8-30

8.1 Flow of Operation Check

This chapter explains how to connect the PC where GM Programmer is installed and the GM1 controller and operate the GM1 controller from the PC.

First, this section explains the flow of operation check for a program that is created.

1. Build

Execute build to check a program that is created. If an error occurs, correct the program and execute build again. If code generation is executed after build is completed normally, an application will be generated.

Operation: From the menu bar, select **Build**>**Build**. From the menu bar, select **Build**>**Generate Code**.

2. Connecting to the GM1 controller

Connect the PC where GM Programmer is installed to the GM1 controller.

Operation: Double-click the Device object in the navigator pane to display the "Communication Settings" window, and then select the "Network Scan" tab and select the GM1 controller to which you want to connect.

3. Login

Log in to the GM1 controller. When you log in to the GM1 controller, applications are downloaded to the GM1 controller.

Operation: From the menu bar, select **Online**>**Login**.

4. Debug

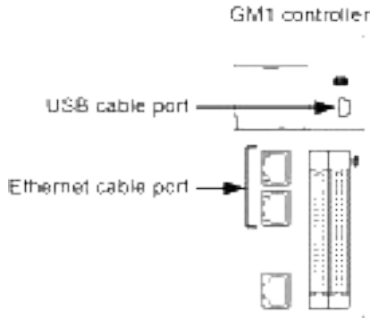
Execute the application and perform debugging. If there are any problems with behaviors, log out of the GM1 controller, correct the problem, and execute build again.

Operation: From the menu bar, select "Debug" and then each debug menu item.

8.2 Connecting the GM1 Controller and PC

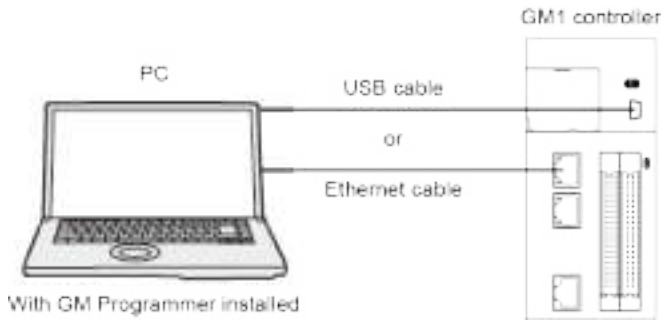
8.2.1 Selecting a Connection Port for GM Programmer

Select either LAN port connection or USB port connection.



8.2.2 Connecting the GM1 Controller and PC with a Cable

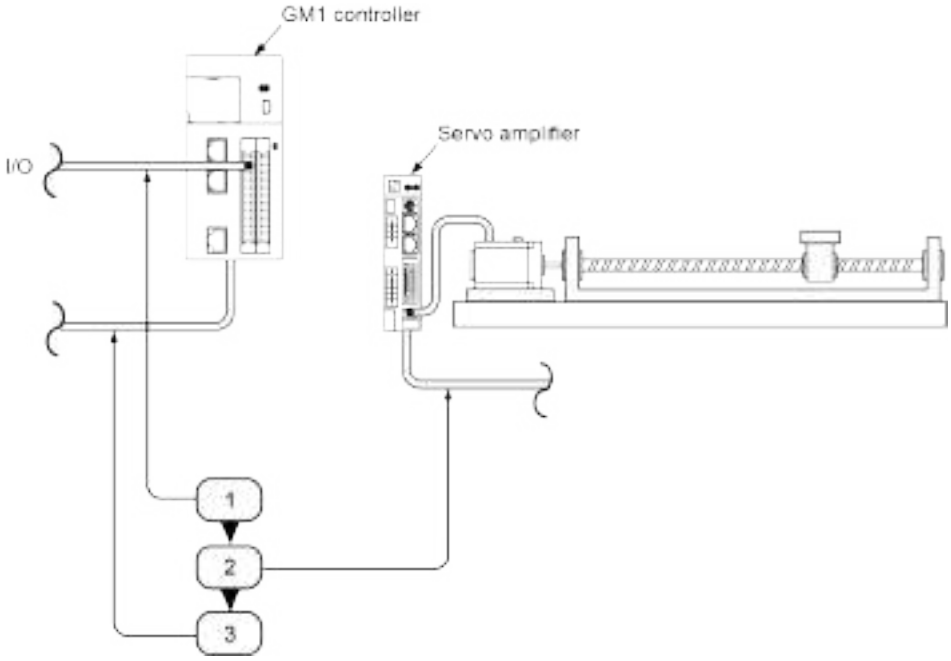
Use an Ethernet cable or USB cable to connect the GM1 controller and a PC on which the GM Programmer is installed.



8.2.3 Operation when Power is ON

When turning ON the power supply to the system incorporating the GM1 controller, consider the nature and statuses of any external devices connected to the system, and take sufficient care so that turning ON the power supply will not initiate unexpected movements.

1. Turn ON the power supplies to the I/O devices connected to the GM1 controller.
2. Turn ON the power supply to the servo amplifier.
3. Turn ON the power supply to the GM1 controller.


GM1 controller
I/O
Servo amplifier
1
2
3

8.3 Operation Mode Switching

■ Switching to the RUN mode

There are the following two methods.

- Press the operation button (►) on the GM Programmer while the STOP LED is lit.
- Set the RUN/STOP switch on the GM1 Controller to RUN.

Info.

- The switch cannot be set to the RUN mode if an error that does not allow to continue operation has occurred or if an exceptional situation has occurred.

■ Switching to the STOP mode

There are the following two methods.

- Press the stop button (■) on the GM Programmer while the RUN LED is lit.
- Set the RUN/STOP switch on the GM1 Controller to STOP.

8.4 Communication Setting

8.4.1 Adding USB Ports

This function allows the user to set a USB port as the communication interface between a tool such as GM Programmer or PANATERM Lite for GM and the GM1 Controller.

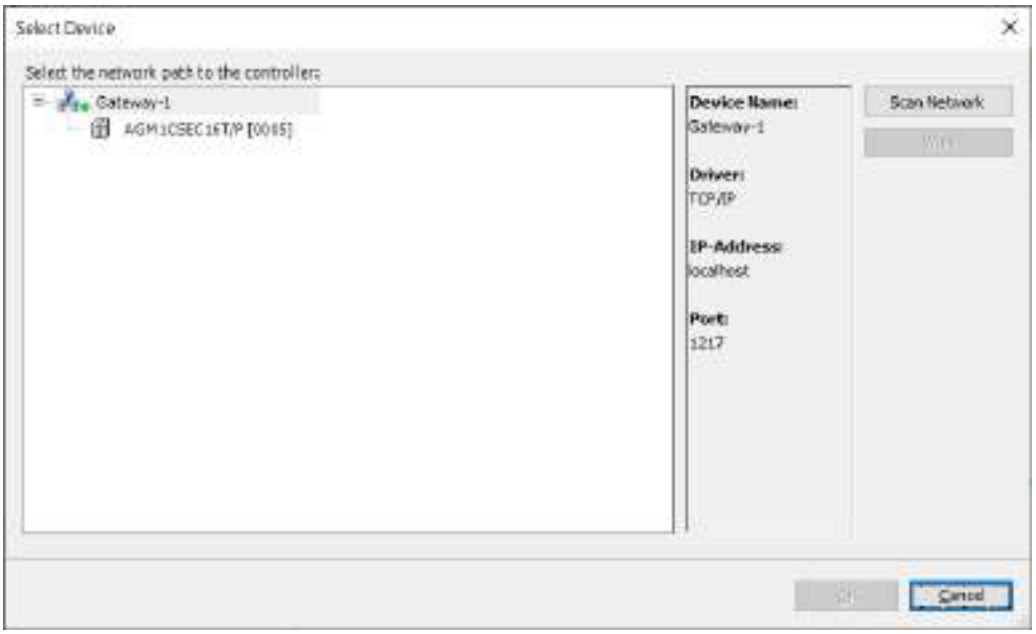
1 2 Procedure

1. Connect the GM1 Controller and PC with a USB cable.
2. From the menu bar, select **Online**>**Add USB Port**.

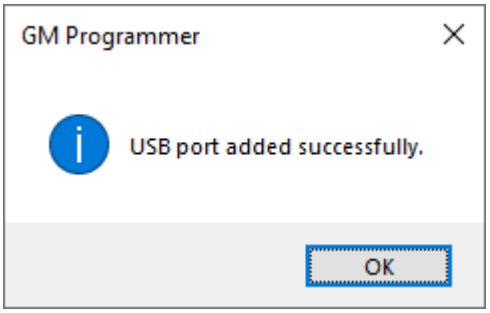
The "Add USB Port" dialog box will be displayed.

3. Click the [OK] button.
   A dialog box to restart the Gateway will be displayed.

4. Click the [OK] button.
The "Select Device" dialog box will be displayed.



5. Select a GM1 Controller to which you want to connect and click the [OK] button.
When the connection is completed, a dialog box will be displayed to notify successful connection.



6. Click the [OK] button.
A USB port will be added as the communication interface between the PC and GM1 Controller.

8.4.2 Setting the LAN Port

The following table shows the default network settings.

When connecting GM Programmer and the GM1 controller via a LAN port, match the network settings of the PC with those of the GM1 controller.

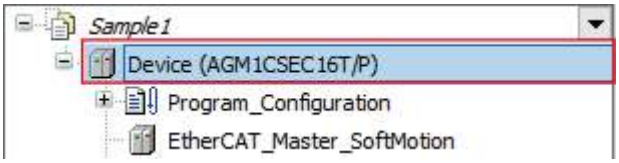
	LAN port 1	LAN port 2
IP address	192.168.1.5	192.168.2.5
Subnet mask	255.255.255.0	255.255.255.0
Default gateway	192.168.1.1	0.0.0.0

Set the IP addresses of LAN port 1 and LAN port 2 so that their network (subnet) addresses are different.

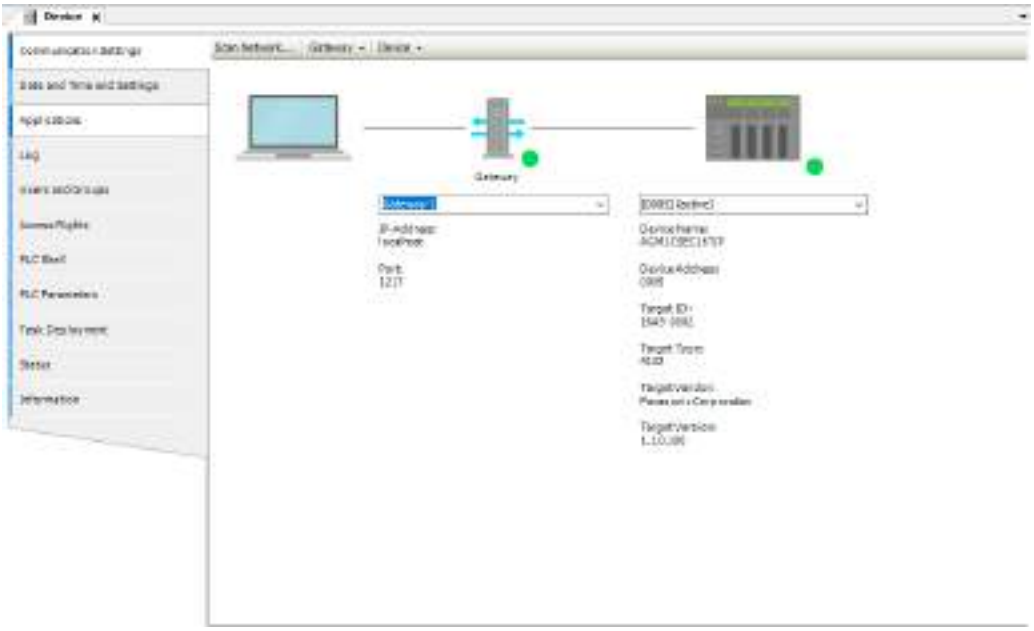
Network settings can be changed using the [PLC parameters] tab of the "Device" object in the navigator pane, as described below.

Procedure

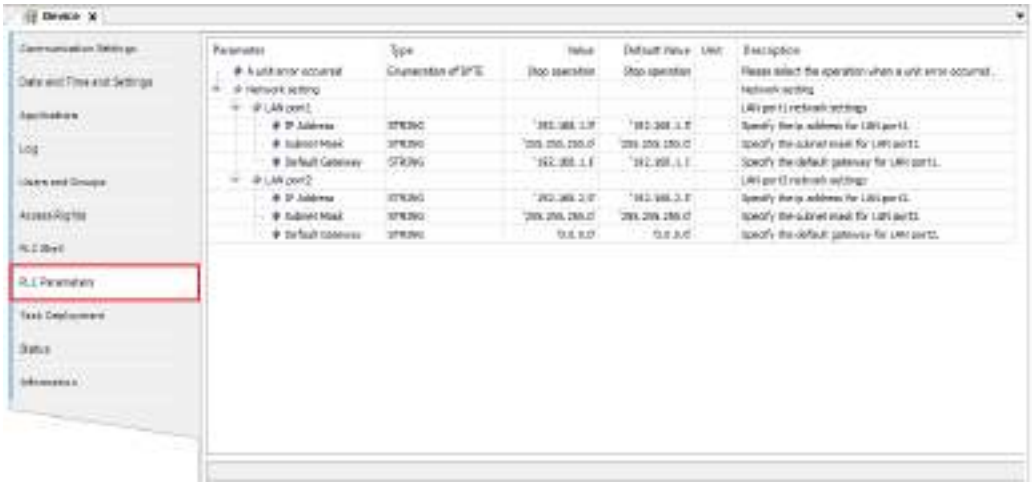
1. Double-click the [Device] object in the navigator pane.



The Device setting window will be displayed.



2. Click the "PLC parameters" tab in the Device setting window.



3. Set the IP address, subnet mask, and default gateway for each LAN port.

4. Download the project to the GM1 controller.

Info.

- If you change the network settings for the LAN port that connects the GM1 controller and GM Programmer with a LAN cable, the connection will be temporarily disrupted.

8.5 Connecting to the GM1 Controller

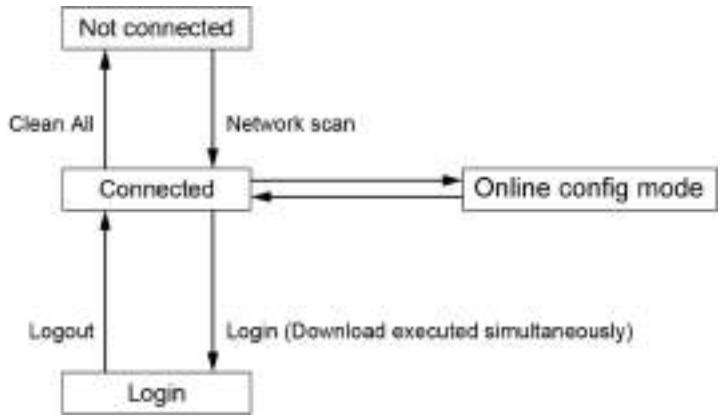
Connect the PC where the GM Programmer is installed to the GM1 Controller.

The connection status of the PC includes "Connected", "Connection as a device user", "Login", and "Online config mode".

Depending on the connection status, operations that can be executed are different.

If the Controller is provided with a device user registration, connection must be made as the device user.

■ Without device user registration



List of available GM1 Controller operations

Function	Not connected	Connected	Login	Online config mode
Setting / acquiring Controller information	×	○	○(Note 1)	×
Application management	×	×(Note 2)	○	×
Reset	×	×(Note 3)	○	×(Note 4)
Security	×	×	○	○
Debug	×	×	○	×
Commissioning	×	×	×	○

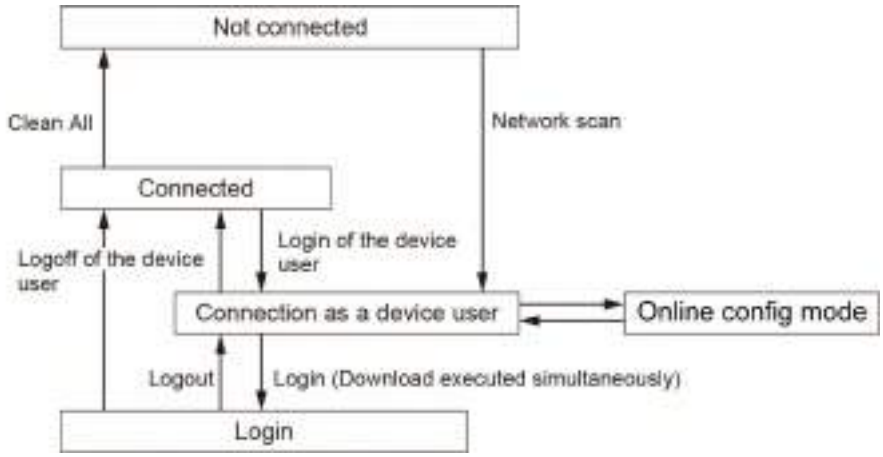
(Note 1) Not possible to operate the PLC Shell.

(Note 2) Possible to upload the source.

(Note 3) Possible to reset the device (PLC initialization) or to delete device application from the device.

(Note 4) Possible to reset the device (PLC initialization).

■ **With device user registration**



List of available GM1 Controller operations

Function	Not connected	Connected	\as a device user	Login	Online config mode
Setting / acquiring Controller information	×	×	○	○(Note 1)	×
Application management	×	×	×(Note 2)	○	×
Reset	×	×	×(Note 3)	○	×(Note 4)
Security	×	×	○(Note 5)	○	○
Debug	×	×	×	○	×
Commissioning	×	×	×	×	○

(Note 1) Not possible to operate the PLC Shell.

(Note 2) Possible to upload the source.

(Note 3) Possible to reset the device (PLC initialization) or to delete device application from the device.

(Note 4) Possible to reset the device (PLC initialization).

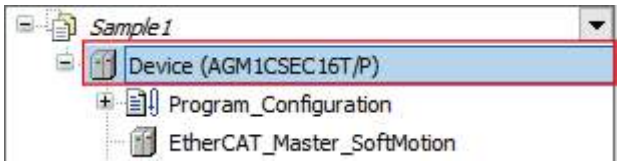
(Note 5) Addition of the device user, changing the password for the device user, or deletion of the device user cannot be made if the user of the Device Editor is not synchronized with "Synchronization" of the group tab.

8.6 Setting Time

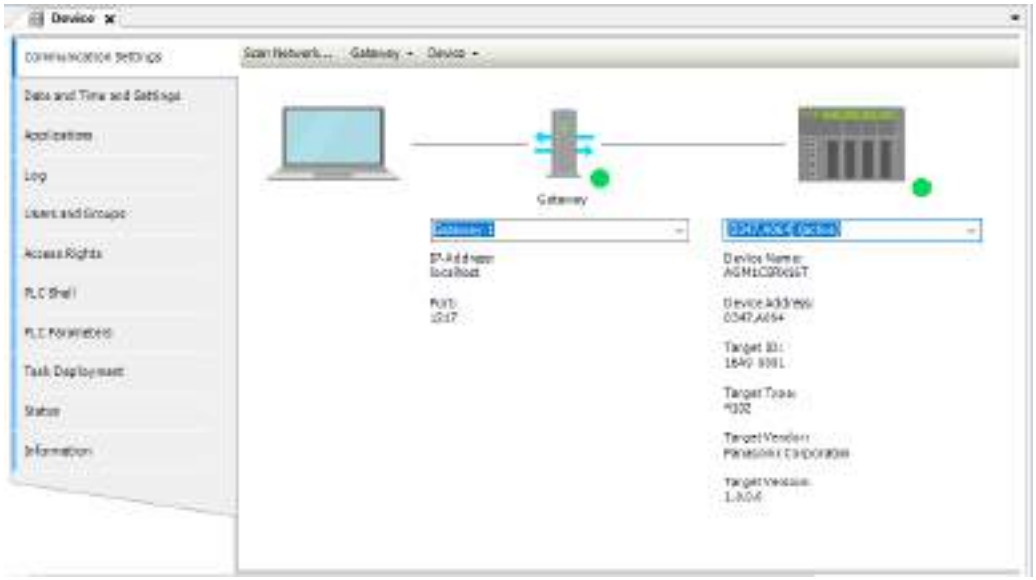
After executing the Network Scan function, set time for the GM1 controller.
You can enter date and time directly or by getting date and time from the PC.

1 2 Procedure

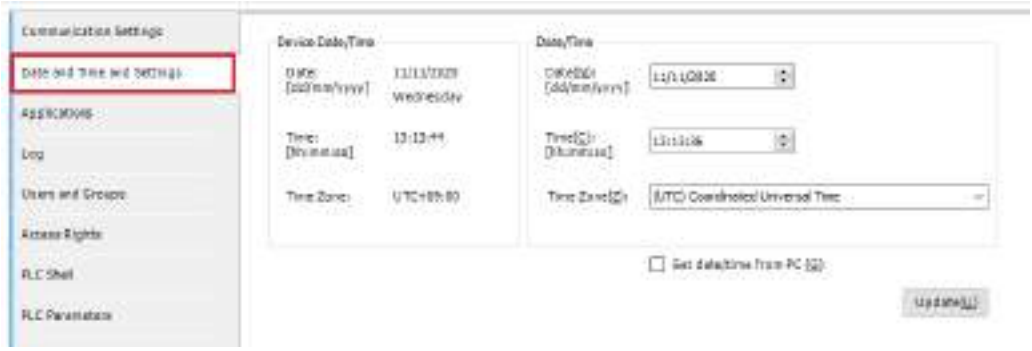
1. Double-click the [Device] object in the navigator pane.



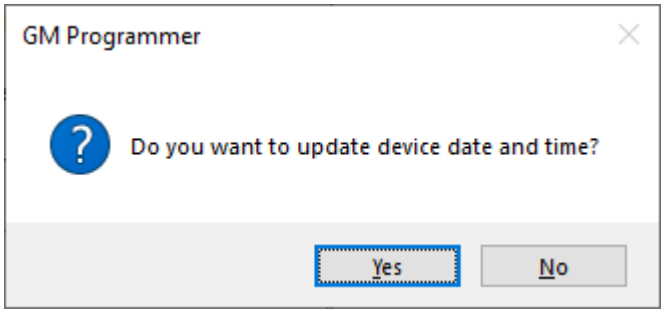
The Communication Settings window for the device will be displayed.



2. Click the "Date and Time and Settings" tab.
   The Date and Time Settings window will be displayed.
   The "Device Date and Time" section displays the current date and time of the GM1 controller.



3. Change the date, time, and time zone in the "Date and Time" section or select the "Get date / time from PC" check box, and click the [Update] button.
   A confirmation message will be displayed.



4. Click the [Yes] button.
   The date and time of the GM1 controller will be updated.

8.7 Other Settings

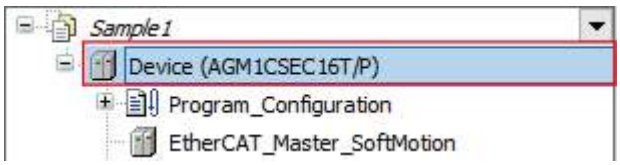
For the connected device, configure settings described in "Change Device Name", "Sending Echo services", "Device preference management", and "Confirmed Online Mode".

8.7.1 Changing the Device Name

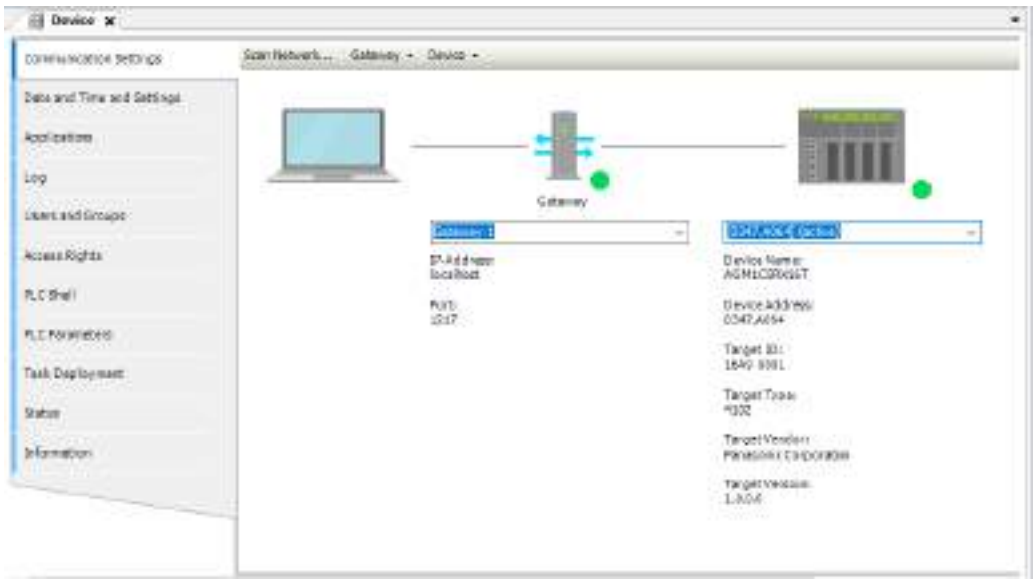
This section explains how to change the name of the device connected via "Network Scan".

Procedure

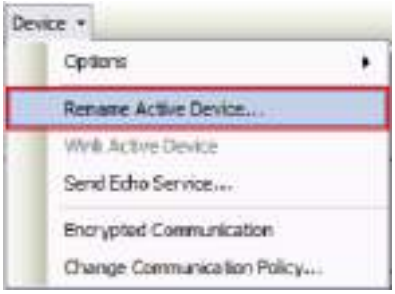
1. Double-click the "Device" object in the navigator pane.



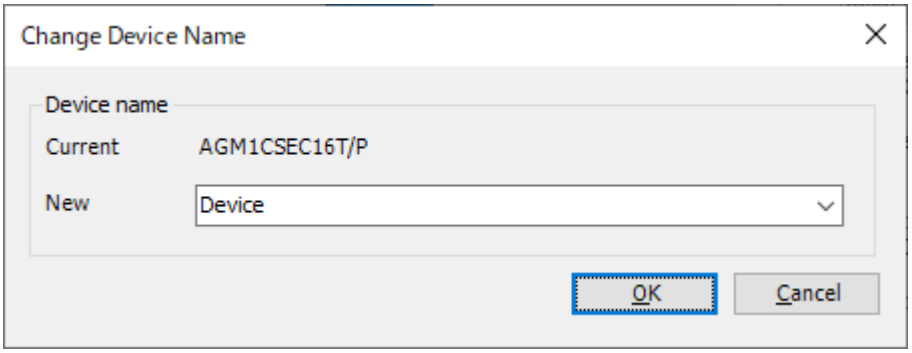
The setting pane will be displayed in the main pane.



2. Select "Rename Active Device" from the "Device" menu.
The "Change Device Name" dialog box will be displayed.



3. Enter a new device name and click the [OK] button.
The device name will be changed.

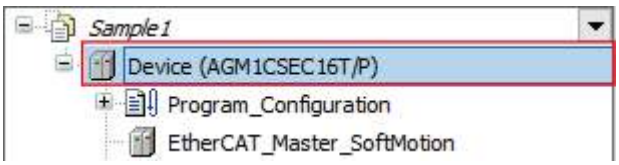


8.7.2 Sending Echo Services

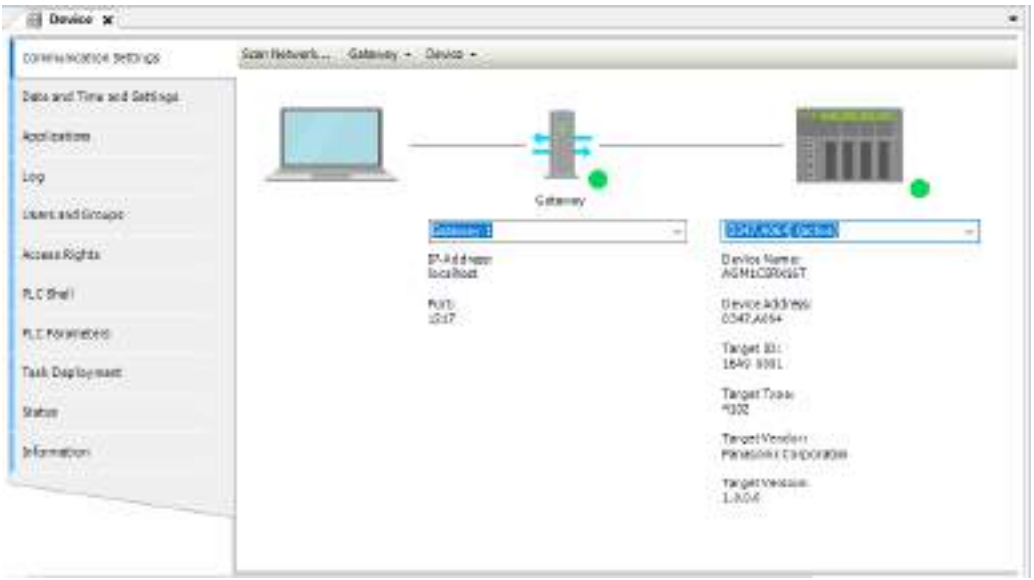
This section explains how to send Echo services to the device connected via "Network Scan". Echo services can be used to conduct a network test.

1 2 Procedure

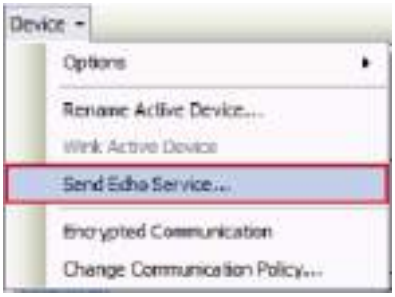
1. Double-click the "Device" object in the navigator pane.



The setting pane will be displayed in the main pane.



2. Select "Send Echo Service" from the "Device" menu.



The results of five transmissions with no data size followed by five transmissions with data size will be displayed.

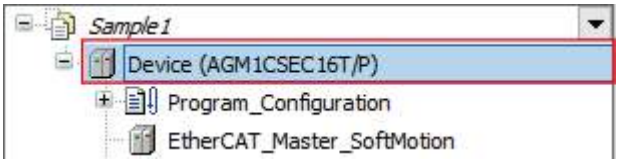
8.7.3 Device Preference Management

Devices connected via "Network Scan" can be managed by registering them as favorite devices.

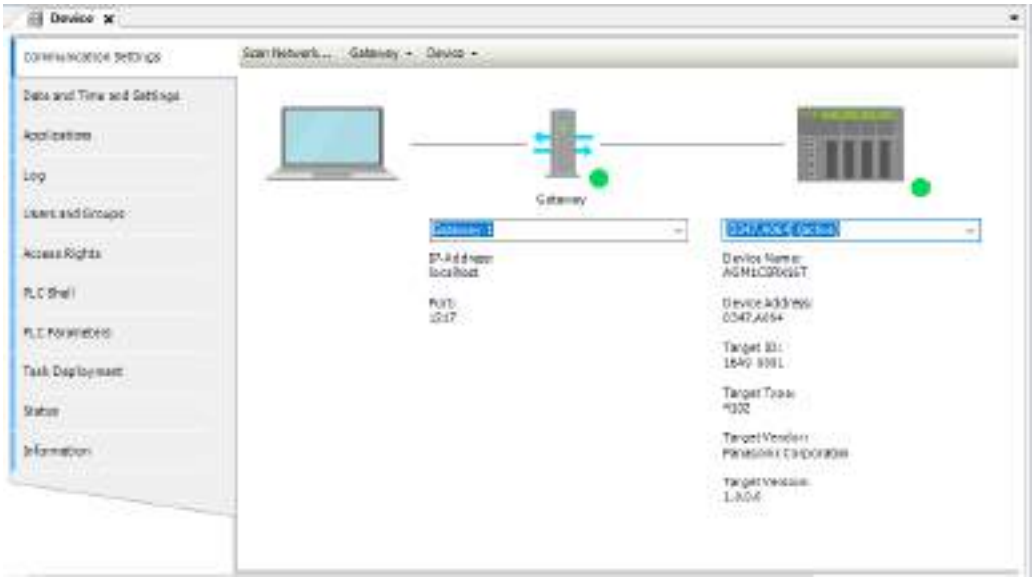
Devices registered as favorite devices will be displayed in the device selection list in the Communication Settings window.

Procedure

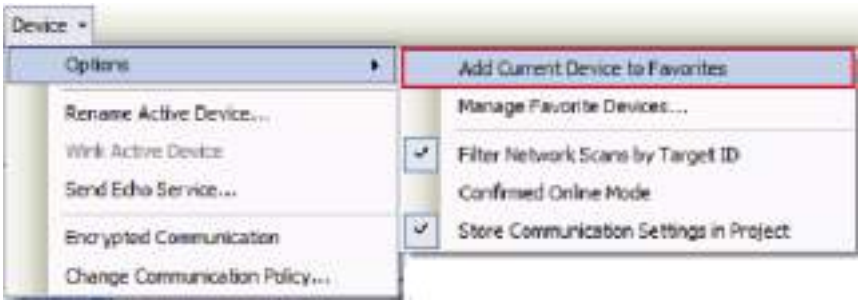
1. Double-click the "Device" object in the navigator pane.



The setting pane will be displayed in the main pane.



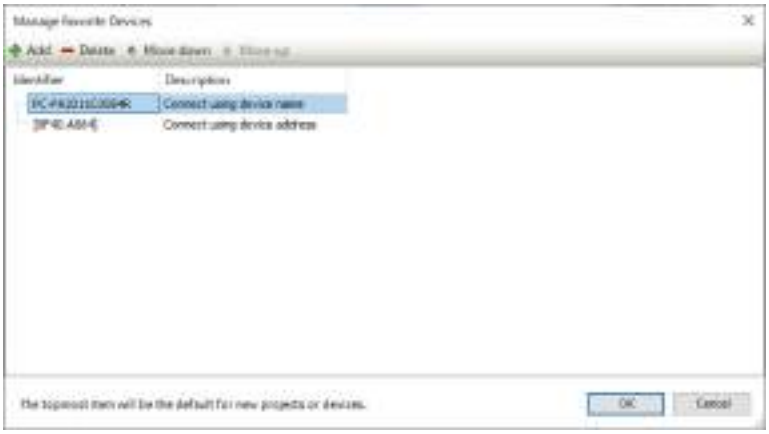
2. From the "Device" menu, select "Options" and then "Add Current Device to Favorites".
The connected device will be registered as a favorite device.



Info.

- Devices registered as favorite devices can be viewed in the "Manage Favorite Devices" dialog box.

  The "Manage Favorite Devices" dialog box can be displayed by selecting "Options" and then "Manage Favorite Devices" from the "Device" menu.



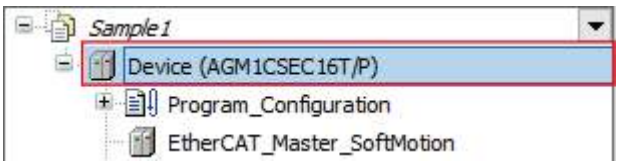
8.7.4 Confirmed Online Mode

A confirmation message can be displayed when an attempt is made to implement the following actions.

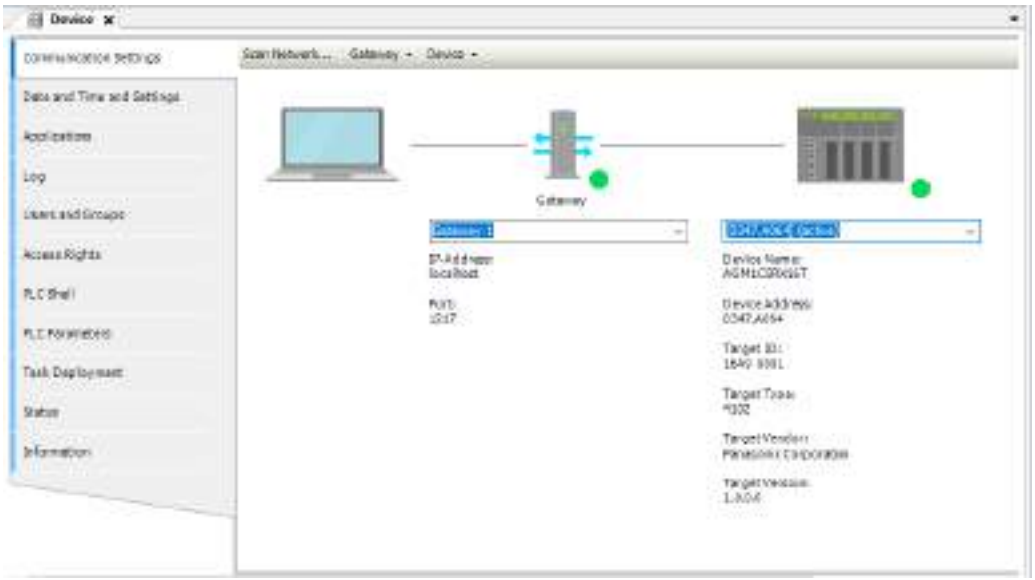
- Login
- Operation
- Stop
- Single Cycle
- Force Values
- Write Values
- Unforce Values

Procedure

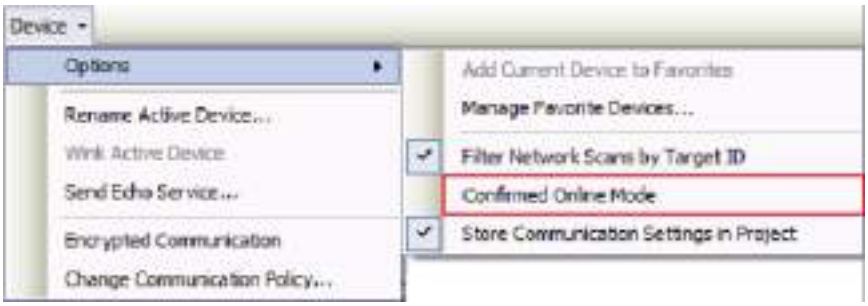
1. Double-click the "Device" object in the navigator pane.



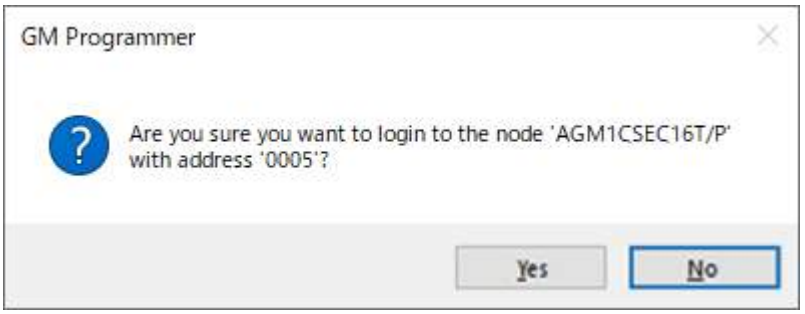
The setting pane will be displayed in the main pane.



2. From the "Device" menu, select "Options" and then "Confirmed Online Mode".



When an attempt is made to log in with "Confirmed Online Mode" selected, the following message is displayed.



8.8 Login / Logout

GM Programmer allows the user to log in to the GM1 controller.

- During login, the application and source code generated by code generation are downloaded to the GM1 controller.

- The combination of the application and source code downloaded to the GM1 controller differs according to the operations shown in the table below.

○: Downloaded

×: Not downloaded

Operation	Boot application	Source code
Login	○	○
Initial download	○	○
Downloading after changing the program "Update boot project" check box: Selected	○	○
Downloading after changing the program "Update boot project" check box: Cleared	×	×
Downloading after changing the project	○	○
Online change "Update boot project" check box: Selected	○	○
Online change "Update boot project" check box: Cleared	×	×
Generating a boot application	○	○

After logging in to the GM1 controller, you can perform debug operations such as starting or stopping the GM1 controller.

8.8.1 Login

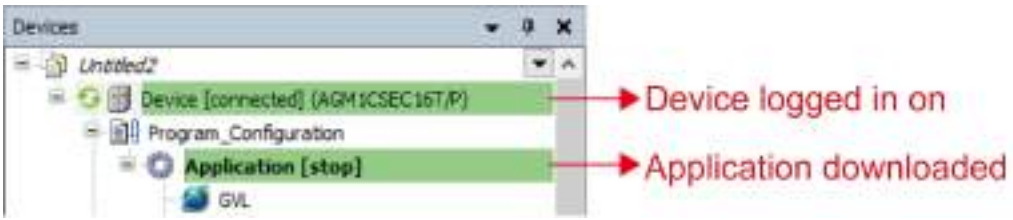
GM Programmer allows the user to log in to the GM1 Controller. When "Login" is executed, applications are downloaded to the GM1 Controller.

Procedure

1. From the menu bar, select **Online**>**Login**, or press the <Alt> key and the <F8> key simultaneously.

   A confirmation message will be displayed, asking whether to download the applications to the GM1 controller (device).

2. Click [Yes].
   The applications will be downloaded to the GM1 Controller at the same time as you log in to the GM1 Controller (device).
   "connected" will be displayed at the [Device] object in the navigator pane and the status of the downloaded applications will be displayed.



Info.

- You can also log in by clicking on the toolbar.
- If you log in again after the applications have been downloaded, the confirmation message will not be displayed.

8.8.2 Logout

This function allows the user to log out from the device to which the user logged in.

Procedure

1. From the menu bar, select **Online**>**Logout**, or press the <Ctrl > + <F8> key simultaneously.
   You will be logged out.

Info.

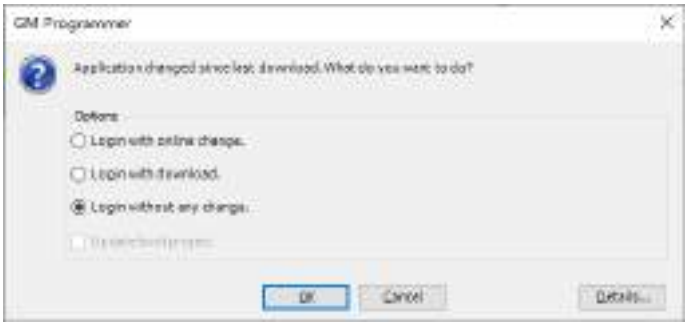
- You can also log out by clicking on the toolbar.

8.8.3 Download

After changing programs and logging in without executing online change, you can download applications while being logged in.
The boot application is also updated during download.

Procedure

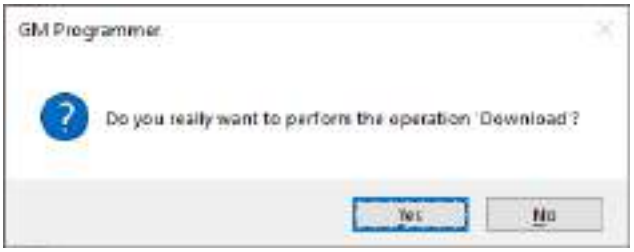
1. When executing "login", select "Login without any change" and click the [OK] button.



Applications will not be downloaded to the GM1 controller.

2. From the menu bar, select **Online**>**Download**.

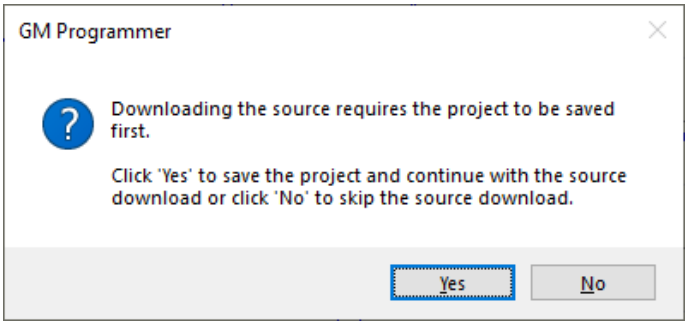
A confirmation message will be displayed, asking whether to download the source code to the GM1 controller.



3. Click [Yes].

If the project has not yet been saved, a confirmation message will be displayed, asking whether to save the project.

4. If you do not save the project, the source code will not be downloaded. In this case, click the [Yes] button.



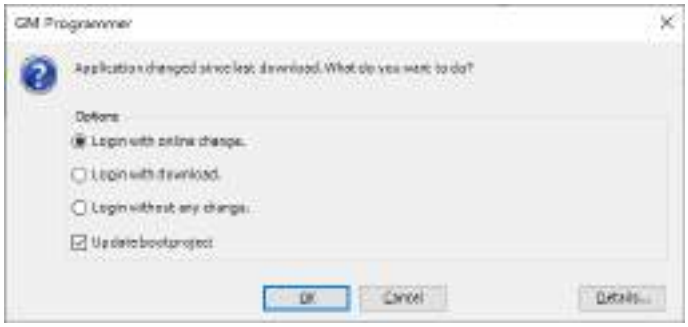
Info.

- To delete the downloaded source code from the GM1 controller, execute "Reset Device". For details on reset, refer to "9.5.1 Reset Warm, Reset Cold, and Reset Origin".

8.8.4 Online Change

Online change allows the user to change applications without having to stop the GM1 controller during operation. Executing online change downloads only different applications to the GM1 controller.

If the applications generated by build during login differ from the applications within the GM1 controller, the following dialog box will be displayed.



Login with online change

Executes login by downloading only different applications without stopping the GM1 controller

Login with download

Executes login by downloading applications generated by build with the GM1 controller stopped

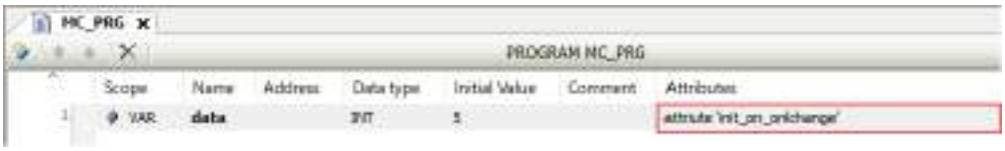
Login without any change

Executes login without downloading the applications generated by build

Info.

- Do not clear the "Update bootproject" check box. If you clear the check box, the applications will not be saved when the GM1 controller is turned OFF.
- When changing the initial value of a variable with online change, be sure to add "attribute 'init_on_onlchange'" declaration to the attribute of the target variable.

  If the declaration is not added, the initial value changed with online change will not be applied.



8.8.5 Code Analysis (Static Analysis Light)

Code analysis (Static Analysis Light) can be conducted on programs that are created.

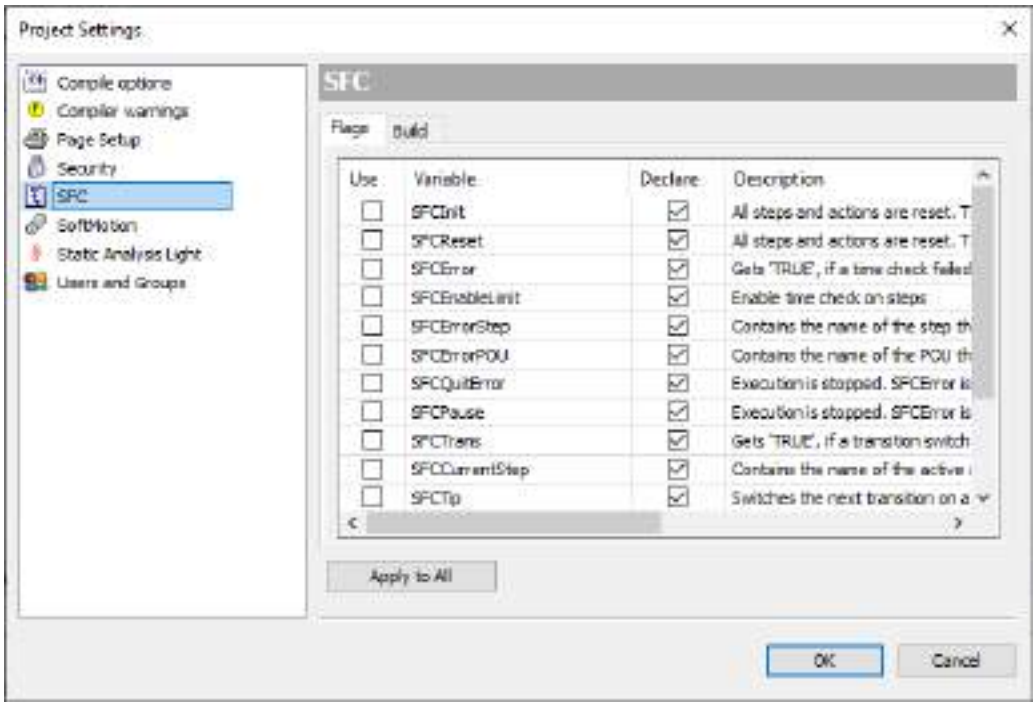
Code analysis can check for the following errors.

Error number	Description
SA0033	Unused variables
SA0028	Overlapping memory areas

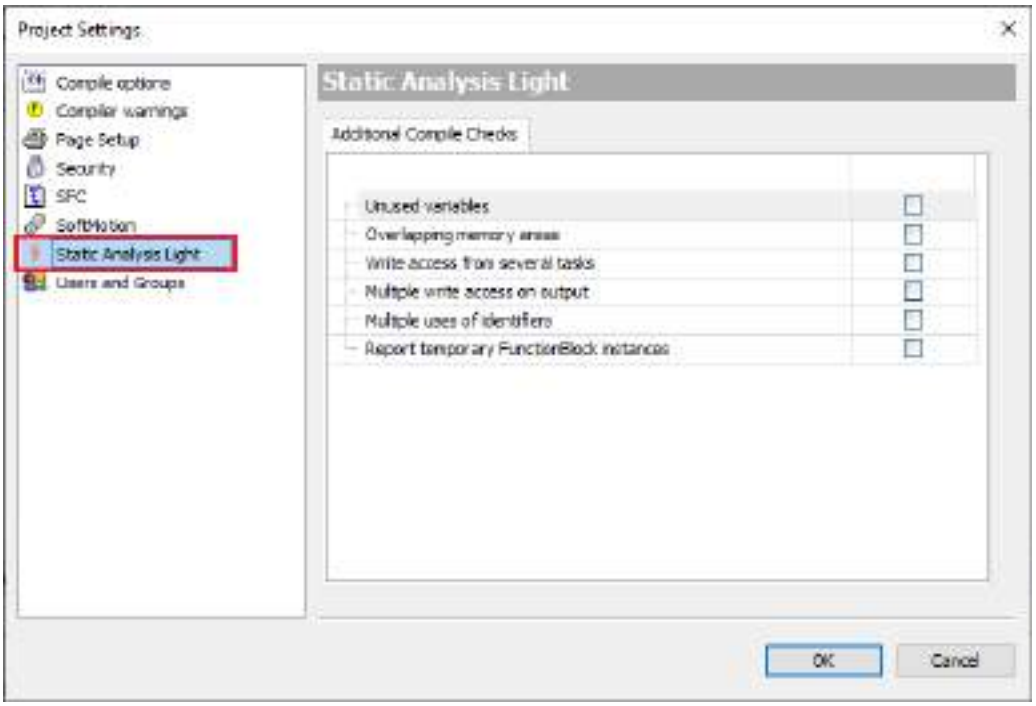
Error number	Description
SA0006	Write access from several tasks
SA0004	Multiple write accesses on output
SA0027	Multiple uses of identifiers
SA0167	Report of temporary function block instance

1 2 Procedure

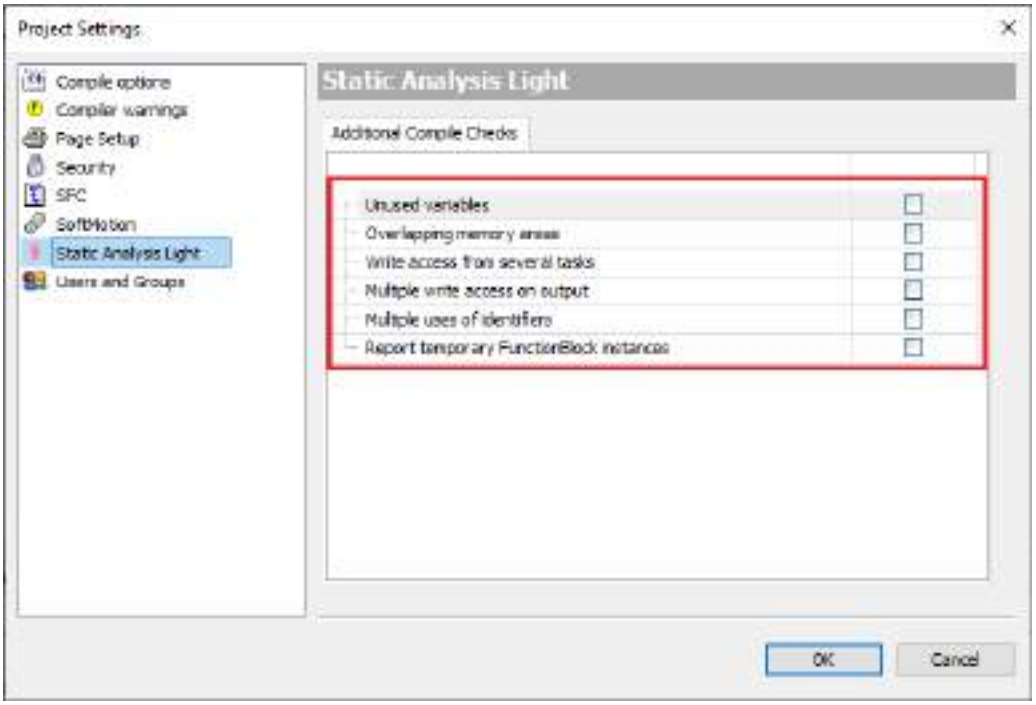
1. From the menu bar, select **Project**>**Project Settings**.
   The Project Settings dialog box will be displayed.



2. In the "Project Settings" dialog box, select the "Static Analysis Light" category.



3. Select the check boxes of the items to be checked.



4. Click the [OK] button.

Info.

- If the items to be checked are set beforehand, code analysis will be performed automatically during login.

8.9 Source Upload

Upload the source code from the GM1 controller to the PC and retrieve it, as below.

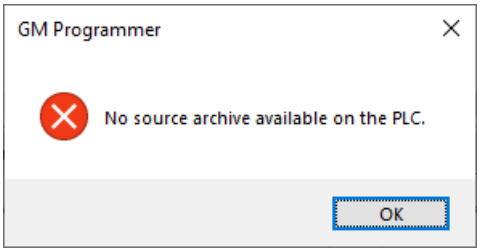
Procedure

1. From the menu bar, select **File**>**Source Upload**.
   The "Select Device" dialog box will be displayed.

2. Select the GM1 controller from which the source code is to be retrieved and click the [OK] button.
   The "Extract Project Archive" dialog box will be displayed.

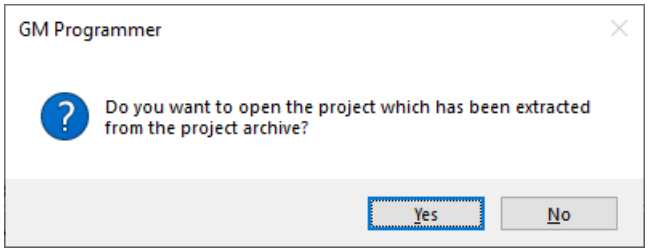
Info.

- If the source code does not exist in the selected GM1 controller, the following error message will be displayed.



3. Specify the upload destination folder and click the [Extract] button.

A confirmation dialog box will be displayed, asking whether to open the uploaded source code as a project file. Click the [Yes] button to open the uploaded source code as a project file.



4. In the folder specified as the upload destination, "Archive.prj" and "<project name>.project" will be created.

8.10 Commissioning

Commissioning can be conducted using GM Programmer.
To conduct commissioning, the GM1 controller must be connected in online config mode.

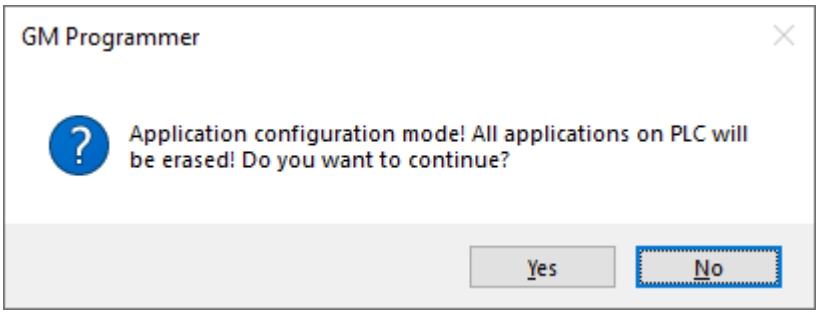
8.10.1 Online Config Mode

When the online config mode is selected, the servo amplifiers are set to be connected to the GM1 Controller.

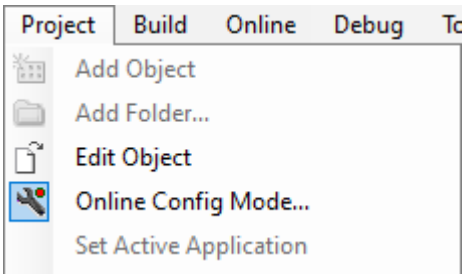
When using the online config mode, perform the setting as described in "8.4 Communication Setting" in advance.

Procedure

1. From the menu bar, select **Project>Online Config Mode**.
   A confirmation message will be displayed, asking whether to remove all applications.



2. Click [Yes].
   All applications will be removed from the GM1 controller, and the GM1 controller and servo amplifiers will be connected in online config mode.
   While online config mode is in progress, "Online Config Mode" in the menu bar remains selected.



Info.

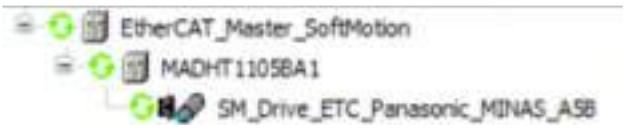
- To cancel the online config mode, select **Project>Online Config Mode** from the menu bar again.

8.10.2 Conducting Commissioning for Servo Amplifiers

While in online config mode, you can conduct commissioning for servo amplifiers.
There is no need to create a program for commissioning.
The following is an example of commissioning using the A5B-series servo amplifiers.

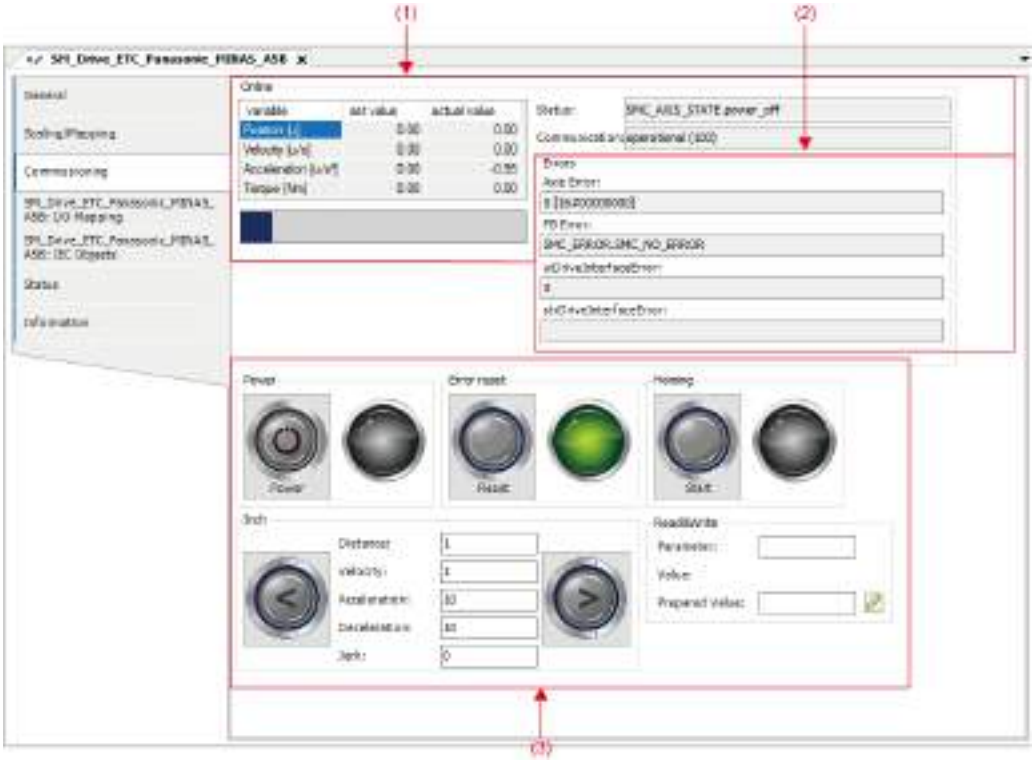
Procedure

1. Double-click the servo amplifier object in the navigator pane.



The "EtherCAT Axis Setting" dialog box will be displayed.

2. Click the "Commissioning" tab.
The Commissioning pane will be displayed.



No.	Group	Description
(1)	Status	Displays the running status of the servo amplifiers during commissioning.
(2)	Error	Displays errors that occurred during commissioning. Allows the user to clear errors.

No.	Group	Description
(3)	Operation	Allows the user to set commissioning parameters. Allows the user to execute commissioning.

3. Click an appropriate button in the Operation group to start commissioning.
Clicking an icon starts the corresponding commissioning.
To change home return parameters, use the "Program" tab.

4. For the servo amplifier status during commissioning, check the "Status" and "Error" groups.

4-1 To clear errors that are displayed, click the [Reset] button in the "Operation" group.
Clicking the [Reset] button clears all errors.

5. From the menu bar, select **Project>Online Config Mode**.
Online config mode will be canceled and commissioning will be terminated.

This completes commissioning for servo amplifiers.

Info.

- Even if communication with the servo amplifier is disrupted during "Inching" or "Home Return" operation, the servo amplifier will continue commissioning operation.
- If online config mode is canceled, commissioning will be terminated. To cancel the online config mode, select **Project>Online Config Mode** from the menu bar again.

(MEMO)

9 Debug

9.1 Running and Stopping the GM1 Controller 9-2
9.1.1 Running and Stopping the GM1 Controller 9-2
9.1.2 Single Cycle 9-3
9.2 Breakpoint 9-5
9.2.1 Setting a Breakpoint 9-5
9.2.2 Setting an Execution Point 9-6
9.2.3 Call Stack View 9-8
9.3 Debug Operations 9-10
9.3.1 Writing Values and Forcibly Changing Values 9-10
9.3.2 Watch 9-11
9.3.3 Flow Control 9-12
9.3.4 Operation Mode 9-14
9.4 Monitoring Function 9-15
9.5 Reset 9-18
9.5.1 Reset Warm, Reset Cold, and Reset Origin 9-19
9.5.2 Executing Device Reset from GM Programmer 9-19
9.5.3 Executing Device Reset from GM1 Controller 9-20
9.6 Checking the Status of GM1 Controller 9-22
9.6.1 Checking Logs 9-22
9.6.2 Checking the Status 9-23
9.6.3 Checking the System Data History 9-24
9.6.4 Task Monitoring 9-25
9.7 Device Trace Function 9-26
9.8 Checking the Performance of GM1 Controller 9-33
9.8.1 Checking Missing EtherCAT Packets 9-33
9.8.2 Performance Check Based on Device Trace 9-34
9.9 Error Notification Function 9-36
9.9.1 Overview of Errors 9-36
9.9.2 Checking and Clearing Errors Using GM Programmer 9-36
9.9.3 Obtaining Error Information Using User Programs 9-37
9.9.4 Error Recovery Processing 9-40
9.9.5 Error Code List 9-42

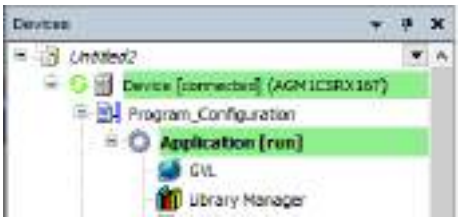
9.1 Running and Stopping the GM1 Controller

This section explains how to run and stop the GM1 controller and how to run a single cycle operation that is executed in units of a cycle.

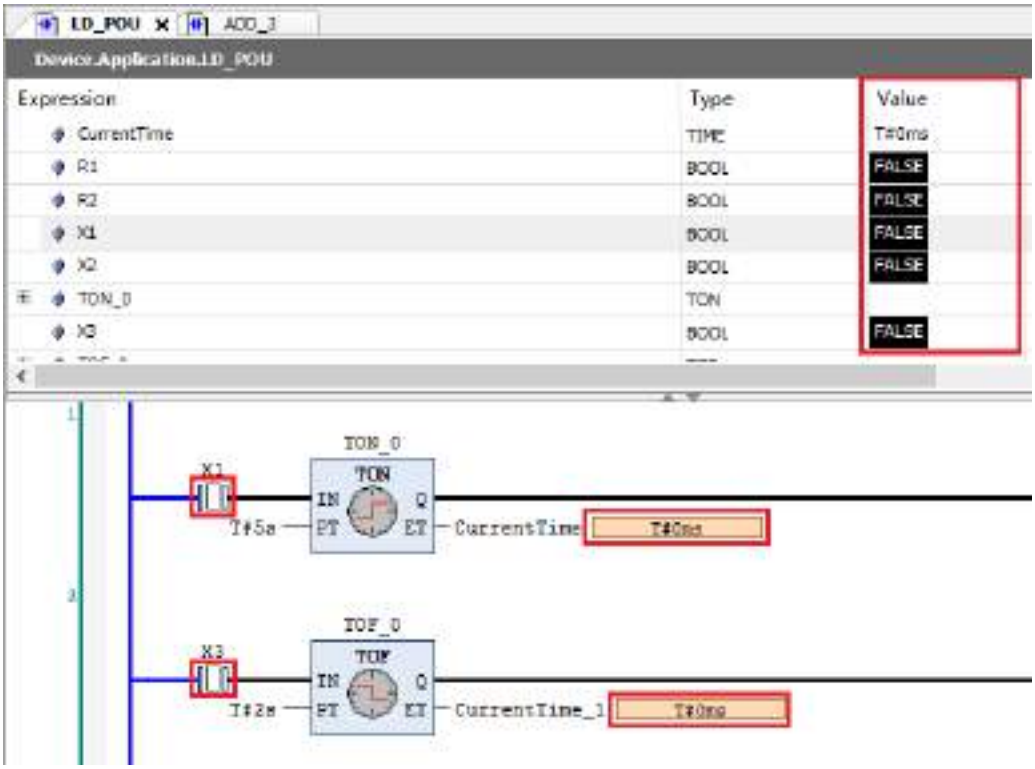
9.1.1 Running and Stopping the GM1 Controller

Procedure

1. After logging in, from the menu bar, select **Debug**>**Start** or press the <F5> key.
   The applications downloaded to the GM1 controller will start running.



2. From the menu bar, select **Debug**>**Stop**, or press the <Shift> key + <F8> key simultaneously.
   The applications will be stopped.
   During debug operation, you can check the current value of each variable in the declaration section and implementation section.



Info.

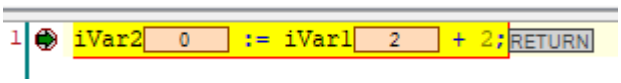
- You can start the GM1 controller by clicking ▶ on the toolbar and also stop it by clicking ■.
- You can select binary, decimal, or hexadecimal as the display format of the variable values to be displayed. From the menu bar, select **Debug**>**Display Mode** and select a display format from those shown.
- If you select confirmed online mode, a confirmation message will be displayed before you start or stop the GM1 controller. For confirmed online mode, refer to "8.7.4 Confirmed Online Mode".

9.1.2 Single Cycle

You can execute the application in simulation mode in a single cycle to check whether a created program is executed as intended.

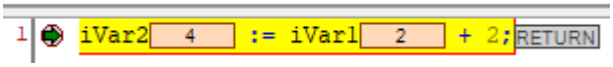
Procedure

1. After logging in, open the POU.



2. From the menu bar, select **Debug**>**Single Cycle** or press <Ctrl+F5>.

The opened POU will enter a state in which it has been executed in a single cycle.



9.2 Breakpoint

By setting a breakpoint in a particular position in a program, you can forcibly stop executing the program and check the variable values.

All programming languages support breakpoints.

9.2.1 Setting a Breakpoint

Procedure

1. Select a position where you want to set a breakpoint. From the menu bar, select **Debug**>**Toggle Breakpoint** or press the <F9> key.

   The breakpoint will be enabled.

   Example: Setting a breakpoint in line 12 in an ST program

   If operation is started, the operation will be stopped when the position of the set breakpoint is reached.

   In the stopped state, the following debug operations can be executed.

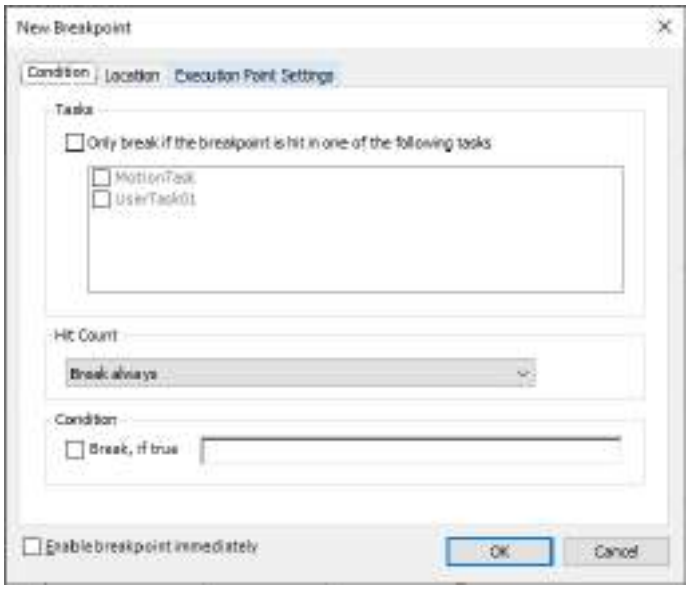
   From the menu bar, select "Debug" and then one of the following menu items.

Menu item	Shortcut key	Icon on the toolbar
Step Over	<F10>	
Step Into	<F8>	
Step Out	<Shift> + <F10>	
Run to Cursor	None	
Set next Statement	None	
Show next Statement	None	

   To cancel set breakpoints, from the menu bar, select **Debug**>**Toggle Breakpoint** or click the <F9> key again.

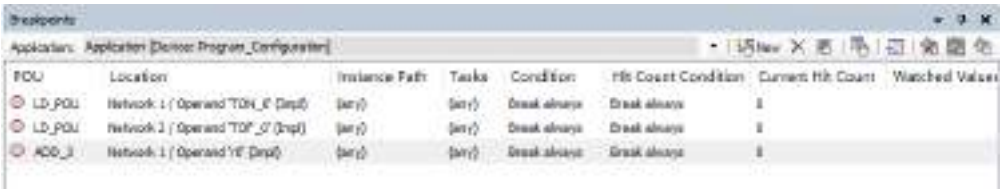
Info.

- You can specify conditions under which operation is stopped when a breakpoint is reached. From the menu bar, select **Debug>New Breakpoint**. The "New Breakpoint" dialog box will be displayed. Select the "Condition" tab and specify conditions under which operation is stopped when a breakpoint is reached.



- The Breakpoint view allows the user to check a list of set breakpoints. You can check breakpoint positions, break conditions, and the hit count was reached. You can also add, delete, enable, and disable breakpoints.

  To display the Breakpoint view, from the menu bar, select **View>Breakpoint**.



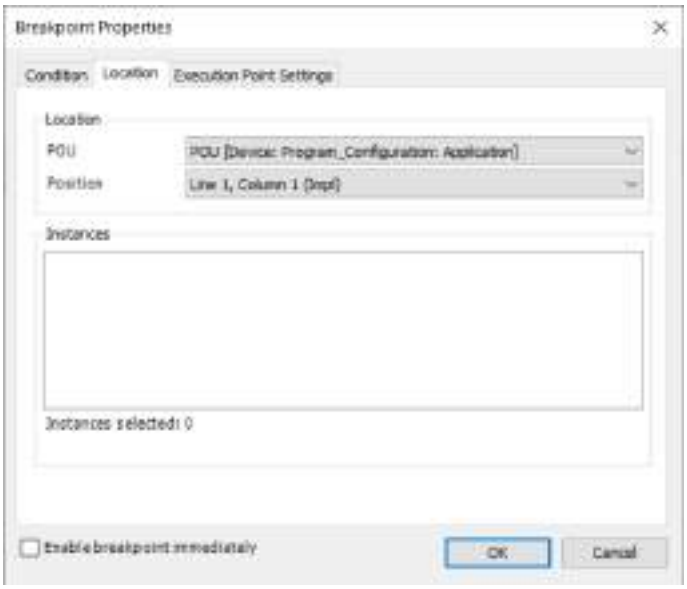
9.2.2 Setting an Execution Point

If an execution point is set, when the position of the execution point is reached, processing that is specified beforehand can be executed and the execution result can be output to the log of the GM1 controller. The application does not stop at the position where an execution point is set.

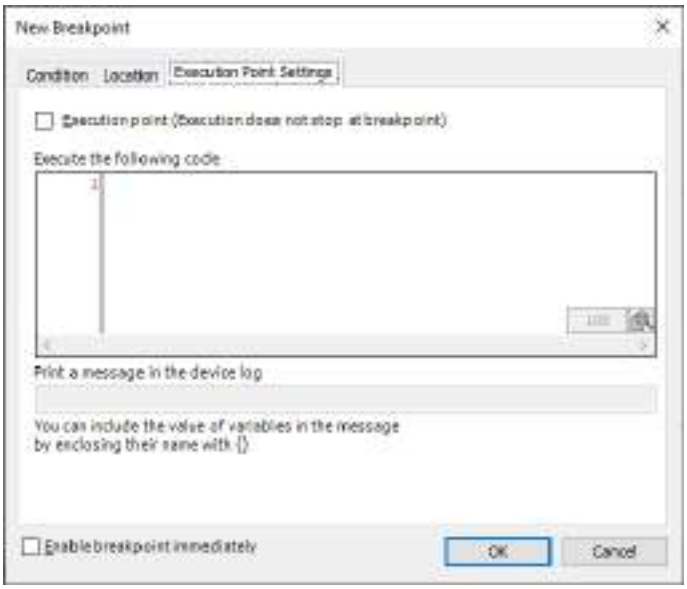
Procedure

1. Select a position where you want to set an execution point. From the menu bar, select **Debug>New Breakpoint**.

   The "New Breakpoint" dialog box will be displayed.

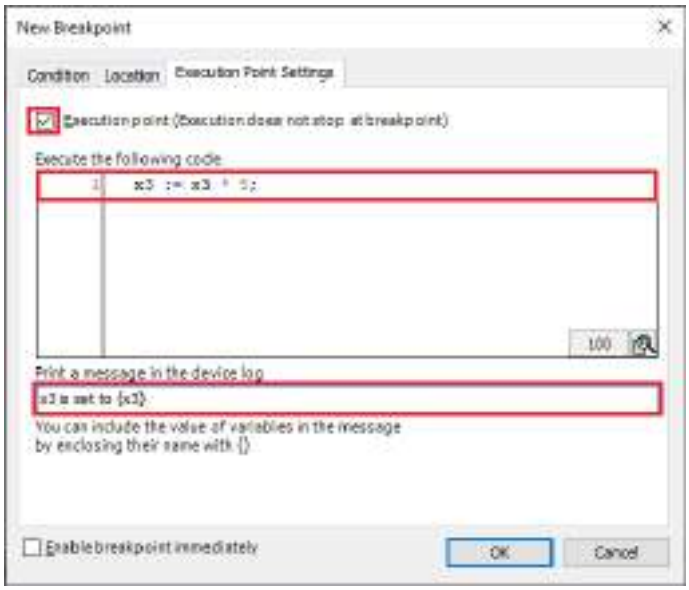


2. Click the "Execution Point Settings" tab.
The Execution Point Settings window will be displayed.



3. Select the "Execution point" check box and enter the code to be executed at the execution point and the message to be output to the log.
In the "Execute the following code" area, enter executable code in structure text format. In the "Print a message in the device log" field, enter the message to be output to the log.

Example: Multiplying the value of "x3" by 5 and outputting the value to the log



4. Click the [OK] button.

The execution point will be set. When the execution point is enabled, ● appears at the execution point.

Info.

- To output a message to the log when an execution point is reached, from the menu bar, select **Project**>**Project Settings**. In the "Project Settings" dialog box, select the "Compile options" category. Change the setting in **Setting**>**Enable logging in breakpoints** to "Enabled".

9.2.3 Call Stack View

In the Call Stack view, you can check a stop position when operation is stopped due to a breakpoint or for some other reason. If the position is called from another block, the position of the block can also be checked.

Procedure

1. From the menu bar, select **View**>**Call Stack**.

The Call Stack view will be displayed.

2. Set a breakpoint and stop the application.
   The stop position and the POU calling the POU at the stop position will be displayed.

Example: When operation stops at line 1 of function "ADD_3" and "ST_POU" calls "ADD_3"

9.3 Debug Operations

This section explains how to perform debug operations such as writing values and watch.

9.3.1 Writing Values and Forcibly Changing Values

Variable values for the GM1 controller can be changed. There are two methods for changing values: Writing values and forcibly changing values.

Writing values: Sets a value (to be changed later) only once. This value can then be changed by the program.

Force Values: Sets a value to be changed in every cycle and maintains the value.

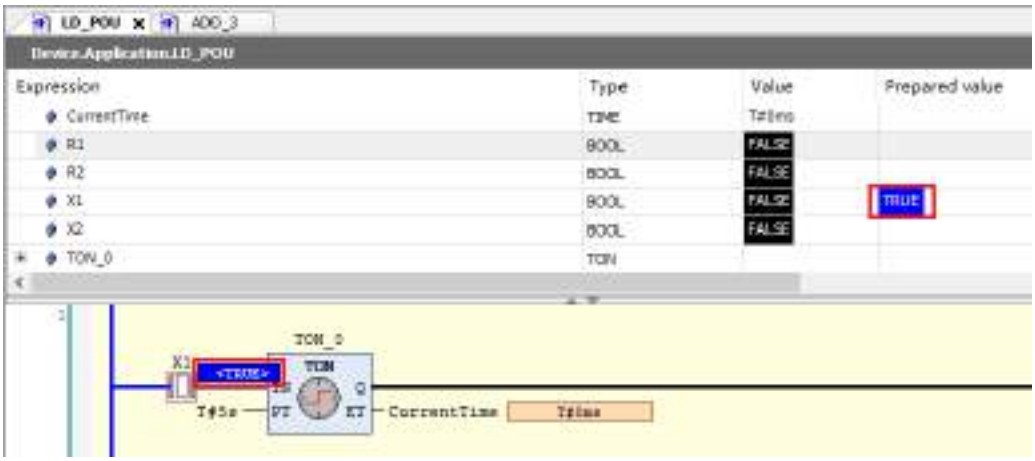
For example, use the following procedure to change the value of Boolean variable "x1" from FALSE to TRUE by writing the value.

Procedure

1. In the implementation section, double-click the element whose value is to be changed.

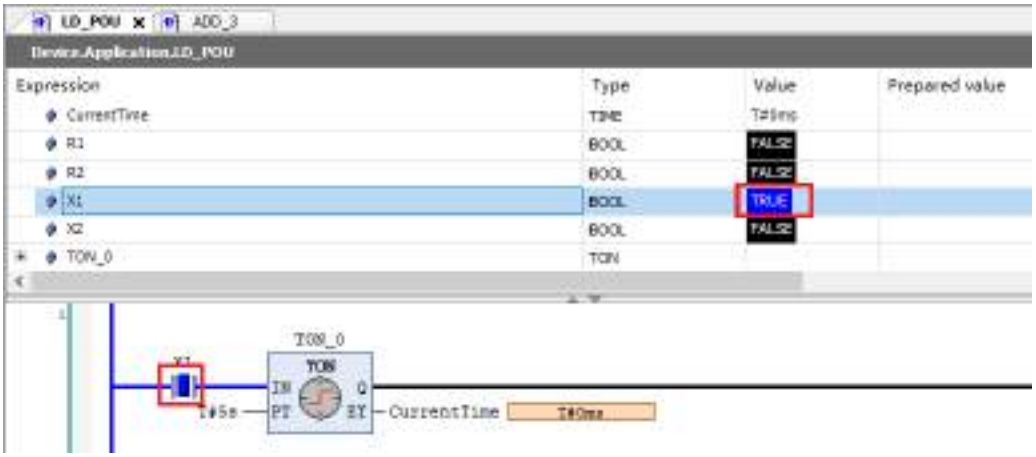
   A new value will be preset.

   You can also preset a value by clicking a cell in the "Preset value" column of the declaration section.

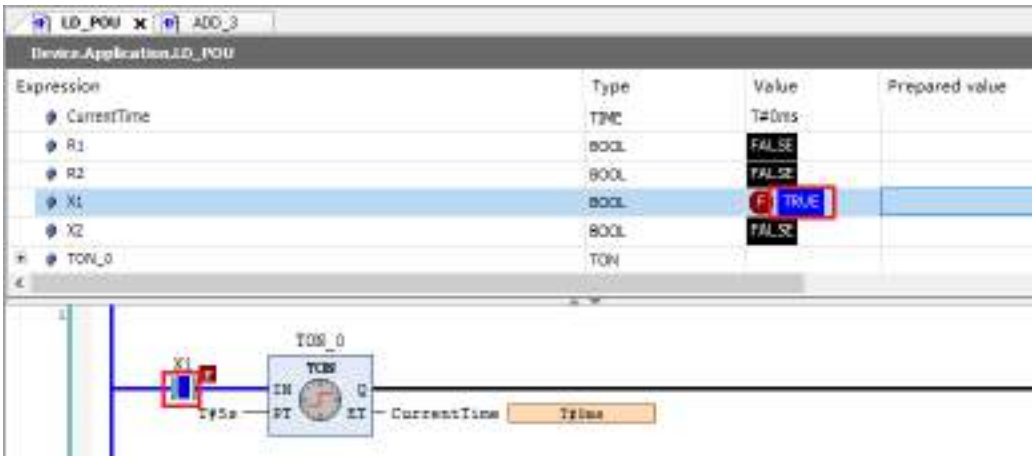


2. From the menu bar, select **Debug**>**Write Values**, or press the <Ctrl> key + <F7> key simultaneously.

   The preset value will be written.



From the menu bar, select **Debug**>**Force Values**, or press the <F7> key. The variabsle value will be forcibly changed. F appears in front of a variable whose value has been forcibly changed, and then the value will not be updated by the program.



From the menu bar, select **Debug**>**Unforce Values**, or press the <Alt> key + <F7> key simultaneously. Forced value change will be canceled.

9.3.2 Watch

By registering variables in the watch view, you can perform variable value management such as checking or changing variable values.

You can use up to four watch views (Watch 1 to Watch 4).

For example, use the following procedure to register variable "x1" in watch view "Watch 1".

Procedure

1. From the menu bar, select **View**>**Watch**>**Watch 1**.
   Watch view "Watch 1" will be displayed.

2. Drag the variable "x1" element in the implementation section and drop it in the watch view.
   Variable "x1" will be registered in the watch view.
   You can also register the variable in the watch view by dragging it from the declaration section and dropping it in the watch view.

This completes the procedure for registering the variable in the watch view. You can check variable values in the Value column.

Info.

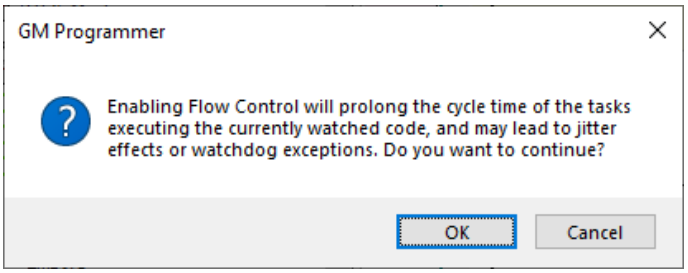
- Variables whose values have been forcibly changed are automatically registered in the "Watch all Forces" view.
  From the menu bar, select **View**>**Watch**>**Watch all Forces**.
- If an execution point has been set, the timing of display in the watch view can be set to the point in time when the execution point is reached. In the "Execution point" column, select the execution point that has been set. For details on how to set execution points, refer to "9.2.2 Setting an Execution Point".

9.3.3 Flow Control

Flow control enables monitoring to be performed by using different colors in positions where the program is executed and in positions where the program is not executed.
Flow control can be used in LD programs, ST programs, and FBD programs.

Procedure

1. After logging in, from the menu bar, select **Debug**>**Toggle Flow Control Mode**.
   The flow control notification dialog box will be displayed.

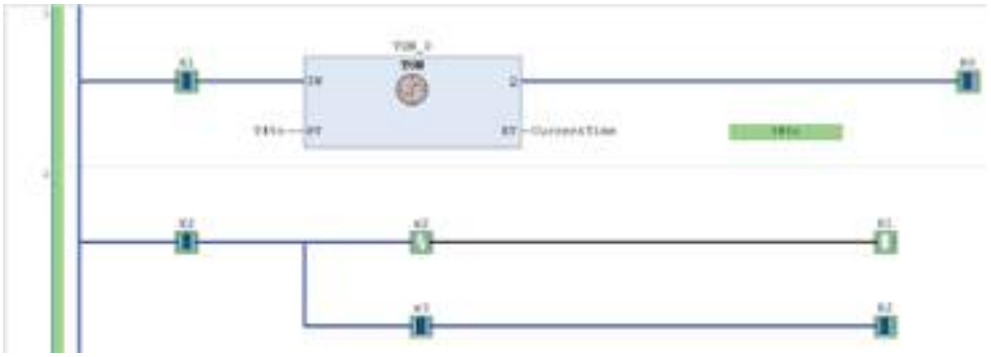


2. Click the [OK] button.

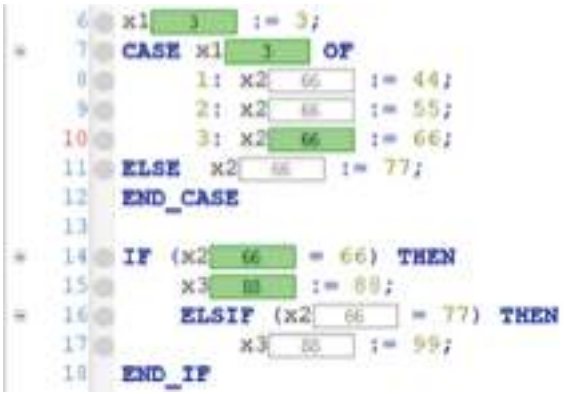
The display will be switched to flow control display.

The positions where the program was executed are displayed in green and the positions where the program was not executed are displayed in white.

Example: Flow control display for LD programs



Example: Flow control display for ST programs



Info.

- By using confirmed online mode, you can have a confirmation message dialog box displayed before you execute flow control. For confirmed online mode, refer to "8.7.4 Confirmed Online Mode".

9.3.4 Operation Mode

Using the operation mode function makes it possible to prevent some debug operations from being executed. This can prevent incorrect operation of the GM1 controller when it is operated accidentally.

The current operation mode is displayed as an icon on the status bar.

Debug ( )

This mode has no restriction.

Locked ( )

Start / stop, new breakpoint setting, and forcing values cannot be executed.

Single cycle operation, writing variables, and unforcing values can be executed.

Operational ( )

Only writing variables can be executed. Start / stop, new breakpoint setting, forced variable change, single cycle, and canceling forced variable change cannot be executed. To use this mode, the following conditions must be satisfied.

- Application is running
- There is no active breakpoint
- There is no variable whose value has been forcibly changed
- The application created in GM Programmer matches the boot application in the GM1 controller

Procedure

1. After logging in, from the menu bar, select **Online**>**Operation Mode**>**Locked**.
   The operation mode will be changed from Debug mode to Locked mode.

9.4 Monitoring Function

The monitoring function allows the user to check the variables in the program and the current values of the device parameters in real time while being logged in the GM1 controller.

■ Monitoring variables in the declaration editor

The variables declared in the declaration editor can be monitored.

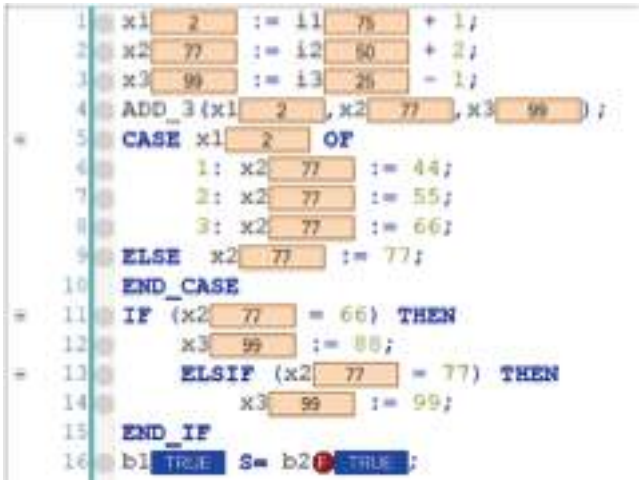
F appears in front of forcibly changed values. For details on forced value change, refer to "9.3.1 Writing Values and Forcibly Changing Values".

■ Monitoring variables in the implementation section of the program

- Variables can be monitored in the implementation section of the program (inline monitoring).
- The current value is displayed on the right side of each variable, such as T#1s805ms or 2.
- Contacts, coils, and connection lines are displayed in blue when the current value is TRUE.

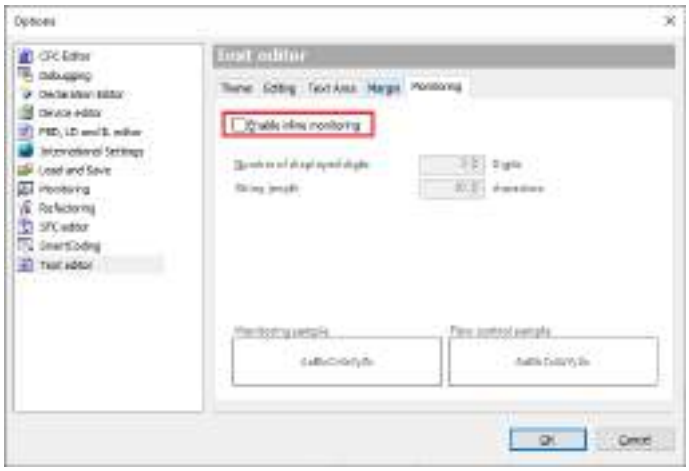
<Inline monitoring for LD programs>

<Inline monitoring for ST programs>



Info.

- Inline monitoring can be disabled. Open the Options window (by selecting **Tools**>**Options**), select the "Text editor" category and then "Monitoring" tab, and clear the "Enable inline monitoring" check box.



■ Monitoring variables in the watch view

By registering variables in the watch view, you can monitor the variables.

You can use up to four watch views, as well as a dedicated view where variables whose values are forcibly changed are automatically registered.

For details on how to register variables in the watch view, refer to "9.3.2 Watch".


Watch 1
Expression
Application
Type
Value
Prepared value
Execution point
Address
LD_PDU.X1
Device Application
BOOL
TRUE
Cyclic Monitoring
LD_PDU.Q
Device Application
BOOL
TRUE
Cyclic Monitoring
LD_PDU.Cur
Device Application
TIME
T#1ms
Cyclic Monitoring
Current values

9.5 Reset

Reset operation resets the active applications and initializes the variables and settings.

Reset is divided into the following four types and variables and settings that are initialized differ according to the reset type.

Reset Warm

Initializes variables other than the RETAIN and PERSISTENT variables.

Reset Cold

Initializes variables other than the PERSISTENT variable.

Reset Origin

Initializes all variables. Active applications are deleted from the GM1 controller.

Device Reset

Initializes all variables and device user management information. Applications and source code are deleted from the GM1 controller.

- The following table shows the items that are initialized by reset or other tool operation or controller operation.

○: Retained

×: Initialized

Update: Updated

Operation		Variables other than "RETAIN / PERSISTENT"	RETAIN variable (RETAIN)	PERSISTENT Variables (PERSISTENT)	(Boot) Application	User management	Source file	IP address	RTC time zone
Tool operation	Stop	○	○	○	○	○	○	○	○
	Reset warm	×	○	○	○	○	○	○	○
	Reset Cold	×	×	○	○	○	○	○	○
	Download	×	×	○	Update	○	Update	Update	○
	Online change	○	○	○	Update	○	Update	○	○
	Power cycling	×	×	×	×	○	○	○	○
	Reset Origin	×	×	×	×	×	×	○	○
Controller operation	Resetting "Device"	×	○	○	○	○	○	○	○
	Resetting "Device" by means of hard switching	×	×	×	×	×	×	○	○

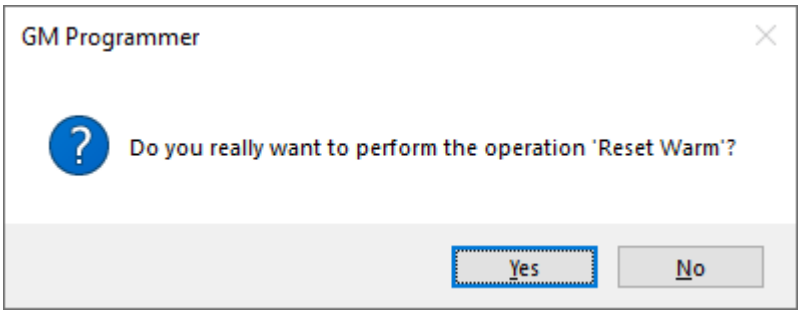
9.5.1 Reset Warm, Reset Cold, and Reset Origin

Execute Warm Reset, Cold Reset, and Reset Origin by selecting them from "Online" on the menu bar. This section explains the execution procedure, using Warm Reset as an example.

Procedure

1. From the menu bar, select **Online**>**Reset Warm**.

   Example: "Reset Warm" execution procedure



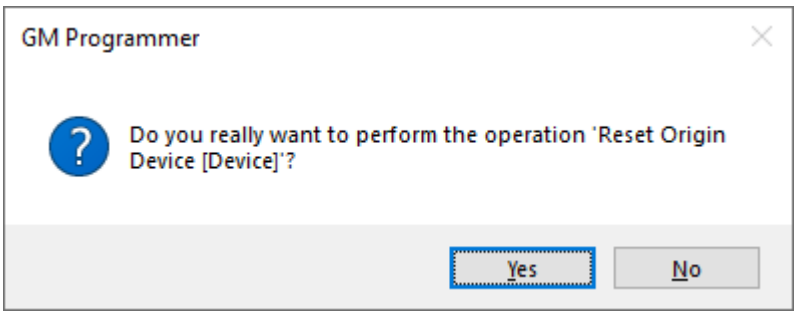
2. Click the [Yes] button.
   Reset warm will be executed.

9.5.2 Executing Device Reset from GM Programmer

Device reset can be executed from the GM1 controller as well as from GM Programmer.
To execute device reset from GM Programmer, right-click in the navigator pane and execute device reset from the context-sensitive menu that is displayed.

Procedure

1. Right-click the [Device] object in the navigator pane and then select "Reset Origin" from the context-sensitive menu that is displayed.
   A confirmation message will be displayed, asking whether to execute device reset.



2. Click the [Yes] button.
   Device reset will be executed. When device reset is executed, you are logged out.

Info.

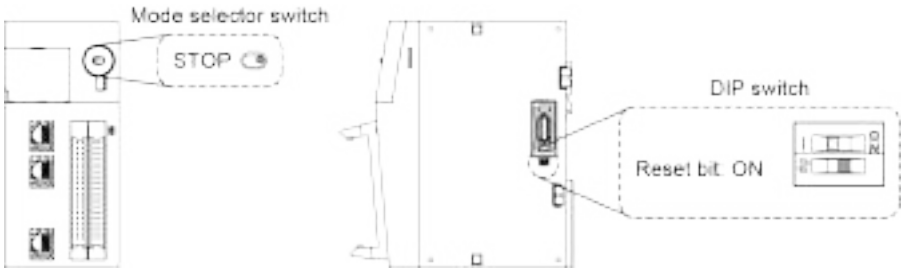
- If you right-click the [Application] object in the navigator pane and select "Delete application from device", the selected application will be removed.

9.5.3 Executing Device Reset from GM1 Controller

Device reset can be executed from the GM1 controller.

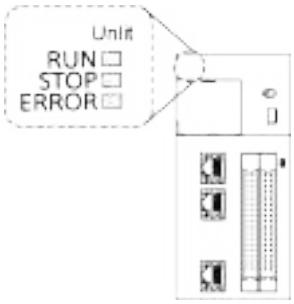
Procedure

1. Check that the power is OFF, set the mode selector switch to STOP, and set the reset bit of the DIP switch to ON.

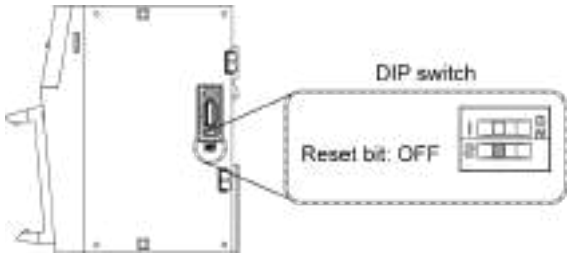


2. Turn the power ON.
   Device reset will be executed.

3. When the "RUN", "STOP", and "ERROR" LEDs go out, device reset is completed.



After device reset is completed, turn the power OFF and set the reset bit of the DIP switch to OFF.


DIP switch
Reset bit: OFF

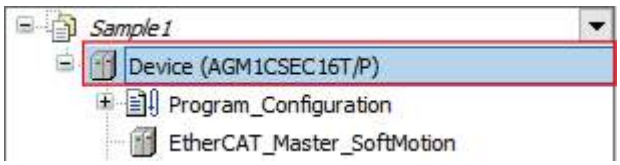
9.6 Checking the Status of GM1 Controller

9.6.1 Checking Logs

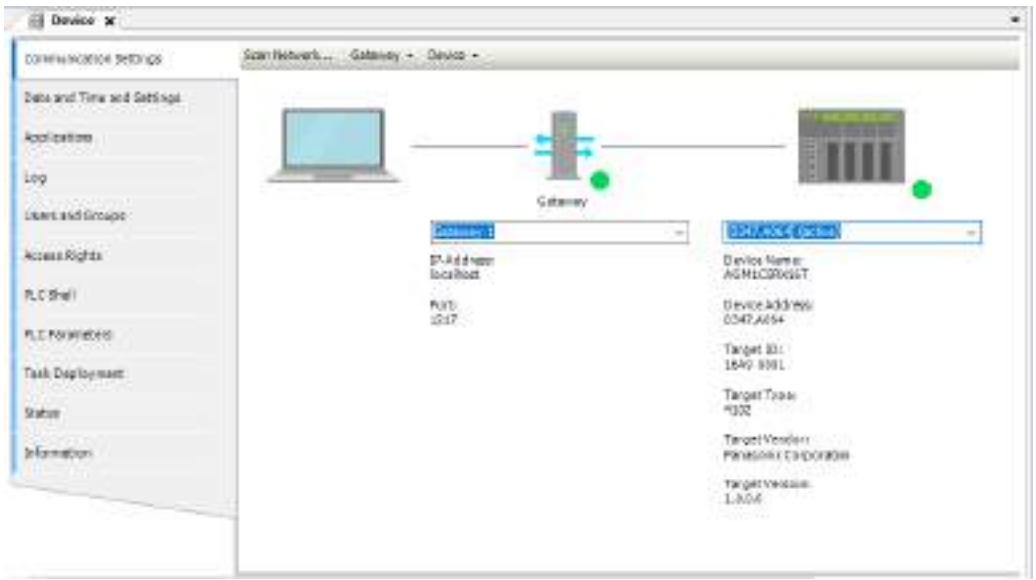
You can check logs of GM1 controller startup, shutdown, application download, and other events.

Procedure

1. Connect the PC where GM Programmer is installed and the GM1 controller.
   For details, refer to "8.5 Connecting to the GM1 Controller".
   Double-click the [Device] object in the navigator pane.



   The Device setting window will be displayed.



2. Click the [Log] tab.
   The log window will be displayed.

3. Click the ↻ icon.
   The log will be displayed.

Info.

- The displayed log can be exported (by clicking ↑) or imported (by clicking ↓) as an XML file.

9.6.2 Checking the Status

You can check only one error item that has the highest severity level among all errors currently occurring in the GM1 controller.

Procedure

1. Connect the PC where GM Programmer is installed and the GM1 controller.
   For details, refer to "8.5 Connecting to the GM1 Controller".
2. From the menu bar, select **Online**>**Status**.
   The "Status" dialog box will be displayed. You can check only one error item that has the highest severity level among all errors currently occurring in the GM1 controller.

3. Click the "Close" button.
   The "Status" dialog box will be closed.

Info.

Error types

The following table shows the types and recovery methods of errors that are displayed.
Clicking the [Error Clear] button deletes the target status item.

Error type	Recovery method
System error (power cycle)	Turn the GM1 controller OFF and then ON.
System error (Reinitialize)	Reinitialize the system. (Applications will be downloaded without executing a reset and the mode will be set to RUN.)
System error (Stop operation)	Click the [Error Clear] button.
System error (Continue operation)	Click the [Error Clear] button.
Incorporation / expansion error	Reinitialize the system.
Unit error	Click the [Error Clear] button or reinitialize the system.

Refer to "9.9.2 Checking and Clearing Errors Using GM Programmer".

9.6.3 Checking the System Data History

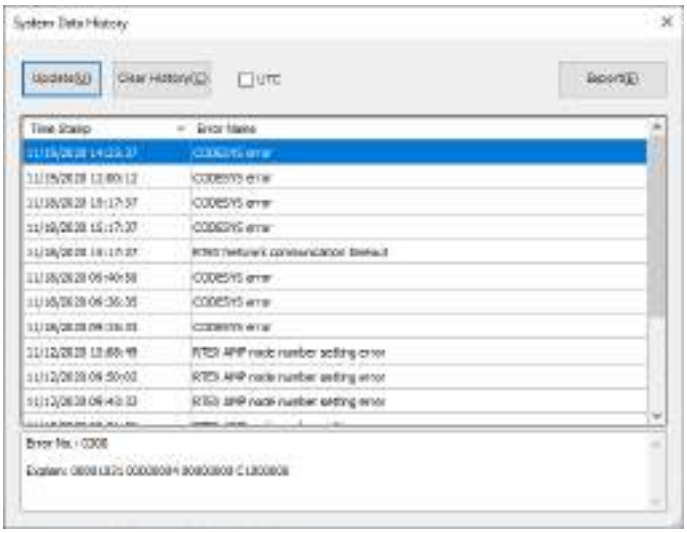
You can check histories of errors that have occurred up until now.

Procedure

1. Connect the PC where GM Programmer is installed and the GM1 controller.
   For details, refer to "8.5 Connecting to the GM1 Controller".

2. From the menu bar, select **Online**>**System Data History**.
   The "System Data History" dialog box will be displayed. You can check errors that have occurred up until now.
   Clicking the [Update] button collects system data histories again.
   Clicking the [Export] button outputs the system data histories displayed in the dialog box to a ".csv" file.

Clicking the [Clear History] button deletes the system data histories that are displayed.



3. Click the "×" button.
   The "System Data History" dialog box will be closed.

Info.

For details, refer to "9.9.2 Checking and Clearing Errors Using GM Programmer".

9.6.4 Task Monitoring

You can check the task status, the number of cycles, cycle time, and jitter while being logged in the GM1 controller.

Double-click the "Task Configuration" object in the navigator pane and select the "Monitor" tab.

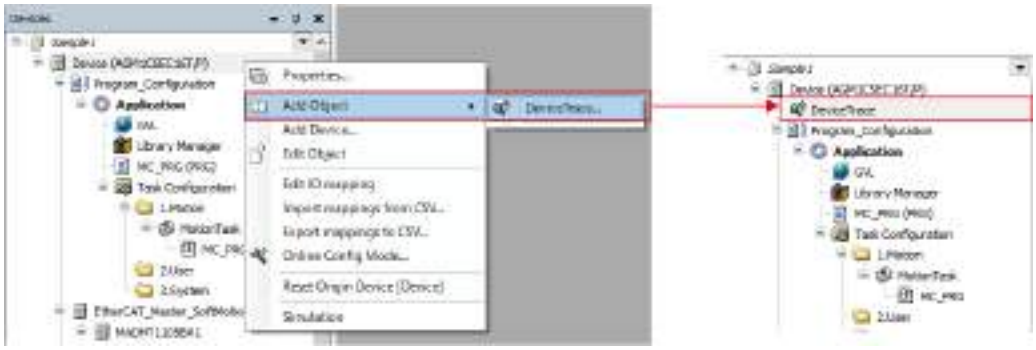
9.7 Device Trace Function

The Device Trace function of GM Programmer can monitor the CPU load factor of the GM1 controller.

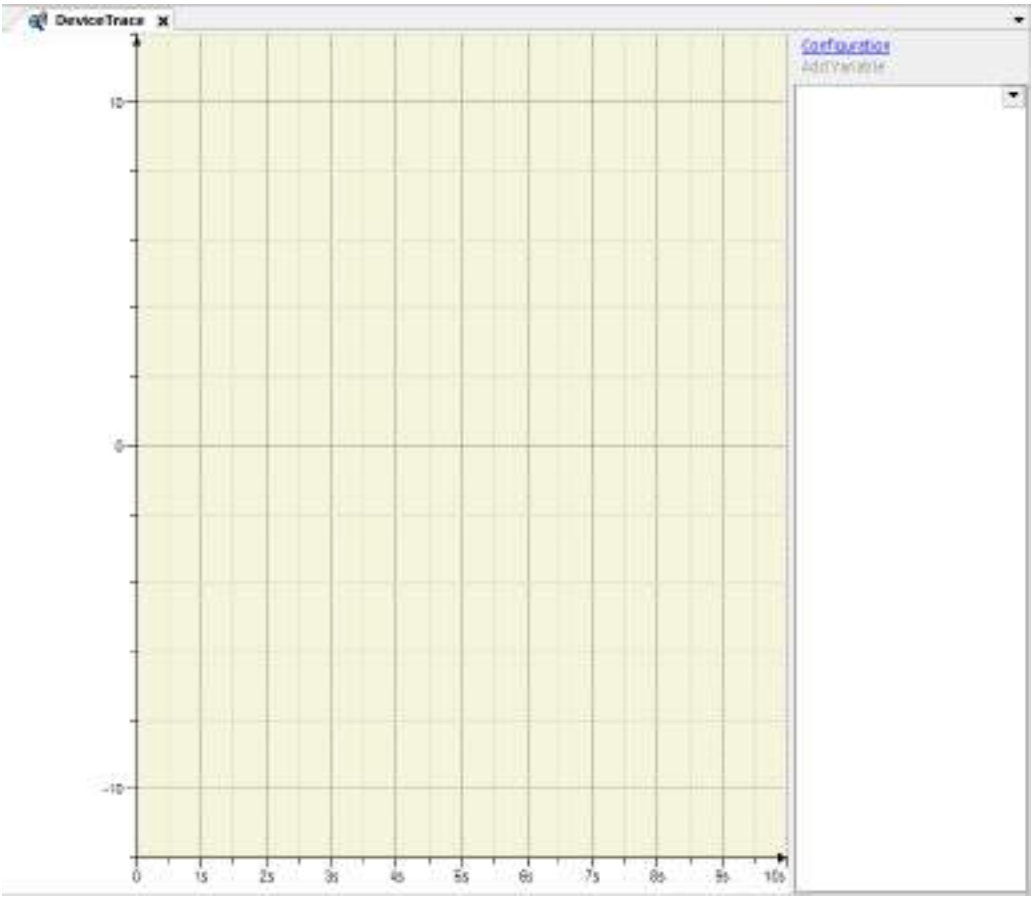
This function allows you to check whether the entire task falls within the appropriate CPU time range.

1 2 Procedure

1. Right-click "Device" and select **Add Object>DeviceTrace...**.



2. You will be logged in to the device.
3. Double-click the "DeviceTrace" object that has been added. The "DeviceTrace" window will be displayed.

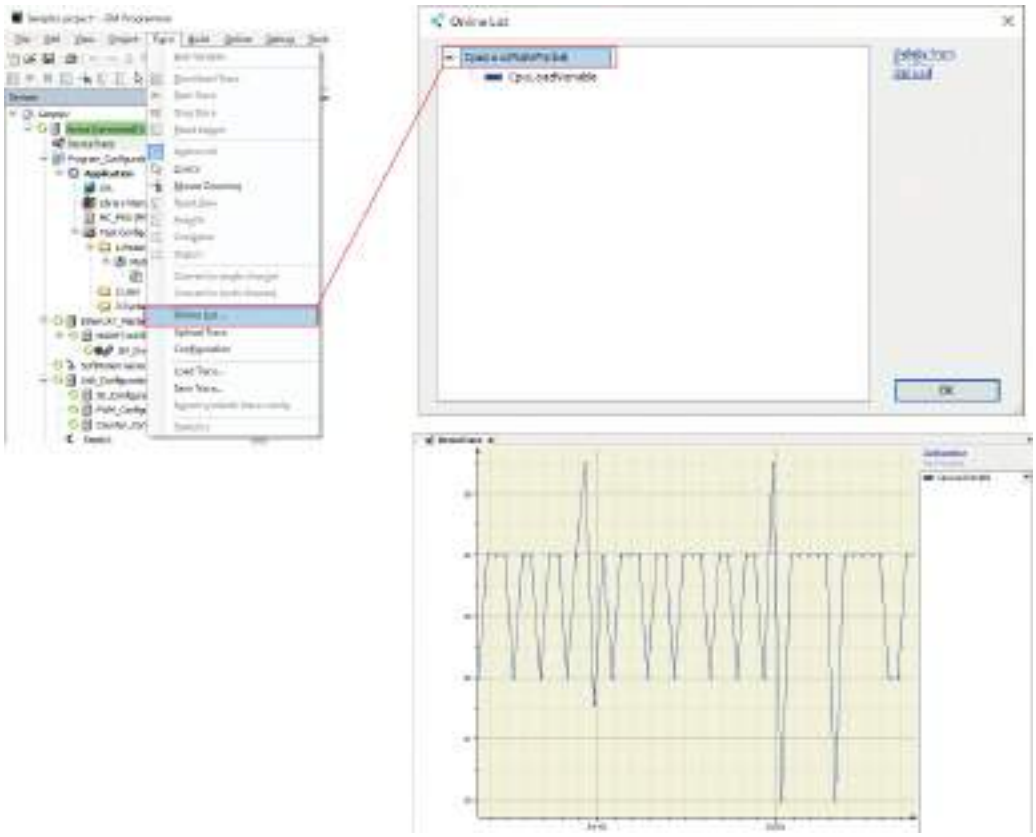


4. From the menu bar, select **Trace**>**Online List...**.

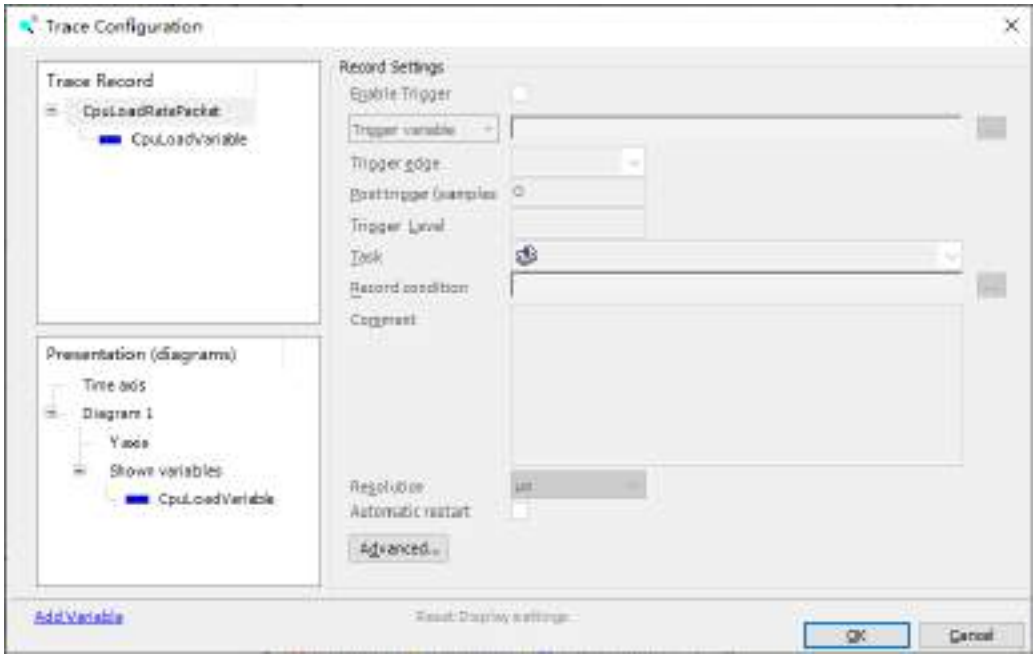
5. The "Online List" window will be displayed. With "CpuLoadRatePacket" selected, click "Upload".

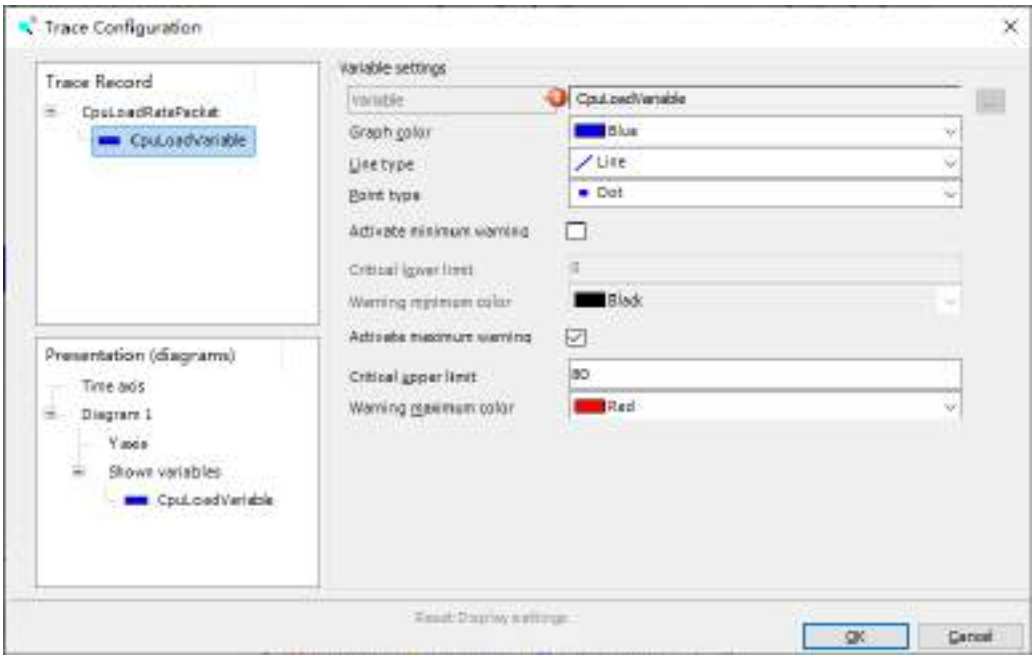
   Plotting the CPU load factor will start.

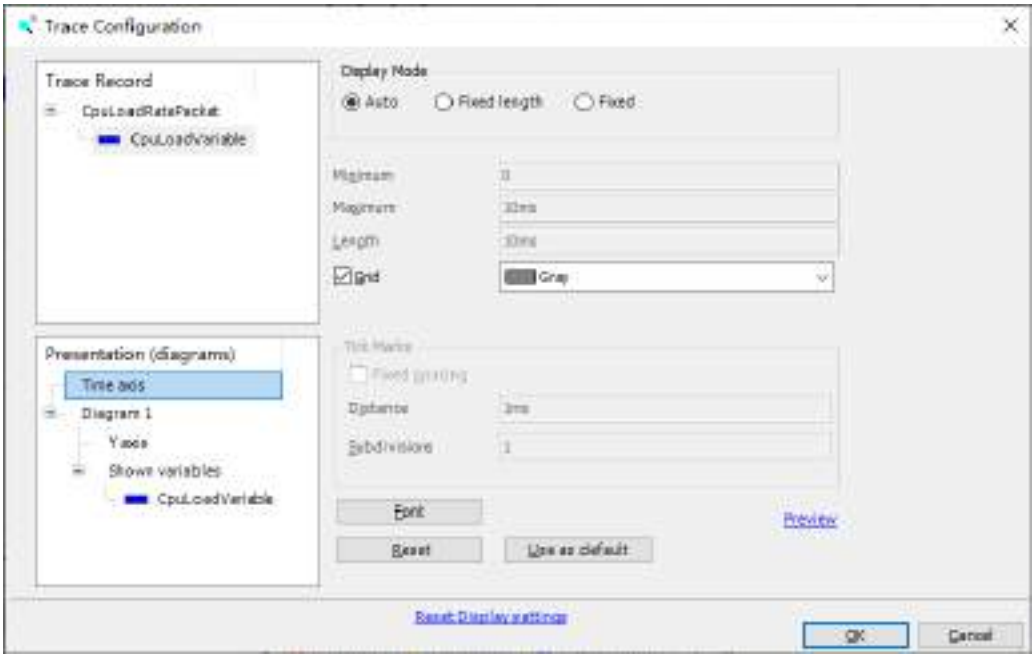
   Note: As the Online List window is still open, click the [OK] button to close the window.

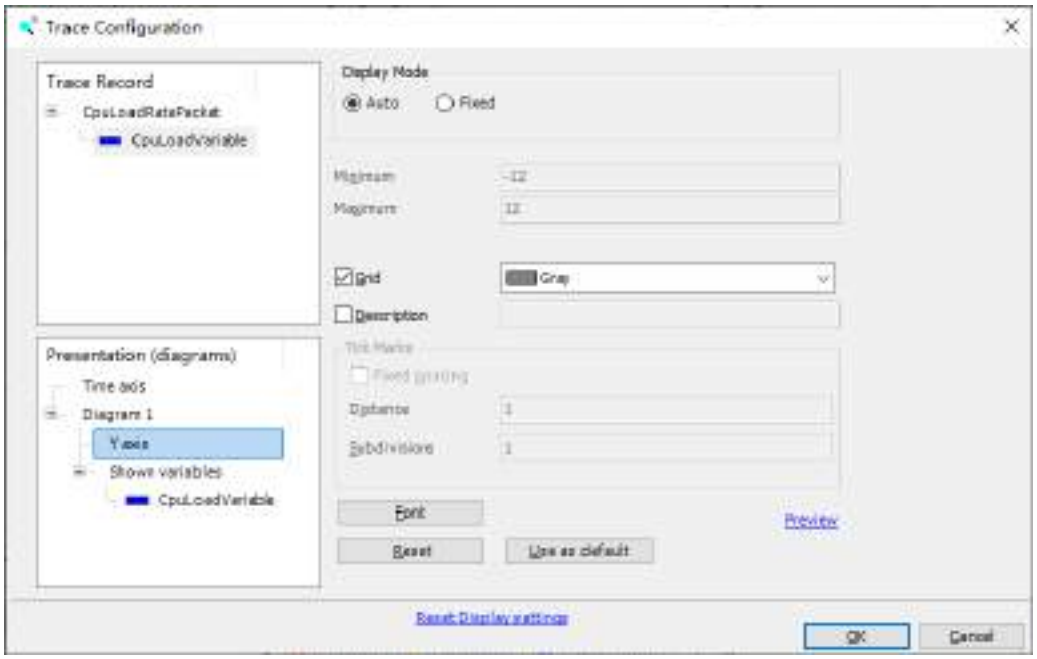


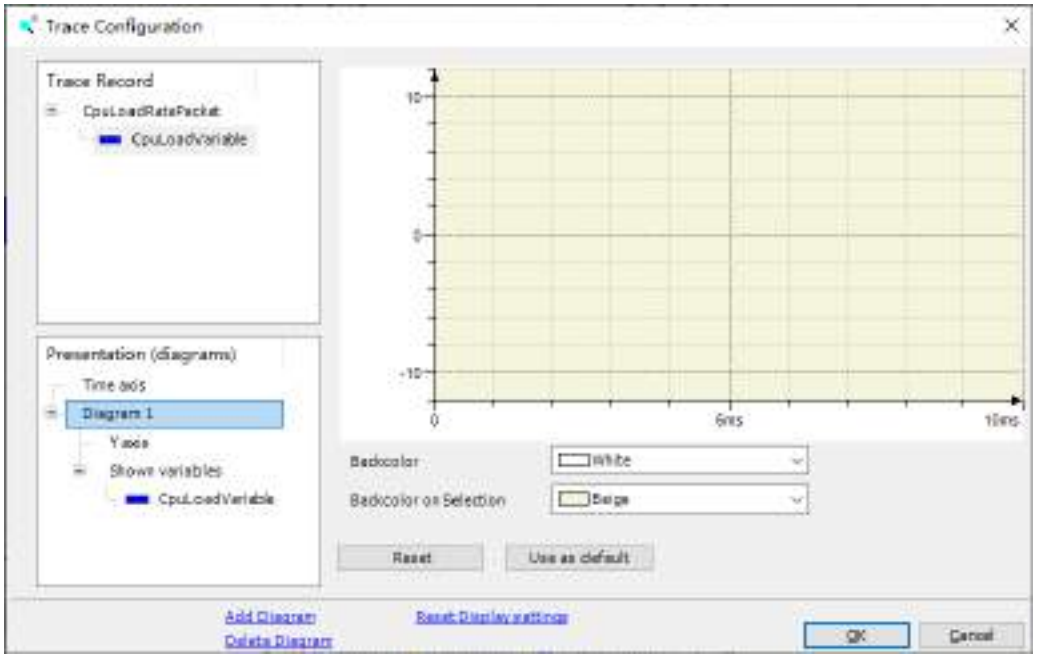
6. To change the graph display settings, click "Configuration" in the top right corner of the "DeviceTrace" window to open the "Trace Configuration" window.

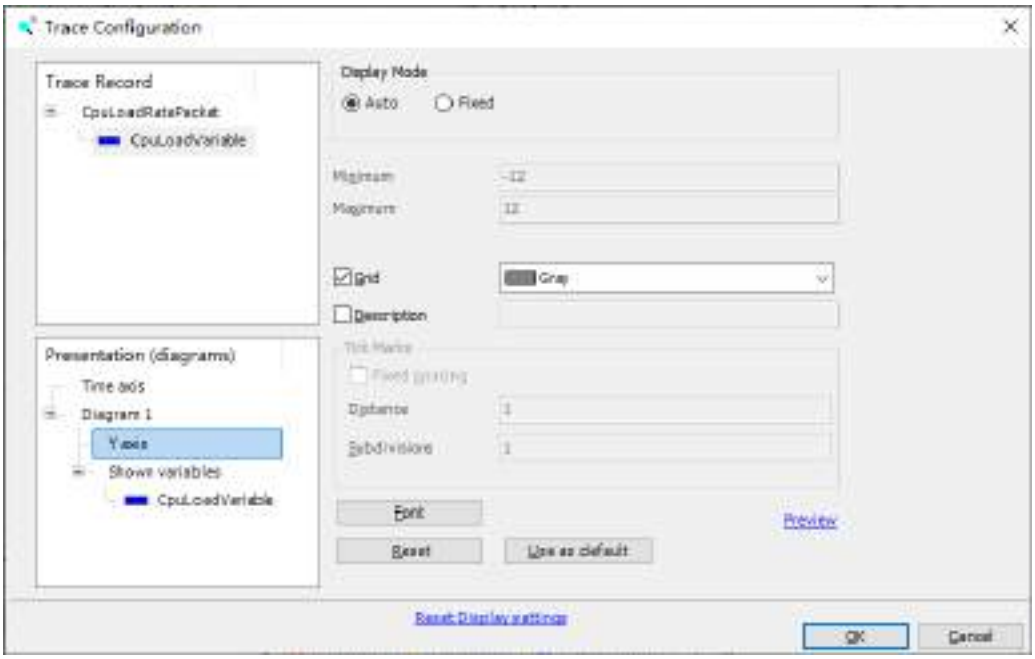



Trace Configuration
Trace Record
CpuLoadRatePacket
CpuLoadVariable
Presentation (diagrams)
Time axis
Diagram 1
Yaxis
Shown variables
CpuLoadVariable
Variable settings
Variable
CpuLoadVariable
Graph color
Blue
Line type
Line
Point type
Dot
Activate minimum warning
Critical lower limit
Warning minimum color
Black
Activate maximum warning
Critical upper limit
80
Warning maximum color
Red
Reset Display settings
OK
Cancel


Trace Configuration
Trace Record
CpuLoadRatePacket
CpuLoadVariable
Presentation (diagrams)
Time axis
Diagram 1
Yaxis
Shown variables
CpuLoadVariable
Display Mode
Auto
Fixed length
Fixed
Minimum
Maximum
Length
Grid
Gray
Tick Marks
Fixed pitching
Distance
Subdivisions
Font
Preview
Reset
Use as default
Reset Display settings
OK
Cancel

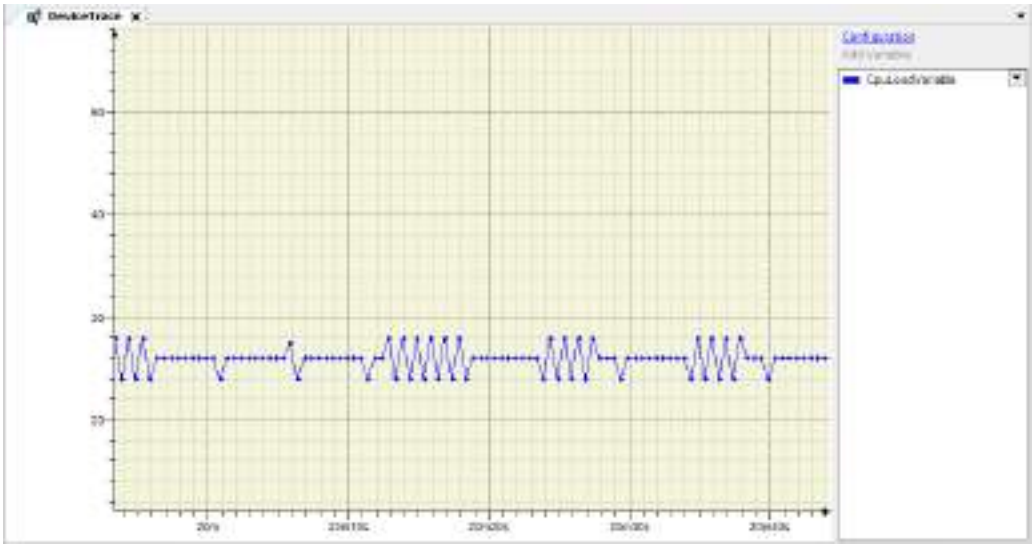

Trace Configuration
Trace Record
CpuLoadRatePacket
CpuLoadVariable
Display Mode
Auto
Fixed
Minimum
-12
Maximum
12
Grid
Gray
Description
Tick Marks
Fixed grading
Distance
1
Subdivisions
1
Font
Preview
Reset
Use as default
Presentation (diagrams)
Time axis
Diagram 1
Yaxis
Show variables
CpuLoadVariable
Reset Display settings
OK
Cancel


Trace Configuration
Trace Record
CpuLoadRatePacket
CpuLoadVariable
10
0
-10
0
5ms
10ms
Backcolor
White
Backcolor on Selection
Beige
Reset
Use as default
Presentation (diagrams)
Time axis
Diagram 1
Yaxis
Show variables
CpuLoadVariable
Add Diagram
Delete Diagram
Reset Display settings
OK
Cancel



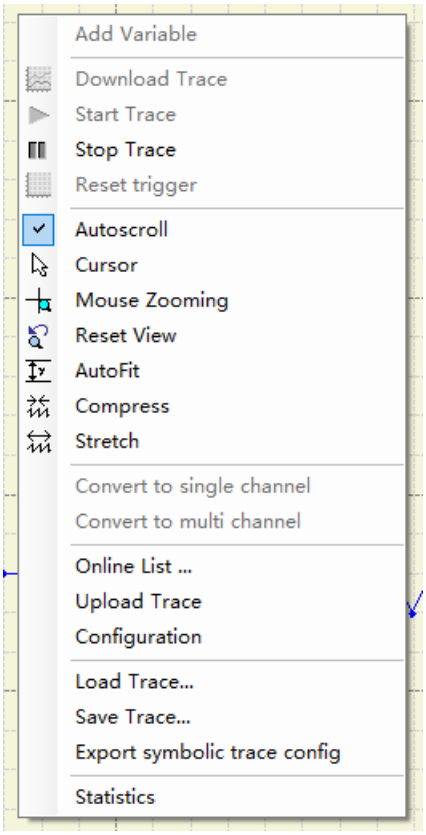
7. The following operations can be performed on the graph.

Dragging the mouse:	To move the time axis
Dragging the mouse while holding down the Ctrl key:	To move the Y-axis
Scrolling the screen:	To lengthen or shorten the time axis
Scrolling the screen while holding down the Ctrl key:	To lengthen or shorten the Y-axis



8. The following right-click menu items can be selected.

Save Trace:	Allows you to save the data plotted on the graph as a file
Load Trace:	Allows you to load the trace file saved by selecting "Save Trace" onto the graph screen



Note

- Use the GM1 controller so that the average CPU load factor is no more than 90%. If 90% is exceeded, stable operation may not be achieved.

9.8 Checking the Performance of GM1 Controller

The GM1 Series motion controller is a system in which multiple tasks run.

To ensure that the GM1 controller operates as a system normally, CPU resources must be allocated to each task properly as described below.

- If CPU resources are insufficient:

  MotionTask and cyclic tasks cannot be operated at the specified intervals. Refer to "9.7 Device Trace Function" and keep the CPU load factor at 80% or less as a guideline.
- If the CPU load factor exceeds 80%:

  Review the program in either one of the following ways.
  - Extend the interval of MotionTask and cyclic tasks.
  - For the program that does not require high speed processing, reassign to the tasks with long intervals.

Also, if the CPU load factor is high, a missing EtherCAT packet may occur. The following section explains how to check a missing EtherCAT packet.

9.8.1 Checking Missing EtherCAT Packets

The GM1 controller allows POU to be executed by allocating the POU to a task. To execute POU normally, the processing time (“cycle time") of the task must be smaller than the specified interval (*1) of the task.

*1: For "MotionTask", the specified interval is equivalent to the cycle time (control cycle) of "EtherCAT_Master_SoftMotion".

■ Task processing time

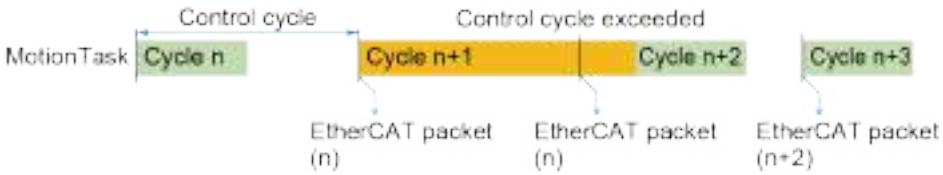
Task processing time can be viewed via the navigator pane and **Task Configuration**>**Monitor**.

In the following example, cycle times of "MotionTask" can be viewed.

Because cyclic communications are performed over the EtherCAT network, if the cycle time exceeds the task interval, the EtherCAT packet will not be updated in the next cycle, causing the commanded position to remain the same as the previous value. (Missing EtherCAT packet)

■ Example of a Missing EtherCAT Packet

In the following example, a missing EtherCAT packet occurs because the cycle time of cycle n +1 exceeds the cycle time (control cycle).



The occurrence of missing EtherCAT packet can be checked in the following ways.

1. Checking in POU

   Using the "IoDrVEtherCAT.GetStatistics.udiLostFrameCount" method, the occurrence situtation of the missing EtherCAT packet can be checked on the program.

For details on "IoDrVEtherCAT.GetStatistics.udiLostFrameCount", refer to the *GM1 Series Reference Manual (Instruction Edition)*.

2. Checking with GM Programmer settings

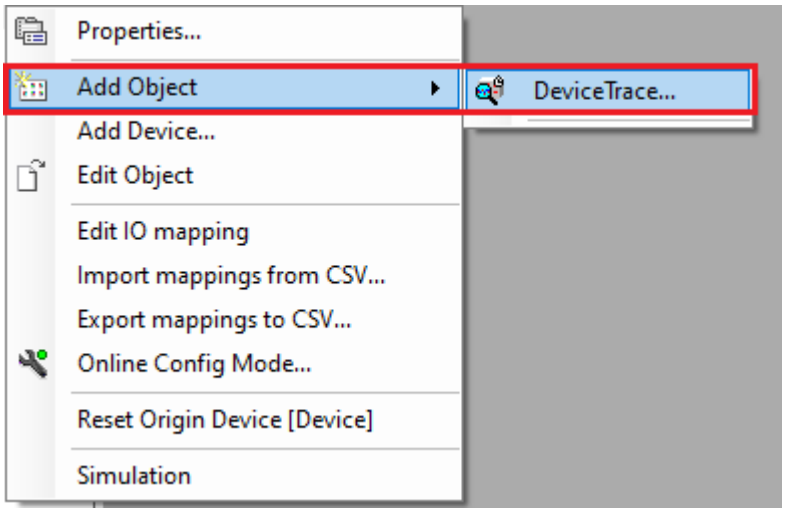
   Using the task watchdog timer function makes it possible to check whether the cycle time has exceeded the target value. In the example shown in the figure above, if time=1ms and sensitivity=1 are set, a watchdog timer error will occur in cycle n+1, causing the program to stop.

9.8.2 Performance Check Based on Device Trace

The GM1 controller allows multiple tasks to run. However, use the GM1 controller so that the CPU load factor (processing load of all tasks) is no more than 80%. The CPU load factor can be checked using the "Device Trace" function.

Procedure

1. Right-click the [Device] object in the navigator pane and then select **Add Object**>**DeviceTrace** from the context-sensitive object that is displayed.



The "Add DeviceTrace" dialog box will be displayed.

The selected "DeviceTrace" object will be added to the navigator pane.

Info.

- If the CPU load factor constantly exceeds 95%, the GM1 controller will judge the system to be out of control, causing the system to terminate with an error.

9.9 Error Notification Function

9.9.1 Overview of Errors

The GM1 controller has a self-diagnostic function which identifies errors and stops operation if necessary.

Indications concerning self-diagnosis are as follows.

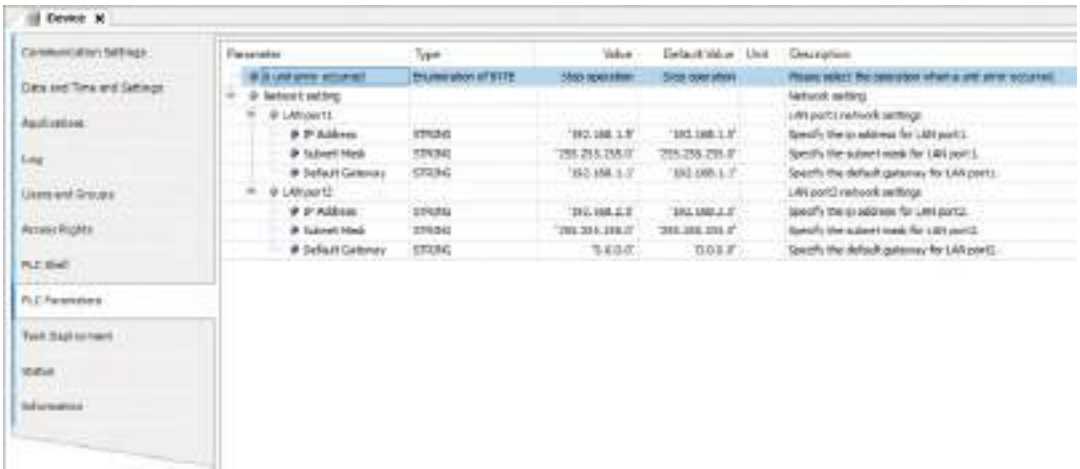
■ LEDs related to self-diagnostic errors

●: Lit, ▲: Flashing, ○: Unlit, —: Indefinite (Lit or unlit)

	LED display				Description	Operation status
	RUN	STOP	ERROR	ALARM		
Normal	●	○	○	○	Normal operation	Operating
	○	●	○	○	STOP mode	Stopped
Error	●	○	▲	○	When a self-diagnostic error occurs (Operation continues.)	Operating
	○	●	▲	○	When a self-diagnostic error occurs (Operation stops.)	Stopped
	○	●	-	●	System error	Stopped

■ PLC parameter setting

Operation mode at the time of error can be set to continue operation or stop operation in the PLC parameter setting.



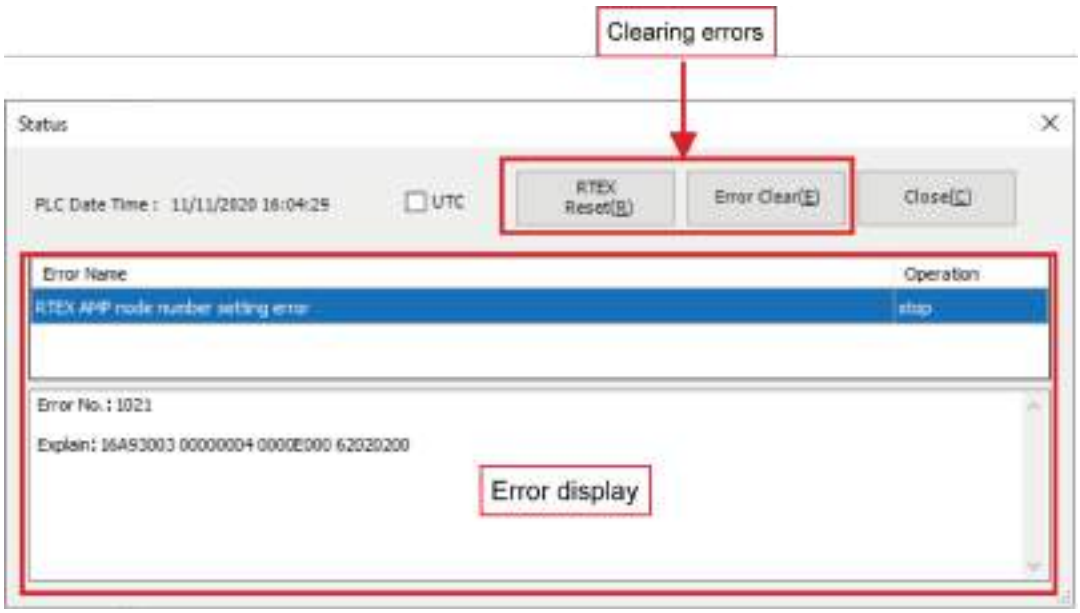
9.9.2 Checking and Clearing Errors Using GM Programmer

Error information can be checked in the status window of GM Programmer.

In case of an operation continue error, the error can be resolved by Error Clear.

Info.

- Since the error resolution method varies depending on the error, refer to "9.9.5 Error Code List".



9.9.3 Obtaining Error Information Using User Programs

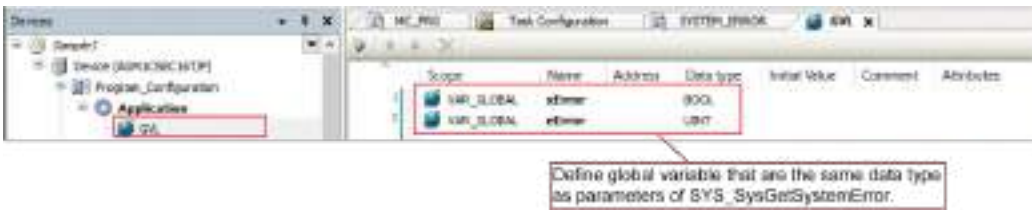
The following function block can be used to obtain error information for the GM1 controller.

- SYS_SysGetSystemError

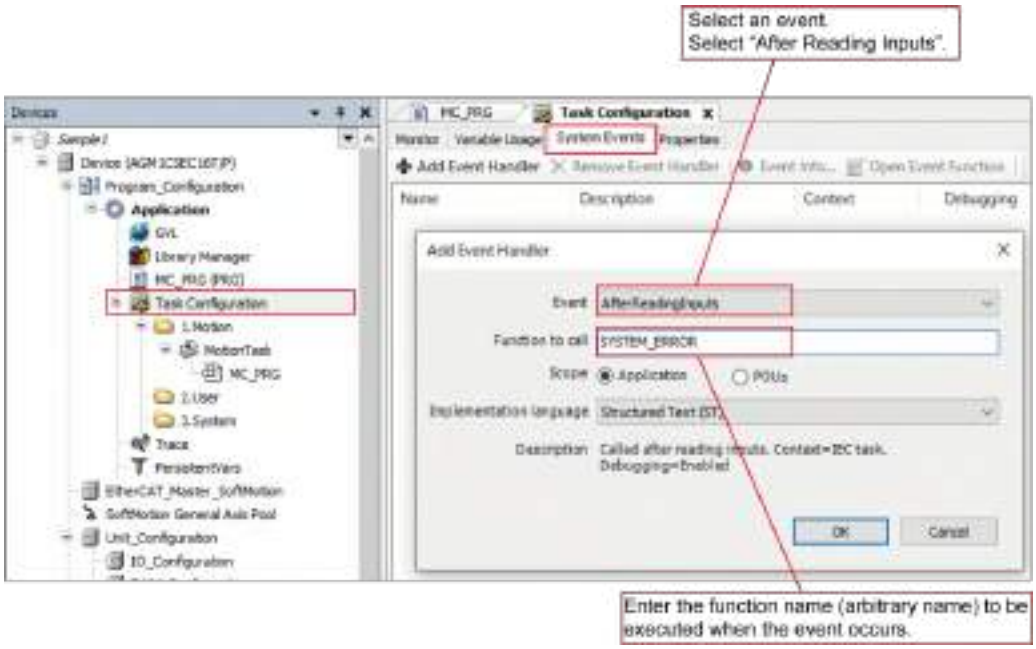
This function block is used to obtain error information for the GM1 controller from external devices such as display units.

Procedure

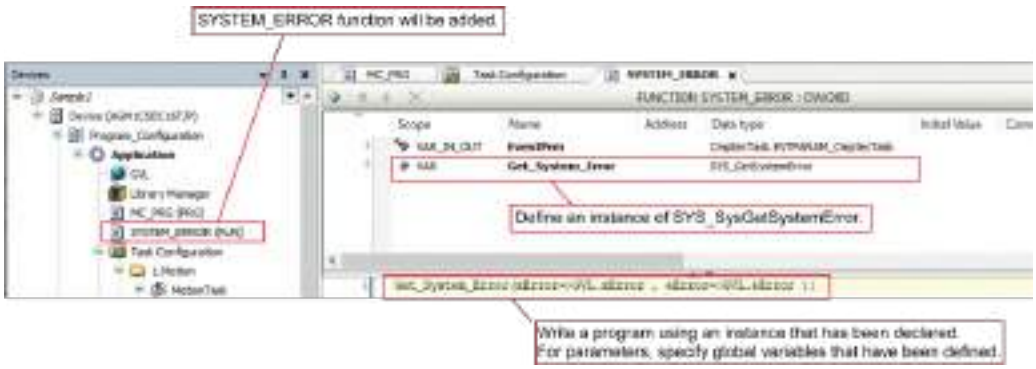
1. Define the variables to be used in the SYS_SysGetSystemError function block, as global variables.



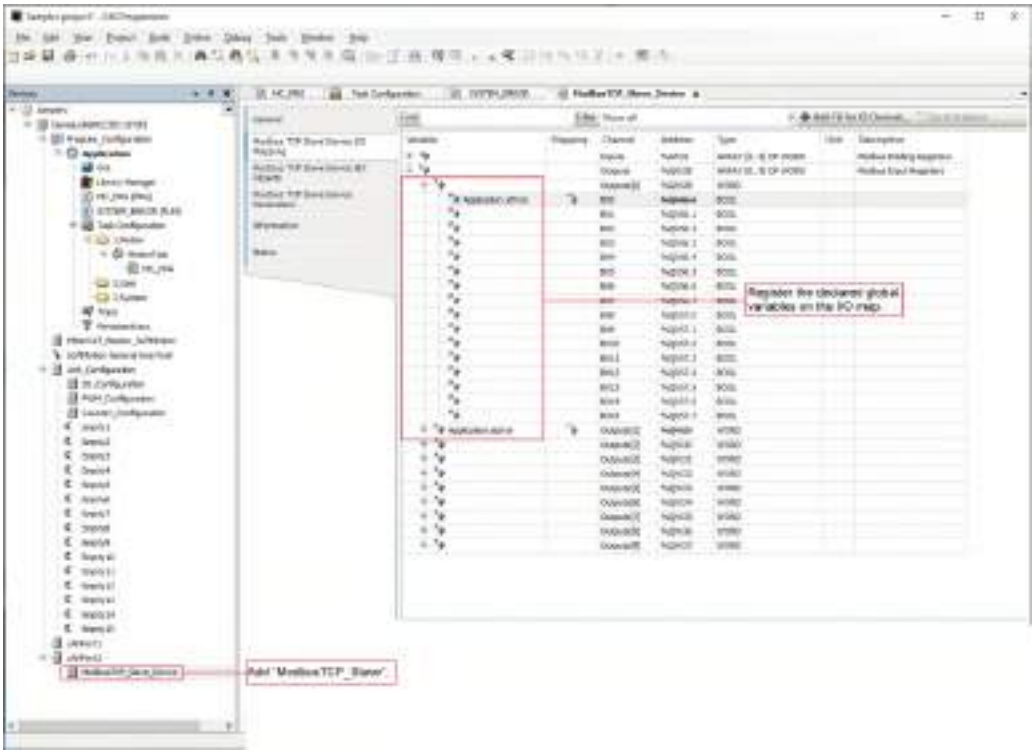
2. Select "Task Configuration" and then the "System Events" tab, and register the function to be executed when a particular event occurs.



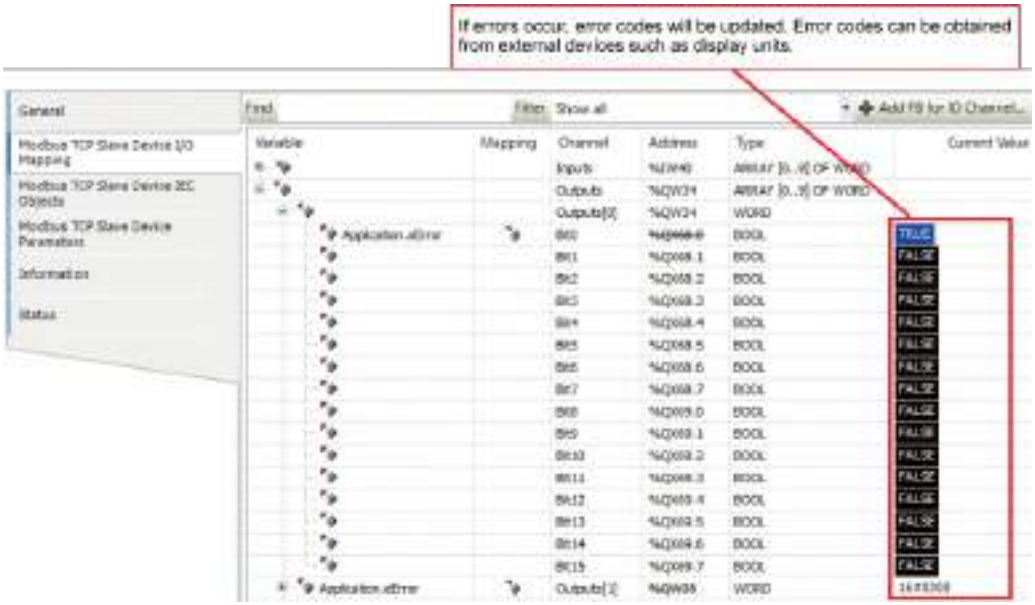
3. In the function that has been added, write a program for SYS_SysGetSystemError.



4. Map the defined global variables to the I/O map of the ModbusTCP Slave Device to allow external devices to get error information for the GM1 controller.



5. If an error occurs, the error code will be set in the variable, so that external devices can get error information for the GM1 controller.



9.9.4 Error Recovery Processing

Info.

- For errors during simulation, check the log window or each function block.

When an error occurs, normally, stop the operation.

■ When ERROR_LED flashes

A self-diagnostic error has occurred.

■ Solution

Check the condition according to the following procedure.

1. On the GM Programmer, select **Online**>**Status** and check the error content (error code).
2. Switch the mode to the PROG mode.
3. Cancel the situation in accordance with the error code.

■ When ALARM_LED lights up

Timeout of the system watchdog timer has been detected.

■ Solution

Check the condition according to the following procedure.

1. Turn the controller OFF and then ON.
   If the problem persists, consult your Panasonic representative.

■ Sample code

For a program that is used for error recovery processing, refer to the following.

<Sample program>

9.9.5 Error Code List

Error No.	Operation	Name	Error description and action method	Recovery method
0x0001	Stop	System error (serious)	Notify us of the error information or system data history information displayed on the status window of GM Programmer.	Power cycle
0x0002	Stop	System error (CODESYS)	Notify us of the error information or system data history information displayed on the status window of GM Programmer.	Power cycle
0x0003	Continue	System error (minor)	Notify us of the error information or system data history information displayed on the status window of GM Programmer.	Power cycle
0x0106	Continue	Device date and time RTC error	Because the duration of the power failure exceeded the guaranteed period (14 days), the RTC time information was initialized. The device must be left ON for at least 10 minutes. Note: Time information is reset to April 1, 2019.	Clear the error
0x0110 to 0x113	Continue	PWM Cycle setting error	The cycle setting is out of range. Check whether the duty ratio is within the allowable range.	Re-download
0x0120 to 0x123	Continue	PWM Duty ratio setting error	The duty ratio setting is out of range. Check whether the duty ratio is within the allowable range.	Re-download
0x0130 to 0x131	Stop	Counter Overflow error	The counter value exceeds the upper limit. Perform reset or preset operation so that the counter value falls within the range between the upper and lower limits.	Re-download
0x0140 to 0x141	Stop	Counter Underflow error	The counter value drops below the lower limit. Perform reset or preset operation so that the counter value falls within the range between the upper and lower limits.	Re-download
0x0150 to 0x151	Continue	Counter Reset abnormal error	The count value becomes 0 under the reset conditions and goes out of the range between the upper and lower limits. Check the settings of the upper and lower limits of the counter. If 0 does not exist in the range between the upper and lower limits, make a preset request.	Re-download
0x0160 to 0x161	Continue	Counter Preset abnormal error	The count value goes out of the range between the upper and lower limits under the preset conditions. Check whether the preset value falls within the range between the upper and lower limits.	Re-download
0x0170 to 0x171	Continue	Counter Current value change abnormal error	The count value goes out of the range between the upper and lower limits when the current value is changed. After the current value is changed, check whether the new value falls within the range between the upper and lower limits.	Re-download
0x0200	Continue	Expansion unit Communication error	An expansion unit communication error has occurred. Check connections.	Clear the error
0x0201	Stop	Expansion unit	More than 15 expansion units are connected. Check connections.	Power cycle

Error No.	Operation	Name	Error description and action method	Recovery method
		Number of connections exceeded		
0x0203	Stop	Expansion unit Startup error	An attempt to upgrade the expansion unit could have failed. Upgrade again.	Re-download
0x0205	Stop	Expansion unit Startup wait timeout error	Waiting for expansion unit startup has timed out. Check connections.	Re-download
0x020C	Stop	Expansion unit Connection number mismatch	The number of expansion units in the project does not match the number of expansion units mounted. Check connections.	Re-download
0x020D	Stop	Expansion unit Model code mismatch	The expansion unit model in the project does not match the expansion unit model mounted. Check connections.	Re-download
0x020E	Stop	Expansion unit Version mismatch	The version of the expansion unit registered in the project does not match the version of the expansion unit mounted.	Re-download
0x020F	Stop	Expansion unit Unit initialization error	Expansion unit initialization has failed. Check connections.	Re-download
0x0221 to 0x022F	Continue	Expansion unit I/O data error	An I/O data error has occurred in the expansion unit. Check the installation environment.	Clear the error
0x0300	Continue	CODESYS error	A CODESYS error has occurred. Check the error details in the log window of GM Programmer.	Clear the error

(MEMO)

10 Useful Functions of GM Programmer

10.1 Simulation Function 10-2

10.2 Security Function 10-3

10.3 Security Function: User Management 10-4
- 10.3.1 Project User Management 10-4
- 10.3.2 Creating a New User and Group 10-4
- 10.3.3 Setting Operation Privileges 10-8
- 10.3.4 Performing Operation with Privileges Set 10-10
- 10.3.5 Device User Management 10-11

10.4 Security Function: Encryption 10-16
- 10.4.1 Encrypting Project Files 10-16
- 10.4.2 Encrypting the Communication Path: Encrypting Communications Using the Certificate Possessed by the GM1 Controller 10-17
- 10.4.3 Encrypting the Communication Path: Encrypting Communications Using a Created Certificate 10-19

10.5 Security Function: Write-protection 10-24
- 10.5.1 Opening Files in Read-only Mode 10-24
- 10.5.2 Setting the "Released" Flag 10-24

10.6 User Library Function 10-26
- 10.6.1 Creating a Library and Adding to the Library Repository 10-26
- 10.6.2 Using Created Libraries 10-31

10.7 POU for implicit checks 10-34
- 10.7.1 Setting up POU for implicit checks 10-34

10.8 Interface 10-36
- 10.8.1 Setting up an Interface Object 10-36
- 10.8.2 Implementing in New Function Block 10-38
- 10.8.3 Implementing in Existing Function Block 10-40
- 10.8.4 Extending the Interface 10-43

10.9 External File Functions 10-46
- 10.9.1 Setting up an External File Object 10-46

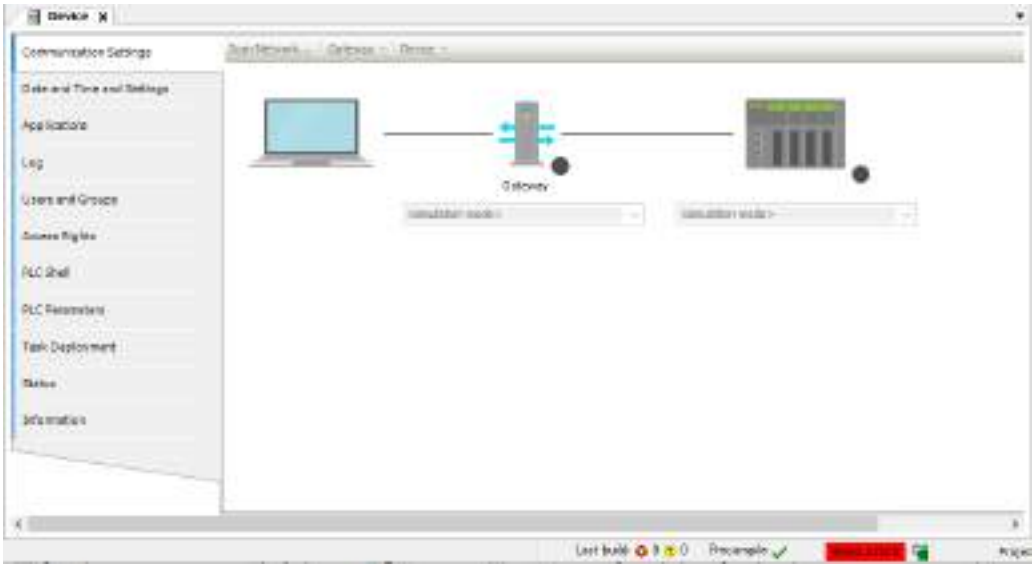
10.10 Servo Amplifier / Motor Operation Function (PANATERM Lite for GM) 10-48
- 10.10.1 Starting PANATERM Lite for GM 10-48

10.1 Simulation Function

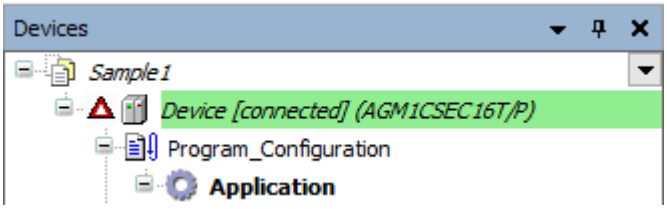
Simulation mode allows the user to perform a login operation without connecting to the GM1 controller. It also allows the user to check behaviors in the same way as if the user logged in.

Procedure

1. From the menu bar, select **Online**>**Simulation**.
   Simulation mode will be invoked and "Simulation" will be displayed on the status bar.



2. From the menu bar, select **Online**>**Login**, or press the <Alt> key and the <F8> key simultaneously.
   Login will occur in simulation mode. When login occurs in simulation mode, the device object is displayed in italic.



Simulation will start.

Info.

- is displayed in front of the object of a device that is operated in simulation mode.
- The cycle time in simulation mode may not operate according to the set cycle because of a difference in processing time depending on the operating environment.

10.2 Security Function

GM Programmer is equipped with a security function that can implement user management (project user management and device user management) and encrypt project files. This section explains security-related functions such as user management and project encryption, and the procedures for operating each security function.

- Each user ID and their corresponding password must be different character strings.
- Password strength must be sufficiently high. Do not use any passwords that can be easily guessed.
- Accounts must be managed properly and must not be shared unnecessarily.

- Use this controller in a secure network environment.
- Use encryption functions properly to protect information assets.
- Use the device user management function to perform authentication protection for the controller during operation.
- After logging in with the initial password, be sure to change the password.
- Implement password management to prevent passwords from being forgotten. If the password is forgotten, device reset must be performed on the controller.
- Implement password management to prevent passwords from being leaked to third parties.

Item	Description	Reference page
User management	Allows execution permissions for operations (such as executing menu commands and adding, editing, and deleting objects) to be assigned to each group in which users are registered. User management also enables logins to be permitted by assigning permission for login to the device to each user and entering passwords.	"P.10-4"
Encryption / signature	Provides password-based encryption for project files and encrypts connections between the GM1 controller and the PC.	"P.10-16"
Write-protection	Provides write-protection for project files and prevents project files from being modified unintentionally by mistake.	"P.10-24"

10.3 Security Function: User Management

Project user management allows execution permissions for operations (such as executing menu commands and adding, editing, and deleting objects) to be assigned to each group in which users are registered. Logon must be performed by a user of a privileged group.

Device user management also enables logins to be permitted by assigning permission for login to the device to each user and entering passwords.

10.3.1 Project User Management

When a new project is created, Owner users, Owner group, and Everyone group are already registered.

Owner users belong to Owner group and can execute all operations.

All users including Owner users are automatically registered in Everyone group.

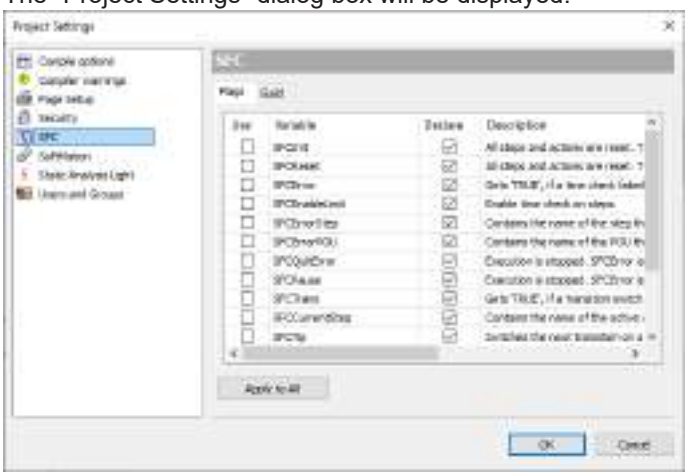
Group	User	Remarks
Owner	Owner	Users can execute all operations. For Owner users, the password field is left blank.
Everyone	Owner	All users are registered automatically.

10.3.2 Creating a New User and Group

In the following example, a group (group name: GroupA) and a user belonging to the group (user name: Fred) are newly created and privileges are set so that users belonging to GroupA can access POU object "POU_1".

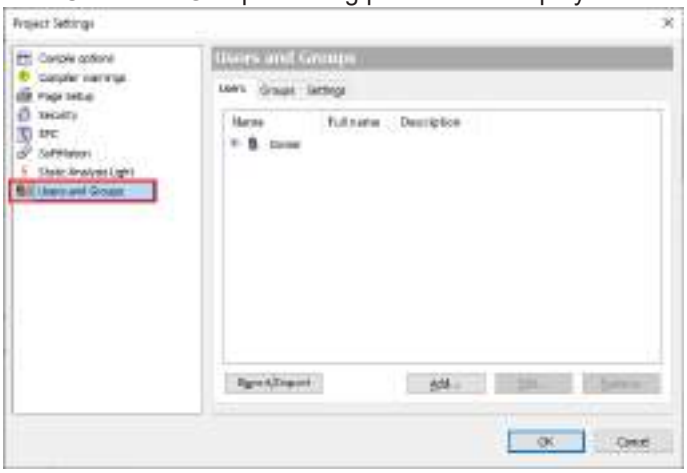
Procedure

1. From the menu bar, select **Project**>**Project Settings**.
   The "Project Settings" dialog box will be displayed.

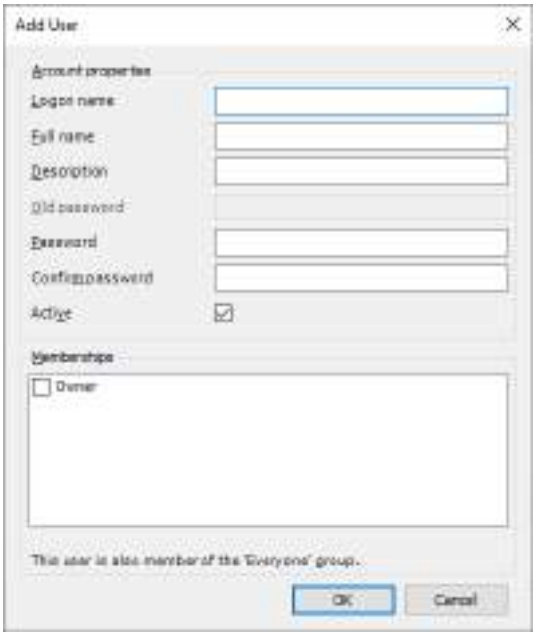


2. In the "Project Settings" dialog box, select the "Users and Groups" category.

The "Users and Groups" setting pane will be displayed.



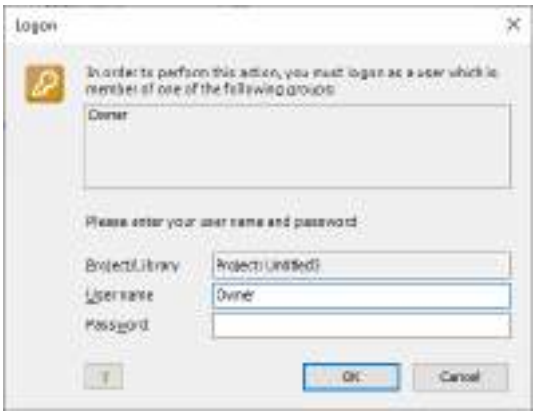
3. Click the [Add] button.
The "Add User" dialog box will be displayed.



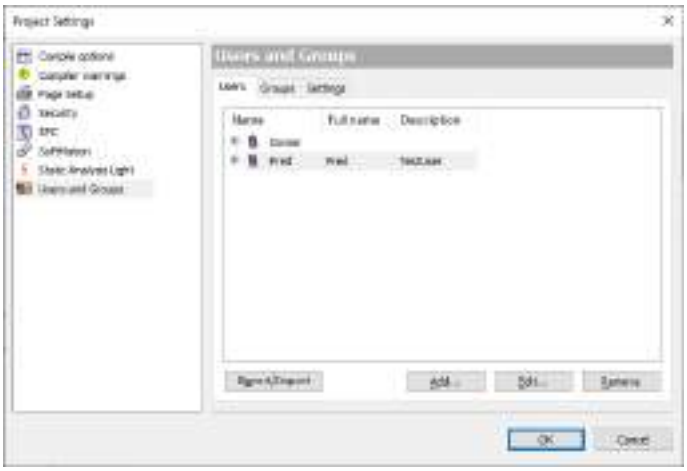
4. Enter information for a new user to be added.
Enter information about a new user (Fred) to be added.

5. Click the [OK] button.
The "Logon" dialog box will be displayed. To add a new user, you must log in as an Owner user.

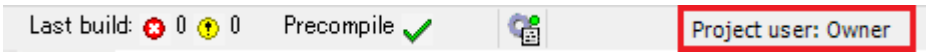
6. In the "User name" field, enter "Owner".
The default password for "Owner" is not set in the Password field.
The "Password" field must be left blank.



7. Click the [OK] button.
Login by the Owner user will be completed and new user "Fred" will be added to the "Users" tab pane.

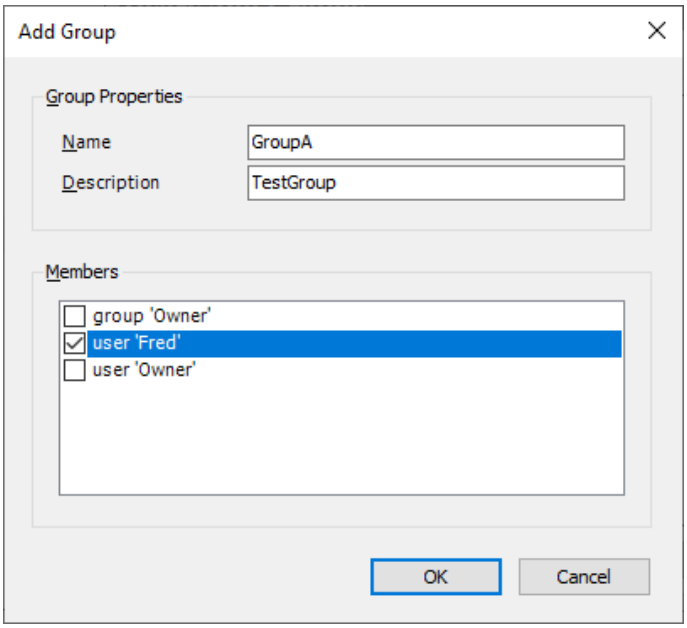


The name ("Owner") of the user who logged on is displayed in the status field.

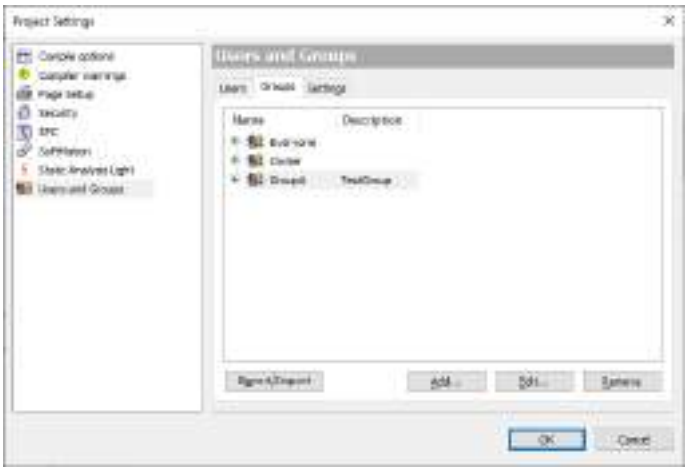


8. Select the "Groups" tab and click the [Add] button.
The "Add Group" dialog box will be displayed.

9. Enter information for a new group to be added.
Enter information about a new group (GroupA) to be added.
In the "Members" section, specify a member that belongs to the group. Select the new user added in step "Step 4".



10. Click the [OK] button.
The new group (GroupA) will be added to the "Groups" tab pane.
User "Fred" is registered in GroupA as a member



11. Click the [OK] button.
The "Project Settings" dialog box will be closed.

This completes the procedure for registering user "Fred" in group "GroupA".
After the procedure is complete, the following groups and users exist in the project.

Group	User	Remarks
Owner	Owner	Group whose users can execute all operations
Everyone	Owner, Fred	Group in which all users are registered automatically
GroupA	Fred	Newly added group

Info.

- User and group information can be exported in XML format. Click the [Export/Import] button in the "Users and Groups" setting pane and select the "Export users and groups" menu item". .users" files can be exported.
  By selecting the "Import User and Group" menu item, you can import ".users" files.

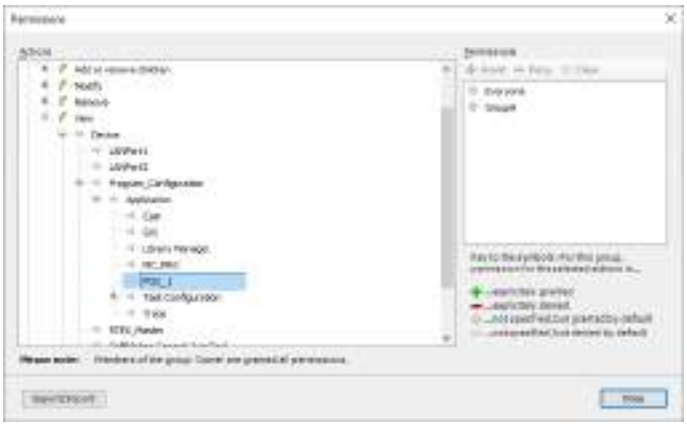
10.3.3 Setting Operation Privileges

In the following example, privileges are set so that users belonging to GroupA can display a POU object (object name: POU_1).
Before performing the following procedure, add a POU object (object name: POU_1) to the project.

Procedure

1. From the menu bar, select **Project>User Management>Permissions**.
The "Permissions" dialog box will be displayed.

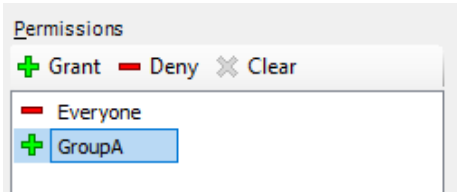
2. In the "Action" pane, select the operation to be permitted.
   Select **Project Object**>**View**>**Device**>**Program Configuration**>**Application**>**POU_1**.



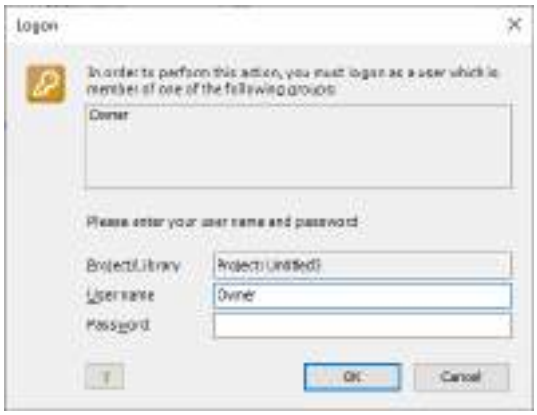
3. In the "Permissions" pane, set privileges to be assigned.

   Select "Everyone" and click Deny. "Reject" will be set.

   Select "GroupA" and click Grant. "Approve" will be set.
   This enables only the users of group "GroupA" to display the object.



   If the "Logon" dialog box is displayed, enter "Owner" in the "User name" field and leave the "Password" field blank before performing a logon.



4. Click the [Close] button.
   The "Permissions" dialog box will be closed.

Info.

- Settings of operation privileges can be exported in XML format. In the "Permissions" dialog box, click the [Export/Import] button and then select the "Export All Permissions" menu item or "Export selected permissions" menu item". .perms" files can be exported.
  By selecting the "Import Permission" menu item, you can import ".perms" files.

10.3.4 Performing Operation with Privileges Set

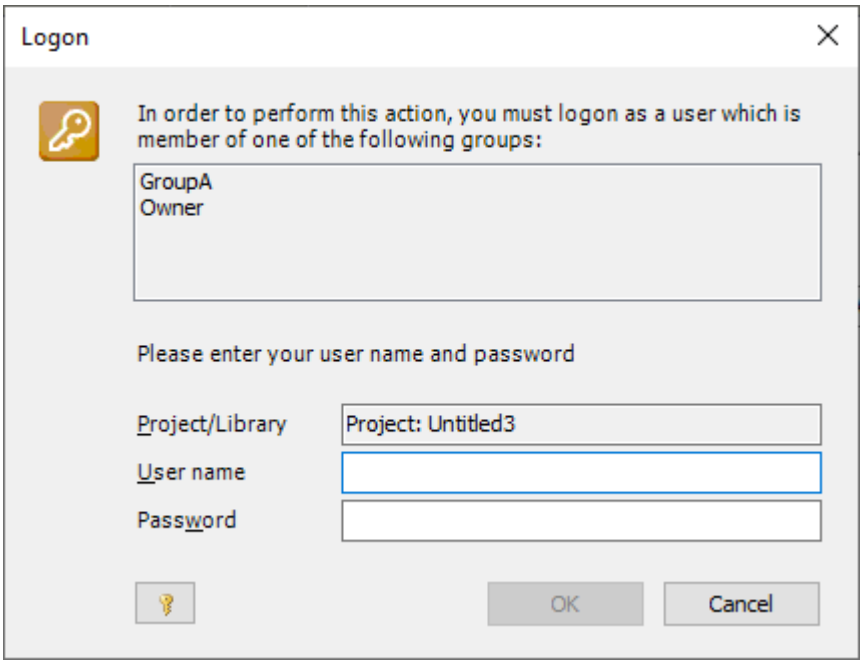
In the following example, an object (POU object: POU_1) is displayed.

Before performing the following procedure, check the status field to see that there is no user who is currently logged on the project.

If there are any users who are currently logged on the project, execute logoff by selecting **Project>User Management>User Logoff**.

Procedure

1. Double-click the POU_1 object in the navigator pane.
   The "Logon" dialog box will be displayed with object display operations restricted.



2. Enter appropriate values in the "User name" field and "Password" field, and click the [OK] button.
   Enter the user name and password of the user added in "10.3.2 Creating a New User and Group".
   Logon will be completed and the POU_1 object will be displayed.
   The user name of the user who logged on is displayed on the status field.

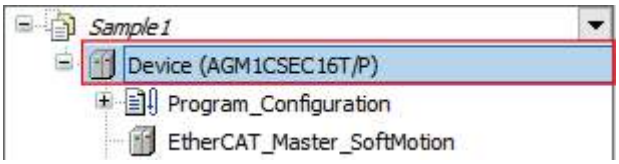
10.3.5 Device User Management

Device user management registers device users and allows only the authorized device users to log in to the device.
A user with user name "Administrator" and password "Administrator" is registered as a device user beforehand.
When you log in as an Administrator user for the first time, you must set any password.

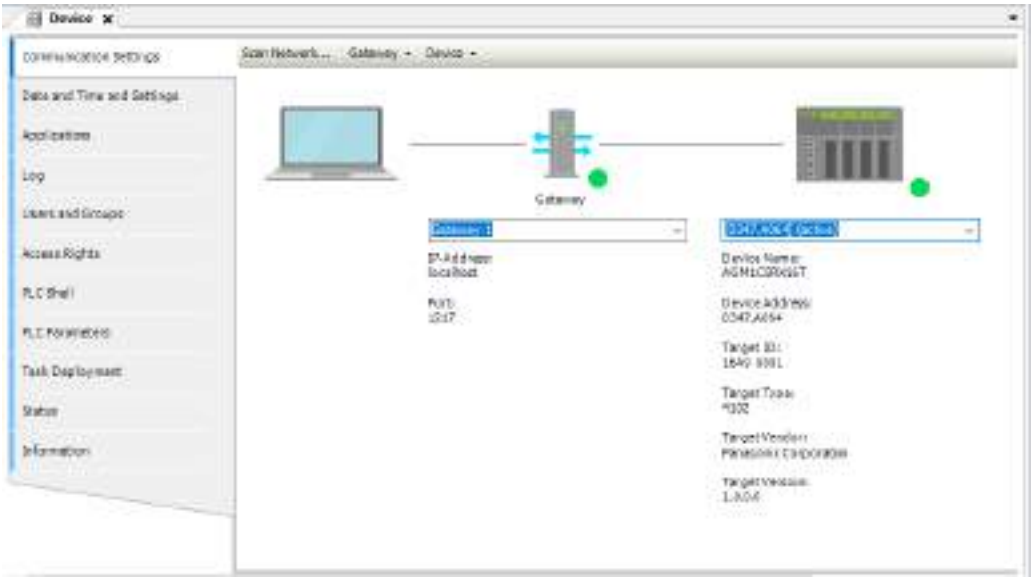
Procedure

1. Connect the PC where GM Programmer is installed and the GM1 controller.
   For details, refer to "8.5 Connecting to the GM1 Controller".

Double-click the [Device] object in the navigator pane.

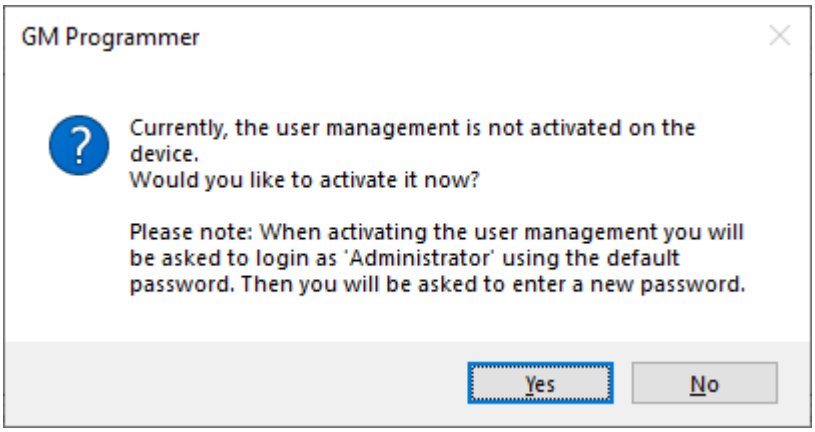


The Device setting window will be displayed.



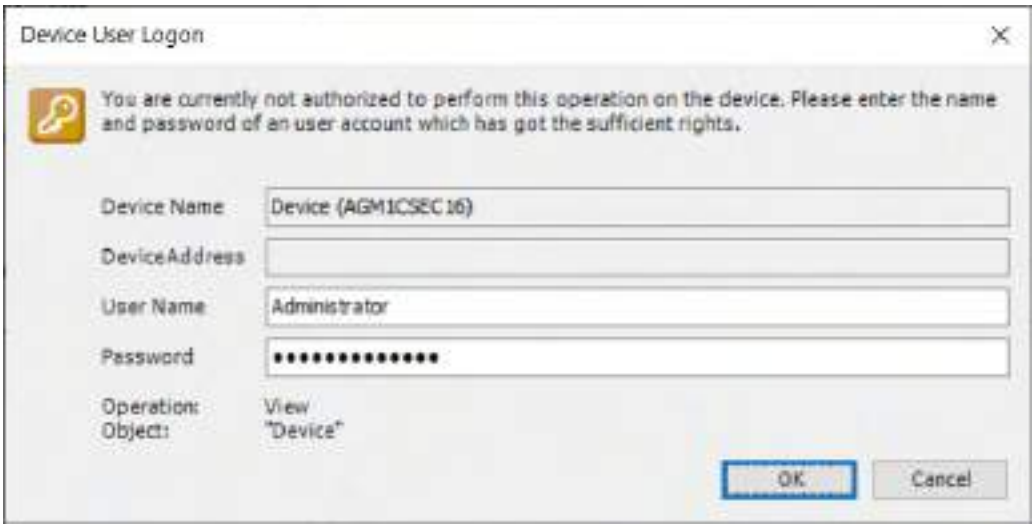
2. Click the "User and Group" tab.
   The "User and Group" pane will be displayed.

3. Click the [ ] icon (Synchronization).
   A confirmation dialog box will be displayed.



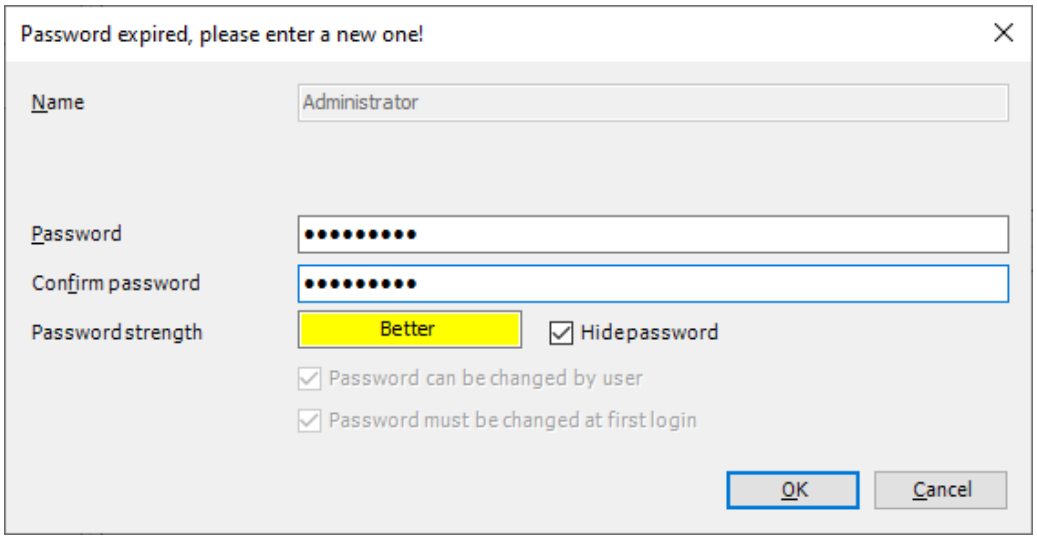
4. Click the [Yes] button.
   The "Device User Login" dialog box will be displayed.

5. Enter "user name" and "password".
   Enter Administrator in the "User name" field and Administrator in the "Password" field.



6. Click the [OK] button.
   The "Password expired, please enter a new one" dialog box will be displayed.

7. Enter any password.
   To set a password for the Administrator user, enter any password.
   If you forget your password, you cannot log in to the device.

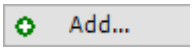


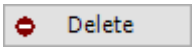
8. Click the [OK] button.
The password will be set for the Administrator user and you will be logged in as an Administrator user.

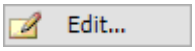
9. From the menu that is displayed, select **Online>Login**.
You can log in to the device as an Administrator user account.

Info.

- To log off logged-in users, from the menu bar, select **Online>Security>Logoff Current Device User**.
- You can add or remove device users or change their passwords by using the "User and Group" pane.

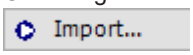
  To add device users: 

  To remove device users: 

  To change device user passwords: 

- Users registered by project user management can be imported as device users.

  Clicking the button displays the "Import Users" dialog box. Select a user to be imported and click the [OK] button. In this case, passwords managed by project user management will not be imported. In the "Users and Groups" pane, click the [Edit] button and set a password for the user that has been imported.
- Device user management information can be exported.

  In the "User and Group" pane, click the [] icon (Export to Disk).

  XML format files (".dum" files) can be saved.

  To import ".dum" files that have been exported, click the [] icon (Import from Disk).
- Device user management information can be initialized by resetting the device.
- If you forget your password, you cannot log in to the GM1 controller. In this case, reset the GM1 controller. For details on how to reset the GM1 controller, refer to the *GM1 Series User's Manual (Hardware Edition)*.

10.4 Security Function: Encryption

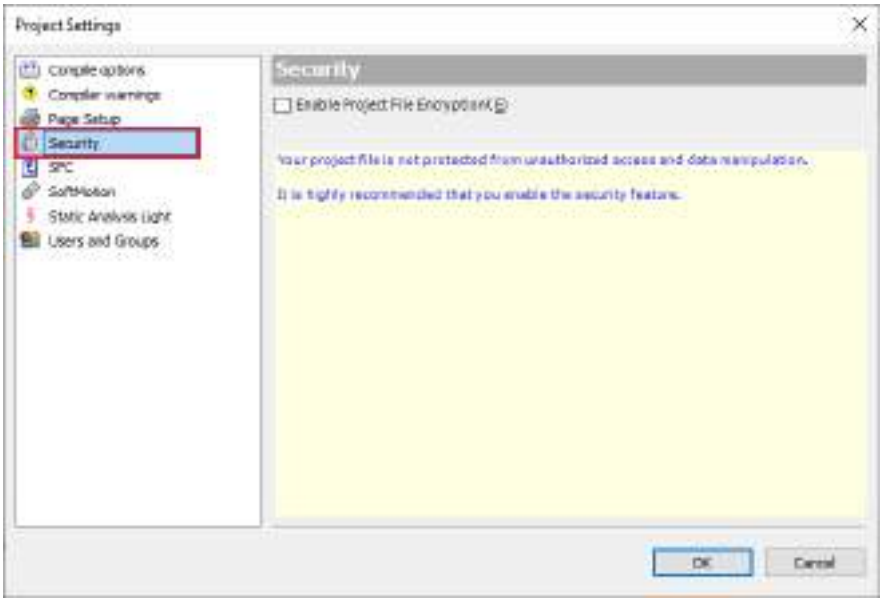
This section explains how to encrypt project files.

10.4.1 Encrypting Project Files

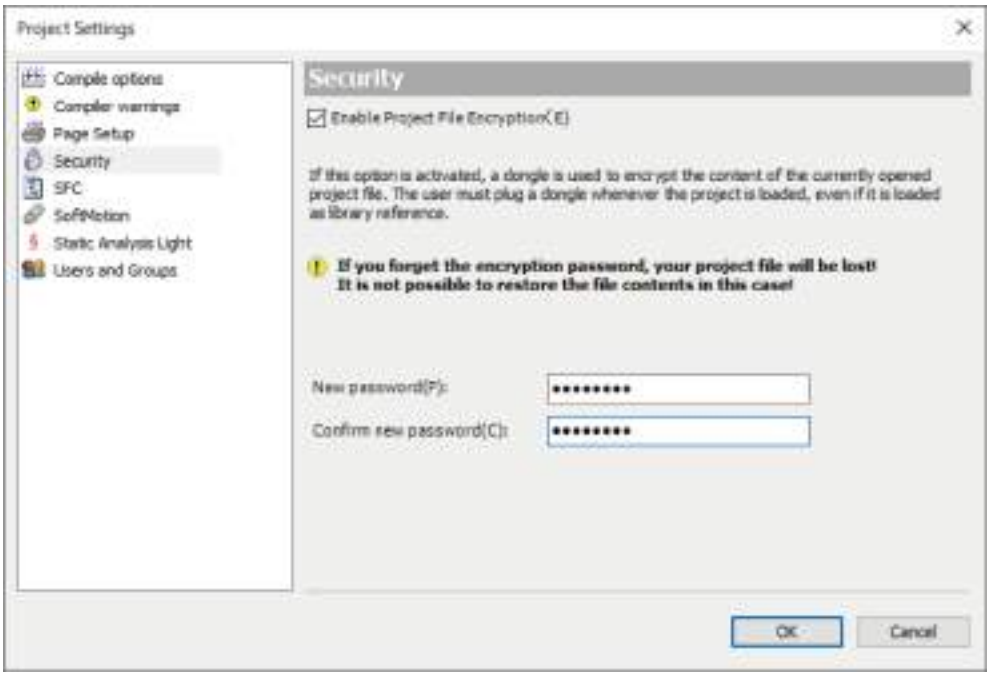
Project files can be encrypted using passwords. If a password is set, the password must be entered when a project file is opened.

Procedure

1. From the menu bar, select **Project**>**Project Settings**.
   The "Project Settings" dialog box will be displayed.
2. In the "Project Settings" dialog box, select the "Security" category.
   The "Security" pane will be displayed.



3. Select the "Enable Project File Encryption" check box, select the Password option, and then enter a password.



4. Click the [OK] button.
The specified password will be set for project files.

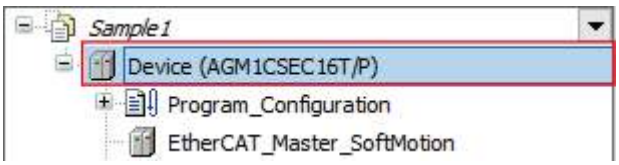
This completes the password setting procedure.
When an attempt is made to open a project file, a window is displayed, asking the user to enter a password. In this situation, enter the specified password.

10.4.2 Encrypting the Communication Path: Encrypting Communications Using the Certificate Possessed by the GM1 Controller

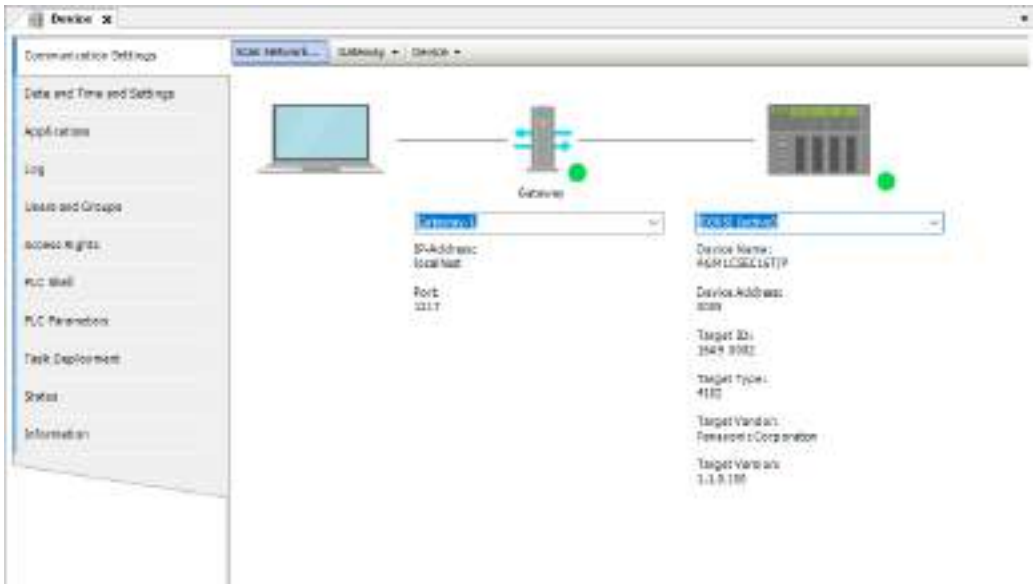
Communications between GM Programmer and the GM1 controller can be encrypted using certificates.
This section explains how to encrypt communications by using the certificate possessed by the GM1 controller as a trusted certificate.

1 2 Procedure

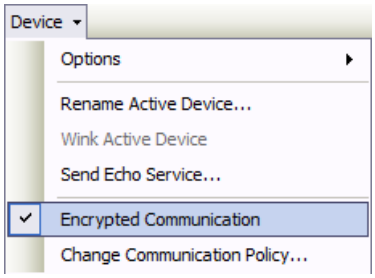
1. Double-click the [Device] object in the navigator pane.



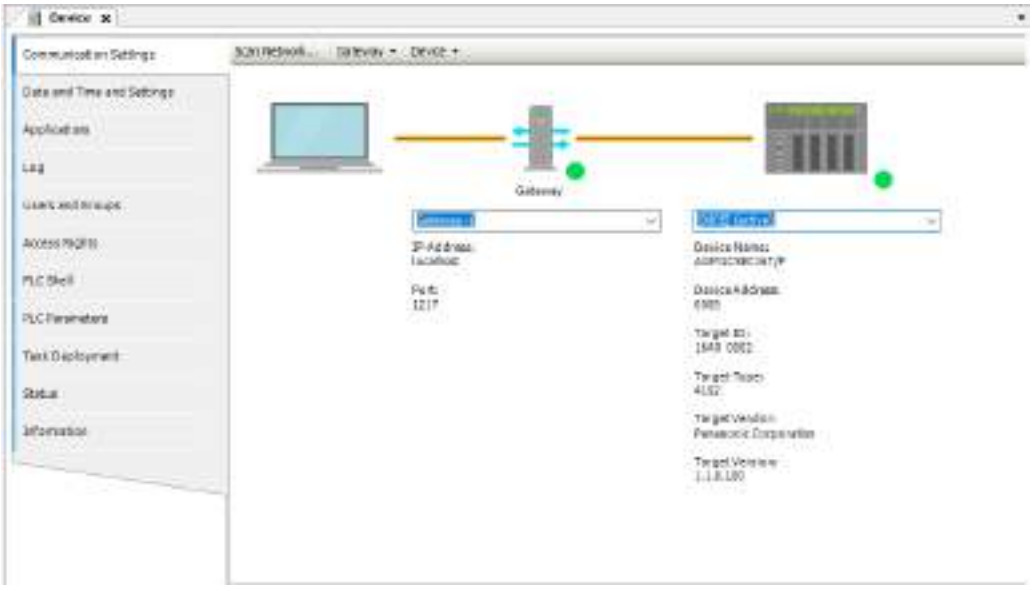
The Device setting window will be displayed.
Open the "Communication Settings" tab.



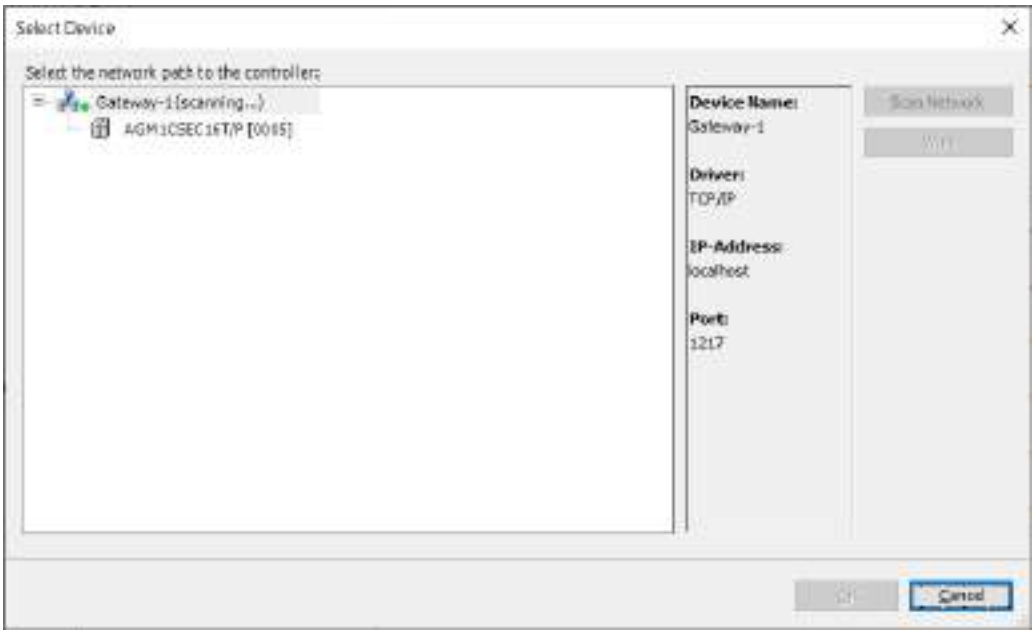
2. In the Device menu, select "Encrypted Communication".



If "Encrypted Communication" is selected, the connection lines between the integrated development environment, gateway, and controller will be displayed in yellow.



3. Click the Network Scan menu.
   The "Select Device" dialog box will be displayed.



4. Select the connected GM1 controller and click the [OK] button.
   A message window will be displayed, indicating that the certificate of the GM1 controller is not certified with a trusted signature for encrypted communication.

5. If the [OK] button is clicked, communications can be encrypted by installing the certificate indicated by the message in local store "Controller Certificates" on the PC to use it as a trusted certificate.
   You can check the registered controller certificate in certmgr.msc in the C:\Windows \System32 folder.
   When the certificate of the GM1 controller is used as a trusted certificate, the validity period of the certificate is 30 days.

Info.

- If the certificate has already expired, the message window shown in step 4 above will be displayed, indicating that the certificate has expired.
  By clicking the [OK] button, you can extend the validity period of the certificate.

10.4.3 Encrypting the Communication Path: Encrypting Communications Using a Created Certificate

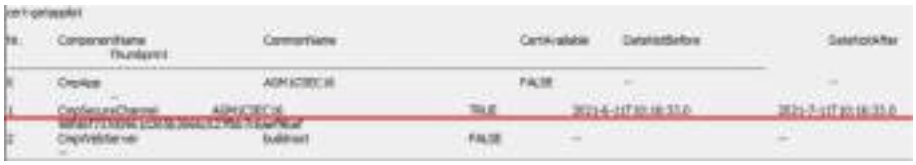
Communications between GM Programmer and the GM1 controller can be encrypted using certificates.

This section explains how to create a trusted certificate for the GM1 controller and encrypt communications using the created certificate.

Procedure

1. Open the device editor and select the "PLC Shell" tab.

2. Enter the "cert-getapplist" command in the input field.
   All certificates that are used will be displayed.



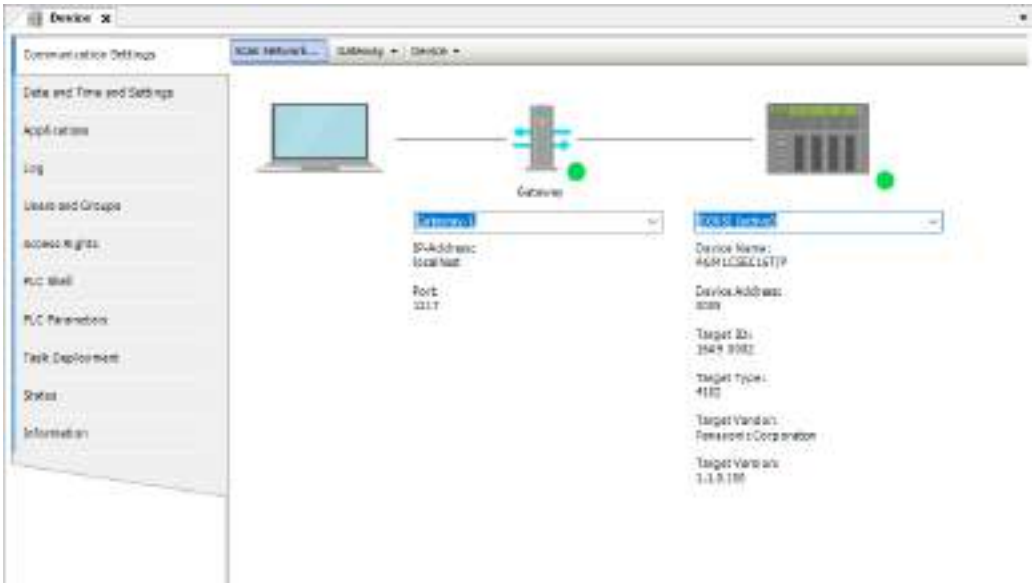
3. Enter the "cert-genselfsigned 1" command in the input field.
   Create a certificate for ComponentName "CmpSecureChannel".

```
cert-genselfsigned 1

Generate selfsigned certificate with given index (1). Check logger to see when finished.
```

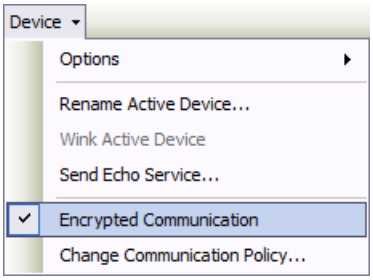
4. Open the "Log" tab of the device editor and click the [Update Information] button.
   Check whether a certificate has been created.

5. In the device editor, open the "PLC Shell" tab and enter the "cert-getapplist" command in the input field.
   Check whether a certificate has been created for ComponentName "CmpSecureChannel".

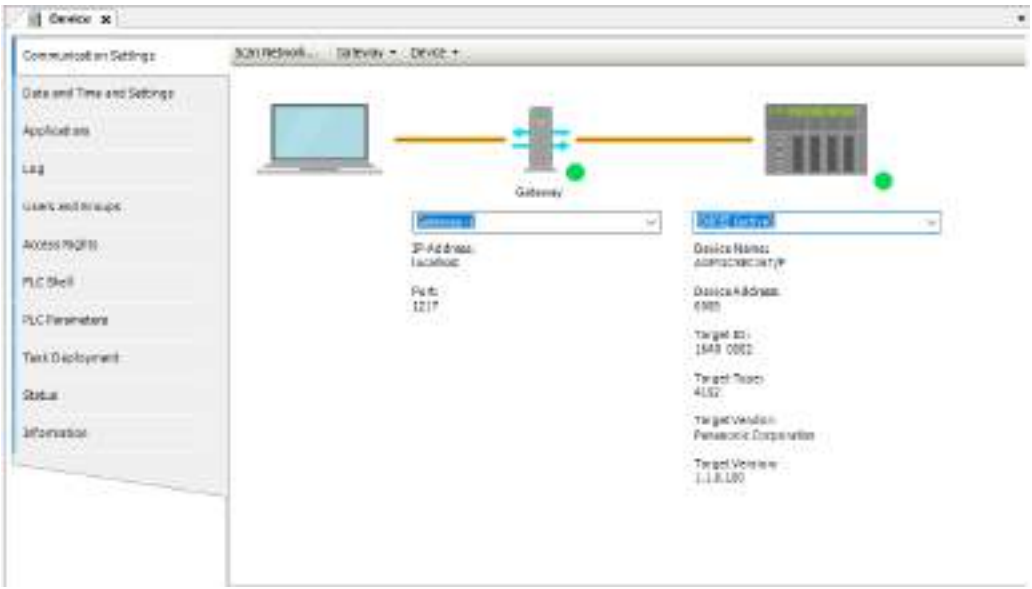
6. Open the device editor and select the "Communication Settings" tab.



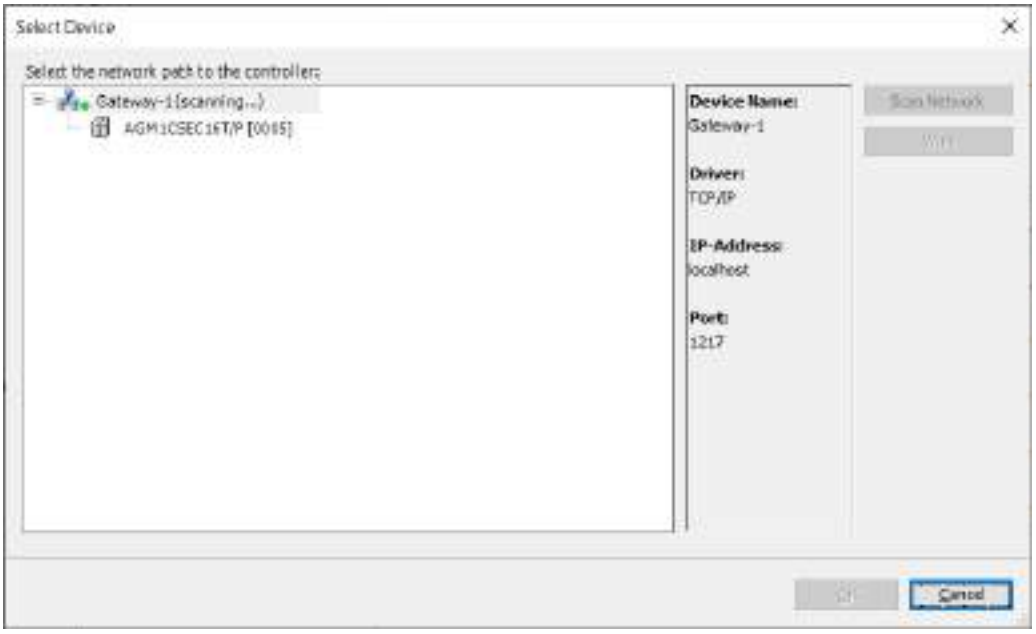
7. In the Device menu, select "Encrypted Communication".



If "Encrypted Communication" is selected, the connection lines between the integrated development environment, gateway, and controller will be displayed in yellow.



8. Click the Network Scan menu.
   The "Select Device" dialog box will be displayed.



9. Select the connected GM1 controller and click the [OK] button.
   A message window will be displayed, indicating that the certificate of the GM1 controller is not certified with a trusted signature for encrypted communication.

10. If the [OK] button is clicked, communications can be encrypted by installing the certificate indicated by the message in local store "Controller Certificates" on the PC to use it as a trusted certificate.

You can check the registered controller certificate in certmgr.msc in the C:\Windows \System32 folder.

When the created certificate is used as a trusted certificate, the validity period of the certificate is 360 days.

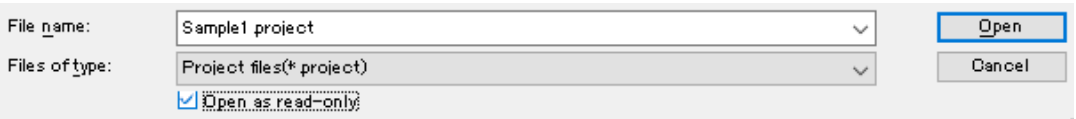
10.5 Security Function: Write-protection

This section explains how to implement write-protection for project files to prevent project files from being modified unintentionally by mistake.

10.5.1 Opening Files in Read-only Mode

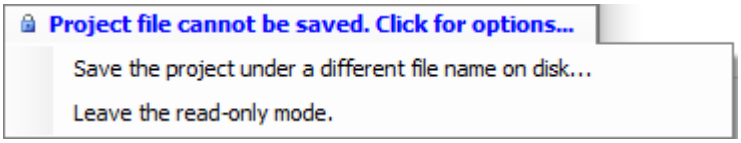
Open a project file in read-only mode.

When selecting a project file to be opened, select the "Open in Read-only Mode" check box.



If a file is opened in read-only mode, it cannot be saved.

To save a project file, select "Project file cannot be saved. Click for options" on the menu bar and select an appropriate menu item that is displayed.



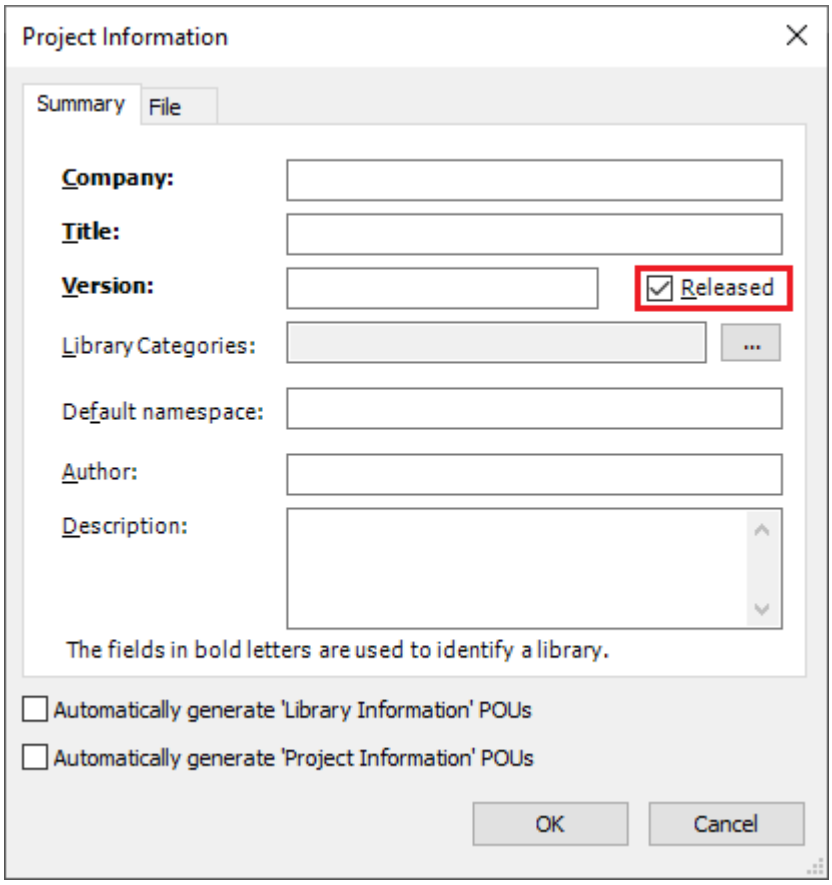
Item	Description
Save the project under a different file name on disk...	Allows the user to rename and save the project file as a writable file.
Leave the read-only mode	Leaves the project file open in read-only mode.

10.5.2 Setting the "Released" Flag

Set a "released" flag in project information in a project file.

If a "released" flag is set in a project file, changes made in the file cannot be saved.

From the menu bar, select **Project**>**Project Information**, and open the "Summary" tab window and then select the "Released" check box.



To save a project file in which the "Released" flag is set, select "Project file cannot be saved. Click for options" on the menu bar and select a menu item that is displayed.

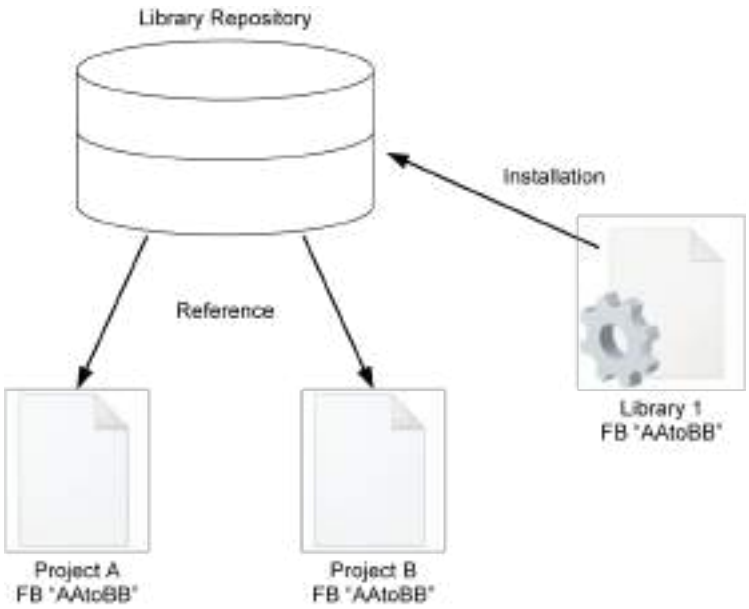
10.6 User Library Function

Combining created functions and function blocks as a library enables other projects to use these functions and function blocks.

The library that has been created must be installed in the library repository. Adding the library installed in the library repository to the project makes it possible to use the functions and function blocks in the library. Libraries in the project are managed by the "Library_Manager" object.

The following sections describe a procedure for creating a library, installing the created library into the library repository, and adding libraries to a project.

Example

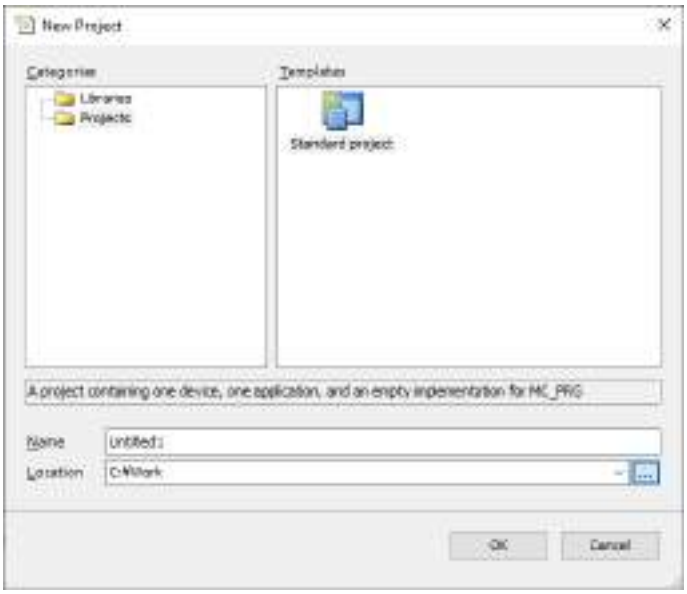


10.6.1 Creating a Library and Adding to the Library Repository

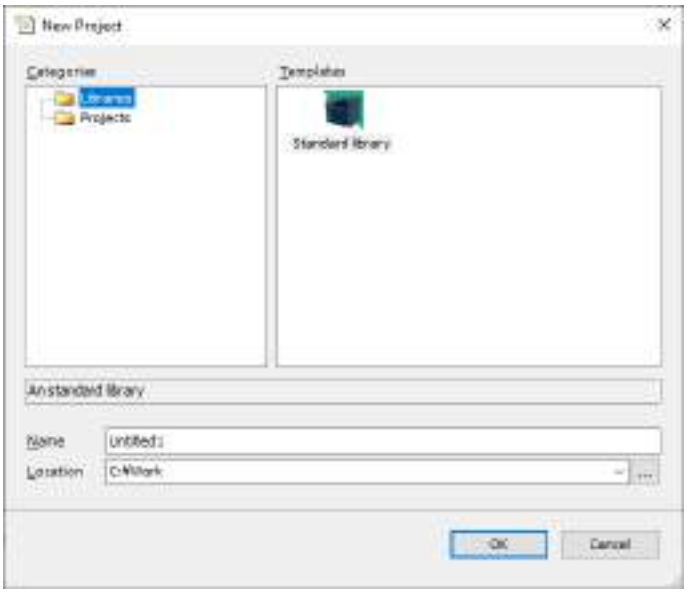
This section explains the entire procedure from creating a project for libraries through to installing libraries in the library repository.

Procedure

1. From the menu bar, select **File**>**New Project**.
   The "New Project" dialog box will be displayed.

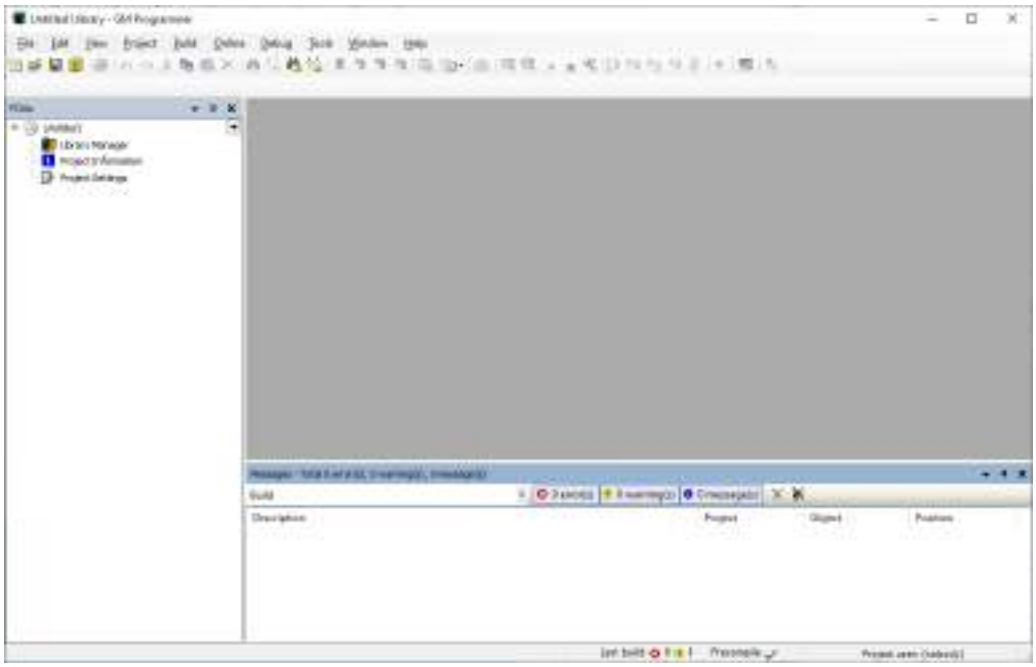


2. Select "Libraries" in the Categories pane and then "Standard library" in the Template pane.



3. Click the [OK] button.

A project for libraries will be created. The extension of project files for libraries is "library". For libraries, the POU view is displayed in the navigator pane. Add objects required for creating libraries to the POU view.

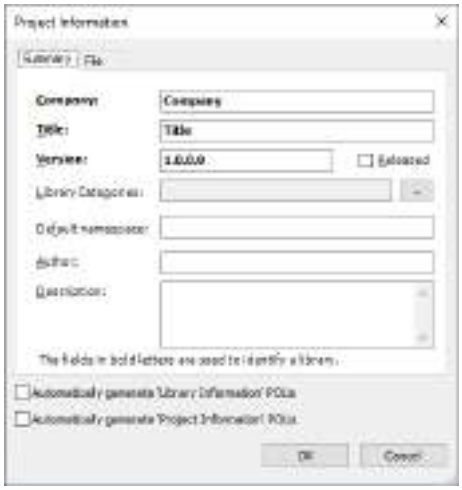


4. Double-click the "Project Information" object in the POU view.

   The "Project Information" dialog box will be displayed.

   If necessary, change the settings in the "Company", "Title", and "Version" fields. The information set in these fields will be displayed in the selection window when created libraries are added to the project.

   If the "Release" check box is selected, a confirmation message will be displayed when an attempt is made to change a library.

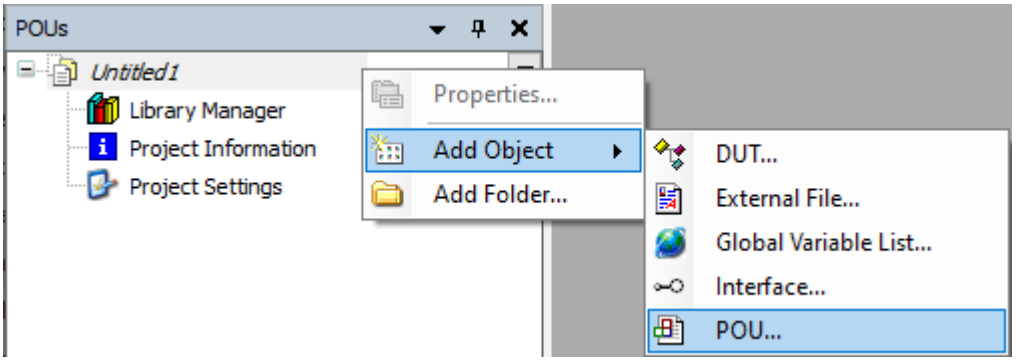


5. Click the [OK] button.

   The project information will be set.

6. Right-click the <file name> object at the top of the navigator pane and then select **Add Object>POU** from the context-sensitive menu that is displayed.

The "Add POU" dialog box will be displayed.



7. Select the "Function block" check box, enter a name in the Name field, and select a programming language from the Implementation Language drop-down list.
For details on functions blocks, refer to "6.6 Function and Function Block".

8. Click the [Add] button.
An object of the function block will be added to the POU view in the navigator pane.

9. Enter a program in the function block.

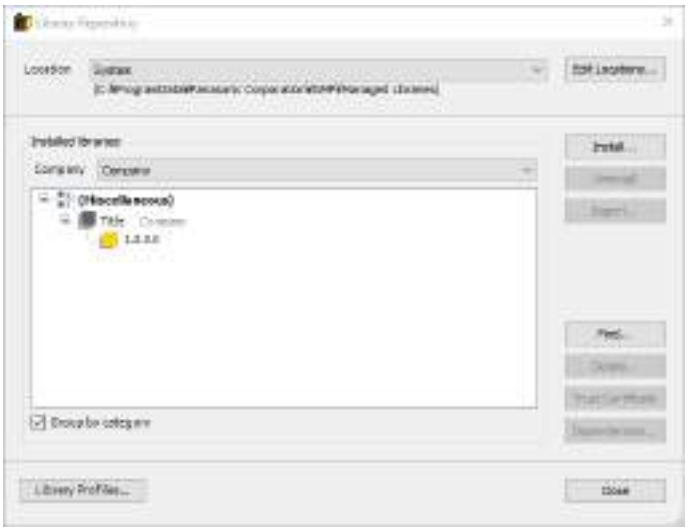
After entering a program, from the menu bar, execute **Build**>**Check All Pooled Objects**. Build will be executed to perform a syntax check. After the above command is executed, if any error is displayed, correct the program and execute build again.

10. From the menu bar, select **File**>**Save Project and Install into Library Repository**.

The library that has been created will be installed in the library repository.

11. From the menu bar, select **Tools**>**Library Repository**.

The "Library Repository" dialog box will be displayed. Check that the created library is displayed in the "Installed library" section.



Clicking the [Detail] button enables the user to check information such as function blocks included in the library.

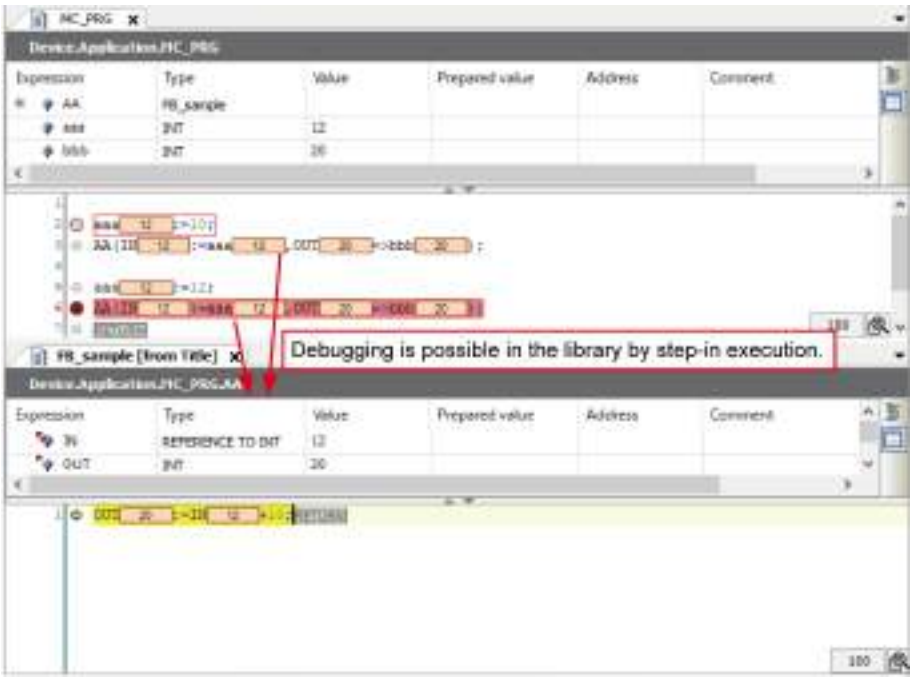
This completes the procedure for installing the library in the library repository.

Info.

- Difference between libraries (.library) and complied libraries (.compiled-library).

  If a library is installed as a library file into the repository, the user can refer to codes by step-in execution during debugging and check execution details.

  The user cannot execute codes in any compiled library while referring to the codes.



- Method for installing a compiled library

  In Step 10, select **File>Save Project as Complied Library** from the menu. The library is saved as a complied library file (.compiled-library).

  Then, click the [Install] button in the "Library Repository" dialog box and select the saved complied library file.

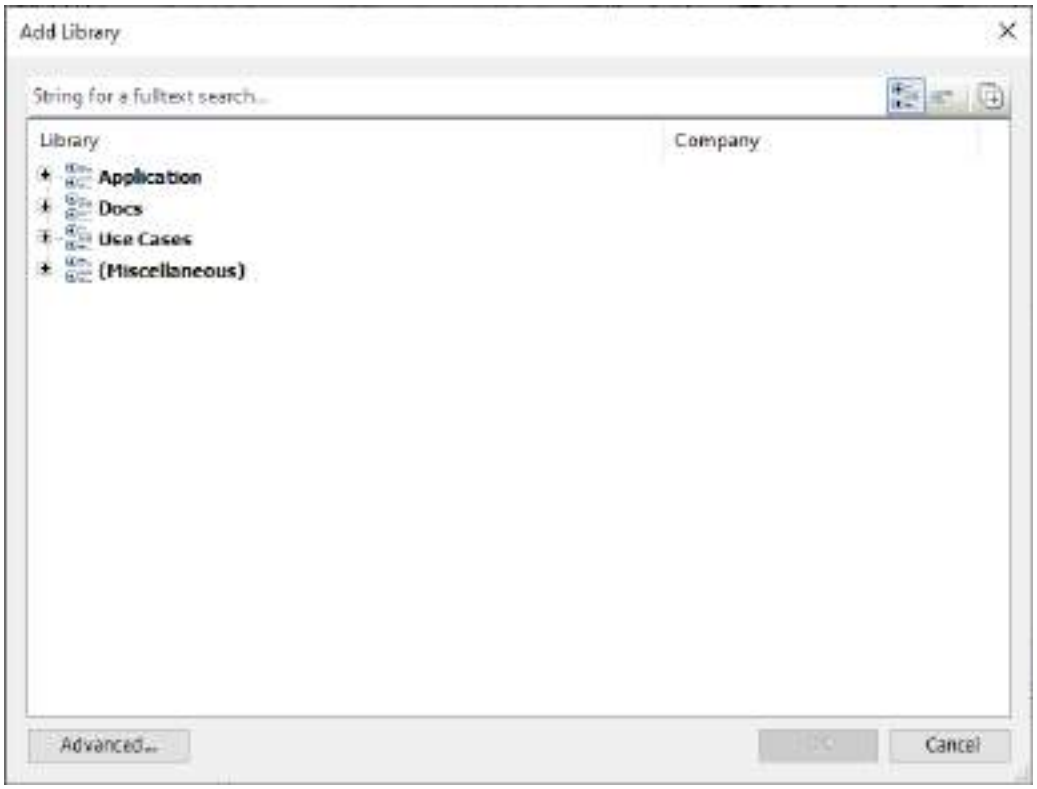
10.6.2 Using Created Libraries

This section explains how to add libraries installed in the library repository to the project.

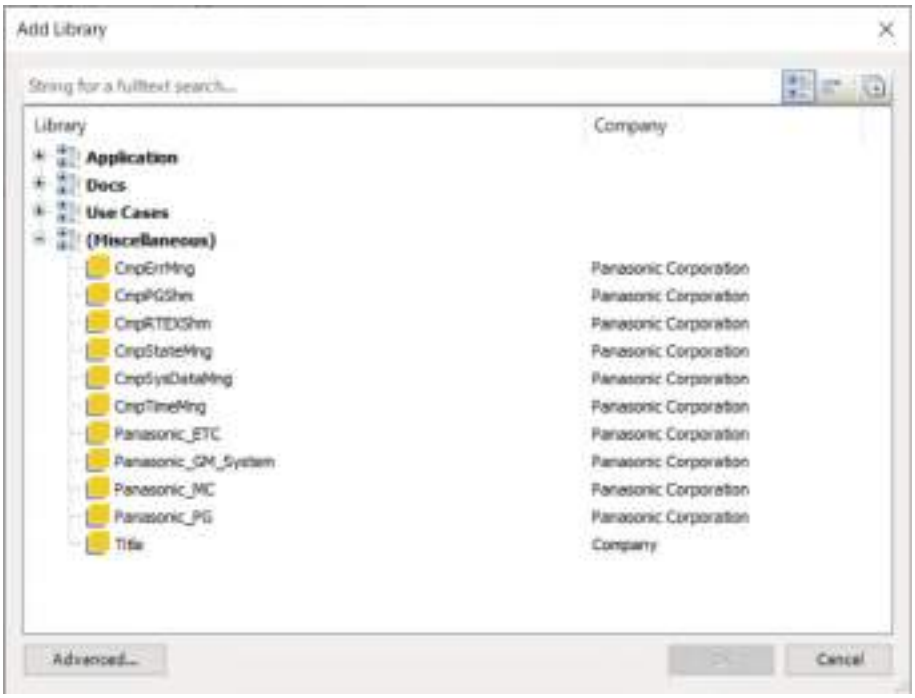
Procedure

1. Open the project file and select the "Library Manager" object.
   The "Library Manager" window will be displayed.

2. Click the [Add Library] button.
   The "Add Library" dialog box will be displayed, showing the libraries added to the library repository.



3. Select a created library and click the [OK] button.
The selected library will be added to the application in the project.
The title and company name specified when the library was created will be displayed.



This completes the procedure for adding the library to the application.
The function blocks in the added library can be used in the program.

10.7 POU for implicit checks

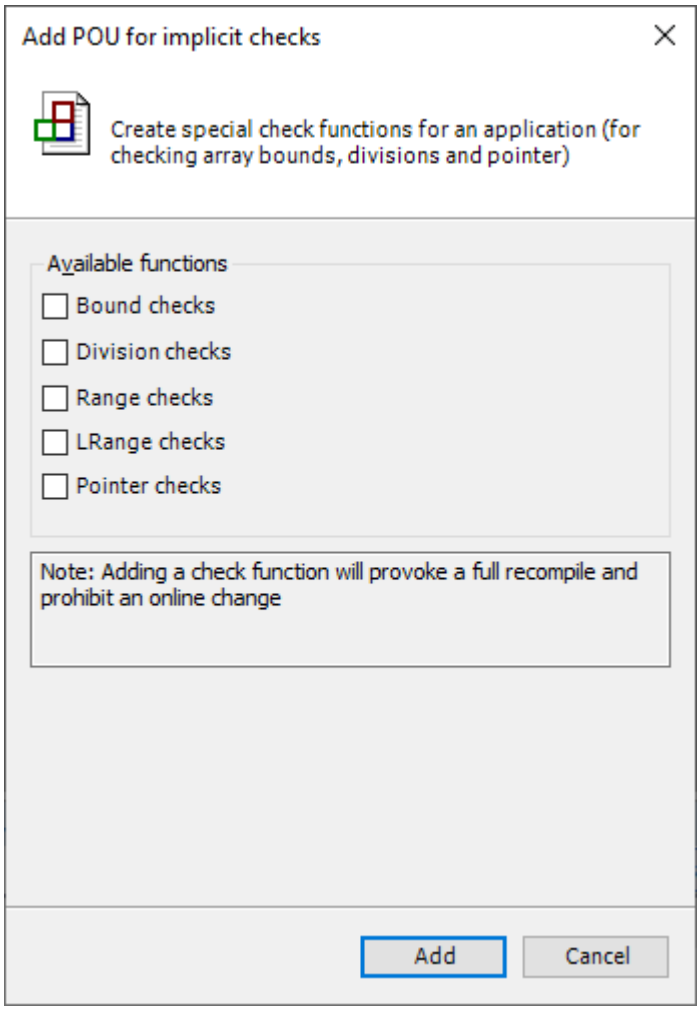
POU for implicit checks is an object with functions that check the range of array indexes or subrange type variables, validity of pointer addresses, and division by zero. Adding this object to the project makes it possible to automatically call these functions and perform checks, without calling the functions explicitly within the program.

10.7.1 Setting up POU for implicit checks

This section explains how to add POU for implicit checks objects.

Procedure

1. Right-click the [Application] object in the navigator pane and then select **Add Object>POU for implicit checks** from the context-sensitive menu that is displayed.
   The "Add POU for implicit checks" dialog box will be displayed.



2. Select the check box of a function that you want to add.
   Multiple functions (objects) will be added, depending on the item that you select.

Check type	Function name (object name)
Bound checks	CheckBounds
Division checks	CheckDivDInt
	CheckDivLInt
	CheckDivLReal
	CheckDivReal
Range checks	CheckRangeSigned
	CheckRangeUnsigned
LRange checks	CheckLRangeSigned
	CheckLRangeUnsigned
Pointer checks	CheckPointer

3. Click the [Add] button.
   The function (object) for the selected item will be added.
   If necessary, edit the implementation section of the object that has been added.

The check details of each check item are as below.

Bound checks

Checks whether boundaries are violated.

Division checks

Checks whether anything is divided by zero.

Range checks

Checks whether values of DINT or UDINT subrange type variables are within the specified range.

LRange checks

Checks whether values of LINT or ULINT subrange type variables are within the specified range.

Pointer checks

Checks whether the returned pointer refers to a valid memory address or whether the contents of a memory address that is referred to match the variable type that refers to the pointer.

For Pointer Checks, a program must be created for the function. Update the program by referring to online help.

10.8 Interface

The interface object defines common methods and properties that are used between different function blocks in the same way.

The interface object is one of the means of implementing object-oriented programming.

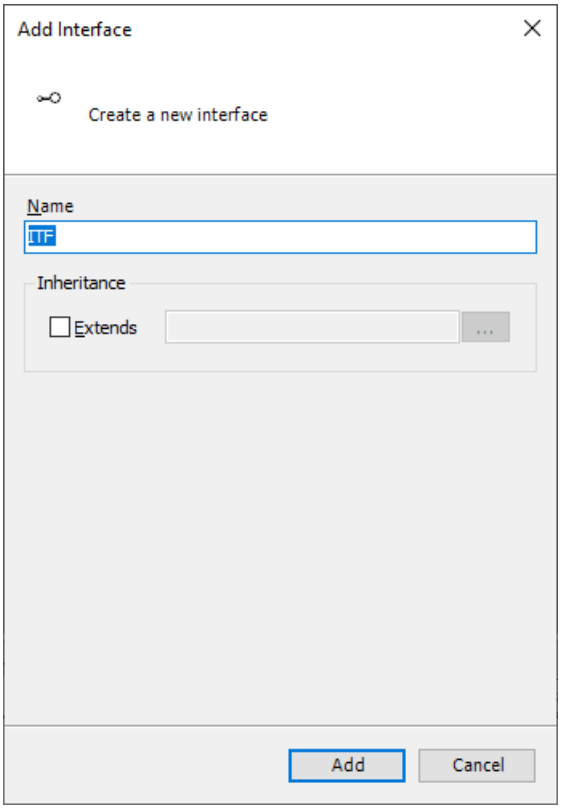
The interface object contains only method and property declarations but does not contain implementation.

10.8.1 Setting up an Interface Object

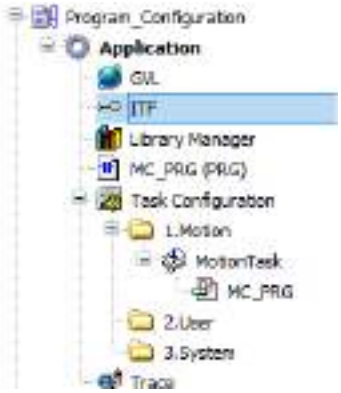
This section explains how to add an interface object.

Procedure

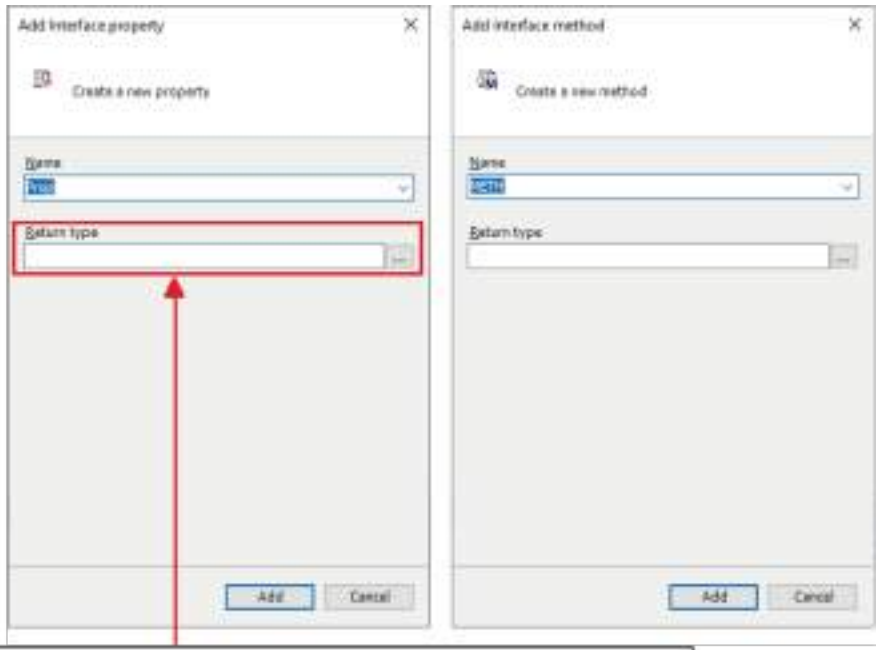
1. Right-click the "Application" object in the navigator pane and then select **Add Object>Interface** from the context-sensitive menu that is displayed.
   The "Add Interface" dialog box will be displayed.



2. Enter a name and click the [Add] button.
   An interface object will be added.
   If the "Extends" check box is selected, the interface entered in the input field can be inherited and extended.

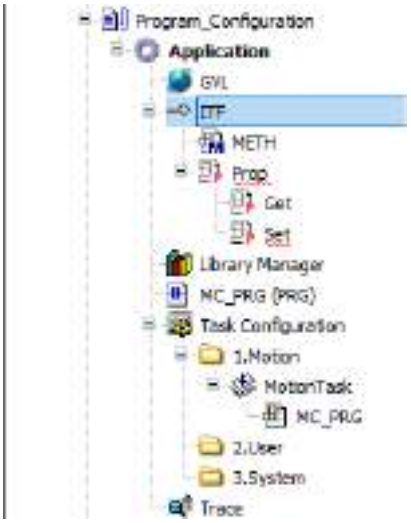


3. Right-click the "ITF" object added to the navigator pane and then select **Add Object**>**Interface property** or **method** from the context-sensitive menu that is displayed.
The "Add Interface property" dialog box or "Add Interface method" dialog box will be displayed.



* For interface properties, the data type of return values must be entered.

4. Enter a name and click the [Add] button.
For interface properties, be sure to enter a value in the "Return type" field.
"Prop" or "METH" object will be added under the "ITF" object.



5. The added "METH" and "Prop" objects are used to define methods and properties, respectively.

   This completes the procedure for creating an interface object.

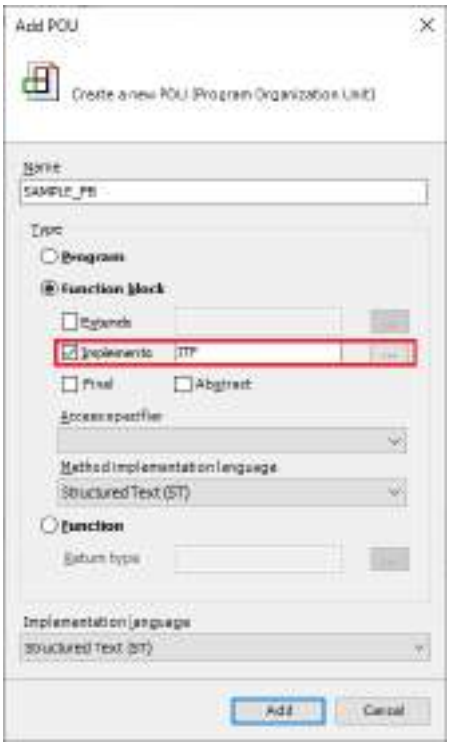
10.8.2 Implementing in New Function Block

This section explains how to implement an interface in a new function block to be created.

Procedure

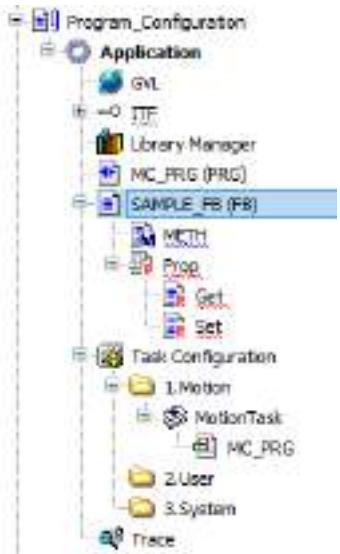
1. Right-click the "Application" object in the navigator pane and then select **Add Object>POU** from the context-sensitive menu that is displayed.

   The "Add POU" dialog box will be displayed.



2. Enter a name. In the Type section, select the Function block option, select the "Implements" check box, and enter an interface to be implemented. Click the [Add] button.

A function block with the interface implemented will be added under the "Application" object.



3. Open the respective editors for the added "METH" and "Prop" objects, and implement internal processing for the methods and properties.

Note

- If methods and properties are added under the "interface" object later, they will not be automatically added to the function block with the interface implemented. Therefore, if they need to be added, perform the procedure starting from "Step 3" 3 in "10.8.3 Implementing in Existing Function Block".

10.8.3 Implementing in Existing Function Block

This section explains how to implement an interface in an existing function block.

Procedure

1. Open the editor of the existing function block from the navigator pane.

Character string format

Table format

2. For character string format, add "IMPLEMENTS <interface name>" to the declaration header section. For table format, open the "Edit Declaration Header" dialog box, enter an interface name in the "IMPLEMENTS" field, and click the [OK] button.

Character string format

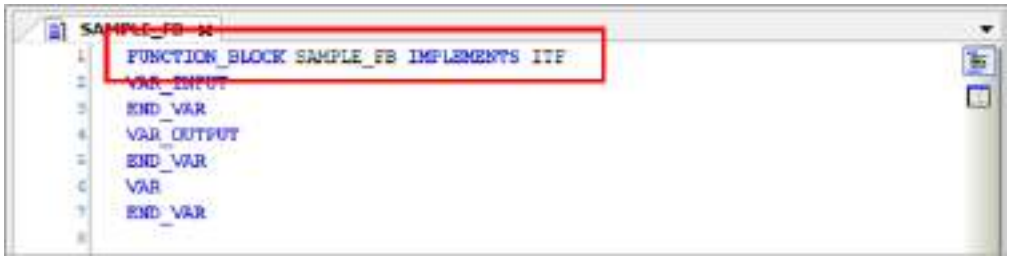
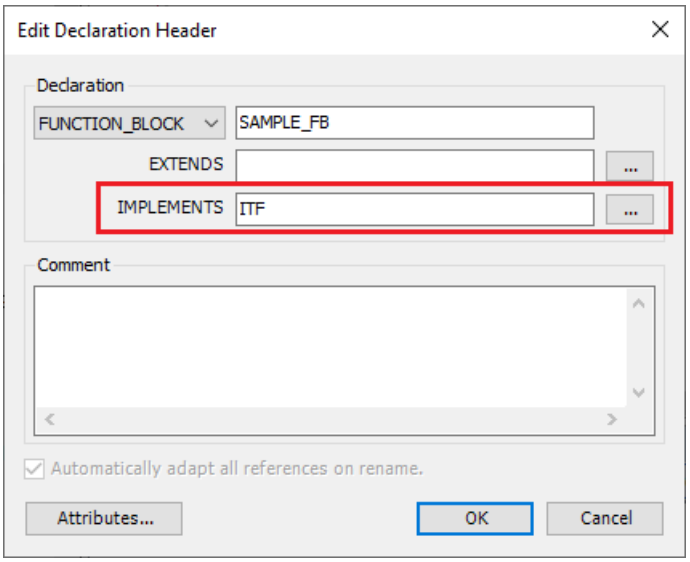
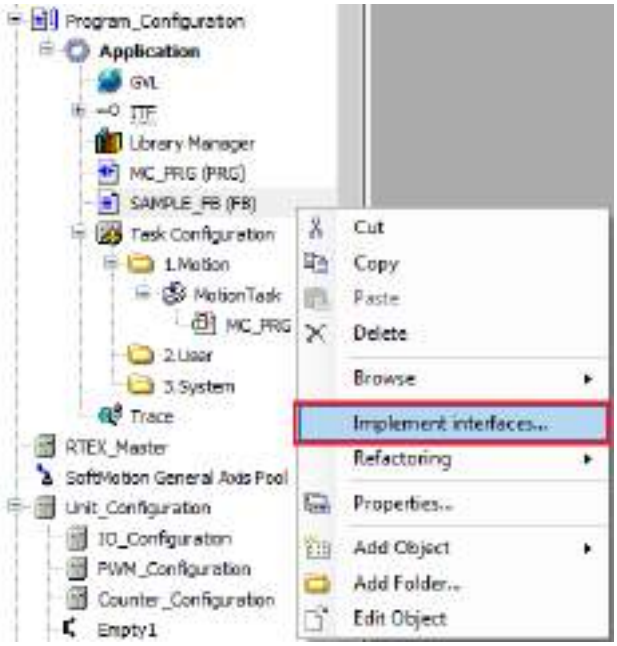
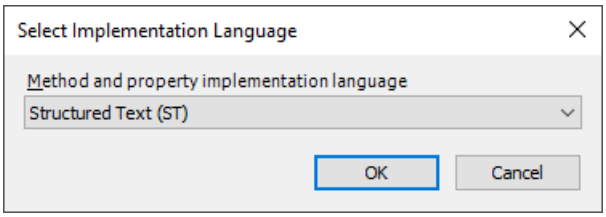


Table format

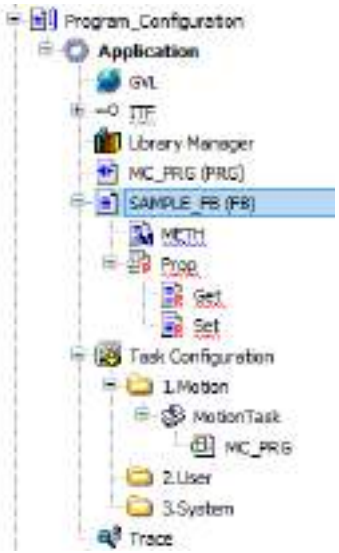


3. Right-click the existing function block in the navigator pane and select "Implement Interfaces" from the context-sensitive menu that is displayed.
The "Select implementation language" dialog box will be displayed.





4. Select a desired programming language and click the [OK] button. The method and property objects will be added under the function block object.



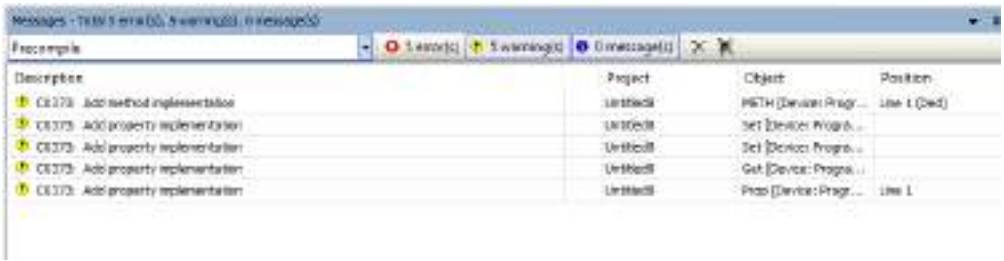
5. Open the respective editors for the added "METH" and "Prop" objects, and implement internal processing for the methods and properties.

Note

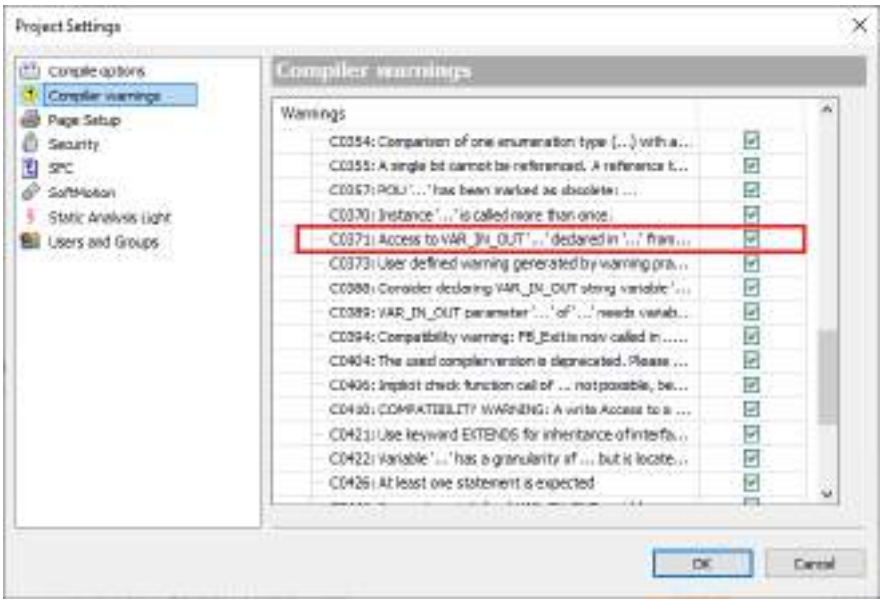
- If methods and properties are added under the "interface" object later, they will not be automatically added to the function block with the interface implemented. Therefore, if they need to be added, perform the procedure starting from Step "Step 3".

Info.

- When methods and properties are implemented, warning messages are automatically implemented. Therefore, when compilation is executed, the following warning messages are displayed.



These warning messages do not indicate any problems. However, to prevent particular warning messages from being displayed, from the menu bar, select **Project**>**Project Settings** and then clear the check boxes of the target warning messages in the Compiler Warnings pane.



10.8.4 Extending the Interface

Existing interfaces can be inherited and extended.

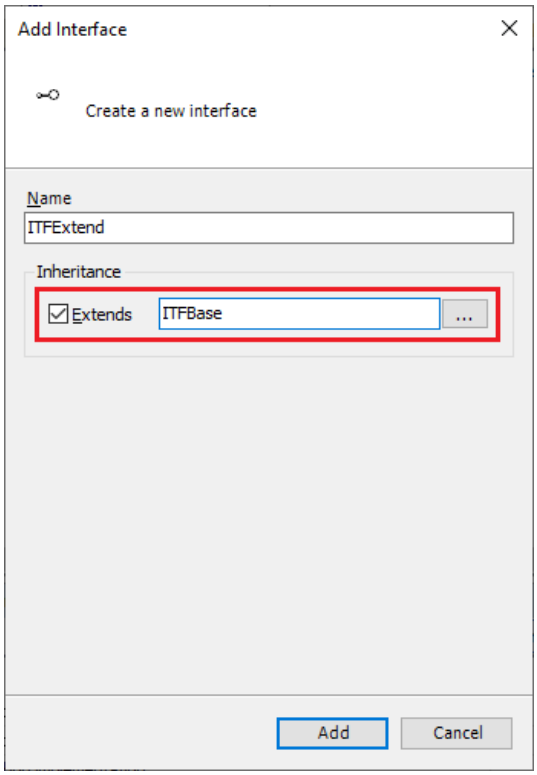
This section explains how to extend existing interface "ITFBase" and create new interface "ITFExtend".

In this example, when interface "ITFBase" exists, create a new interface as below.

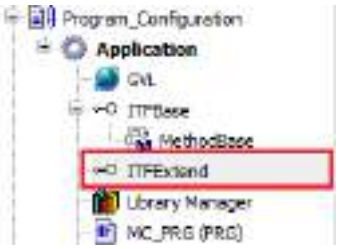
Procedure

1. Right-click the "Application" object in the navigator pane and then select **Add Object**>**Interface** from the context-sensitive menu that is displayed.

The "Add Interface" dialog box will be displayed.

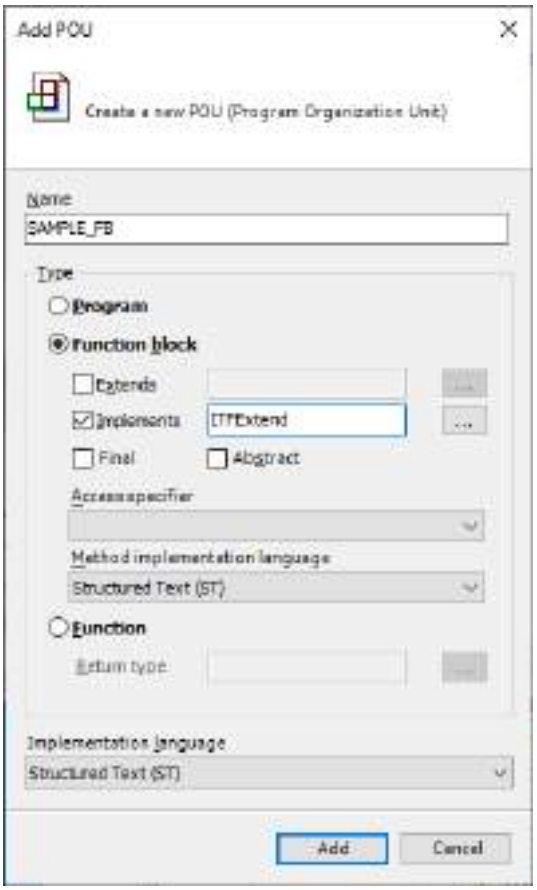


2. In the "Add Interface" dialog box, select the "Extends" check box and enter an interface to be inherited.
   Interface "ITFExtend" will be created.



3. Right-click the "ITFExtend" object added to the navigator pane and then select **Add Object>Interface property** or **method** from the context-sensitive menu that is displayed in order to add a property or method for the "ITFExtend" object.

4. Right-click the "Application" object in the navigator pane and then select **Add Object>POU** from the context-sensitive menu that is displayed.
   The "Add POU" dialog box will be displayed.



5. Enter a name. In the Type section, select the Function block option, select the "Implement" check box, and enter "ITFExtend". Click the [Add] button.

A function block with properties and methods for both interfaces "ITFExtend" and "ITFBase" will be added under the "Application" object.

10.9 External File Functions

The external file object allows text files, image files, and other files to be saved in the project.

10.9.1 Setting up an External File Object

This section explains how to add an external file object.

Procedure

1. Right-click the "Application" object in the navigator pane and then select **Add Object**>**External File** from the context-sensitive menu that is displayed.
   The "Add External File" dialog box will be displayed.

2. In the File path field, specify a file to be registered. In the Name field, enter the name of the file.
   In the "What do you want to do with the external file?" section, select an appropriate option as the method for registering the external file in the project.
   If you select the "Remember the link and embed into project" option in the "What do you want to do with the external file?" section, select an appropriate option in the "When the external file changes, then" section to specify settings for update processing to be performed when the external file is changed.
   "What do you want to do with the external file?":

   "Remember the link"
   Stores only the path to the file in the project.
   If the file does not exist at the link destination, it cannot be used in the project.

"Remember the link and embed into project"

Saves a copy of the file and a link to the file to the project at the same time.

As long as the file exists at the link destination, the setting of the "When the external file changes, then" section applies.

If the file does not exist, a version of the file that is stored in the project will be used.

"Embed into project"

Saves a copy of the file to the project.

"When the external file changes, then":

"reload the file automatically"

Updates a copy of an external file in the project when the corresponding external file is changed.

"prompt whether to reload the file"

Opens a dialog box, asking whether to update a copy of an external file in the project, when the corresponding external file is changed.

"do nothing"

Does not update a copy of an external file in the project even if the corresponding external file is changed.

3. Click the [Add] button.

The external file will be added under the "Application" object.

Info.

- When the added external file is opened in GM Programmer, a copy of the file is temporarily created in the following folder:

  C:\Users\<user name>\AppData\Local\Temp
- Files added as "external file" objects cannot be accessed from programs such as POU.

10.10 Servo Amplifier / Motor Operation Function (PANATERM Lite for GM)

You can start PANATERM Lite for GM, which allows you to check the settings of servo amplifiers, the operating states of servo amplifiers and motors, and the error status of servo amplifiers via the GM1 controller.

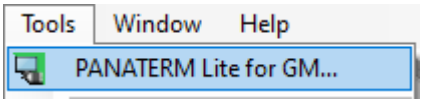
For details, refer to "15 Overview of PANATERM Lite for GM".

10.10.1 Starting PANATERM Lite for GM

This section explains how to start PANATERM Lite for GM from GM Programmer.

Procedure

1. From the menu bar, select **Tools**>**PANATERM Lite for GM**. PANATERM Lite for GM will be started.



11 Motion Control

11.1 EtherCAT Master Setting 11-3
11.1.1 EtherCAT_Master_SoftMotion 11-3
11.2 EtherCAT Slave Settings 11-6
11.2.1 EtherCAT Slave Settings 11-6
11.3 EtherCAT Axis Settings 11-12
11.3.1 Overview of EtherCAT Axis Settings 11-12
11.3.2 Basic Settings for EtherCAT Axes 11-12
11.3.3 EtherCAT Axis Extended Setting 11-16
11.4 Basic Preparations for Operation 11-23
11.4.1 Overview of Basic Preparations for Operation 11-23
11.4.2 Servo ON or OFF 11-23
11.4.3 Home Return 11-24
11.4.4 JOG Operation 11-36
11.5 Single-axis Operation 11-38
11.5.1 Overview of Single-axis Operation 11-38
11.5.2 Position Control 11-38
11.5.3 Switching the Control Mode 11-45
11.5.4 Velocity Control 11-46
11.5.5 Torque Control 11-48
11.5.6 Stop 11-50
11.6 Synchronous Operation 11-52
11.6.1 Synchronous Cam Operation 11-52
11.6.2 Synchronous Gear operation 11-58
11.7 Multi-axis Operation 11-61
11.7.1 Overview of Interpolation Control 11-61
11.7.2 Linear Interpolation and Circular Interpolation 11-62
11.7.3 How to Use Interpolation Control 11-63
11.7.4 Registering a CNC Table 11-64
11.7.5 Overview of G-code 11-75
11.7.6 G-code Editor and Coding Rules 11-77
11.7.7 Movements Executed by Each G-code and Setting Methods 11-78
11.7.8 SMC_CNC_REF and SMC_OUTQUEUE 11-90
11.7.9 Interpolation Operation Programming: How to Create a Program for Executing Operation 11-91
11.7.10 Interpolation Operation Programming: Explanation of Function Block (FB) 11-94
11.7.11 Interpolation Operation Programming: Specifying the Starting Coordinates 11-97

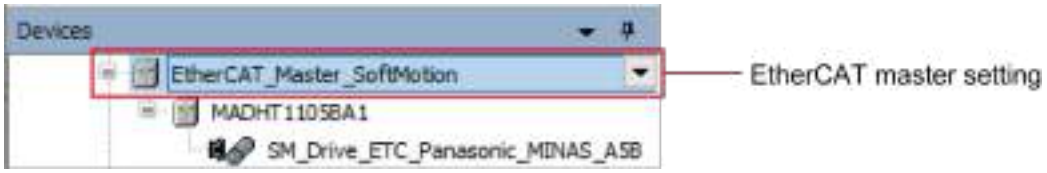
11.7.12 Interpolation Operation Programming: P-point Control and C-point Control 11-101
11.7.13 Interpolation Operation Programming: Settings in CNC Table for C-point Control 11-103
11.7.14 Interpolation Operation Programming: Settings in POU for P-point Control and C-point Control 11-104
11.7.15 Interpolation Operation Programming: Joining and Repeating CNC Tables 11-105
11.7.16 Interpolation Operation Programming: Changing Parameter Settings (Converting to Variables in CNC Table) 11-108
11.8 Motion Function Errors 11-110
11.8.1 Overview of Motion Function Errors 11-110
11.8.2 Error Check Method 11-111
11.8.3 Clearing Errors 11-112

11.1 EtherCAT Master Setting

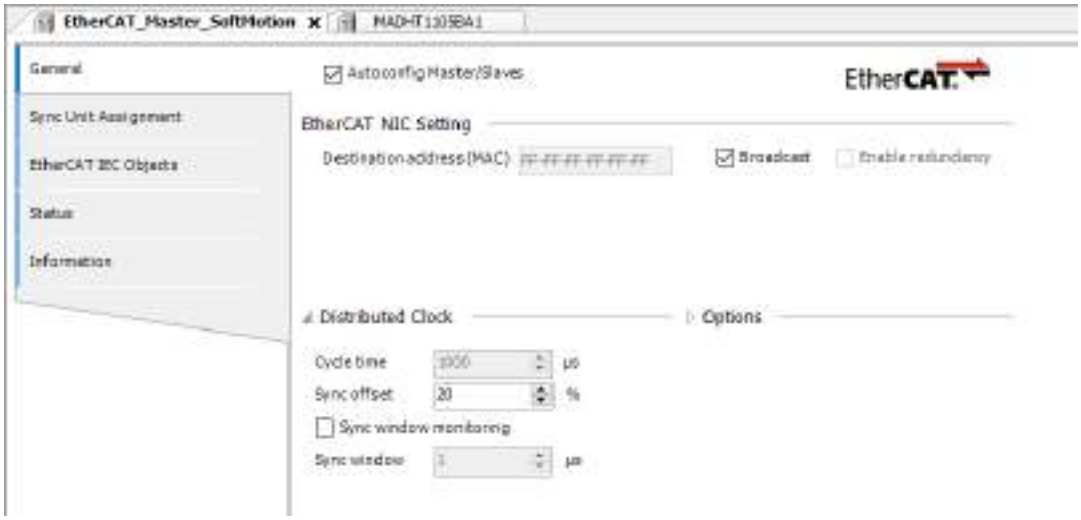
11.1.1 EtherCAT_Master_SoftMotion

■ **Overview**

The EtherCAT master must be configured as described below.
This section explains various tabs and settings of EtherCAT_Master_SoftMotion editor.



Set EtherCAT Master parameters on the General window.



List of parameters

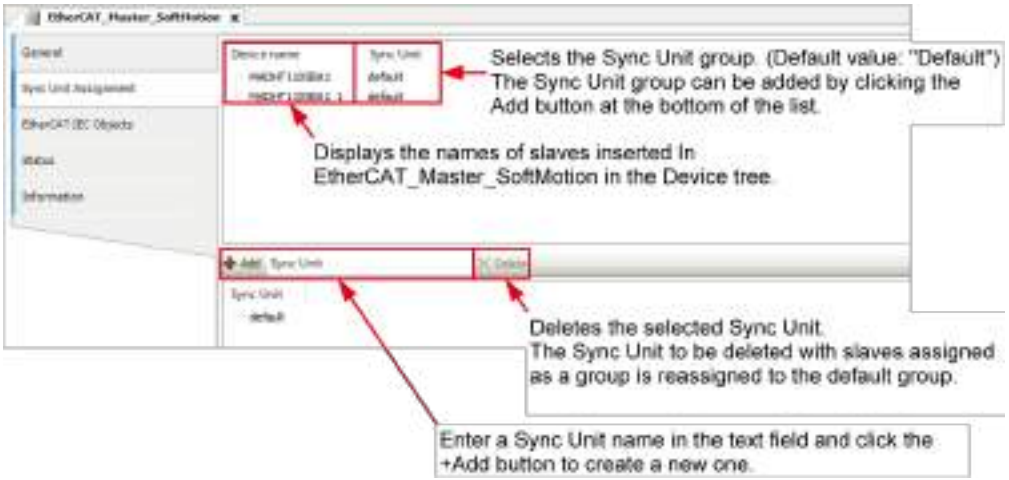
Name		Description
Automatic Master / Slave Configuration		When this parameter is enabled, it automatically performs configuration of the master and slave. When this parameter is disabled, all configuration settings for the master and slave must be made manually. Since configuration settings require expertise, it is recommended to set to automatic settings.
EtherCAT NIC Setting	Storage location address (MAC)	Specifies the destination address.
	Broadcast	When this setting is enabled, a broadcast address (FF-FF-FF-FF-FF-FF) is set in the storage location address (destination). When this setting is disabled, set a multicast address in the storage location address (destination).
Distributed Clock	Cycle Time	Sets the EtherCAT communication cycle. The MotionTask cycle is set.

Name		Description
	Synchronous Offset	Sets the delay time between the DC reference time of the EtherCAT slave and the EtherCAT task (PLC's cycle start time).
	Synchronous Window Monitoring	When this setting is enabled, it monitors slave synchronization.
	Synchronous Window	Sets the synchronous window monitoring time When all slave's synchronizations occur within the set time, xSyncInWindow (IoDrvEtherCAT) turns to TRUE.
Options	Use LRW instead of LWR / LRD.	When this setting is enabled, the LRW command (read/write) is used. When this setting is disabled, the LRD command (read) and LWR command (write) are used as separate commands.
	Enable message for every task.	When this setting is enabled, the PDO mapped variables are updated at the point of use of the POU. When this setting is disabled, the PDO mapped variables are updated by a MotionTask.
	Autostart slave.	When a communication breakdown occurs, automatically the master attempts to restart the slave immediately.
Master Configuration		It can be set only when Automatic Master / Slave Configuration is disabled.
	Image In Address	Sets the first logical address of the first slave used for PDO input data.
	Image Out Address	Sets the first logical address of the first slave used for PDO output data.

■ Sync Unit Assignment

In Sync Unit Assignment, multiple slaves can be grouped together and managed as a Sync Unit.

For example, one slave within a Sync Unit becomes unknown, the rest of the slaves in the Sync Unit are displayed as unknown. Since other Sync Units not affected remain operable, the unknown group can be promptly corrected using the device diagnostics.



■ Status tab

Diagnostic messages of the EtherCAT_Master_SoftMotion device are displayed. They can be checked with the program using the "GetStatistics" method of "IoDevEtherCAT".

Item	Description
SendFrameCount	The number of EtherCAT send frames is displayed.
FramesPerSecond	The number of send frames per second is displayed.
LostFrameCount(Note 1)	The number of frame losses is displayed.
TxErrorCount	The number of send errors is counted.
RxErrorCount	The number of receive errors is counted.

(Note 1) Check the following when the number of frame losses is other than 0.

- Influence of noise
- Connection status with slaves
- Program load

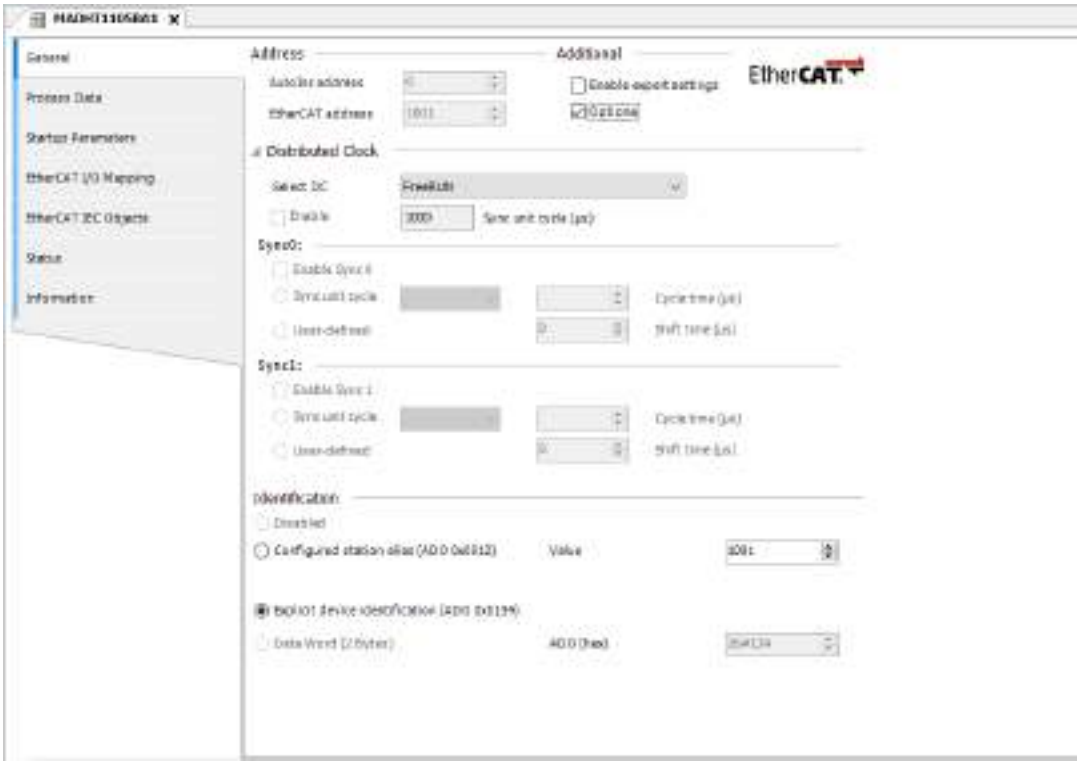
11.2 EtherCAT Slave Settings

11.2.1 EtherCAT Slave Settings

■ **Overview**

The EtherCAT slave must be configured as described below. The description below explains how to set the servo amplifier (A5B: MADHT1105BA1) manufactured by Panasonic.

Set EtherCAT slave parameters on the General window. In the Sync Unit cycle, the cycle set in MotionTask is set.



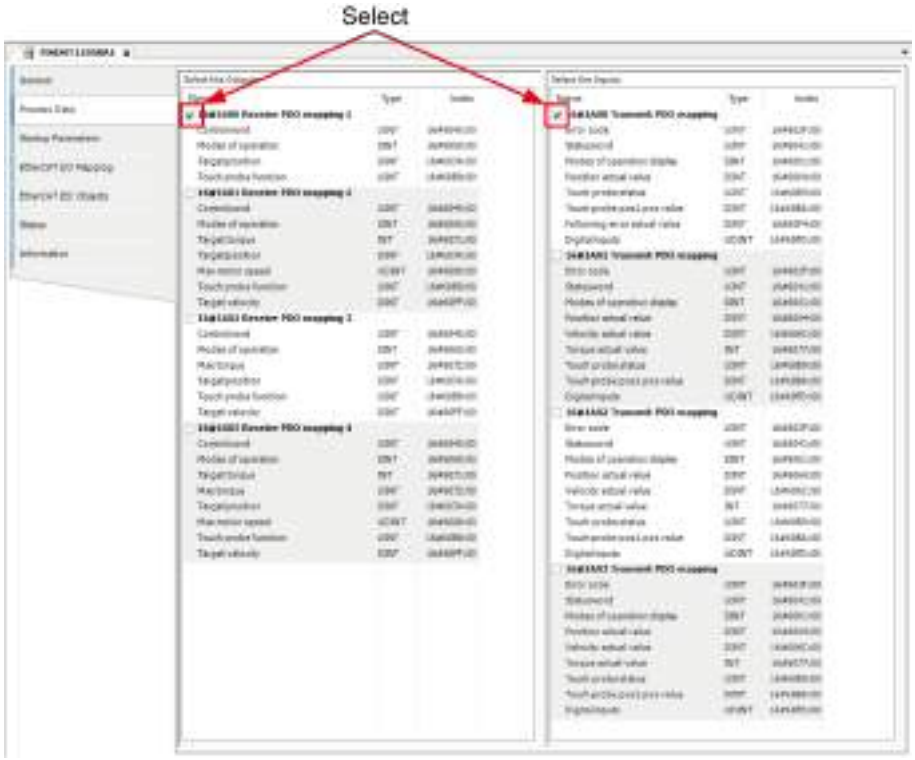
List of parameters

Name		Description
Address		It can be set only when "Automatic Master / Slave Configuration " is disabled in EtherCAT master setting.
	AutoInc address	Sets a tentative address used before the master assign an EtherCAT address to the slave.
	EtherCAT address	Sets an EtherCAT address to the slave.
Add	Enable expert settings	When this setting is enabled, expert setting can be made. Since expert settings are not required for standard applications, it is recommended to disable the settings.
	Option	When this setting is enabled, the slave is defined as an option. As a result, an error message is not generated even when a device is

Name		Description
		not found in the bus system. When this setting is enabled, assign a unique ID in the setting for the identification section. ● Requirements for use • The "Automatic Master / Slave Configuration".parameter on the "General" tab of EtherCAT_Master_SoftMotion editor is enabled. • The EtherCAT slave supports this function.
Distributed Clock	DC selection	Sets the synchronization method between the master and the slave. Example: ● FreeRUN: Non-synchronize with master's cycle time. ● DC Sync0: Based on the time of the 1st axis, synchronize time information of other slaves.
Identification	Disable	Slave identification is not checked.
	Configured station alias value	Sets the address stored in the slave device EEPROM.
	Explicit device identification	Sets a device identification number for hardware such as a DIP switch of the slave device.
	Data word (2 bytes)	A 2-byte identification number is saved. (MINAS: Not supported)

■ Process data

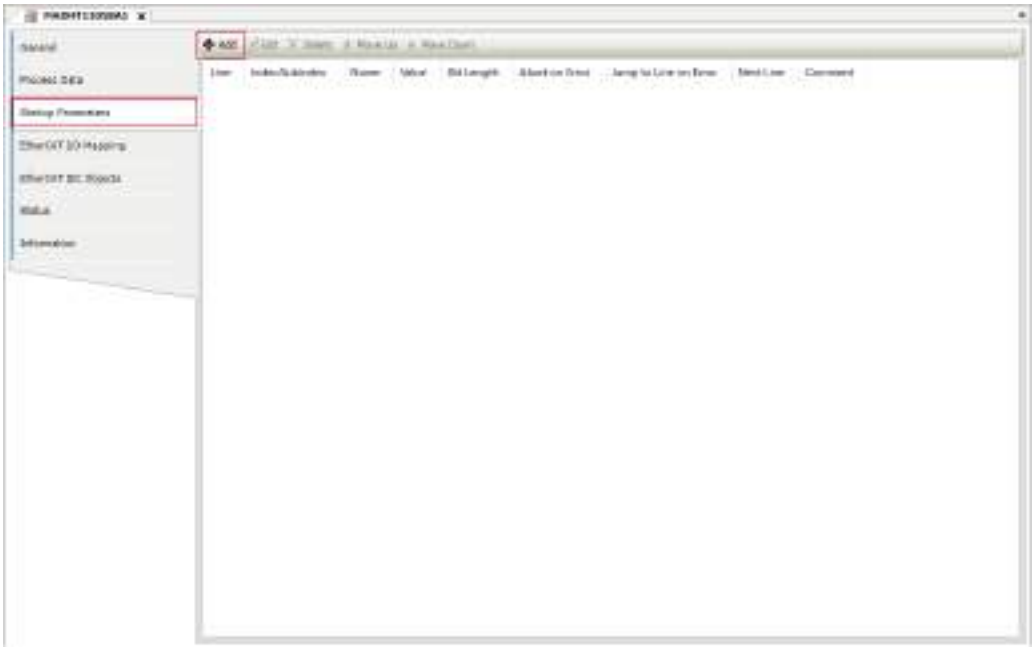
PDO (Process Data Object) is data that is updated between the master and the slave for every EtherCAT communication cycle. The configuration of the PDO varies depending on the slave. Select an appropriate one for each axis control method. The selected PDO mapping is reflected on the "EtherCAT I/O mapping " and can be used in the program.



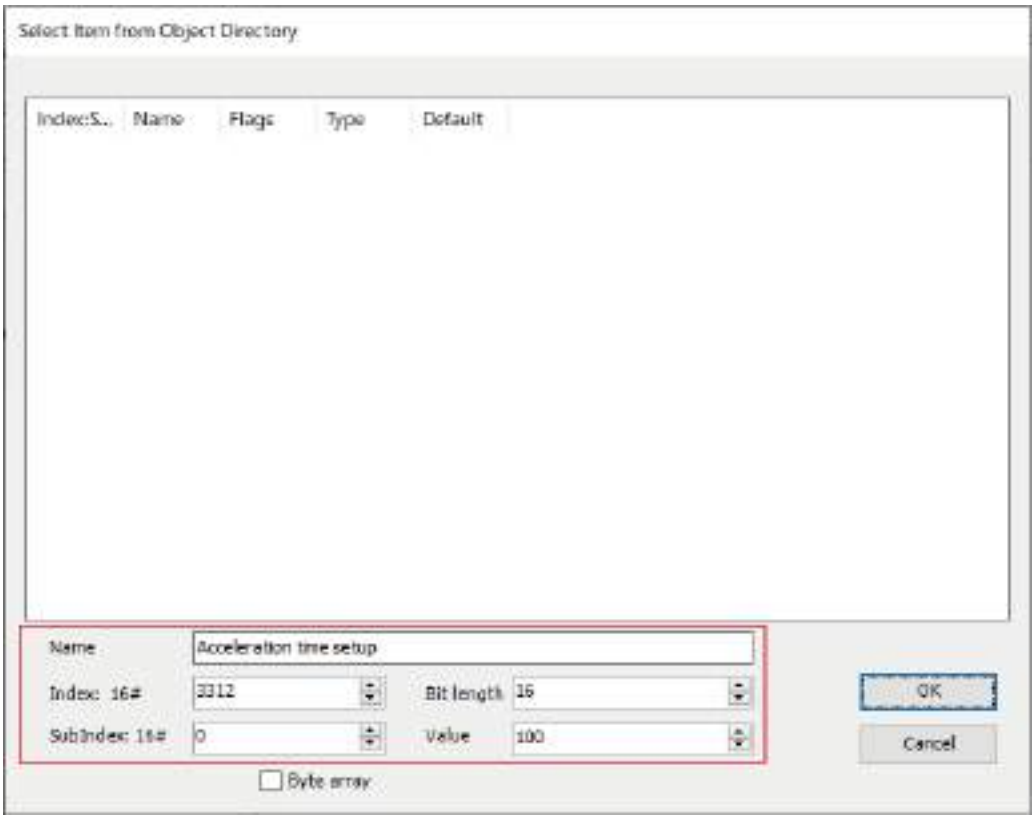
■ **Startup parameter**

Using the SDO (Service Data Object), slave parameters can be set at the start of RUN.

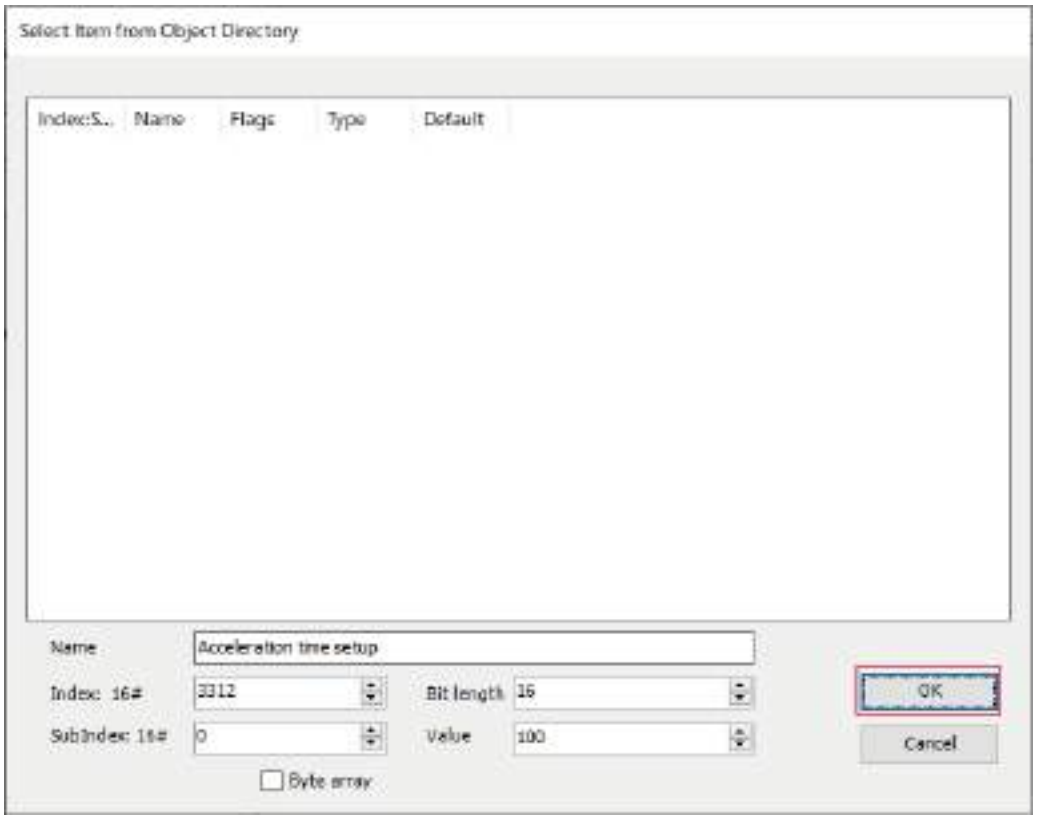
1. Click "Add".



2. Enter object information to be changed.
   The following explanation is provided for an example where "Acceleration time setting" is changed.
   - Name: Any name
   - Index 16#: 3312
   - Sub-index 16#: 0
   - Bit length: 16
   - Value: 100



3. Click the [OK] button.



The data will be displayed as follows.

■ Online

When set to online, the status of the ESM (EtherCAT State Machine) can be changed and displayed, firmware files can be written or read, and configuration information can be written or read to EEPROM.

- Requirements for use
  - "Enable expert settings" on the "General" tab is enabled.
  - You are already logged in to the controller.

Note

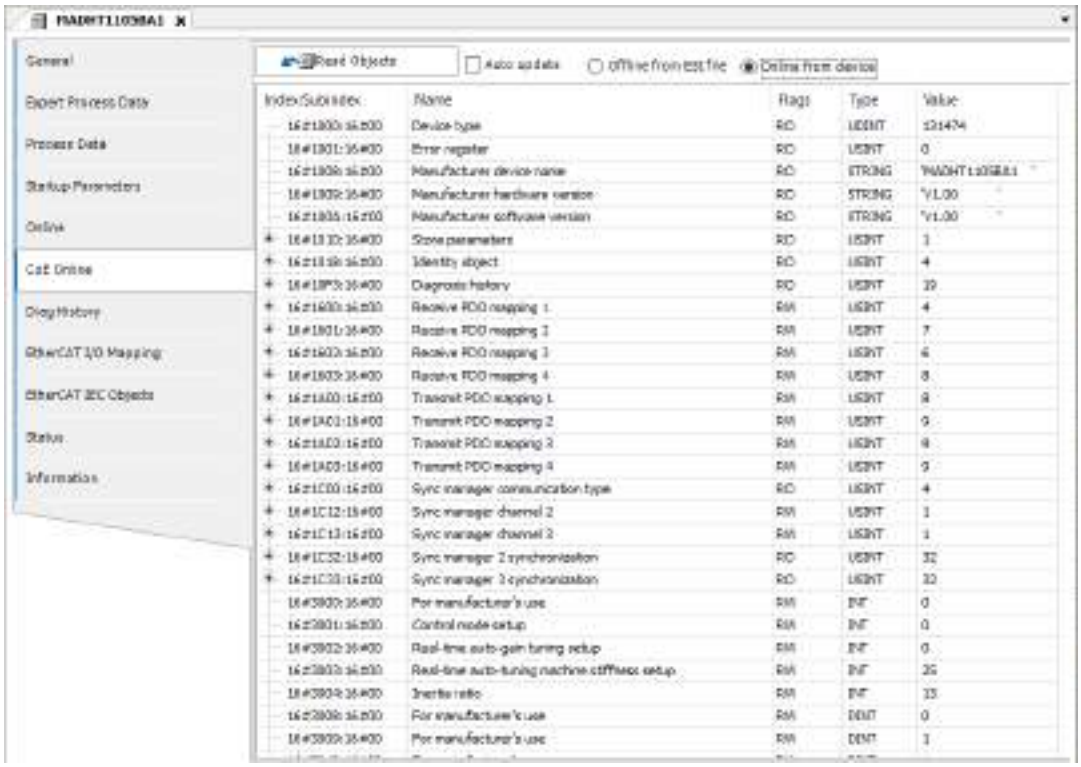
- Writing a wrong file or incorrect configuration information may cause the slave to operate in an abnormal manner. Be careful when using the online mode.

■ CoE online tab

Set values can be written or read from the ESI file or the object dictionary of the slave device.

- Requirements for use
  - "Enable expert settings" on the "General" tab is enabled.

- You are already logged in to the controller.



11.3 EtherCAT Axis Settings

11.3.1 Overview of EtherCAT Axis Settings

To use the motion function, you must configure axis settings for EtherCAT.
This section explains how to set axis information for EtherCAT in GM Programmer.

Info.

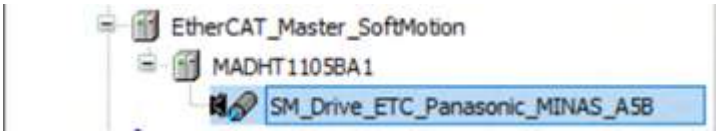
- For details on how to set up EtherCAT and add axes, refer to "5.3 Setting up Motion Control".

11.3.2 Basic Settings for EtherCAT Axes

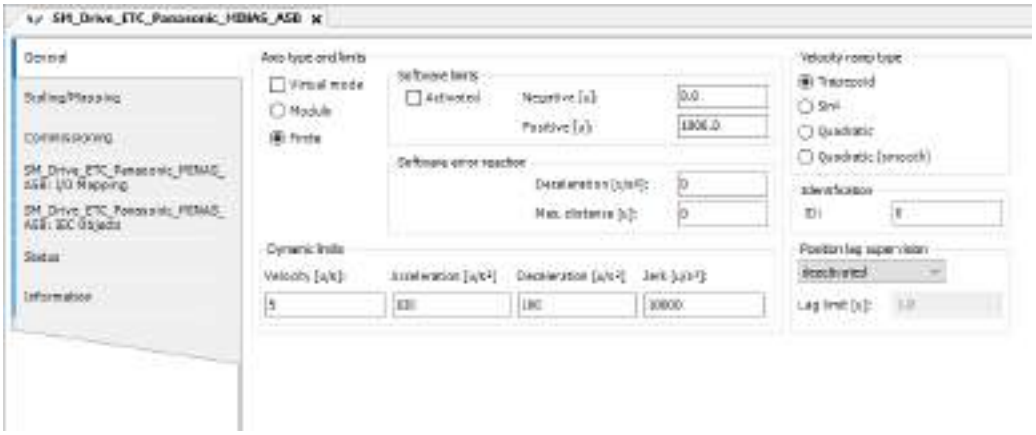
- Be sure to configure settings for EtherCAT axes.

Procedure

1. Double-click the servo amplifier object in the navigator pane.

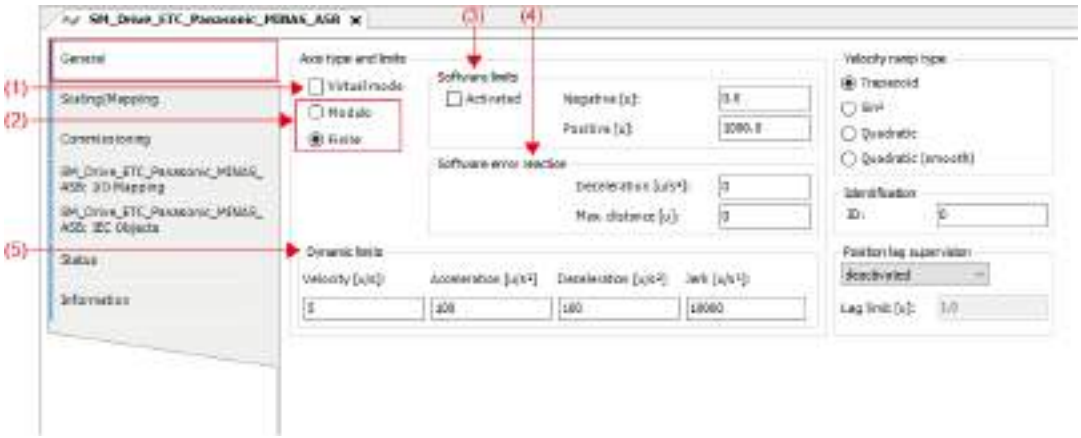


The setting pane will be displayed in the main pane.



General Settings

Select the "General" tab and set the following items.



(1) Virtual mode

You can set real axes or virtual axes.

Use of real axes: Real axes are used to actually control the servo amplifier.

Use of virtual axes: Virtual axes create and execute a virtual servo amplifier within the GM1 Controller.

(2) Modulo / Finite

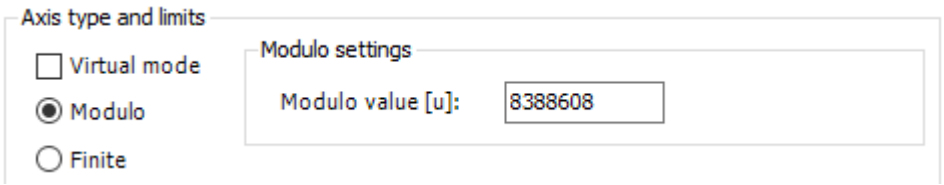
The axis type can be specified.

- Modulo

  Modulo: The motor rotates infinitely (belt drive, etc.) without limiting the travel range.

  - The value of the commanded position keeps looping between 0 and modulo value.
  - The maximum settable modulo value is "255×units in application" (*1).

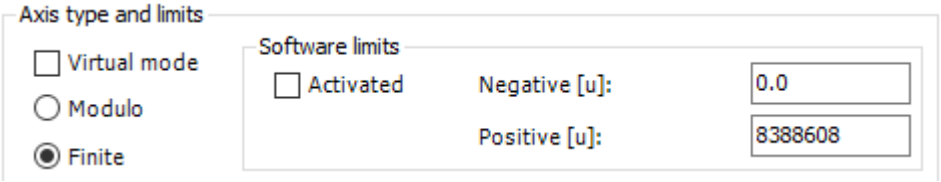
    *1: For "units in application", specify settings in the "Scaling / Mapping" tab.
  - A negative value cannot be set. (A warning is issued. If a download is performed without correcting the value, an error will occur when the GM1 Controller is started.



- Finite

  The set value of the commanded position is a finite value.

  Soft limits can be set. Note that an error will occur if a 32-bit real number is exceeded.

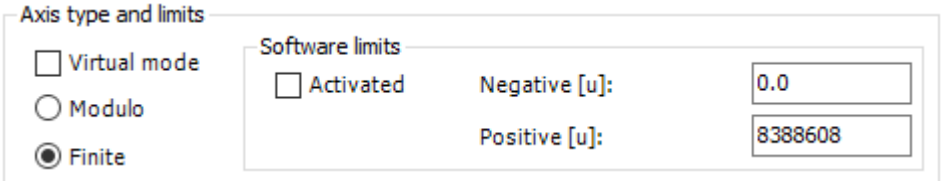


(3) Soft limits

If the axis type is set to "Finite", soft limits can be set.

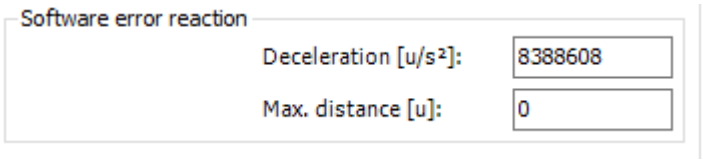
If the commanded position falls outside the soft limit setting range, an error stop will occur, causing the operation to stop.

If the operation is stopped because the soft limit setting range is exceeded, the value specified in "Deceleration" or “Max. distance" in "Soft error reaction" or the value specified in "Deceleration" in "Dynamic limits" will be applied, whichever is the shortest time from when deceleration starts until a stoppage occurs.



(4) Software error reaction

Settings can be configured to stop operation when an error occurs.

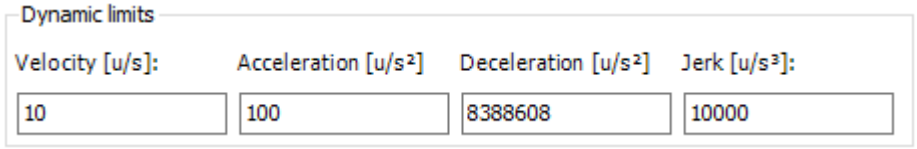


Info.

- If the mode is switched from run to stop during operation, an emergency stop will be executed, regardless of the settings in "Software error reaction".
- For stop operation that takes place when an error stop occurs or when the soft limit range is exceeded, one of the values specified in the following items is applied, whichever is the shortest time from when deceleration starts until a stoppage occurs.
  - "Deceleration" in "Software error reaction"
  - "Max. distance" in "Software error reaction"
  - "Deceleration" in "Dynamic limits"
- If "Deceleration" and "Max. distance" in "Software error reaction" are set to 0, these settings will be disabled. In this case, operation is stopped according to the value specified in "Deceleration" in "Dynamic limits".

(5) Dynamic limits

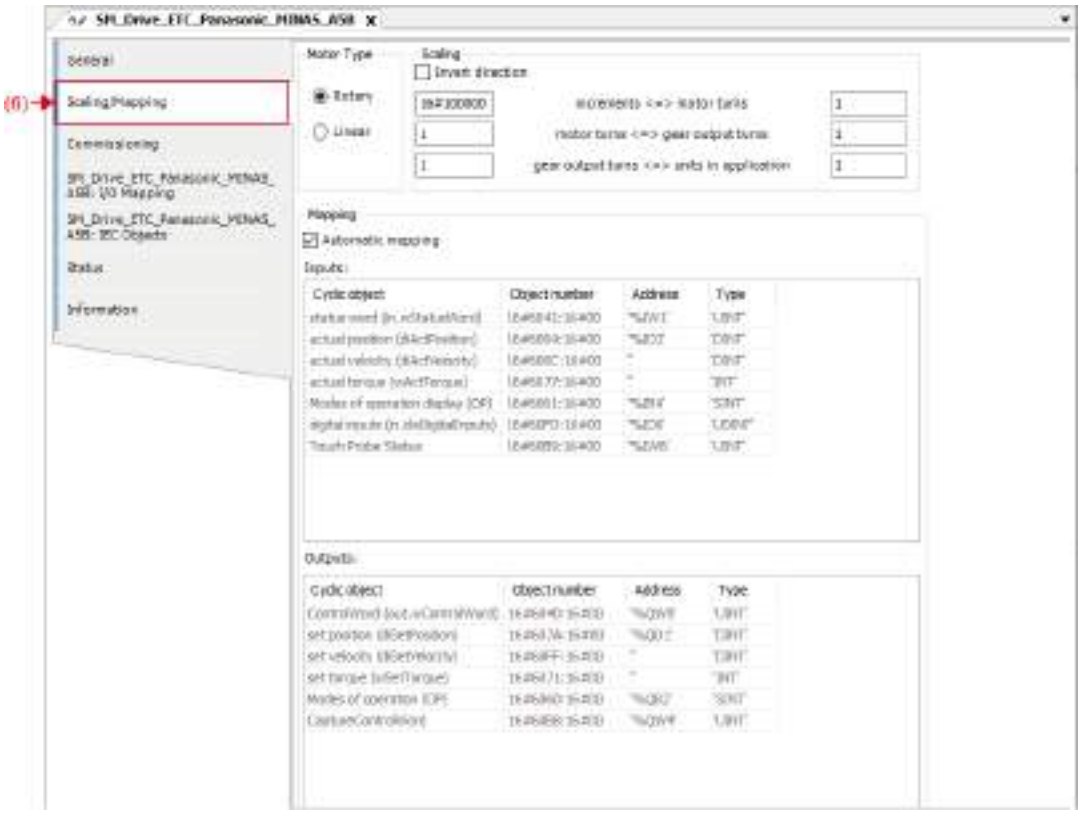
"Velocity", "Acceleration", and "Deceleration" cannot be set to 0. If they are set to 0, a warning will be issued.



The "SMC_CheckLimits" function block can be used to check whether the values set in "Dynamic limits" were exceeded during axis operations. Note that the "SMC_CheckLimits" function block cannot detect whether the value set in "Jerk” was exceeded. Therefore, do not use the "Jerk" field.

Scaling / Mapping Settings

Select the "Scaling / Mapping" tab and set the following items.



(6) Scaling / Mapping

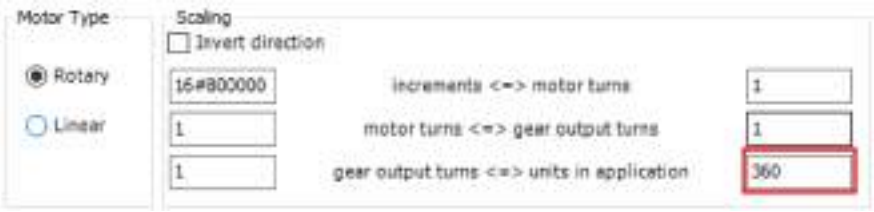
- Rotary type

  When the axis type is set to "Modulo", the ratio in conversion from drive increments to the application unit is set.

  The unit on the servo amplifier and the unit on the application (POU) are converted.

  Example:

  One revolution of the MINAS A6B MADHT1105BA1 is 0x800000. To treat one revolution as 360 in the application, set this ratio to 360.



  Invert direction: The direction is reversed.

- Linear type

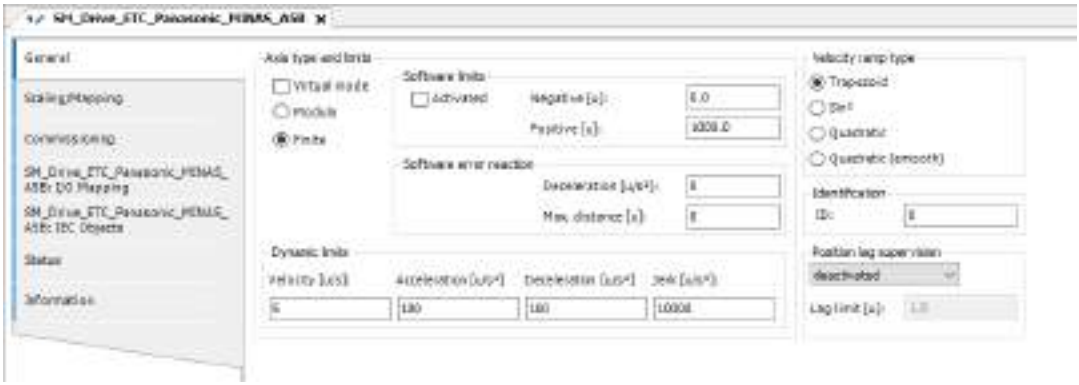
  When the axis type is set to "Finite", the ratio in conversion from drive increments to the application unit is set.

Invert direction: The direction is reversed.

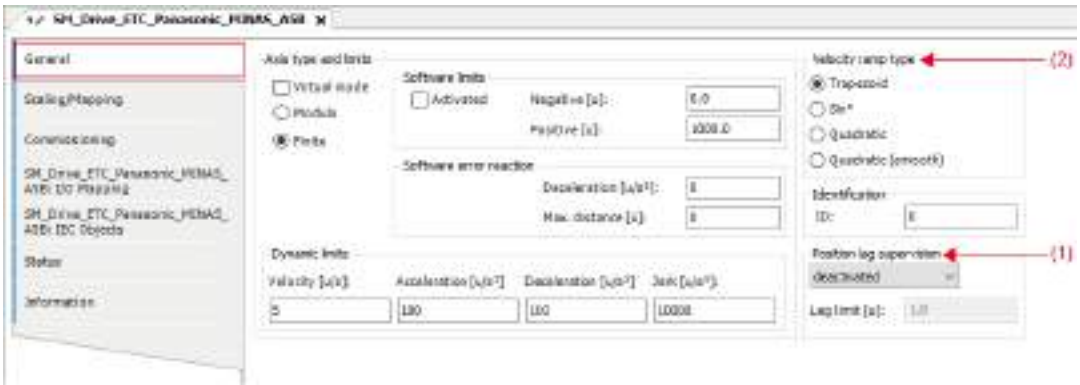
11.3.3 EtherCAT Axis Extended Setting

Configure extended settings as required.

Right-click the object in the navigator pane and then select "Edit Object" from the context-sensitive menu that is displayed. The "EtherCAT Axis Setting" dialog box will be displayed.



Select the "General" tab and set the following items.



(1) Position lag supervision

The GM1 controller does not support position lag supervision.

(2) Velocity ramp type

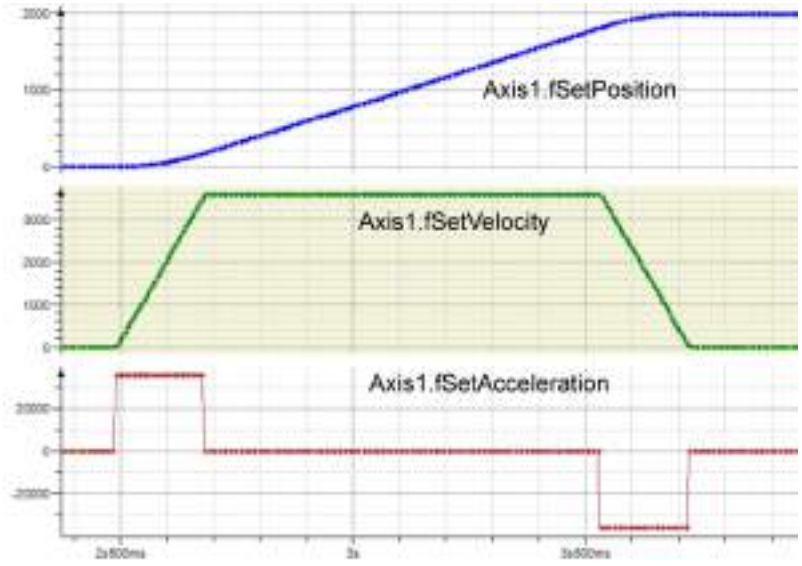
Velocity profiles are defined for each axis.

- Trapezoid

  In the trapezoidal velocity profile, velocity continues linearly.

  Therefore, acceleration can rise sharply.

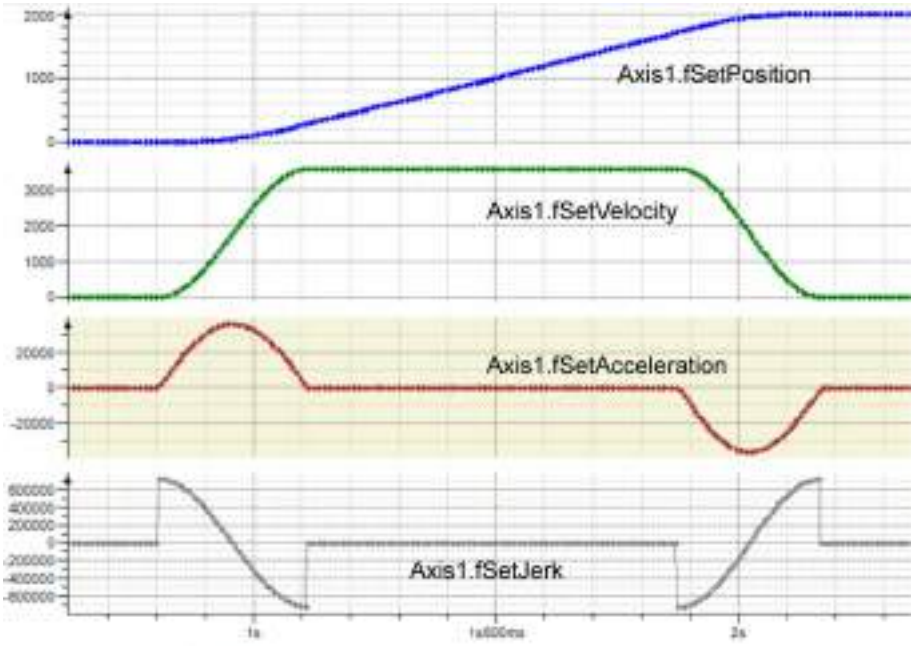
  In this mode, jerk restriction does not work in each function block.



- Sin^2

  In the velocity profile defined with the Sin^2 function, transition motion within each section of the velocity profile is smooth and acceleration rises less sharply.

  In this mode, jerk restriction basically does not work in each function block. However, jerk restriction works only when acceleration is not zero at the start of axis movement and suspended deceleration and acceleration ramps cannot be continued seamlessly.

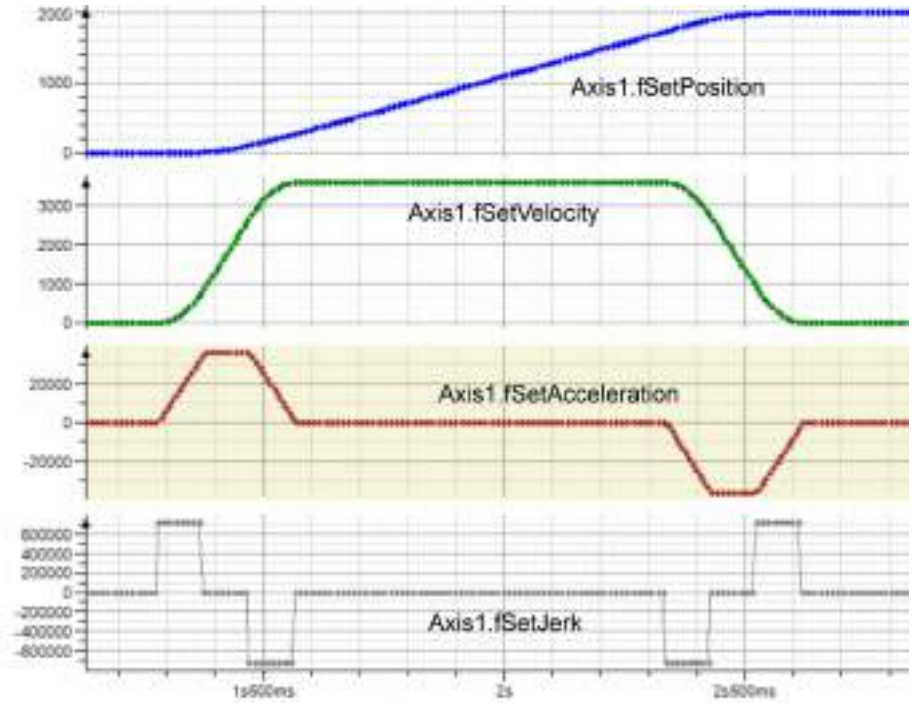


- Quadratic

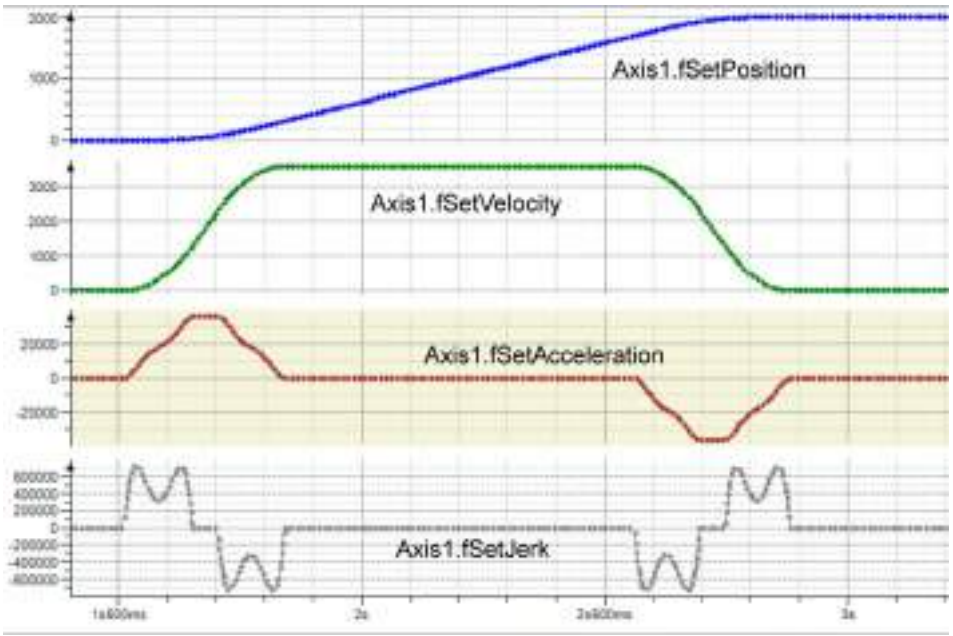
  This is a trapezoidal acceleration profile with jerk restriction.

  Acceleration changes partially linearly and continuously and jerk rises sharply.

  The maximum value of jerk can be limited by jerk values in each function block.

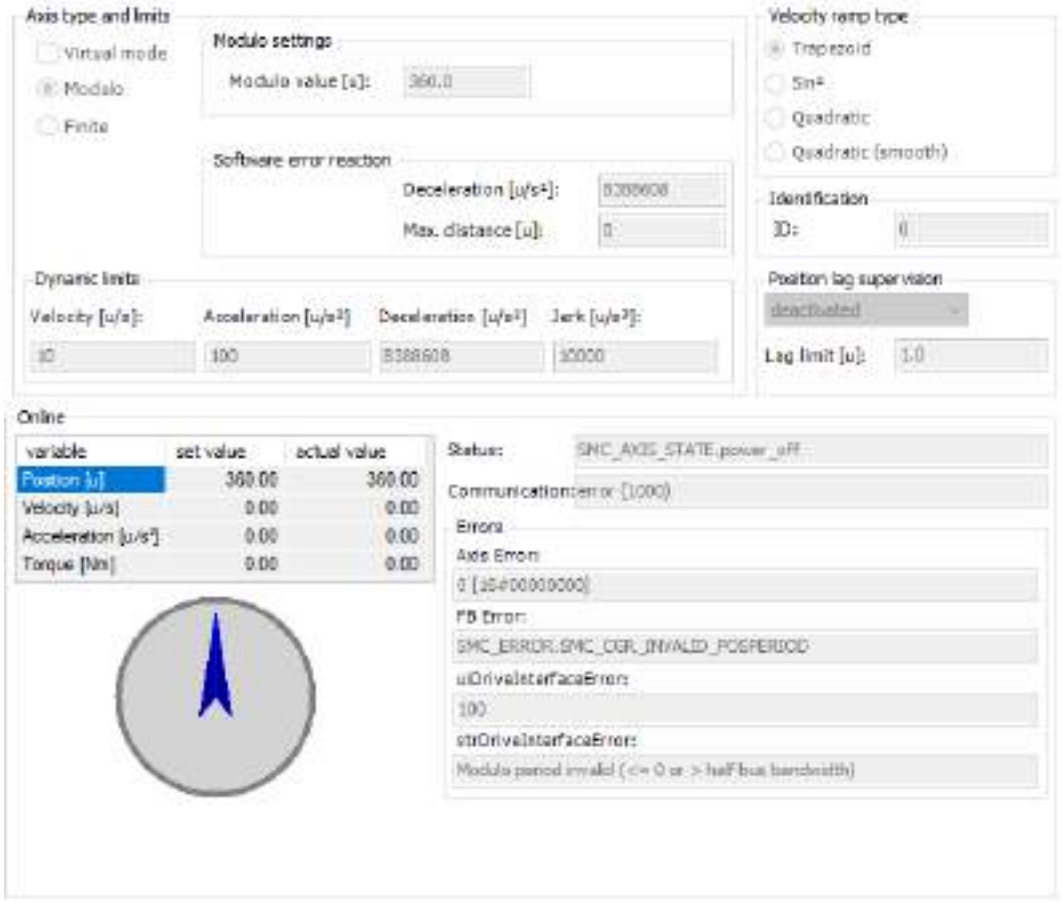


- Quadratic (Smooth)
  This is similar to "Quadratic", except that jerk does not rise sharply.
  The maximum value of jerk can be limited by jerk values in each function block.

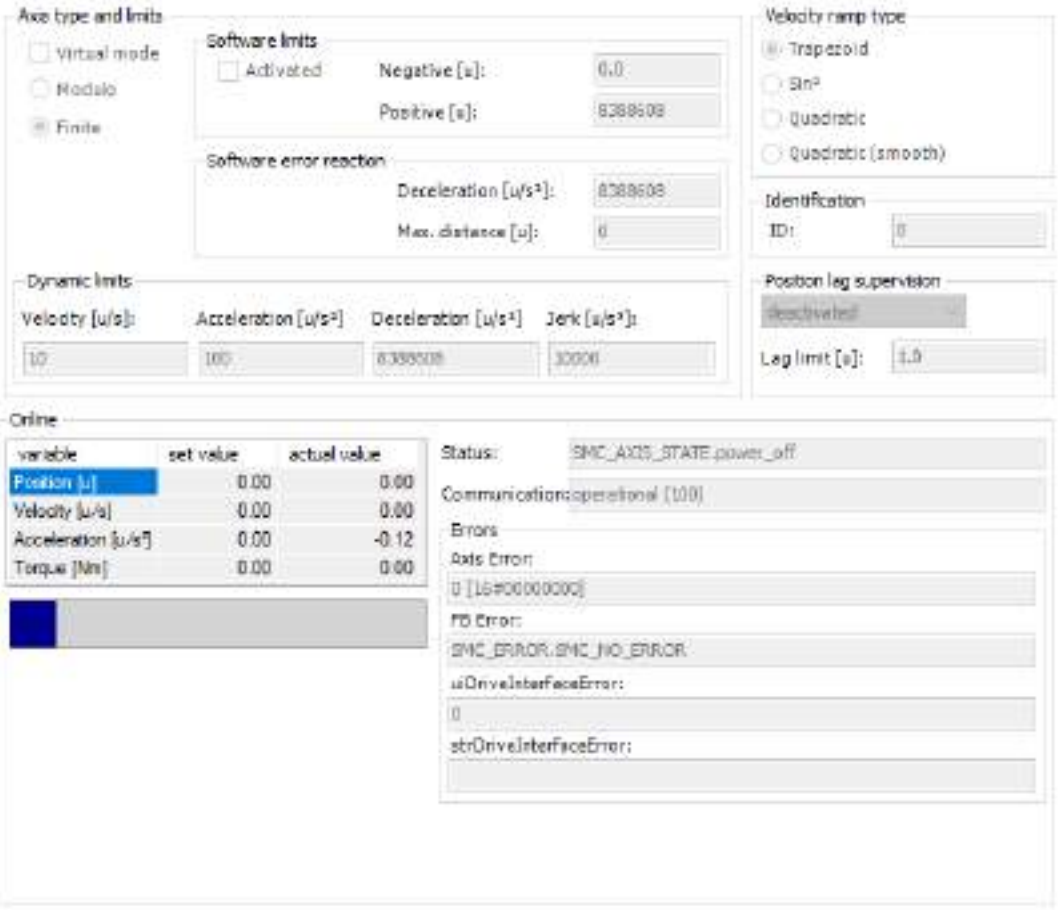


(3) Axis state window in online mode

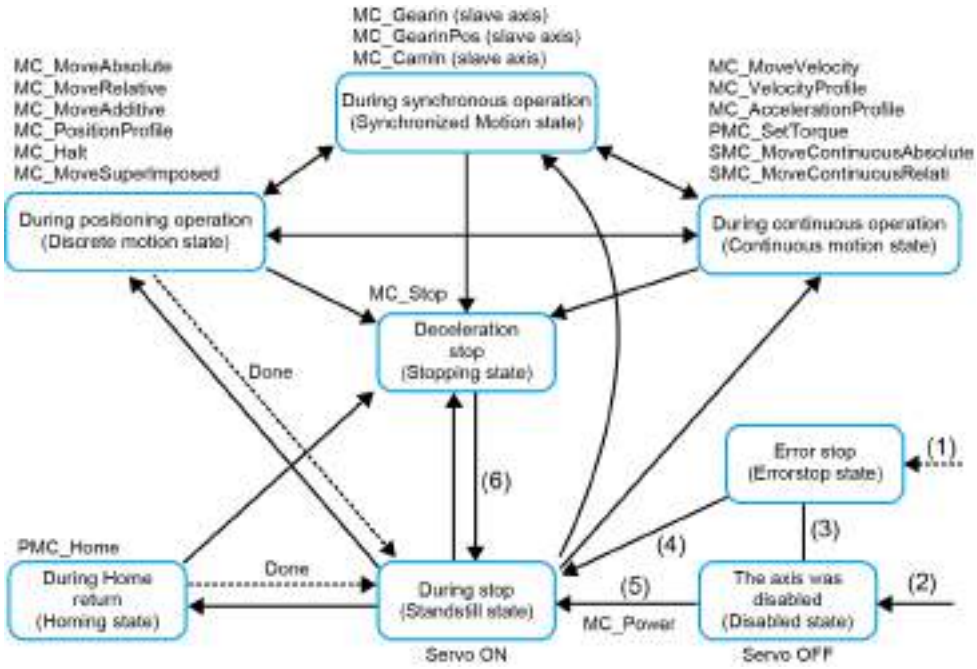
When "Modulo" is selected



When "Finite" is selected



(4) "Online" - "State"
This indicates one of the axis states in the following figure.



(5) "Online" - "Communication"

This indicates one of the following communication states.

Stopped
Variable initialized
Basic communication initialized
Drive initialized
Drive synchronization on standby
Initialization done
Operational
Re-initialized
Error
Unknown

(6) "Online" - "Error"

- FB error

  The oldest error that occurred on the axis, "SMC_ERROR", is displayed.

  This is the same error as the one that can be obtained by the "SMC_ReadFBError" function block.

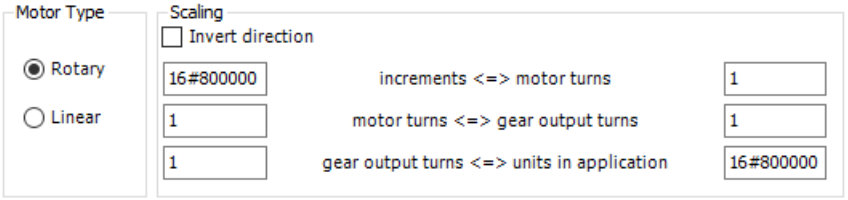
- uiDriveInterfaceError / strDriveInterfaceError

  This is an internal error in the GM1 controller.

(7) Scaling / Mapping

- Rotary type

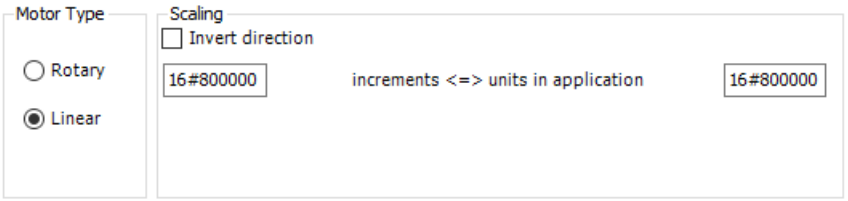
  When the axis type is set to "Modulo", the ratio in the conversion from the drive increment to the application unit is set.



Note: Invert direction: The direction is inverted.

- Linear type

  When the axis type is set to "Finite", the ratio in the conversion from the drive increment to the application unit is set.



Note: Invert direction: The direction is inverted.

11.4 Basic Preparations for Operation

11.4.1 Overview of Basic Preparations for Operation

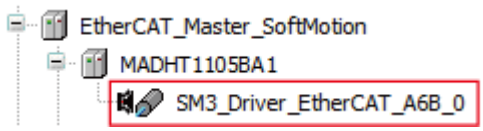
This section explains how to run and stop the motor.

11.4.2 Servo ON or OFF

To turn ON or OFF the servo motor, use the "MC_Power" function block.

■ Explanation of function block

- For input "Axis", specify the axis corresponding to the servo motor.
  Example) In the case of SM3_Driver_EtherCAT_A6B_0, substitute "SM3_Driver_EtherCAT_A6B_0" for “Axis” of MC_Power.



- When inputs "Enable", "bRegulatorOn", and "bDriveStart" are set to TRUE, the servo turns ON.
  When input "bRegulatorOn" is set to FALSE, the servo turns OFF.

■ Program examples

The following are LD program and ST program examples that execute the "MC_Power" function block.
Setting variable "SW_Power" to TRUE turns ON the servo and setting it to FALSE turns OFF the servo.

LD program

ST program

```
PROGRAM ST_POU
VAR
    MC_Power_0: MC_Power;
    SW_Power: BOOL := FALSE;
    PStatus: BOOL := FALSE;
END_VAR
```

```
MC_Power_0(
    Axis:= SM_Drive_ETC_Panasonic_MINAS_A5B,
    Enable:= TRUE,
    bRegulatorOn:= SW_Power,
    bDriveStart:= TRUE,
    Status=> PStatus,
    bRegulatorRealState=> ,
    bDriveStartRealState=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> );
```

Info.

- When executing "MC_Power", confirm in advance that communication has been established. The communication state can be checked using the "SMC_CheckAxisCommunication" function block.

11.4.3 Home Return

Home return is an operation that returns the motor to its home position.

The GM1 controller supports various home return methods.

Home return can be achieved by executing the "MC_Home" function block.

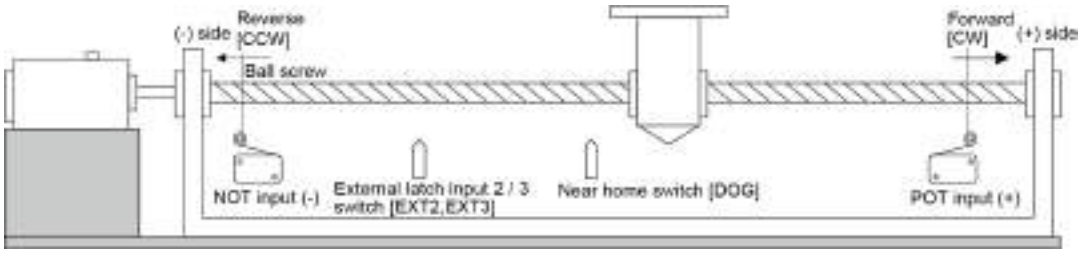
Types of home return

Home return is a function that moves the axis to the preset reference position (home position) and set the coordinates of the position to 0.

If an incremental encoder is used for the servomotor, the home return methods shown in the table below can be selected.

Info.

- The GM1 controller also supports home return using an absolute encoder. When implementing home return using an absolute encoder, use MINAS V1.24 or later.



Settings and operations of home return	Behavior overview
Edge detection of near home switch + Home position (Z phase) based on front edge	After the rising edge (front edge) of the near home switch (DOG) is detected, the rising edge of the first home position (Z phase) is detected and the motor stops. The stopping position is set as the home position.
Edge detection of near home switch	The rising edge of the near home switch (DOG) is detected and the motor stops. The stopping position is set as the home position.
Edge detection of near home switch + Home position (Z phase) based on rear edge	After the falling edge (rear edge) of the near home switch (DOG) is detected, the rising edge of the first home position (Z phase) in the home return direction is detected and the motor stops. The stopping position is set as the home position.
Edge detection of limit switch + Home position (Z phase) based on front edge	After the rising edge of the limit switch on the opposite side of the home return direction is detected, the rotation of the motor is reversed. Then, the rising edge of the first home position (Z phase) is detected and the motor stops. The stopping position is set as the home position.
Edge detection of limit switch	The rising edge of the limit switch in the home return direction is detected and the motor stops. The stopping position is set as the home position.
Edge detection of home position (Z phase)	The axis moves from the current value toward the direction of home return. Then, the rising edge of the first home position (Z phase) is detected and the motor stops. The stopping position is set as the home position.
Data set method	The current value is set as the home position.

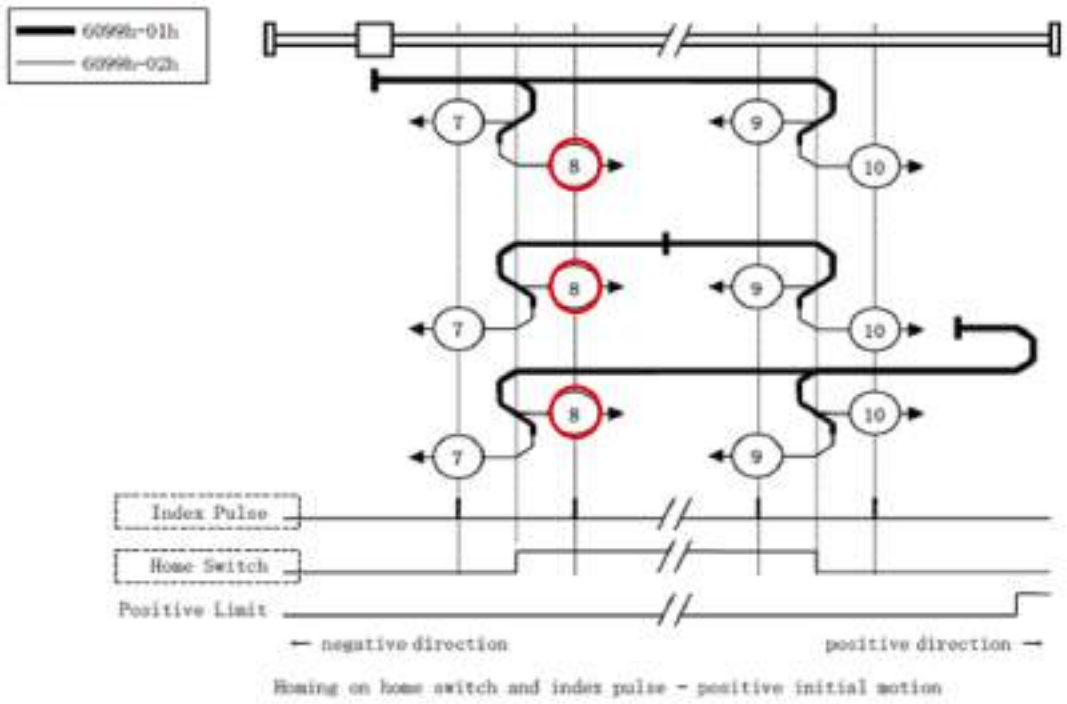
Edge detection of near home switch + Home position (Z phase) based on front edge

After the rising edge (front edge) of the near home switch (DOG) is detected, the rising edge of the first home position (Z phase) is detected and the motor stops. The stopping position is set as the home position.

(Note 1) If the home position (Z phase) is ON at the time of startup, it will not be regarded as a home position (Z phase). Searches for a near home switch (DOG) will be started.

■ Method 8 (Home return direction: positive direction)

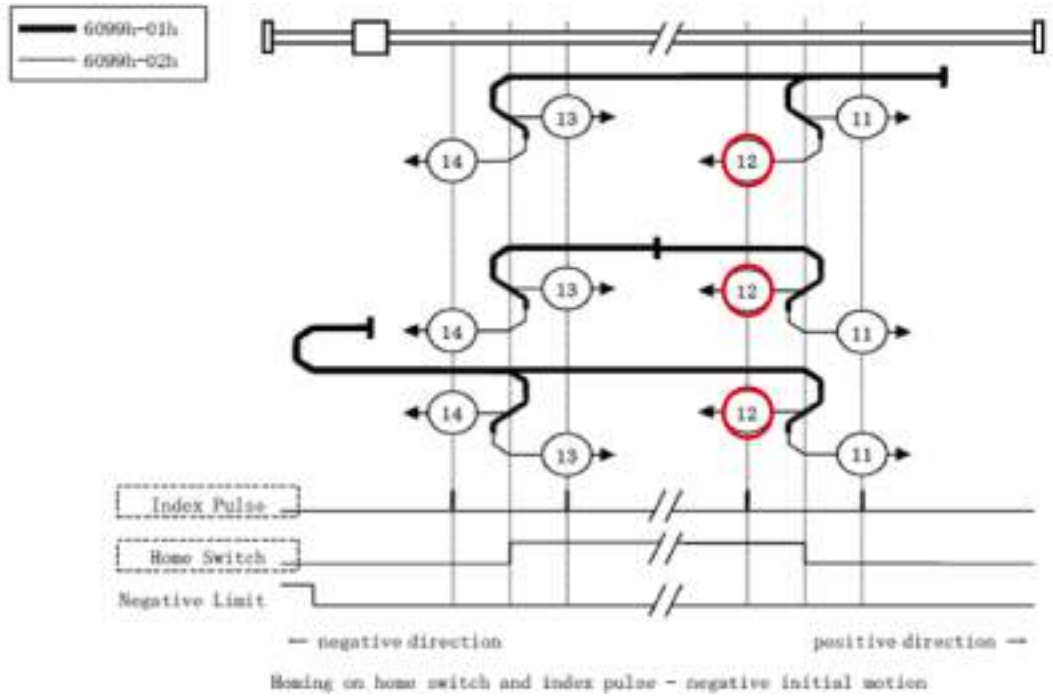
- This method uses the home switch (HOME) and the index pulse (Z phase).
- The initial motion direction is a negative direction when the home switch (HOME) is active at the start of operation.
- The home detection position is the index pulse (Z phase) near the rising edge or the falling edge of the home switch (HOME). (Refer to the following figure.)
- If HOME and NOT are not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



Homing on home switch and index pulse - positive initial motion

■ Method 12 (Home return direction: negative direction)

- This method uses the home switch (HOME) and the index pulse (Z phase).
- The initial motion direction is a positive direction when the home switch (HOME) is active at the start of operation.
- The home detection position is the index pulse (Z phase) near the rising edge or the falling edge of the home switch (HOME). (Refer to the following figure.)
- If HOME and NOT are not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.

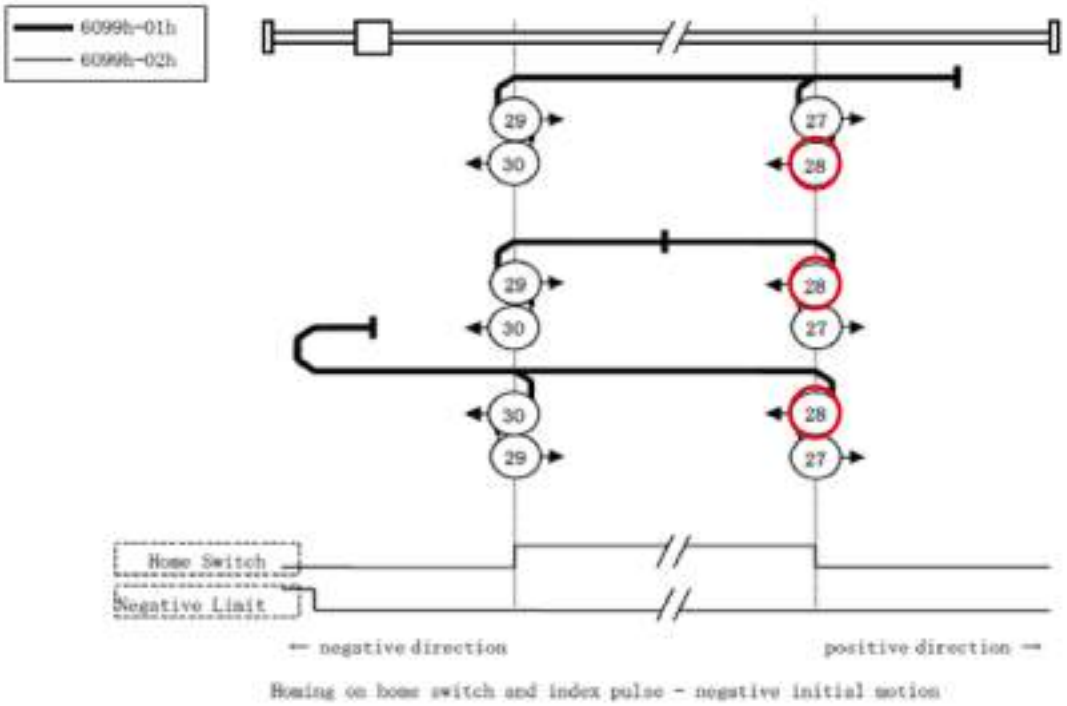


Edge detection of near home switch

The rising edge of the near home switch (DOG) is detected and the motor stops. The stopping position is set as the home position.

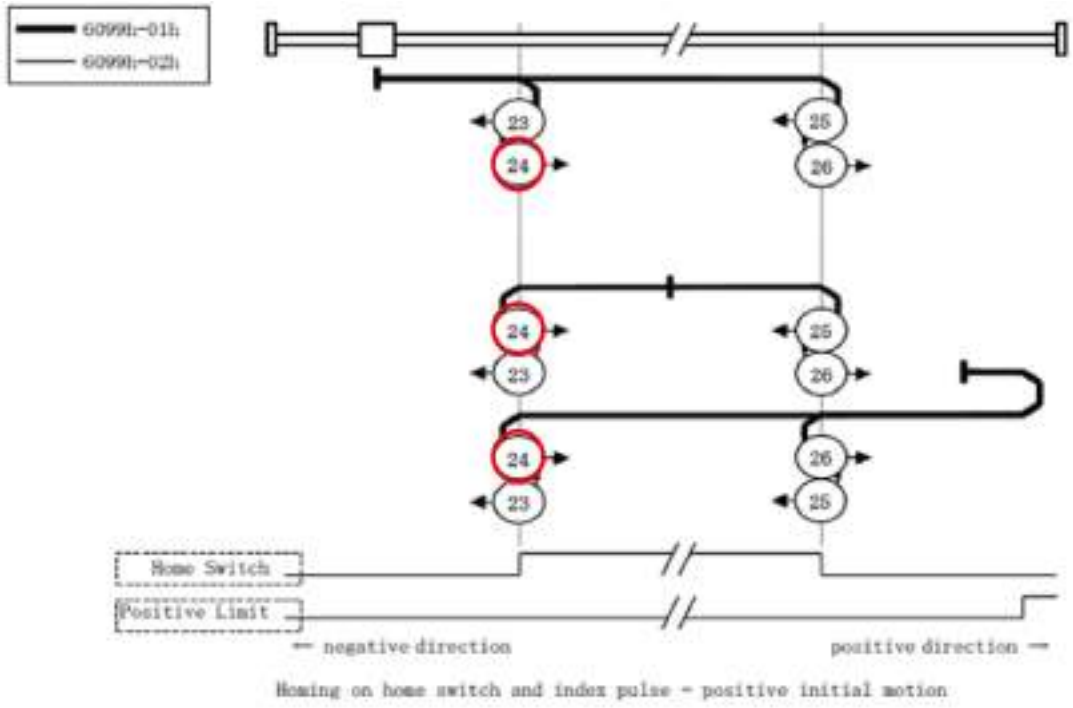
■ Method 28 (Home return direction: negative direction)

- This method uses the home switch (HOME).
- The initial motion direction is a positive direction when the home switch (HOME) is active at the start of operation.
- The home detection position is the position where the home switch (HOME) changes its state. (Refer to the following figure.)
- If HOME and NOT are not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



■ Method 24 (Home return direction: positive direction)

- This method uses the home switch (HOME).
- The initial motion direction is a negative direction when the home switch (HOME) is active at the start of operation.
- The home detection position is the position where the home switch (HOME) changes its state. (Refer to the following figure.)
- If HOME and NOT are not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



Homing on home switch and index pulse - positive initial motion

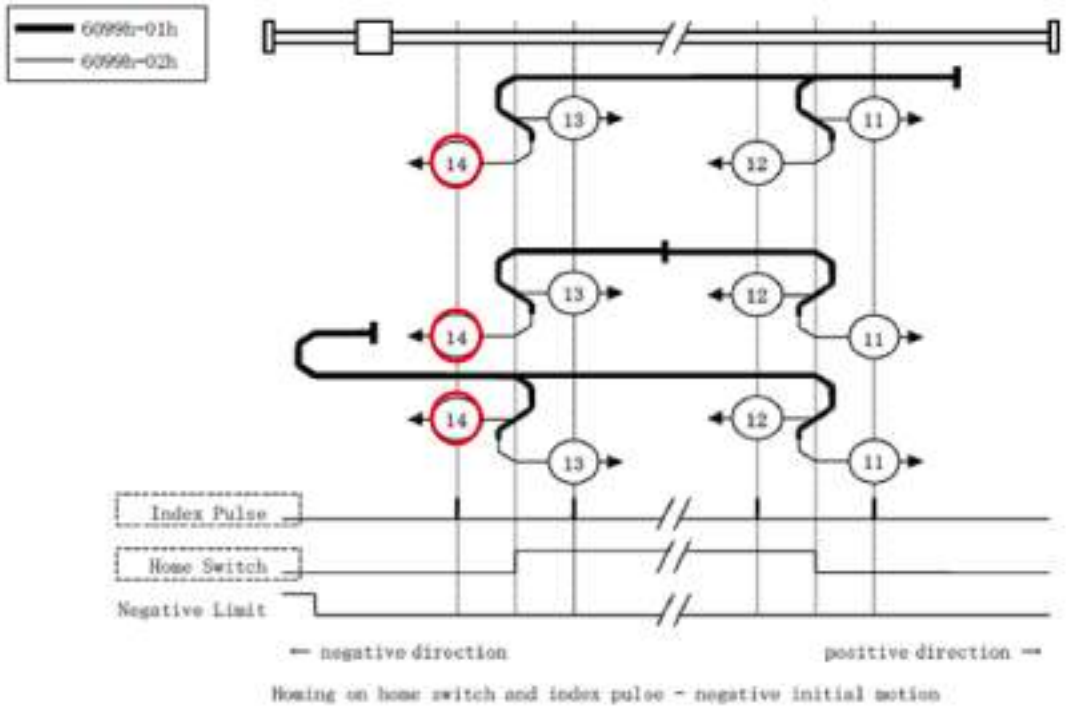
Edge detection of near home switch + Home position (Z phase) based on rear edge

After the falling edge (rear edge) of the near home switch (DOG) is detected, the rising edge of the first home position (Z phase) in the home return direction is detected and the motor stops. The stopping position is set as the home position.

(Note 1) If the home position (Z phase) is ON at the time of startup, it will not be regarded as a home position (Z phase). Searches for a near home switch (DOG) will be started.

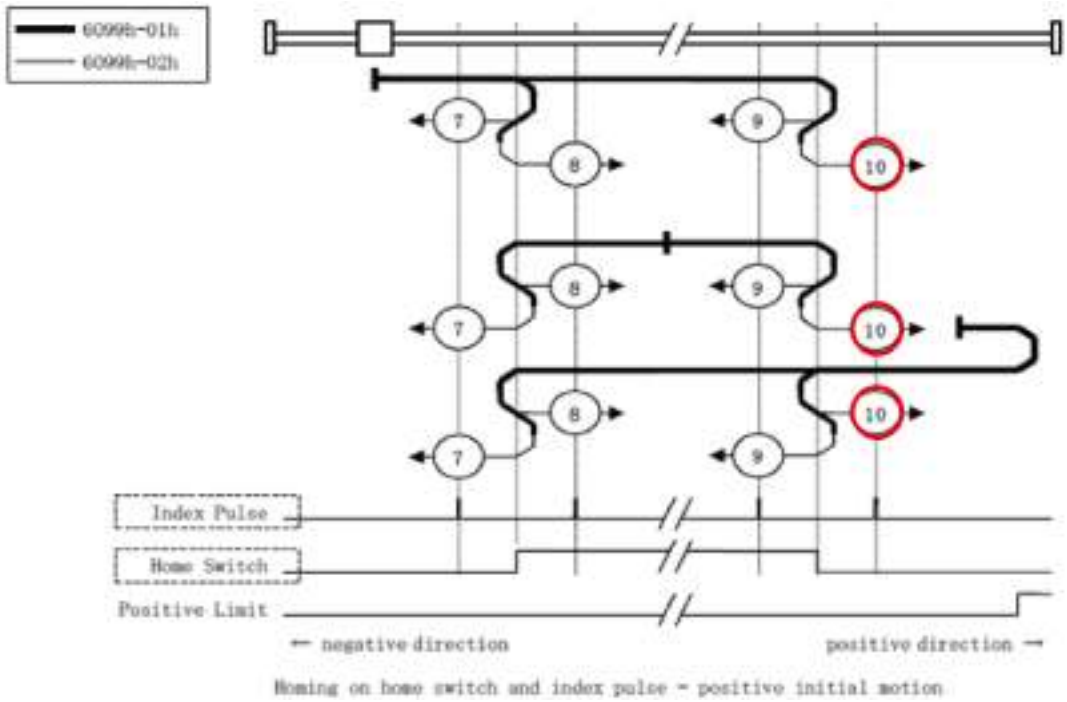
■ Method 14 (Home return direction: negative direction)

- This method uses the home switch (HOME) and the index pulse (Z phase).
- The initial motion direction is a positive direction when the home switch (HOME) is active at the start of operation.
- The home detection position is the index pulse (Z phase) near the rising edge or the falling edge of the home switch (HOME). (Refer to the following figure.)
- If HOME and NOT are not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



■ Method 10 (Home return direction: positive direction)

- This method uses the home switch (HOME) and the index pulse (Z phase).
- The initial motion direction is a negative direction when the home switch (HOME) is active at the start of operation.
- The home detection position is the index pulse (Z phase) near the rising edge or the falling edge of the home switch (HOME). (Refer to the following figure.)
- If HOME and NOT are not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



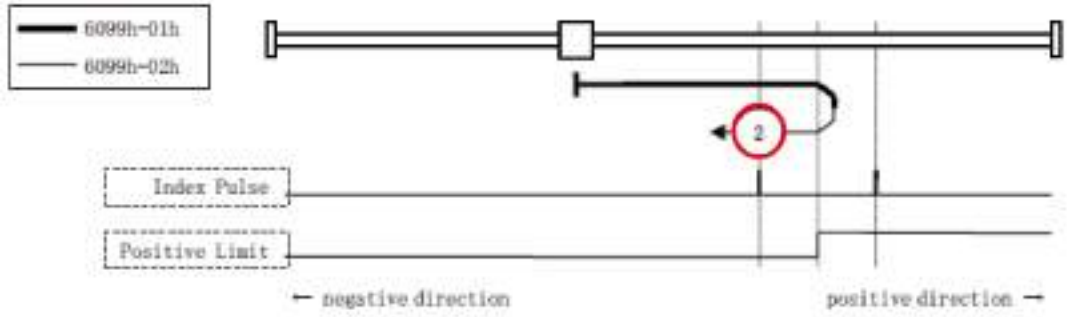
Edge detection of limit switch + Home position (Z phase) based on front edge

After the rising edge of the limit switch on the opposite side of the home return direction is detected, the rotation of the motor is reversed. Then, the rising edge of the first home position (Z phase) is detected and the motor stops. The stopping position is set as the home position.

(Note 1) If the home position (Z phase) is ON at the time of startup, it will not be regarded as a home position (Z phase). Searches for a limit switch will be started.

■ Method 2 (Home return direction: negative direction)

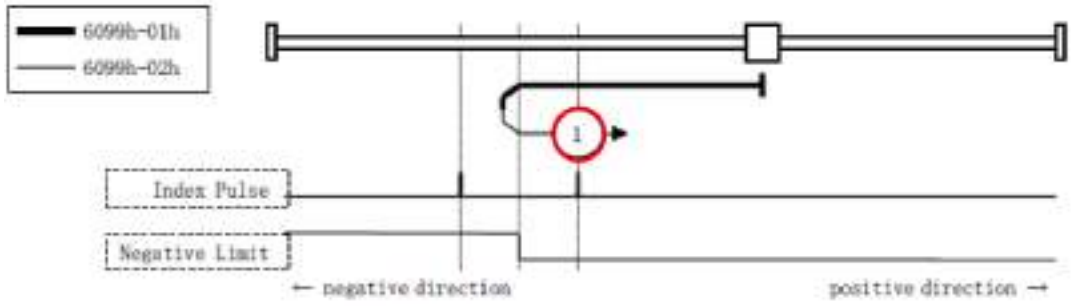
- In this method, the initial motion direction is a positive direction when the positive limit switch (POT) is inactive. (The inactive state is indicated as a low level state in the figure.)
- The home detection position is the first index pulse (Z phase) detection position on the negative side after the positive limit signal (POT) turns inactive. (Refer to the following figure.)
- If POT is not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



Homing on positive limit switch and index pulse

■ Method 1 (Home return direction: positive direction)

- In this method, the initial motion direction is a negative direction when the negative limit switch (NOT) is inactive. (The inactive state is indicated as a low level state in the figure.)
- The home detection position is the first index pulse (Z phase) detection position on the positive side after the negative limit signal (NOT) turns inactive.
- If NOT is not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



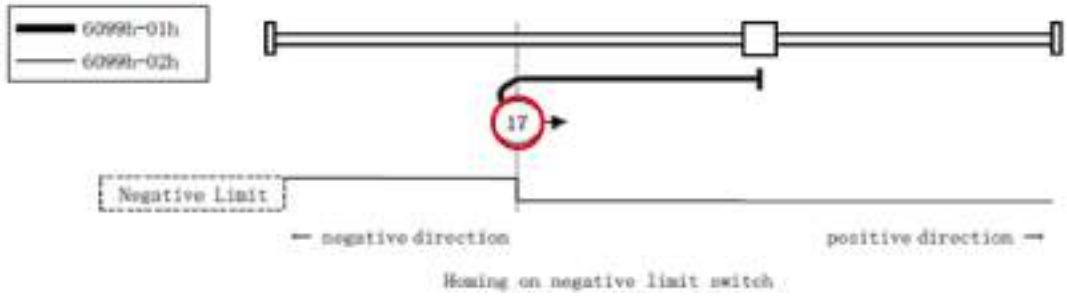
Homing on negative limit switch and index pulse

Edge detection of limit switch

The rising edge of the limit switch in the home return direction is detected and the motor stops. The stopping position is set as the home position.

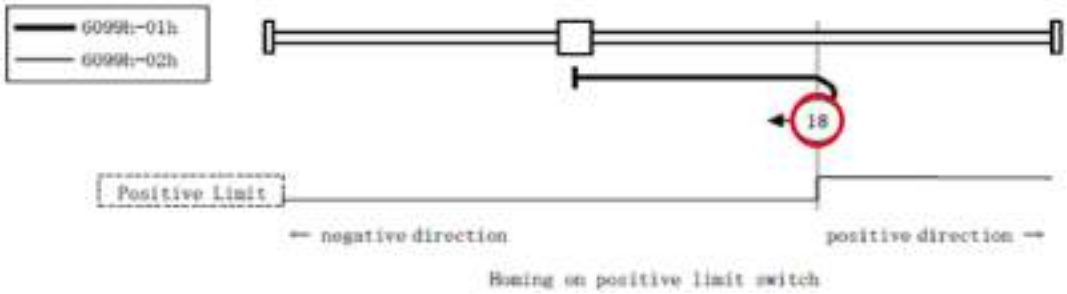
■ Method 17 (Home return direction: positive direction)

- In this method, the initial motion direction is a negative direction when the negative limit switch (NOT) is inactive. (The inactive state is indicated as a low level state in the figure.)
- The home detection position is the position where the negative limit switch (NOT) changes its state.
- If NOT is not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.



■ **Method 18 (Home return direction: negative direction)**

- In this method, the initial motion direction is a positive direction when the positive limit switch (POT) is inactive. (The inactive state is indicated as a low level state in the figure.)
- The home detection position is the position where the positive limit switch (POT) changes its state.
- If NOT is not assigned, an error occurs in the MC_Home function block. For details on signal assignment, refer to relevant manuals for the MINAS series.

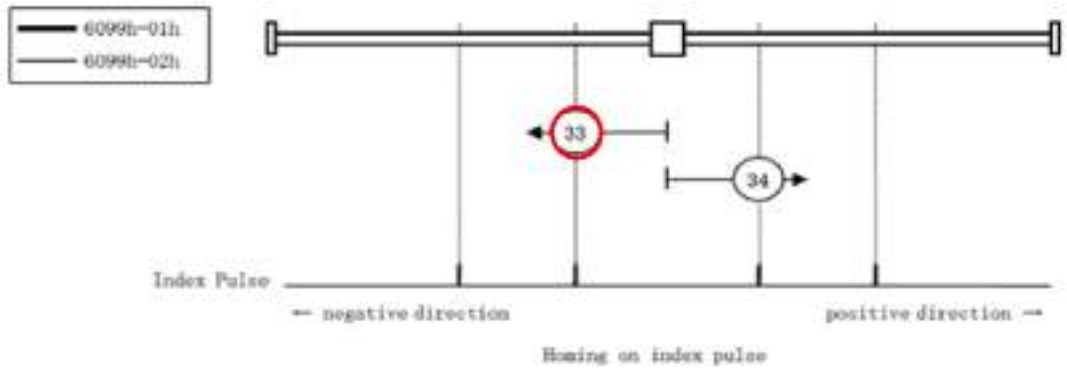


Edge detection of home position (Z phase)

The axis moves from the current value toward the direction of home return. Then, the rising edge of the first home position (Z phase) is detected and the motor stops. The stopping position is set as the home position.

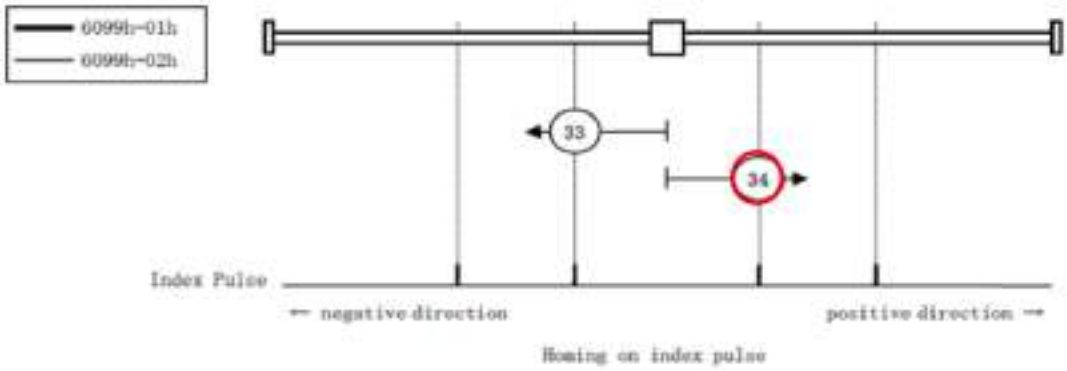
■ **Method 33 (Home return direction: negative direction)**

- This method uses only the index pulse (Z phase).
- The index pulse (Z phase) that is detected after operating in the direction shown in the figure is regarded as the home detection position.
- For detailed specifications, refer to relevant manuals for the MINAS series.



■ **Method 34 (Home return direction: positive direction)**

- This method uses only the index pulse (Z phase).
- The index pulse (Z phase) that is detected after operating in the direction shown in the figure is regarded as the home detection position.
- For detailed specifications, refer to relevant manuals for the MINAS series.



Data set method

The current position is set as the home position.

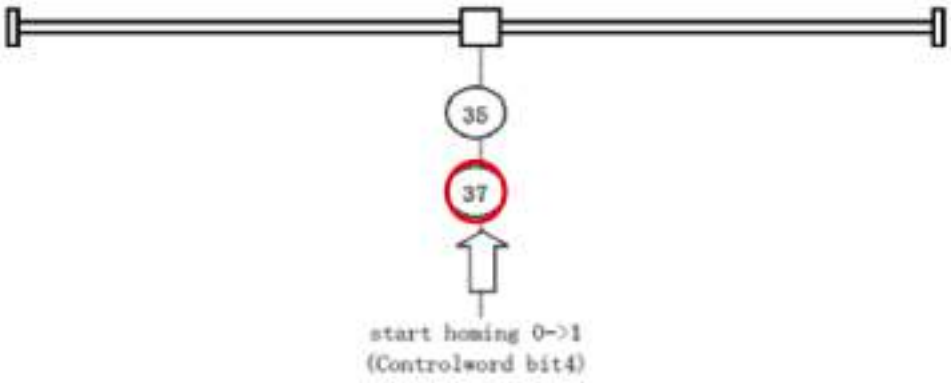
■ **Method 37**

- This method is used when setting the coordinate system (position information setting) of the servo amplifier.

  The following objects are initialized (preset) based on the position where MC_Home is executed.

  6062h (Position demand value) = 6064h (Position actual value) = 607Ch (Home offset)

  6063h (Position actual internal value) = 60FCh (Position demand internal value) = 0
- For detailed specifications, refer to relevant manuals for the MINAS series.



Settings and operations of home return

■ Settings in GM Programmer

To use the home return function, set servo amplifiers of each manufacturer using the ETC_CO_SdoWrite function block and execute "MC_Home".

Specify the following settings according to the home return function to be used. The following is an example of setting the MINAS A6B.

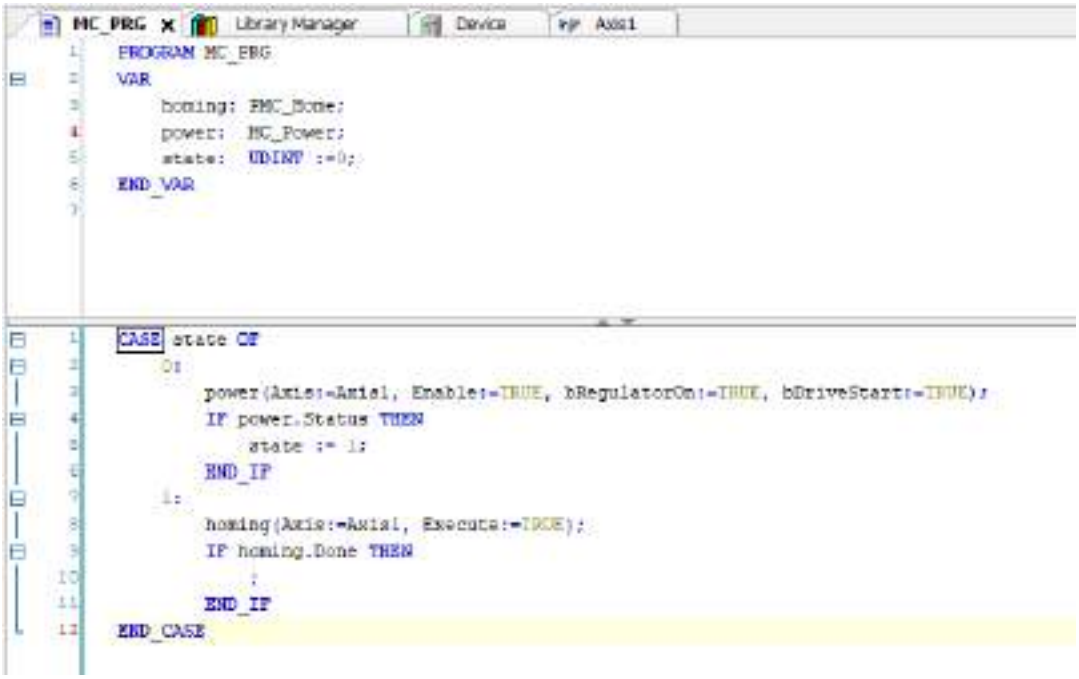
Index	Sub-Index	Name	Description
6098h	00h	Homing method	Home return method
6099h	01h	Speed during search for switch	Home return speed
6099h	02h	Speed during search for zero	Home return creep speed
609Ah	03h	Homing acceleration	Home return acceleration / deceleration

Info.

- For details, refer to relevant manuals for the MINAS series.

Home return is performed by executing "MC_Home" in POU.

Program example



11.4.4 JOG Operation

Executing a JOG operation continues to run the motor at the specified speed.
JOG operation can be executed with the "MC_Jog" function block.

■ Explanation of functions

- While input "JogForward" is set to TRUE, the motor continues to run in the forward direction. While input "JogBackward" is set to TRUE, the motor continues to run in the reverse direction. If both "JogForward" and "JogBackward" are set to TRUE, JOG operation will be stopped.
- You can specify velocity ("Velocity", unit: u/s), acceleration ("Acceleration", unit: u/s^2), deceleration ("Deceleration", unit: u/s^2), and jerk ("Jerk", unit: u/s^3) that are used during JOG operation.

■ **Program examples**

Example: ST program that executes JOG operation after the servo is turned ON

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Execute the MC_Jog with JogForward
        MC_Jog_0(
            Axis:=Axis1 ,
            JogForward:=TRUE ,
            JogBackward:=FALSE ,
            Velocity:=360 ,
            Acceleration:=3600 ,
            Deceleration:=3600
        );
        MC_ReadActualPosition_0(
                            Axis:=Axis1 ,
                            Enable:=TRUE ,
        );
        IF MC_ReadActualPosition_0.Valid = TRUE THEN
            IF MC_ReadActualPosition_0.Position > 1000 THEN
                Process := 2;
            END_IF
        END_IF
    2://Execute the MC_Jog with BackForward
        MC_Jog_0(
            Axis:=Axis1 ,
            JogForward:=FALSE ,
            JogBackward:=TRUE ,
            Velocity:=720 ,
            Acceleration:=3600 ,
            Deceleration:=3600
        );
END_CASE
```

JOG operation is started at the rising edge of "JogForward" or "JogBackward".

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, MC_Jog_0 must be called every cycle. Otherwise, the operation will terminate with an error.

"JogForward" and "JogBackward" must not be turned ON at the same time. Otherwise, JOG operation will terminate. As shown in this program example, if you want to switch the direction from "JogForward" to "JogBackward", set "JogForward" to FALSE and then set "JogBackward" to TRUE.

11.5 Single-axis Operation

This section explains single-axis operations using function blocks.

11.5.1 Overview of Single-axis Operation

The motion function of the GM1 controller supports position control, velocity control, and torque control as control modes.

While switching the control mode according to the purpose, you can use motion-related function blocks.

This section explains how to switch the control mode and how to do programming for typical control methods in each control mode.

For the detailed specifications of function blocks supported by the GM1 controller, refer to the instruction reference.

11.5.2 Position Control

Position control is control that runs the motor until the specified position or distance is reached.

The "MC_MoveAbsolute" function block is used to specify a movement destination position ("Position") and the "MC_MoveRelative" function block is used to specify a movement distance ("Distance").

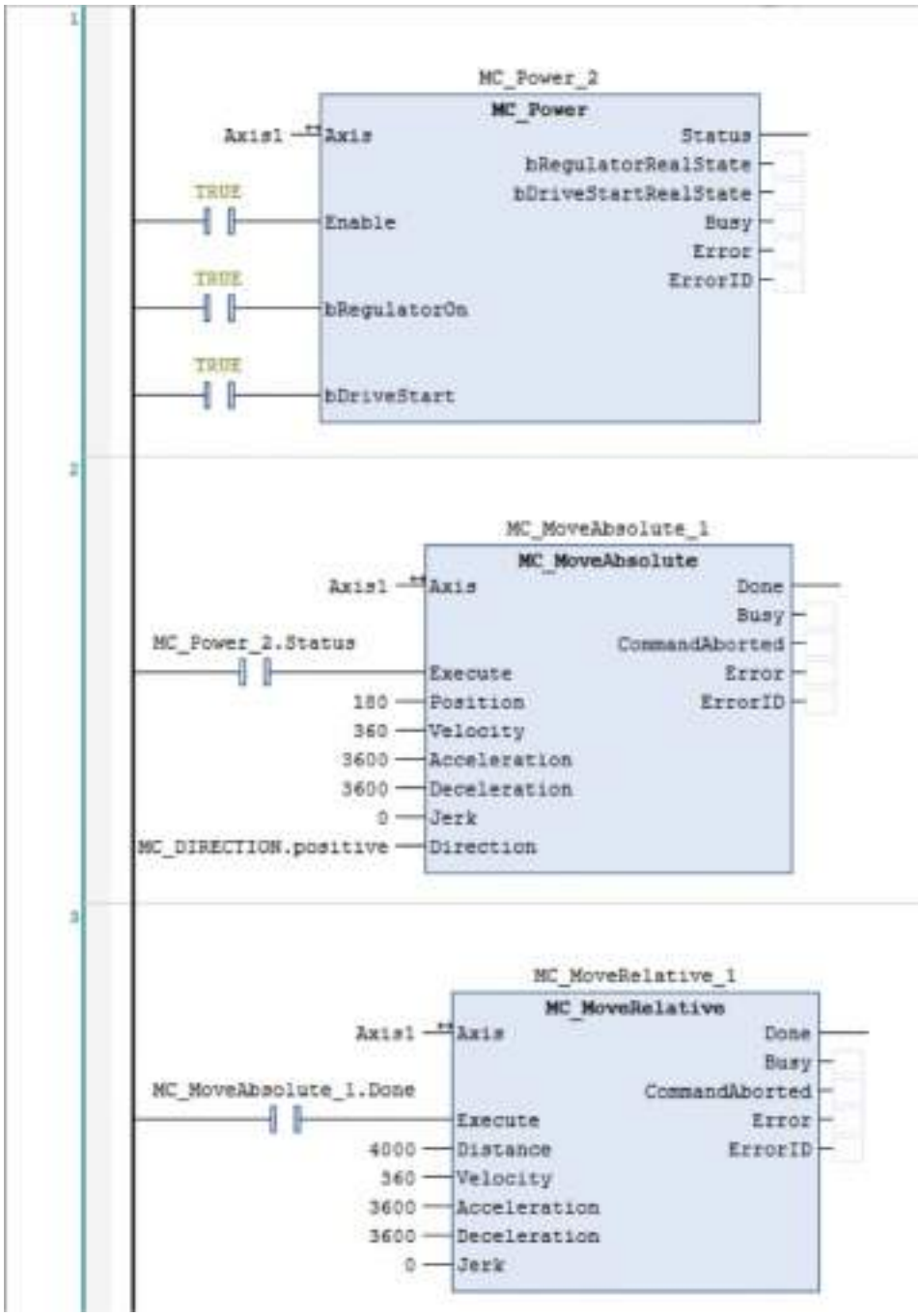
■ Program examples

In the following LD program example, the "MC_MoveAbsolute" function block is used to run the motor in the forward direction until position 180 is reached and the "MC_MoveRelative" function block is used to run the motor until distance 4000 is reached. Before executing the "MC_Absolute" function block, use the "MC_Power" function block to turn ON the servo corresponding to the target axis.

Info.

- For details, refer to the *GM1 Series Reference Manual (Instruction)*.

LD program



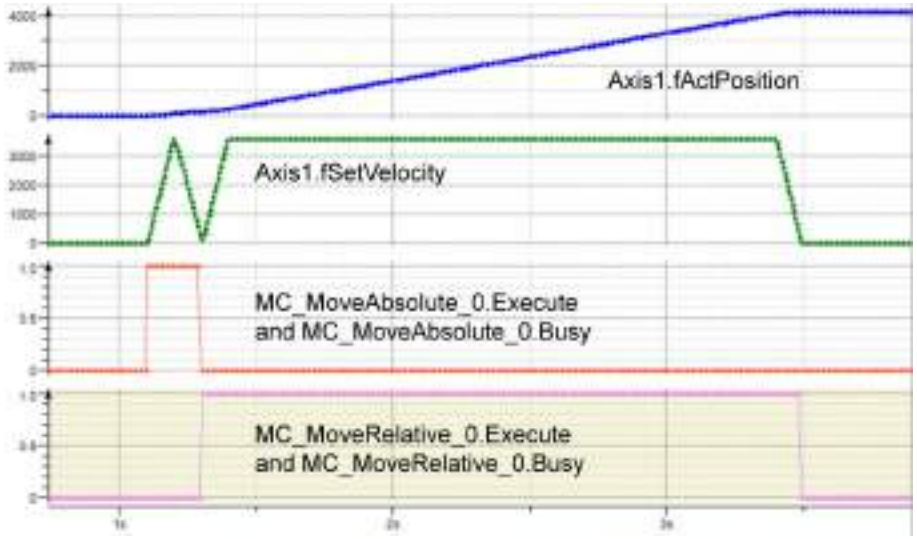
ST program

The following ST program example executes the same behavior as the LD program on the previous page.

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Execute the MC_MoveAbsolute
        MC_MoveAbsolute_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Position:=180 ,
                    Velocity:=3600 ,
                    Acceleration:=72000 ,
                    Deceleration:=72000 ,
                    Direction:=positive
        );
        IF MC_MoveAbsolute_0.Done = TRUE THEN
            MC_MoveAbsolute_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 2;
        END_IF
    2://Execute the MC_MoveRelative
        MC_MoveRelative_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Distance:=4000 ,
                    Velocity:=3600 ,
                    Acceleration:=72000 ,
                    Deceleration:=72000
        );
        IF MC_MoveRelative_0.Done = TRUE THEN
            MC_MoveRelative_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 3;
        END_IF
    3://End
        //No Operation
END_CASE
```

"MC_MoveAbsolute" and "MC_MoveRelative" are started at the rising edge of the "Execute" flag. If processing is completed normally, the "Done" flag will be set to TRUE.

In this program example, the actual position eventually becomes 4180.



The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, an instance of the function block must be called every cycle. Otherwise, the operation will terminate with an error.

Motion function blocks for the GM1 controller allow the user to change parameters during operation.

To change parameter settings such as target positions ("Position") during operation, temporarily set the "Execute" flag to FALSE beforehand. After parameter settings have been changed, if the "Execute" flag is set back to TRUE, the changed parameter settings will be applied.

In the following example, "Position" in "MC_MoveAbsolute" is changed from 90 to 180. While "MC_MoveAbsolute" is being executed (Busy=TRUE), parameters can be switched.

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Execute the MC_MoveAbsolute
        MC_MoveAbsolute_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Position:=90 ,
                    Velocity:=360 ,
                    Acceleration:=3600 ,
                    Deceleration:=3600 ,
                    Direction:=positive
        );
        IF MC_MoveAbsolute_0.Busy = TRUE THEN
            MC_MoveAbsolute_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 2;
        END_IF
    2://Change the position of the MC_MoveAbsolute_0
        MC_MoveAbsolute_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Position:=180
        );
        IF MC_MoveAbsolute_0.Done = TRUE THEN
            MC_MoveAbsolute_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 3;
        END_IF
    3://End
        //No Operation
END_CASE
```

Motion function blocks for the GM1 controller can be overwritten with other instances during operation.

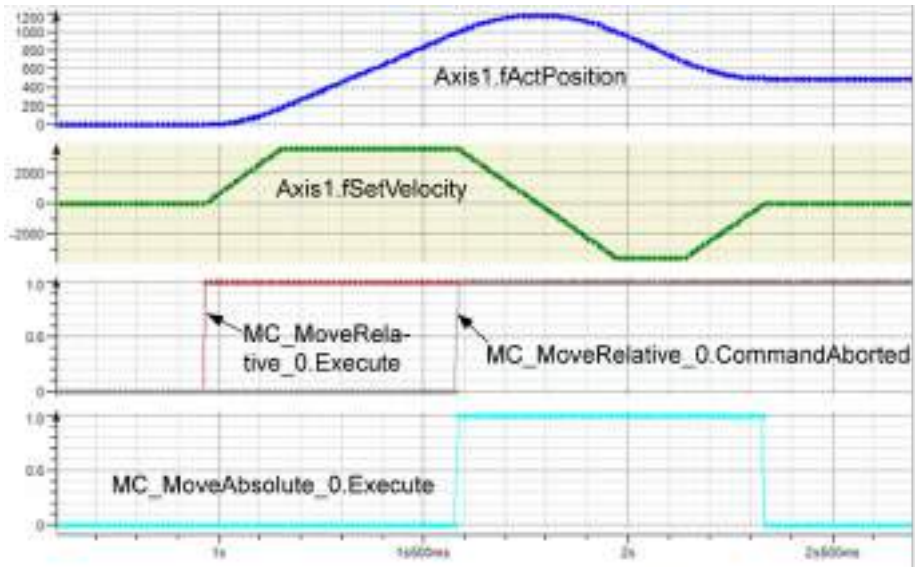
In the following example, an instance of "MC_MoveRelative" is overwritten with another instance of "MC_MoveAbsolute" during its execution.

During the execution of "MC_MoveRelative", when the actual position reaches 1000, its instance is overwritten with another instance of "MC_MoveAbsolute".

The "CommandAborted" flag of "MC_MoveRelative" is set to TRUE, causing the processing to be suspended and "MC_MoveAbsolute" to be executed.

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Execute the MC_MoveRelative
        MC_MoveRelative_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Distance:=3000 ,
                    Velocity:=3600 ,
                    Acceleration:=36000 ,
                    Deceleration:=36000
        );
        MC_ReadActualPosition_0(
                            Axis:=Axis1 ,
                            Enable:=TRUE ,
        );
        IF MC_ReadActualPosition_0.Valid = TRUE THEN
            IF MC_ReadActualPosition_0.Position > 1000 THEN
                Process := 2;
            END_IF
        END_IF
    2://Execute the MC_MoveAbsolute
        MC_MoveRelative_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
        );
        MC_MoveAbsolute_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Position:=500 ,
                    Velocity:=3600 ,
                    Acceleration:=36000 ,
                    Deceleration:=36000 ,
                    Direction:=positive
        );
        IF MC_MoveAbsolute_0.Done = TRUE THEN
            MC_MoveAbsolute_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 3;
        END_IF
    3://End
        //No Operation
END_CASE
```

As shown below, if overwriting occurs when the "Execute" flag of "MC_MoveAbsolute" is set to TRUE, the "CommandAborted" flag of "MC_MoveRelative" will be set to TRUE during execution, causing the processing to be suspended.



"MC_ReadActualPosition" is a function block that acquires the actual position of the axis. For details on each function block, refer to the *GM1 Series Reference Manual (Instruction)*.

11.5.3 Switching the Control Mode

This function allows the user to switch the control mode between torque control, velocity control, and position control that can be used by the motion function of the GM1 controller.

The control mode to be used can be switched using "SMC_SetControllerMode" of the "SM3_Basic" function block.

■ Explanation of functions

For the control mode to be used, specify one of the values of enumeration "SMC_CONTROLLER_MODE" that are shown in the following table.

Control mode	Value	Description
SMC_torque	1	Torque control mode
SMC_velocity	2	Velocity control mode
SMC_position	3	Position control mode

Info.

- For the GM1 controller, the default is position control mode.
- For program examples of velocity control mode, refer to "11.5.4 Velocity Control". For program examples of torque control mode, refer to "11.5.5 Torque Control".

11.5.4 Velocity Control

Velocity control is control that runs the motor at the specified velocity.
Velocity control can be executed with the "MC_MoveVelocity" function block.

■ Explanation of functions

- When "Execute" is set to TRUE, the motor continues to run in the specified direction ("Direction") and at the specified velocity ("Velocity").
- You can specify velocity ("Velocity", unit: u/s), acceleration ("Acceleration", unit: u/s^2), deceleration ("Deceleration", unit: u/s^2), and jerk ("Jerk", unit: u/s^3) that are used during control.

■ Program example

The following is an ST program example that performs velocity control in the forward direction at a velocity of 360 u/s, at an acceleration of 3,600 u/s^2, and at a deceleration of 3,600 u/s^2. Before executing the "MC_MoveVelocity" function block, use the "MC_Power" function block to turn ON the servo corresponding to the target axis.

ST program

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Change controller mode to SMC_velocity
        SMC_SetControllerMode_0(
                            Axis:=Axis1 ,
                            bExecute:=TRUE ,
                            nControllerMode:=SMC_velocity
        );
        IF SMC_SetControllerMode_0.bDone = TRUE THEN
            SMC_SetControllerMode_0(
                                Axis:=Axis1 ,
                                bExecute:=FALSE
            );
            Process := 2;
        END_IF
    2://Execute the MC_MoveVelocity
        MC_MoveVelocity_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Velocity:=360 ,
                    Acceleration:=3600 ,
                    Deceleration:=3600 ,
                    Direction:=positive
        );
        IF MC_MoveVelocity_0.InVelocity = TRUE THEN
            MC_MoveVelocity_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 3;
        END_IF
    3://Change Velocity to 0
        MC_MoveVelocity_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Velocity:=0 ,
                    Acceleration:=3600 ,
                    Deceleration:=3600 ,
                    Direction:=positive
        );
END_CASE
```

In the GM1 controller, "MC_MoveVelocity" is executed in position control mode or velocity control mode (Default: Position control mode).

In the program example, "SMC_SetControllerMode" is used to switch to velocity control mode (SMC_velocity).

In the program example, velocity control is started at the rising edge of the "Execute" flag of "MC_MoveVelocity".

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, "MC_MoveAbsolute_0" must be called every cycle. Otherwise, the operation will terminate with an error.

When the velocity of the axis reaches the target velocity ("Velocity"), the "InVelocity" flag is set to TRUE.

To change parameter settings such as target velocity ("Velocity") during operation, temporarily set the "Execute" flag to FALSE beforehand. After parameter settings have been changed, if the "Execute" flag is set back to TRUE, the changed parameter settings will be applied.

To stop the axis, change the setting of "Velocity" to 0 .

11.5.5 Torque Control

Torque control is performed at the specified torque.

Torque control can be executed with the "PMC_SetTorque" function block.

■ Explanation of functions

- While input "bEnable" is set to TRUE, torque control is executed with input "fTorque" (unit: %).
- Before executing the "PMC_SetTorque" function block, use the "MC_Power" function block to turn ON the servo corresponding to the target axis.

■ Program example

The following is an ST program example that executes torque control with a torque of 30%.

ST program

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Change controller mode to SMC_velocity
        SMC_SetControllerMode_0(
                            Axis:=Axis1 ,
                            bExecute:=TRUE ,
                            nControllerMode:=SMC_torque
        );
        IF SMC_SetControllerMode_0.bDone = TRUE THEN
            SMC_SetControllerMode_0(
                                Axis:=Axis1 ,
                                bExecute:=FALSE
            );
            Process := 2;
        END_IF
    2://Execute the PMC_SetTorque
        PMC_SetTorque_0(
                    Axis:=Axis1 ,
                    bEnable:=TRUE ,
                    fTorque:=30
        );
        PMC_ReadActualTorque_0(
                            Axis:=Axis1 ,
                            Enable:=TRUE
        );
        IF PMC_ReadActualTorque_0.Valid = TRUE THEN
            IF PMC_ReadActualTorque_0.Torque >= 30 THEN
                PMC_SetTorque_0(
                            Axis:=Axis1 ,
                            bEnable:=FALSE
                );
                Process := 3;
            END_IF
        END_IF
    3://Change Torque to 0
        PMC_SetTorque_0(
                    Axis:=Axis1 ,
                    bEnable:=TRUE ,
                    fTorque:=0
        );
END_CASE
```

In the GM1 controller, "PMC_SetTorque" is executed in torque control mode (Default: Position control mode).

In the program example, "SMC_SetControllerMode" is used to switch to torque control mode (SMC_torque).

The torque value is changed at the rising edge of the "Execute" flag.

To change the torque value ("fTorque") during operation, temporarily set the "Execute" flag to FALSE beforehand. After parameter settings have been changed, if the "Execute" flag is set back to TRUE, the changed parameter settings will be applied.

"MC_ReadActualTorque" is a function block that acquires the actual torque value of the axis. For details on each function block, refer to the *GM1 Series Reference Manual (Instruction Edition)*.

Info.

- In the torque control mode (SMC_torque), the axis cannot be stopped using "MC_Stop".
- To stop the axis, set "fTorque" to 0 and execute again.
- When performing torque control for MINAS A5B/A6B, the speed is limited by the Max motorspeed (16#6080:00) set in the PDO mapping. Set the speed in advance.

11.5.6 Stop

This function stops the motor.

Stop can be executed with the "MC_Halt" or "MC_Stop" function block.

■ Explanation of functions

- MC_Halt: Allows the user to execute another function block during stop processing.
- MC_Stop: Does not allow the user to execute another function block during stop processing. When input "Execute" is set to FALSE, this function allows the user to execute another function block.
- You can specify deceleration ("Deceleration", unit: u/s^2) and jerk ("Jerk", unit: u/s^3) that are used during stop processing.

■ Program examples

The following is an ST program example.

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Axis1 ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE THEN
            Process := 1;
        END_IF
    1://Execute the MC_MoveRelative
        MC_MoveVelocity_0(
                    Axis:=Axis1 ,
                    Execute:=TRUE ,
                    Velocity:=360 ,
                    Acceleration:=3600 ,
                    Deceleration:=3600 ,
                    Direction:=positive
        );
        IF MC_MoveVelocity_0.InVelocity = TRUE THEN
            MC_MoveVelocity_0(
                        Axis:=Axis1 ,
                        Execute:=FALSE
            );
            Process := 2;
        END_IF
    2://Execute the MC_stop
        MC_Stop_0(
            Axis:=Axis1 ,
            Execute:=TRUE ,
            Deceleration:=1000
        );
        IF MC_Stop_0.Done = TRUE THEN
            MC_Stop_0(
                Axis:=Axis1 ,
                Execute:=FALSE ,
            );
            Process := 1;
        END_IF
END_CASE
```

In the program example, stop operation is started at the rising edge of the "Execute" flag of "MC_Stop".

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, "MC_Stop_0" must be called every cycle. Otherwise, the operation will terminate with an error.

When the stop operation is completed, the "Done" flag is set to TRUE.

Unless the "Execute" flag of "MC_Stop" is set to FALSE, axis control cannot be executed by other function blocks.

11.6 Synchronous Operation

This section explains synchronous cam operation, synchronous gear operation, and synchronous phase operation that use function blocks.

11.6.1 Synchronous Cam Operation

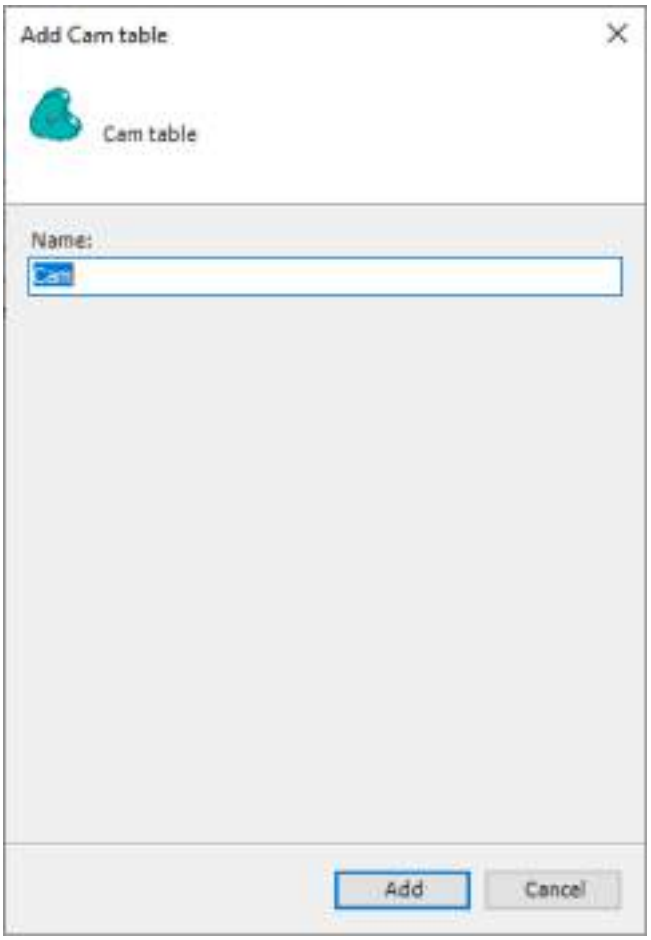
To perform synchronous cam operation, you must set up a cam table.

Synchronous cam operation: How to use cam

This section explains how to change graphs in a cam table after adding a cam table object.

1 2 Procedure

1. Right-click the [Application] object in the navigator pane and then select **Add Object>Cam table** from the context-sensitive menu that is displayed.
   The "Add Cam table" dialog box will be displayed.



2. Enter a cam table name in the "Name" field and then click the [Add] button.

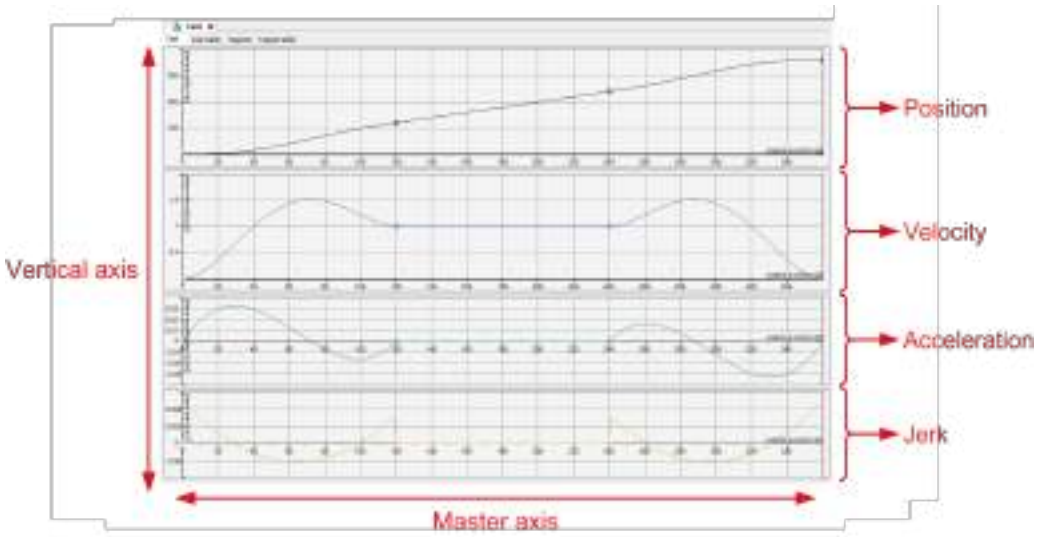
A cam table object will be added.

The entered cam table name will be used when you specify a cam table with a function block.

The ""Cam" tab displays position, velocity, acceleration, and jerk graphs.

The horizontal axis represents the master axis and the vertical axis represents the slave axis.

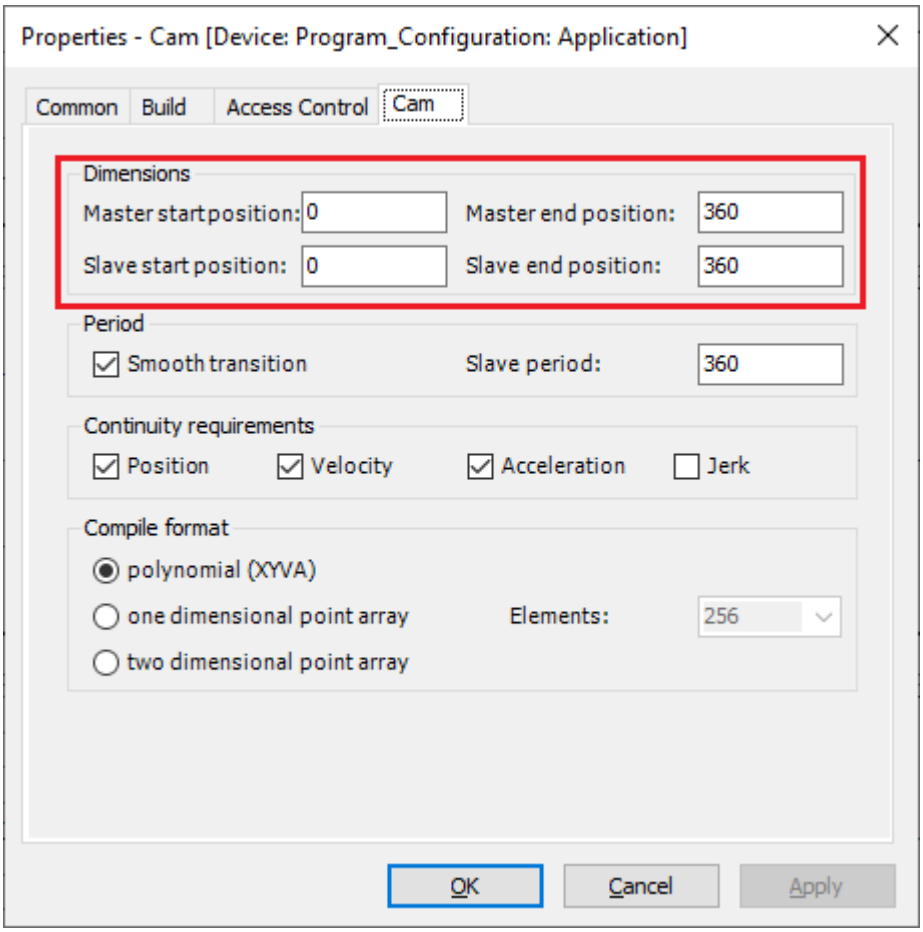
The "Cam table" tab displays numerical values that represent the graphs displayed in the "Cam" tab.



Info.

- You can change the minimum and maximum scale values on the horizontal axis (master axis) and vertical axis (slave axis) within a graph.

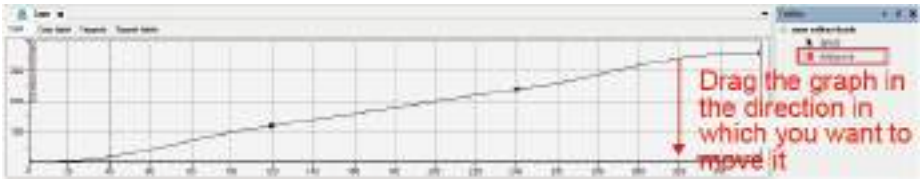
  Right-click the [Cam] object in the navigator pane and then select "Properties" from the context-sensitive menu that is displayed. In the Properties window, click the "Cam" tab and change the values in the Dimensions section.



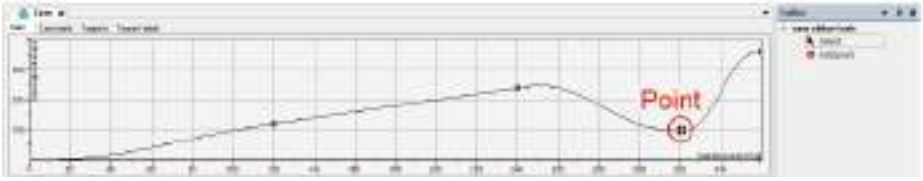
In the "Compile format" section, select "polynomial (XYVA)". Do not select another option.

3. Select "Add pointer" in Toolbox, click the graph and then drag it in the direction in which you want to move it.

A cam table object will be added.

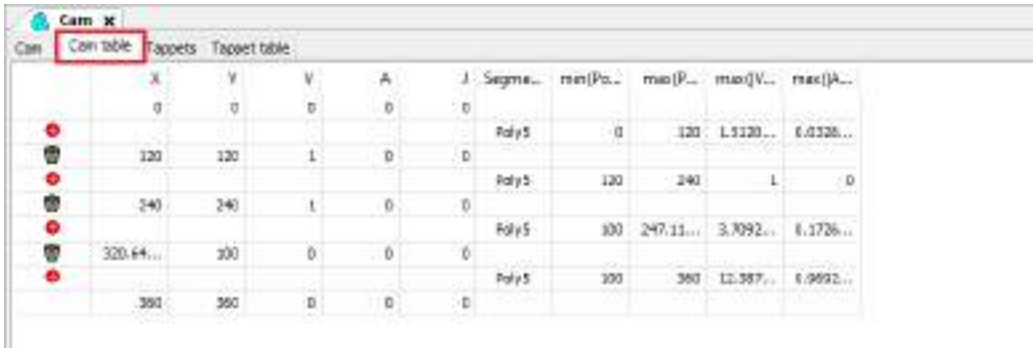


A point will be added to the position at which you click the mouse and the graph will be moved in the direction in which you drag the point.



Info.

- To delete the point, select the point and press the "Del" key.
- You can also check and change a cam table numerically, not in graph format. Select the "Cam table" tab. Numerical values that are changed will also be reflected in graphs in the "Cam" tab.



Synchronous cam operation: Cam control (POU programming)

After specifying the master axis and slave axis of the cam table with the "MC_CamTableSelect" function block, start synchronous cam operation with the "MC_CamIn" function block.

The following is an ST program example that moves the slave axis in conjunction with a cam table that is created.

For the cam table name, enter the object name of the cam table that has been added.

The following is an ST program example that moves the slave axis in conjunction with a cam table that is created.

For the cam table name, enter the object name of the cam table that has been added.

```
CASE Process OF
    0:(*Servo On.
        MC_Power_0(
                Axis:=Master ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        MC_Power_1(
                Axis:=Slave ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE AND MC_Power_1.Status = TRUE THEN
            Process := 1;
        END_IF
    1:(*Load the cam table using MC_CamTableSelect.
        MC_CamTableSelect_0(
                        Master:=Master ,
                        Slave:=Slave ,
                        CamTable:=Cam ,
                        Execute:=TRUE ,
                        Periodic:=TRUE ,
                        MasterAbsolute:=TRUE ,
                        SlaveAbsolute:=TRUE
        );
        IF MC_CamTableSelect_0.Done = TRUE THEN
            Process := 2;
        END_IF
    2:(*Start cam control. Sync is complete when MC_CamIn_0.InSync becomes TRUE.
        MC_CamIn_0(
                Master:=Master ,
                Slave:=Slave ,
                Execute:=TRUE ,
                MasterOffset:=0 ,
                SlaveOffset:=0 ,
                MasterScaling:=1 ,
                SlaveScaling:=1 ,
                StartMode:=ramp_in ,
                CamTableID:= MC_CamTableSelect_0.CamTableID,
                VelocityDiff:=360 ,
                Acceleration:=3600 ,
                Deceleration:=3600
        );
        IF MC_CamIn_0.InSync = TRUE THEN
            Process := 3;
        END_IF
    3:(*Control the master axis with MC_MoveRelative and control the slave axis synchronously.
        MC_MoveVelocity_0(
                    Axis:=Master ,
                    Execute:=TRUE ,
                    Velocity:=360 ,
                    Acceleration:=3600 ,
                    Deceleration:=3600 ,
                    Direction:=positive
        );
        MC_CamIn_0(
                Master:=Master ,
                Slave:=Slave ,
                Execute:=TRUE ,
        );
    4:(*Call MC_CamOut to end synchronization.
        MC_CamIn_0(
                Master:=Master ,
                Slave:=Slave ,
                Execute:=FALSE ,
        );
        MC_MoveVelocity_0(
                    Axis:=Master ,
                    Execute:=FALSE
        );
        MC_CamOut_0(
                Slave:=Slave ,
                Execute:=TRUE
        );

END_CASE
```

In the program example, the master axis is controlled by "MC_MoveVelocity" and the slave axis moves in sync with the master axis.

When the synchronous operation is completed, the "MC_CamIn_0.InSync" flag is set to TRUE.

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, "MC_MoveVelocity_0" and "MC_CamIn_0" must be called every cycle. Otherwise, the operation will terminate with an error.

To separate the slave axis from the master axis and finish the synchronous cam operation, use the "MC_CamOut" function block. When "MC_CamOut" is called, the synchronous operation is terminated and the slave axis continues to move at the same velocity as when the synchronous operation is terminated. Because the operation continues even after the slave axis is separated from the master axis, you must stop the slave axis with the "MC_Halt" function block or another function block.

11.6.2 Synchronous Gear operation

This function performs synchronous gear operation.

Synchronous gear operation: Gear control (POU programming)

Synchronous gear operation can be started and finished using the following function blocks.

- MC_GearIn: Specifies the gear ratio of the master axis and slave axis and starts synchronous gear operation
- MC_GearOut: Separates the slave axis from the master axis and finishes synchronous gear operation

Because the operation continues even after the slave axis is separated from the master axis, you must stop the slave axis with the "MC_Halt" function block or another function block.

The following is an ST program example that performs synchronous gear operation with a gear ratio of 2:1.

```
CASE Process OF
    0://Servo On
        MC_Power_0(
                Axis:=Master ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        MC_Power_1(
                Axis:=Slave ,
                Enable:=TRUE ,
                bRegulatorOn:=TRUE ,
                bDriveStart:=TRUE
        );
        IF MC_Power_0.Status = TRUE AND MC_Power_1.Status =TRUE THEN
            Process := 1;
        END_IF
    1://Control the master axis with MC_MoveVelocity and control the slave axis synchronously.
        MC_GearIn_0(
                Master:=Master ,
                Slave:=Slave ,
                Execute:=TRUE ,
                RatioNumerator:=2 ,
                RatioDenominator:=1 ,
                Acceleration:=3600 ,
                Deceleration:=3600
        );
        MC_MoveVelocity_0(
                    Axis:=Master ,
                    Execute:=TRUE ,
                    Velocity:=360 ,
                    Acceleration:=3600 ,
                    Deceleration:=3600 ,
                    Direction:=positive
        );
        IF MC_GearIn_0.InGear = TRUE THEN
            //Gear In OK
        END_IF
    2://Call MC_GearOut to end synchronization.
        MC_GearIn_0(
                Master:=Master ,
                Slave:=Slave ,
                Execute:=FALSE
        );
        MC_MoveVelocity_0(
                    Axis:=Master ,
                    Execute:=FALSE
        );
        MC_GearOut_0(
                Slave:=Slave ,
                Execute:=TRUE
        );
        IF MC_GearOut_0.Done THEN
            //Gear Out OK
        END_IF
END_CASE
```

In the program example, the master axis is controlled by "MC_MoveVelocity" and the slave axis moves in sync with the master axis.

When the synchronous operation is completed, the "MC_GearIn_0.InGear" flag is set to TRUE.

The "Busy" flag is set to TRUE during execution. While the "Busy" flag is set to TRUE, "MC_MoveVelocity_0" and "MC_GearIn_0" must be called every cycle. Otherwise, the operation will terminate with an error.

To separate the slave axis from the master axis and finish the synchronous gear operation, use the "MC_GearOut" function block. When "MC_GearOut" is called, the synchronous operation is terminated and the slave axis continues to move at the same velocity as when the synchronous operation is terminated. Because the operation continues even after the slave axis is separated

from the master axis, you must stop the slave axis with the "MC_Halt" function block or another function block.

11.7 Multi-axis Operation

This section explains multi-axis operations such as linear interpolation and circular interpolation.

11.7.1 Overview of Interpolation Control

Interpolation control is an operation that controls the loci of multiple axes by interlocking them. The GM1 controller supports the following interpolation controls.

Function			Description	Remarks
Settings	Target position specification method		Relative coordinates / Absolute coordinates	Relative: Value relative to the current value Absolute: Coordinates based on the machine zero point
	Operation setting method	Tools	CNC editor	
		Setting method	G-code input	
	Interpolation operation startup		Setting: CNC editor Operation startup: Startup by FB	
	Interpolation grouping		Determining the axis to be used at the time of startup by FB	
Interpolation	Linear interpolation	Composite speed specification	Maximum 3-axis linear interpolation	Configuration parameters Target coordinates (X, Y, Z), velocity (F), acceleration / deceleration (E)
	Circular interpolation	Center point specification	Circular interpolation by specifying a center point	Configuration parameters Circular interpolation plane (XY, YZ, ZX) Target coordinates (X, Y, X), center point (I, J, K) Velocity (F), acceleration / deceleration (E)
		Radius	Circular interpolation based on the current value, target coordinates, and radius	Configuration parameters Circular interpolation plane (XY, YZ, ZX) Target coordinates (X, Y, X), radius (R) Velocity (F), acceleration / deceleration (E)
	Helical interpolation		Circular interpolation + Helical axis	Same as circular interpolation (Center point and radius can be specified)
Coordinate conversion(Note 1)	Absolute coordinate conversion		Coordinates are converted by moving the coordinate system in parallel from the reference coordinate system or by rotating it using an absolute value.	Configuration parameters Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C)

Function		Description	Remarks
	Relative coordinate conversion	Coordinates are converted by moving the coordinate system in parallel from the reference coordinate system or by rotating it using a relative value.	Configuration parameters Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C)
	Coordinate system setting	Sets the coordinate system definitions using the absolute position for the current orientation and current position. (For coordinate system alignment calibration)	Configuration parameters Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C)
	Coordinate conversion resetting	Resets from the state after coordinate conversion was performed as listed above to the reference coordinate system existed before the coordinate conversion was performed.	
Other movements	Dwell time	Specifying the waiting time between set movements	
	P-point/C-point operation	P-point operation: Proceeds to the next operation speed without stopping at every operation command. C-point operation: Proceeds to the next operation after pausing when an operation is completed.	Determined by the argument of SMC_CheckVelocities and the angle formed during operation.
	Jump	Jumps to the line number specified by the G code and repeats processing.	
Program-based movement change	Change of parameter settings such as target values	Using G-code to specify parameters to be changed and starting them from FB (New FB must be created)	Only SMC_CNC_REF can be used.
	Repetitive execution of CNC table	Executing a preset CNC table (interpolation settings) repeatedly	Dedicated FB is used (new FB must be added). Only SMC_CNC_REF can be used.

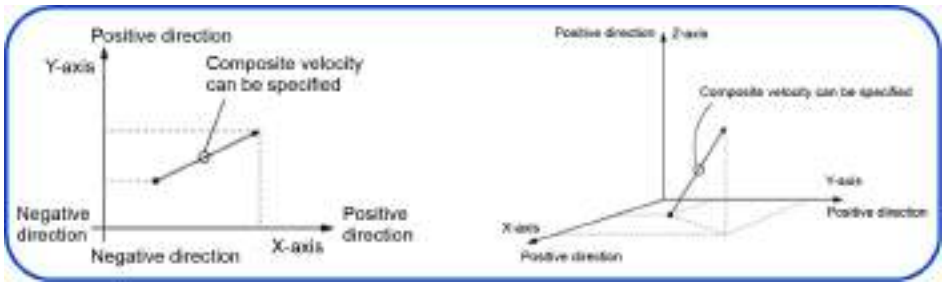
(Note 1) Use SMC_NCDecoder to execute coordinate conversion.

11.7.2 Linear Interpolation and Circular Interpolation

The GM1 controller supports linear interpolation, circular interpolation, and helical interpolation as interpolation controls.

■ Linear interpolation

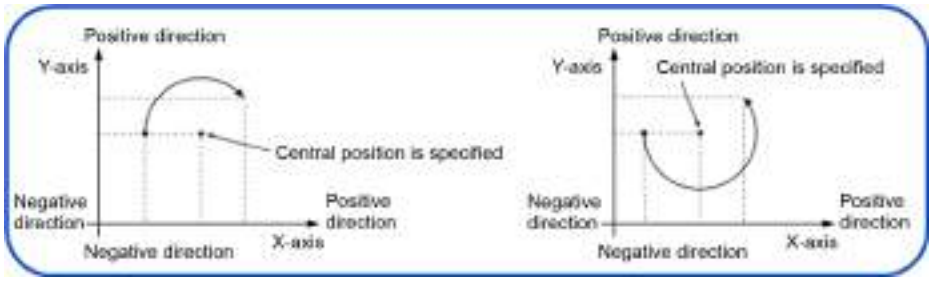
Linear interpolation can be used for two axes or three axes.



■ Circular interpolation

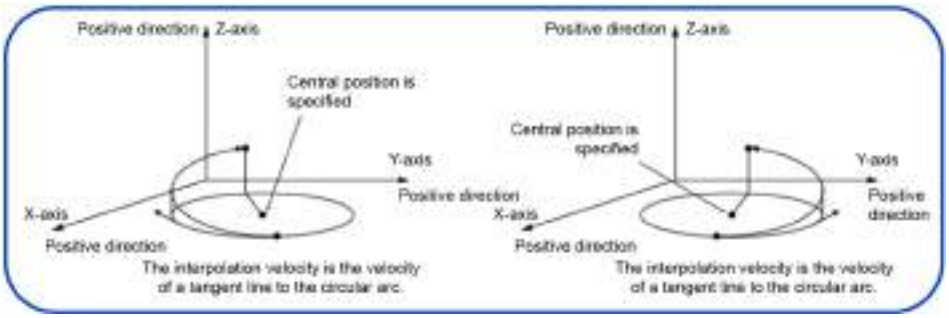
Circular interpolation can be used for two axes and each axis moves along to the locus of a circular arc.

For circular arcs, specify a center point or the radius of the circular arc.



■ Helical interpolation

Helical interpolation can be used for three axes. While performing circular interpolation, helical interpolation allows each axis to move in parallel with the central axis of the circular arc so that they can achieve helical motion.



11.7.3 How to Use Interpolation Control

To use interpolation control with the GM1 controller, you must configure settings and do programming as described below.

■ Setting up a CNC table

Setting up a CNC table makes it possible to perform continuous movements by combining interpolation controls. You can set up a CNC table using GM Programmer and transfer it to the GM1 controller by downloading the project.

For details on how to set up a CNC table, refer to "11.7.4 Registering a CNC Table".

■ Programming in POU

To perform interpolation control using a CNC table that has been set up, you must do programming using function blocks in POU.

The function blocks in the table below are available to perform interpolation control.

List of function blocks for interpolation control

PMC_NCDecoder	This function block decodes the specified SMC_CNC_REF type CNC table into a SMC_OUTQUEUE type CNC table. If a CNC table has been created using SMC_CNC_REF, execute this function block.
PMC_Interpolator2D	This function block executes 2-axis interpolation control using a predefined CNC table. For CNC tables for 2-axis interpolation control, execute this function block.
PMC_Interpolator3D	This function block executes 3-axis interpolation control using a predefined CNC table. For CNC tables for 3-axis interpolation control, execute this function block.

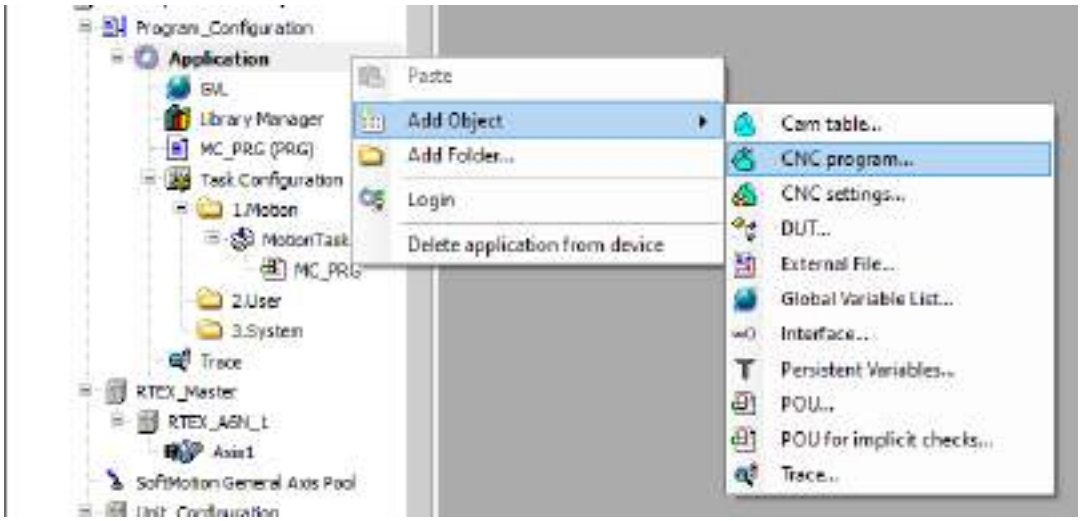
For details on function blocks for interpolation control, refer to "11.7.9 Interpolation Operation Programming: How to Create a Program for Executing Operation" to "11.7.16 Interpolation Operation Programming: Changing Parameter Settings (Converting to Variables in CNC Table)".

POU in which programming has been completed must be registered in MotionTask.

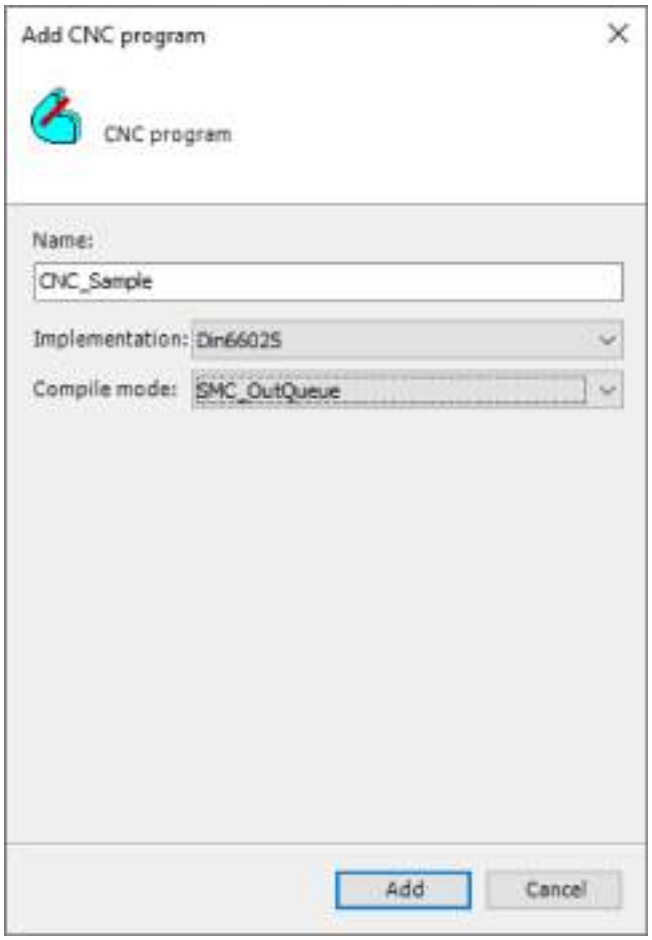
11.7.4 Registering a CNC Table

Set up a CNC table (operation patterns for interpolation control) using GM Programmer.

Right-click the "Application" object in the navigator pane and then select **Add Object**>**CNC program** from the context-sensitive menu that is displayed.



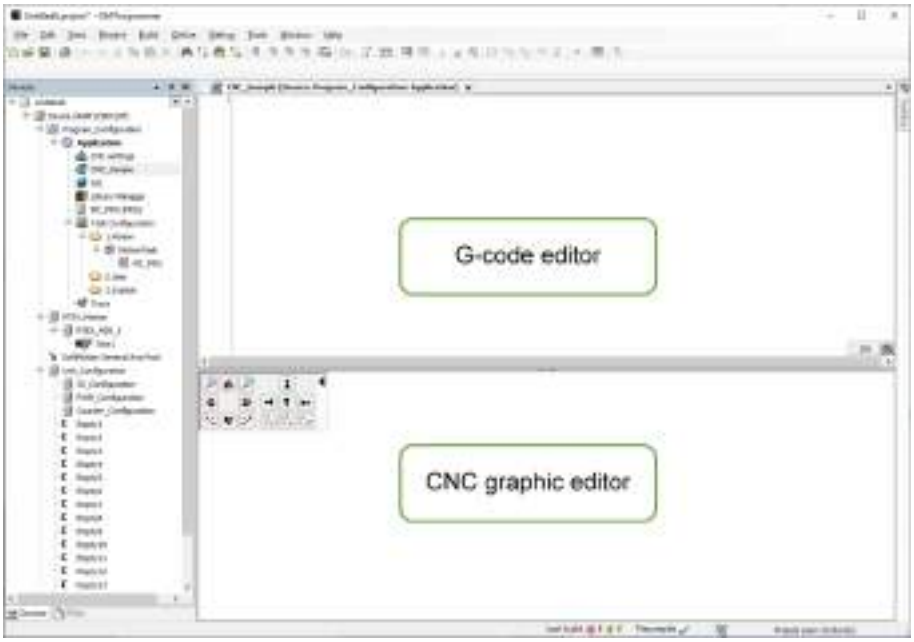
When "CNC program" is selected, the following window is displayed. Specify settings and click [OK].



Setting item		Description
Name	Settings	Specify a name for a CNC table to be created.
Implementation	Din66025	[Recommended] A CNC table is created using G-code.
	Table	[Not supported] A CNC table is created in simple table input format. Because this item is not supported by the GM1 controller, select "Din66025".
Compile mode	SMC_CNC_REF	[Recommended] All interpolation control functions of the GM1 controller can be used.
	SMC_OUTQUEUE	[Not recommended] Movement speed is improved, but changing the settings in CNC tables, joining CNC tables, and other similar operations cannot be performed.
	File	[Not supported] CNC tables are saved as external reference files. Because this mode is not supported by the GM1 controller, do not select.

"CNC settings" and "CNC Program" will be created in the Device tree. "CNC Program" is displayed as the name registered earlier ("CNC_Sample").

The G-code editor and CNC graphic editor are displayed in the right pane of the GM Programmer screen.



■ G-code editor

Enter an interpolation control pattern using G-code.

For details on the input method, refer to "11.7.6 G-code Editor and Coding Rules".

■ CNC graphic editor

The movements coded in the G-code editor are graphically displayed in real time. You can also rotate the display and change the scale.

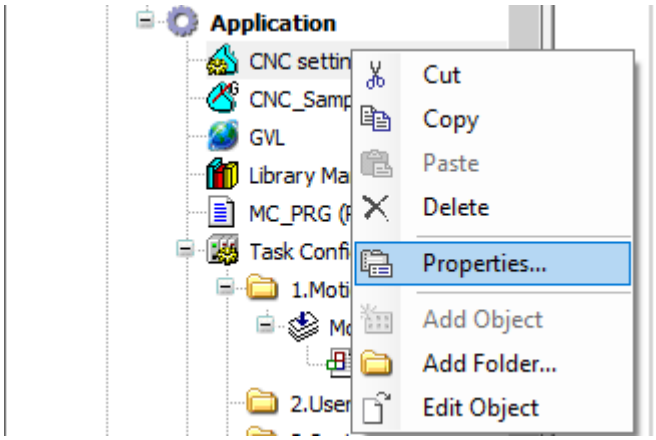
It is possible to change figures in this editor, but this will change codes in the G-code editor, making settings difficult. For this reason, do not change figures.

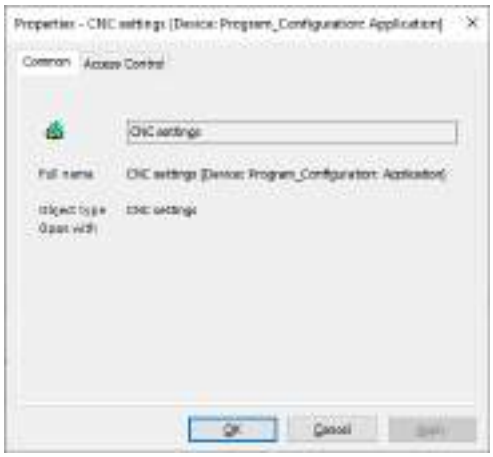
The respective properties of the "CNC Setting" and "CNC Program" objects displayed in the Device tree can be displayed and set by right-clicking each object and then left-clicking appropriate items.

Each window and setting items are explained below.

■ "CNC settings" - "Properties"

In the Device tree of GM Programmer, right-click **Application**>**CNC settings**>**Properties** in this order.

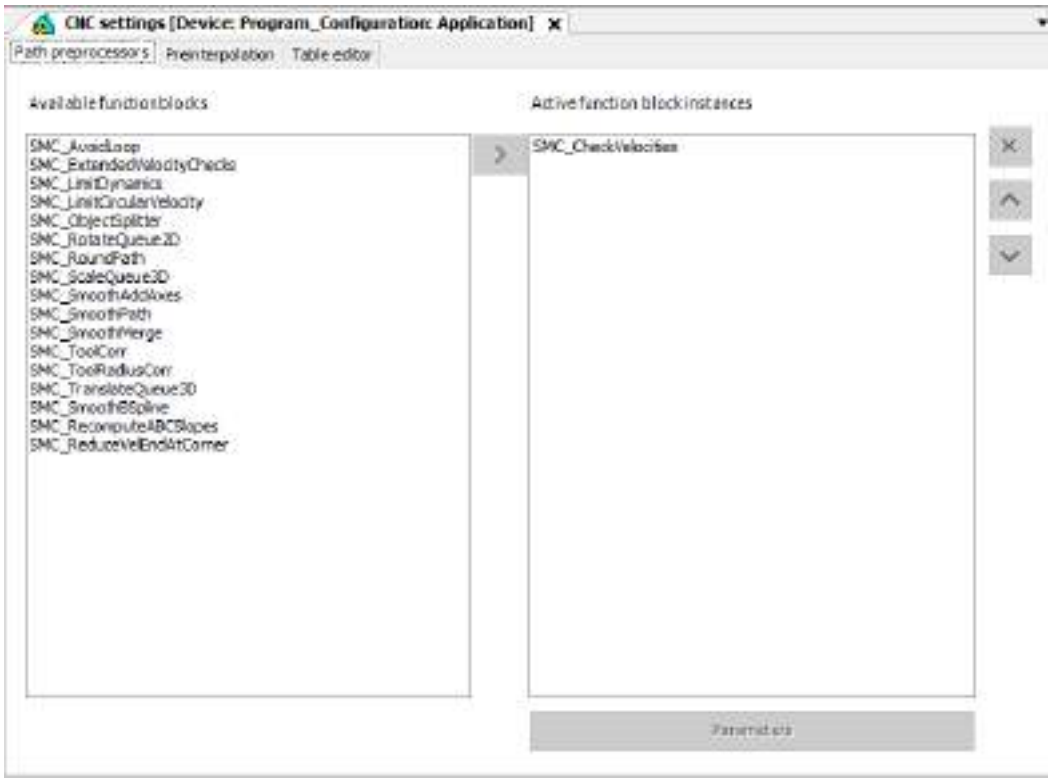




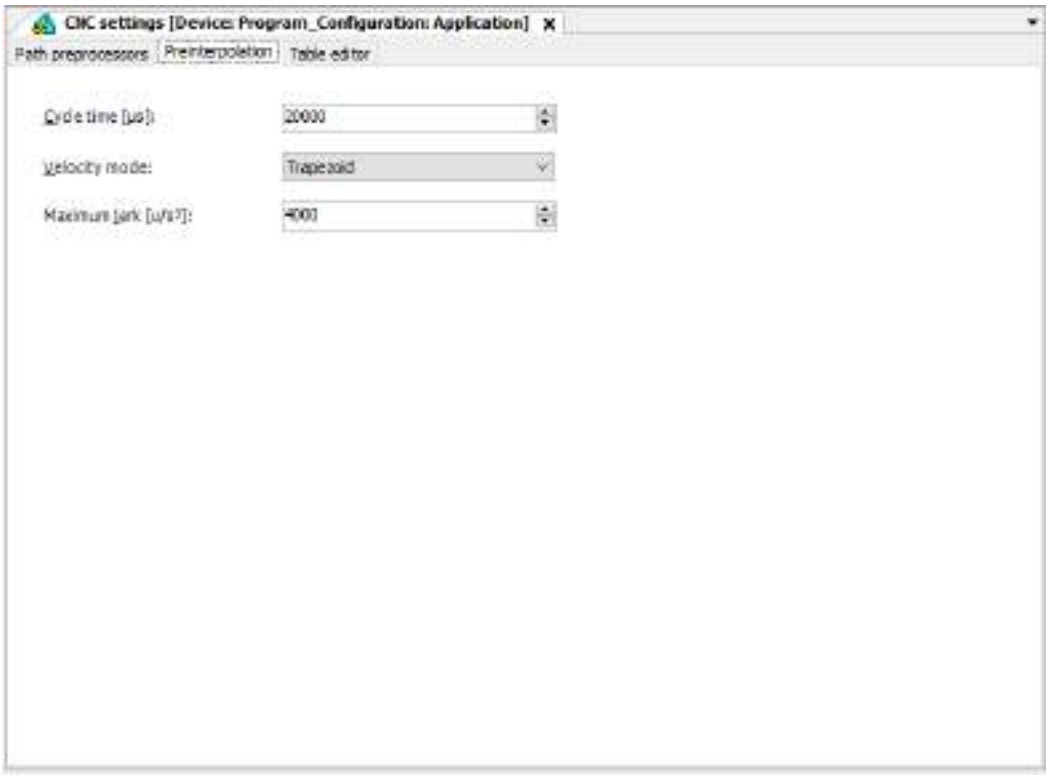
Setting item		Description
Tab name	Settings	
Common	-	You can check the detailed settings of the CNC table.
Access Control	-	You can set access permissions.

■ "CNC settings" - Setting window

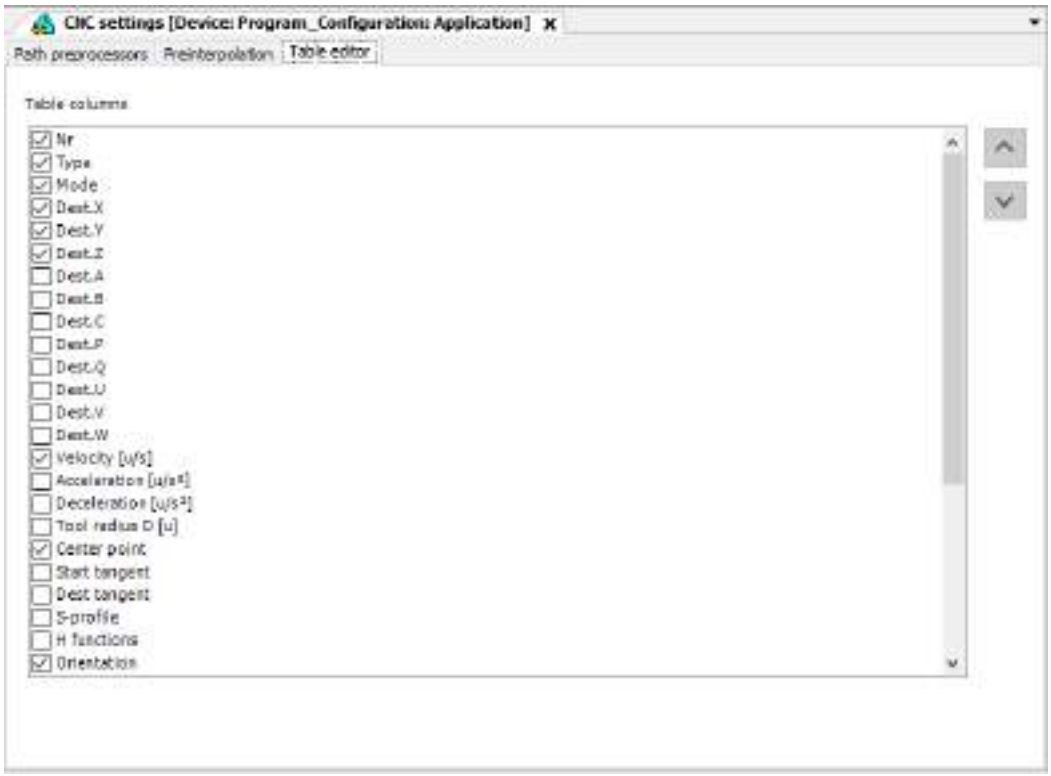
- "Path preprocessors" tab



- "Preinterpolation" tab



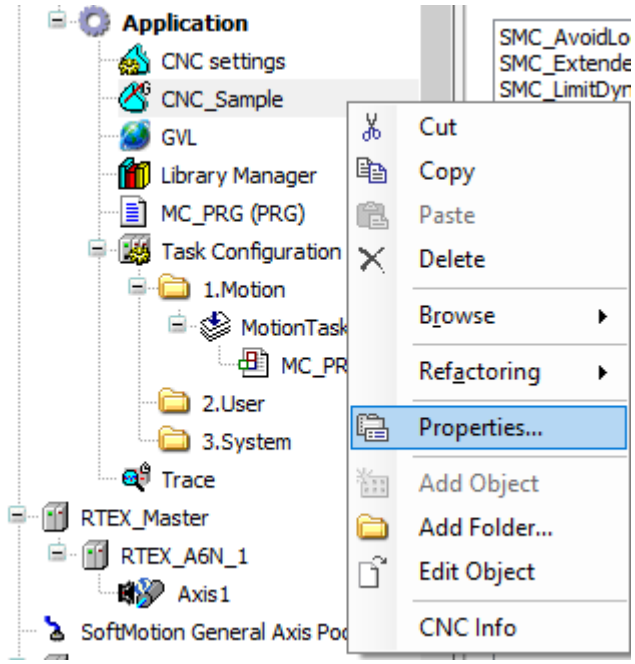
- "Table editor" tab



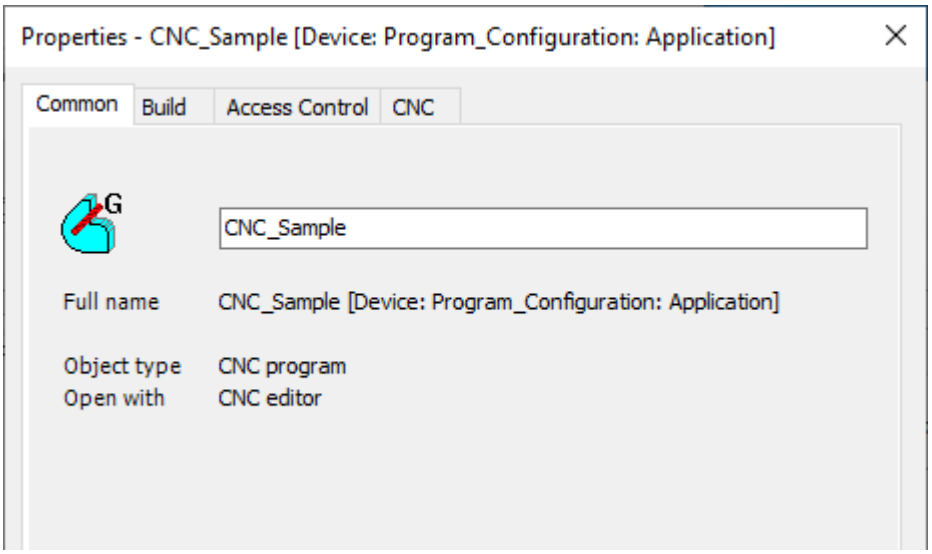
Setting item		Description
Tab name	**Settings**	
Path preprocessors	-	[Not supported] Use this item to perform preprocessing on devices for which interpolation control is to be performed. Because this mode is not supported by the GM1 controller, do not use.
Preinterpolation	Cycle time	Check that the time matches the time displayed in the "MotionTask" window.
	Velocity mode	[Unchangeable] Check that the mode is set to "Trapezoid".
	Maximum jerk	Specify a maximum value for jerk during interpolation control operation.
Table editor	-	This tab is used to perform CNC in "table format". Because the GM1 controller uses G-code (Din66025), there is no need to configure settings in this window.

■ "CNC Program" - "Properties"

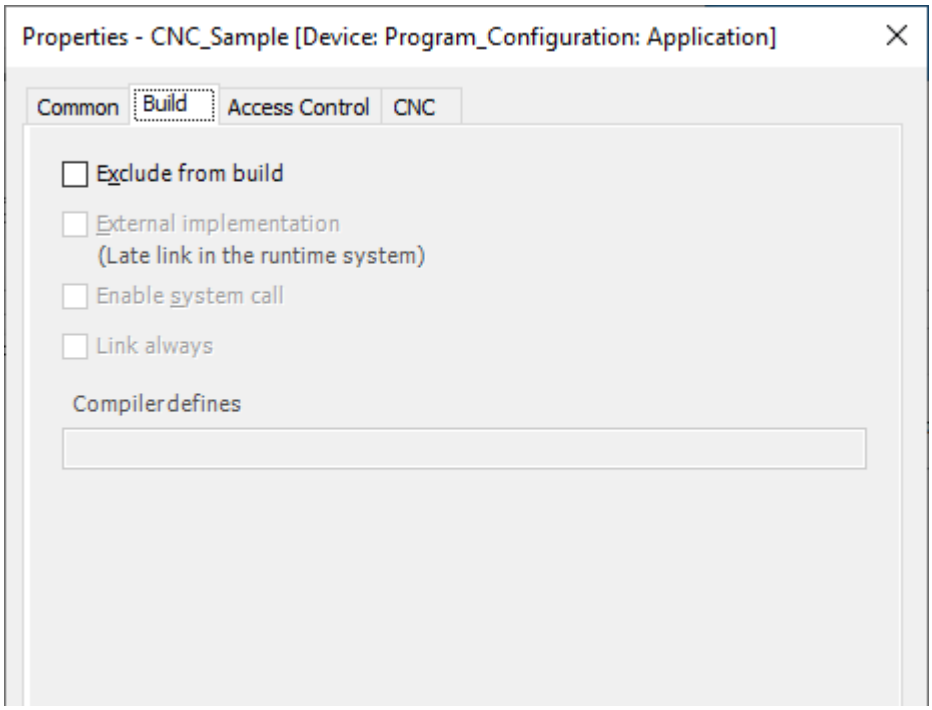
In the Device tree of GM Programmer, right-click **Application**>**CNC_Sample**>**Properties** in this order.



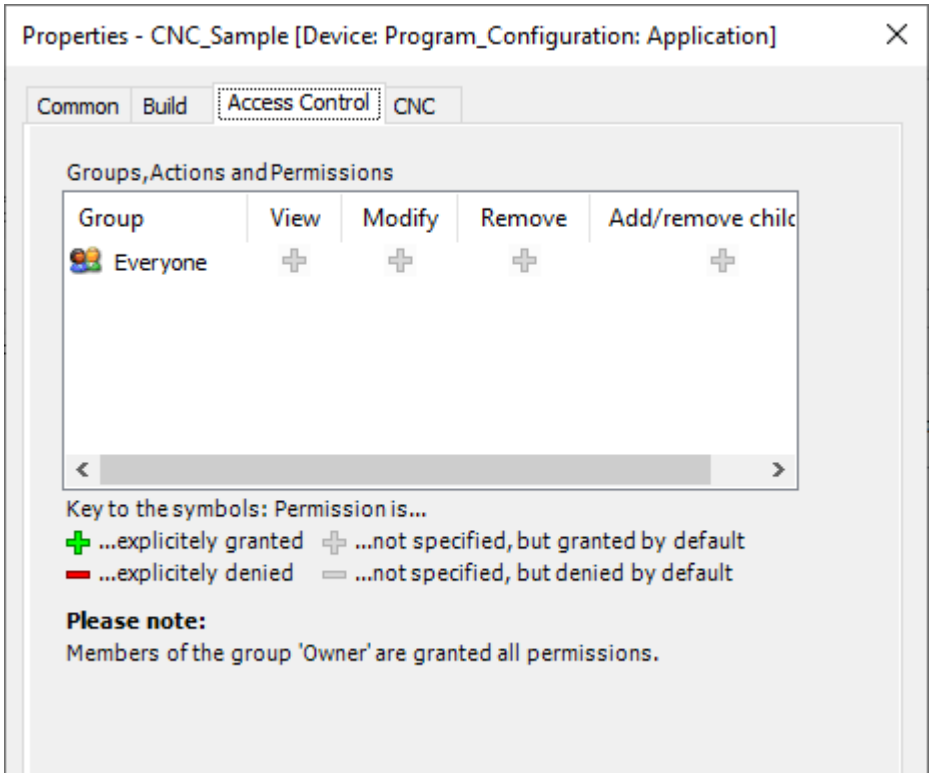
- "Common" tab



- "Build" tab

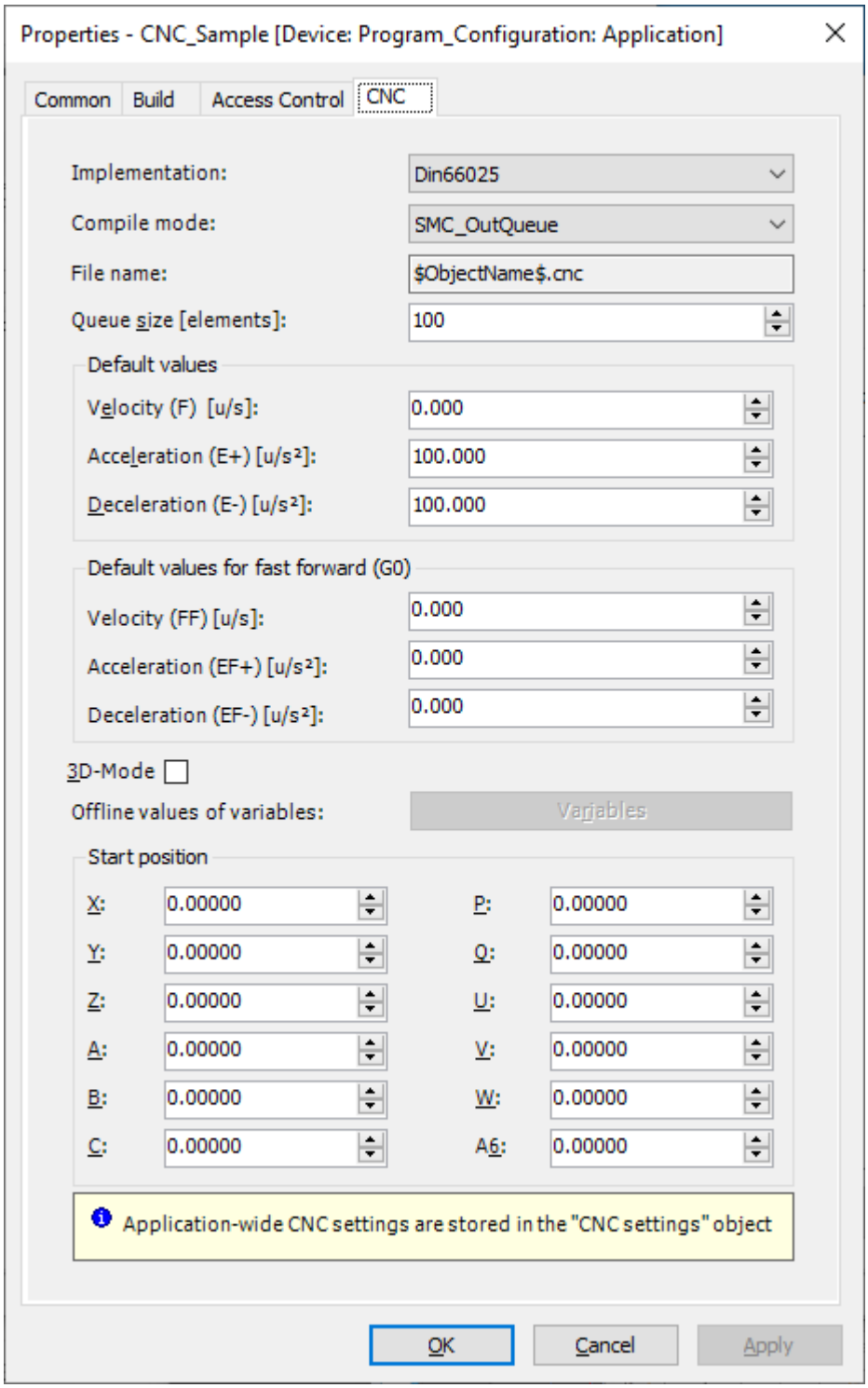


- "Access Control" tab



Setting item		Description
Tab name	**Settings**	
Common	-	This tab displays the CNC table name and object type. To change the CNC table name, change the CNC table name in the white field and click [OK].
Build	-	If the created CNC table is not to be used, select the "Exclude from build" check box and click [OK].
Access Control	-	You can set access permissions.

- "CNC" tab



This window is used to configure initial settings for CNC operation.

Setting item	Description
Implementation	[Recommended] ● Check that "Din66025" has been selected. ● If "Table" has been selected, change to "Din66025".
Compile mode	[Recommended] ● Check that "SMC_CNC_REF" has been selected. ● If "SMC_Outqueue" or "File" has been selected, change to "SMC_CNC_REF".
File name	[Not supported] Leave the default value unchanged. This item is not used.
Queue size	
Default values	Specify default values for velocity, acceleration, and deceleration during interpolation control operation. Unless velocity, acceleration, and deceleration are specified using G-code, interpolation control will be performed using these default values.
Default values for fast forward (G0)	[Not supported] Do not change the default value.
3D-Mode	[Not supported] Do not change the default value.
Start position	For the SMC_OUTQUEUE type, specify the coordinates from which interpolation control is to be started. Because the GM1 controller has the starting position controlled within function blocks, do not change the default values in this area.

11.7.5 Overview of G-code

In the GM1 controller, G-code is used to set operation patterns for interpolation control.

G-code is a coding system used for machine NC programming. The GM1 controller complies with the "Din66025" standards.

For details on operations executed by each G-code, refer to "11.7.7 Movements Executed by Each G-code and Setting Methods".

The G codes listed in the following table are subject to the Warranty.

All other G codes not listed in the following table are not subject to the Warranty.

G-code	Function	Description	Remarks
G1	Linear interpolation	Executes linear interpolation	
G2	Circular interpolation (clockwise)	Executes circular interpolation (clockwise)	End point: X, Y, Z Center point: I (X), J (Y), K (Z) Radius: R
G3	Circular interpolation (counterclockwise)	Executes circular interpolation (counterclockwise)	
G4	Dwell time	Sets a time to wait until next movement is started	Specified time: seconds
G17	Plane specification (X / Y)	Specifies the plane in which circular interpolation is performed, as XY-plane	
G18	Plane specification (X/ Z)	Specifies the plane in which circular interpolation is performed, as XZ-plane	

G-code	Function	Description	Remarks
G19	Plane specification (Y/ Z)	Specifies the plane in which circular interpolation is performed, as YZ-plane	
G20	Conditional jump	Jumps to the line number specified by the G code and executes the described content in a loop unless the jump condition is other than 0.	Jump condition (K) Jump target line (L)
G36	Variable setting	Writes a value to the variable. The write variable can be used to specify the number of jumps for G20. Written to the internal variable unless a variable is specified. (The internal variable is 32-bit variable type: 0 to 4294967295.)	Set value (D) Set variable (O)
G37	Variable increment/ decrement	Increments or decrements the variable set with G36 by the specified value. Applies to the internal variable unless a variable is specified.	Increment/decrement value (D) Increment/decrement variable (O)
G53	Coordinate conversion resetting(Note 1)	Resets the decoder coordinate system (DCS) and returns to the reference coordinate system (MCS) existed before the coordinate conversion was executed.	
G54	Absolute coordinate conversion(Note 1)	Converts from the reference coordinate system (MCS) to the decoder coordinate system (DCS) using an absolute value.	Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C)
G55	Relative coordinate conversion(Note 1)	Converts from the reference coordinate system (MCS) to the decoder coordinate system (DCS) using a relative value.	Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C)
G56	Coordinate reference point resetting(Note 1)	Converts the current orientation and position of the reference coordinate system (MCS) to the specified decoder coordinate system (DCS).	Coordinate system shift values (X, Y, Z) Coordinate system rotation values (A, B, C)
G75	Timing synchronization with SMC_Interpolator	Synchronizes timing with SMC_Interpolator	
G90	Absolute coordinate specification	Specifies target coordinates as absolute coordinates (If G91 is not specified, absolute coordinate specification will be used.)	After G90 is set, absolute coordinate specification remains effective until G91 is set.
G91	Relative coordinate specification	Specifies target coordinates as relative coordinates	After G91 is set, relative coordinate specification remains effective until G90 is set.
G98	Absolute coordinate specification (center point)	Specifies the center point of circular interpolation as absolute coordinates	After G98 is set, absolute coordinate specification remains effective until G99 is set.

G-code	Function	Description	Remarks
G99	Relative coordinate specification (center point)	Specifies the center point of circular interpolation as relative coordinates (If G98 is not specified, relative coordinate specification will be used.)	After G99 is set, relative coordinate specification remains effective until G98 is set.

(Note 1) Use SMC_NCDecoder to execute coordinate conversion.

11.7.6 G-code Editor and Coding Rules

The following coding rules apply to input of G-codes used with the GM1 controller.

■ Line number

For G-code, always enter a line number (N**) at the beginning of each line. Each line number must be unique.

Example:

N10 G01 X0 Y0 Z0 F0

N20 G01 X100 Y100 Z100 F10

Lines can also be renumbered by selecting **CNC>Change CNC Program Numbers** from the menu bar of GM Programmer.

■ Only one G-code per line

For the GM1 controller, only one G-code can be entered in each line. Split each line as below.

- Acceptable coding

  N01 G17

  N02 G20 X_ Y_ I_ J_ F_
- Unacceptable coding

  N01 G17 G20

■ Velocity, acceleration, and deceleration settings

Movement velocity, acceleration, and deceleration settings can be omitted. If they are omitted, the "default values" in the CNC table setting properties will be applied to movements.

To change velocity, acceleration, or deceleration settings in the CNC table, use G-code to set values as described below.

Set value	Code	Remarks
Velocity	Fxxx (xxx: Velocity)	Specify velocity in u/s.
Acceleration, deceleration	Exxx (xxx: Acceleration, deceleration)	xxx > 0: Acceleration specification xxx < 0: Deceleration specification

Velocity, acceleration, and deceleration can be specified in the following ways.

- Example 1: Batch specification

  Once velocity, acceleration, and deceleration are specified, the same velocity, acceleration, and deceleration will be used until their values are changed.

  N00 F100 E10 E-10

N10 G01 X100 Y50 Z10

- Example 2: Sequential specification

  Velocity, acceleration, and deceleration are specified for each G-code. If velocity, acceleration, and deceleration are changed in the CNC table more than once, using this input method can prevent input mistakes.

  N10 G01 X100 Y50 Z10 F100 E10 E-10

  N20 G01 X150 Y100 Z50 F200 E5 E-5

11.7.7 Movements Executed by Each G-code and Setting Methods

This section explains movements executed by G-codes that the GM1 controller supports.

Linear interpolation (G01)

■ Setting rules for linear interpolation

The following rules apply when G-code is used to set linear interpolation for the GM1 controller.

- Settings for axes untargeted for interpolation

  For 2-axis control, set only parameters for the target plane. Do not set parameters for untargeted axes.

- Parameters set for linear interpolation

  The following table shows parameters that must be set for linear interpolation.

Parameter name	Input value
X-axis	X xxx (xxx: Target coordinate)
Y-axis	Y xxx (xxx: Target coordinate)
Z-axis	Z xxx (xxx: Target coordinate)
Velocity	F xxx (xxx: Composite velocity [u/sec])
Acceleration / deceleration	E xxx (xxx > 0: Acceleration [u/sec^2]), (xxx < 0: Deceleration [u/sec^2]) * When E = 0, an error occurs.

- Linear interpolation specification

  Linear interpolation can be specified with G-code.

G-code	Function
G01	Linear interpolation

■ Linear interpolation setting examples

Examples of setting linear interpolation with G-code are shown below. In these examples, target coordinates are set as absolute coordinates.

- 2-axis linear interpolation

  [Setting example]

G-code setting example:
N04 G01 X100 Y100 F100 E1 E-1

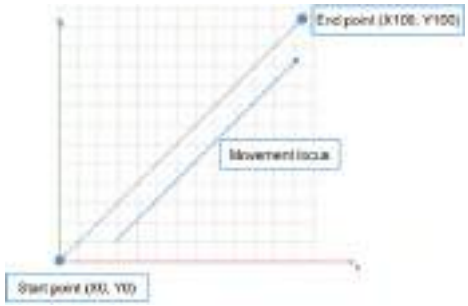
- Explanation of G-code

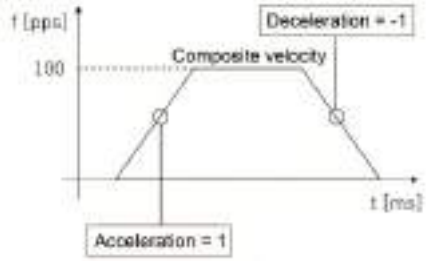
  N04: The X-axis and Y-axis are specified to form an XY-plane. Linear interpolation can be set in the XY-plane according to the following values.

  Current value (X0, Y0), end point (X100, Y100)

  Velocity 100

  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]





- 3-axis linear interpolation

  [Setting example]

G-code setting example: N11 G01 X100 Y100 Z100 F100 E1 E-1

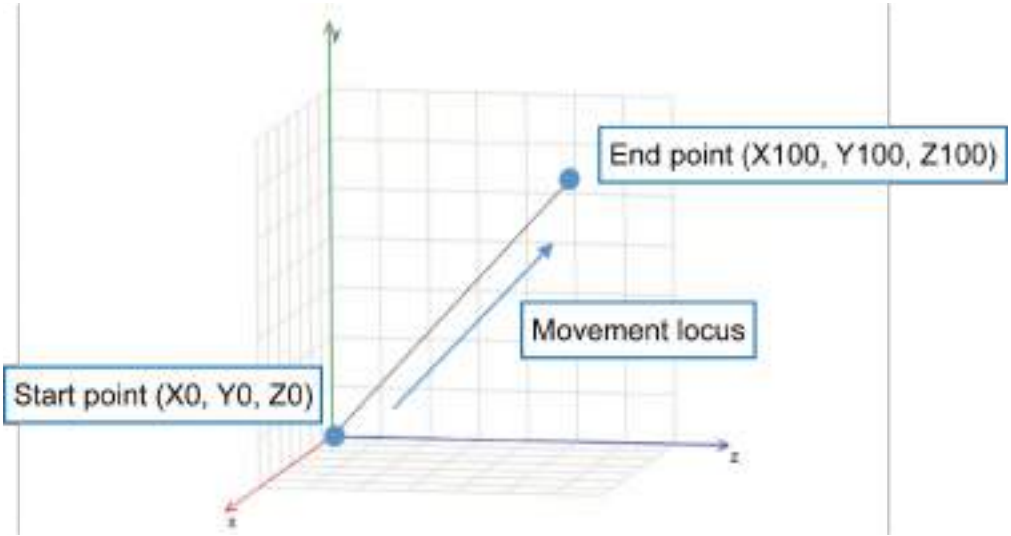
- Explanation of G-code

  N11: Linear interpolation is performed in the XYZ-plane according to the following values.

  Current value (X0, Y0, Z0), end point (X100, Y100, Z100)

  Velocity 100

  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



Circular / helical interpolation (G02, G03) and plane specification (G17, G18, G19)

■ Setting rules for circular interpolation

The following rules apply when G-code is used to set circular interpolation for the GM1 controller.

- Specifying the coordinate plane for circular interpolation

  For circular interpolation, it is necessary to determine the plane targeted for interpolation. If no plane is specified, the XY-plane will be set by default.

  The coordinate plane can be switched according to the G-code specification.

G-code	Function
G17	Circular interpolation in the XY-plane
G18	Circular interpolation in the XZ-plane
G19	Circular interpolation in the YZ-plane

- Settings for axes untargeted for interpolation

  For axes untargeted for interpolation (Z-axis in the XY-plane), do not enter parameters (for coordinates).

  Note that if axes untargeted for circular interpolation (Z-axis in the above example) are specified, "helical interpolation" will be set. For details on helical interpolation, refer to "Helical interpolation setting examples (operation settings)".

- Parameters set for circular interpolation

  The following table shows parameters that must be set for circular interpolation.

Parameter name		Input value
Target coordinates	X-axis	X xxx (xxx: Target coordinate)
	Y-axis	Y xxx (xxx: Target coordinate)
	Z-axis	Z xxx (xxx: Target coordinate)
Center point	X-axis	I xxx (xxx: Center point coordinates)
	Y-axis	J xxx (xxx: Center point coordinates)
	Z-axis	K xxx (xxx: Center point coordinates)
Radius		R xxx (xxx: Circle radius)
Velocity		F xxx (xxx: Composite velocity [u/sec])
Acceleration / deceleration		E xxx (xxx > 0: Acceleration [u/sec^2]), (xxx < 0: Deceleration [u/sec^2]) * When E = 0, an error occurs.

Info.

- For circular arcs, it is necessary to specify a start point and target position, as well as a radius (R) or center point (I, J, K). Set either a center point or radius.

- Specifying the rotational direction of circular arc

  The rotational direction of a circular arc can be switched by specifying a G-code.

G-code	Function
G2	Circular interpolation (clockwise)

G-code	Function
G3	Circular interpolation (counterclockwise)

- Determining a coordinate specification method

  The target position and center point of circular interpolation can be specified as relative coordinates or absolute coordinates. For details on how to set relative coordinates or absolute coordinates, refer to "Absolute coordinate specification (G90)" or "Relative coordinate specification (G91)".

■ Circular interpolation setting examples (center point specification)

Examples of setting circular interpolation with G-code are shown below. In these examples, target coordinates are set as absolute coordinates and center point coordinates are set as relative coordinates.

For circular interpolation, it is necessary to specify the coordinate plane targeted for interpolation. When using circular interpolation, specify a coordinate plane.

- Circular interpolation (XY-plane)

  The following is an example of circular interpolation using an XY-plane.

  [Setting example 1]

```
G-code setting example:
N03   G17
N04   G02 X100 Y0 I50 J0 F100 E1 E-1
```

  - Explanation of G-code

    N03: An XY-plane is selected. (This can be omitted when there is no need to change the plane.)

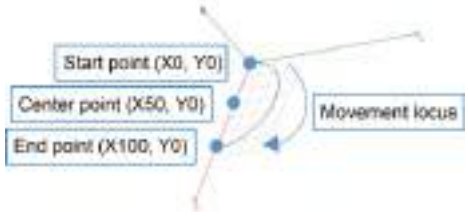
    N04: Circular interpolation is performed in the XY-plane according to the following values.

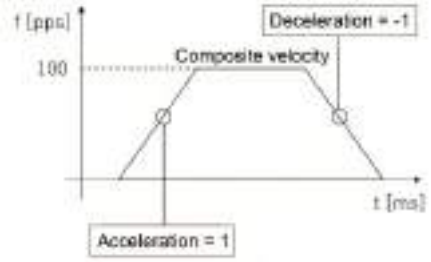
    Current value (X0, Y0), end point (X100, Y0)

    Center point (X50, Y0)

    Velocity 100

    Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]





* A round circle can be specified by omitting an end point or entering the same coordinates for the start point and end point as below. (For center point specification only)

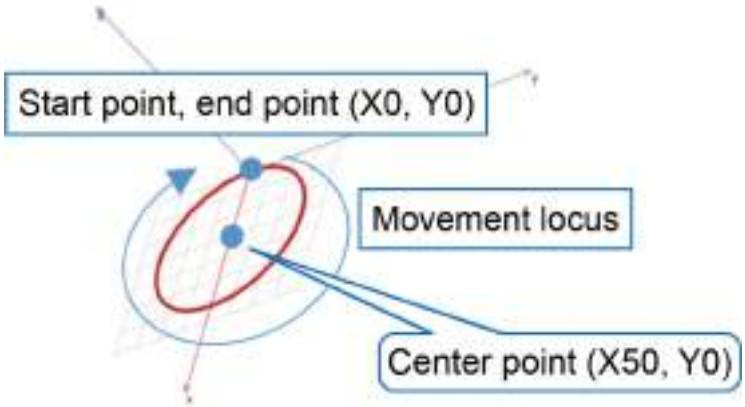
[Setting example 2] (When end point is omitted)

```
G-code setting example:
N03 G17
N04   G02 X100 Y0 I50 J0 F100 E1 E-1
```

[Setting example 3] (When start point = end point)

G-code setting example:
N03 G17
N04 G02 X0 Y0 I50 J0 F100 E1 E-1

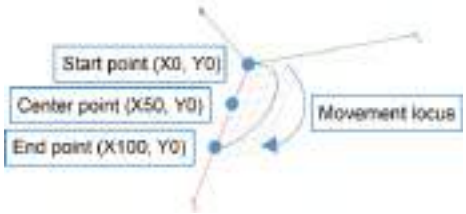
- Explanation of G-code
  N03: An XY-plane is selected.
  N04: Circular interpolation is performed in the XY-plane according to the following values.
  Current value (X0, Y0), end point (X0, Y0 or omitted)
  Center point (X50, Y0)
  Velocity 100
  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]

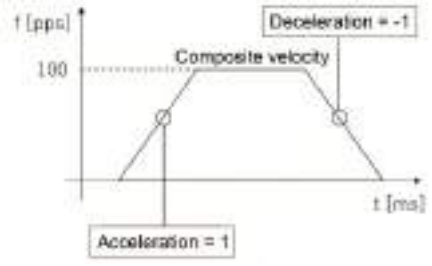


* Circular interpolation can be specified with a radius instead of a center point, as below.
[Setting example 1]

G-code setting example:
N03 G17
N04 G02 X100 Y0 R50 F100 E1 E-1

- Explanation of G-code
  N03: An XY-plane is selected.
  N04: Circular interpolation is performed in the XY-plane according to the following values.
  Current value (X0, Y0), end point (X100, Y0)
  Radius 50
  Velocity 100
  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]

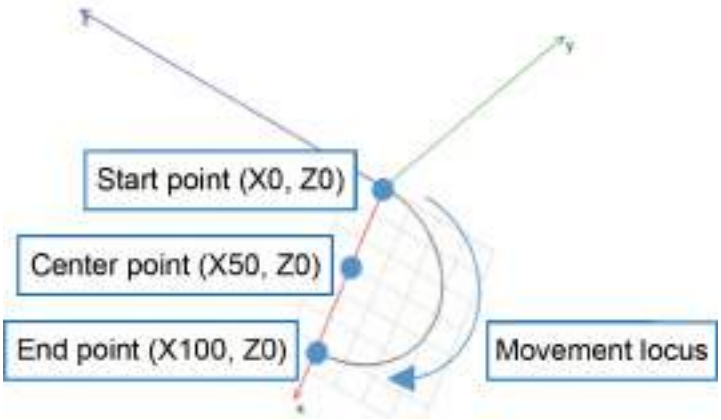




- Circular interpolation (XZ-plane)
  The following is an setting example of circular interpolation using an XZ-plane.
  [Setting example 4]

```
G-code setting example:
N11 G18
N12 G02 X100 Z0 I50 K0 F100 E1 E-1
```

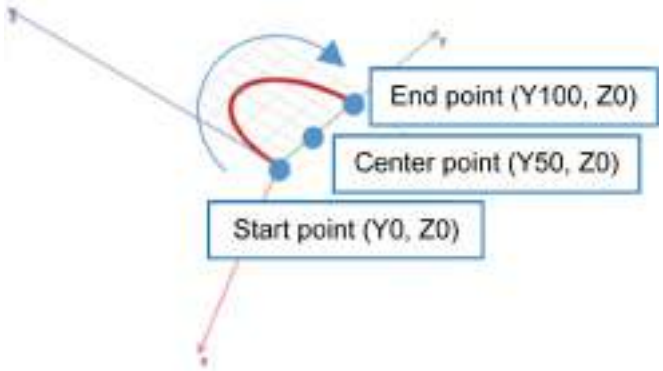
  - Explanation of G-code
    N11: An XZ-plane is selected.
    N12: Circular interpolation is performed in the XZ-plane according to the following values.
    Current value (X0, Z0), end point (X100, Z0)
    Center point (X50, Z0)
    Velocity 100
    Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



- YZ-plane
  The following is an setting example of circular interpolation using an YZ-plane.
  [Setting example 5]

```
G-code setting example:
N10 G19
N11 G02 Y100 Z0 J50 K0 F100 E1 E-1
```

  - Explanation of G-code
    N10: An YZ-plane is selected.
    N11: Circular interpolation is performed in the YZ-plane according to the following values.
    Current value (Y0, Z0), end point (Y100, Z0)
    Center point (Y50, Z0)
    Velocity 100
    Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



Info.

- Note that if axes other than those in the target plane are selected and used, helical interpolation will be performed. For details on helical interpolation, refer to "Helical interpolation setting examples (operation settings)".

■ Helical interpolation setting examples (operation settings)

Examples of setting helical interpolation with G-code are shown below.

[Setting example 1]

```
G-code setting example:
N10 G17
N11 G02 X0 Y0 Z100 I50 J0 F100 E1 E-1
```

- Explanation of G-code

  N10: An XY-plane is selected. (This can be omitted when there is no need to change the plane.)

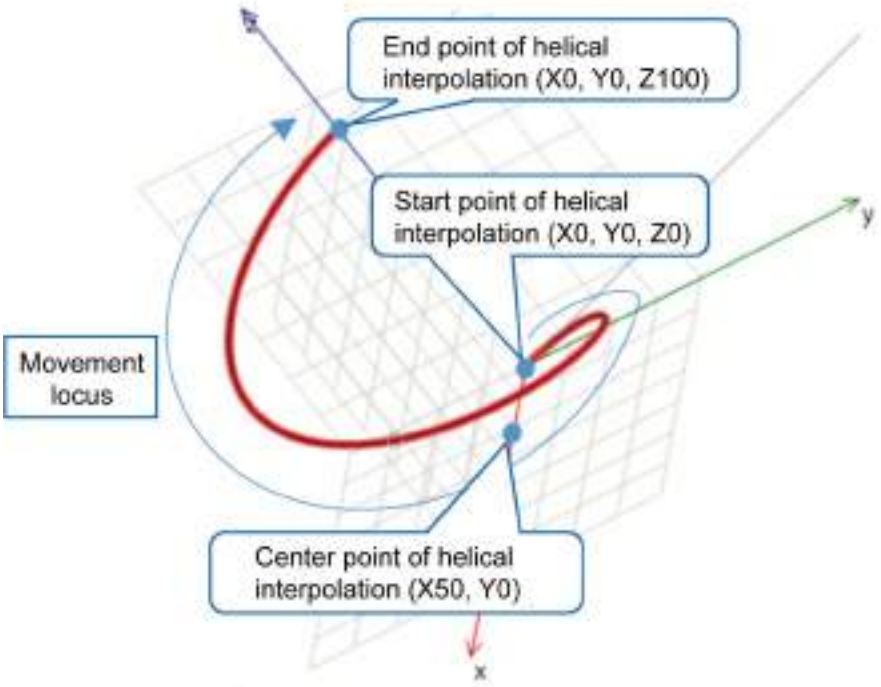
  N11: Circular interpolation is performed in the XY-plane according to the following values.

  Current value (X0, Y0, Z0), end point (X0, Y0, Z100)

  Center point (X50, Y0)

  Velocity 100

  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



Info.

- There is no need to enter the center point (K) of the Z-axis.

Absolute coordinate specification (G90)

Absolute coordinate specification is a method that specifies coordinates as absolute coordinates based on a predetermined origin (coordinate value: 0).

To use absolute coordinate specification, specify G90. Because absolute coordinate specification is the default value for target position settings for the GM1 controller, absolute coordinate specification will be used unless G91 (relative coordinate specification) is specified.

G-code	Function	Description	Remarks
G90	Absolute coordinate specification	Specifies target coordinates as absolute coordinates (If G91 is not specified, absolute coordinate specification will be used.)	After G90 is set, absolute coordinate specification remains effective until G91 is set.

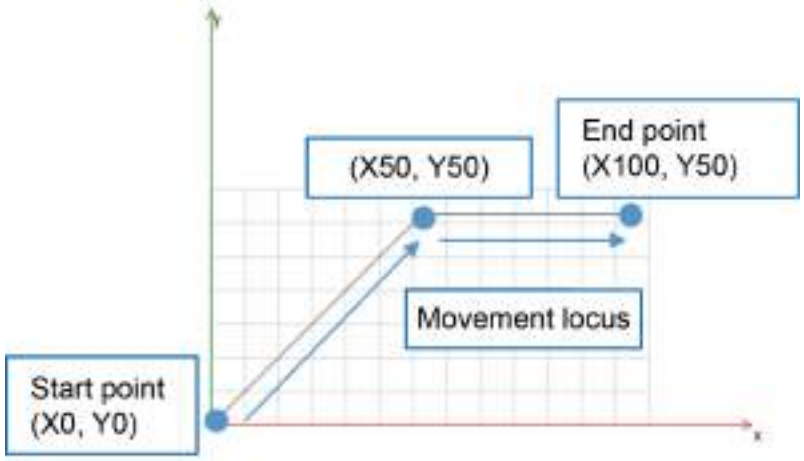
■ Absolute coordinate specification setting example (operation settings)

[Setting example]

```
G-code setting example:
N10 G90
N20 G01 X50 Y50 F100 E1 E-1
N30 G01 X100 Y50 F100 E1 E-1
```

- Explanation of G-code

N10: Absolute coordinate specification is set.
N20: Linear interpolation is performed according to the following values.
Current value (X0, Y0), end point (X50, Y50)
Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]
N30: Linear interpolation is performed according to the following values.
Current value (X50, Y50), end point (X100, Y50)
Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



Relative coordinate specification (G91)

In relative coordinate specification, the current value is used as the base and a movement distance from the current value is specified as relative coordinates.
To use relative coordinate specification, specify G91. Note that if G91 is not specified, absolute coordinate specification will be used.

G-code	Function	Description	Remarks
G91	Relative coordinate specification	Specifies target coordinates as relative coordinates	After G91 is set, relative coordinate specification remains effective until G90 is set.

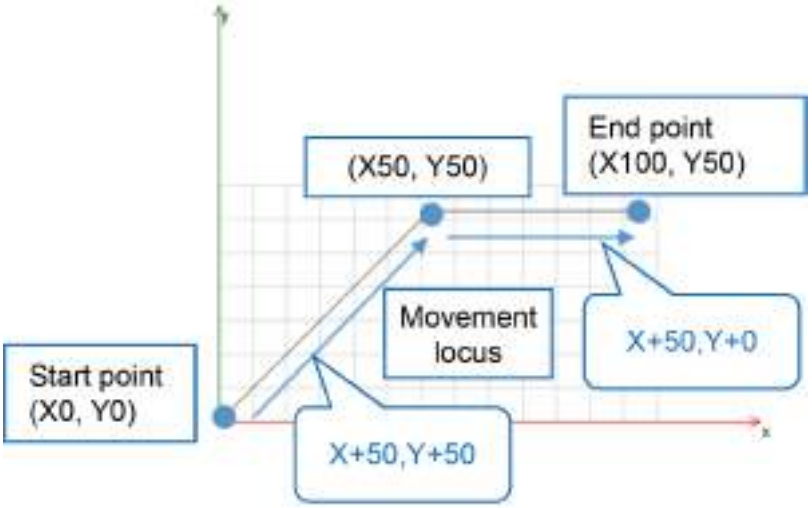
■ Relative coordinate specification setting example (operation settings)

[Setting example]

```
G-code setting example:
N10 G91
N20 G01 X50 Y50 F100 E1 E-1
N30 G01 X50 Y0 F100 E1 E-1
```

- Explanation of G-code
  N10: Relative coordinate specification is set.
  N20: Linear interpolation is performed according to the following values.
  Current value (X0, Y0), end point (X+50, Y+50)
  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]

N30: Linear interpolation is performed according to the following values.
Current value (X50, Y50), end point (X+50, Y+0)
Actual end point (X100, Y50)
Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



Center point relative / absolute coordinate specification (G98, G99)

As is the case with target coordinates, center point coordinates can also be specified by selecting relative coordinates or absolute coordinates.
Center point coordinates are specified by selecting G-code G98 or G99.

G-code	Function	Description	Remarks
G98	Absolute coordinate specification (center point)	Specifies the center point of circular interpolation as absolute coordinates	After G98 is set, absolute coordinate specification remains effective until G99 is set.
G99	Relative coordinate specification (center point)	Specifies the center point of circular interpolation as relative coordinates (If G98 is not specified, relative coordinate specification will be used.)	After G99 is set, relative coordinate specification remains effective until G98 is set

■ Notes on center point coordinate specification

- For relative and absolute coordinate specifications, different G-codes are used for target coordinates and center point coordinates.
- We recommend that relative coordinates (default setting) be used for center point coordinates, as using relative coordinates makes input easier.

■ Center point relative coordinate specification (G99)

Center points can be specified as relative coordinates, as below. Coordinates other than center point coordinates are specified as absolute coordinates.
[Setting example]

G-code setting example:
N00 G90
N01 G01 X100 Y100 F100 E1 E-1
N02 G99
N03 G17
N04 G02 X200 Y100 I50 J0 F100 E1 E-1

- Explanation of G-code

  N00: The target position is specified as absolute coordinates. (This specification can be omitted if the default setting of G90 is applied.)

  N01: Movement to X100 / Y100 coordinates (linear interpolation) is performed.

  N02: A center point is specified as relative coordinates.

  N03: An XY-plane is selected.

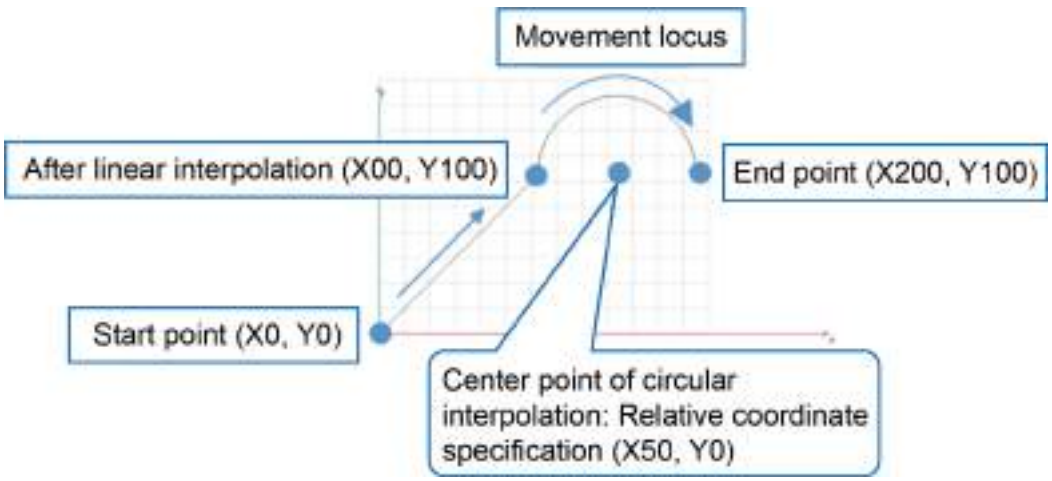
  N04: Circular interpolation is performed in the XY-plane according to the following values.

  Current value (X100, Y100), end point (X200, Y100)

  Center point entered as relative coordinates (X50, Y0)

  Velocity 100

  Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



■ Center point absolute coordinate specification (G98)

Center points can be specified as absolute coordinates, as below. Coordinates for all movements including center point coordinates are specified as absolute coordinates.

[Setting example]

G-code setting example:
N00 G90
N01 G01 X100 Y100 F100 E1 E-1N02 G98
N03 G17
N04 G02 X200 Y100 I150 J100 F100 E1 E-1

- Explanation of G-code

  N00: Absolute coordinate specification is set. (This specification can be omitted if the default setting of G90 is applied.)

  N01: Movement to X100 / Y100 coordinates (linear interpolation) is performed.

N02: The center point is specified as absolute coordinates. (This specification can be omitted if the default setting of G90 is applied.)

N03: An XY-plane is selected.

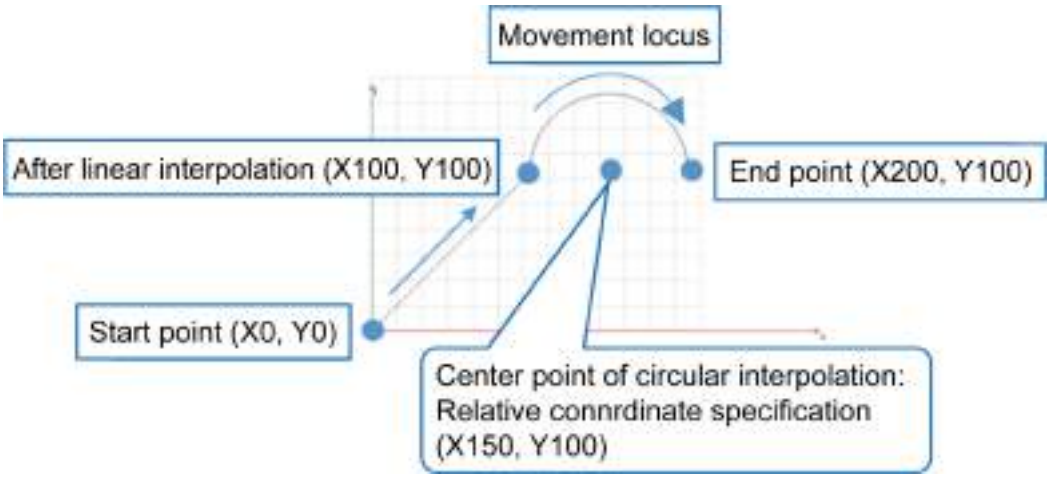
N04: Circular interpolation is performed in the XY-plane according to the following values.

Current value (X100, Y100), end point (X200, Y100)

Center point (X150, Y100)

Velocity 100

Acceleration 1 [u/sec^2], deceleration -1 [u/sec^2]



Dwell time (G04)

■ Setting rules for dwell time

Dwell time is a time to wait until next interpolation operation is executed. It is used for purposes such as waiting for a particular operation. For dwell time, enter G-code "G04" in a position where you want to set waiting time.

■ Dwell time setting example

[Setting example]

```
G-code setting example:
N00 E1 E-1
N10 G01 X100 Y100 F100
N20 G04 T1.50
N30 G01 X100 Y0 F100
```

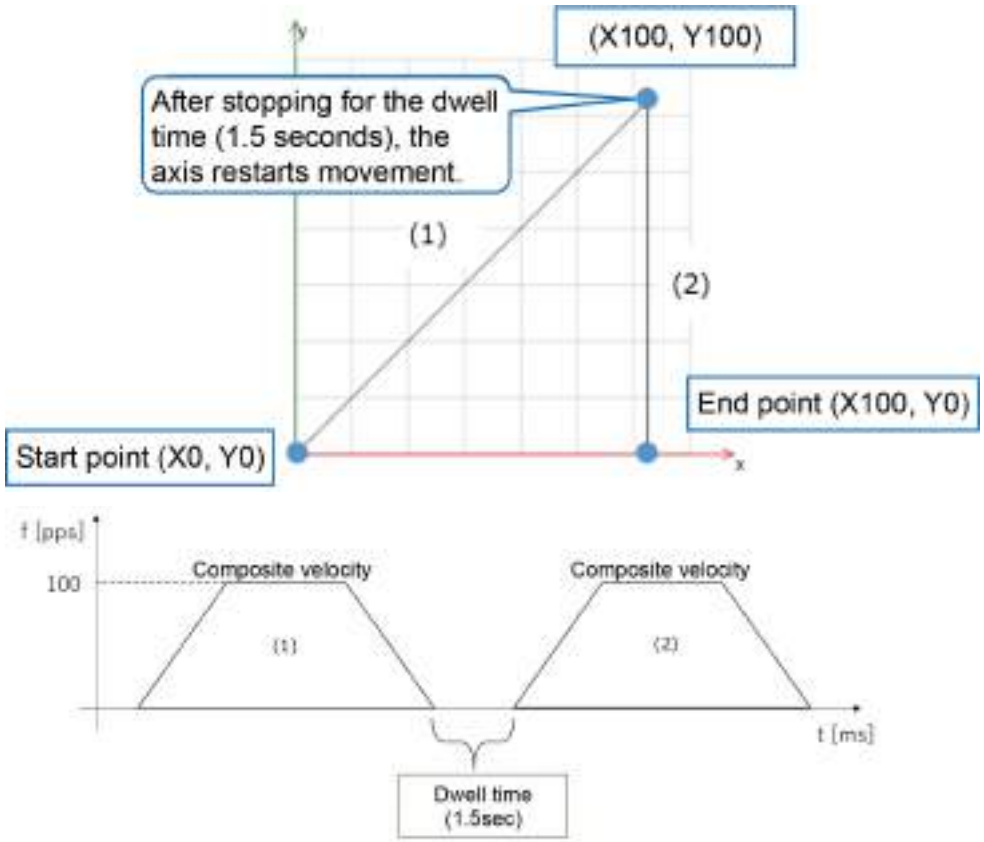
- Explanation of G-code

N00: Acceleration (1 [u/sec^2] and deceleration (-1 [u/sec^2]) are set collectively.

N10: Linear interpolation (X100, Y100) is performed. (Section (1) in the figure below)

N20: The system waits for the dwell time (1.5 seconds).

N30: Linear interpolation (X100, Y0) is performed. (Section (2) in the figure below)



11.7.8 SMC_CNC_REF and SMC_OUTQUEUE

Two compilation modes are available for CNC tables. They can be selected when a table is created.

For details on how to set compilation mode, refer to "11.7.4 Registering a CNC Table".

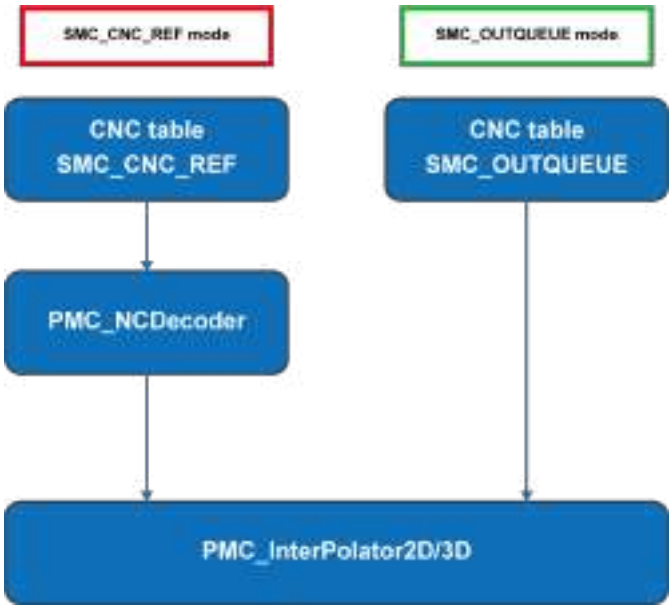
■ SMC_OutQueue (Pre-registration mode)

- Data structure is generated during compilation.
- A CNC table that has been set up can be specified directly in PMC_Interpolator2D or PMC_Interpolator3D.
- Because calculations are performed during compilation, high-speed operation can be achieved.
- Operation cannot be changed using programs.
- CNC tables cannot be joined or repeated.

■ SMC_CNC_REF

- After being decoded with PMC_NCDecoder, a CNC table that has been set up must be executed in PMC_Interpolator2D or PMC_Interpolator3D.
- Operation can be changed using programs (refer to "11.7.16 Interpolation Operation Programming: Changing Parameter Settings (Converting to Variables in CNC Table)").

- CNC tables can be joined and repeated (refer to "11.7.15 Interpolation Operation Programming: Joining and Repeating CNC Tables").
- Because calculations are performed for each execution, more processing time is required.



11.7.9 Interpolation Operation Programming: How to Create a Program for Executing Operation

■ Programming / execution procedure

To perform interpolation operation with the GM1 controller, you must configure settings, do programming, and execute the program using the following procedure.

1. CNC table and POU settings

   To perform interpolation operation using the CNC table that has been set up in Section 11.7.3, you must do programming by using function blocks in a Program Organization Unit (POU).

   You must use dedicated function blocks to perform interpolation operation. The GM1 controller supports the following three function blocks: PMC_NCDecoder, PMC_Interpolator2D, and PMC_Interpolator3D. For details on each function block, refer to "11.7.10 Interpolation Operation Programming: Explanation of Function Block (FB)".

2. Programming in POU

   Do programming in POU and register the POU in MotionTask.

   For details, refer to "7.9 Tasks".

3. Executing the program

   When the program is completed, execute "build". When the build process is completed, execute "login". After executing login, select "Login with download" in the "Options" section and then click "OK". When the GM1 controller is connected, the program is downloaded automatically. For details, refer to "7.8 Build".

   When download is completed, start executing the program.

Info.

- Before executing "build", check the confirmation items and notes in the following sections.

■ Confirmation items during programming

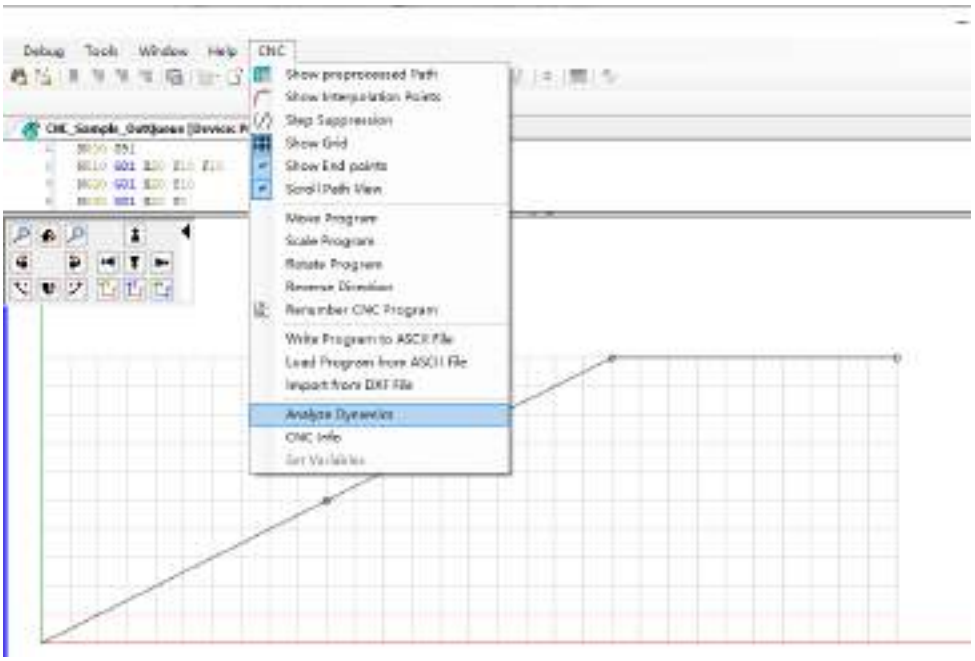
To ensure normal operation, check the following items.

- Dynamical analysis for CNC table

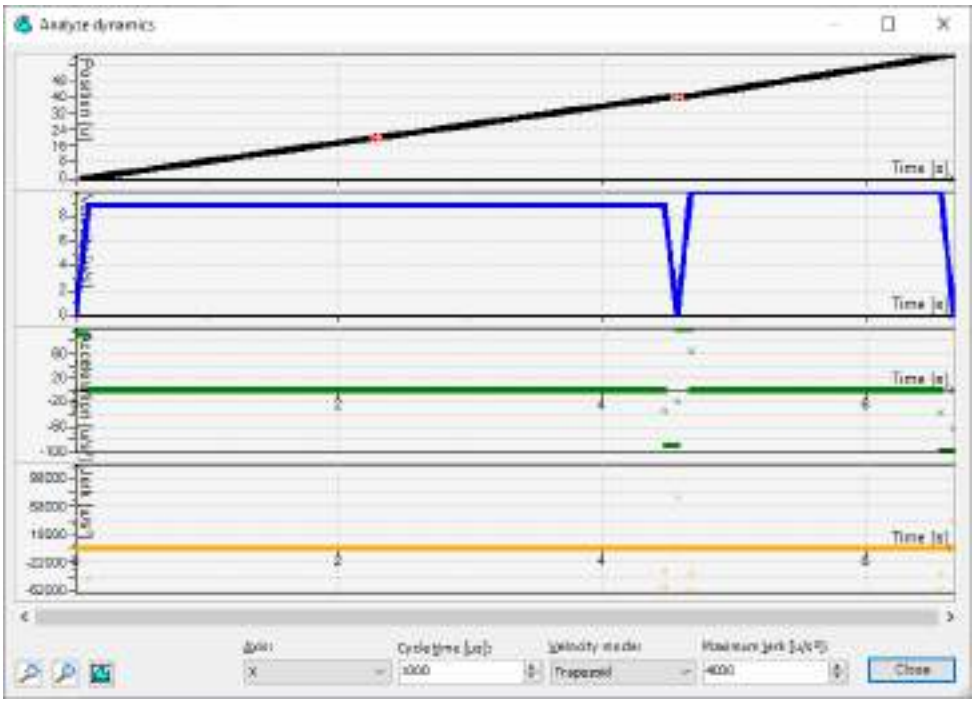
  CNC tables that have been created can be subject to dynamical analysis to preliminarily check how each axis moves.

  - Verification method for dynamical analysis

    1. In the CNC table editing window, from the menu bar, select **CNC**>**Analyze Dynamics**.



    2. The "Dynamical Analysis" window will be displayed. Specify appropriate settings in the "Axis", "Cycle time", "Velocity mode", and "Maximum jerk" fields.



Note

- If variables are used in CNC tables, the "Dynamical Analysis" window cannot be displayed. For details on how to convert to variables in CNC tables, refer to "11.7.16 Interpolation Operation Programming: Changing Parameter Settings (Converting to Variables in CNC Table)".
- If starting coordinates are specified or CNC tables are joined or repeated in POU, actual movements will differ from the results of dynamical analysis.

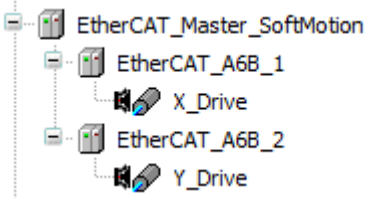
- Checking the CNC table
  1. Check that the CNC table to be used is registered in **Program Configuration**>**Application** in the Device tree.
  2. Check that the CNC table is SMC_CNC_REF type. For details, refer to "11.7.4 Registering a CNC Table".
- Checking registration of target axes

  Check that the target axes for interpolation operation have been registered beforehand.

  Axis setting example)

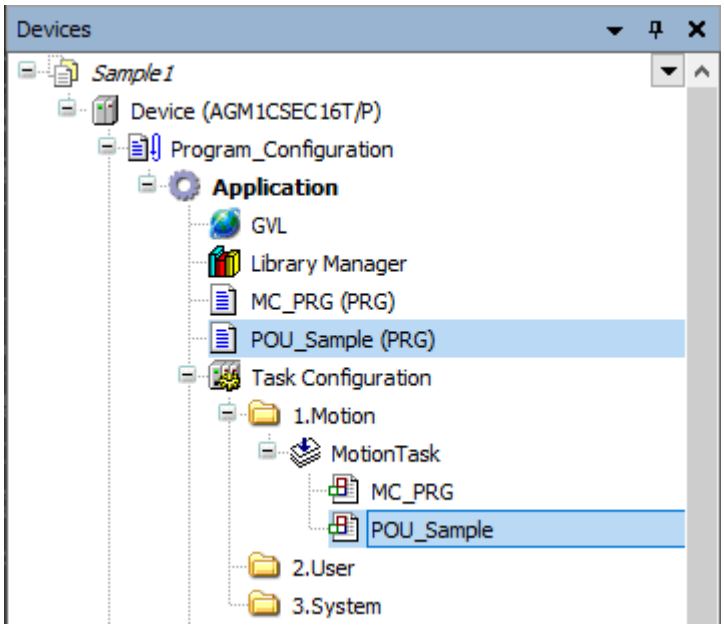
  In this example, the names of axes used in the function block are "X_Drive and Y_Drive".

  In "EtherCAT_Master_SoftMotion" in the Device tree, "X_Drive and Y_Drive" are registered in "EtherCAT_A6B_1 and EtherCAT_A6B_2", respectively. (For three axes, "Z_Drive" is also registered.)



■ Notes on POU registration

Created POU ("POU_Sample" in the example) must be registered in **Program Configuration>Application>Task Configuration>1.Motion>MotionTask** in the Device tree. Do not register it in "2.User" or "3.System".



■ Confirmation items of MotionTask cycle

Check the cycle time (control cycle) of EtherCAT_Master_SoftMotion. The cycle time (control cycle) of EtherCAT_Master_SoftMotion can be checked by double-clicking "EtherCAT_Master_SoftMotion" in the Device tree and then checking "EtherCAT parameter".

Enter the confirmed cycle time (control cycle) of EtherCAT_Master_SoftMotion into the "dwIpoTime" input section in the "PMC_Interpolator2D/3D" function block.

11.7.10 Interpolation Operation Programming: Explanation of Function Block (FB)

■ Types of function block

This manual supports the following function blocks.

Name	Function	Overview
PMC_NCDecoder	Decode	This function block decodes the SMC_CNC_REF type CNC table into an SMC_OUTQUEUE type CNC table.
PMC_Interpolator2D	2-axis interpolation operation	This function block executes 2-axis interpolation operation according to the specified SMC_OUTQUEUE type CNC table.
PMC_Interpolator3D	3-axis interpolation operation	This function block executes 3-axis interpolation operation according to the specified SMC_OUTQUEUE type CNC table.

(Note 1) Do not use the standard function blocks supported by CODESYS.

■ PMC_NCDecoder (decode)

This function block decodes the SMC_CNC_REF type CNC table into an SMC_OUTQUEUE type CNC table.

For the specifications of PMC_NCDecoder, refer to the *GM1 Series Reference Manual (Instruction Edition)*.

For the programming method, refer to "PMC_Interpolator2D (2-axis interpolation)".

■ PMC_Interpolator2D (2-axis interpolation)

This function block (FB) performs 2-axis interpolation control according to the specified CNC table.

For the detailed specifications of the function block, refer to the *GM1 Series Reference Manual (Instruction Edition)*.

Programming method for PMC_Interpolator2D

[Setting example 1] ST program example: SMC_CNC_REF type CNC table

```
G-code (SMC_CNC_REF): CNC_Table1
N01   G01 X10 Y10 F10 E1 E-1
```

```
Variable declaration section:
//FB instances
NCDecoder              : PMC_NCDecoder;
Interpolator2D         : PMC_ Interpolator2D;
ReadSetPosition        : SMC_ReadSetPosition          //Read current value
//Variables
buf                    : ARRAY[0..10] OF SMC_GEOINFO; //Buffer for decoding
fSetXPosition          : LREAL;                       //Current value on X-axis
fSetYPosition          : LREAL;                       //Current value on Y-axis
bStart                 : BOOL;                        //Execution flag
```

- Explanation of declaration section
  - In Lines 2 to 4, function blocks to be used are defined by assigning arbitrary names to them.
  - In Line 5, a buffer to be used for PMC_NCDecoder is defined.

    Paths in the CNC table that are as many as arrays can be stored. (In the above example, 11 paths can be stored.)

    * Note that if the buffer size is 5 or less, PMC_NCDecoder cannot be executed.
  - In Line 6, a flag for starting execution is defined.

```
Control section:
IF bStart THEN
        ReadSetPosition(Axisx:=X_Drive,Enable:=TRUE, Position=>fSetXPosition)
        ReadSetPosition(Axisy:=Y_Drive,Enable:=TRUE, Position=>fSetYPosition)
        NCDecoder(nSizeOutQueue := SIZEOF(buf),pbyBufferOutQueue := ADR(buf),
                dXStartPosition:=fSetXPosition,dYStartPosition:=fSetYPosition,
                ncprog :=CNC_Table1,bExecute:=TRUE);
```

```
        Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=TRUE,
                poqDataIn:=NCDecoder.poqDataOut,dwIpoTime:=1000,);
END_IF
```

- Explanation of control section
  - When the "bStart" flag is set to TRUE, NCDecoder and Interpolator2D start being executed.
  - PMC_NCDecoder and PMC_Interpolator2D must be executed at the same time.
  - For dXStartPosition and dYStartPosition in PMC_NCDecoder, specify the respective positions on the X-axis and Y-axis when interpolation control starts being executed. (For details, refer to "11.7.11 Interpolation Operation Programming: Specifying the Starting Coordinates".)

[Setting example 2] ST program example: SMC_OUTQUEUE type CNC table

```
G-code (SMC_OUTQUEUE): CNC_Table2
N01   G01 X10 Y10 F10 E1 E-1
```

```
Variable declaration section:
//FB instances
Interpolator2D          : PMC_ Interpolator2D;
//Variables
bStart                  : BOOL;                         //Execution flag
```

- Explanation of declaration section
  - In Line 2, a function block to be used is defined by assigning an arbitrary name to it.
  - In Line 4, a flag for starting execution is defined.

```
Control section:
IF bStart THEN
        Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=TRUE,
                poqDataIn:=ADR(CNC_Table2),dwIpoTime:=1000,);
END_IF
```

- Explanation of control section
  - When the "bStart" flag is set to TRUE, Interpolator2D starts being executed.
  - For PMC_Interpolator2D, specify the address of the SMC_OUTQUEUE that has been created.
  - In PMC_Interpolator2D, match the position at the start of operation and the current value.
    (For details, refer to "11.7.11 Interpolation Operation Programming: Specifying the Starting Coordinates".)

■ PMC_Interpolator3D (3-axis interpolation)

This function block (FB) performs 3-axis interpolation control according to the specified CNC table.

For the detailed specifications of the function block, refer to the *GM1 Series Reference Manual (Instruction Edition)*.

- When executing multiple interpolation controls, do not use the same buffer for different instances.
- When executing multiple interpolation controls at the same time, do not use the same table for different instances. Otherwise, normal operation may not occur. (If the same table is used for different instances, create multiple tables with the same contents.)
- Do not execute single-axis operation function blocks such as MC_MoveRelative, MC_Stop, and MC_Halt during interpolation control.
- Do not change any variables within the SMC_CNC_REF or SMC_OUTQUEUE structure in POU.

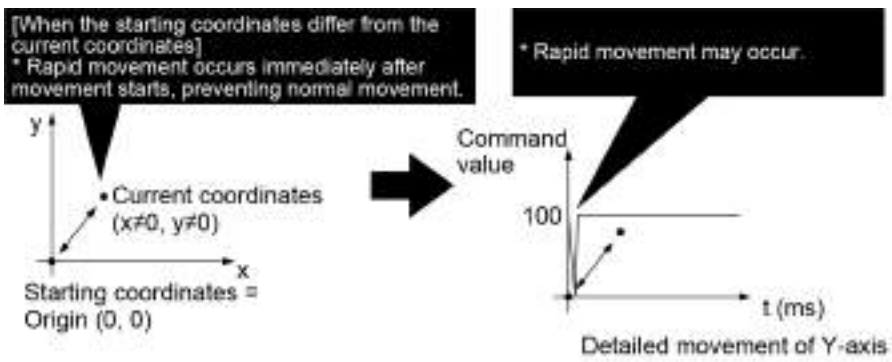
Info.

- When PMC_NCDecoder and PMC_Interpolator2D or PMC_Interpolator3D are executed at the same time, the contents of the buffer are rewritten according to the operation execution. Therefore, even if the buffer size is not large enough to store the number of paths in the CNC table, normal operation will occur.
- If PMC_Interpolator2D or PMC_Interpolator3D is executed after the processing of PMC_NCDecoder is complete, define a buffer size that can store all paths in the CNC table.

11.7.11 Interpolation Operation Programming: Specifying the Starting Coordinates

In CNC tables programmed in G-code, by default, the starting coordinates are defined as the origin (0, 0). Normally, if the starting coordinates in the CNC table are identical with the operation starting coordinates (current coordinates), normal operation will occur.

If the operation starting position (current coordinates) is not the origin, when operation is started from the current coordinates, rapid movements may occur from the current coordinates through to the origin immediately after startup, thereby preventing normal operation from being performed.



To achieve normal operation, therefore, if the starting coordinates in the CNC table differ from the operation starting coordinates (current coordinates), they must be matched. In such a case, specify starting coordinates.

The method for specifying starting coordinates differs between SMC_CNC_REF and SMC_OUTQUEUE.

* In the following descriptions, the starting coordinates in the CNC table are referred to as "starting coordinates" and the coordinates at the start of interpolation control operation are referred to as "operation starting coordinates".

■ For SMC_CNC_REF

Starting coordinates can be specified in PMC_NCDecoder when decoding is executed.

```
Variable declaration section:
//FB instances
NCDecoder              : PMC_NCDecoder;
Interpolator2D         : PMC_ Interpolator2D;
ReadSetPosition        : SMC_ReadSetPosition          //Read current value
//Variables
buf                    : ARRAY[0..10] OF SMC_GEOINFO; //Buffer for decoding
fSetXPosition          : LREAL;                       //Current value on X-axis
fSetYPosition          : LREAL;                       //Current value on Y-axis
bStart                 : BOOL;                        //Execution flag
```

```
Control section:
IF bStart THEN
        ReadSetPosition(Axisx:=X_Drive,Enable:=TRUE, Position=>fSetXPosition)
        ReadSetPosition(Axisy:=Y_Drive,Enable:=TRUE, Position=>fSetYPosition)
        NCDecoder(nSizeOutQueue := SIZEOF(buf),pbyBufferOutQueue :=
        ADR(buf),dXStartPosition:=fSetXPosition,dYStartPositon:=fSetYPosition,
                ncprog := CNC_Table1,bExecute:=TRUE);
        Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=TRUE,
                poqDataIn:=NCDecoder.poqDataOut,dwIpoTime:=1000);
END_IF
```

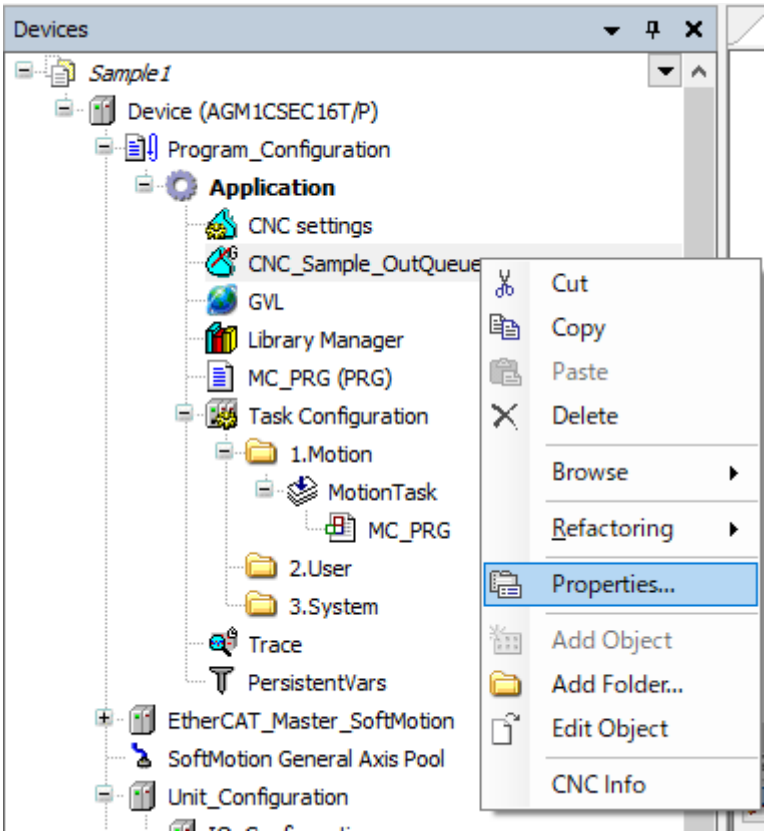
- When operation is started, SMC_ReadSetPosition reads the current coordinates and PMC_NCDecoder specifies the coordinates as starting coordinates.

Note

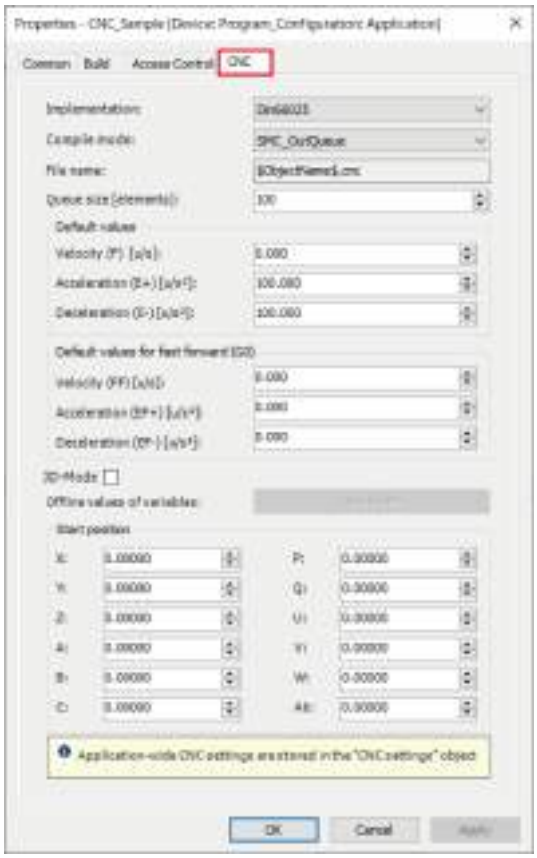
- If the specified starting coordinates differ from the operation starting coordinates, there is a risk that rapid movement may occur.

■ For SMC_OUTQUEUE

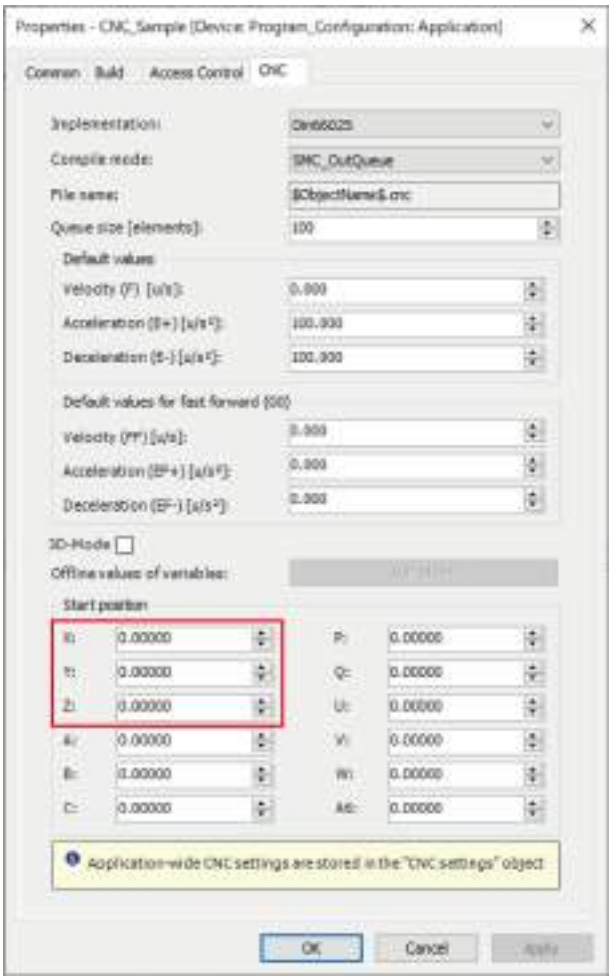
1. Right-click a CNC table that has been created and select "Properties" from the context-sensitive menu that is displayed.



2. Click the "CNC" tab in the upper section of the Properties window that is displayed.



3. In the "Start position" section, specify the respective operation starting coordinates for the X, Y, and Z axes and then click [OK].



Start interpolation control operation using the starting coordinates specified in the window.

Note

- If the specified starting coordinates differ from the operation starting coordinates, there is a risk that rapid movement may occur.

11.7.12 Interpolation Operation Programming: P-point Control and C-point Control

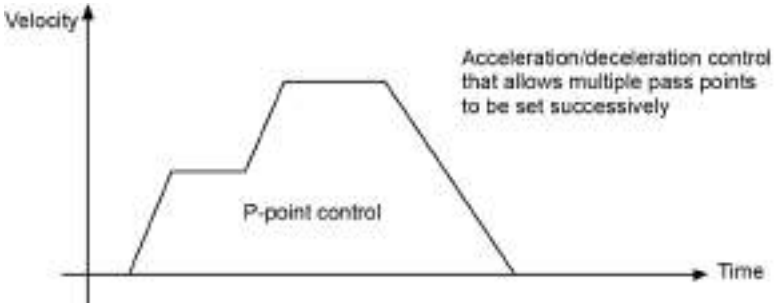
When composite interpolation operations are executed by the GM1 controller, continuity between interpolated movements becomes an issue.

The GM1 controller supports C-point control from the following two types of control.

■ P-point control

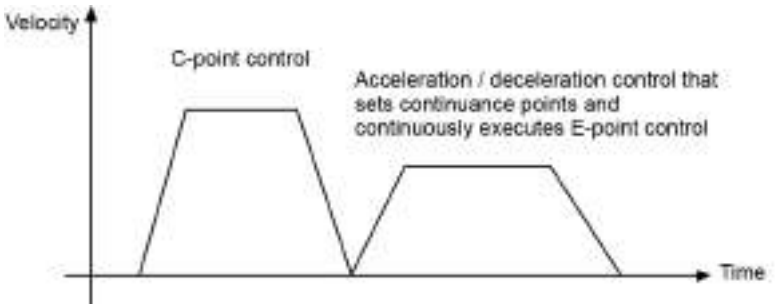
P-point control refers to control passing through a "Pass Point". In this manual, this control is referred to as "P-point control" for the sake of convenience.

This method is used when target multi-stage velocities are specified in a sequence of motions.



■ C-point control

C-point control refers to control passing through a "Continuance Point". In this manual, this control is referred to as "C-point control" for the sake of convenience. This method is used to execute consecutive E-point controls by one-time startup.



Info.

- When one movement is completed, it stops and then next movement is started.
- This also applies to movements when dwell time is set. (If dwell time is set to 0 ms, continuous movements will occur without waiting time.)

For C-point control, the GM1 controller supports the use case patterns shown in the table below.

C-point control for connection between paths

This refers to connection between movements within a single CNC table.

C-point control for connection between CNC tables

This refers to connection between movements across CNC tables.

Use case	Applicable function block	Restriction	Remarks
C-point control for connection between paths	PMC_Interpolator2D/3D	You cannot change movements only in a particular path connection section. (Changes must be unified within C-point control.)	Specify relevant parameters for PMC_Interpolator2D or PMC_Interpolator3D. There is no need to create a new use case. Refer to "11.7.14 Interpolation Operation Programming: Settings in POU for P-point Control and C-point Control".

Use case	Applicable function block	Restriction	Remarks
C-point control for connection between CNC tables	PMC_NCDecoder		For details, refer to "11.7.15 Interpolation Operation Programming: Joining and Repeating CNC Tables".

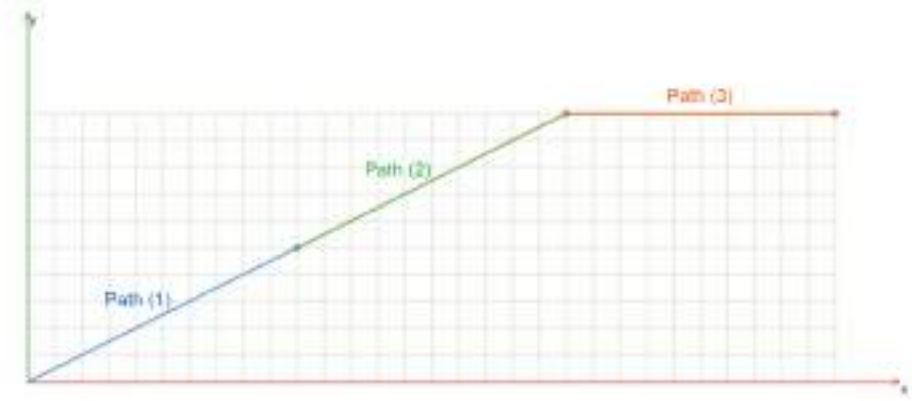
11.7.13 Interpolation Operation Programming: Settings in CNC Table for C-point Control

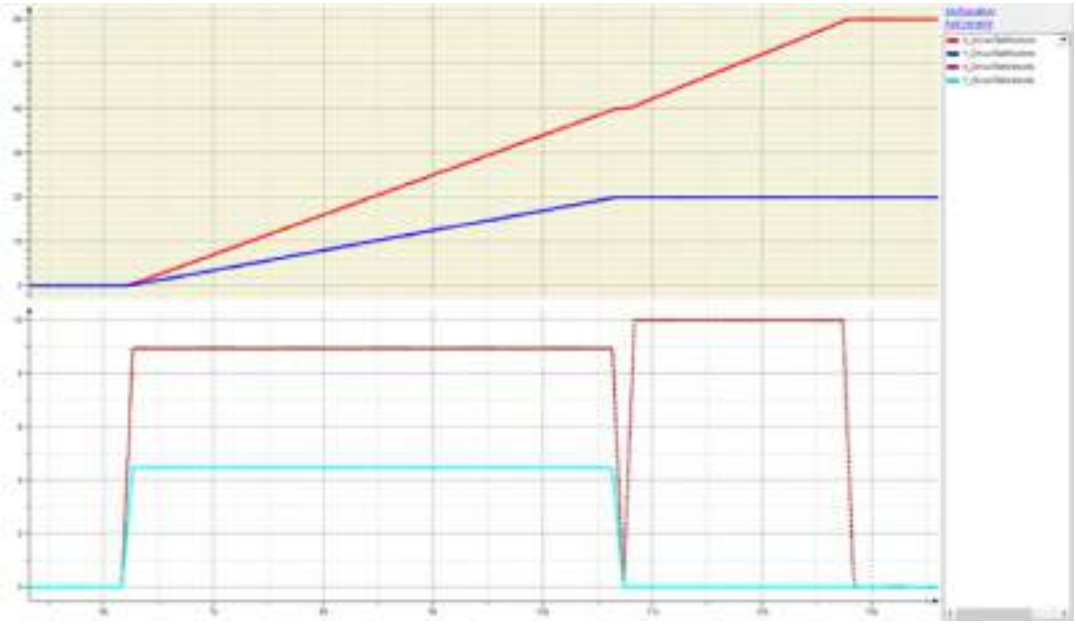
The GM1 controller basically uses C-point control to perform operation and uses P-point control only when paths are connected linearly.

■ Operation example

[Setting example] C-point control

```
N000 G91
N010 G01 X20 Y10 F10//Path (1)
N020 G01 X20 Y10//Path (2)
N030 G01 X20 Y0//Path (3)
```





- P-point control is used when paths are connected linearly (with no angle), as shown in the connection between Path (1) and Path (2).
- C-point control is used when paths are connected with an angle, as shown in the connection between Path (2) and Path (3).

* Even if paths are connected linearly, if G-code is used to set the dwell time between paths to 0, the respective composite velocities of the X-axis and Y-axis will converge to zero, causing the interpolation control to switch to C-point control.

[Setting example]

```
N00 G91
N10 G01 X20 Y10 F10
N15 G04 T0
N20 G01 X20 Y10
N30 G01 X20 Y10
```

11.7.14 Interpolation Operation Programming: Settings in POU for P-point Control and C-point Control

■ Settings for connection between each path

By setting the bSingleStep parameter in PMC_Interpolator2D or PMC_Interpolator3D, connection between each path can be achieved by C-point control.

[Setting example]

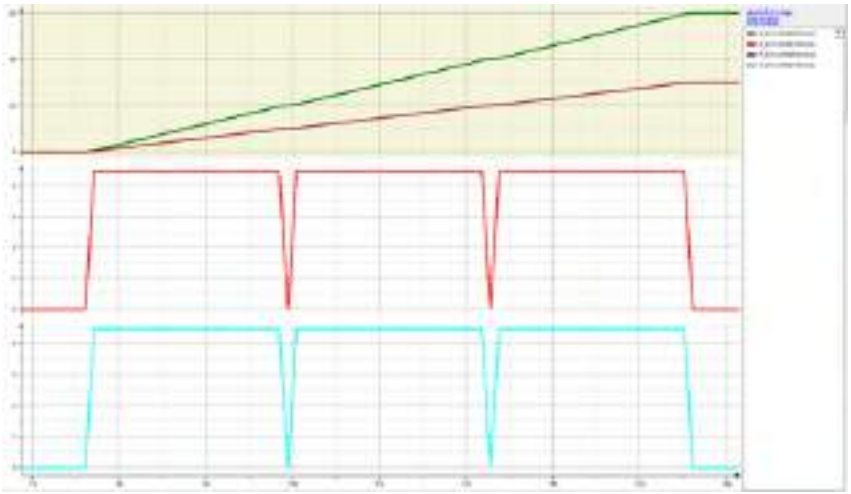
G-code is set so that connection between paths is achieved by P-point control (linear connection).

```
N00 G91
N10 G01 X20 Y10 F5
N20 G01 X20 Y10
```

```
N30 G01 X20 Y10
```

In PMC_Interpolator2D, bSingleStep is set to TRUE.

```
Control section:
...
Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=TRUE,
poqDataIn:=,dwIpoTime:=1000, bSingleStep:=TRUE);
...
```



Connection between each path is achieved by C-point control.

11.7.15 Interpolation Operation Programming: Joining and Repeating CNC Tables

■ Joining CNC tables

- PMC_NCDecoder

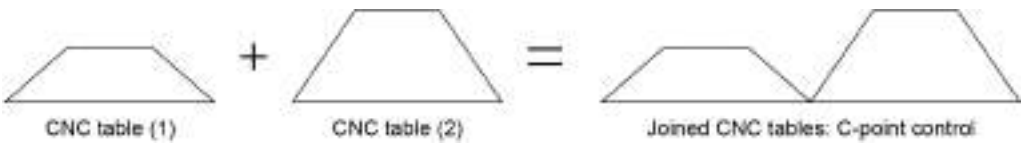
  CNC tables can be joined by setting the "bAppend" join flag in PMC_NCDecoder to TRUE and decoding each table to be joined in order.

  Movements between each CNC table are performed by C-point control.

  SMC_OUTQUEUE which is output is executed as a CNC table in PMC_Interpolator2D or PMC_Interpolator3D.

For joining CNC tables, the GM1 controller supports the following use case patterns.

Joining CNC tables: C-point control



[Setting example 1] ST program example: Joining CNC tables using PMC_NCDecoder

```
G-code: CNC_Table1
N01    G01 X10 Y10 F10 E100 E-100
```

```
G-code: CNC_Table2
N01   G01 X20 Y20 F10 E100 E-100
```

```
Variable declaration section:
//FB instances
NCDecoder                    : PMC_NCDecoder;
Interpolator2D               : PMC_ Interpolator2D;
//Variables
buf: ARRAY[0..10] OF SMC_GEOINFO;                    //Buffer for decoding
iSequence:                   : INT;                  //Execution control No.
```

```
Control section:
Case iSequence OF
    1:
        NCDecoder(nSizeOutQueue := SIZEOF(buf),pbyBufferOutQueue := ADR(buf),
            ncprog := CNC_Table1,bExecute:= TRUE,   bAppend:=TRUE);
        Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=TRUE,
            poqDataIn:=NCDecoder.poqDataOut,dwIpoTime:=1000,);
        IF Interpolator2D.bDone THEN
            iSequence:=2;
        END_IF
    2:
        NCDecoder(nSizeOutQueue := SIZEOF(buf), pbyBufferOutQueue := ADR(buf),
            ncprog := CNC_Table1, bExecute:= FALSE, bAppend:=TRUE);
        Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=FALSE,
            poqDataIn:=NCDecoder.poqDataOut,dwIpoTime:=1000,);
        iSequence := 3;
    3:
        NCDecoder(nSizeOutQueue := SIZEOF(buf), pbyBufferOutQueue := ADR(buf),
            ncprog := CNC_Table2, bExecute:= TRUE, bAppend:=TRUE);
        Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=TRUE,
            poqDataIn:=NCDecoder.poqDataOut,dwIpoTime:=1000,);
END_CASE
```

- The first CNC table is specified in PMC_NCDecoder, which is then executed together with PMC_Interpolator2D.
- After the first instance of PMC_NCDecoder is completed, the second CNC table is specified in PMC_NCDecoder, which is then executed again.

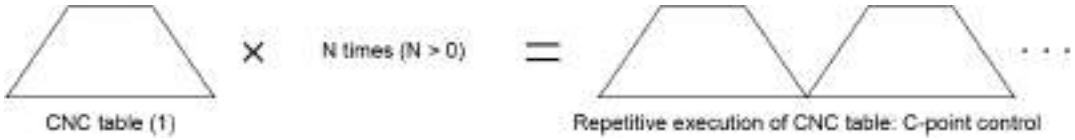
■ Repeating CNC tables

- PMC_NCDecoder

  Movements between each CNC table repeated are performed by C-point control.

SMC_OUTQUEUE which is output is executed as a CNC table in PMC_Interpolator2D or PMC_Interpolator3D.

Joining CNC tables: C-point control [PMC_Interpolator 2D(3D)]



[Setting example 1] ST program example: PMC_NCDecoder

```
G-code: CNC_Table
N01   G01 X10 Y0 F10 E100 E-100
N02   G01 X10 Y10
N03   G01 X0 Y0
```

```
Variable declaration section:
//FB instances
NCDecoder                    : PMC_NCDecoder;
Interpolator2D               : PMC_ Interpolator2D;
//Variables
buf: ARRAY[0..6] OF SMC_GEOINFO;                    //Buffer for decoding
iSequence:                   : INT;                 //Execution control No.
bDec                         : BOOL;                //Decode flag
bStart                       : BOOL;                //Interpolation control execution flag
iCount                       : INT:=0;              //Specified number of repetitions
```

```
Control section:
NCDecoder(nSizeOutQueue := SIZEOF(buf),pbyBufferOutQueue := ADR(buf),
                    ncprog := CNC_Table,bExecute:= bDec,   bAppend:=TRUE);
Interpolator2D(Axisx:=X_Drive,Axisy:=Y_Drive,bExecute:=bStart,
                    poqDataIn:=NCDecoder.poqDataOut,dwIpoTime:=1000,);
Case iSequence THEN
        1:          bDec:=TRUE;
                    bStart:=TRUE;
                    IF NCDecoder.bDone THEN
                                iCount := iCount + 1;
                                IF iCount < 3 THEN
                                            bDec:=FALSE;
                                END_IF
                    END_IF
END_CASE
```

- NCDecoder is executed the specified number of repetitions. (In the above example, NCDecoder is repeated three times.)

Info.

- If NCDecoder is repeated, set the buffer size to 7 or more. If the buffer size is less than 7, normal operation may not occur.
- If NCDecoder is repeated, do not set a buffer size that is too large for the number of paths in the CNC table (for one execution). If a too large buffer size is specified, normal operation may not occur.

11.7.16 Interpolation Operation Programming: Changing Parameter Settings (Converting to Variables in CNC Table)

This section explains how to edit G-codes in CNC tables to convert arbitrary parameter settings (such as target values) to variables and change them to arbitrary target values.

■ Notes on changing parameter settings

The following method is used to change parameters.

Variable reference type:

Parameter variables are set beforehand and used by reading them when necessary.

Note

- Use the SMC_CNC_REF compilation type.
- Change variables before starting PMC_NCDecoder.

■ Parameters such as target values

If you want to make parameter changes supported by the GM1 controller, the following parameters are applicable.

Applicable parameter	Variable name (global variable)	Variable name (specific for CNC tables)
X-axis	g_x	g_x
Y-axis	g_y	g_y
Z-axis	g_z	g_z
Velocity	g_f	g_f
Acceleration	g_accel	g_accel
Deceleration	g_decel	g_decel

(Note 1) If you want to make parameter changes using variables, you must declare parameter variables to be used as global variables.

(Note 2) If you want to use variables in CNC tables, enclose each variable parameter with $.

■ Changing parameters such as target values

The following is a programming example of variables for parameters to be changed.

- Define variables for parameters to be changed as global variables.
- Each variable of parameters to be changed in a CNC table must be enclosed with $.

CNC table example:

```
N01 G01 X$ g_x $    Y$ g_y $ Z$ g_z $ F$ g_f $ E$ g_accel $ E$ g_decel $
```

Explanation of G-code

N01

Values to be set are specified as variables.

[Setting example 1] ST program

```
Example of global variable declaration section:
VAR_GLOBAL
                g_x : REAL := 100;
                g_y : REAL := 100;
                g_f : REAL := 100;
                g_accel : REAL := 100;
                g_decel : REAL := -100;
END_VAR
```

Explanation of program:

- All variables are defined as global variables. All variables are the REAL data type.
- Values to be substituted for variables can be specified as either variable values or constants.

■ Changing parameters during operation (using G75)

The variable reference type allows parameters to be changed during operation.

G75 can synchronize timing with PMC_Interpolator2D or PMC_Interpolator3D.

Using this function makes it possible to perform the following operations.

- Parameters can be changed while the variable reference type of the parameter change function is being executed. (However, only parameter changes where G75 is specified and subsequent changes are applied.)
- During the execution of a CNC table, the contents of variables are reacquired when the line where G75 is specified is executed. The contents of variables that are reacquired are reflected in the next line. In this processing, re-decoding is not required.

```
CNC table example:
N01 G01 X$ g_x $    Y$ g_y $ Z$ g_z $ F$ g_f $ E$ g_accel $ E$ g_decel $
N02 G75
N03 G01 X$ g_x $    Y$ g_y $ Z$ g_z $ F$ g_f $ E$ g_accel $ E$ g_decel $
```

Explanation of G-code

N01

Values to be set are specified as variables.

N02

G75 is specified to synchronize timing with PMC_Interpolator2D or PMC_Interpolator3D.

N03

Values to be set are specified as variables. The contents of all parameters are also reacquired and updated.

11.8 Motion Function Errors

11.8.1 Overview of Motion Function Errors

Motion function errors can be classified as below.

■ Function block (FB) errors

These errors occur when motion function blocks are executed. They are defined as SMC_ERROR in CODESYS.

Errors can be classified as below.

- Errors resulting in errorstop

  In CODESYS, if an error that is judged to be operation non-continuable occurs, the axis state will be set to "errorstop" and control for the relevant axes will stop.

  In a function block that is being executed, the "CommandAborted" flag is set to TRUE, causing the function block to terminate.

  The "errorstop" state is released by executing the MC_Reset function block.

- Errors not resulting in errorstop

  In CODESYS, if an error that is judged to be operation continuable (an error that can be cleared by re-executing the function block, such as a parameter error to the function block) occurs, the "Error" flag of the function block will be set to TRUE and an error code will be set in ErrorID. The error is cleared by setting the "Execute" flag in the function block to FALSE, enabling the function block to be re-executed.

■ DriveInterfaceError

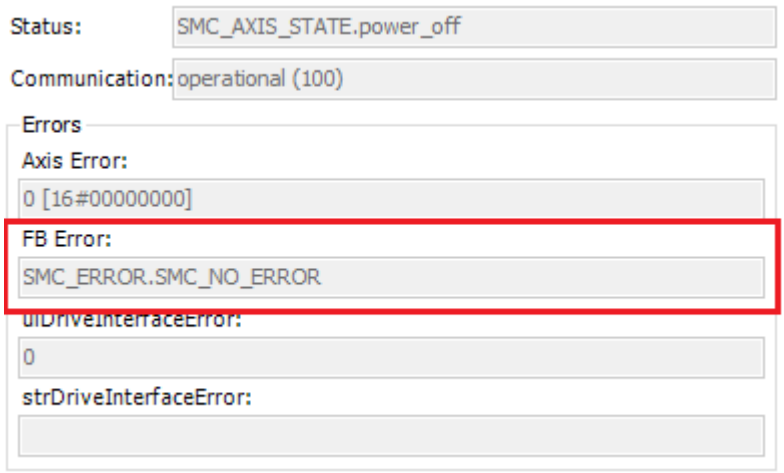
Errors that occur in SM_Drive_ETC_Panasonic_MINAS_A5B are output as DriveInterfaceError. DriveInterfaceError is an internal error.

■ Amplifier alarm

Alarms and warnings occur in servo amplifiers.

11.8.2 Error Check Method

■ Online monitor in the axis setting window for SM_Drive_ETC_Panasonic_MINAS_A5B



The oldest error is displayed in the "FB error" field. This is the same error result as that output by the MC_ReadFBError function block.

* Some FB errors are not displayed (SMC_RP_REQUESTING_ERROR, etc.).

■ Device log

All errors that occurred in the past are displayed. This is regardless of whether "errorstop" occurred.

* Some FB errors are not displayed (SMC_RP_REQUESTING_ERROR, etc.).

■ Errors that can be output to programs

Return value to function block	When an FB error occurs, the "Error" flag is set to TRUE and the error ID of SMC_ERROR is output to ErrorID.
MC_ReadFBError	FB errors that occurred are displayed, regardless of "errorstop". Some FB errors are not displayed (SMC_RP_REQUESTING_ERROR, etc.).

Amplifier alarm	ETC_CO_SdoRead FB or PDO mapping "Error code"

11.8.3 Clearing Errors

When errorstop occurs	Execute MC_Reset.
When an error returns to a function block	Execute the same function block with the "Execute" flag set to FALSE.
When an amplifier alarm occurs	Execute SMC3_ReinitDrive.

12 Unit Control

This chapter explains unit control for the GM1 controller.

12.1 Overview of Unit Control 12-2

12.2 IO Parameters for Unit Control 12-3

12.3 I/O Mapping for Unit Control 12-4

12.4 General-purpose I/O 12-5
- 12.4.1 Overview of General-purpose I/O Function 12-5
- 12.4.2 Setting Parameters with GM Programmer 12-5
- 12.4.3 Setting Items of IO_Configuration Parameters 12-6
- 12.4.4 I/O Mapping for General-purpose I/O 12-7

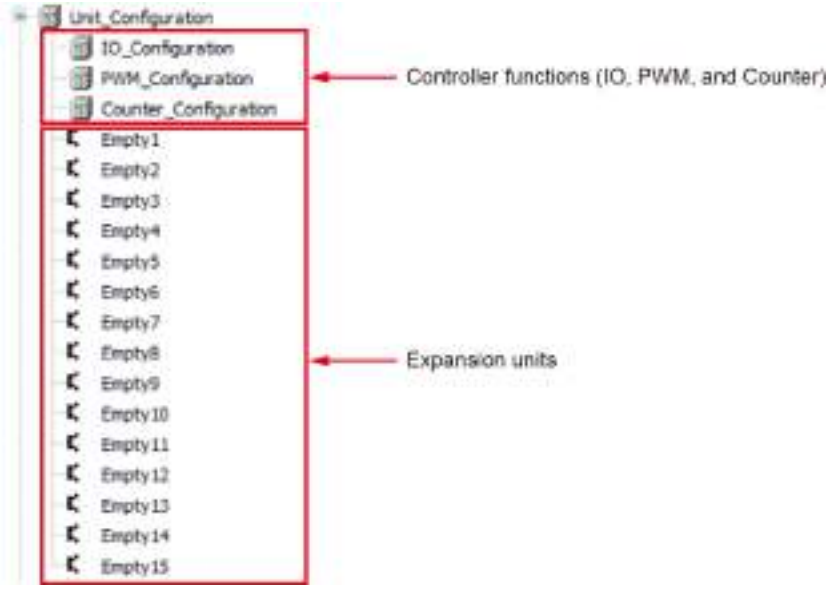
12.5 PWM Output 12-9
- 12.5.1 Overview of PWM Output 12-9
- 12.5.2 Setting Output Ports with GM Programmer 12-9
- 12.5.3 I/O Mapping for PWM Output 12-10
- 12.5.4 Data Update Timing (Output Frequency) 12-11
- 12.5.5 Data Update Timing (Duty Ratio) 12-12
- 12.5.6 PWM Output Setting Example 12-12

12.6 High-speed Counter Function 12-18
- 12.6.1 Overview of High-speed Counter Function 12-18
- 12.6.2 Setting Parameters with GM Programmer 12-19
- 12.6.3 Counter Parameter Setting Items 12-22
- 12.6.4 I/O Mapping for High-speed Counter Output 12-25
- 12.6.5 Operation Ready Request 12-28
- 12.6.6 Count Function 12-31
- 12.6.7 Comparison Function 12-41
- 12.6.8 External Output Function 12-47
- 12.6.9 Capture Function 12-48
- 12.6.10 Unit Error 12-60

12.7 Settings of I/O Unit 12-62
- 12.7.1 Parameter Settings 12-62
- 12.7.2 I/O Mapping for I/O Unit 12-62

12.1 Overview of Unit Control

Unit control provides control for the controller functions (I/O, PWM, and counter) and expansion units.



12.2 IO Parameters for Unit Control

The parameter window is used to set up parameters for the controller functions (I/O, PWM, and counter) and each expansion unit.

This section explains I/O mapping, using IO_Configuration as an example.

In the navigator pane, double-click the "IO_Configuration" object. The "IO_Configuration" pane will be displayed.

In the "IO_Configuration" pane, click the "IO Parameters" tab. The parameter settings sub-pane will be displayed. Change the settings according to your need.

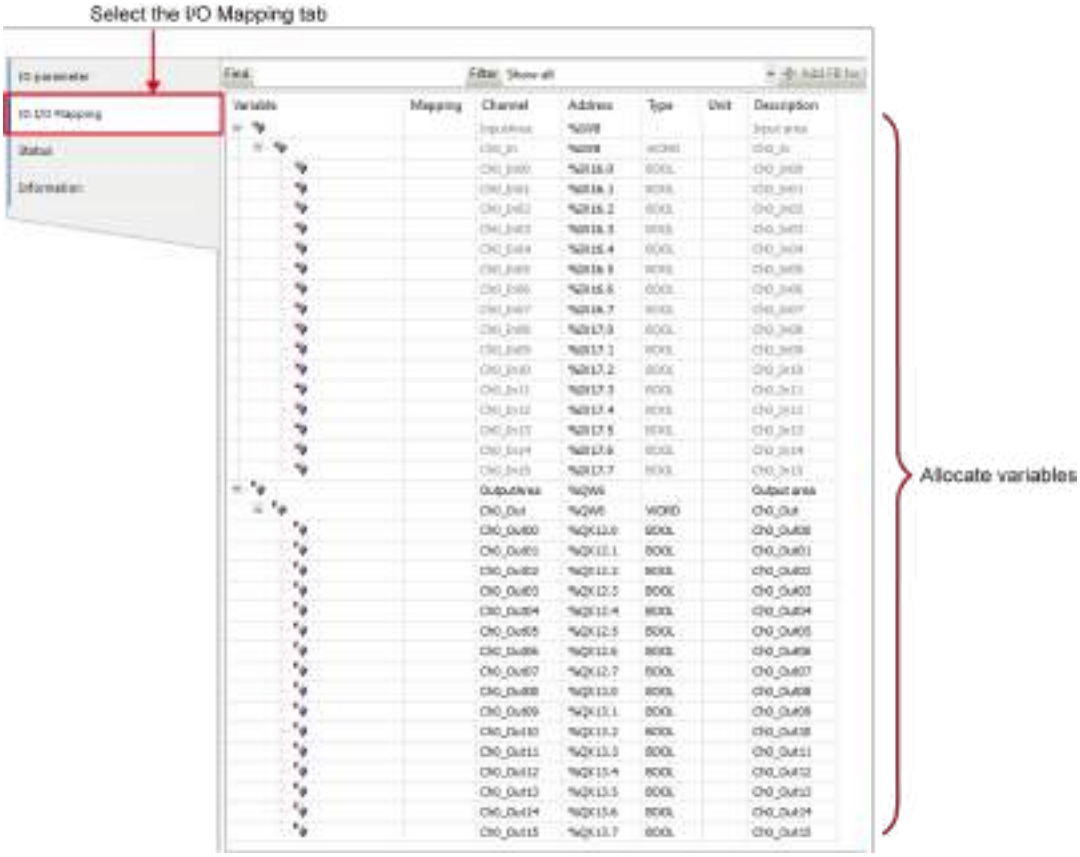
12.3 I/O Mapping for Unit Control

The I/O mapping window is used to allocate variables to I/O mapping for the controller functions (I/O, PWM, and counter) and each expansion unit.

This section explains I/O mapping, using IO_Configuration as an example.

Click the "I/O Mapping" tab in the Device pane. The I/O mapping pane will be displayed.

Register variables directly from the variable or I/O mapping window declared in the Application object.



12.4 General-purpose I/O

12.4.1 Overview of General-purpose I/O Function

The general-purpose I/O function allows use of up to 16 input points and 16 output points.

* However, the general-purpose I/O function shares some ports with the high-speed counter function and PWM output function. Therefore, use IO parameter settings to select functions to be used.

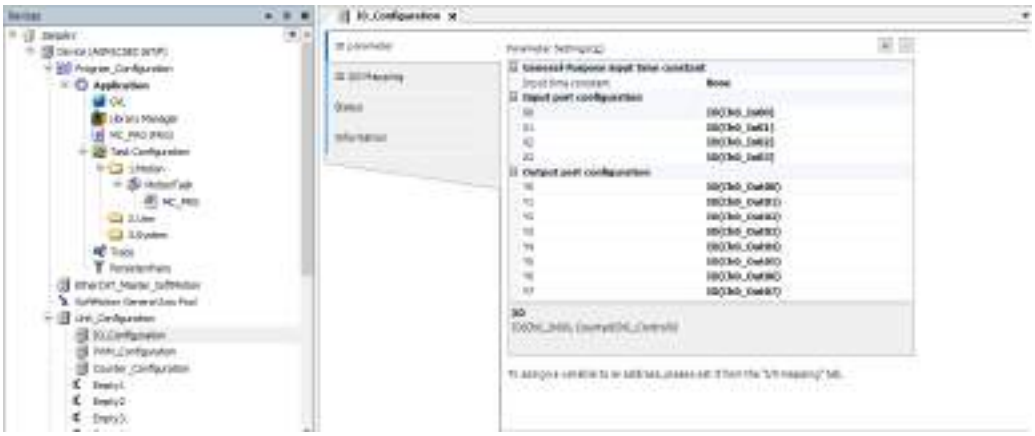
■ Specification overview

Item	Specifications	Remarks
Number of input ports	Max. 16 points	X0 to X3 are shared with the counter function. Use IO parameter settings to select functions to be used.
Number of output ports	Max. 16 points	Y0 to Y3 are shared with the counter function and Y4 to Y7 are shared with the PWM output function. Use IO parameter settings to select functions to be used.
Input port number	X0 to X15	
Output port number	Y0 to Y15	

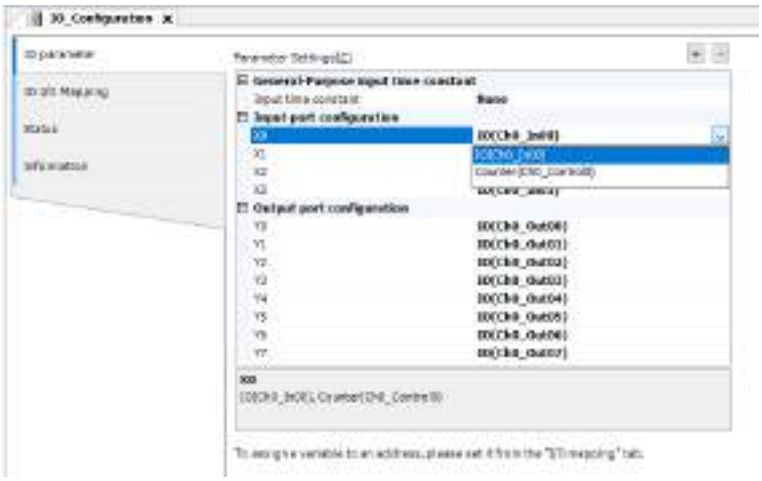
12.4.2 Setting Parameters with GM Programmer

Procedure

1. In the navigator pane, double-click the "IO_Configuration" object.
   The general-purpose IO setting pane will be displayed.



2. Set up general-purpose IO parameters.
   Select an IO parameter to be changed and then select a desired item from the drop-down list.



3. Select the "IO I/O Mapping" tab and set the correspondence (mapping) between the channel and variable in the mapping setting pane.

   Click the "Variable" column corresponding to the channel to be used by the program and enter a variable name.

   Clicking the mark in the "Mapping" column allows you to change the type of mapping.

Info.

- You can copy the variable name set in the Channel column.

  Select a channel (CH0 or CH1) in the "Category Selection" column, right-click, and then select [Copy] from the context-sensitive menu that is displayed. Next, select another channel, right-click, and then select [Paste] from the context-sensitive menu that is displayed.

12.4.3 Setting Items of IO_Configuration Parameters

■ IO parameters

Setting item	Setting item	Settings	Default value	Description
General-purpose input time constant	Input time constant	None 0.1 ms 0.5 ms 1 ms 5 ms 10 ms	None	Input time constant

Setting item	Setting item	Settings	Default value	Description
		20 ms 70 ms		
Input function setting	X0	IO(Ch0_In00) Counter(Ch0_Control0)	IO(Ch0_In00)	Select X0
	X1	IO(Ch0_In01) Counter(Ch0_Control1)	IO(Ch0_In01)	Select X1
	X2	IO(Ch0_In02) Counter(Ch1_Control0)	IO(Ch0_In02)	Select X2
	X3	IO(Ch0_In03) Counter(Ch1_Control1)	IO(Ch0_In03)	Select X3
Output function setting	Y0	IO(Ch0_Out00) Counter(Ch0_ExternalOutput0)	IO(Ch0_Out00)	Select Y0
	Y1	IO(Ch0_Out01) Counter(Ch0_ExternalOutput1)	IO(Ch0_Out01)	Select Y1
	Y2	IO(Ch0_Out02) Counter(Ch1_ExternalOutput0)	IO(Ch0_Out02)	Select Y2
	Y3	IO(Ch0_Out03) Counter(Ch1_ExternalOutput1)	IO(Ch0_Out03)	Select Y3
	Y4	IO(Ch0_Out04) PWM(Ch0_PWM_Output)	IO(Ch0_Out04)	Select Y4
	Y5	IO(Ch0_Out05) PWM(Ch1_PWM_Output)	IO(Ch0_Out05)	Select Y5
	Y6	IO(Ch0_Out06) PWM(Ch2_PWM_Output)	IO(Ch0_Out06)	Select Y6
	Y7	IO(Ch0_Out07) PWM(Ch3_PWM_Output)	IO(Ch0_Out07)	Select Y07

12.4.4 I/O Mapping for General-purpose I/O

Channel	Type	Description	Remarks
Ch0_In	WORD	Ch0_In	
Ch0_Out	WORD	Ch0_Out	

■ Ch0_In

Channel	Type	Description	Remarks
Ch0_In00	BOOL	Ch0_In00	
Ch0_In01	BOOL	Ch0_In01	

Channel	Type	Description	Remarks
Ch0_In02	BOOL	Ch0_In02	
Ch0_In03	BOOL	Ch0_In03	
Ch0_In04	BOOL	Ch0_In04	
Ch0_In05	BOOL	Ch0_In05	
Ch0_In06	BOOL	Ch0_In06	
Ch0_In07	BOOL	Ch0_In07	
Ch0_In08	BOOL	Ch0_In08	
Ch0_In09	BOOL	Ch0_In09	
Ch0_In10	BOOL	Ch0_In10	
Ch0_In11	BOOL	Ch0_In11	
Ch0_In12	BOOL	Ch0_In12	
Ch0_In13	BOOL	Ch0_In13	
Ch0_In14	BOOL	Ch0_In14	
Ch0_In15	BOOL	Ch0_In15	

■ Ch0_Out

Channel	Type	Description	Remarks
Ch0_Out00	BOOL	Ch0_Out00	
Ch0_Out01	BOOL	Ch0_Out01	
Ch0_Out02	BOOL	Ch0_Out02	
Ch0_Out03	BOOL	Ch0_Out03	
Ch0_Out04	BOOL	Ch0_Out04	
Ch0_Out05	BOOL	Ch0_Out05	
Ch0_Out06	BOOL	Ch0_Out06	
Ch0_Out07	BOOL	Ch0_Out07	
Ch0_Out08	BOOL	Ch0_Out08	
Ch0_Out09	BOOL	Ch0_Out09	
Ch0_Out10	BOOL	Ch0_Out10	
Ch0_Out11	BOOL	Ch0_Out11	
Ch0_Out12	BOOL	Ch0_Out12	
Ch0_Out13	BOOL	Ch0_Out13	
Ch0_Out14	BOOL	Ch0_Out14	
Ch0_Out15	BOOL	Ch0_Out15	

12.5 PWM Output

12.5.1 Overview of PWM Output

The PWM output function enables up to 100 kHz of PWM output to be obtained within a range of 0% to 100%.

■ Outline of specifications

Item	Specifications	Remarks
Number of output channels	Max. 4 channels	
Output port number	Y4 to Y7	
Output frequency	1 Hz to100 kHz (Settable by 1 Hz)(Note 1)	
Output duty ratio	0% to 100% (Settable by 0.1%)	
Control input	Enable request or start request	

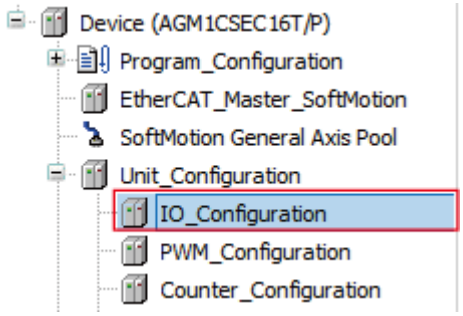
(Note 1) This specification applies when push-pull is set and output current is 0.1 A. It varies according to loads.

12.5.2 Setting Output Ports with GM Programmer

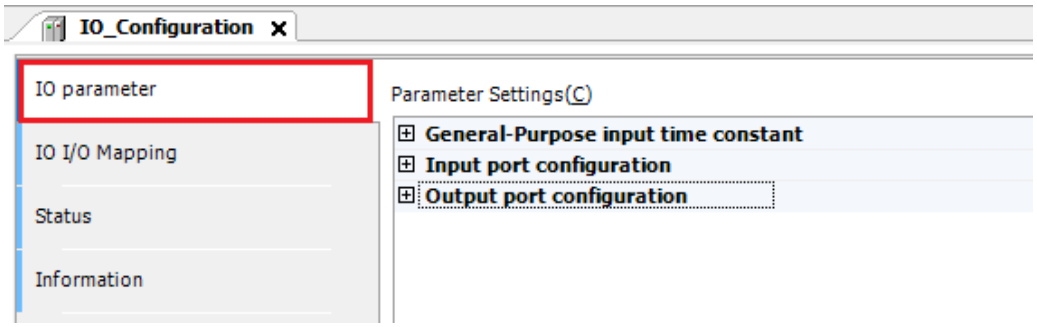
You can set output ports for PWM output via **IO_Configuration**>**Edit Object** in GM Programmer.

Procedure

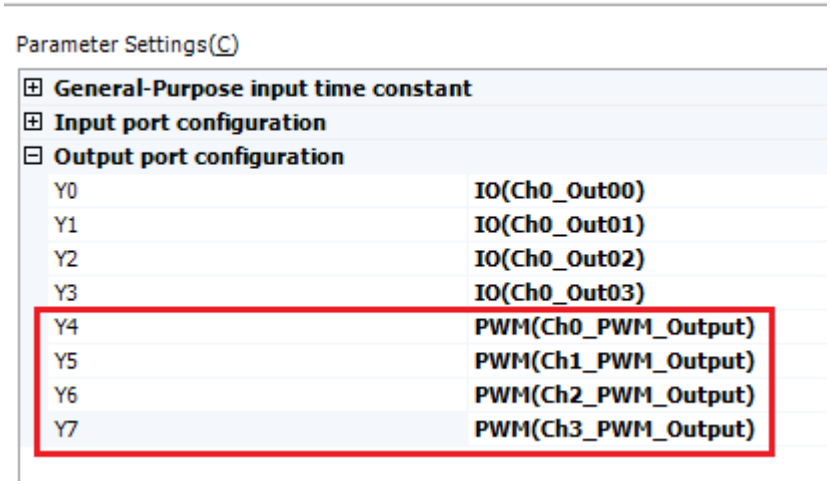
1. From "Device view" in the navigator pane, double-click "IO_Configuration".



2. Click the "IO Parameters" tab.



3. If necessary, change Y4 to Y7 in "Output port configuration" to PWM(Ch0_PWM_Output) to PWM(Ch3_PWM_Output), respectively.



12.5.3 I/O Mapping for PWM Output

The high-speed counter function is controlled by user programs.

■ InputArea (input area)

Channel	Type	Description	Remarks
PwmStatusRegister	WORD	Input area	-

■ PwmStatusRegister (PWM status register)

Channel	Type	Description	Remarks
Ch*_PwmStatus	BOOL	Ch* PWM output status	Indicates the PWM output state. FALSE: OFF TRUE: ON

■ OutputArea (output area)

Channel	Type	Description	Remarks
PwmRequestRegister	WORD	PWM request register	-
Ch*_FrequestValue	UDINT	Ch* frequency set value	Unit: Hz (0 to 100,000 Hz)
Ch* DutyValue	UINT	Ch* duty ratio set value	Unit: 0.1% (0 to 100.0%)

■ PwmRequestRegister (PWM request register)

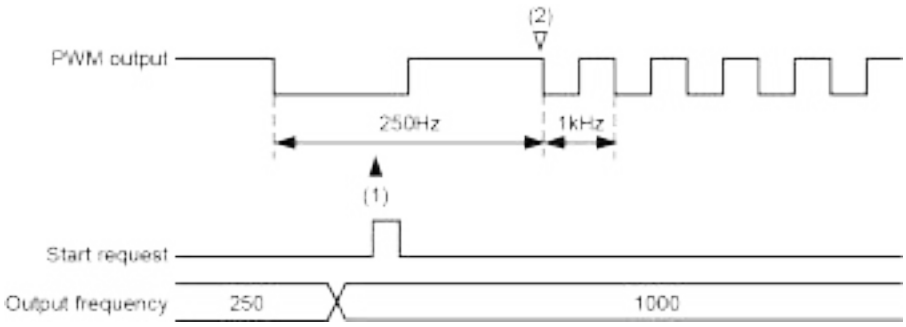
Channel	Type	Description	Remarks
Ch*_PwmStartRequest	BOOL	Ch* start request	PWM output is started at the rising edge.
Ch*_PwmEnableRequest	BOOL	Ch* enable request	FALSE: Disables PWM output TRUE: Enables PWM output

12.5.4 Data Update Timing (Output Frequency)

The data update timing for output frequency during PWM output is descried below.

■ Data update at the rising edge of start request bit

In this mode, the frequency value to be changed is written to the frequency set value and updated with data at the point in time when the start request bit is switched from OFF to ON. The frequency value changed in this timing is reflected at the falling edge of the pulse that is being output.



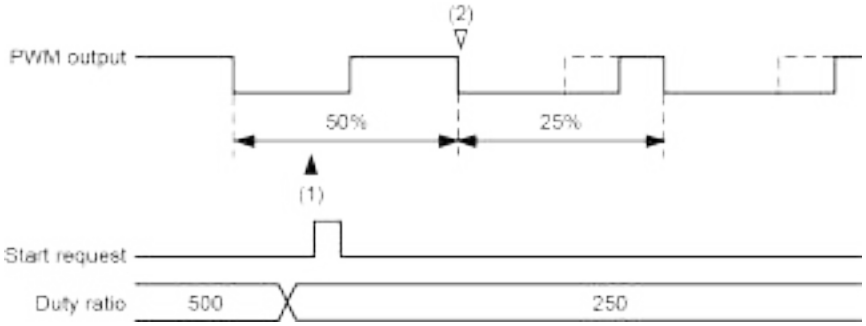
No.	Name	Description
(1)	Data update timing	Indicates the timing in which the frequency set value is reflected
(2)	Output update timing	Indicates the timing in which the changed frequency is reflected as actual output

12.5.5 Data Update Timing (Duty Ratio)

The data update timing for duty ratios during PWM output is descried below.

■ Data update at the rising edge of start request bit

In this mode, the duty ratio to be changed is written to the duty ratio set value and updated with data at the point in time when the start request bit is switched from OFF to ON. The duty ratio changed in this timing is reflected at the falling edge of the pulse that is being output.

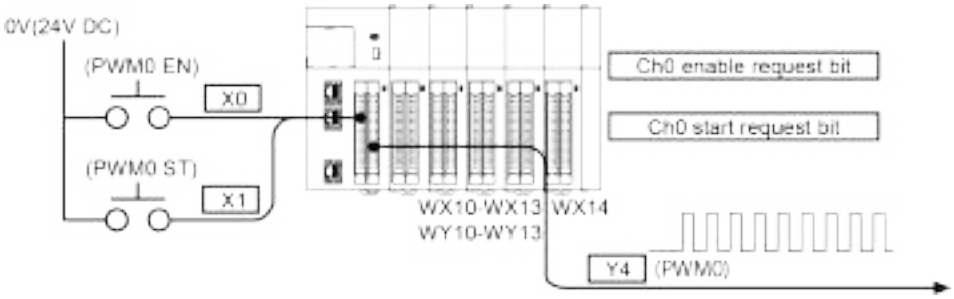


No.	Name	Description
(1)	Data update timing	Indicates the timing in which the duty ratio set value is reflected
(2)	Output update timing	Indicates the timing in which the changed duty ratio is reflected as actual output

12.5.6 PWM Output Setting Example

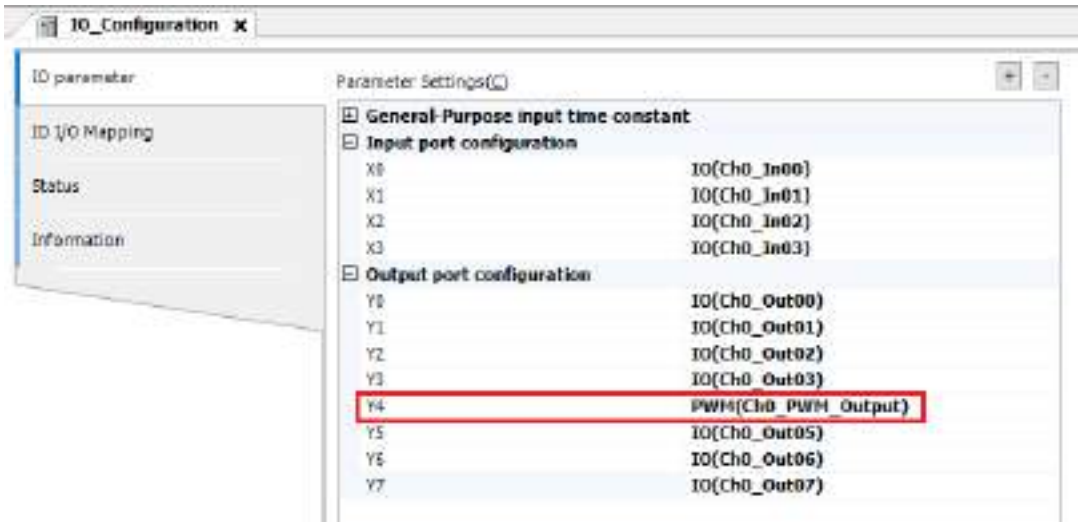
■ Overview

PWM output is performed. It is controlled by the switch input (X0 or X1) connected to the GM1 controller. If the start request bit is turned ON when the enable request bit is ON, PWM output will be started.



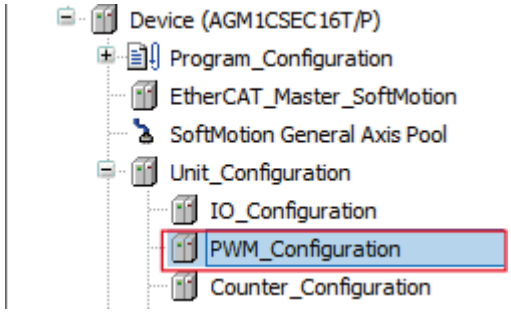
■ Settings in GM Programmer

For example, to set PWM output for Y4, use the procedure in "12.5.2 Setting Output Ports with GM Programmer" to configure settings as shown below.

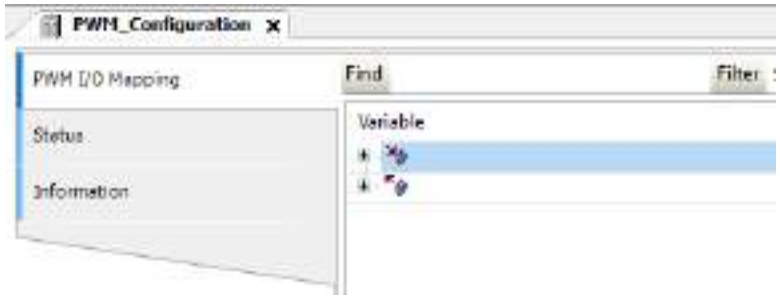


Next, perform I/O mapping for variables created in POU.

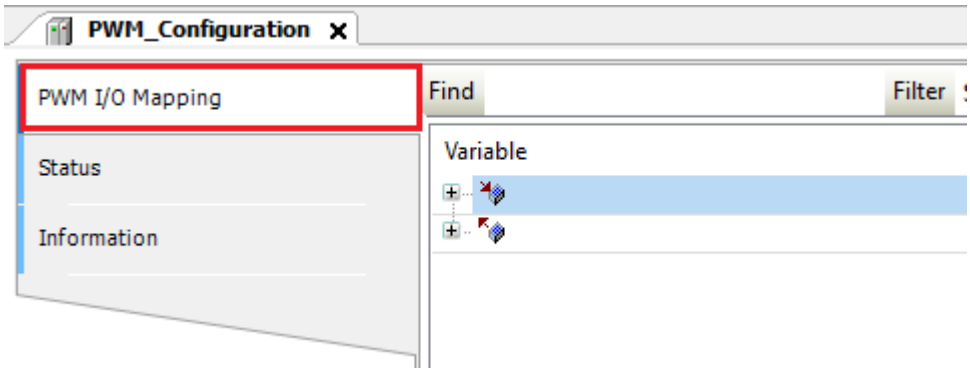
1. From "Device view" in the navigator pane, double-click "PWM_Configuration".



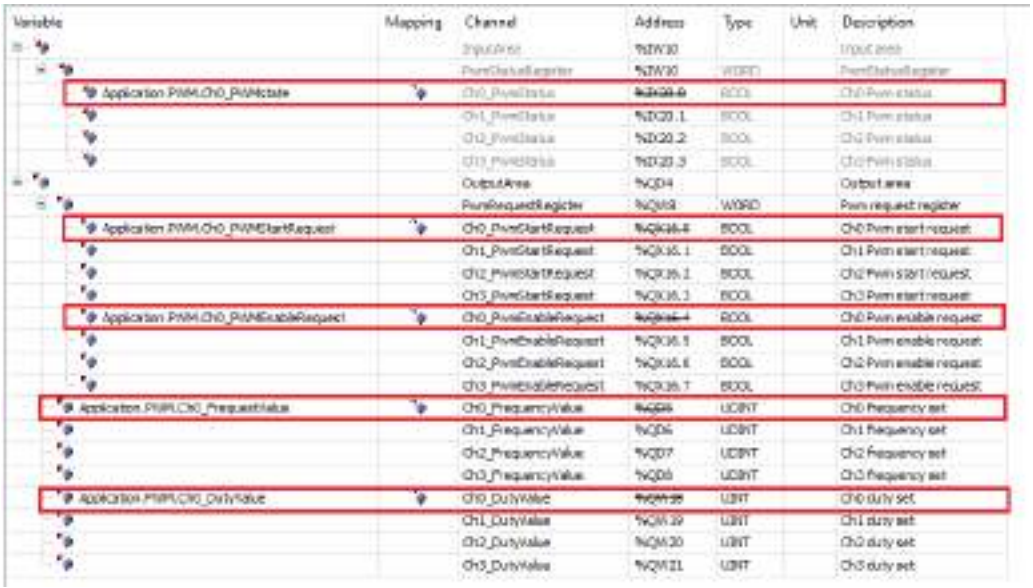
The "PWM_Configuration" window will be displayed.



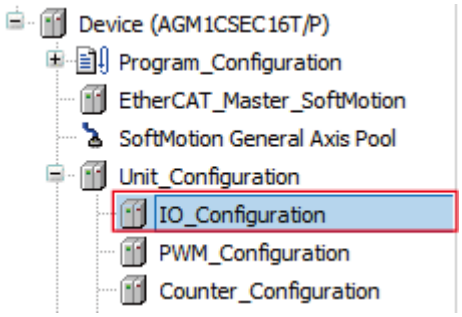
2. In the "PWM_Configuration" window, click the "PWM I/O Mapping" tab.



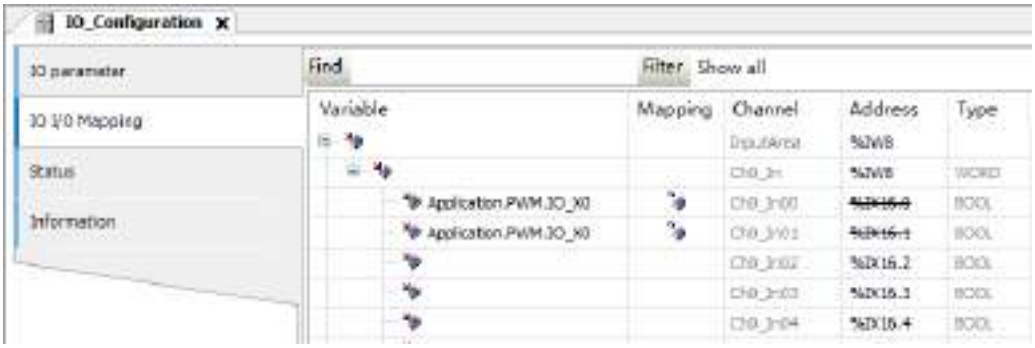
3. In this example, variables with the same name as the channel name are mapped to the channels required for PWM Ch0 (the name of sample POU is created as "PWM").



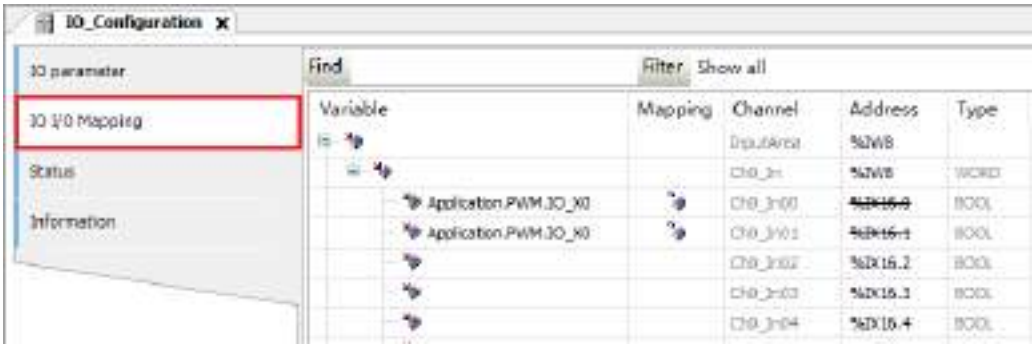
4. From Device view in the navigator pane, double-click "IO_Configuration".



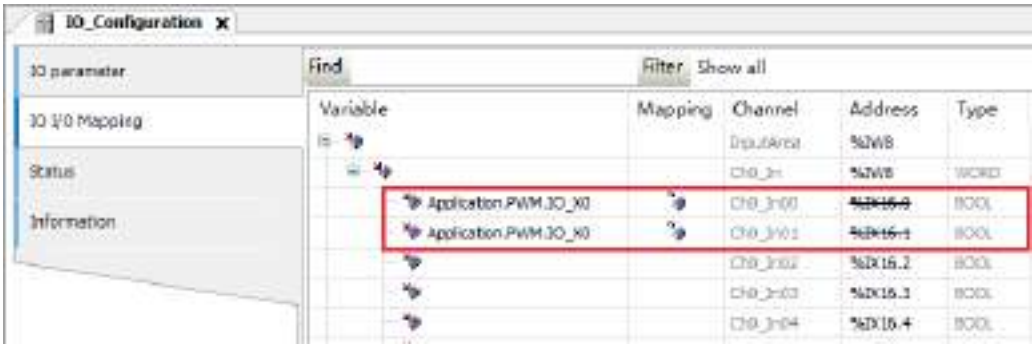
The "IO_Configuration" window will be displayed.



5. Click the "IO I/O Mapping" tab.

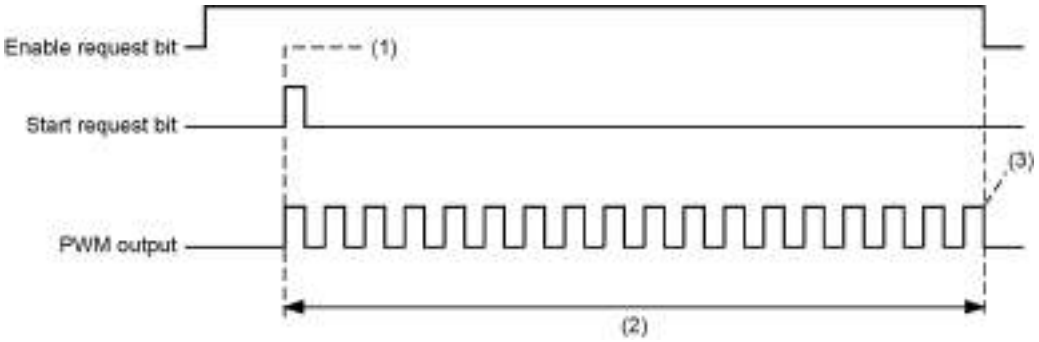


6. Variables will be mapped.



■ Timing chart

If the rising edge of the start request bit is detected when the enable request bit is ON, Y4 will start PWM output. When the enable request bit is set to OFF, PWM output stops.

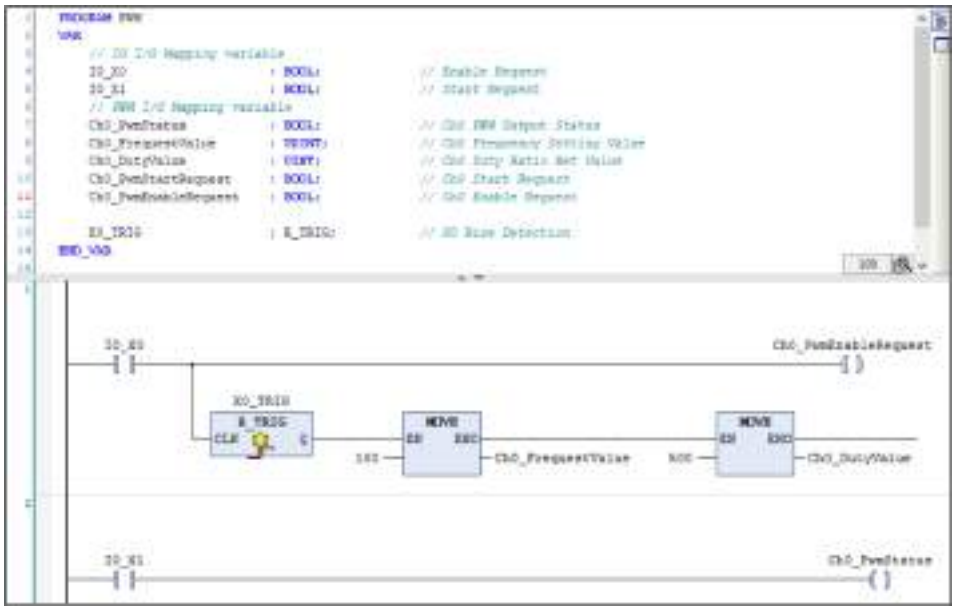


No.	Description
(1)	If the rising edge of the start request bit is detected when the enable request bit is ON, PWM output will be started.
(2)	PWM output is performed with a duty ratio of 50% and at a frequency of 100 Hz.
(3)	When the enable request bit turns OFF, PWM output stops.

■ Sample program

The following are LD program and ST program examples for sample POU (PWM).
The state of X0 is output to the Ch0 Enable Request bit.
Ch0 frequency and Ch0 duty ratio are set at the rising edge of X0.
The state of X1 is output to the Ch0 Start Request bit.

LD program



ST program

```
PROGRAM PWM
VAR
    // IO I/O Mapping variable
    IO_X0                   : BOOL;             // Enable Request
    IO_X1                   : BOOL;             // Start Request
    // PWM I/O Mapping variable
    Ch0_PwmStatus           : BOOL;             // Ch0 PWM Output Status
    Ch0_FrequestValue       : UDINT;            // Ch0 Frequency Setting Value
    Ch0_DutyValue           : UINT;             // Ch0 Duty Ratio Set Value
    Ch0_PwmStartRequest     : BOOL;             // Ch0 Start Request
    Ch0_PwmEnableRequest    : BOOL;             // Ch0 Enable Request

    X0_TRIG                 : R_TRIG;           // X0 Rise Detection
END_VAR
```

```
Ch0_PwmEnableRequest := IO_X0;      // Output X0 status to Ch0 enable request

X0_TRIG( CLK:=IO_X0 );
IF X0_TRIG.Q = TRUE THEN            // X0 rise detection
    Ch0_FrequestValue := 100;       // Set frequency to 100HZ
    Ch0_DutyValue := 500;           // Duty ratio set to 50%
END_IF

Ch0_PwmStartRequest := IO_X1;       // Output X1 status to Ch0 start request
```

12.6 High-speed Counter Function

12.6.1 Overview of High-speed Counter Function

■ Two 4-MHz, signed 32-bit high-speed counters are provided

- High-speed counting of input signals is available for up to the maximum frequency 4 MHz (or 16 MHz for 2-phase input 4 multiple). Two-phase input (phase differential input), individual input, or direction identification input can be selected according to the input device such as encoders or sensors.

■ 24 VDC, 12 VDC, and 5 VDC inputs and line driver input are supported

- The count input circuit supports both open collector output and line driver output (differential output: equivalent to AM26LS31).

■ Ring counter or linear counter can be selected

- Both the ring counter and linear counter are supported. Both types can use the Z-phase of an encoder as count reset timing.

■ Internal clock counting is possible

- Internal clocks can be selected as count input signals. High-accuracy time measurements can be made with a maximum resolution of 0.25 μs. Selectable internal clocks are 0.25 μs (4 MHz), 1 μs (1 MHz), 10 μs (100 kHz), and 100 μs (10 kHz).

■ Capture function and sampling capture function are provided

- It is possible to store the count value at the moment of the occurrence of a trigger assigned to a capture flag. Count values can be checked, independently of I/O refresh.
- The capture function executes capturing at the rising and falling edges of a capture flag. The sampling capture function executes capturing at every sampling time according to the input of a capture flag.

■ Various counter operations can be selected

1. Enable count operation
2. Reset count operation
3. Preset count operation
4. Enable reset count operation
5. Enable preset count operation

■ Band comparison function and target value match comparison function are provided

- The band comparison function allows up to 16 pairs of upper and lower limits to be set for each counter. Up to 16 comparison match flags can be turned ON or OFF for each comparison condition.
- The target value match comparison function allows up to16 target values to be set for each counter. Target values can be set or reset individually according to the direction of counting (incrementation or decrementation) when the count value reaches the target value or when

up to 16 comparison match flags for each comparison condition match their respective target values.

- For each counter, up to two comparison match flags and external output signals can be linked with each counter.

■ **ON hold time setting (for band comparison function only)**

- The ON state of external output signals is retained for the set time (1 to 1,000 ms).

■ **Input time constant (noise filter)**

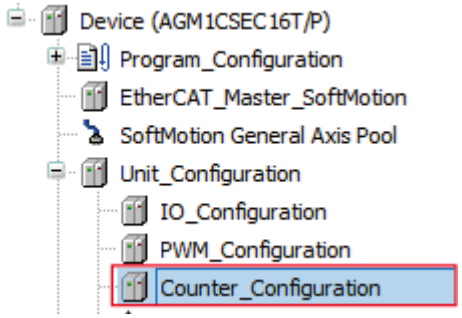
- Input time constants can be set as input signals (A-phase, B-phase, and Z-phase) and control signals for each counter.

12.6.2 Setting Parameters with GM Programmer

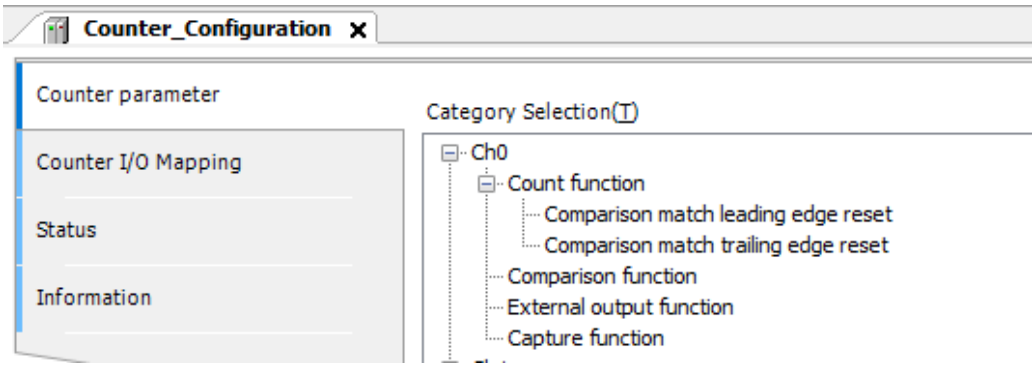
You can set parameters for the high-speed counter via Couter_Configuration in GM Programmer.

Procedure

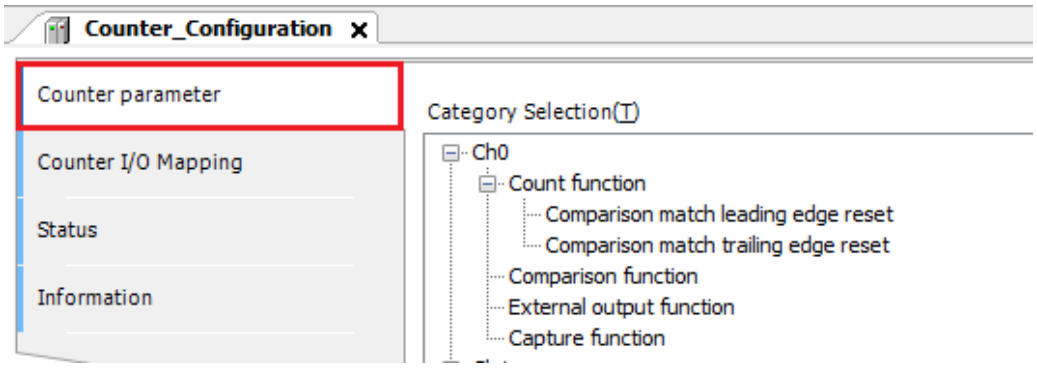
1. From "Device view" in the navigator pane, double-click "Counter_Configuration".



The "Counter_Configuration" setting window will be displayed.



2. In the "Counter_Configuration" window, click the "Counter parameter" tab.



3. Set up parameters of each function for each counter.

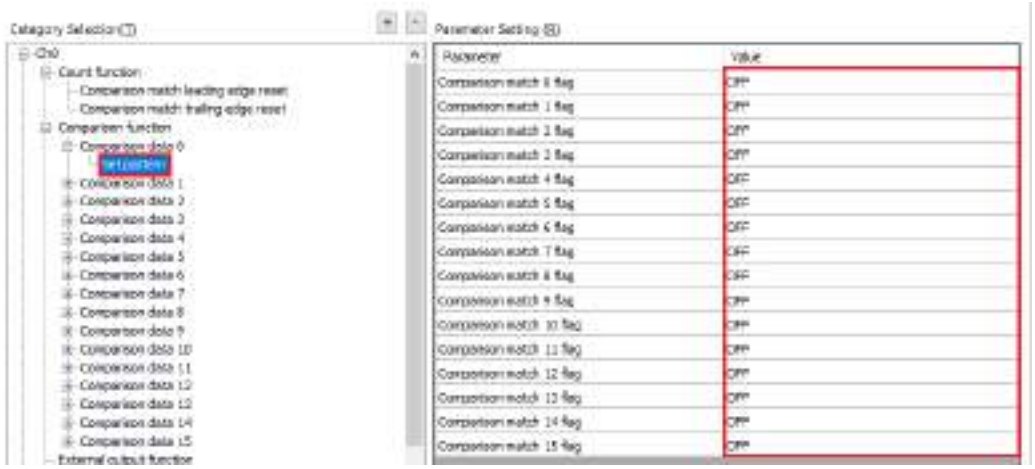
Counter function

Comparison match rising edge reset / Comparison match falling edge reset

Comparison function (band comparison)

Comparison function (band comparison) comparison data

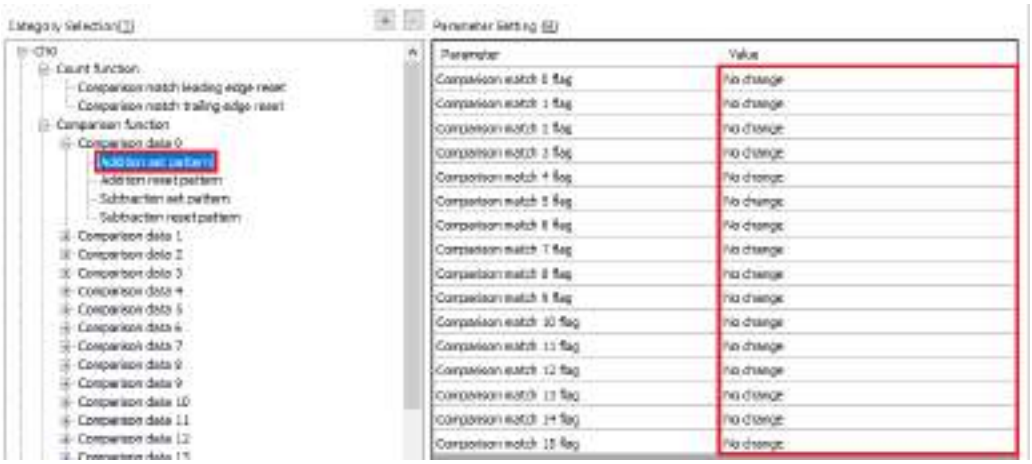
Set patterns for comparison function (band comparison) comparison data



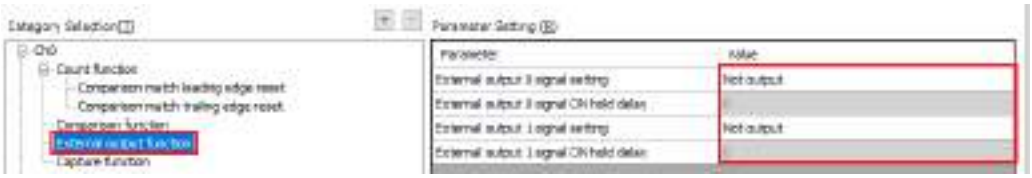
Comparison function (target value match comparison)

Comparison function (target value match comparison) comparison data

Set / reset patterns for comparison function (target value match comparison) comparison data



External output function



Capture function

Info.

- You can copy the parameter set in a counter. To do so, select a channel (Ch0 or Ch1) in the "Category Selection" column and click the [Copy] button.
  Next, select another counter and click the [Paste] button.
- For details on each parameter, refer to "12.6.3 Counter Parameter Setting Items".

12.6.3 Counter Parameter Setting Items

■ Count function (Settable for each counter)

Setting item	Settings	Default value
Counter type	Linear counter / Ring counter	Linear counter
Enable/Disable overflow/ underflow	Disable / Enable	Disable

Setting item	Settings	Default value
Counter upper limit	-2,147,483,647 to 2,147,483,647	2,147,483,647
Counter lower limit	-2,147,483,648 to 2,147,483,646	-2,147,483,648
Specify count direction	Count in normal direction / Count in reverse direction	Count in normal direction
Select count input	Count signal Internal clock 0.25 μs (4 MHz) Internal clock 1.00 μs (1 MHz) Internal clock 10 μs (100 kHz) Internal clock 100 μs (10 kHz)	Count signal
Count method	2-phase input 1 multiple / 2-phase input 2 multiple / 2-phase input 4 multiple / Individual input 1 multiple / Individual input 2 multiple / Direction detection input 1 multiple / Direction detection input 2 multiple	2-phase input 1 multiple
Input Z signal function setting	Not used Reset operation at rising edge Reset operation at falling edge Positive logic reset operation Negative logic reset operation Preset operation at rising edge Preset operation at falling edge Positive logic preset operation Negative logic preset operation	Not used
Control 0 signal function setting	Not used Positive logic enable operation Negative logic enable operation Positive logic enable operation, reset operation at rising edge Negative logic enable operation and reset operation at falling edge Positive logic enable operation and preset operation at rising edge Negative logic enable operation and preset operation at falling edge	Not used
Control 1 signal function setting	Not used Positive logic enable operation Negative logic enable operation	Not used
Default value	Overwrites the count value with the default value when the power is turned ON	0
Input A signal/Input B signal input time constant	No input time constant / 0.1 μs (2 MHz) / 0.2 μs (1 MHz) / 0.5 μs (500 kHz) / 1.0 μs (250 kHz) / 2.0 μs (100 kHz) / 10.0 μs (10 kHz)	2.0 μs (100 kHz)
Input Z signal input time constant	No input time constant / 0.1 μs (2 MHz) / 0.2 μs (1 MHz) / 0.5 μs (500 kHz) / 1.0 μs (250 kHz) / 2.0 μs (100 kHz) / 10.0 μs (10 kHz)	
Control signal input time constant	No input time constant / 2 μs / 5 μs / 10 μs / 20 μs / 50 μs / 100 μs / 500 μs / 1.0 ms / 2.0 ms / 5.0 ms / 10.0 ms	2.0 ms

(Note 1) The control 0 signal and control 1 signal cannot be assigned to the capture function if they are assigned to the enable operation.

Setting item	Settings	Default value
Comparison match leading edge reset / Comparison match trailing edge reset	Selects a comparison match flag that resets the count value at the rising edge or falling edge. Not reset / Reset	Not reset

■ Comparison function (Settable for each counter)

Setting item	Settings	Default value
Select comparison function	Not use / Band comparison / Target value match comparison	Not use
Select comparison input	Count value	Count value
Set number of comparison data	Sets the number of data items to be compared Setting range: 1 to 16	16

If you select "Band comparison" or "Target value match comparison" for "Select comparison function", set parameters for each comparison data item.

Setting item	Settings	Default value
Comparison data 0 to Comparison data 15 (for band comparison)	Specifies the lower and upper limits for each comparison data (Note 1) Setting range: -2,147,483,648 to 2,147,483,647	0
	Specifies the state of the comparison match flag when the current value falls within the specified band ON / OFF	OFF
Comparison data 0 to Comparison data 15 (for target value match comparison)	Specifies target values for each comparison data Setting range: -2,147,483,648 to 2,147,483,647	0
	Comparison match flags to be set or reset can be selected for each status (incrementation or decrementation) of comparison data that has reached the target value. ● Addition set pattern: Set / No change ● Addition reset pattern: Reset / No change ● Subtraction set pattern: Set / No change ● Subtraction reset pattern: Reset / No change	No change

(Note 1) Lower and upper limits can be set within the range between the lower and upper limits for the counter.

For linear counters, set each limit so that the lower limit is less than the upper limit. For ring counters, lower and upper limits can be set in any range.

■ External output function (Settable for each counter)

Setting item	Settings	Default value
External output 0 signal setting	Not output / Output	Not output
External output 0 signal ON hold time	Setting range: 0 to 1,000 (ms)	0 ms
External output 1 signal setting	Not output / Output	Not output

Setting item	Settings	Default value
External output 1 signal ON hold delay	Setting range: 0 to 1,000 (ms)	0 ms

(Note 1) "ON hold time" is enabled only when the band comparison function is used.

■ Capture function (Settable for each counter)

Setting item	Settings	Default value
Capture 0 setting	Not use capture 0 function Capture function at rising edge of control 0 signal Capture function at falling edge of control 0 signal Capture function at rising edge of control 1 signal Capture function at falling edge of control 1 signal Control 0 signal positive logic sampling capture function Control 0 signal negative logic sampling capture function Control 1 signal positive logic sampling capture function Control 1 signal negative logic sampling capture function Output relay (Y relay) sampling capture function	Not use capture 0 function
Capture 1 setting	Not use capture 1 function Capture function at rising edge of control 0 signal Capture function at falling edge of control 0 signal Capture function at rising edge of control 1 signal Capture function at falling edge of control 1 signal	Not use capture 1 function
Capture function operation setting	One operation / Continuous operation	Continuous operation
Sampling time (ms)	1 to 65,535	1

(Note 1) If any value related to the sampling capture function is selected for "Capture 0 setting", "Capture 1 setting" will be disabled.

12.6.4 I/O Mapping for High-speed Counter Output

The high-speed counter function is controlled by user programs.

■ InputArea (input area)

Channel	Type	Description	Remarks
Ch*_StatusRegister	WORD	Ch* status register	-
Ch*_ComparisonMatchRegister	WORD	Ch* comparison match flag	-
Ch*_CountValue	DINT	Ch* count value	-
Ch*_Capture0Value	DINT	Ch* capture 0 value	-
Ch*_Capture1Value	DINT	Ch* capture 1 value	-

Channel	Type	Description	Remarks
Ch*_CaptureDifferenceValue	DINT	Ch* capture differential value	Stores the value (Capture 1 value - Capture 0 value)

■ Ch*_StatusRegister (Ch* status register)

Channel	Type	Description	Remarks
Ch*_OperationReadyStatus	BOOL	Ch* operation ready status	Indicates whether the count function is ready to run. 0: Getting ready, 1: Ready
CH*_CountEnableStatus	BOOL	Ch* count enable status	Indicates whether count operation is in progress. 0: Stopped, 1: Operation in progress
Ch*_CountDirectionStatus	BOOL	Ch* count direction status	Indicates the direction of counting. 0: Reverse rotation (decrementation direction), 1: Forward rotation (incrementation direction)
Ch*_Capture0Status	BOOL	Ch* capture 0 status	Indicates that the count value is stored as capture 0 value at the rising edge or falling edge of the control signal, whichever is enabled.
Ch*_Capture1Status	BOOL	Ch* capture 1 status	Indicates that the count value is stored as capture 1 value at the rising edge or falling edge of the control signal, whichever is enabled.
Ch*_ExternalOutput0Status	BOOL	Ch* external output 0 status	Indicates the output status of external output 0 signal. 0: Output OFF, 1: Output ON
Ch*_ExternalOutput1Status	BOOL	Ch* external output 1 status	Indicates the output status of external output 1 signal. 0: Output OFF, 1: Output ON
Ch*_InputAStatus	BOOL	Ch* input A status	Indicates the input status of input A signal. 0: Output OFF, 1: Output ON
Ch*_InputBStatus	BOOL	Ch* input B status	Indicates the input status of input B signal. 0: Output OFF, 1: Output ON
Ch*_InputZStatus	BOOL	Ch* input Z status	Indicates the input status of input Z signal. 0: Output OFF, 1: Output ON
Ch*_Control0Status	BOOL	Ch* control 0 status	Indicates the input status of control 0 signal. 0: Output OFF, 1: Output ON
Ch*_Control1Status	BOOL	Ch* control 1 status	Indicates the input status of control 1 signal. 0: Output OFF, 1: Output ON

■ Ch*_ComparisonMatchRegister (Ch* comparison match flag)

Channel	Type	Description	Remarks
Ch*_ComparisonMatch0Status to	BOOL	Ch* comparison match 0 flag	Outputs the result of the band comparison function or target value match function. 0: Unmatched, 1: Matched

Channel	Type	Description	Remarks
Ch*_ComparisonMatch15Status		to Ch* comparison match 15 flag	

■ OutputArea (output area)

Channel	Type	Description	Remarks
Ch*_RequestRegister	WORD	Ch* request register	-
Ch*_TemporaryPresetValue	DINT	Ch* temporary preset value	Stores the value to replace the preset value. -2,147,483,648 to 2,147,483,647
Ch*_TemporaryCurrentValue	DINT	Ch* temporary current value	Stores the value to replace the count value. -2,147,483,648 to 2,147,483,647

■ Ch*_RequestRegister (Ch* request register)

Channel	Type	Description	Validity condition	Remarks
Ch*_OperationReadyRequest	BOOL	Ch* operation ready request	Level	Specifies whether to enable operation preparation for the count function. 0: Disable, 1: Enable
Ch*_CountEnableRequest	BOOL	Ch* count enable request	Level	Specifies whether to enable count operation. 0: Disable, 1: Enable
Ch*_ResetRequest	BOOL	Ch* reset request	ON edge	Specifies whether to reset the count value. 0: Do not reset 1: Reset
Ch*_PresetRequest	BOOL	Ch* preset request	ON edge	Specifies whether to change the count value to a preset value. 0: Do not change, 1: Change
Ch*_ResetEnableRequest	BOOL	Ch* reset enable request	Level	Capture function: Specifies whether to enable reset count operation for the input Z signal or comparison match flag. 0: Disable, 1: Enable Sampling capture function Specifies whether to enable reset count operation. 0: Disable, 1: Enable
Ch*_CurrentValueChangeRequest	BOOL	Ch* current value change request	ON edge	Specifies whether to change the count value to a temporarily current value. 0: Do not change, 1: Change
Ch*_PresetValueChangeRequest	BOOL	Ch* preset value change request	ON edge	Specifies whether to change the preset value to a temporarily preset value. 0: Do not change, 1: Change

Channel	Type	Description	Validity condition	Remarks
Ch*_CaptureEnable Request	BOOL	Ch* capture enable request(Note 1)	Level	Capture function: Specifies whether to enable the capture function. 0: Disable, 1: Enable Sampling capture function Used as a capture flag.
Ch*_ExternalOutput 0ForcedONRequest	BOOL	Ch* external output 0 forced ON request	Level	Relay to forcibly turn on the external output 0 signal
Ch*_ExternalOutput 0ForcedOFFReques t	BOOL	Ch* external output 0 forced OFF request	Level	Relay to forcibly turn off the external output 0 signal
Ch*_ExternalOutput 1ForcedONRequest	BOOL	Ch* external output 1 forced ON request	Level	Relay to forcibly turn on the external output 1 signal
Ch*_ExternalOutput 1ForcedOFFReques t	BOOL	Ch* external output 1 forced OFF request	Level	Relay to forcibly turn off the external output 1 signal
Ch*_ErrorClearRequ est	BOOL	Ch* error clearing request	ON edge	Specifies whether to clear the error. 0: Do not clear, 1: Clear

(Note 1) The behavior of the "Ch* capture enable request" bit differs according to the function to be used.

12.6.5 Operation Ready Request

■ Operation ready request program

To enable the settings of the high-speed counter function, you must configure parameter settings with GM Programmer and issue an operation ready request.

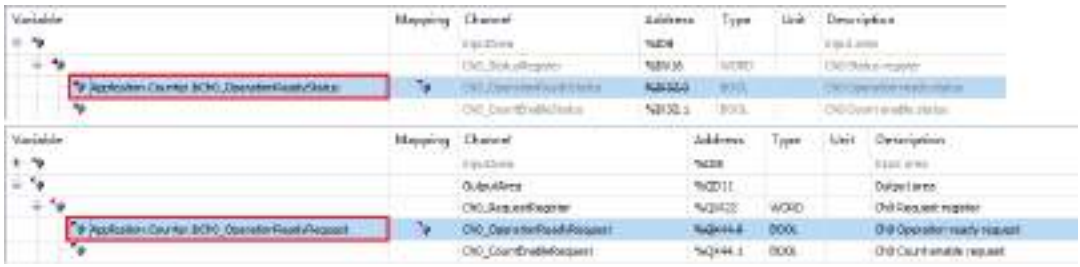
Ensure that the following operation ready request is issued before the high-speed counter is used.

Example: A program to request preparation for CH0 operation of the high-speed counter function

First, perform I/O mapping for variables created in POU.

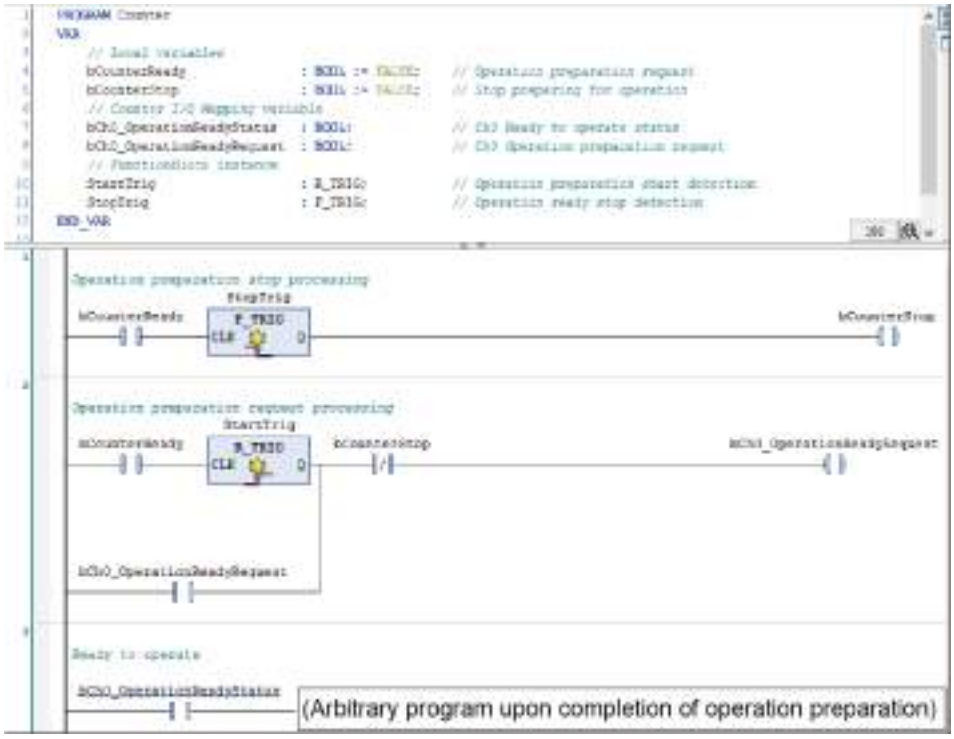
From "Device view" in the navigator pane, double-click "Counter_Configuration" and select the "Counter I/O mapping" tab.

In this example, variables are mapped to the channel used for high-speed counter Ch0 (the name of sample POU is created as "Counter").



The following are LD program and ST program examples for sample POU (Counter).
When the "OperationReadyRequest" flag is set to TRUE, operation preparation is started.

LD program



ST program

```
PROGRAM Counter
VAR
    // Local variables
    bCounterReady              : BOOL := FALSE;   // Operation preparation request
    bCounterStop               : BOOL := FALSE;   // Stop preparing for operation
    // Counter I/O Mapping variable
    bCh0_OperationReadyStatus  : BOOL;            // Ch0 Ready to operate status
    bCh0_OperationReadyRequest : BOOL;            // Ch0 Operation preparation request
    // FunctionBlock instance
    StartTrig                  : R_TRIG;          // Operation preparation start detection
    StopTrig                   : F_TRIG;          // Operation ready stop detection
END_VAR
```

```
// Operation preparation stop processing
StopTrig( CLK:=bCounterReady );
bCounterStop := StopTrig.Q;

// Operation preparation request processing
StartTrig( CLK:=bCounterReady );
bCh0_OperationReadyRequest := (StartTrig.Q OR bCh0_OperationReadyRequest) AND NOT(bCounterStop);

// Ready to operate
IF bCh0_OperationReadyStatus = TRUE THEN

    (Arbitrary program upon completion of operation preparation)

END_IF
```

■ Downloading IO parameters

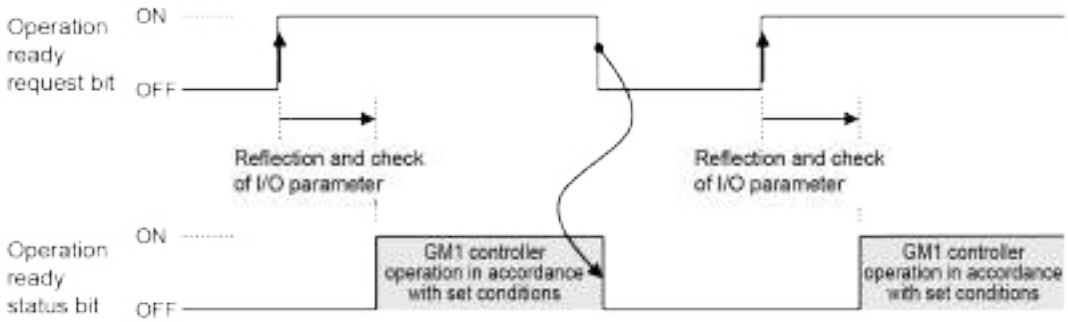
The IO parameters of the high-speed counter function are downloaded to the GM1 controller together with user programs.

When an operation ready request is issued in RUN mode and the operation ready status bit turns ON, each function of the high-speed counter function is enabled.

■ Behaviors when operation ready request program is executed

When the operation ready request program is executed and each behavior set in the GM1 controller becomes executable, the operation ready status bit turns ON.

Behaviors of operation ready request bit



(Note 1) Ensure that the operation ready request bit remains ON when the high-speed counter is used.

Info.

- If the default value or preset value is out of range, the operation ready status bit will not turn ON.

12.6.6 Count Function

■ Setup procedure

1. From "Device view" in the navigator pane, double-click "Counter_Configuration".
2. Click the "Counter Parameters" tab.
3. For each channel, select the count function and set up each parameter.

Info.

- For details on how to set up parameters, refer to "12.6.2 Setting Parameters with GM Programmer".

■ Counter upper limit and Counter lower limit

Set the upper limit and lower limit values for each counter.

Counter upper limit: Any value between -2,147,483,647 and 2,147,483,647 can be set (Default value: 2,147,483,647)

Counter lower limit: Any value between -2,147,483,648 and 2,147,483,646 can be set (Default value: -2,147,483,648)

■ Specify count direction

Set the rotational direction of count input.

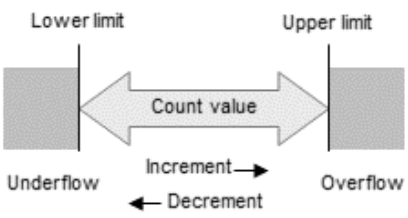
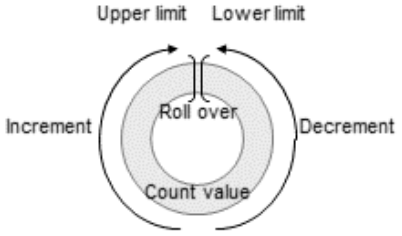
Count in normal direction: Counts in the direction stated in the manual

Count in reverse direction: Counts in the direction reverse to the one stated in the manual

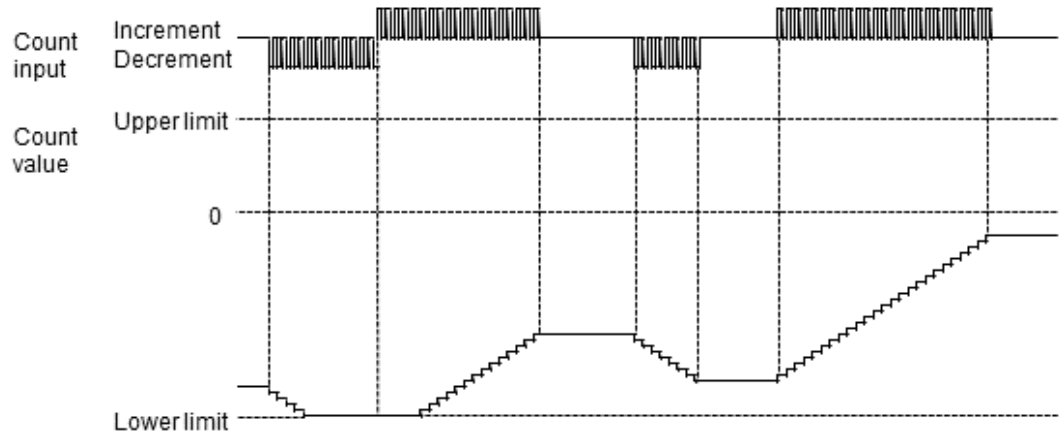
■ Counter type

The counter behavior differs according to the type of the counter as below.

Difference in behavior between counter types

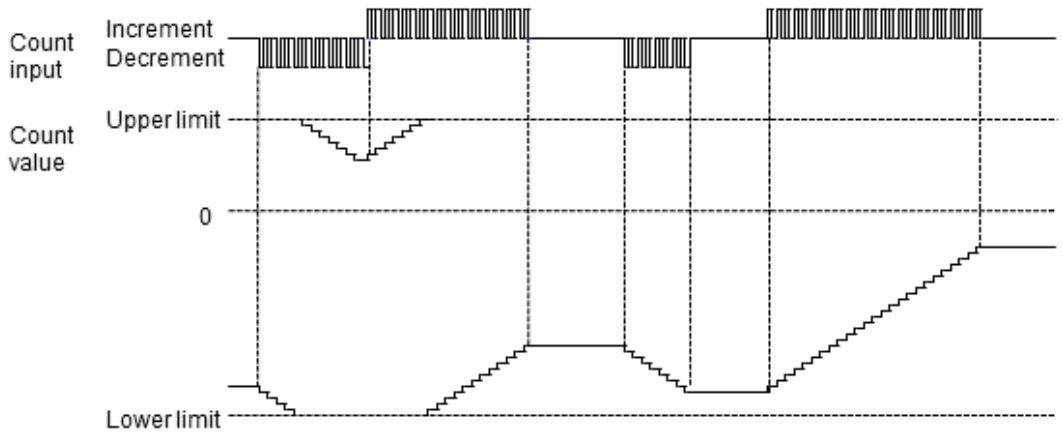
Item	Linear counter	Ring counter
Behavior image		

Behavior example of linear counter



Behavior example of ring counter

- If the count is decremented from the lower limit, the counter will roll over the count and continue a decremental count from the upper limit.
- If the count is incremented from the upper limit, the counter will roll over the count and continue an incremental count from the lower limit.



■ Enable / Disable overflow / underflow (for linear counters only)

Set the counter behavior to be performed when the count value reaches the specified upper limit or lower limit.

Disable: Continues counting within the countable range for the system (-2,147,483,648 to 2,147,483,647) even if the specified upper limit or lower limit is reached.

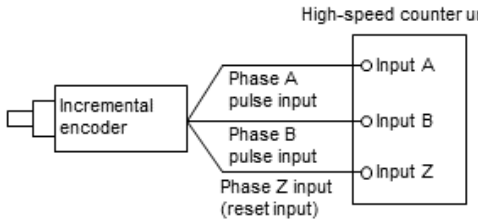
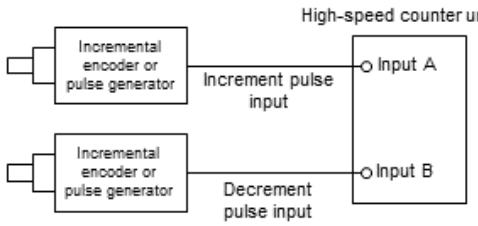
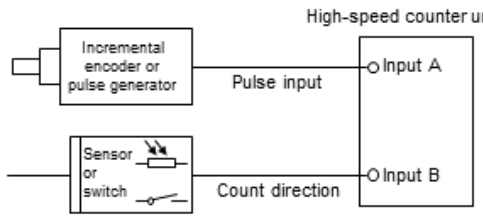
■ Select count input

- To import input signals from external devices, select "Count input".
- To measure the frequencies of external input signals or time based on the internal clock, select "Internal clock".
- For internal clocks, you can select from 0.25 µs (4 MHz), 1.00 µs (1 MHz), 10 µs (100 kHz), and 100 µs (10 kHz).

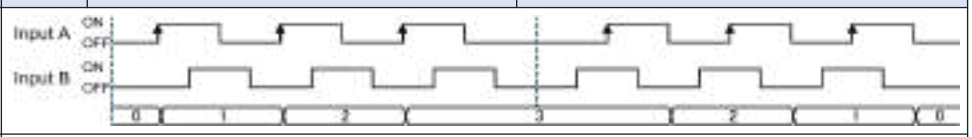
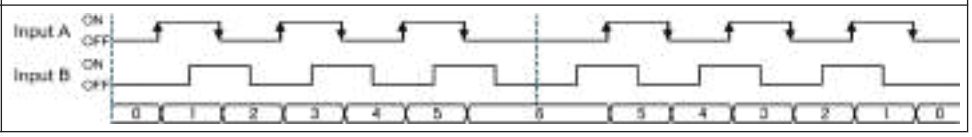
■ Count method

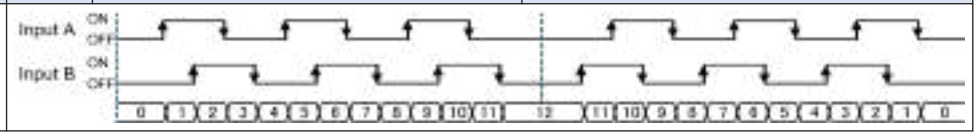
- You can select from the three types shown in the table below according to the input device to be connected.
- The count behavior changes according to the settings of a multiplication factor, as described on the following pages.

Count method

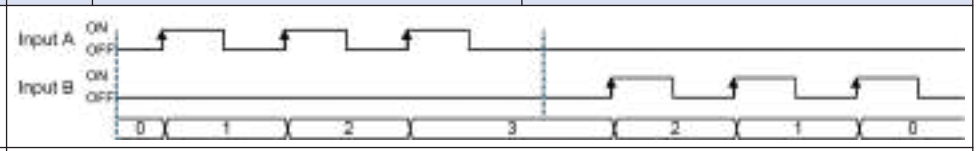
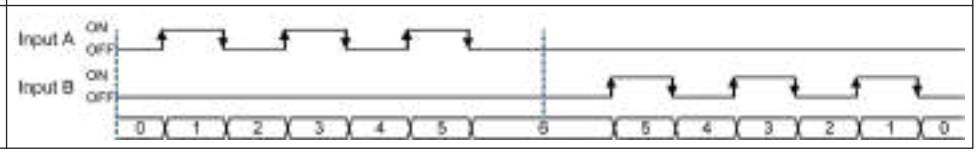
Method	Connected	Count
2-phase (Phase difference)		For 2-phase input, the input A signal and input B signal of each counter are connected to phase A and phase B, respectively, in the encoder. The count direction depends on the phase difference between phases A and B. When phase A is ahead of phase B by 90 degrees in terms of the electrical angle, the count value is incremented. When phase A is behind phase B by 90 degrees in terms of the electrical angle, the count value is decremented.
Individual		For individual input, the counter is incremented when the level of the input A signal rises or falls, and decremented when the level of the input B signal rises or falls.
Direction detection		For direction detection input, the count signal is connected to the input A signal. The count direction is controlled by the direction signal level of the input B signal. When the input B signal is OFF, the counter is incremented when the level of the input A signal rises or falls. When the input B signal is ON, the counter is decremented.

Count operation of 2-phase input (Phase difference input)

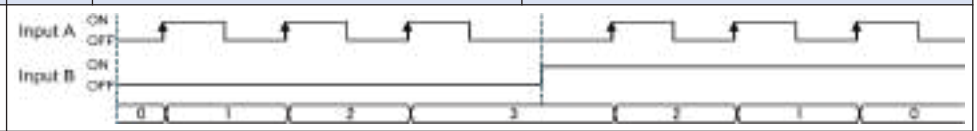
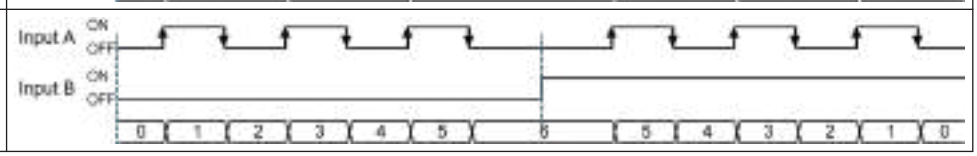
Multiplication	Timing chart	
	Incrementation	Decrementation
Multiply by 1		
Multiply by 2		

Multiplication	Timing chart	
	Incrementation	Decrementation
Multiply by 4		

Count operation of individual input

Multiplication	Timing chart	
	Incrementation	Decrementation
Multiply by 1		
Multiply by 2		

Count operation of direction detection input

Multiplication	Timing chart	
	Incrementation	Decrementation
Multiply by 1		
Multiply by 2		

■ Input time constant

Input time constants can be set for input signals (phases A, B, and Z) and control signals.

Input signal name	Settings
Input A signal, input B signal (The same time constant for both signals)	No input time constant / 0.1 µs (2 MHz) / 0.2 µs (1 MHz) / 0.5 µs (500 kHz) / 1.0 µs (250 kHz) / 2.0 µs (100 kHz) / 10.0 µs (10 kHz)
Input Z signal	No input time constant / 0.1 µs (2 MHz) / 0.2 µs (1 MHz) / 0.5 µs (500 kHz) / 1.0 µs (250 kHz) / 2.0 µs (100 kHz) / 10.0 µs (10 kHz)
Control signal	No input time constant / 2 µs / 5 µs / 10 µs / 20 µs / 50 µs / 100 µs / 500 µs / 1.0 ms / 2.0 ms / 5.0 ms / 10.0 ms

■ Types of count operation

Enable count operation

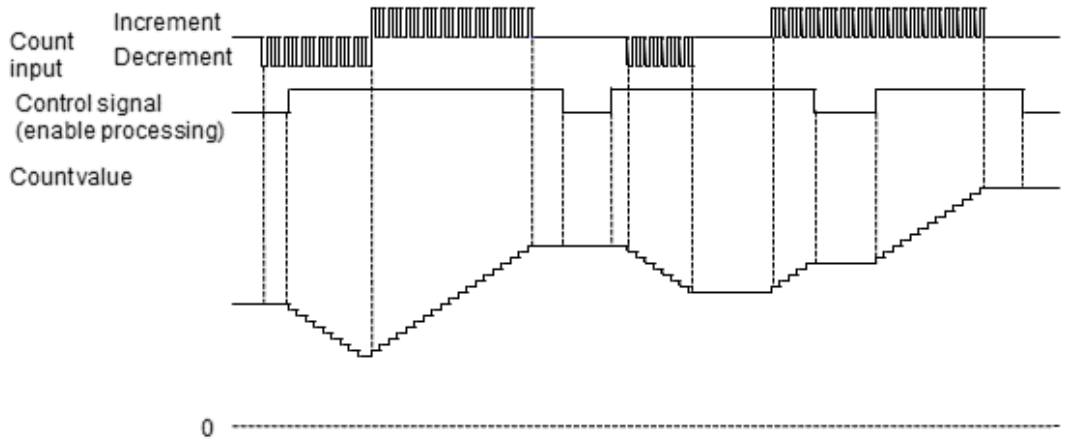
- Enable count operation is used by allocating the enable function to the control flag. Count operation is performed while the control flag is enabled.
- Enable count operation can be set using the methods shown in the following table.

Types of control flag (enable processing)

Signal	Setting method using the Counter_Configuration parameters	Enable condition	
		ON – OFF	ON – OFF –
Count enable request bit	- (No need to set)	●	
Control 0 signal	Setting the enable operation conditions by setting up the control 0 signal function	●	●
Control 1 signal	Setting the enable operation conditions by setting up the control 1 signal function	●	●

(Note 1) If you set the control 0 signal or control 1 signal as the enable operation condition, do not use the count enable request bit.

Count enable operation example

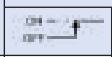
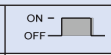
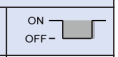


■ Reset count operation

- Reset count operation is used by allocating the reset function to the control flag.
- The count value is reset to 0 according to the change (rising, falling, positive logic, or negative logic) of the control flag.
- Reset count operation can be set using the methods shown in the following table.

Types of control flag (reset processing)

Signal	Setting method using the Counter_Configuration parameters	Reset condition			
				ON – OFF	ON – OFF –
Reset request bit	- (No need to set)	●			

Signal	Setting method using the Counter_Configuration parameters	Reset condition			
					
Input Z signal	Setting the reset operation conditions by setting up the input Z signal function	●	●	●	●
Comparison match status bit	Setting the "Comparison match rising edge reset" or "Comparison match falling edge reset" function to "Reset"	●	●		

(Note 1) If you set the control 0 signal or control 1 signal as the enable operation condition, the count enable request bit will be disabled.

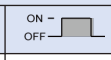
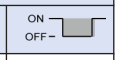
Info.

- To use the input Z signal and comparison match flag as reset signals, turn ON the reset enable request bit through user programs.

■ Preset count operation

- Preset count operation is used by allocating the preset function to the control flag.
- Preset count operation rewrites the count value as the preset value according to the change (rising, falling, positive logic, or negative logic) of the control flag.
- Preset count operation can also be used to start counter operation from the preset value.
- Preset count operation can be set using the methods shown in the following table.

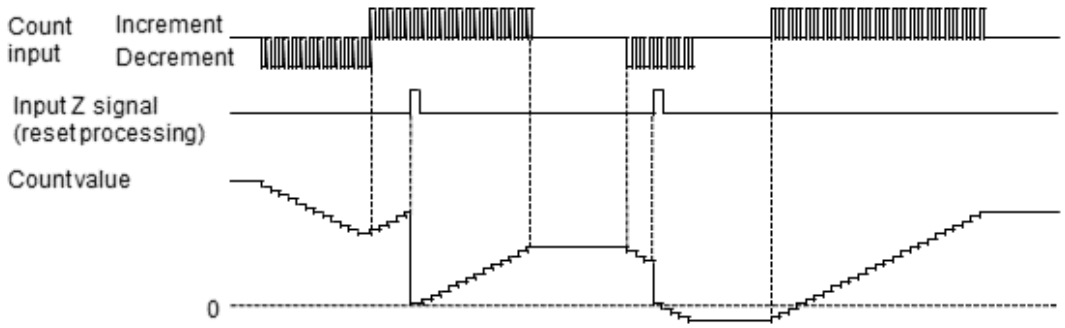
Types of control flag (preset processing)

Signal	Setting method using the Counter_Configuration parameters	Preset condition			
					
Preset request bit	- (No need to set)	●			
Input Z signal	Setting the preset operation conditions by setting up the input Z signal function	●	●	●	●

Info.

- To set a preset value, you must use a user program to set a temporary preset value and turn ON the preset value change request bit.

Reset (preset) count operation example



■ Enable reset count operation

- Enable reset count operation is used by allocating the enable function and reset function to the control flag.
- The count value is reset to zero when the counter becomes enabled due to the change of the control flag (enable reset processing).

Types of control flag (enable reset processing)

Signal	Setting method using the Counter_Configuration parameters	Reset condition		Enable condition	
				ON / OFF	ON / OFF
Control 0 signal	Setting the following conditions by setting up the control 0 signal function Positive logic enable operation and reset operation at rising edge Negative logic enable operation and reset operation at falling edge	•	•	•	•

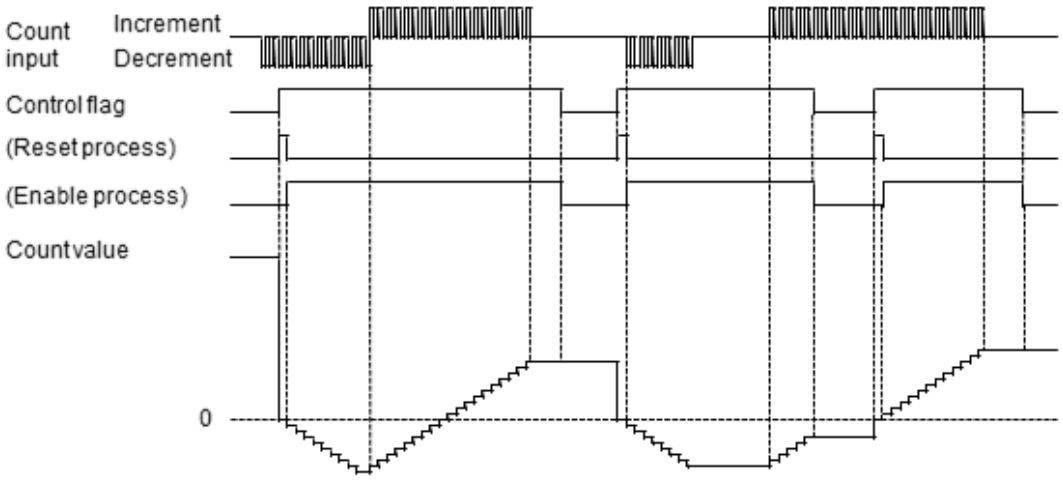
■ Enable preset count operation

- Enable preset count operation is used by allocating the enable function and preset function to the control flag.
- The count value is set as a preset value when the counter becomes enabled due to the change of the control flag (enable preset processing).

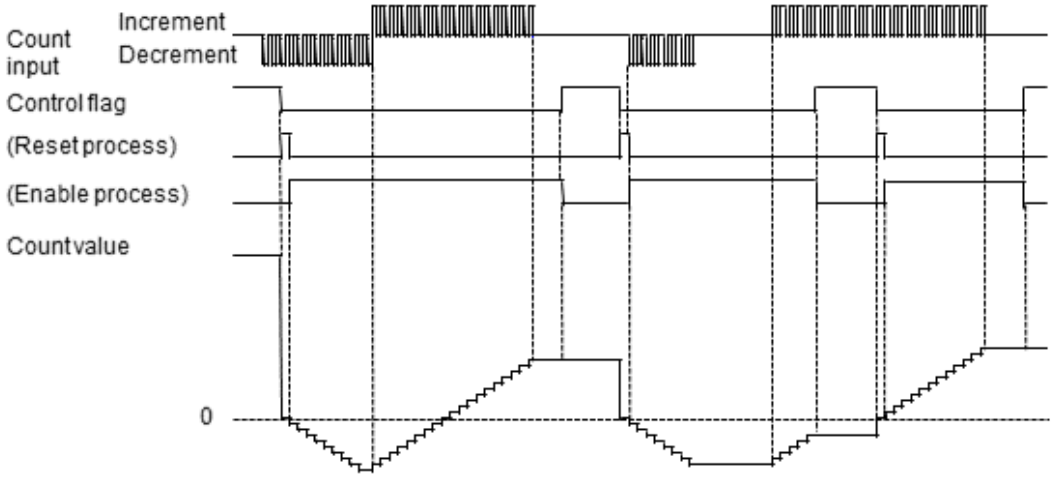
Types of control flag (enable preset processing)

Signal	Setting method using the Counter_Configuration parameters	Preset condition		Enable condition	
				ON / OFF	ON / OFF
Control 0 signal	Setting the following conditions by setting up the control 0 signal function Positive logic enable operation and preset operation at rising edge Negative logic enable operation and preset operation at falling edge	•	•	•	•

Positive logic enable operation and reset (preset) operation at rising edge

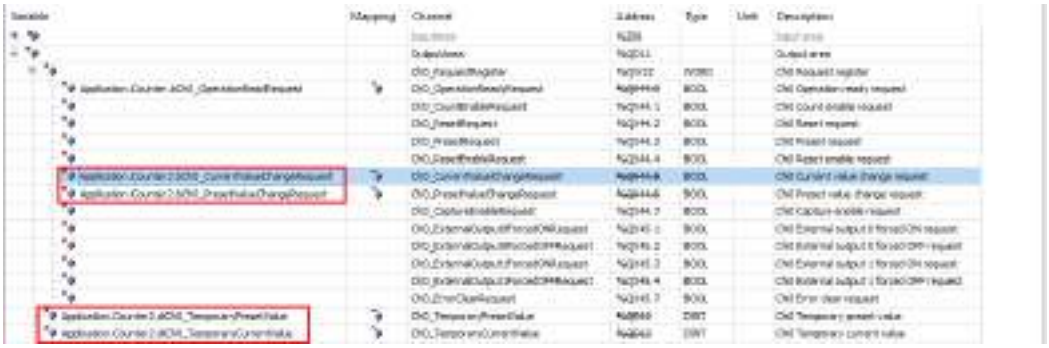


Negative logic enable operation and reset (preset) operation at falling edge



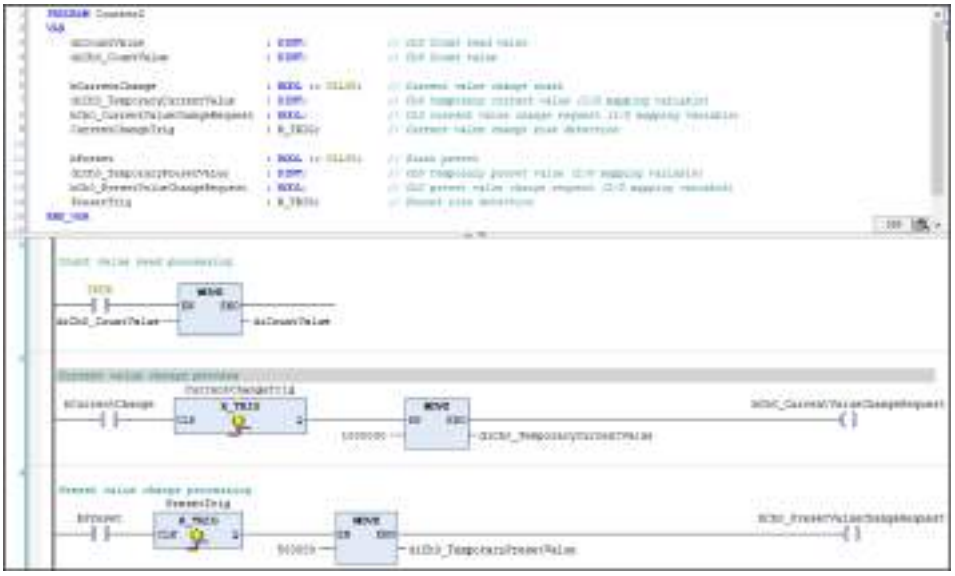
■ Reading the count value or changing the current count value or preset value

- To read and write channel data, variables are mapped to channels in the same way as in "12.6.5 Operation Ready Request". In this example, variables are mapped to channels used for reading the count value of Counter Ch0 and changing the current count value and preset value.



- The following are LD program and ST program examples for sample POU (Counter2).
  - In this example, count values are read for each scan.
  - When the current value change start bit is set to TRUE, the current value of Ch0 is set to 1000000.
  - When the start preset bit is set to TRUE, the preset value of Ch0 is set to 500000.

LD program



ST program

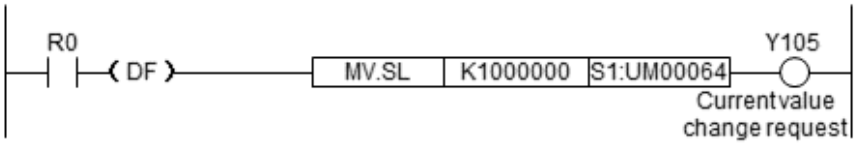
■ Count value when the power is turned on

- When the power is turned on, the count value is "0".

■ Changing the current count value

- The current count value can be changed to any value as necessary.
- Set a value in the temporary current value channel and turn ON the current value change request bit.

Example: A program to change the current value of CH0 to 1000000



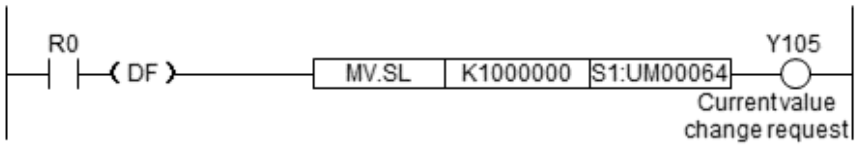
Info.

- Count values cannot be written directly to the count value channel (Ch*_CountValue).

■ Changing the preset value

- The preset value can be changed to any value as necessary.
- Set a value in the temporary preset value channel and turn ON the preset value change request bit.

Example: A program to change the preset value of CH0 to 1000000



Info.

- Preset values that are set cannot be read by programs.

12.6.7 Comparison Function

■ Setup procedure

1. From "Device view" in the navigator pane, double-click "Counter_Configuration".
2. Click the "Counter Parameters" tab.
3. For each counter, select the comparison function and set up each parameter.

Info.

- For details on how to set up parameters, refer to "12.6.2 Setting Parameters with GM Programmer".

■ Types of comparison function

This function compares the current value of the high-speed counter with a preset target value and, when these values match, it reflects the value in the comparison match flag.

- There are target value match comparison and band comparison.
- A total of 16 comparison data items can be set.
- Comparison results can be output externally.
- Comparison methods can be selected for each counter.

Comparison match function specifications

Item	Specifications
Set number of comparison data	Up to 16 data items for each counter (Comparison data 0 to comparison data 15)
Comparison match flag	Up to 16 flags for each counter (Comparison match 0 flag to comparison match 15 flag) Behaviors of 16 comparison match flags can be set for a single comparison data item.
Select comparison function	Target value match comparison Sets or resets the comparison match flag when the elapsed value matches the target value.
	Band comparison Turns ON or OFF the comparison match flag when the elapsed value falls within the range between the lower and upper limits that are set.
External output function	Up to two flags for each counter

Item	Specifications
	Comparison match 0 flag or comparison match 1 flag can be allocated to external output.
	ON hold time can be set only when the band comparison function is used. ON hold time: 0 to 1,000 ms

Info.

- Only the comparison match 0 flag or comparison match 1 flag can be set as the external output function.
- By default, the external output 0 and external output 1 signals are set to "Not output". When necessary, change the setting in the Counter_Configuration parameter window.
- There is no need to arrange comparison data items in ascending or descending order.

■ Target value match comparison and band comparison

The main differences are as below.

Main differences in characteristics

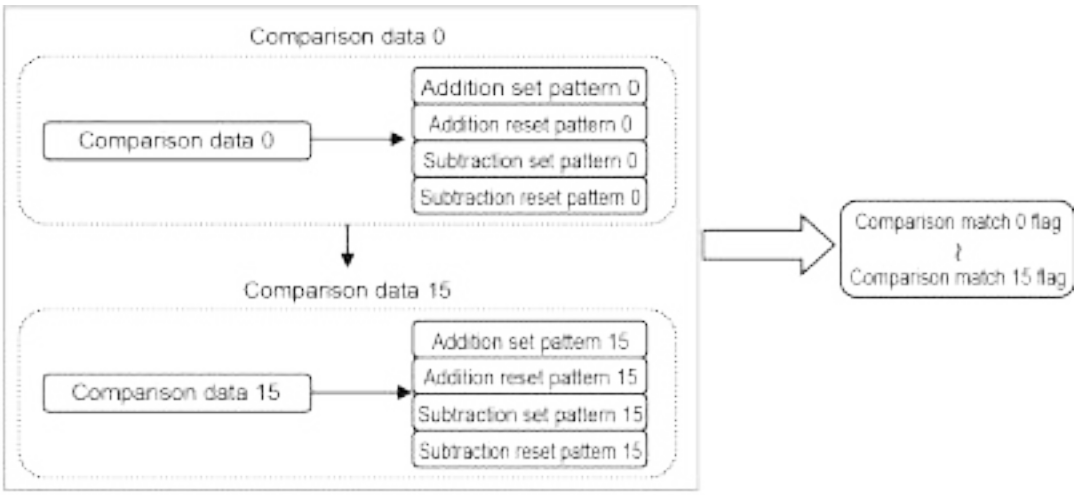
Item	Target value match comparison	Band comparison
Setting of comparison value data	Comparison value data is specified as a target value.	Comparison value data is specified as a band (lower and upper limits).
Setting for comparison value data match	One of the following four options is specified as the comparison match flag behavior to be performed when the target value is reached. "Addition set pattern": Sets the flag when the current value and comparison value match at the time of incrementation "Addition reset pattern": Resets the flag when the current value and comparison value match at the time of incrementation "Subtraction set pattern": Sets the flag when the current value and comparison value match at the time of decrementation "Subtraction reset pattern": Resets the flag when the current value and comparison value match at the time of decrementation	ON or OFF is specified as the behavior of the comparison match flag when the current value falls within the specified band.
Behavior when comparison value data matches	The behavior that is performed when the current value matches the same comparison value data may differ between incremental count and decremental count, depending on the settings.	The behavior that is performed when the current value matches the same comparison value data is the same for incremental count and decremental count.
External output signal ON hold time	Cannot be set	ON hold time: 0 to 1,000 ms

■ Parameter settings for target value match comparison

Parameter setting procedure

1. Click **Counter parameter**>**Counter (Ch0 or Ch1)**>**Comparison function**, change the "Select comparison function" parameter to "Target value match comparison", and execute "Set number of comparison data".

2. Click **Counter parameter**>**Counter (Ch0 or Ch1)**>**Comparison function**>**Comparison data** and specify target values for each comparison data item.
3. Select "Addition set pattern", "Addition reset pattern", "Subtraction set pattern", and "Subtraction reset pattern" separately and set ""No change", "Set output", or "Reset output"" for each comparison match flag.
4. Configure these settings for each comparison data item.



Info.

- Settings can be configured individually according to the count direction (incremental or decremental direction) at the time of comparison data match.
- A total of 16 comparison match flags can be set separately for "Addition set pattern", "Addition reset pattern", "Subtraction set pattern", and "Subtraction reset pattern".
- For details on how to set up parameters, refer to "12.6.2 Setting Parameters with GM Programmer".

■ Setting example for target value match function

Output setting example

Comparison data	Target value	Output setting	Comparison match 0 flag	Comparison match 1 flag	Comparison match 2 flag	Comparison match 3 flag
0	+500	Addition set pattern		○		
		Addition reset pattern	○			
		Subtraction set pattern				
		Subtraction reset pattern		○	○	○
1	+1,250	Addition set pattern			○	
		Addition reset pattern		○		○
		Subtraction set pattern				
		Subtraction reset pattern				
2	+2,500	Addition set pattern				○

Comparison data	Target value	Output setting	Comparison match 0 flag	Comparison match 1 flag	Comparison match 2 flag	Comparison match 3 flag
		Addition reset pattern				
		Subtraction set pattern		○		○
		Subtraction reset pattern				
3	+3,750	Addition set pattern		○		
		Addition reset pattern			○	
		Subtraction set pattern				
		Subtraction reset pattern				○
4	+5,000	Addition set pattern				
		Addition reset pattern				○
		Subtraction set pattern			○	○
		Subtraction reset pattern	○	○		
5	+6,250	Addition set pattern			○	
		Addition reset pattern				
		Subtraction set pattern				
		Subtraction reset pattern				○
6	+7,500	Addition set pattern	○			○
		Addition reset pattern		○		
		Subtraction set pattern	○	○		
		Subtraction reset pattern				
7	+8,750	Addition set pattern				
		Addition reset pattern			○	
		Subtraction set pattern				○
		Subtraction reset pattern				

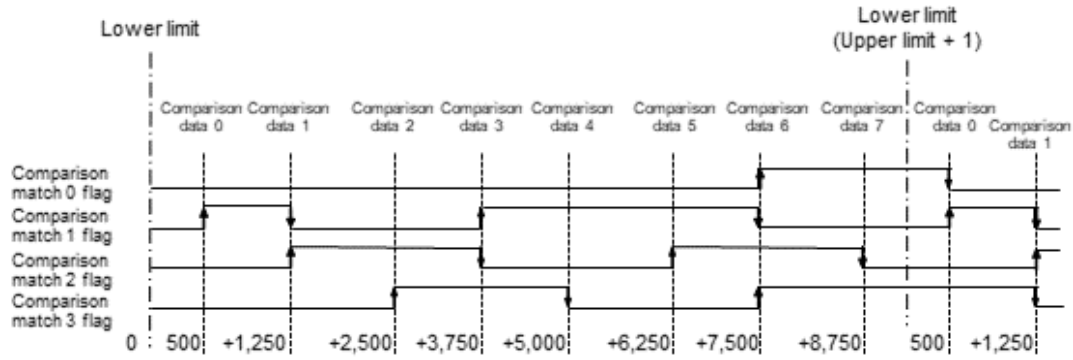
Info.

- The behavior of the comparison match flag that is performed when the count value reaches the target value can be changed separately for incrementation and decrementation.
- If the contents of comparison data 0 to 15 are duplicated and reset conditions are different, comparison data is prioritized in the following order.

  (High) 0 > 1 > 2 > 3 > 4 > 5 > 6 > 7 > 8 > 9 > 10 > 11 > 12 > 13 > 14 > 15 (Low)

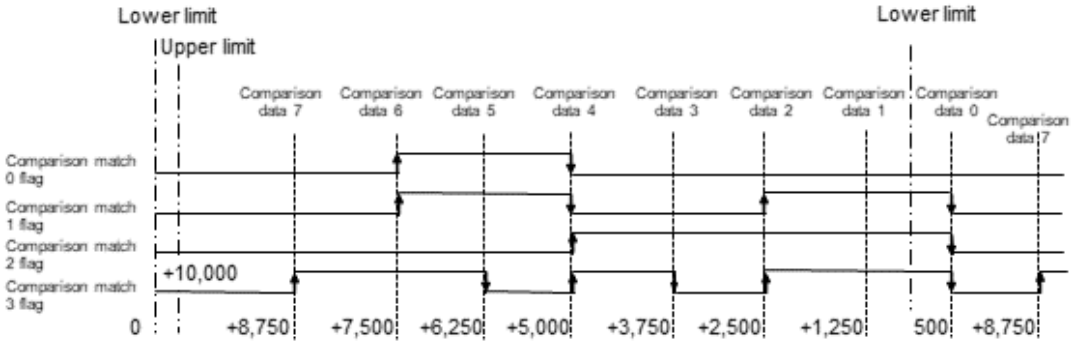
Behavior of comparison match flag during incremental count

When the current value matches the comparison data, the comparison match flag behaves according to the setting of "Addition set pattern" or "Addition reset pattern".



Behavior of comparison match flag during decrementation count

When the current value matches the comparison data, the comparison match flag behaves according to the setting of "Subtraction set pattern" or "Subtraction reset pattern".



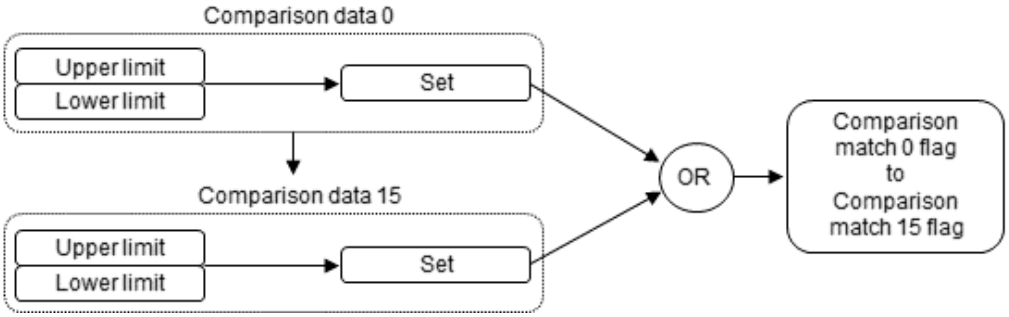
Info.

- For ring counters, comparison data can be set in an area including the lower and upper limits where count values are rolled over.

■ Parameter settings for band comparison

Parameter setting procedure

1. Click **Counter parameter**>**Counter (Ch0 or Ch1)**>**Comparison function**, change the "Select comparison function" parameter to "Band comparison", and execute "Set number of comparison data".
2. Click **Counter parameters**>**Counter (Ch0 or Ch1)**>**Comparison function**>**Comparison data** and specify upper and lower limits for each comparison data item.
3. Set whether to turn ON or OFF the comparison match flag when the count value exists in the zone (between the upper and lower limits).
4. Configure these settings for each comparison data item.



Info.

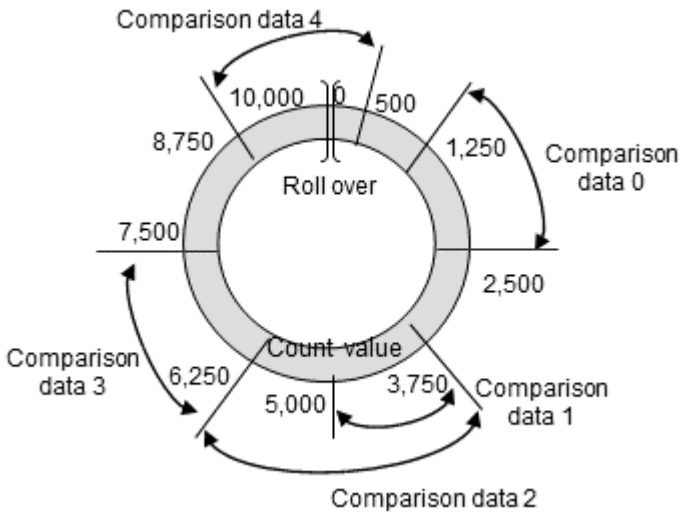
- A total of 16 comparison match flags can be set separately for each comparison data item.
- Multiple comparison data items can be set for the same band.
- For details on how to set up parameters, refer to "12.6.2 Setting Parameters with GM Programmer".

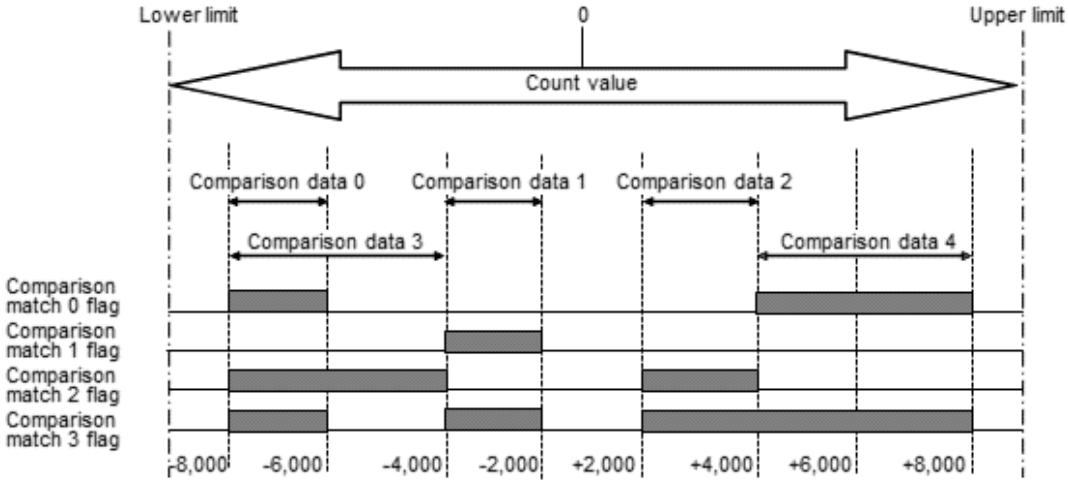
■ Setting example for band comparison

Output setting example

Comparison data	Band comparison value		Output setting	Comparison match 0 flag	Comparison match 1 flag	Comparison match 2 flag	Comparison match 3 flag
	Lower limit	Upper limit					
0	+1,250	2,500	Set	ON	OFF	OFF	OFF
1	+3,750	+5,000	Set	OFF	ON	OFF	ON
2	+3,750	+7,500	Set	OFF	OFF	OFF	ON
3	+6,250	+7,500	Set	ON	ON	ON	ON
4	+8,750	+500	Set	OFF	OFF	ON	OFF

Behavior of comparison match flag during band comparison





Info.

- For ring counters, comparison data can be set in an area including the lower and upper limits where count values are rolled over.

■ Parameter settings for external output function

- The comparison match 0 flag and comparison match 1 flag can be output externally using parameter settings.
- The Counter_Configuration parameter window is used to allocate the comparison match 0 flag and comparison match 1 flag to the external output function.

Info.

- For details on how to set up parameters, refer to "12.6.2 Setting Parameters with GM Programmer".

12.6.8 External Output Function

■ Overview of external output function

The comparison match 0 flag and comparison match 1 flag can be output externally using parameter settings.

Setup procedure

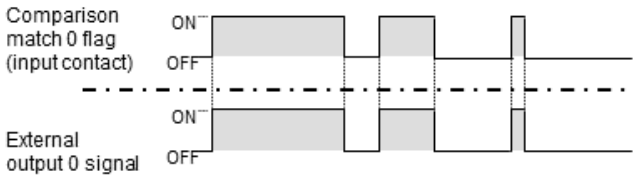
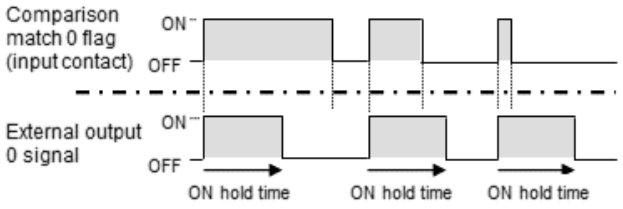
1. From "Device view" in the navigator pane, double-click "Counter_Configuration".
2. Click the "Counter parameter" tab.
3. For each counter, select "External output function "and set up each parameter.

Counter_Configuration parameter setting example

■ ON hold time (for band comparison only)

When the band comparison function is used, ON hold time can be set as an output signal.

Differences in behavior between settings

ON hold time	Timing chart for comparison match flag and external output signal
0	
1 to 1,000 ms	

■ Forced output function

- If the Ch* external output 0* forced ON / OFF request bit is used, the external output 0 signal and external output 1 signal can be turned ON or OFF through user programs.
- The forced output function can be used to check wiring and for other purposes.

12.6.9 Capture Function

■ Setup procedure

1. From "Device view" in the navigator pane, double-click "Counter_Configuration".
2. Click the "Counter Parameters" tab.
3. For each counter, select "Capture function" and set up each parameter.

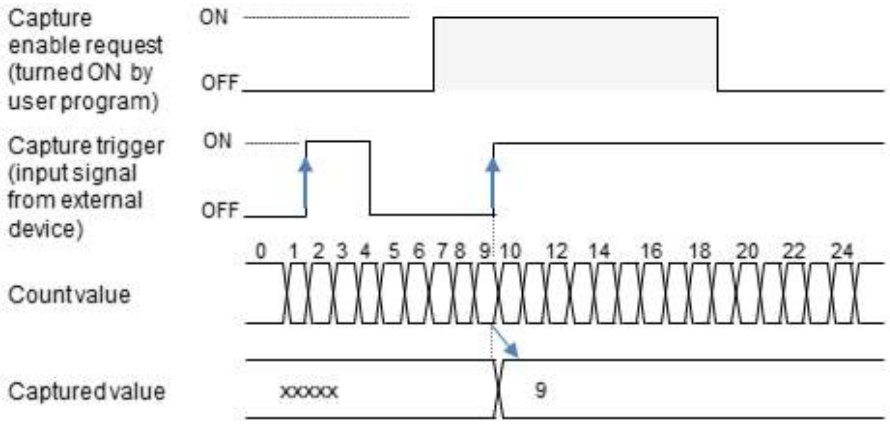
Info.

- For details on how to set up parameters, refer to "12.6.2 Setting Parameters with GM Programmer".

■ Types of capture function

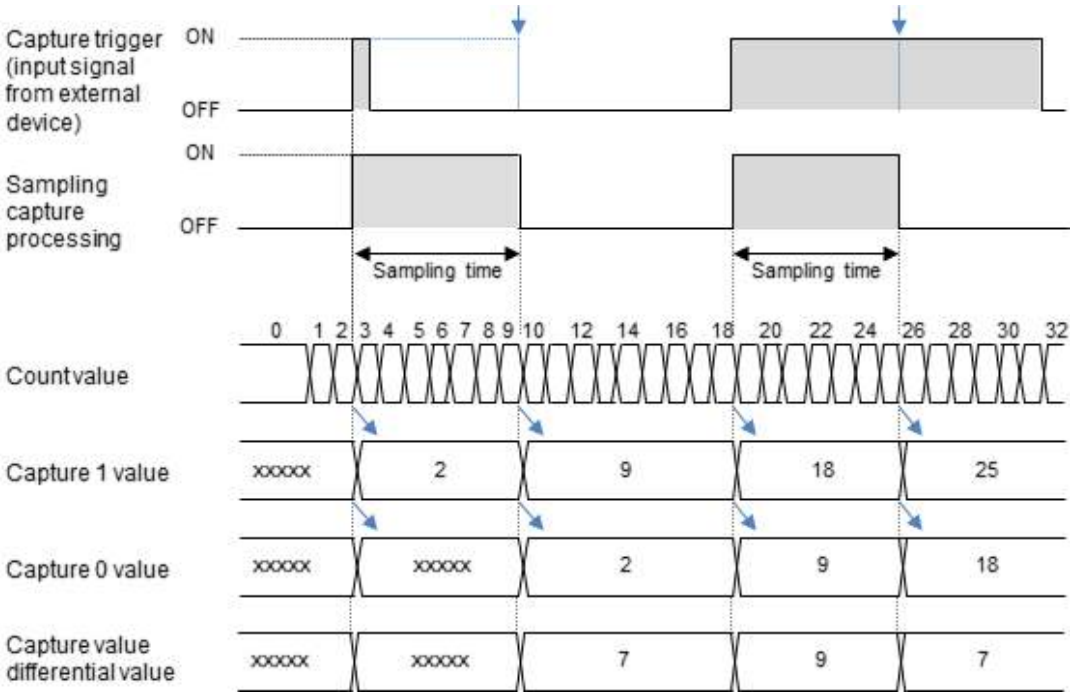
Capture function

- The count value at the point in time when the input signal from an external device changes is stored in the capture 0 value or capture 1 value register.



Sampling capture function

- The count value when the specified sampling time elapses after the input signal from an external device changes is stored in the capture 0 value or capture 1 value registers.
- The count value equivalent to the sampling time can be monitored by reading the capture differential value.



■ Comparison between capture function and sampling capture function

Available conditions differ between the functions.

Comparison between both functions

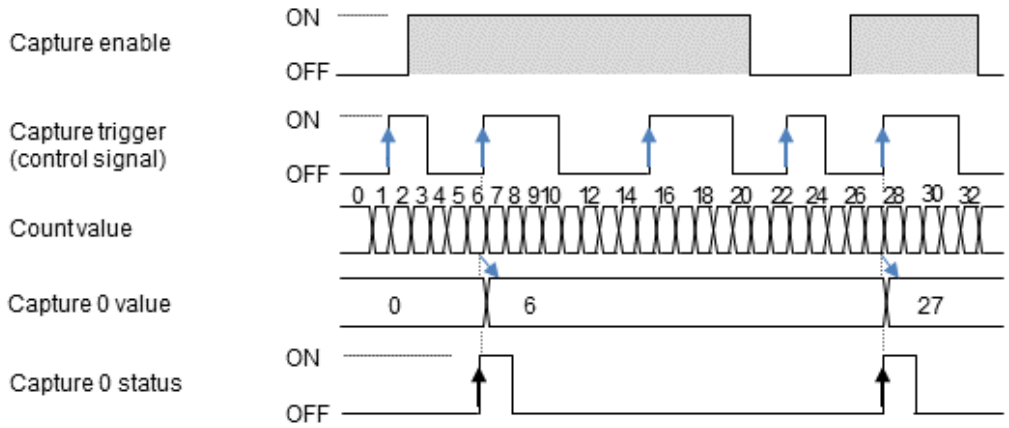
Item		Capture function	Sampling capture function
Number of points that can be used		Max. 2 points	Max. 1 point
		The capture function cannot be used when the sampling capture function is used.	
Registers used		Capture 0 value: 1 register (2 words) Capture 1 value: 1 register (2 words) These registers can be used individually.	Capture 0 value: 1 register (2 words) Capture 1 value: 1 register (2 words) These registers are used simultaneously.
		Capture differential value: 1 register (2 words)	
Enable condition		The capture function is enabled while the capture enable request bit is ON.	Always enabled
Capture flags		The Counter_Configuration parameter window is used to allocate capture flags.	
	Control 0 signal	Used as a trigger for capture 0 or capture 1.	Used as a trigger for the sampling capture function
	Control 1 signal	Used as a trigger for capture 0 or capture 1.	Not use
	Validity condition	Activated when either of the following conditions is met. At rising edge of control 0 / 1 signal At falling edge of control 0 / 1 signal	Activated when either of the following conditions is met. Control 0 signal (positive logic) Control 0 signal (negative logic)

Item		Capture function	Sampling capture function
		By allocating one of the control signals to the same capture number, the capture function can be allocated as the rising or falling edge of the signal.	Capture enable request bit (positive logic)
Clearing the capture * status bit		The capture * status bit is cleared automatically each time I/O refresh occurs.	

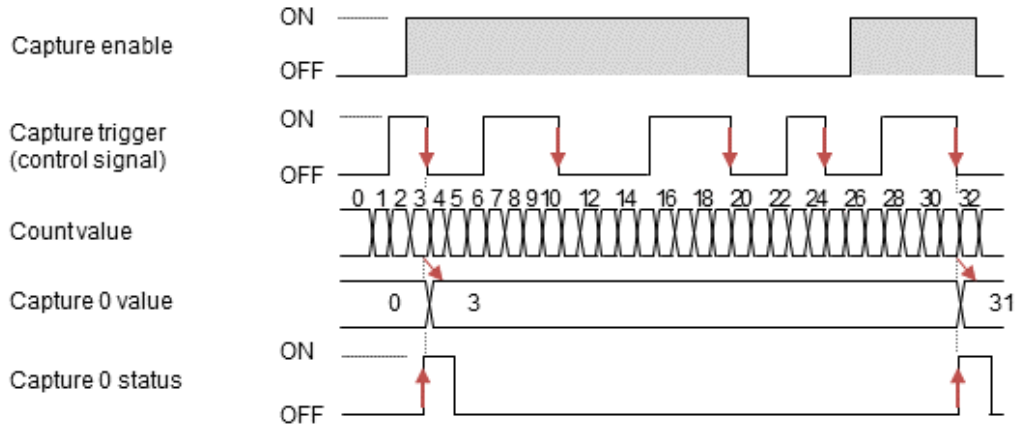
■ One operation

- When the capture enable request bit enables the capture function, capture is executed when the first capture flag becomes enabled.
- The behaviors differ according to the validity condition (rising edge or falling edge) of the capture flag to be enabled, as below.

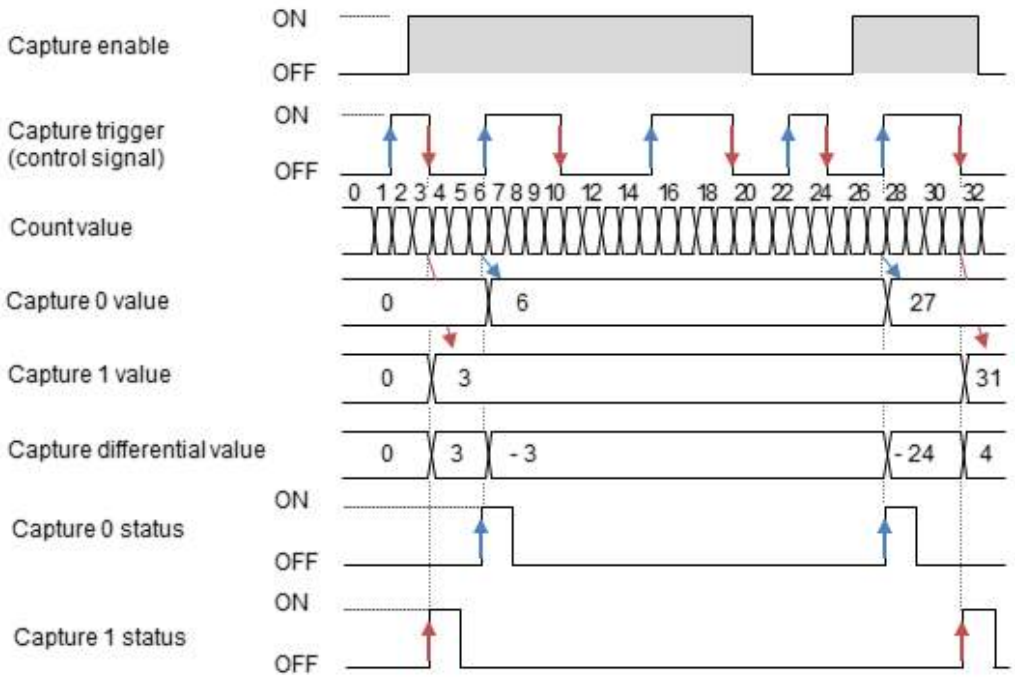
When "rising edge" is specified as the capture trigger condition



When "falling edge" is specified as the capture trigger condition



When "rising edge" and "falling edge" of the same signal are specified as the capture trigger conditions



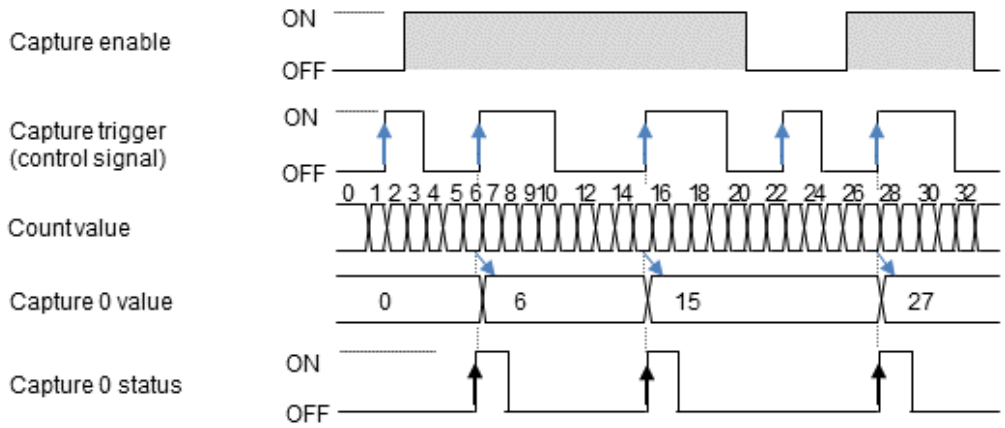
Info.

- The sign of the capture differential value changes according to the sequence of the capture enable request bit and capture flag.

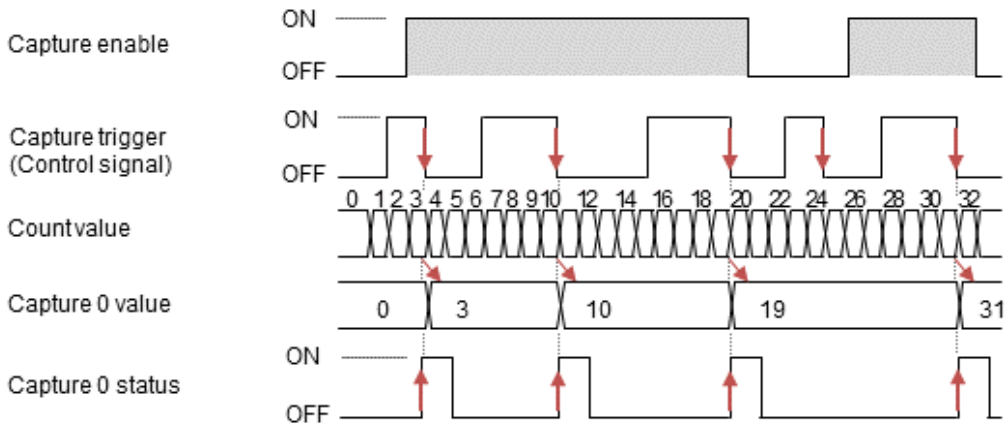
■ Continuous operation

- When the capture enable request bit enables the capture function, capture is executed every time a capture flag becomes enabled.
- The behaviors differ according to the validity condition (rising edge or falling edge) of the capture flag to be enabled, as below.

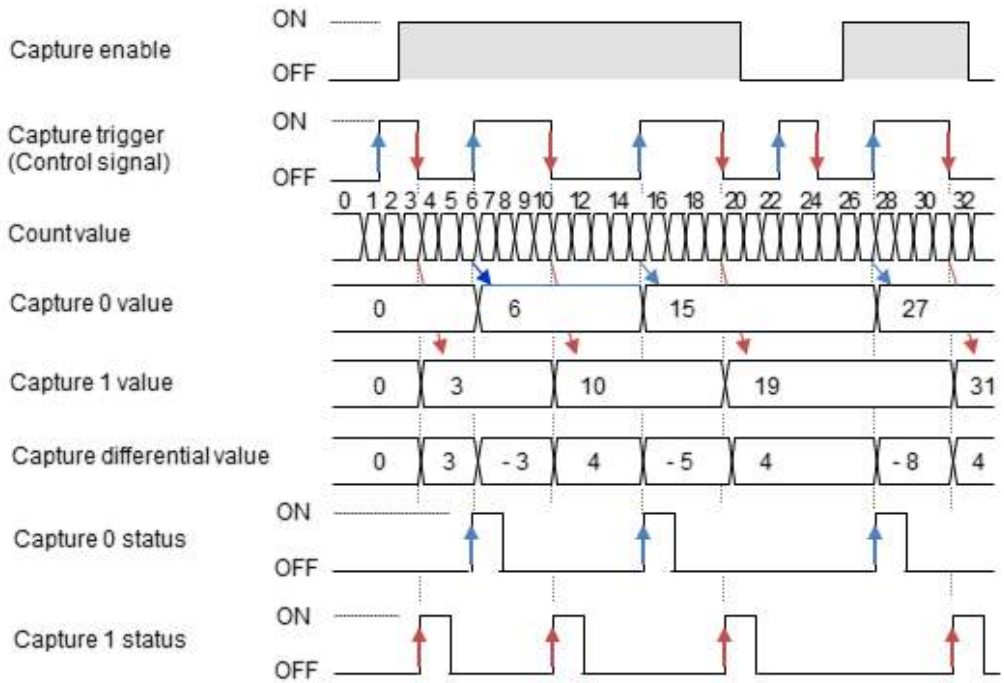
When "rising edge" is specified as the capture trigger condition



When "falling edge" is specified as the capture trigger condition



When "rising edge" and "falling edge" of the same signal are specified as the capture trigger conditions



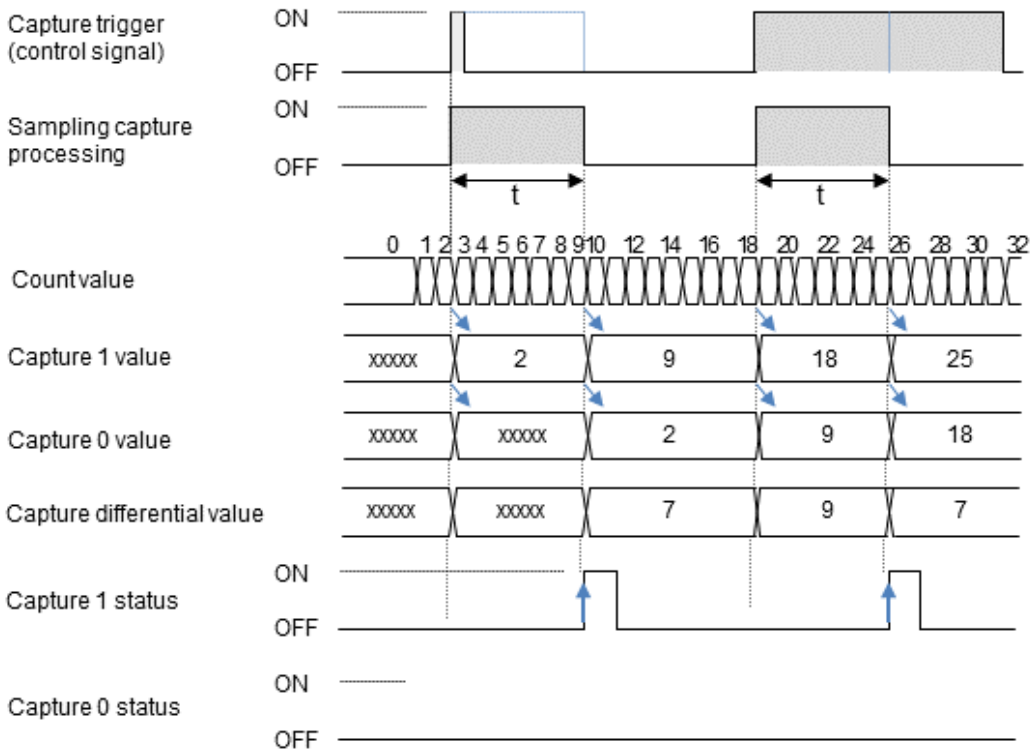
Info.

- The sign of the capture differential value changes according to the sequence of the capture enable request bit and capture flag.
- For continuous operation, capture 0 value, capture 1 value, and capture differential value are overwritten each time a capture operation is completed.

■ Sampling capture function (one operation)

- When the specified sampling time elapses after the capture flag turns ON or OFF, the count value is stored in the capture 0 value and capture 1 value registers and the differential value is stored in the capture differential value register.
- The sampling capture function is always executable when the control 0 signal is allocated to the sampling capture function.
- The trigger condition that starts sampling capture can be selected from control 0 signal (positive logic), control 1 signal (negative logic), and capture enable request bit (positive logic).
- For one operation, the capture 1 status bit is activated. Note that the capture 0 status bit is not activated.

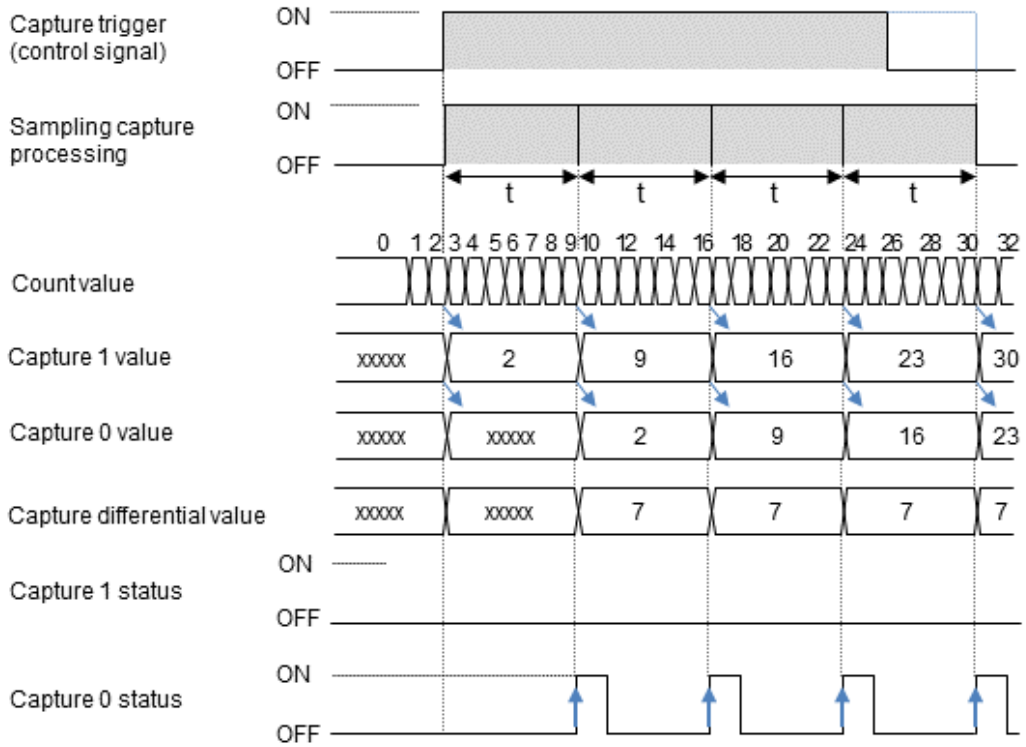
Timing chart



■ Sampling capture function (continuous operation)

- Each time the specified sampling time elapses after the capture flag turns ON or OFF, successively, the count value is stored in the capture 0 value and capture 1 value registers and the differential value is stored in the capture differential value register.
- The sampling capture function is always executable when the control 0 signal is allocated to the sampling capture function.
- The trigger condition that starts sampling capture can be selected from control 0 signal (positive logic), control 0 signal (negative logic), and capture enable request bit (positive logic).
- For continuous operation, the capture 0 status bit is activated. Note that the capture 1 status bit is not activated.

Timing chart



■ Reading captured data

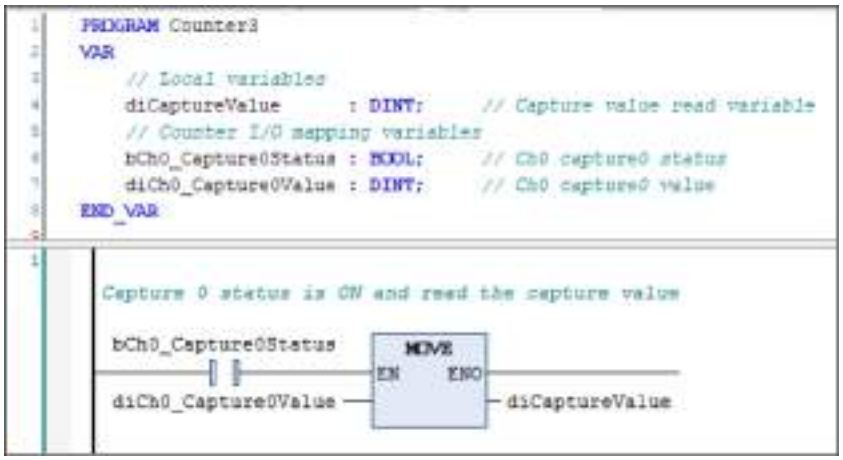
Areas where captured data is stored

- The latest captured data is stored in the capture 0 value and capture 1 value registers.
- Captured data is stored as signed 32-bit data (-2,147,483,648 to 2,147,483,647).

Sample program

To read and write channel data, variables are mapped to channels in the same way as in "12.6.5 Operation Ready Request". In this example, variables are mapped to the channel used for reading captured data for Counter Ch0 (the name of sample POU is created as "Counter3").

LD program

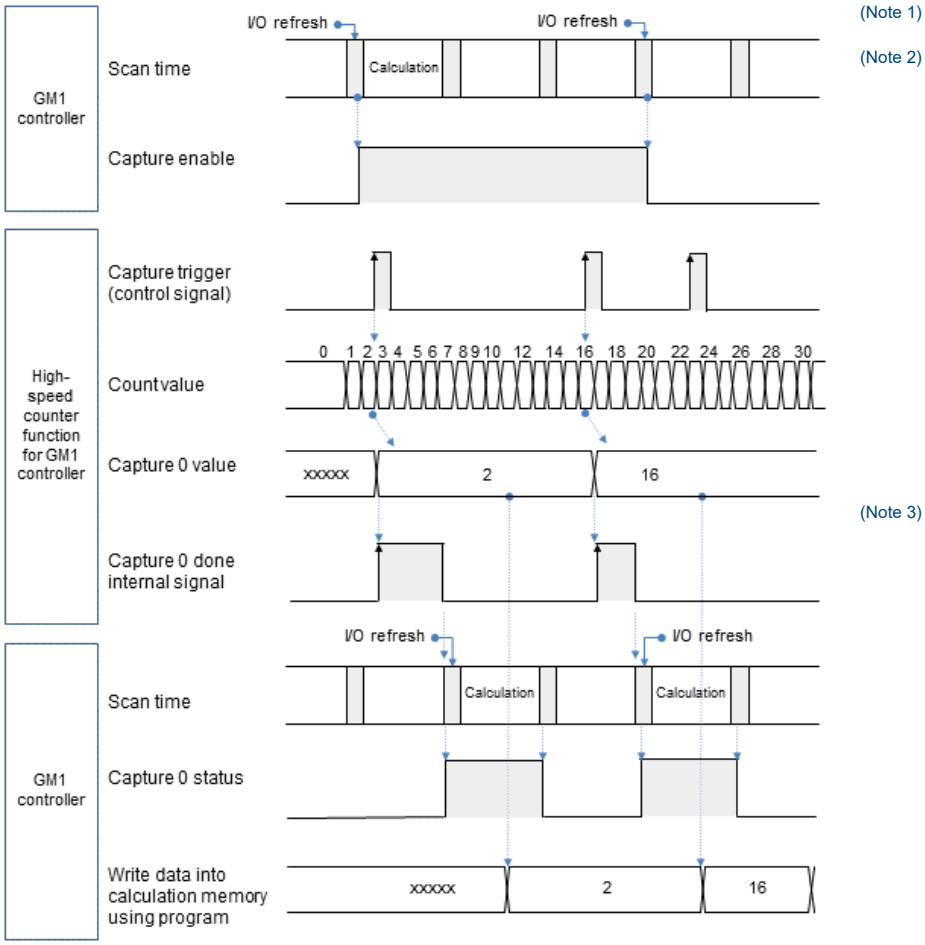


ST program

■ Behaviors and read operations of capture * status bits

Basic behaviors

- When capture is completed, it is reflected in the capture * status bit for each I/O refresh.
- For the capture function, the capture * status bit is reset when the capture enable request bit turns ON.

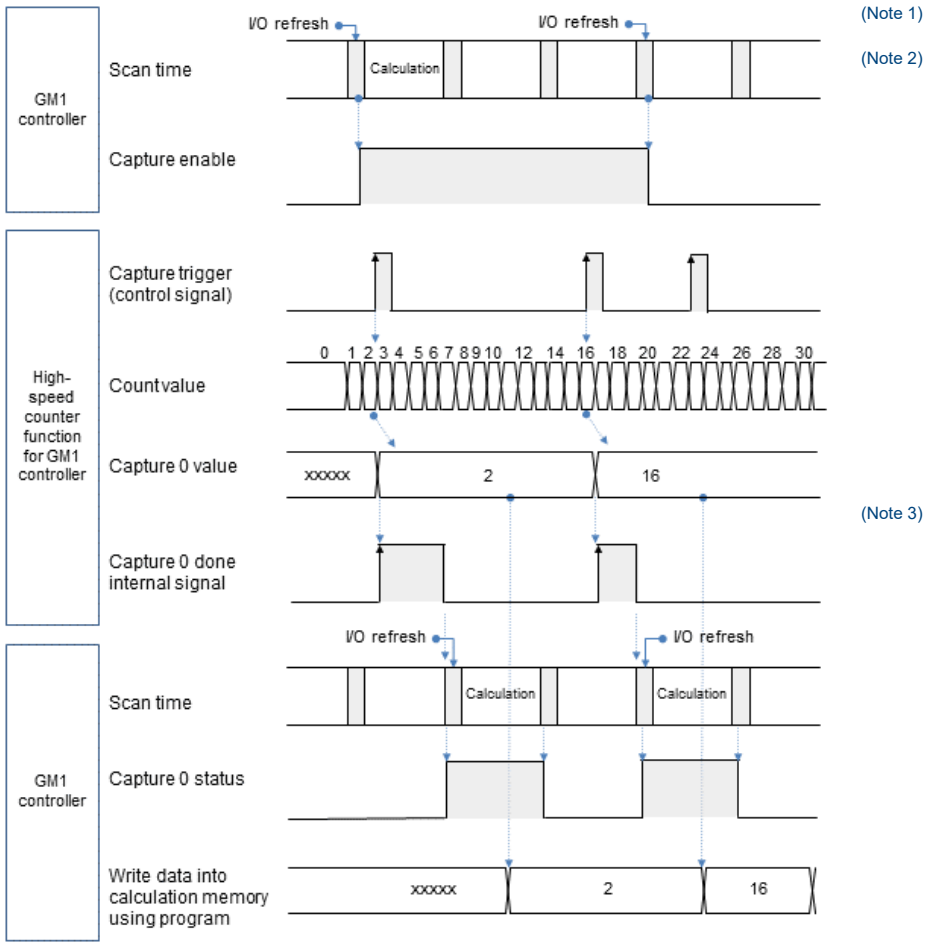


(Note 1) The capture enable request bit is turned ON by a user program.

(Note 2) Each time a capture flag turns ON, a capture operation is performed asynchronously with user program execution. The capture completion internal flag used by the system is reset each time an I/O refresh is performed.

(Note 3) Capture * status bits are reflected each time an I/O refresh is performed. Capture * status bits are used to read capture 0 value, capture 1 value, and capture differential value as arbitrary variables through user programs. These values are read at the time of relevant calculation processing.

Processing when capture trigger input occurs frequently



(Note 1) The capture enable request bit is turned ON by a user program.

(Note 2) Each time a capture flag turns ON, a capture operation is performed asynchronously with user program execution. The capture completion internal flag used by the system is reset each time an I/O refresh is performed.

(Note 3) Capture * status bits are reflected each time an I/O refresh is performed. If multiple capture operations are performed continuously, the capture 0 completion status bit or capture 1 completion status bit will remain ON.

Info.

- If control signals used as capture triggers are input frequently, the capture 0 completion status bit or capture 1 completion status bit will remain ON. Take care when reading multiple captured data items.

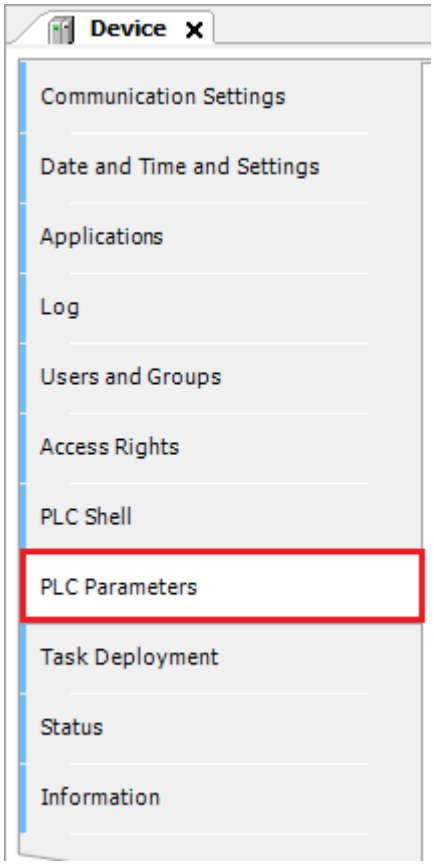
12.6.10 Unit Error

Overflow and underflow errors with the counter unit are operation stop errors.
To continue the operating status when an error occurs, change the value of PLC parameter "A unit error occurred" to "Continue operation".

Procedure

1. From "Device view" in the navigator pane, double-click the "Device" object.

2. Click the "PLC parameters" tab in the Device window.

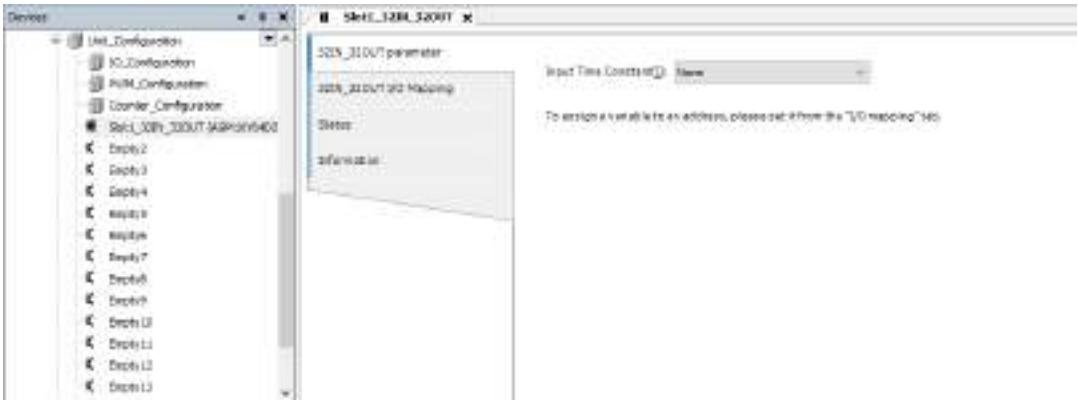


3. Change the value of "A unit error occurred" to "Continue operation".

12.7 Settings of I/O Unit

Expansion I/O units are classified into 64-point input, 64-point output, and 32-point I/O units. This section explains 32-point I/O units as an example.

Setting Parameters with GM Programmer



12.7.1 Parameter Settings

■ **Parameter**

Setting item	Settings	Default value	Description
Input time constant	None 0.1 ms 0.5 ms 1 ms 5 ms 10 ms 20 ms 70 ms	None	Input time constant

12.7.2 I/O Mapping for I/O Unit

Channel	Type	Description	Remarks
Ch0_In	WORD	Ch0_In	
Ch1_In	WORD	Ch1_In	
Ch0_Out	WORD	Ch0_Out	
Ch1_Out	WORD	Ch1_Out	

■ **Ch*_In**

* represents 0 or 1.

Channel	Type	Description	Remarks
Ch*_In00	BOOL	Ch*_In00	
Ch*_In01	BOOL	Ch*_In01	
Ch*_In02	BOOL	Ch*_In02	
Ch*_In03	BOOL	Ch*_In03	
Ch*_In04	BOOL	Ch*_In04	
Ch*_In05	BOOL	Ch*_In05	
Ch*_In06	BOOL	Ch*_In06	
Ch*_In07	BOOL	Ch*_In07	
Ch*_In08	BOOL	Ch*_In08	
Ch*_In09	BOOL	Ch*_In09	
Ch*_In10	BOOL	Ch*_In10	
Ch*_In11	BOOL	Ch*_In11	
Ch*_In12	BOOL	Ch*_In12	
Ch*_In13	BOOL	Ch*_In13	
Ch*_In14	BOOL	Ch*_In14	
Ch*_In15	BOOL	Ch*_In15	

■ Ch*_Out

* represents 0 or 1.

Channel	Type	Description	Remarks
Ch*_Out00	BOOL	Ch*_Out00	
Ch*_Out01	BOOL	Ch*_Out01	
Ch*_Out02	BOOL	Ch*_Out02	
Ch*_Out03	BOOL	Ch*_Out03	
Ch*_Out04	BOOL	Ch*_Out04	
Ch*_Out05	BOOL	Ch*_Out05	
Ch*_Out06	BOOL	Ch*_Out06	
Ch*_Out07	BOOL	Ch*_Out07	
Ch*_Out08	BOOL	Ch*_Out08	
Ch*_Out09	BOOL	Ch*_Out09	
Ch*_Out10	BOOL	Ch*_Out10	
Ch*_Out11	BOOL	Ch*_Out11	
Ch*_Out12	BOOL	Ch*_Out12	
Ch*_Out13	BOOL	Ch*_Out13	
Ch*_Out14	BOOL	Ch*_Out14	
Ch*_Out15	BOOL	Ch*_Out15	

(MEMO)

13 Communication Function

13.1 Overview of Communication Function 13-2
13.1.1 Adding Network Communication Devices 13-2
13.1.2 Adding Serial Communication Devices 13-3
13.2 General-purpose Communication 13-6
13.2.1 General-purpose Communication (Ethernet) 13-6
13.2.2 General-purpose Communication (Serial) 13-20
13.3 MODBUS 13-24
13.3.1 What is Modbus TCP? 13-24
13.3.2 Modbus-TCP Master Communication 13-24
13.3.3 Modbus-TCP Slave Communication 13-28
13.3.4 Modbus-RTU Master Communication 13-30
13.3.5 Modbus-RTU Slave Communication 13-38
13.4 EtherNet/IP 13-42
13.4.1 What is EtherNet/IP? 13-42
13.4.2 Cyclic Communication Function 13-42
13.4.3 EtherNet/IP Scanner Function 13-42
13.4.4 Setting up the EtherNet/IP Scanner Function 13-42
13.4.5 EtherNet/IP Scanner Operation 13-49
13.4.6 EtherNet/IP Adapter Function 13-51
13.4.7 Setting up the EtherNet/IP Adapter Function 13-51
13.4.8 EtherNet/IP Adapter Operation 13-55

13.1 Overview of Communication Function

The GM1 controller allows general-purpose communication, Modbus communication, and EtherNet/IP communication with external devices via COM and LAN ports.

The following table shows the functions supported by each port.

Port	Supported protocol	Maximum number of connections
COM port	General-purpose Communication MODBUS-RTU (Master / Slave)	1 port
LAN port 1	General-purpose Communication MODBUS-TCP (Master / Slave)	16 connections
LAN port 2	General-purpose Communication MODBUS-TCP (Master / Slave) EtherNet/IP (Scanner / Adapter)	32 connections

Info.

- The IP address of the LAN port can be changed using the "PLC Parameters" tab in the Device setting window.
  For details, refer to "5.2 Setting up the GM1 Controller".

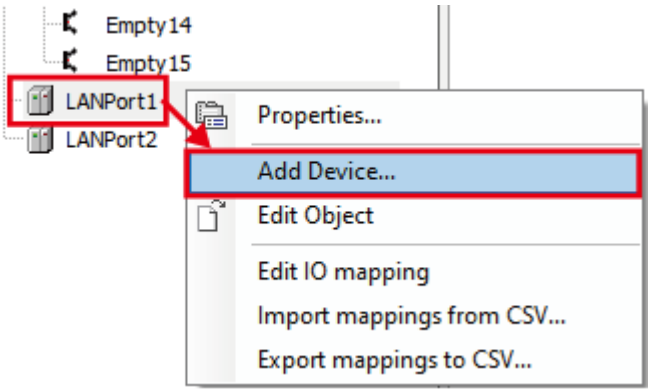
13.1.1 Adding Network Communication Devices

Add a communication device object to the device object of a LAN port.

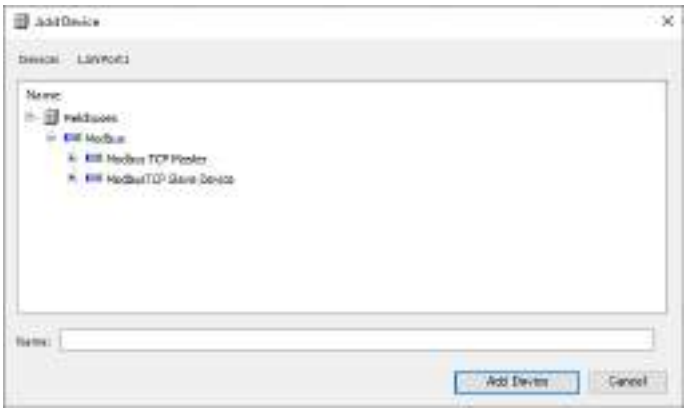
For example, use the following procedure to add "ModbusTCP Slave Device" to LAN port 1.

Procedure

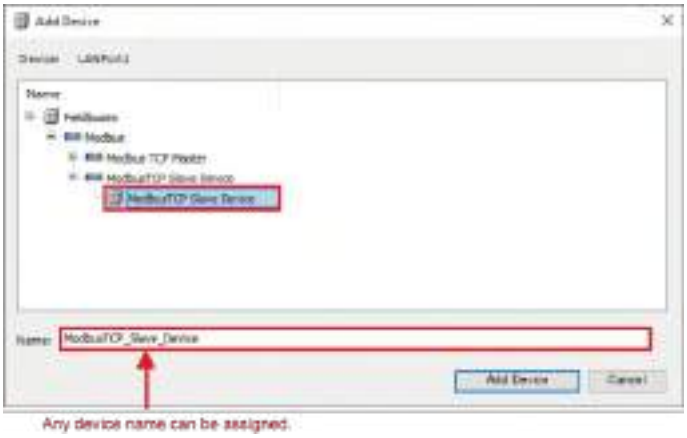
1. Right-click the "LANPort1" object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.



The "Add Device" dialog box will be displayed.



2. Select "ModbusTCP Slave Device".



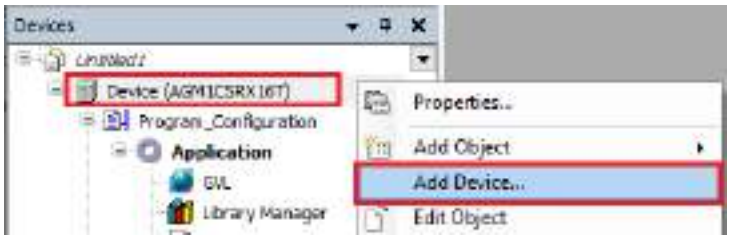
3. Click the [Add Device] button.
   The selected "ModbusTCP_Slave_Device" object will be added to the navigator pane.

For details on how to set up Modbus TCP, refer to "13.3 MODBUS".

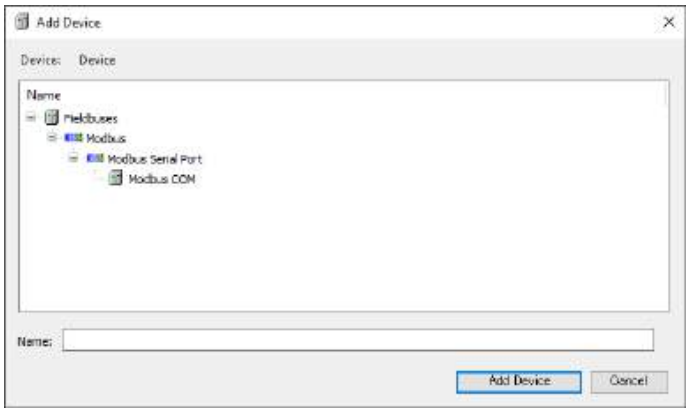
13.1.2 Adding Serial Communication Devices

Procedure

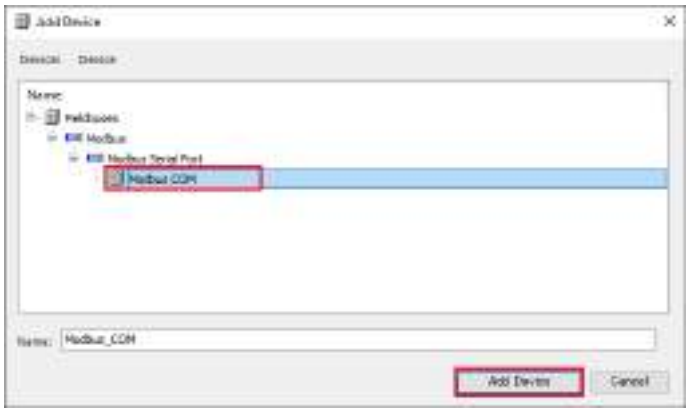
1. Right-click the "Device" object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.



The "Add Device" dialog box will be displayed.



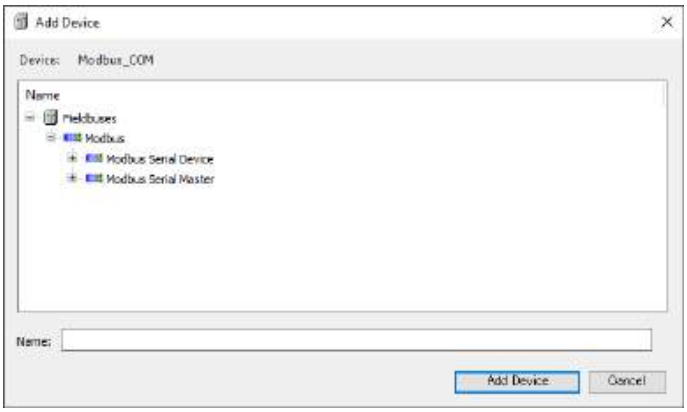
2. Select "Modbus COM".



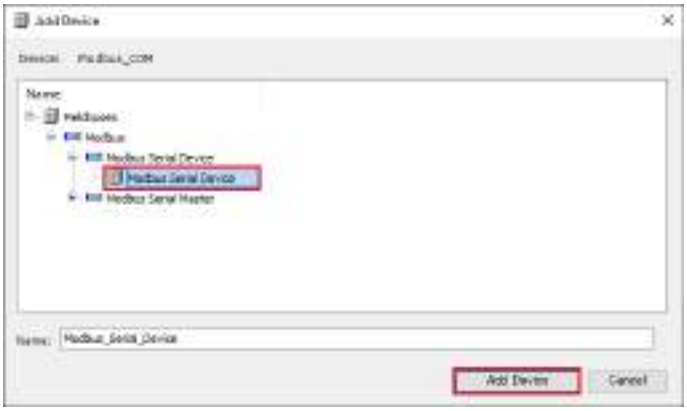
3. Click the [Add Device] button.
The selected "Modbus_COM" object will be added to the navigator pane.

4. Right-click the "Modbus_COM" object and then select "Add Device" from the context-sensitive menu that is displayed.

The "Add Device" dialog box will be displayed.



5. Select "Modbus_Serial_Device".



6. Click the [Add Device] button.
The selected "Modbus_Serial_Device" object will be added to the navigator pane.

For details on how to set up, refer to "13.3 MODBUS".

13.2 General-purpose Communication

13.2.1 General-purpose Communication (Ethernet)

This section explains how to use the CAA_NetBaseServices library, in the following order.

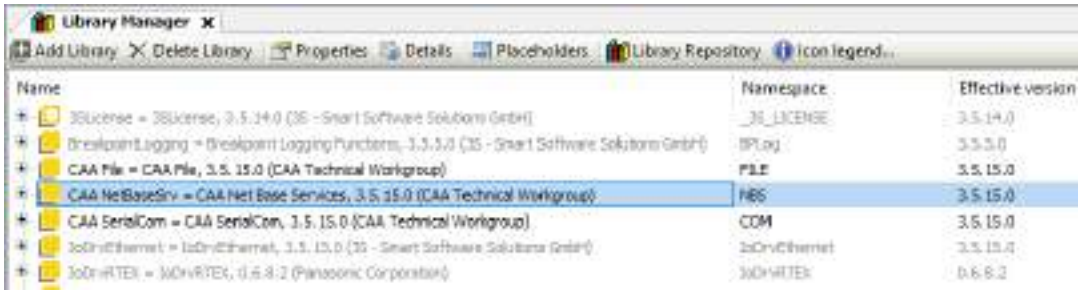
1. Library_Manager
2. TCP CLIENT processing example
3. TCP SERVER processing example
4. UDP processing example

Double-click Library_Manager in the navigator pane.

The Library_Manager setting window will be displayed.

■ Library_Manager

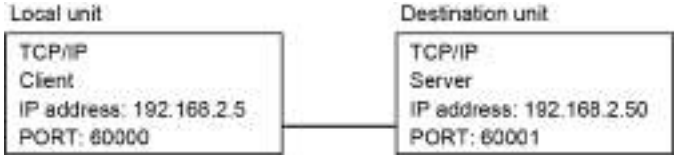
Check that the following CAA_NetBaseServices library is registered in Library_Manager.



■ TCP CLIENT processing example

The following is a processing example of data transmission / reception via TCP when the local unit is TCP CLIENT.

This processing example assumes the following operating environment.



Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- TCP client connection processing

- Reception start processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = TCP client connection processing

2 = Reception start processing

3 = Transmission processing

ClientAddr

The IP address of the destination unit is set.

Port

The port number of the destination unit is set.

TimeOut

A connection timeout period is set.

SendData

Data to be sent is set.

RecvBuf

Received data is stored.

RecvCount

The number of receptions is stored.

RecvSize

The size of received data is stored.

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- NBS_ClientError: Result of TCP client connection processing
- NBS_WriteError: Result of transmission processing
- NBS_ReadError: Result of reception start processing

Operation example

The TCP client connects to the TCP server.

- The value of "Process" is changed to 1.

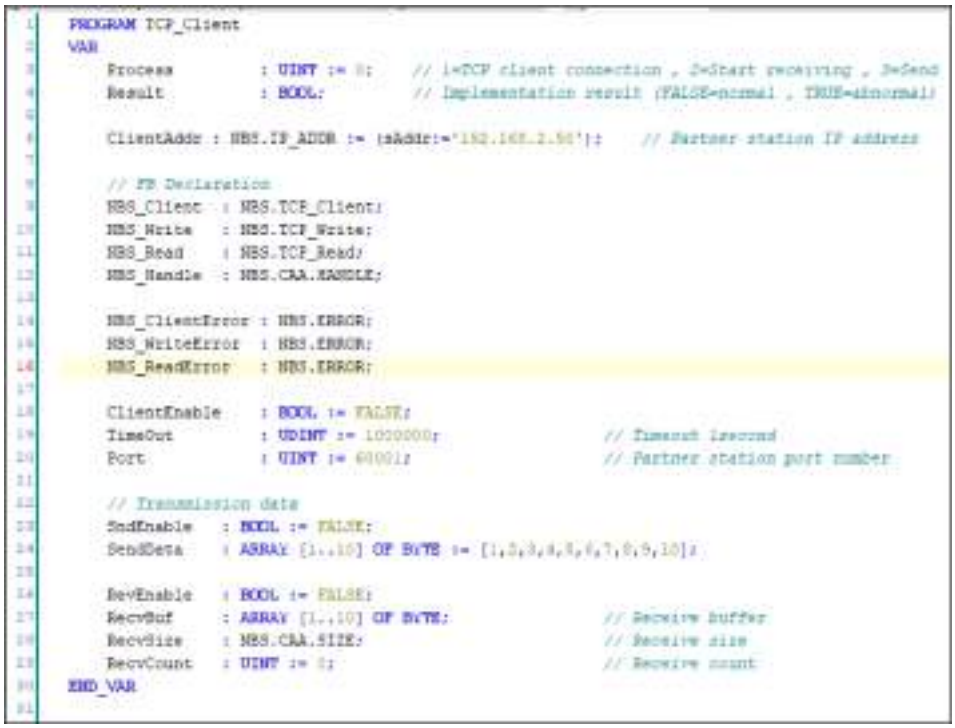
The local unit is ready to receive data. In this state, the local unit can receive data from the destination unit.

- The value of "Process" is changed to 2.

The local unit sends data to the destination unit. 10-byte data is sent to the destination unit.

- The value of "Process" is changed to 3.

Declaration section (common to ST and LD programming languages)

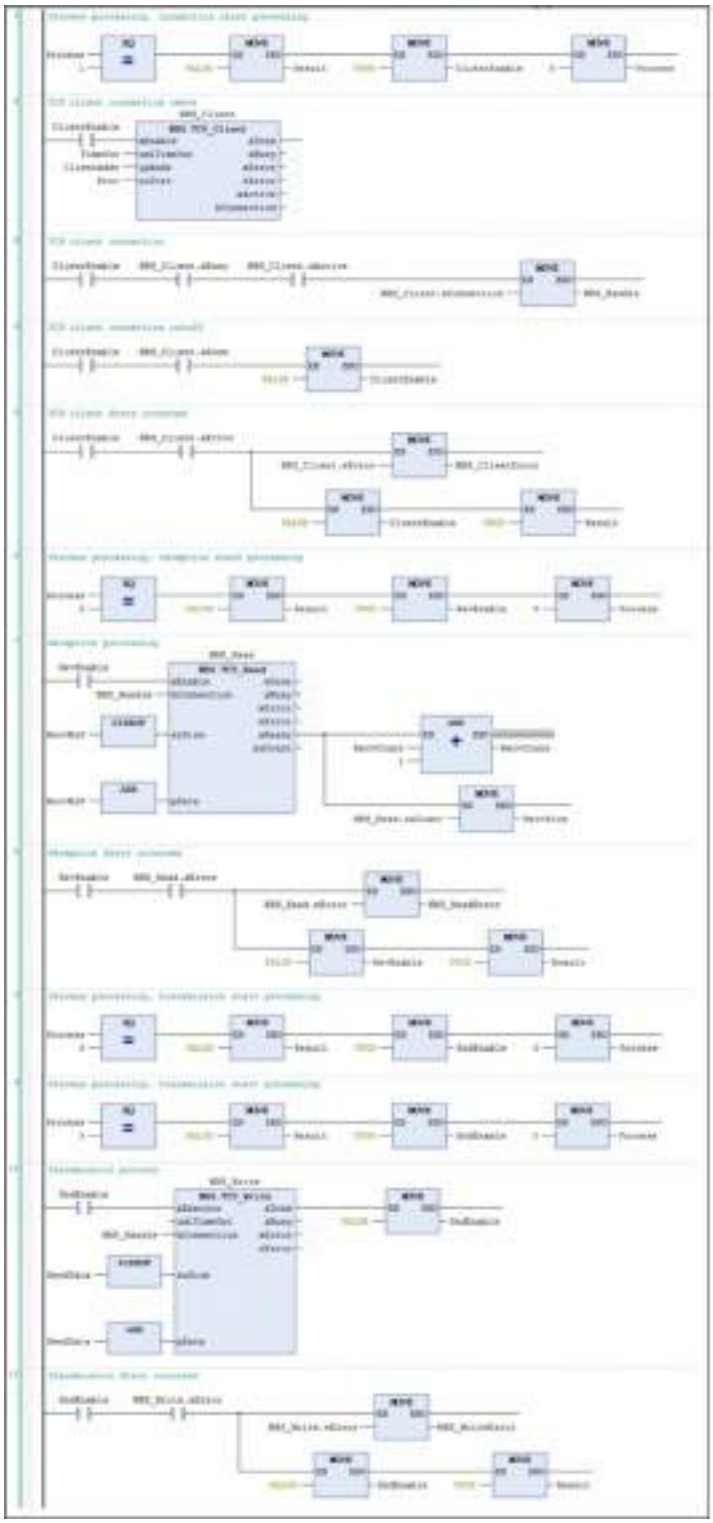


```
PROGRAM TCP_Client
VAR
    Process         : UINT := 0;    // 1=TCP client connection , 2=Start receiving , 3=Send
    Result          : BOOL;         // Implementation result (FALSE=normal , TRUE=abnormal)

    ClientAddr : NBS.IP_ADDR := (sAddr:='192.168.2.50');    // Partner station IP address

    // FB Declaration
    NBS_Client  : NBS.TCP_Client;
    NBS_Write   : NBS.TCP_Write;
    NBS_Read    : NBS.TCP_Read;
    NBS_Handle  : NBS.CAA.HANDLE;

    NBS_ClientError : NBS.ERROR;
    NBS_WriteError  : NBS.ERROR;
    NBS_ReadError   : NBS.ERROR;

    ClientEnable    : BOOL := FALSE;
    TimeOut         : UDINT := 1000000;             // Timeout 1second
    Port            : UINT := 60001;                // Partner station port number

    // Transmission data
    SndEnable   : BOOL := FALSE;
    SndData     : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];

    RcvEnable   : BOOL := FALSE;
    RecvBuf     : ARRAY [1..10] OF BYTE;            // Receive buffer
    RecvSize    : NBS.CAA.SIZE;                     // Receive size
    RecvCount   : UINT := 0;                        // Receive count
END_VAR
```

Implementation section (ST programming language)

```
// TCP client connection check
IF ClientEnable = TRUE THEN
    MBS_Client();
    IF (MBS_Client.xBusy = TRUE)  AND (MBS_Client.xActive = TRUE) THEN
        // TCP client connection
        MBS_Handle := MBS_Client.hConnection;
    ELSIF MBS_Client.xDone = TRUE THEN
        // TCP client connection cutoff
        ClientEnable := FALSE;
        MBS_Client( xEnable := ClientEnable);
    ELSIF MBS_Client.xError = TRUE THEN
        // Error occurred
        MBS_ClientError := MBS_Client.eError;
        ClientEnable := FALSE;
        MBS_Client( xEnable := ClientEnable);
        Result := TRUE;
    END_IF
END_IF

// Reception enabled
IF RcvEnable = TRUE THEN
    // Receipt confirmation
    MBS_Read();
    IF MBS_Read.xReady = TRUE THEN
        RecvCount := RecvCount + 1;          // Received number update
        RecvSize := MBS_Read.szCount;        // Receive size
    ELSIF MBS_Read.xError = TRUE THEN        // Error occurred
        MBS_ReadError := MBS_Read.eError;    // Error information storage
        RcvEnable := FALSE;
        MBS_Read( xEnable := RcvEnable );    // Stop receiving
        Result := TRUE;
    END_IF
END_IF

IF SndEnable = TRUE THEN
    // Transmission completion process
    MBS_Write();
    IF MBS_Write.xDone = TRUE THEN           // send completely
        MBS_Write( xExecute := FALSE );
        Process := 0;
    ELSIF MBS_Write.xError = TRUE THEN       // Error occurred
        MBS_WriteError := MBS_Write.eError; // Error information storage
        MBS_Write( xExecute := FALSE );      // Stop sending
        Result := TRUE;
        Process := 0;
    END_IF
END_IF

CASE Process OF
    0:  // TCP client connection
        Result := FALSE;
        ClientEnable := TRUE;

        MBS_Client( xEnable := ClientEnable ,    // Execute
            udiTimeOut := TimeOut ,              // Time-out setting
            ipAddr := ClientAddr ,               // Partner station IP address
            uiPort := Port );                    // Partner station port number
        Process := 1;

    1:  // Reception start processing
        Result := FALSE;
        RcvEnable := TRUE;
        MBS_Read( xEnable := RcvEnable ,         // Start receiving
                hConnection := MBS_Handle ,      // Connection handle
                pData := ADR(RecvBuf) ,          // Receive buffer
                szSize := SIZEOF(RecvBuf));      // Receive size
        Process := 2;

    2:  // Transmission process
        Result := FALSE;
        MBS_Write( xExecute := TRUE ,            // Start sending
                hConnection := MBS_Handle ,      // Connection handle
                pData := ADR(SendData) ,         // Send buffer
                szSize := SIZEOF(SendData));     // Send size
        SndEnable := TRUE;
        Process := 3;                            // Transmission completion wait processing

END_CASE
```

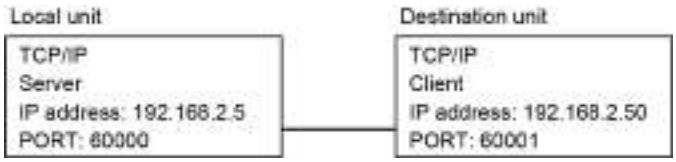
Implementation section (LD programming language)



■ TCP SERVER processing example

The following is a processing example of data transmission / reception via TCP when the local unit is TCP SERVER.

This processing example assumes the following operating environment.



Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- TCP server open processing
- TCP connection processing
- Reception start processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = TCP server open processing

2 = TCP connection processing

3 = Reception start processing

4 = Transmission processing

MyAddr

The IP address of the local unit is set.

MyPort

The port number of the local unit is set.

SendData

Data to be sent is set.

RecvBuf

Received data is stored.

RecvCount

The number of receptions is stored.

RecvSize

The size of received data is stored.

sClientAddr

The IP address of the connected data destination is stored.

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- NBS_ServError: Result of TCP server open processing
- NBS_ConErrorr: Result of TCP connection processing

- NBS_ReadError: Result of reception start processing
- NBS_WriteError: Result of transmission processing

Operation example

The TCP server is opened and connected to the TCP client.

- The value of "Process" is changed from 1 to 2.

The local unit is ready to receive data. In this state, the local unit can receive data from the destination unit.

- The value of "Process" is changed to 3.

The local unit sends data to the destination unit. 10-byte data is sent to the destination unit.

- The value of "Process" is changed to 4.

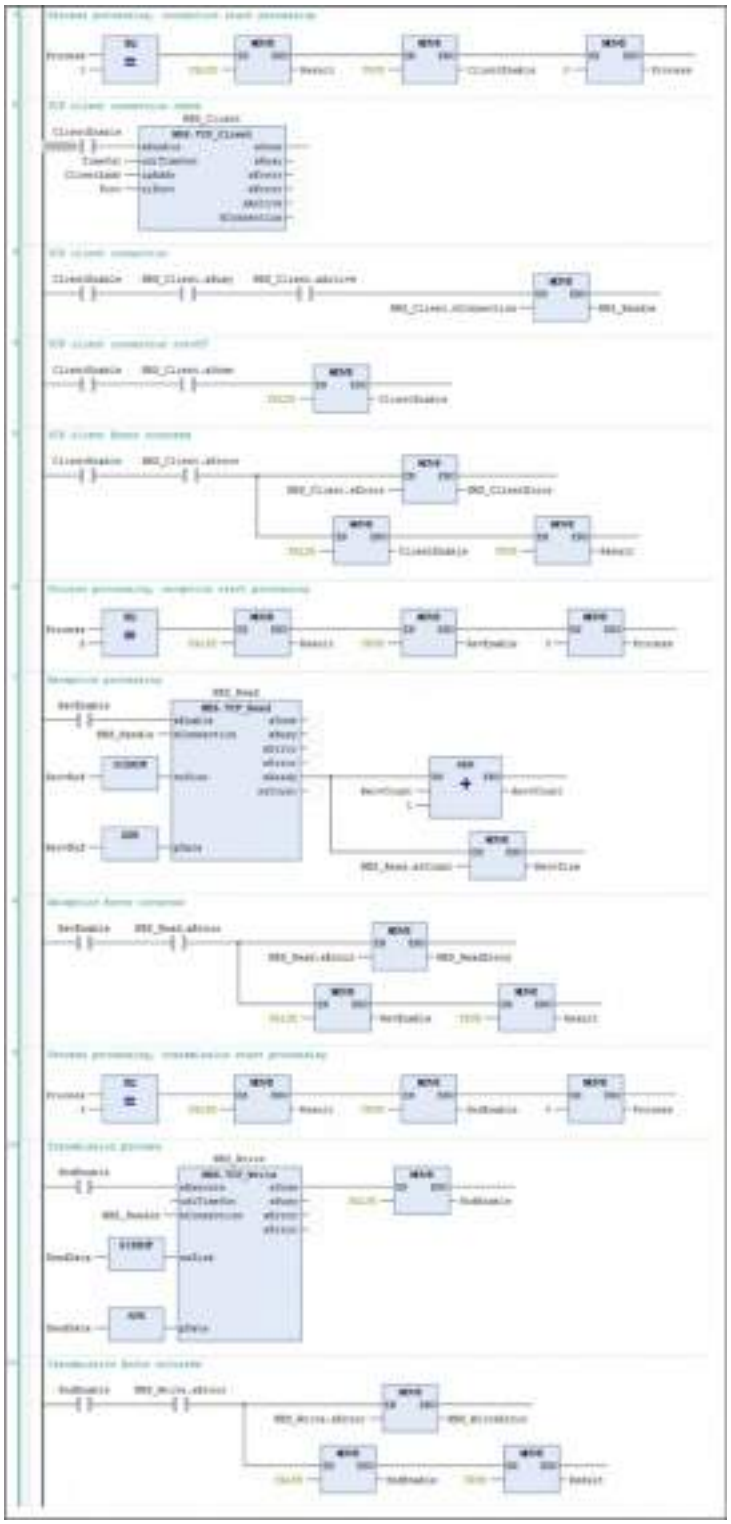
Declaration section (common to ST and LD programming languages)

```
PROGRAM TCP_Server
VAR
    Process : UINT := 0;    // 1=server open , 2=connect , 3=Start receiving , 4=Send
    Result  : BOOL;         // Implementation result (FALSE=normal , TRUE=abnormal)

    MyAddr  : NBS.IP_ADDR := (sAddr:='192.168.2.3');    // Own station IP address
    MyPort  : UINT := 60000;                            // Own station port number

    // FB Declaration
    NBS_Server      : NBS.TCP_Server;
    NBS_Connection  : NBS.TCP_Connection;
    NBS_Read        : NBS.TCP_Read;
    NBS_Write       : NBS.TCP_Write;

    NBS_Handle      : NBS.CAA.HANDLE;
    NBS_SrvError    : NBS.ERROR;
    NBS_ConError    : NBS.ERROR;
    NBS_ReadError   : NBS.ERROR;
    NBS_WriteError  : NBS.ERROR;

    // Transmission data
    SendData    : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];

    ServerEnable: BOOL := FALSE;            // Server port open process in progress
    ConEnable   : BOOL := FALSE;            // In process of connecting
    SndEnable   : BOOL := FALSE;            // Sending in progress
    RevEnable   : BOOL := FALSE;            // Receive processing
    RecvBuf     : ARRAY [0..10] OF BYTE;    // Receive buffer
    RecvCount   : UINT := 0;                // Number of receptions
    RecvSize    : NBS.CAA.SIZE;             // Received data size
    ClientAddr  : NBS.SysSocket.INADDR;     // IP address of the client
    sClientAddr : STRING;                   // Destination IP address
END_VAR
```

Implementation section (ST programming language)

```
// Server port open process
HBS_Server( xEnable:=ServerEnable , ipAddr:=MyAddr , uiPort := MyPort );
IF ServerEnable = TRUE THEN
    // Connection confirmation
    IF HBS_Server.xError = TRUE THEN            // Error occurred
        HBS_SerError := HBS_Server.eError;      // Error information storage
        ServerEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Waiting for connection completion
HBS_Connection( xEnable:=ConEnable , hServer:=HBS_Server.hServer );
IF HBS_Connection.xActive = TRUE THEN
    HBS_Handle := HBS_Connection.hConnection;   // Connection handle
    ClientAddr := HBS_Connection.IPAddress;     // Get the IP address of the connection destination
    sClientAddr := HBS.UDINT_TO_IPSTRING( udiIPAddress := ClientAddr.ulAddr );
                                                // IP address translation
ELSIF HBS_Connection.xError = TRUE THEN         // Error occurred
    HBS_ConError := HBS_Connection.eError;      // Error information storage
    ConEnable := FALSE;
    Result := TRUE;
END_IF

// Reception enabled
HBS_Read( xEnable:=RevEnable , hConnection:=HBS_Handle , pData := ADR(RecvBuf) , szSize:= SIZEOF(RecvBuf));
IF RevEnable = TRUE THEN
    // Receipt confirmation
    IF HBS_Read.xReady = TRUE THEN
        RecvCount := RecvCount + 1;         // Received number update
        RecvSize := HBS_Read.szCount;       // Receive size
    ELSIF HBS_Read.xError = TRUE THEN       // Error occurred
        HBS_ReadError := HBS_Read.eError;   // Error information storage
        RevEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Transmission completion process
HBS_Write( xExecute:=SndEnable , hConnection:=HBS_Handle , pData:=ADR(SendData) , szSize:=SIZEOF(SendData));
IF SndEnable = TRUE THEN
    IF HBS_Write.xDone = TRUE THEN          // send completely
        SndEnable := FALSE;
    ELSIF HBS_Write.xError = TRUE THEN      // Error occurred
        HBS_WriteError := HBS_Write.eError; // Error information storage
        SndEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

CASE Process OF
    1:  // TCP server open
        ServerEnable := TRUE;
        Result := FALSE;
        Process := 0;

    2:  // TCP connect
        ConEnable := TRUE;
        Result := FALSE;
        Process := 0;

    3:  // Reception start processing
        RevEnable := TRUE;
        Result := FALSE;
        Process := 0;

    4:  // Transmission process
        SndEnable := TRUE;
        Result := FALSE;
        Process := 0;

END_CASE
```

Implementation section (LD programming language)



When multiple clients are connected simultaneously to the same port, multiple TCP_Connection instances are created. The hServer handle acquired by one TCP_Server is set to the multiple TCP_Connection instances.

Example: When two clients are connected simultaneously to the same port

Declaration section

```
iServer: NBS.TCP_Server;// TCP_Server instance
iConnection: ARRAY [0..1] OF NBS.TCP_Connection;    // TCP_Connection instance (two instances)
```

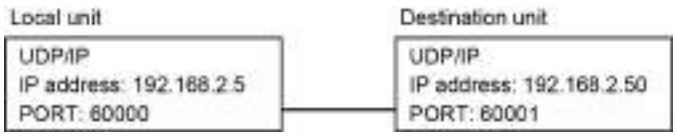
Implementation section

```
iServer( xEnable:=TRUE , ipAddr:=ipAddr , uiPort:=uiPort );    // Server opened
// Omitted (Waiting for TCP_Server completion)
iConnection[0]( xEnable := TRUE , hServer := iServer.hServer );    // For 1st client
iConnection[1]( xEnable := TRUE , hServer := iServer.hServer );    // For 2nd client
```

■ UDP processing example

An example of processing for data transmission / reception via UDP is as follows:

This processing example assumes the following operating environment.



Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- Port open processing
- Reception start processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = Port open processing

2 = Reception start processing

3 = Transmission processing

MyipAddr

The IP address of the local unit is set.

MyPort

The port number of the local unit is set.

SendAddr

The IP address of the destination unit is set.

SendPort

The port number of the destination unit is set.

SendData

Data to be sent is set.

RecvBuf

Received data is stored.

RecvCount

The number of receptions is stored.

RecvPort

The port that received data is stored.

RecvSize

The size of received data is stored.

RecvIpAddr

The IP address of the received data destination is stored.

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- NBS_PeerError: Result of port open processing
- NBS_RecError: Result of reception start processing
- NBS_ReadError: Result of transmission processing

Operation example

The port is opened and the local unit is ready to receive data. In this state, the local unit can receive data from the destination unit.

- The value of "Process" is changed from 1 to 2.

The local unit sends data to the destination unit. 10-byte data is sent to the destination unit.

- The value of "Process" is changed to 3.

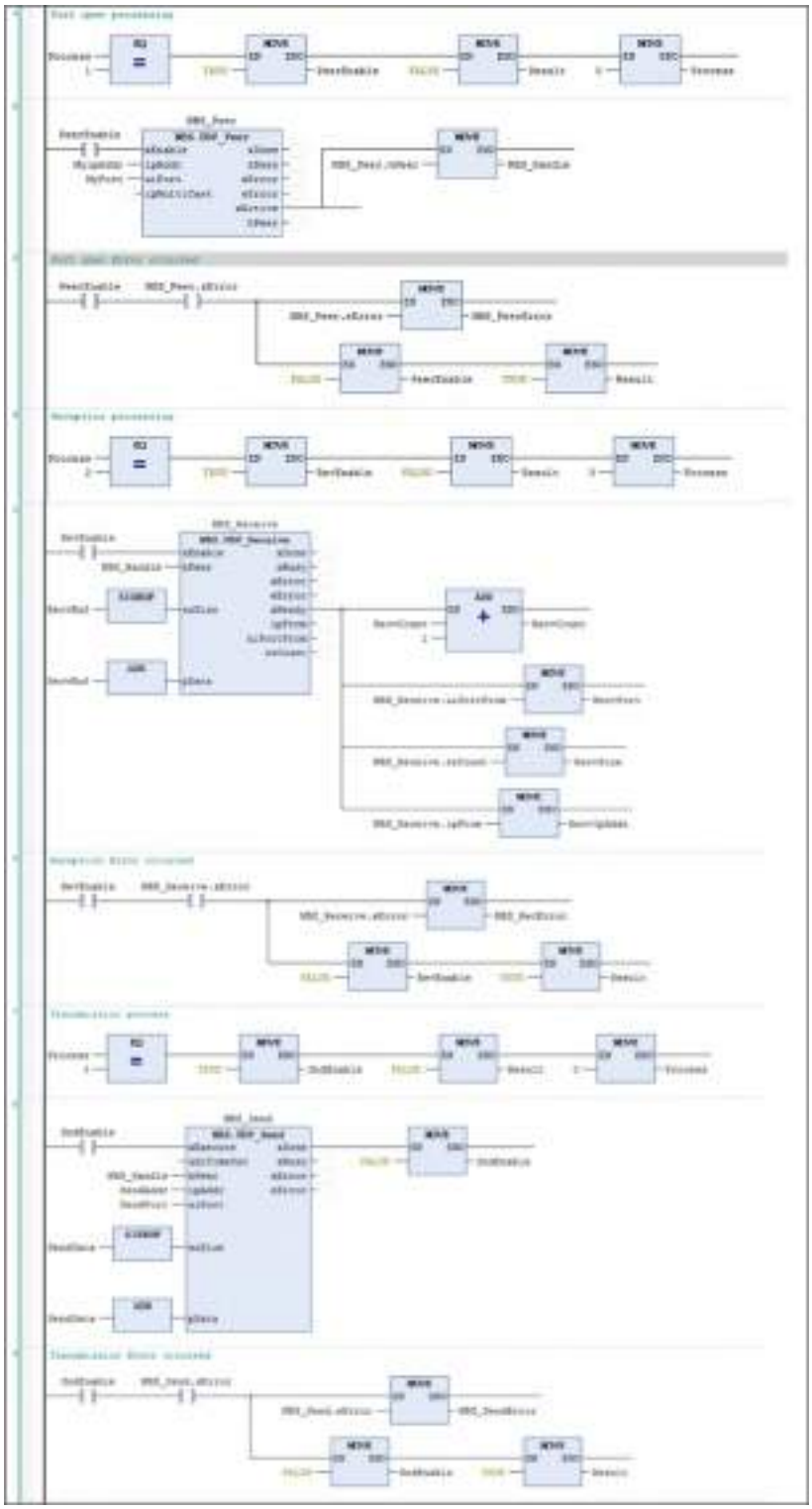
Declaration section (common to ST and LD programming languages)

```
PROGRAM UDP
VAR
    Process     : UINT := 0;    // 1=Port open , 2=Receive start , 3=Send
    Result      : BOOL;         // Implementation result (FALSE=normal , TRUE=abnormal)

    MyIpAddr : NBS.IP_ADDR := (sAddr:='192.168.2.5');   // Own station IP address
    SendAddr : NBS.IP_ADDR := (sAddr:='192.168.2.50');  // Partner station IP address
    MyPort   : UINT := 60000;                           // Own station PORT number
    SendPort : UINT := 60001;                           // Partner station PORT number

    // FB Declaration
    NBS_Peer    : NBS.UDP_Peer;
    NBS_Receive : NBS.UDP_Receive;
    NBS_Send    : NBS.UDP_Send;

    NBS_Handle      : NBS.CAA.HANDLE;       //  PORT handle
    NBS_PeerError   : NBS.ERROR;            // UDP_Peer Error information
    NBS_RecError    : NBS.ERROR;            // UDP_Receive Error information
    NBS_SendError   : NBS.ERROR;            // UDP_Send Error information

    // Transmission data
    SendData : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];

    PeerEnable  : BOOL := FALSE;
    SndEnable   : BOOL := FALSE;
    RcvEnable   : BOOL := FALSE;

    RecvBuf : ARRAY [1..10] OF BYTE;    // Receive buffer
    RecvCount : UINT := 0;              // Receive count
    RecvPort : UINT;                    // Receive port
    RecvSize : NBS.CAA.SIZE;            // Receive size
    RecvIpAddr : NBS.IP_ADDR;           // Destination IP address
END_VAR
```

Implementation section (ST programming language)

```
// Port open processing
NBS_Peer( xEnable:=PeerEnable , ipAddr:=MyipAddr , uiPort:=MyPort );
IF PeerEnable = TRUE THEN
    IF NBS_Peer.xActive = TRUE THEN          // Successful port opening
        NBS_Handle := NBS_Peer.hPeer;        // Get handle
    ELSIF NBS_Peer.xError = TRUE THEN        // Error occurred
        NBS_PeerError := NBS_Peer.eError ;   // Error information storage
        PeerEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Reception processing
NBS_Receive( xEnable:=RevEnable , hPeer:=NBS_Handle , pData:=ADR(RecvBuf) ,
             szSize:=SIZEOF(RecvBuf));
IF RevEnable = TRUE THEN
    IF NBS_Receive.xReady = TRUE THEN        // Received data available
        RecvCount := RecvCount + 1;          // Received number update
        RecvPort := NBS_Receive.uiPortFrom;  // Destination PORT
        RecvSize := NBS_Receive.szCount;     // Receive size
        RecvIpAddr := NBS_Receive.ipFrom;    // Destination IP address
    ELSIF NBS_Receive.xError = TRUE THEN     // Error occurred
        NBS_RecError := NBS_Receive.eError;  // Error information storage
        RevEnable := FALSE;
        Result := TRUE;
    END_IF
END_IF

// Transmission process
NBS_Send( xExecute:=SndEnable , hPeer:=NBS_Handle , ipAddr:=SendAddr ,
          uiPort:=SendPort , pData:=ADR(SendData) , szSize:=SIZEOF(SendData));
IF SndEnable = TRUE THEN
    IF NBS_Send.xDone = TRUE THEN            // Successful transmission
        SndEnable := FALSE;                  // Transmission processing stopped
    ELSIF NBS_Send.xError = TRUE THEN        // Error occurred
        NBS_SendError := NBS_Send.eError;    // Error information storage
        SndEnable := FALSE;                  // Transmission processing stopped
        Result := TRUE;
    END_IF
END_IF

CASE Process OF
    1: // Port open processing
        PeerEnable := TRUE;
        Result := FALSE;
        Process := 0;

    2:  // Reception processing
        RevEnable := TRUE;
        Result := FALSE;
        Process := 0;

    3: // Transmission process
        SndEnable := TRUE;
        Result := FALSE;
        Process := 0;
END_CASE
```

Implementation section (LD programming language)



13.2.2 General-purpose Communication (Serial)

This section explains how to use the CAA SerialCom library, in the following order.

1. Library_Manager
2. COM transmission / reception processing example

■ Library_Manager

Check that the following CAA SerialCom library is registered in Library_Manager.

■ COM transmission / reception processing example

Send and receive data via SerialCom.
Specify communication settings as below.

COM number	1
Baud rate	115200 bps
Data bits	8
Parity bit	Odd
Stop bit	1

Processing for data transmission / reception

The processing for data transmission / reception is as follows:

- Serial port open processing
- Serial port close processing
- Reception processing
- Transmission processing

Explanation of variables

Process

When the value is rewritten, the following processing is executed. After the execution is completed, the variable is set to 0 (invalid value).

1 = Serial port open processing

2 = Reception processing

3 = Transmission processing

4 = Serial port close processing

SendBuf

Data to be sent is set.

SendBufLen

The length of data to be sent is set.

RecvBuf

The buffer to store received data is set.

ReadBufLen

The length of receiver buffer is set.

ReadSize

The size of received data is stored.

Result

The result of processing execution is stored. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the following error code.

- ComErr: COM processing result

Operation example

Serial port is opened.

- The value of "Process" is changed to 1.

Received data is read.

- The value of "Process" is changed to 2.

10-byte data is sent.

- The value of "Process" is changed to 3.

Serial port is closed.

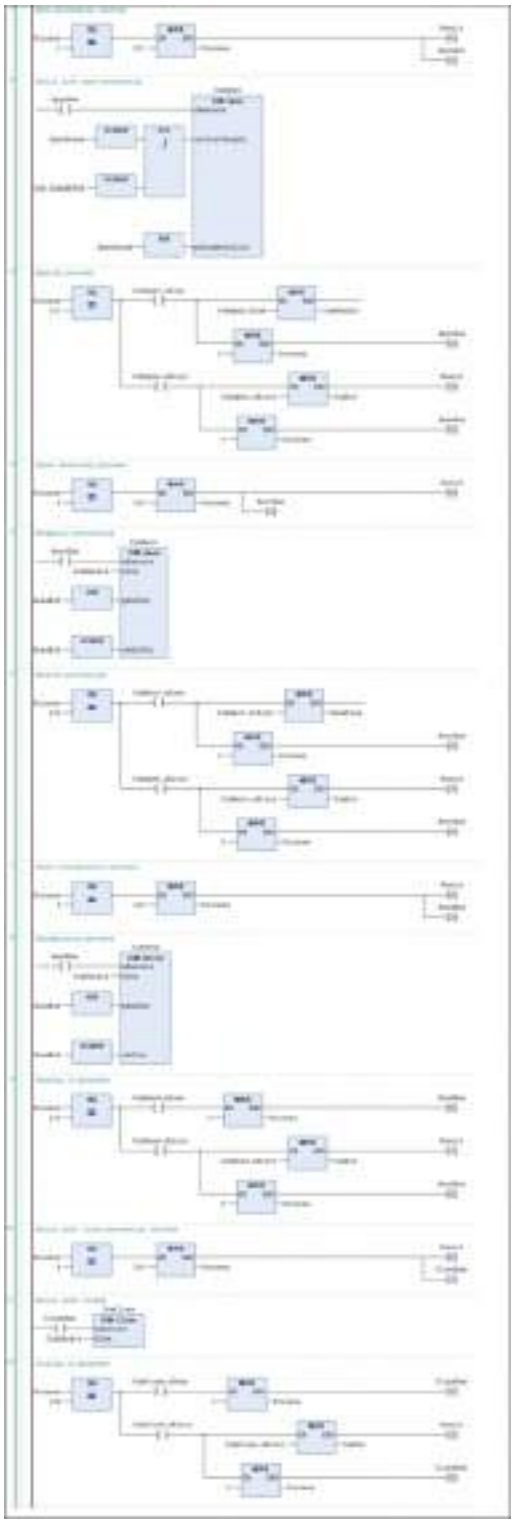
- The value of "Process" is changed to 4.

Declaration section (common to ST and LD programming languages)

```
PROGRAM ST_000
VAR
    Process         : UINT := 0;    // 1=COM_OPEN 2=READ 3=SEND 4=COM_CLOSE
    Result          : BOOL;         // Implementation result (FALSE=normal / TRUE=abnormal)

    ComOpen   : COM.Open;
    ComClose  : COM.Close;
    ComSend   : COM.Write;
    ComRecv   : COM.Read;
    ComHandle: COM.CAS_HANDLE := 0;     // COM device handle
    ComErr    : COM.ERROR;              // COM error code

    // Communication parameters
    OpenParam : ARRAY [1..7] OF COM.PARAMETER := [
        (udiParameterId := COM.CAA_Parameter_Constants.udiPort,     udiValue := 1),
        (udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate, udiValue := 115200),
        (udiParameterId := COM.CAA_Parameter_Constants.udiParity,   udiValue := INT_TO_UDINT(COM.PARITY.ODD)),
        (udiParameterId := COM.CAA_Parameter_Constants.udiStopBits, udiValue := INT_TO_UDINT(COM.STOPBIT.ONESTOPBIT)),
        (udiParameterId := COM.CAA_Parameter_Constants.udiTimeout,  udiValue := 0),
        (udiParameterId := COM.CAA_Parameter_Constants.udiByteSize, udiValue := 8),
        (udiParameterId := COM.CAA_Parameter_Constants.udiBinary,   udiValue := 1)
    ];

    OpenExe     : BOOL := FALSE;
    RecvExe     : BOOL := FALSE;
    SendExe     : BOOL := FALSE;
    CloseExe    : BOOL := FALSE;

    ReadBuf     : ARRAY [1..10] OF BYTE;        // Read buffer
    ReadSize    : UDINT;                        // Read data size

    SendBuf : ARRAY [1..10] OF BYTE := [1,2,3,4,5,6,7,8,9,10];  // Transmission data
END_VAR
```

Implementation section (ST programming language)

Implementation section (LD programming language)



13.3 MODBUS

13.3.1 What is Modbus TCP?

The GM1 controller can communicate with HMI and controllers via the Modbus protocol. Master communication or slave communication can be performed by adding a master or slave device for Modbus to the project file.

13.3.2 Modbus-TCP Master Communication

The Modbus-TCP master function can be used to send commands to slave devices in the following two ways.

1) When device object settings are used

- Transmission method based on slave initialization
- Transmission method
  - Cyclic
  - Rising edge
  - Application (ModbusChannel function block)

2) When device object settings are not used

- A method by which a user program (ModbusRequest function block) generates and sends commands

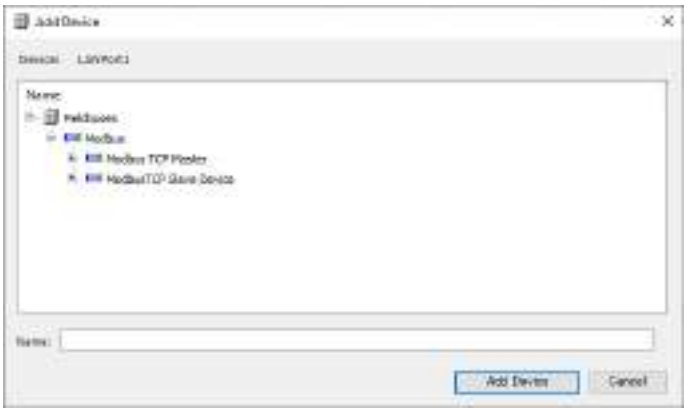
Info.

- For details on how to use ModbusRequest, refer to the *GM1 Series Reference Manual (Instruction)*.

The method in 1) is explained below.

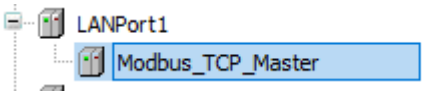
Procedure

1. Right-click the "LANPort1" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
   The "Add Device" dialog box will be displayed.

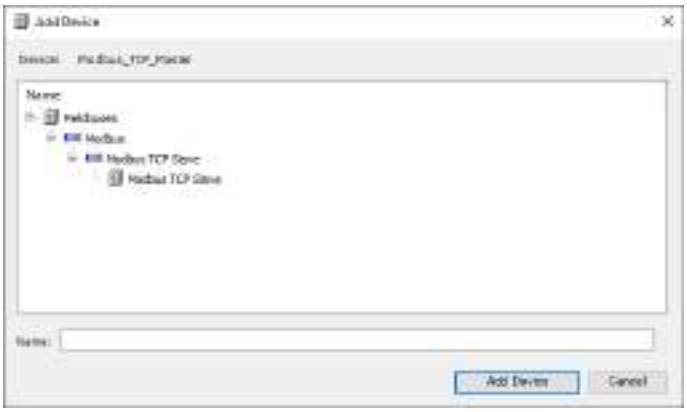


2. Select "Modbus TCP Master" under "Modbus" and click the [Add Device] button.
"Modbus_TCP_Master" will be added to the "LANPort1" object.

Example: When master communication is performed via LAN port1

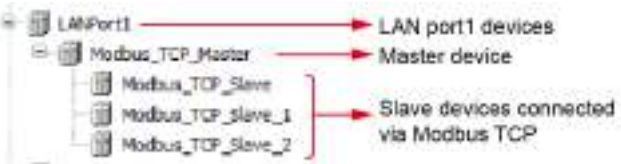


3. Right-click the "Modbus_TCP_Master" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
The "Add Device" dialog box will be displayed.

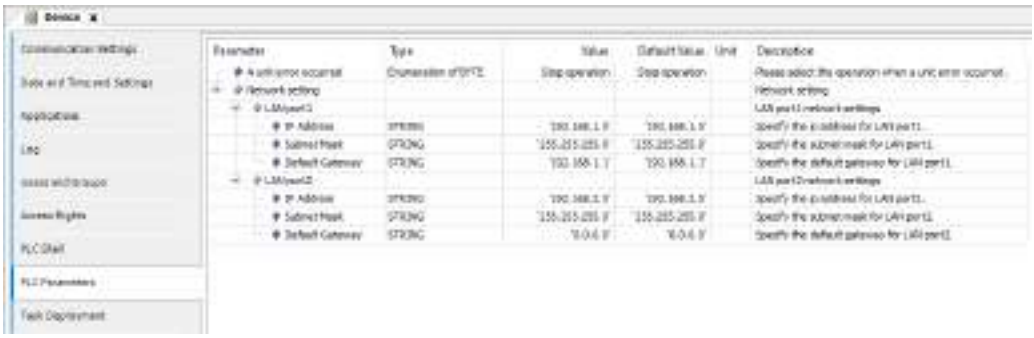


4. Select "Modbus TCP Slave" under "Modbus TCP Slave" and click the [Add Device] button.
"Modbus_TCP_Slave" will be added to the "Modbus_TCP_Master" object.

Example: When master communication is performed via LAN port1



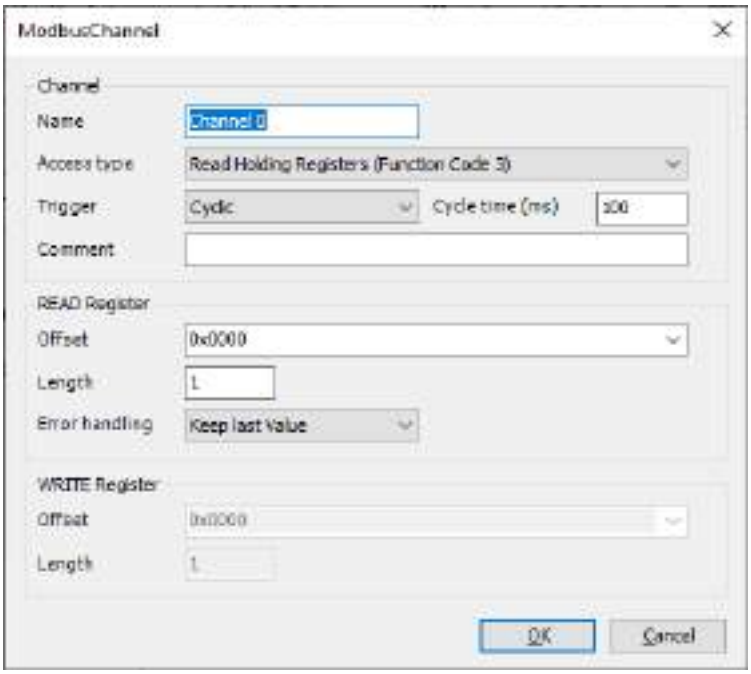
5. Open the device (AGM1CSR16T1) of the GM1 controller, select the "'PLC' Parameters" tab, and set the IP address of LAN port1.



6. Double-click "Modbus_TCP_Slave" in the navigator pane.
   The "Modbus_TCP_Slave" object will be displayed.

7. Select the "Modbus Slave Channel" tab.

8. Click the [Add Channel] button.
   The "Modbus Channel" dialog box will be displayed.
   Enter information for channels to be used. Up to 100 channels can be set.



Access type:

Select an access type (function code) and change the value of the READ or WRITE register parameter according to the selected access type.

Function code	Access type	Description
1	Read Coils	Reads from coils
2	Read Discrete Inputs	Reads from discrete inputs
3	Read Holding Registers	Reads from holding registers
4	Read Input Registers	Reads from input registers
5	Write Single Coil	Writes to single coil
6	Write Single Register	Writes to single register
15	Write Multiple Coils	Writes to multiple coils
16	Write Multiple Registers	Writes to multiple registers
23	Read / Write Multiple Registers	Reads from or writes to multiple registers

Trigger:

Select conditions for command transmission.

Access type	Description
Cyclic	Commands are sent periodically. Enter a transmission interval in the Cycle time field.
Rising edge	Commands are sent at the rising edge of a Boolean trigger variable. The trigger variable area is defined in the I/O Mapping tab.
Application	Commands are sent using the ModbusChannel function block in a user program.

READ register settings

Item	Description
Offset	Specifies the starting address from which read operation is to be started.
Length	Specifies the number of registers to be read from. The value of the parameter depends on the function code.
Error handling	Defines data that identifies communication errors. ● "Keep last value": Holds the last value that is read ● "Set to ZERO": Sets 0

WRITE register settings

Item	Description
Offset	Specifies the starting address from which write operation is to be started. For SP15, do not specify offset "65535".
Length	Specifies the number of registers to be written to. The value of the parameter depends on the function code.

This completes the device object settings for the method for sending commands based on slave initialization and the method for sending commands based on channel settings.

13.3.3 Modbus-TCP Slave Communication

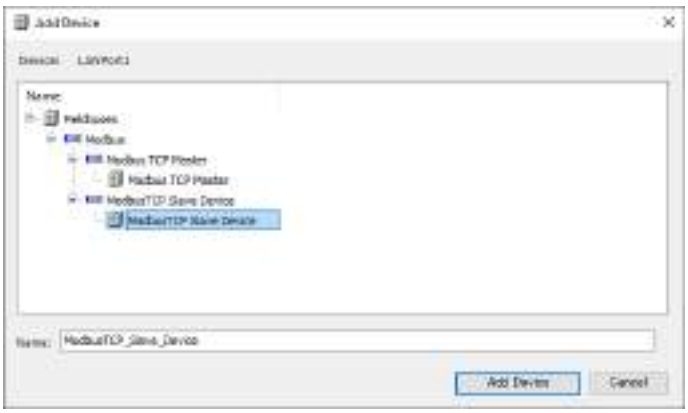
This section explains how to use the Modbus-TCP slave function.

The slave function is used by setting up the device object.

1 2 Procedure

1. Right-click the "LANPort1" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.

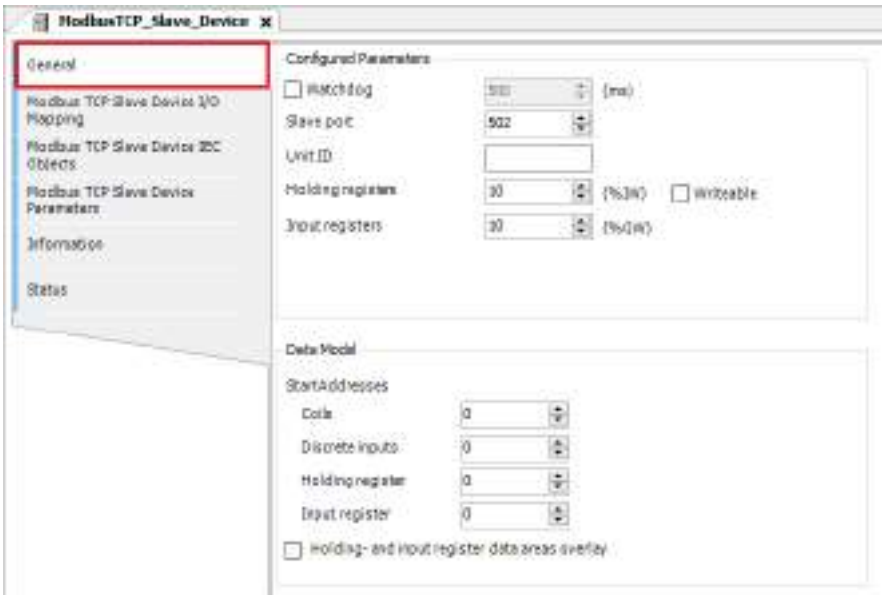
   The "Add Device" dialog box will be displayed.



2. Select "ModbusTCP Slave Device" under "ModbusTCP Slave Device" and click the [Add Device] button.

The "ModbusTCP_Slave_Device" object will be added to the "LANPort1" object.

3. Double-click "ModbusTCP_Slave_Device" in the navigator pane.
The "LANPort1" object will be displayed.
Select the "General" tab and set appropriate values for each item.



Item	Description
Watchdog	If no valid command is received from the master during the preset time period, the holding register (%IW) will be set to zero. Settable value: 500 to 200000
Slave port	Port number used by the slave (local unit) Settable value: 1 to 65535
Unit ID	A station number is set. Settable value: 1 to 247
Holding registers (%IW)	The number of holding registers is set. Buffer size of holding register: 1 to 4096
Input registers (%QW)	The number of input registers is set. Buffer size of input register: 1 to 4096

4. Select the "Modbus TCP Slave Device I/O Mapping" tab.
You can allocate variables to holding registers and input registers.

Registers correspond to each access type (function code)

Function code	Access type	Register	
		When the check box is not selected(Note 1)	When the check box is selected(Note 1)
1	Read Coils	Holding register	Input register
2	Read Discrete Inputs	Input register	Input register
3	Read Holding Registers	Holding register	Input register
4	Read Input Registers	Input register	Input register
5	Write Single Coil	Holding register	Holding register
6	Write Single Register	Holding register	Holding register
15	Write Multiple Coils	Holding register	Holding register
16	Write Multiple Registers	Holding register	Holding register
23	Read / Write Multiple Registers	Holding register (Read/ Write)	Input register (Read) Holding register (Write)

(Note 1) The register to be used is changed according to whether the "Holding register data area overlay and input register data area overlay" check box is selected.

13.3.4 Modbus-RTU Master Communication

The Modbus-RTU master function can be used to send commands to slave devices in the following two ways.

1) When device object channel settings are used

- Transmission method based on slave initialization
- Transmission method based on channel settings
  - Cyclic
  - Rising edge
  - Application (ModbusChannel function block)

2) When device object channel settings are not used (Note 1)

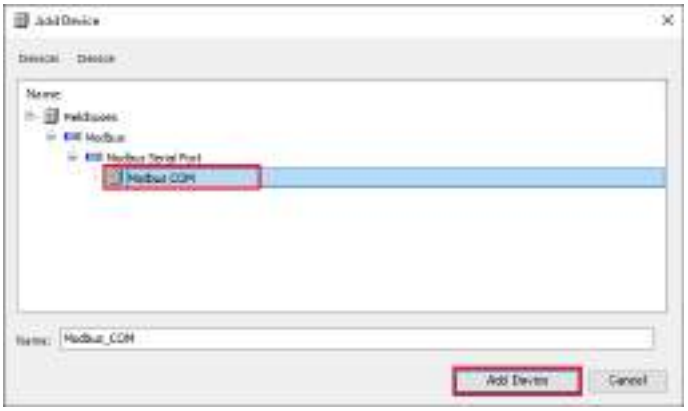
- A method by which a user program (ModbusRequest function block) generates and sends commands

  (Note 1) Device object registration is required.

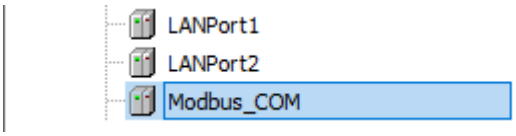
The method in 1) is explained below.

12 Procedure

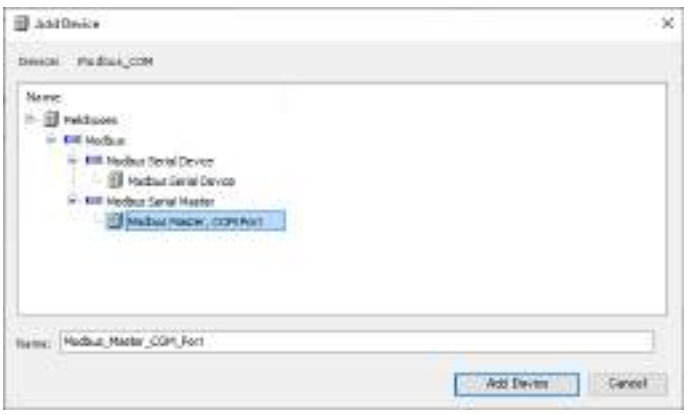
1. Right-click the "Device" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
   The "Add Device" dialog box will be displayed.



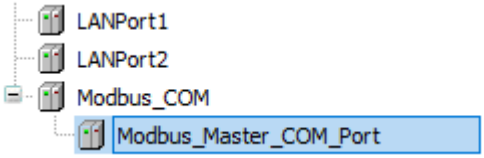
2. Select "Modbus COM" under "Modbus Serial Port" and click the [Add Device] button.
   The "Modbus_COM" object will be added to the "Device" object.



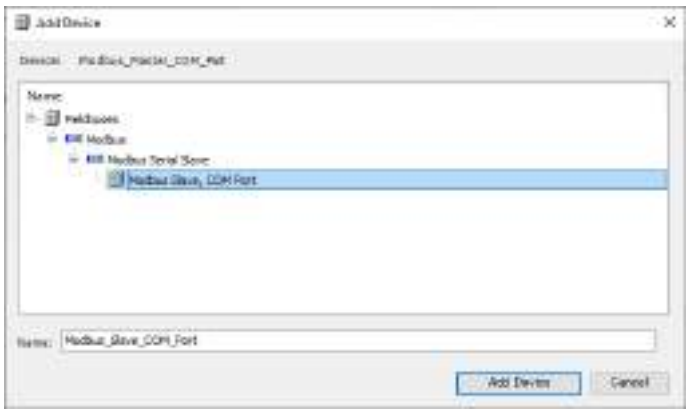
3. Right-click the "Modbus_COM" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
   The "Add Device" dialog box will be displayed.



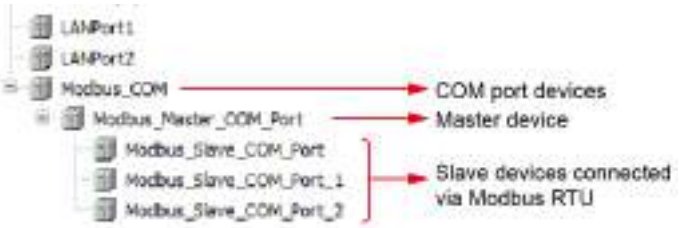
4. Select "Modbus Master, COM Port" under "Modbus Serial Master" and click the [Add Device] button.
   The "Modbus_Master_COM_Port" object will be added below the "Modbus_COM" object.



5. Right-click the "Modbus_Master_COM_Port" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
The "Add Device" dialog box will be displayed.



6. Select "Modbus Slave, COM Port" under "Modbus Serial Slave" and click the [Add Device] button.
The "Modbus_Slave_COM_Port" object will be added below the "Modbus_Master_COM_Port" object.



7. Double-click "Modbus_COM" in the navigator pane.
"Modbus_COM"object will be displayed.
Select the"General"tab and set appropriate values for each item.

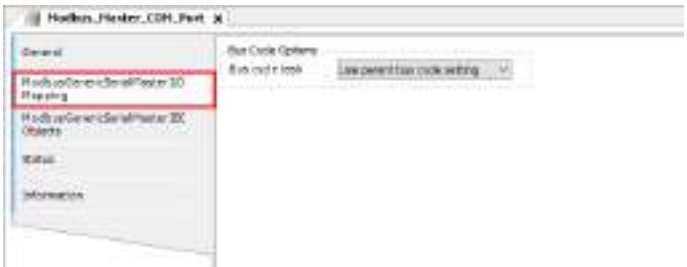
Item	Description
COM port	Settable value: 1 to 99 The COM port of the GM1 controller is fixed at 1.

Item	Description
Baud rate	Can be selected from 9600, 19200, 38400, 57600, and 115200
Parity	Can be selected from EVEN, ODD, and NONE
Data bits	Data bit length between start bit and stop bit Settable value: 7 bits and 8 bits (Settable: 0 to 255) The default value of Modbus is 8 bits.
Stop bit	Settable value: 1 bit and 2 bits The default value of Modbus is 1 bit.

8. Double-click "Modbus Master, COM Port" in the navigator pane.
The "Modbus_Master_COM_Port"object will be displayed.
Select the "General"tab and set appropriate values for each item.

項目	内容
Transmission mode	RTU: Binary transmission ASCII: ASCII code transmission (Not supported)
Response timeout [ms] [0..65535]	Waiting time for response from slave * If a response timeout period is set in the slave device, the settings in the slave device will take effect.
Time between frames [ms] [0..65535]	Time period during which master transmission is paused from when the last response is received until the next command is sent.
Auto-restart communication	When the check box is selected: After a communication error occurs, the communication status is automatically checked. When the communication is restored, reconnection is performed. When the check box is not selected: After a communication error occurs, reconnection is not performed.

"ModbusGenericSerialMaster I/O マッピング"tab: Select a bus cycle task that performs Modbus communication.

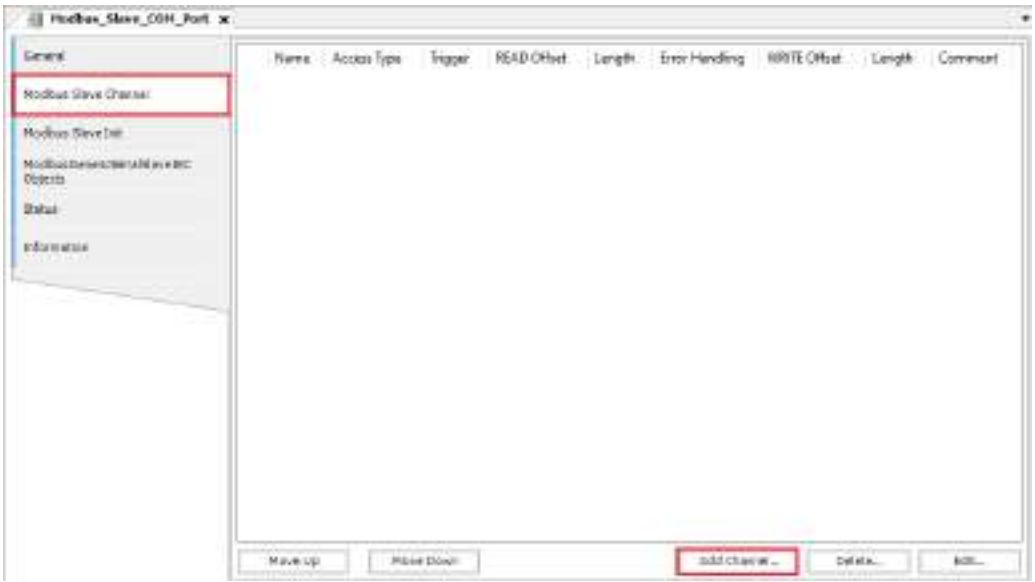


9. Double-click "Modbus Slave, COM Port" in the navigator pane.
The "Modbus_Slave_COM_Port"object will be displayed.

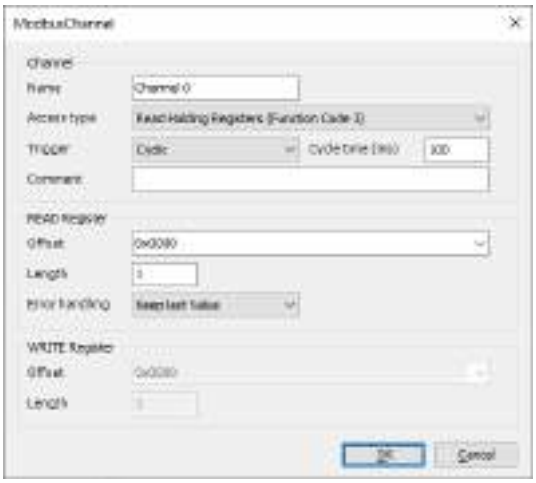
Select the "General"tab and set appropriate values for each item.

Item	Description
Slave address [1..247]	Specifies the address (station number) of the slave device.
Response timeout (ms) [2..65535]	Waiting time for response from slave * The response timeout value for the master device is overwritten.

10. Select the "Modbus Slave Channel" tab.



11. Click the [Add Channel] button.
The "Modbus Channel"dialog box will be displayed.
Enter information for channels to be used. Up to 100 channels can be set.



Access type :

Select an access type (function code) and change the value of the READ or WRITE register parameter according to the selected access type.

Function code	Access type	Description
1	Read Coils	Reads from coils
2	Read Discrete Inputs	Reads from discrete inputs
3	Read Holding Registers	Reads from holding registers
4	Read Input Registers	Reads from input registers
5	Write Single Coil	Writes to single coil
6	Write Single Register	Writes to single register
15	Write Multiple Coils	Writes to multiple coils
16	Write Multiple Registers	Writes to multiple registers
23	Read/Write Multiple Registers	Reads from or writes to multiple registers

Trigger :

Select conditions for command transmission.

Access type	Description
Cyclic	Commands are sent periodically. Enter a transmission interval in the Cycle time field.
Rising edge	Commands are sent at the rising edge of a Boolean trigger variable. The trigger variable area is defined in the I/O Mapping tab.
Application	Commands are sent using the ModbusChannel function block in a user program.

READ register settings

Item	Description
Offset	Specifies the starting address from which read operation is to be started. For SP15, do not specify offset "65535".

Item	Description
Length	Specifies the number of registers to be read from. The value of the parameter depends on the function code.
Error handling	Defines data that identifies communication errors. ● "Keep last value": Holds the last value that is read ● "Set to ZERO": Sets 0

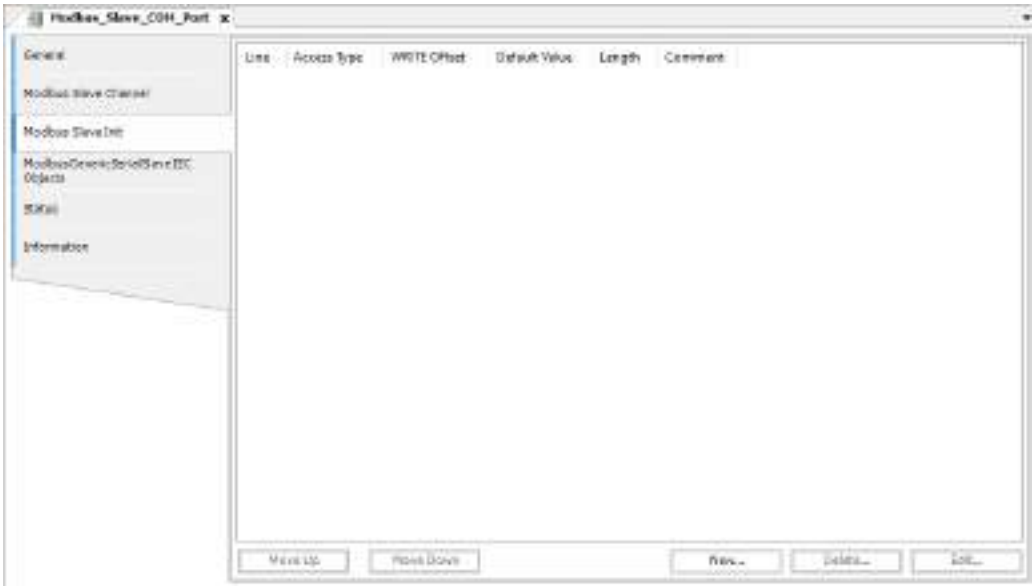
WRITE register settings

Item	内容
Offset	Specifies the starting address from which write operation is to be started.
Length	Specifies the number of registers to be written to. The value of the parameter depends on the function code.

12. Select the "Initialize Modbus Slave" tab.

Slave devices can be initialized.

Slave initialization is executed once when a slave is activated at the time of startup or RUN mode.Click the [New]button and enter information for channels to be used. Up to 20 commands can be added for each device.

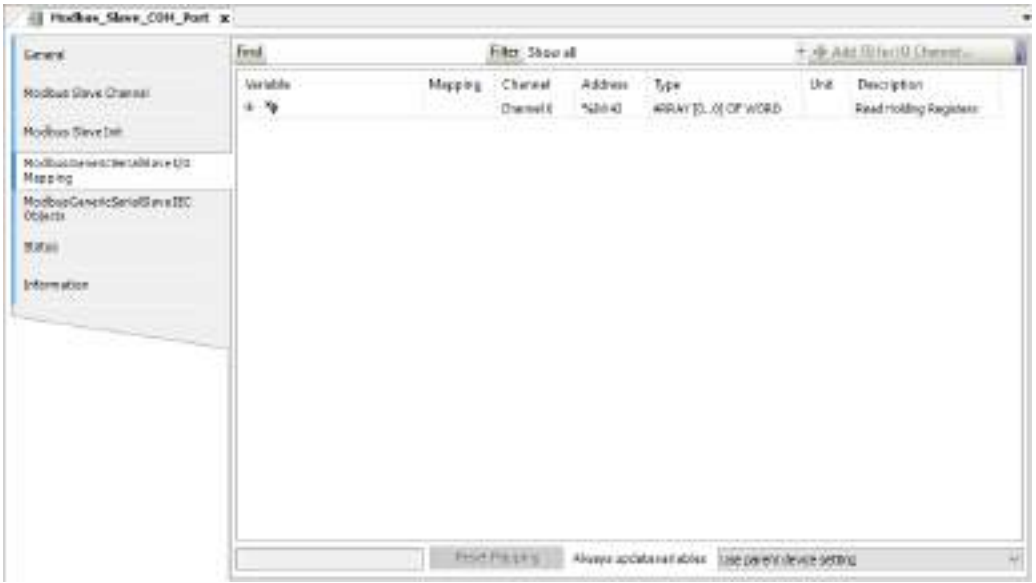


List of access types (function codes)

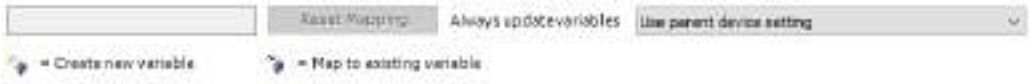
Code	Access type	Description
5	Write Single Coil	Writes to single coil
6	Write Single Register	Writes to single register
15	Write Multiple Coils	Writes to multiple coils
16	Write Multiple Registers	Writes to multiple registers

13. Select the "ModbusGenericSerialSlave I/O Mapping" tab.

Read areas, write areas, and trigger variable areas are defined according to the channel information created in Step 10. Allocate variables as necessary.



Update settings for I/O variables



Item	Description
Use parent device settings	Updates I/O variables according to the parent device settings
Enable 1 (Bus cycle task if not used by any tasks)	Updates I/O variables in the bus cycle if not used by any other task
Enable 2 (Always use bus cycle task)	Updates all I/O variables in each cycle of the bus cycle task

This completes the device object settings for the method for sending commands based on slave initialization and the method for sending commands based on channel settings.

The following is an example of creating an LD program that sends commands when a trigger is set as an "application" in channel settings.

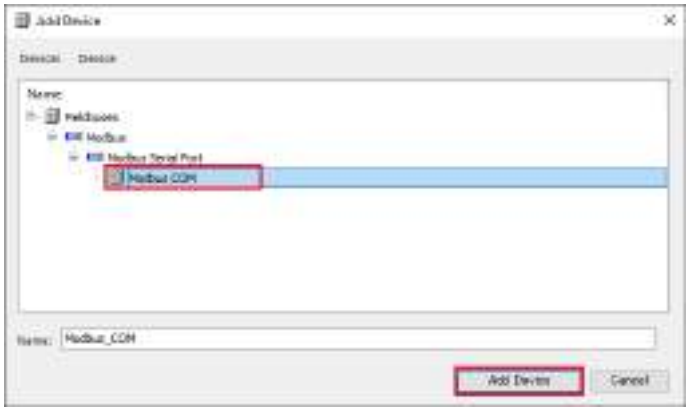
The ModbusChannel function block is used for command transmission. The slave device added to the navigator pane is specified in the slave operand, and the index of the channel that has been added to the "Modbus Slave Channel" tab and that is used to send commands is specified in the iChannelIndex command.

13.3.5 Modbus-RTU Slave Communication

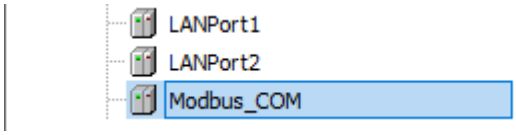
This section explains how to use the Modbus-RTU slave function.
The slave function is used by setting up the device object.

1 2 Procedure

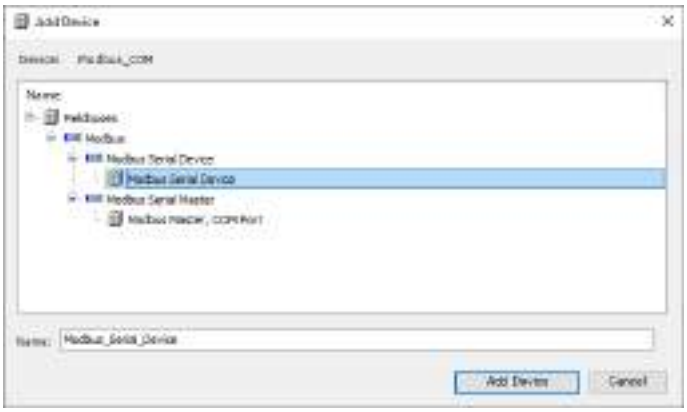
1. Right-click the "Device" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
   The "Add Device" dialog box will be displayed.



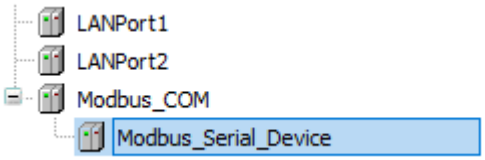
2. Select "Modbus COM" under "Modbus Serial Port" and click the [Add Device] button.
   The "Modbus_COM" object will be added to the "Device" object.



3. Right-click the "Modbus_COM" object in the navigator pane and then select Add Device from the context-sensitive menu that is displayed.
   The "Add Device" dialog box will be displayed.



4. Select "Modbus Serial Device" under "Modbus Serial Device" and click the [Add Device] button.
   The "Modbus_Serial_Device" object will be added below the "Modbus COM" object.

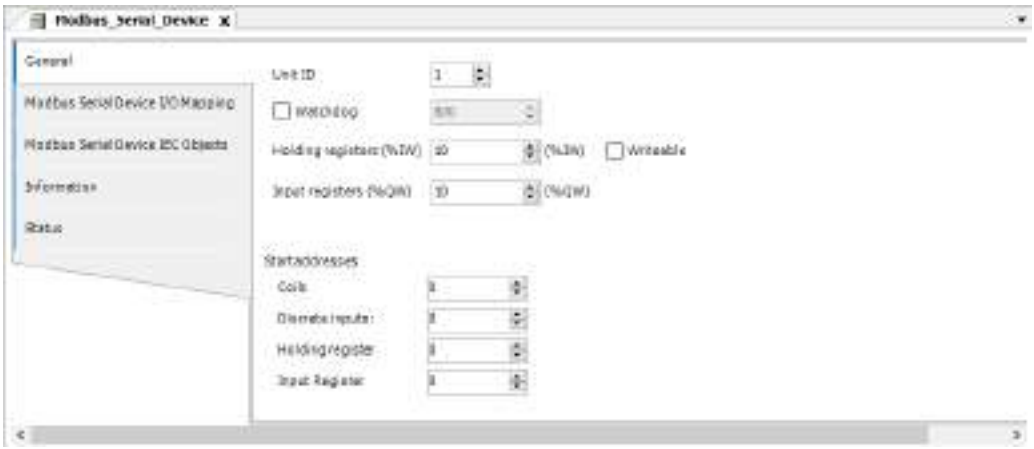


5. Double-click "Modbus_COM" in the navigator pane.
   The "Modbus_COM"object will be displayed.
   Select the "General"tab and set appropriate values for each item.

Item	Description
COM port	Settable value: 1 to 99 The COM port of the GM1 controller is fixed at 1.
Baud rate	Can be selected from 9600, 19200, 38400, 57600, and 115200
Parity	Can be selected from EVEN, ODD, and NONE
Data bits	Data bit length between start bit and stop bit Settable value: 7 bits and 8 bits (Settable: 0 to 255) The default value of Modbus is 8 bits.
Stop bit	Settable value: 1 bit and 2 bits The default value of Modbus is 1 bit.

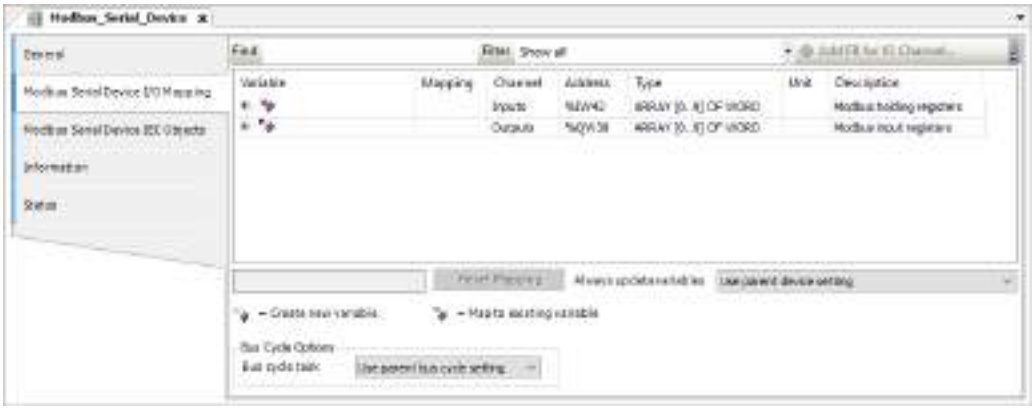
6. Double-click "Modbus_Serial_Device" in the navigator pane.
   The "Modbus_Serial_Device"object will be displayed.

Select the "General"tab and set appropriate values for each item.



Item	Description
Watchdog	If no valid command is received from the master during the preset time period, the holding register (%IW) will be set to zero. Settable value: 500 to 200000
Unit ID	A station number is set. Settable value: 1 to 247
Holding registers (%IW)	The number of holding registers is set. Buffer size of holding register: 1 to 500
Input registers (%QW)	The number of input registers is set. Buffer size of input register: 1 to 500

7. Select the "Modbus Serial Device I/O Mapping" tab.
You can allocate variables to holding registers and input registers.



Registers correspond to each access type (function code)

Function code	Access type	Description
1	Read Coils	Holding register
2	Read Discrete Inputs	Input register

Function code	Access type	Description
3	Read Holding Registers	Holding register
4	Read Input Registers	Input register
5	Write Single Coil	Holding register
6	Write Single Register	Holding register
15	Write Multiple Coils	Holding register
16	Write Multiple Registers	Holding register
23	Read/Write Multiple Registers	Holding register (Read/Write)

13.4 EtherNet/IP

13.4.1 What is EtherNet/IP?

EtherNet/IP (Ethernet Industrial Protocol) is an industrial multi-vendor real-time Ethernet system that executes a communication protocol for controlling the Common Industrial Protocol (CIP) in the application layer implemented over standard Ethernet.

For details on CIP, refer to ODVA documentation.

13.4.2 Cyclic Communication Function

After the scanner device connects to an adapter device and a connection is established, the cyclic communication function allows them to send data mutually at the requested packet interval (RPI).

- Scanner device: Controllers such as PLC
- Adapter device: Robot controllers, encoders, I/O devices, etc.

The EtherNet/IP function of the GM1 controller consists of the scanner function and adapter function.

Supplementary note: About GM1 controllers used as adapter devices

- Adapter devices are classified into the following two types: Adapter devices connected under the control of the GM1 controller used as a scanner device and adapter devices that are the GM1 controller itself. To avoid confusion, the following two different terms are used in this manual.
  - Local adapter
    Adapter device that is the GM1 controller itself
  - Remote adapter
    Adapter device connected to the GM1 controller used as a scanner

13.4.3 EtherNet/IP Scanner Function

The EtherNet/IP scanner function allows the GM1 controller to communicate with EtherNet/IP adapter devices.

Communication settings for an adapter device to be connected can be configured by loading the EDS file of the adapter device. Multiple adapter devices can be connected to a scanner device.

13.4.4 Setting up the EtherNet/IP Scanner Function

This section explains how to set up the EtherNet/IP scanner function.

Adding devices

Add an EtherNet/IP scanner device and remote adapter device to the Device tree, as described below.

Procedure

1. Add an EtherNet/IP scanner device.

   1-1 Right-click the "LANPort2" object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.

   The "Add Device" dialog box will be displayed.

   1-2 Select "EtherNet_IP_Scanner" and click the [Add Device] button.

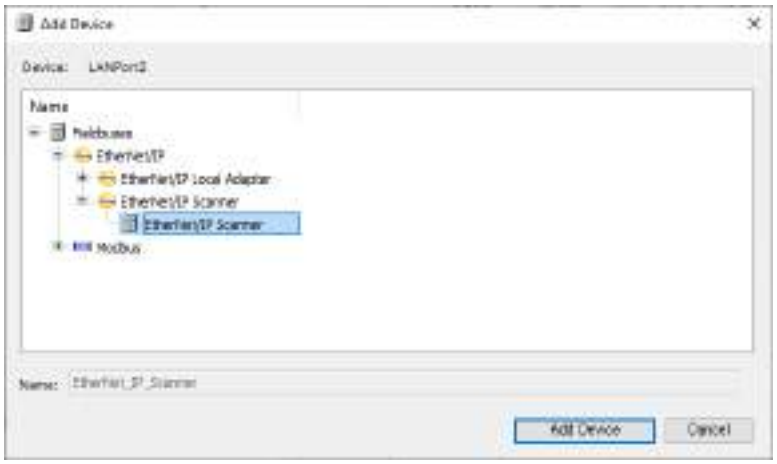
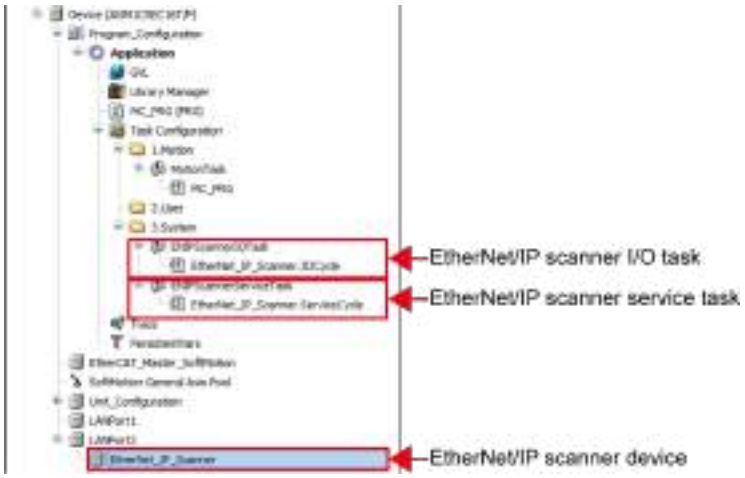


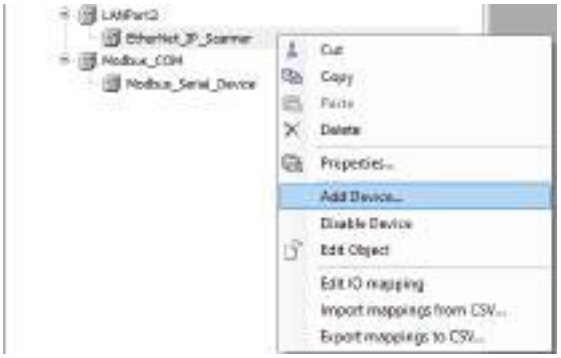
Image of added device and tasks

After an EtherNet/IP scanner device has been added, a device and tasks are added to the Device tree, as shown below.



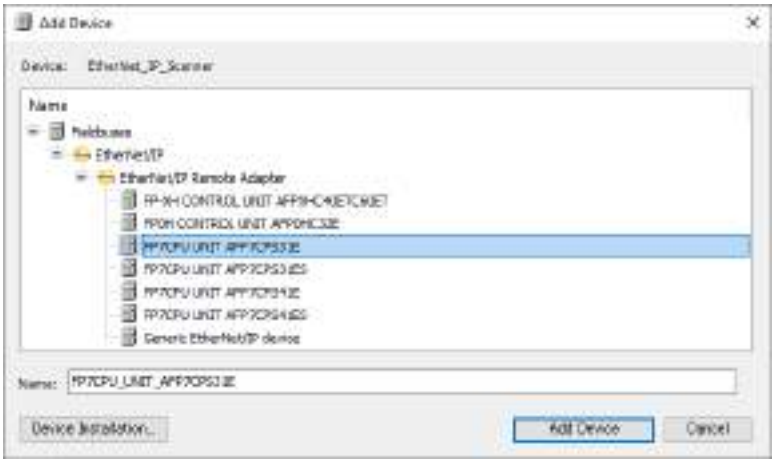
2. Add a remote adapter device.

2-1 Right-click the "EtherNet/IP scanner device" object added in "Step 1" and select "Add Device" from the context-sensitive menu that is displayed.



The "Add Device" dialog box will be displayed.

2-2 Select a remote adapter device to be added and click the [Add Device] button.



A new remote adapter device can also be added by selecting an EDS file. Click the [Install Device] button and select a desired EDS file.

Device tree after devices are added

Setting up an EtherNet/IP scanner device

Set up an EtherNet/IP scanner device as below.

1 2 Procedure

1. Double-click "EtherNet_IP_Scanner" in the navigator pane.
2. In the "General" tab, select the "Auto-reestablish connections" check box.
   When the check box is not selected: The device is stopped in the event of a communication error.
   When the check box is selected: The device is reconnected automatically in the event of a communication error.

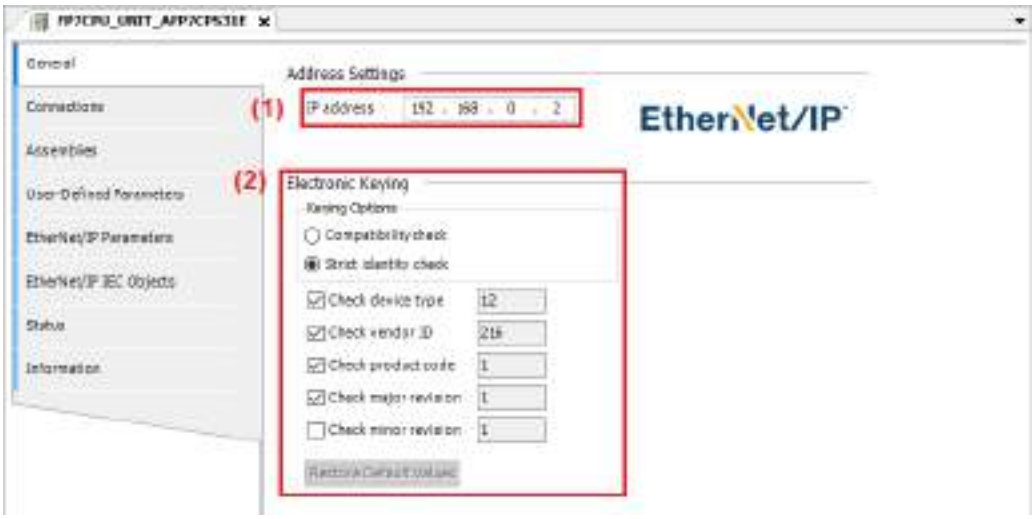
Setting up a remote adapter device

Set up a remote adapter device, as below.

Setting items for remote adapter devices differ according to the EDS file. The following procedure is explained, using Panasonic "AFP7CPS31E" as an example.

1 2 Procedure

1. Double-click "FP7CPU_UNIT_AFP7CPS31E" in the navigator pane.
2. In the "General" tab, set an IP address and items to be checked at the time of connection.



(1) IP address
Set the IP address of the adapter device.

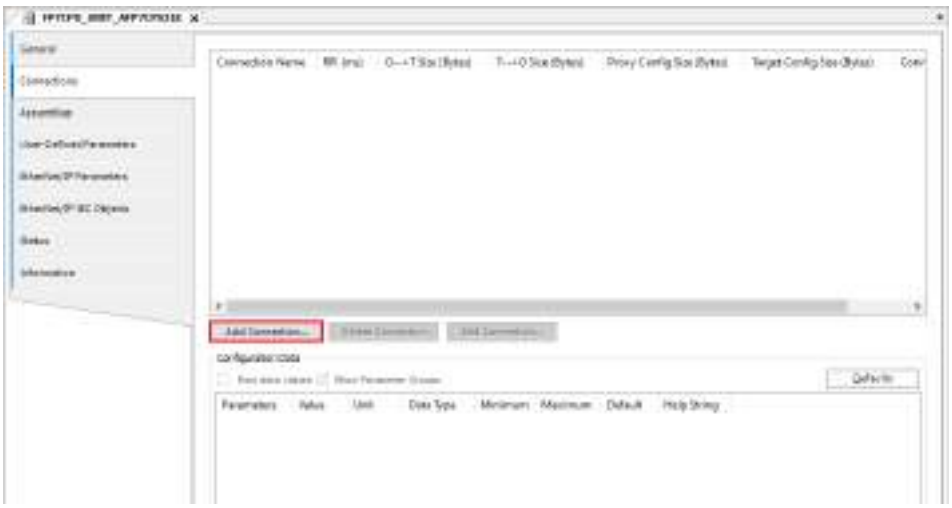
(2) Electronic Keying
Select items to be checked at the time of connection.

- Compatibility check
  The adapter device executes its own compatibility check.
  In this case, the user can select only the "Check device type" check box.
- Strict identity check
  The user specifies check items individually.

Normally, it is OK to use the default values.

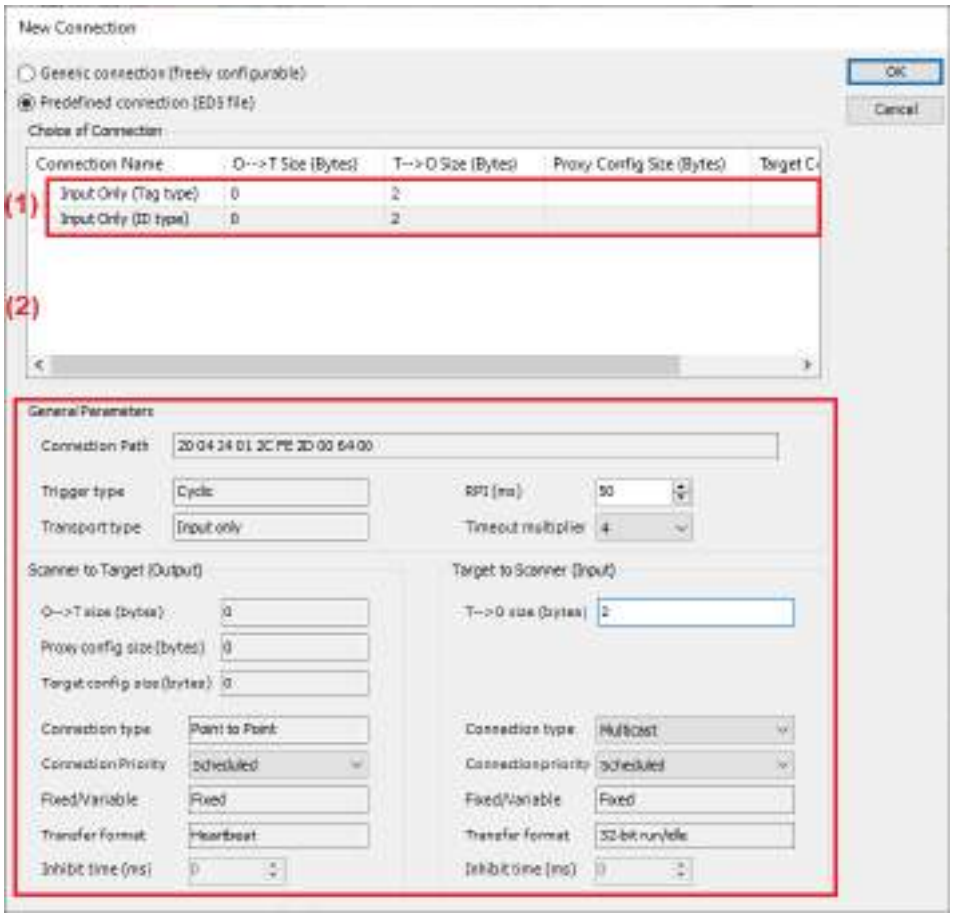
3. Set a connection point.

3-1 In the "Connection" tab, click the [Add Connection] button.



The "New Connection" window will be displayed.

3-2 Set up parameters required for connection.



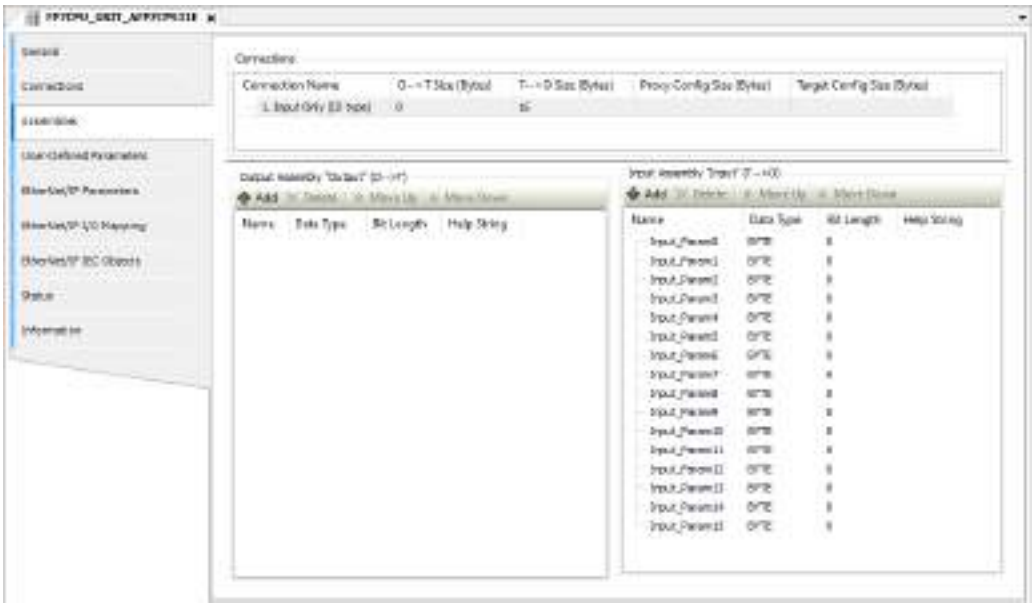
(1) This section displays the connection points that are supported by the adapter device according to the EDS file.
Select a connection point to be used.
Example: Select "Input Only (ID type)".

(2) The parameters in the "General parameters" section differ according to the selected connection point.
Example: Set "RPI" to 10 (ms) and "T→O size" to 16 (bytes).

Info.

- For tag connection, uppercase English letters cannot be used for tag names.

4. In the "Assemblies" tab, set up a data configuration.
If "T→O size" in the "General parameters" section is set to 16 bytes, the default data configuration will be as shown below.



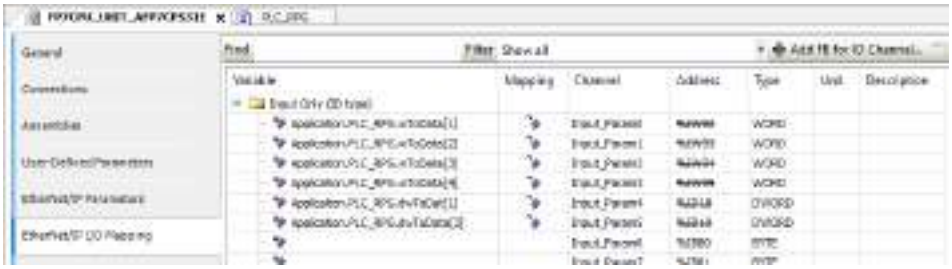
To change the data structure, click a relevant data type and select a desired data type.

In the following example, the BYTE type (16-byte data structure) is changed to the WORD type (4-word data structure) and the DWORD type (2-word data structure).

5. In the "EtherNet/IP I/O Mapping" tab, map data to variables in the application POU, as below.

5-1 In the POU, create variables to which data is to be mapped.

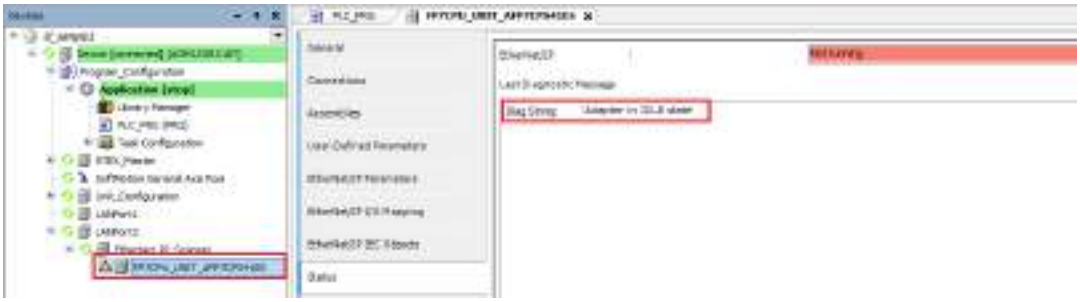
5-2 Map I/O data to each variable that has been created.



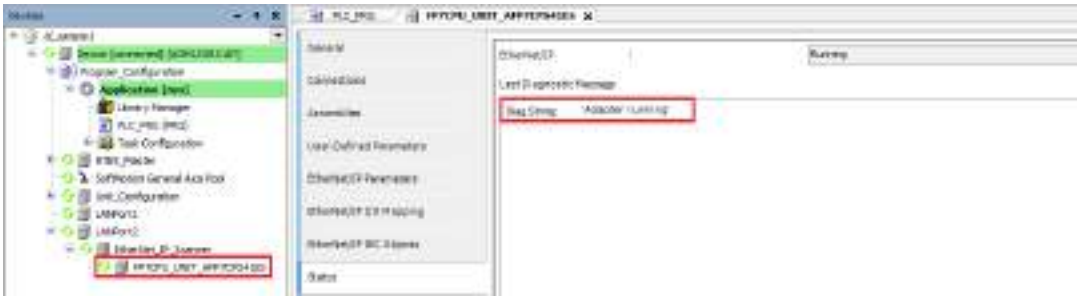
13.4.5 EtherNet/IP Scanner Operation

When a project in which EtherNet/IP scanner settings have been configured is downloaded to the GM1 controller and then an adapter is connected, cyclic communication is started, regardless of whether the GM1 controller is set to RUN or STOP mode.

When the GM1 controller is set to STOP mode, the remote adapter is placed in "Adapter in IDLE state" and displayed as △ in the Device tree.



When the GM1 controller is set to RUN mode, the remote adapter is placed in "Adapter running" and displayed as ○ in the Device tree.



To perform a status check or reset using a program, use I/O of the remote adapter device.

The following is an example of adapter operation using device I/O.

Declaration section (common to ST and LD programming languages)

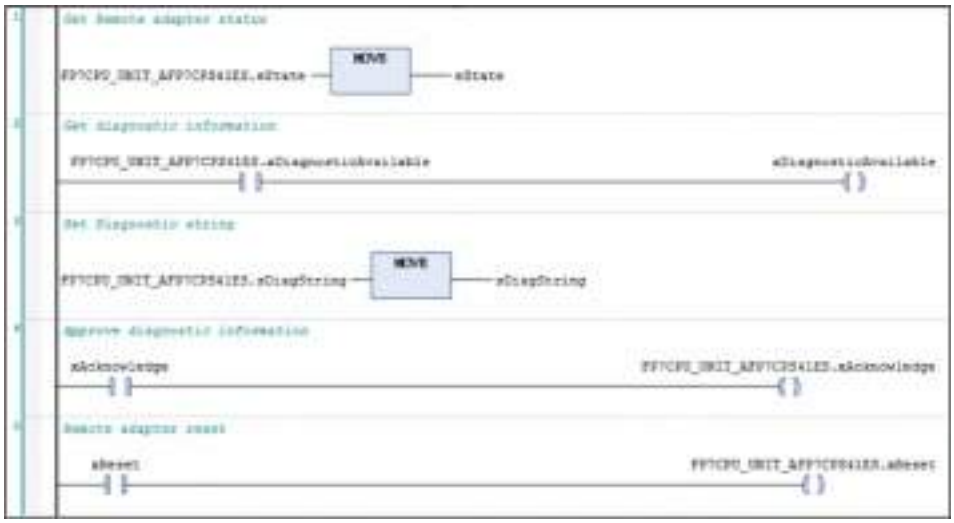
```
PROGRAM PLC_PRG
VAR
                    eState                  :
                                            IoDrvEtherNetIP.AdapterStat
                                            e;
                                                                    // Remote adapter status
```

```
                xDiagnosticAvailable    : BOOL;          // TRUE if there is diagnostic
                                                            information
                sDiagString             : STRING;        // Diagnostic string
                xAcknowledge            : BOOL := FALSE; // Approve diagnostic information
                xReset                  : BOOL := FALSE; // Remote adapter reset
END_VAR
```

Implementation section (ST programming language)

```
eState := FP7CPU_UNIT_AFP7CPS41ES.eState;

xDiagnosticAvailable := FP7CPU_UNIT_AFP7CPS41ES.xDiagnosticAvailable;
sDiagString := FP7CPU_UNIT_AFP7CPS41ES.sDiagString;
FP7CPU_UNIT_AFP7CPS41ES.xAcknowledge := xAcknowledge;

FP7CPU_UNIT_AFP7CPS41ES.xReset := xReset;
```

Implementation section (LD programming language)



The current state of the remote adapter is stored in eState.

Example	When line is connected normally:	RUNNING
	When line is disconnected:	ENCAPSULATION_CONFIG

If an error occurs, xDiagnosticAvailable will be set to TRUE and a message will be found in sDiagString.

If xAcknowledge is set to TRUE, xDiagnosticAvailable will return to FALSE.

If xReset is set to TRUE, the line will be closed temporarily and then reconnected.

(To reset all remote adapters simultaneously, use xReaset of the EtherNet/IP scanner device.)

13.4.6 EtherNet/IP Adapter Function

The EtherNet/IP adapter function allows the GM1 controller to communicate with EtherNet/IP scanner devices.

13.4.7 Setting up the EtherNet/IP Adapter Function

This section explains how to set up the EtherNet/IP adapter function.

Adding devices

Add a local adapter device and module device to the Device tree, as described below.

1 2 Procedure

1. Add a local adapter device.
A local adapter device serves as a connection point to which the scanner device connects.

1-1 Right-click the "LANPort2" object in the navigator pane and then select "Add Device" from the context-sensitive menu that is displayed.

The "Add Device" dialog box will be displayed.

1-2 Select "EtherNet_IP_Adapter" and click the [Add Device] button.

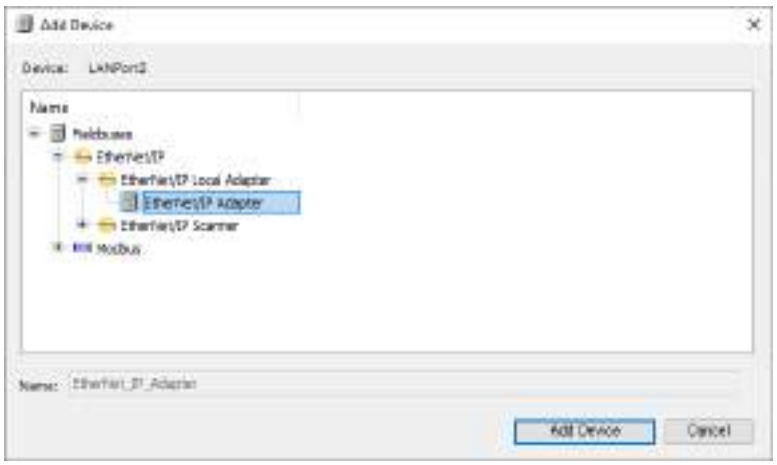
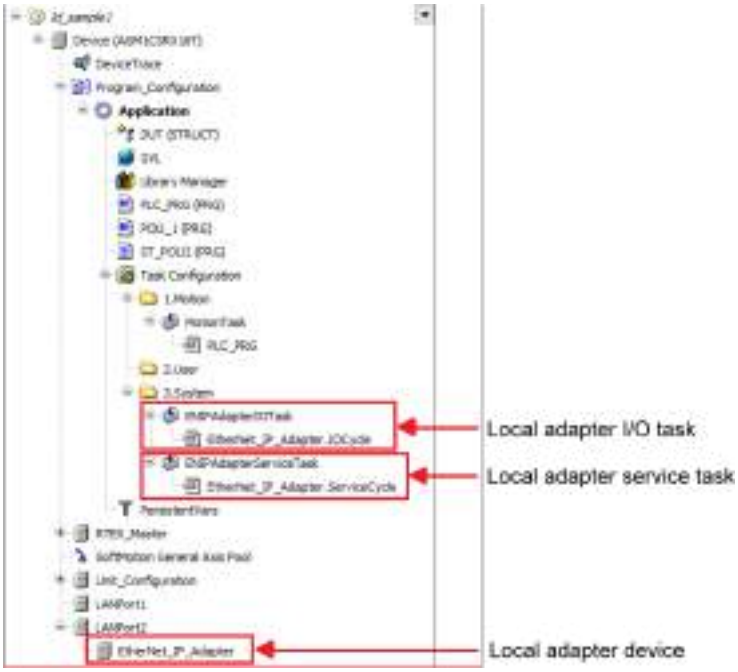


Image of added device and tasks

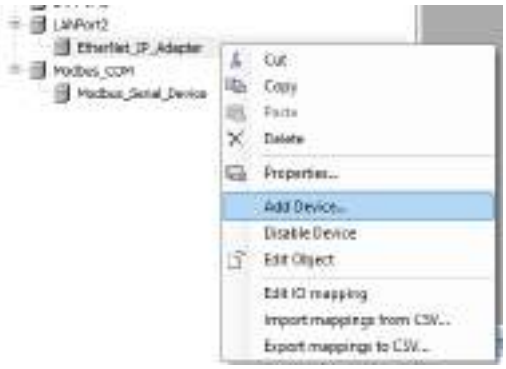
After a local adapter device has been added, a device and tasks are added to the Device tree, as shown below.



2. Add a module device.

A module device defines data to be transferred via cyclic communication.

2-1 Right-click the "Local adapter device" object added in "Step 1" and select "Add Device" from the context-sensitive menu that is displayed.



The "Add Device" dialog box will be displayed.

2-2 Select the "EtherNet/IP Module" object to be added and click the [Add Device] button.

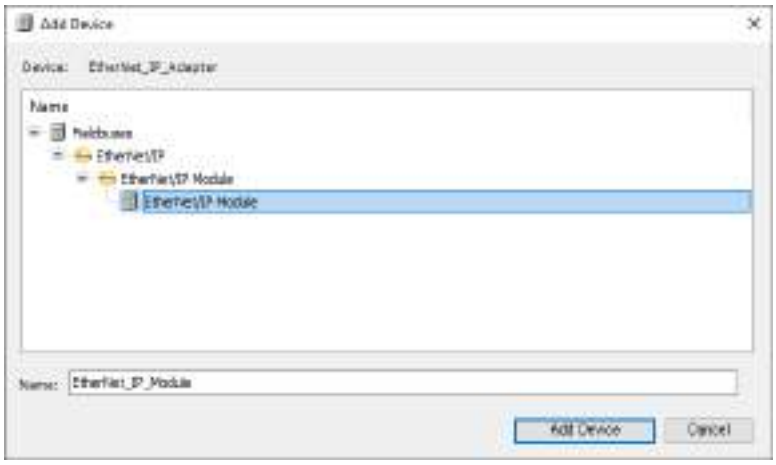


Image of added devices

Multiple module devices can be added within a local adapter device.

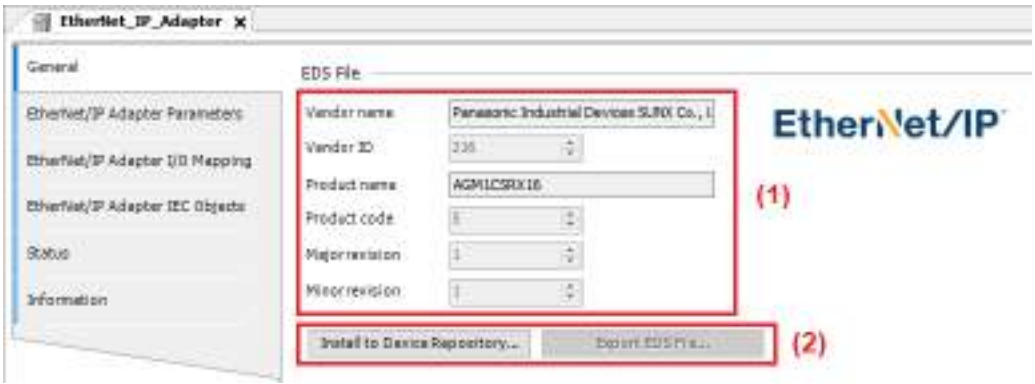
The following is an example of four module devices added to a local adapter device.

Settings of local adapter device

Check the settings of the local adapter device.

Procedure

1. Double-click "EtherNet_IP_Adapter" in the navigator pane.
2. In the "General" tab, check the settings of the local adapter device.



(1) The settings of the local adapter device are shown below. However, the settings cannot be changed.

Vendor name: Panasonic Industrial Devices SUNX Co., Ltd.
Vendor ID: 216
Product name: AGM1CSEC16
Product code: 6
Major revision: 1
Minor revision: 1

(2) For the following items, the settings cannot be changed.

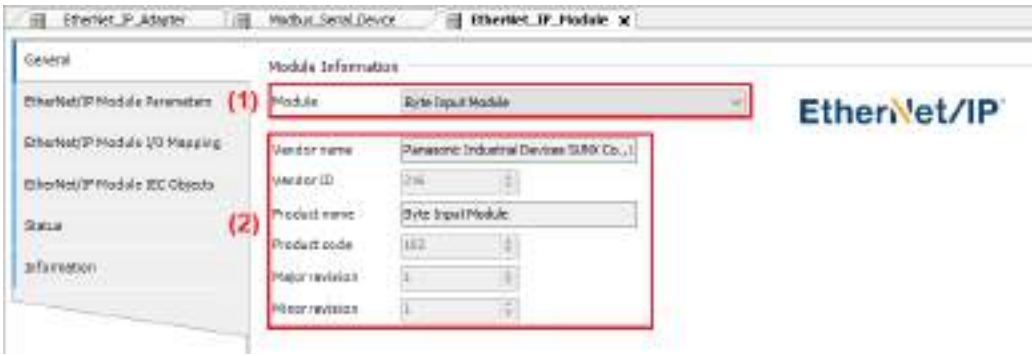
- Install to Device Repository
  The remote adapter device specified here is registered as a device in CODESYS.
- Export EDS File
  The EDS file specified here is output.
  For scanner device settings, use the EDS file provided by Panasonic ("PanasonicGM1CSEC16_0006_0101.eds").

Setting up a module device

Set up a module device, as below.

Procedure

1. Double-click "EtherNet_IP_Module" in the navigator pane.
2. In the "General" tab, set module information.



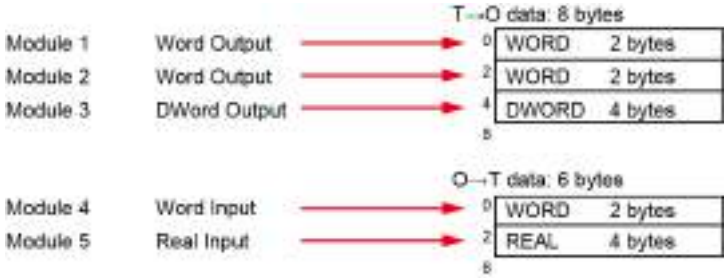
2-1 (1) Select a desired module type from the following 10 types.

Module type	Size	Direction
Byte Input	1 byte	O→T
Byte Output	1 byte	T→O
Word Input	1 word (2 bytes)	O→T
Word Output	1 word (2 bytes)	T→O
DWord Input	1 double-word (4 bytes)	O→T
DWord Output	1 double-word (4 bytes)	T→O
Real Input	1 single-precision real number (4 bytes)	O→T
Real Output	1 single-precision real number (4 bytes)	T→O

Module type	Size	Direction
Big Input	509 bytes	O→T
Big Output	505 bytes	T→O

By generating multiple module devices and setting module types, data structure can be created within cyclic data.

Example) T→O data: 8 bytes, O→T data: 6 bytes



Note

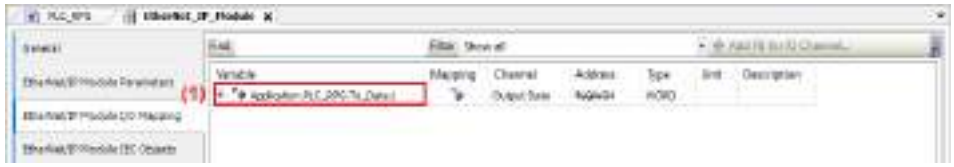
- The maximum data length within a single connection point is as follows:
  O→T data: 509 bytes
  T→O data: 505 bytes

2-2 There is no need to set all items in (2), as values cannot be entered.

3. In the "EtherNet/IP Module I/O Mapping" tab, map data to variables in the application POU, as below.

3-1 In the POU, create variables to which data is to be mapped.

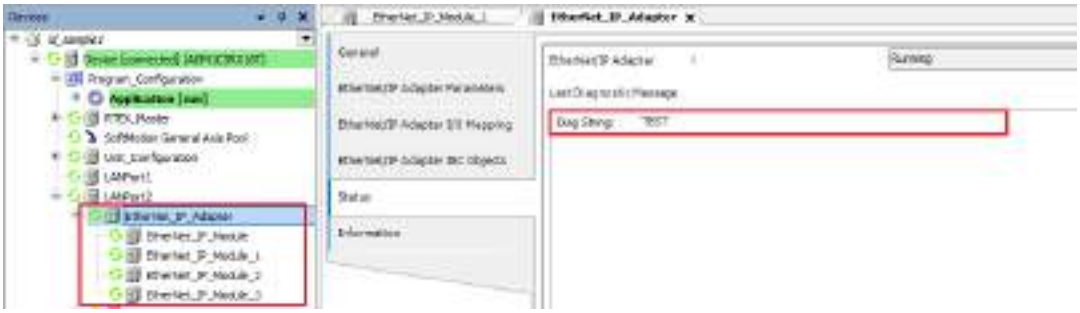
3-2 Map I/O data to each variable that has been created. Double-click the section indicated by (1) and select a variable to which I/O data is to be mapped.



13.4.8 EtherNet/IP Adapter Operation

When a project in which EtherNet/IP adapter settings have been configured is downloaded to the GM1 controller and then RUN mode is invoked, the adapter device responds to a ForwordOpen packet from the scanner device and cyclic communication is started.

The following shows the respective states of the local adapter device and module devices during normal operation.



The local adapter device and module devices are displayed with "○" symbol in the Device tree and the status of the local adapter device is displayed as "Adapter running".

The local adapter device can be connected from multiple scanner devices.

Note

- ExclusiveOwner connection (using O→T data) is allowed for only one scanner device. This is to prevent the same variable from being overwritten with input data from multiple scanner devices.

14 Other Controller Functions

14.1 SD Card Access 14-2
14.1.1 Overview of SD Card Access Function 14-2
14.1.2 File Manipulations Using the CAA File Library 14-2
14.2 Time Function 14-10
14.2.1 Overview of Time Function 14-10
14.2.2 Settings Based on GM Programmer 14-10
14.2.3 Settings Based on Function Blocks 14-10
14.3 Trace Function 14-11
14.3.1 Setting up Trace 14-11
14.3.2 Executing Trace 14-14
14.4 Recipe Manager Functions 14-16
14.4.1 Setting the Recipe Manager 14-16
14.4.2 Setting the Recipe Definition 14-17
14.4.3 Recipe Operation Using the GM Programmer 14-19

14.1 SD Card Access

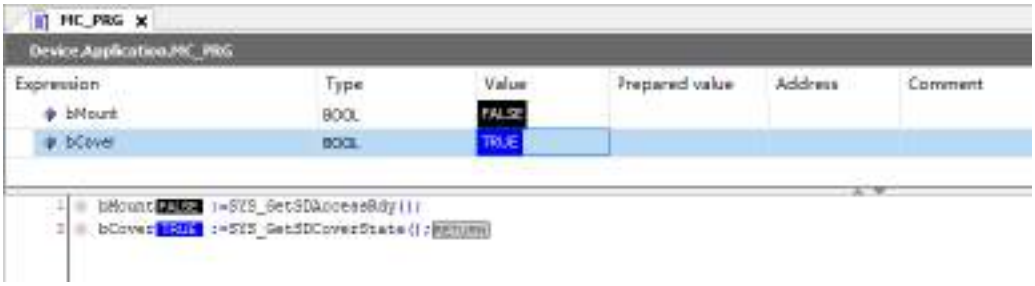
14.1.1 Overview of SD Card Access Function

The GM1 controller allows directories and files to be written to and read from the SD card via the CAA File library.

■ Preparation for SD card access

1. Before starting operation using the CAA File library, always use the following function of the Panasonic_GM_System library to check whether the SD card can be accessed.
   a) SYS_GetSDAccessRdy: Reads the mount state of the SD card
      TRUE: SD card is accessible. An SD card has been inserted.
      FALSE: SD card is inaccessible. No SD card has been inserted.
2. Considering situations such as removal of the SD card during operation, use the following function to check whether the SD card cover is open or closed.
   a) SYS_GetSDCoverState: Reads the open / closed state of the SD card slot cover
      TRUE: The SD card slot cover is closed.
      FALSE: The SD card slot cover is open.

   When the SD card slot cover is open, the SD card can be removed safely by stopping processing such as writing or reading directories or files using the CAA File library.



14.1.2 File Manipulations Using the CAA File Library

This section explains how to use the CAA File library to access files on the SD card, in the following order.

1. Library_Manager
2. Example of file write processing
3. Example of file read processing

1. Library_Manager

Check that the following CAA File library is registered in Library_Manager.

2. Example of file write processing

SampleDir/SampleFile.txt is created in the SD card and specified data is written to the file.

■ File write processing sequence

The file write processing sequence is shown below.

- File open processing (overwrite mode, insert mode)
  - Overwrite mode: For existing files, the contents of the file are cleared.
  - Insert mode: For existing files, the contents of the file are not cleared.
  - When write is executed, data is written following the end of the previous data.
- File write processing
- File close processing

■ Explanation of variables

uiProcess:

Executes processing when file open processing is set to 1 (overwrite mode) or 2 (insert mode). After the execution is completed, the variable is set to 0 (invalid value).

sFileName:

Specifies a directory or file name.

sWriteData:

Sets data to be written.

bResult:

Substitutes the result of processing execution. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

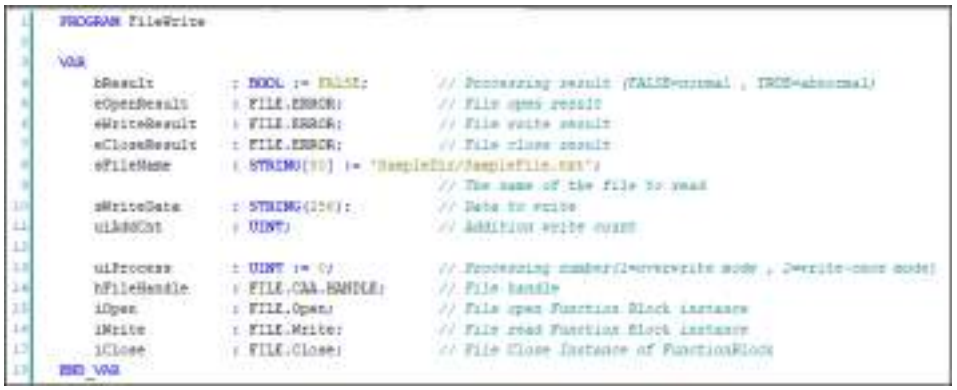
- eOpenResult: Result of file open processing
- eWriteResult: Result of file write processing
- eCloseResult: Result of file close processing

■ Operation example

- In this example, operations are performed by setting the value of uiProcess to 1, 2, and 2 in this order.
- The contents of SampleFile.txt which is output are as follows:

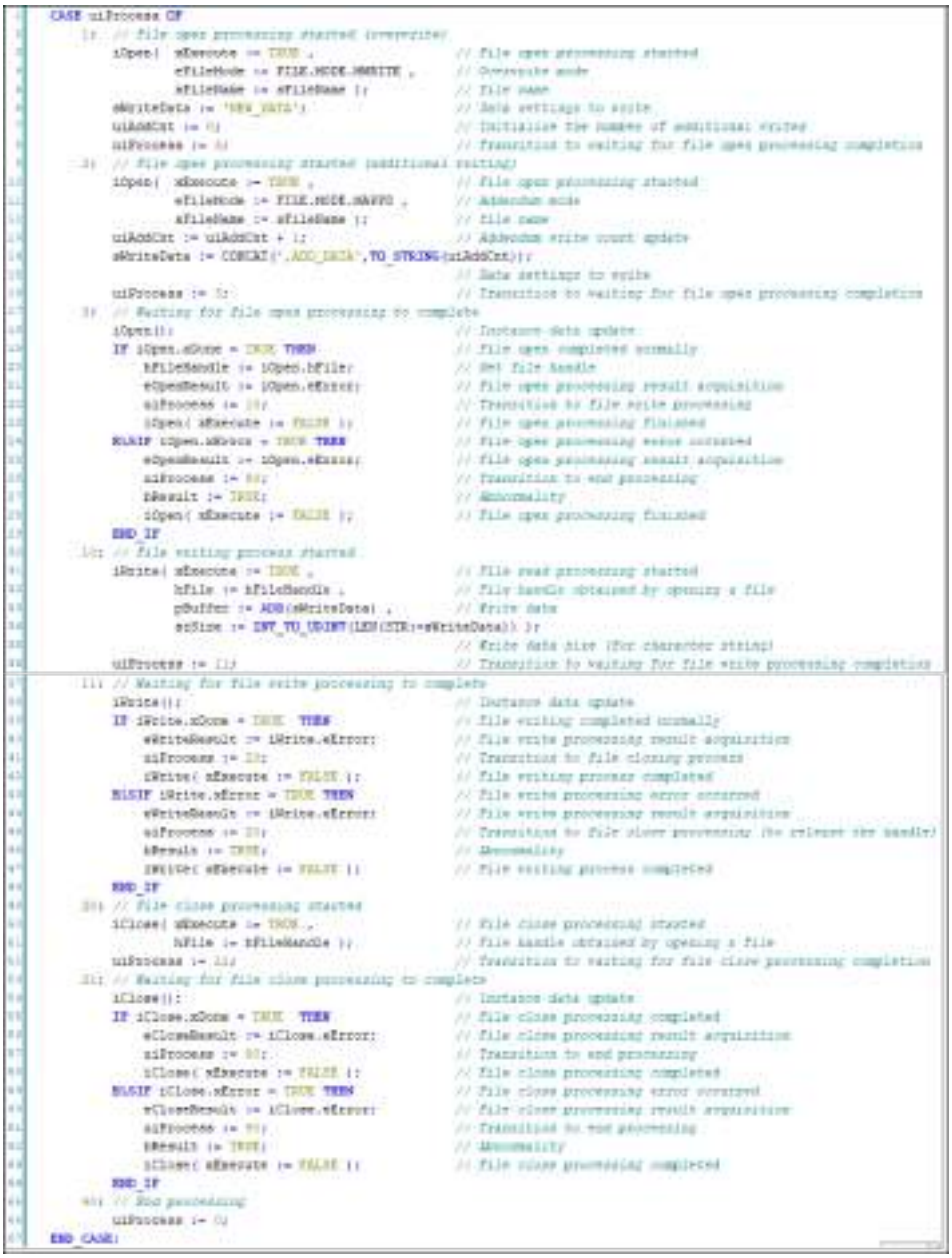
  NEW_DATA
  ADD_DATA1
  ADD_DATA2
- Sample program

Declaration section (common to ST and LD programming languages)



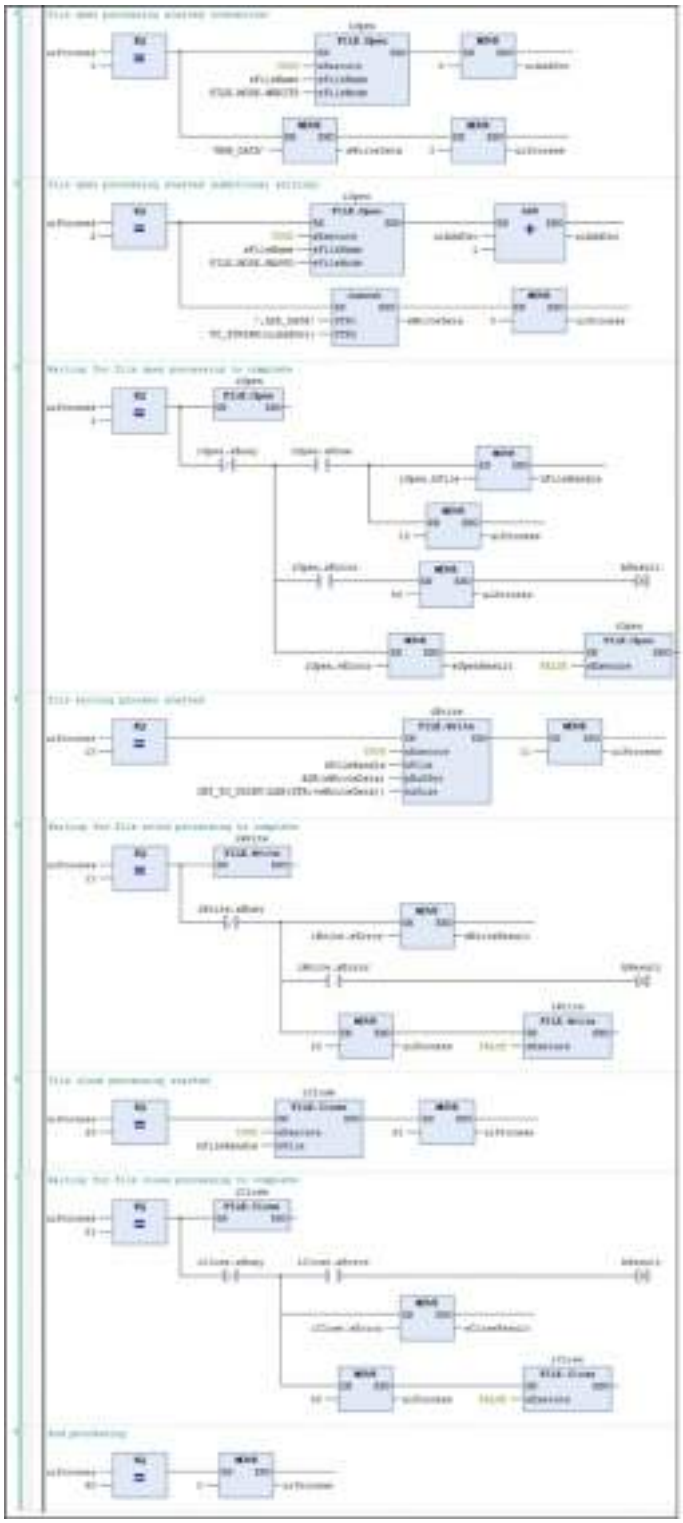
```
PROGRAM FileWrite

VAR
    bResult         : BOOL := FALSE;        // Processing result (FALSE=normal , TRUE=abnormal)
    eOpenResult     : FILE.ERROR;           // File open result
    eWriteResult    : FILE.ERROR;           // File write result
    eCloseResult    : FILE.ERROR;           // File close result
    sFileName       : STRING(80) := 'SampleDir/SampleFile.txt';
                                            // The name of the file to read
    sWriteData      : STRING(128);          // Data to write
    uiAddCnt        : UINT;                 // Addition write count

    uiProcess       : UINT := 0;            // Processing number(1=overwrite mode , 2=write-once mode)
    hFileHandle     : FILE.CAA.HANDLE;      // File handle
    iOpen           : FILE.Open;            // File open Function Block instance
    iWrite          : FILE.Write;           // File read Function Block instance
    iClose          : FILE.Close;           // File Close Instance of FunctionBlock
END_VAR
```

Implementation section (ST programming language)



```
CASE uiProcess OF
    1: // File open processing started (overwrite)
        iOpen( xExecute := TRUE ,                   // File open processing started
               eFileMode := FILE_MODE.nWRITE ,      // Overwrite mode
               sFileName := sFileName );            // File name
        sWriteData := 'SEN_DATA';                   // Data settings to write
        uiAddCnt := 0;                              // Initialize the number of additional writes
        uiProcess := 2;                             // Transition to waiting for file open processing completion
    2: // File open processing started (additional writing)
        iOpen( xExecute := TRUE ,                   // File open processing started
               eFileMode := FILE_MODE.nAPPD ,       // Addendum mode
               sFileName := sFileName );            // File name
        uiAddCnt := uiAddCnt + 1;                   // Addendum write count update
        sWriteData := CONCAT('ADD_DATA',TO_STRING(uiAddCnt));
                                                    // Data settings to write
        uiProcess := 3;                             // Transition to waiting for file open processing completion
    3: // Waiting for file open processing to complete
        iOpen();                                    // Instance data update
        IF iOpen.xDone = TRUE THEN                  // File open completed normally
            hFileHandle := iOpen.hFile;             // Get file handle
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 10;                        // Transition to file write processing
            iOpen( xExecute := FALSE );             // File open processing finished
        ELSIF iOpen.xError = TRUE THEN              // File open processing error occurred
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 99;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iOpen( xExecute := FALSE );             // File open processing finished
        END_IF
    10: // File writing process started
        iWrite( xExecute := TRUE ,                  // File read processing started
                hFile := hFileHandle ,              // File handle obtained by opening a file
                pBuffer := ADR(sWriteData) ,        // Write data
                cbSize := INT_TO_UDINT(LEN(STR:=sWriteData)) );
                                                    // Write data size (for character string)
        uiProcess := 11;                            // Transition to waiting for file write processing completion
    11: // Waiting for file write processing to complete
        iWrite();                                   // Instance data update
        IF iWrite.xDone = TRUE  THEN                // File writing completed normally
            eWriteResult := iWrite.eError;          // File write processing result acquisition
            uiProcess := 20;                        // Transition to file closing process
            iWrite( xExecute := FALSE );            // File writing process completed
        ELSIF iWrite.xError = TRUE THEN             // File write processing error occurred
            eWriteResult := iWrite.eError;          // File write processing result acquisition
            uiProcess := 99;                        // Transition to file close processing (to release the handle)
            bResult := TRUE;                        // Abnormality
            iWrite( xExecute := FALSE );            // File writing process completed
        END_IF
    20: // File close processing started
        iClose( xExecute := TRUE ,                  // File close processing started
                hFile := hFileHandle );             // File handle obtained by opening a file
        uiProcess := 21;                            // Transition to waiting for file close processing completion
    21: // Waiting for file close processing to complete
        iClose();                                   // Instance data update
        IF iClose.xDone = TRUE  THEN                // File close processing completed
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 99;                        // Transition to end processing
            iClose( xExecute := FALSE );            // File close processing completed
        ELSIF iClose.xError = TRUE THEN             // File close processing error occurred
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 99;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iClose( xExecute := FALSE );            // File close processing completed
        END_IF
    99: // End processing
        uiProcess := 0;
END_CASE;
```

Implementation section (LD programming language)



3. Example of file read processing

Data in SampleDir/SampleFile.txt in the SD card is read into the buffer.

The effective range of data read into the buffer is judged from the data size information that is output after read processing.

■ File read processing sequence

The file read processing sequence is shown below.

- File open processing (read mode)
- File read processing
- File close processing

■ Explanation of variables

uiProcess:

Executes processing when the variable is set to 1 (read mode).

sFileName:

Specifies a directory or file name.

sReadData:

Sets a buffer into which data is to be read.

szReadSize:

Stores the size of read data after read processing.

bResult:

Substitutes the result of processing execution. (TRUE: Error occurrence, FALSE: Normal termination)

If the result of processing execution is abnormal, check the error code of each processing.

- eOpenResult: Result of file open processing
- eReadResult: Result of file read processing
- eCloseResult: Result of file close processing

■ Operation example

- In this example, operations are performed according to the following contents of SampleFile.txt.

  NEW_DATA
  ADD_DATA1
  ADD_DATA2
- Read data and data size are as below.

  Data (STRING type): 'NEW_DATARNADD_DATA1RNADD_DATA2RNADD_DATA3'
  Data size: 41
- Sample program

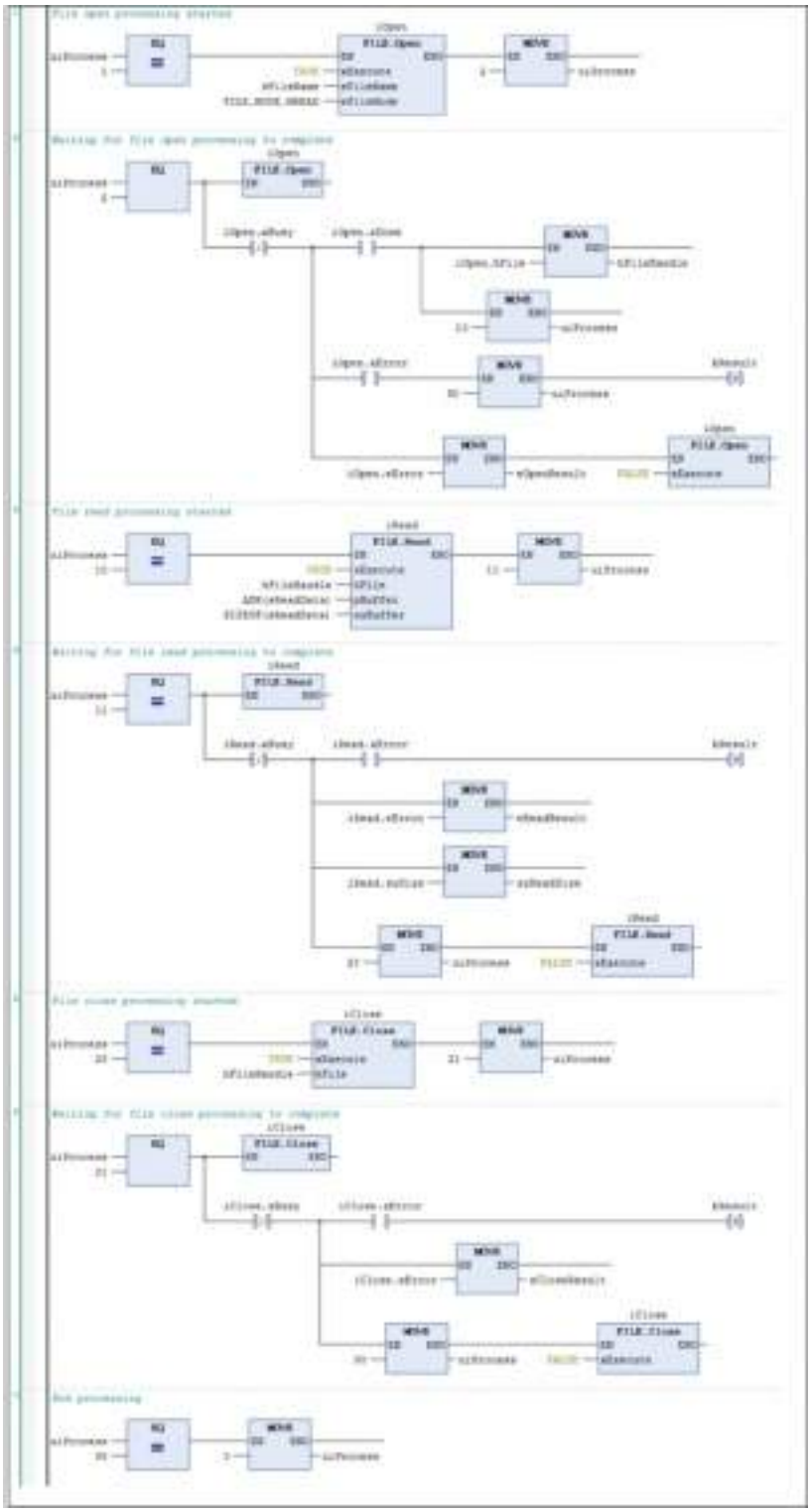
Declaration section (common to ST and LD programming languages)

```
PROGRAM FileRead

VAR
    bResult         : BOOL := FALSE;        // Processing result (FALSE=normal , TRUE=abnormal)
    eOpenResult     : FILE.ERROR;           // File open result
    eReadResult     : FILE.ERROR;           // File read result
    eCloseResult    : FILE.ERROR;           // File close result
    sFileName       : STRING[80] := 'SampleDir/Samplefile.txt';
                                            // The name of the file to read
    cbReadSize      : FILE.CBA.SIZE;        // Read data size
    sReadData       : STRING[256];          // Read data

    uiProcess       : UINT := 0;            // Process number(1=read mode)
    hFileHandle     : FILE.CBA.HANDLE;      // File handle
    iOpen           : FILE.Open;            // File open Function Block instance
    iRead           : FILE.Read;            // File read Function Block instance
    iClose          : FILE.Close;           // File Close Instance of FunctionBlock
END_VAR
```

Implementation section (ST programming language)

```
CASE uiProcess OF
    0: // File open processing started
        iOpen( xExecute := TRUE ,                   // File open processing started
               sFileMode := FILE.MODE.READ ,        // Read mode
               sFileName := sFileName );            // File name
        uiProcess := 10;
    10: // Waiting for file open processing to complete
        iOpen();                                    // Instance data update
        IF iOpen.xDone = TRUE THEN                  // File open completed normally
            hFileHandle := iOpen.hFile;             // Get file handle
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 11;                        // Transition to file read processing
            iOpen( xExecute := FALSE );             // File open processing finished
        ELSIF iOpen.xError = TRUE THEN              // File open processing error occured
            eOpenResult := iOpen.eError;            // File open processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iOpen( xExecute := FALSE );             // File open processing finished
        END_IF
    11: // File read processing started
        iRead( xExecute := TRUE ,                   // File read processing started
               hFile := hFileHandle ,               // File handle obtained by opening a file
               pBuffer := ADR(sReadData) ,          // Data storage buffer address
               nBuffer := SIZEOF(sReadData) );      // Data storage buffer size
        uiProcess := 12;
    12: // Waiting for file read processing to complete
        iRead();                                    // Instance data update
        IF iRead.xDone = TRUE THEN                  // File read normal end
            eReadResult := iRead.eError;            // File read processing result acquisition
            cbReadSize := iRead.cbSize;             // Get read size
            uiProcess := 20;                        // Transition to file read processing
            iRead( xExecute := FALSE );             // File read processing completed
        ELSIF iRead.xError = TRUE THEN              // File read processing error occurred
            eReadResult := iRead.eError;            // File read processing result acquisition
            cbReadSize := 0;                        // Read size initialization
            uiProcess := 20;                        // Transition to file close processing (to release the handle)
            bResult := TRUE;                        // Abnormality
            iRead( xExecute := FALSE );             // File read processing completed
        END_IF
    20: // File close processing started
        iClose( xExecute := TRUE ,                  // File close processing started
                hFile := hFileHandle );             // File handle obtained by opening a file
        uiProcess := 21;                            // Transition to waiting for file close processing completion
    21: // Waiting for file close processing to complete
        iClose();                                   // Instance data update
        IF iClose.xDone = TRUE THEN                 // File close processing completed
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            iClose( xExecute := FALSE );            // File close processing completed
        ELSIF iClose.xError = TRUE THEN             // File close processing error occurred
            eCloseResult := iClose.eError;          // File close processing result acquisition
            uiProcess := 90;                        // Transition to end processing
            bResult := TRUE;                        // Abnormality
            iClose( xExecute := FALSE );            // File close processing completed
        END_IF
    90: // End processing
        uiProcess := 0;
END_CASE
```

Implementation section (LD programming language)



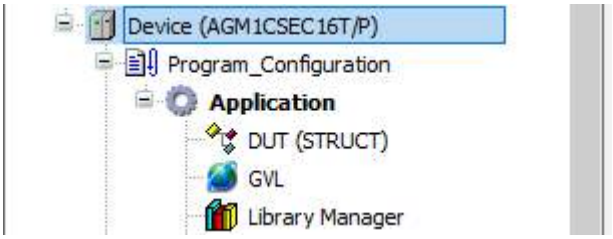
14.2 Time Function

14.2.1 Overview of Time Function

This section explains the time function that uses date and time settings in GM Programmer and function blocks.

14.2.2 Settings Based on GM Programmer

Date and time settings and monitoring for the GM1 controller can be performed using the "Date and Time Settings" window of GM Programmer. Connect to the GM1 controller and select the "Date and Time Settings" tab of the "Device" window.



Date and Time Settings pane

The "Device Date and Time" frame on the left side of the pane displays the date / time information of the GM1 controller. To set date and time, enter date and time in the "Date and Time" frame on the right side of the pane and click the [Update] button. The entered date and time will be set in the GM1 controller.

If the "Get date / time from PC" check box is selected, the date/time information of the PC will be automatically set in the GM1 controller. In this case, the time zone of the PC will be automatically set in the GM1 controller.

14.2.3 Settings Based on Function Blocks

For details on how to use this function, refer to the *GM1 Series Reference Manual (Instruction)*.

14.3 Trace Function

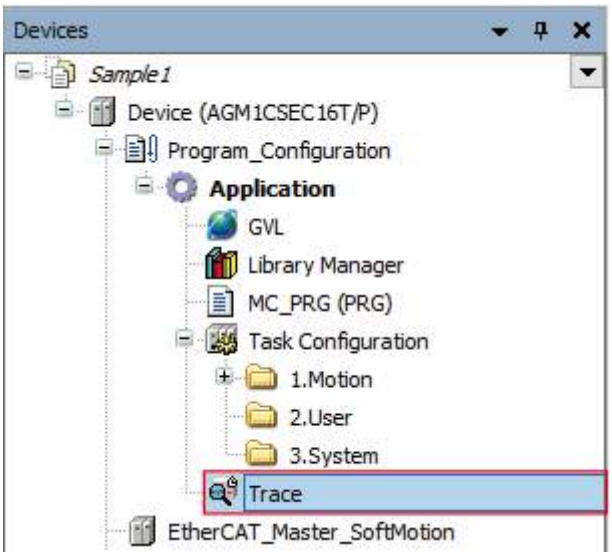
The trace function allows the variable value histories in the GM1 controller to be checked in GM Programmer. Trace start timing can be specified with options. Recorded data can be checked in the form of a graph. Data obtained by the trace function can also be saved in XML, text, or CSV format.

14.3.1 Setting up Trace

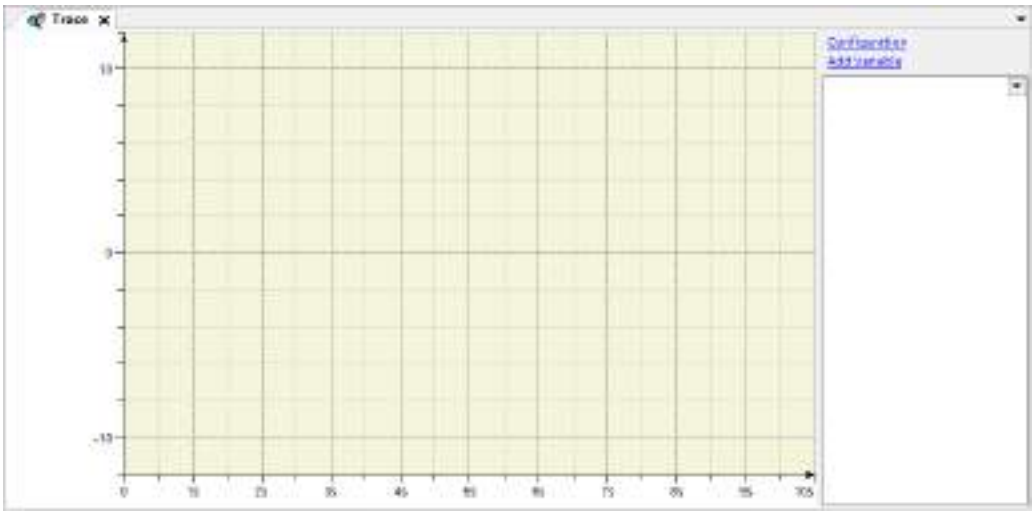
To use the trace function, after adding a trace object, you must register variables to be traced.

1 2 Procedure

1. Double-click the Trace object in the main pane.

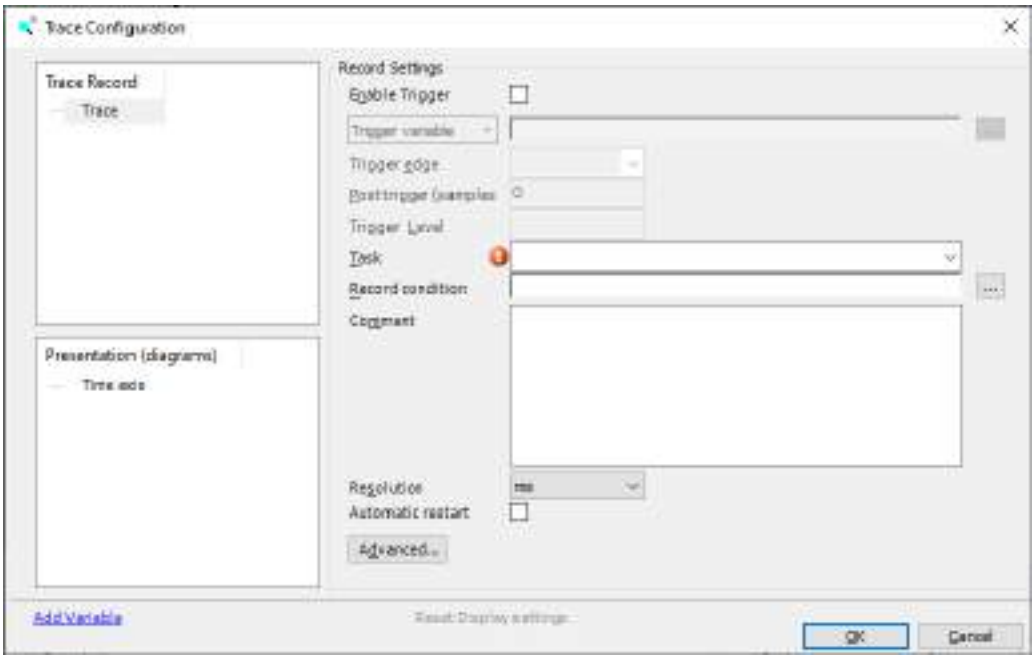


A trace object will be added.

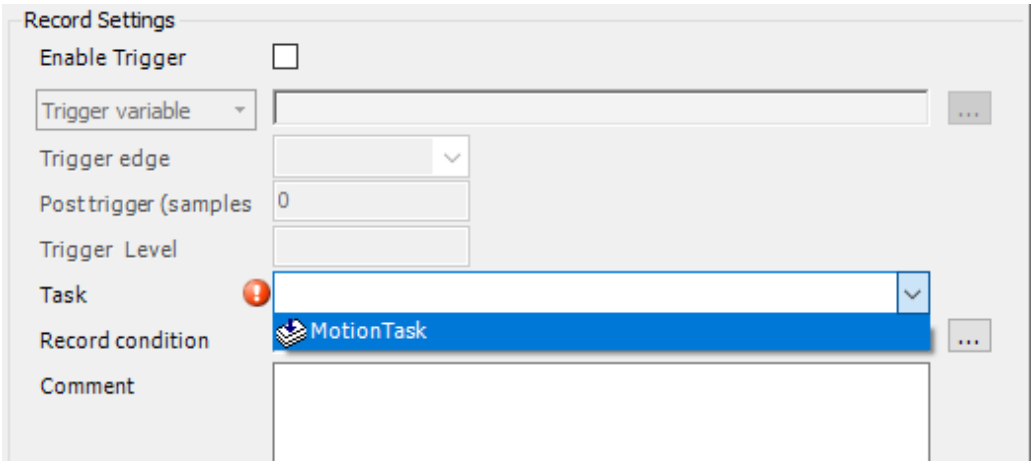


This completes the procedure for adding a trace object.
Next, the procedure for registering a task and variable to be traced is explained.

2. Click the "Configuration" link.
The "Trace Configuration" dialog box will appear with the "Record Settings" pane displayed.

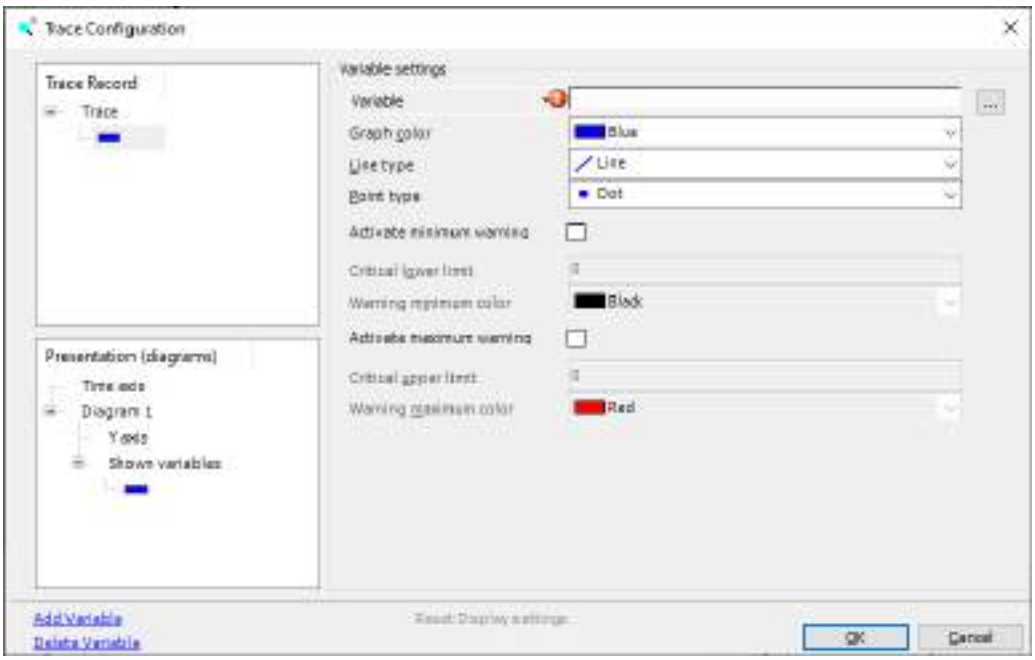


3. In **Record settings>Task** field, select a task to be traced.
You can select a task from a list of tasks registered in the project.
Select a task that you want to trace.



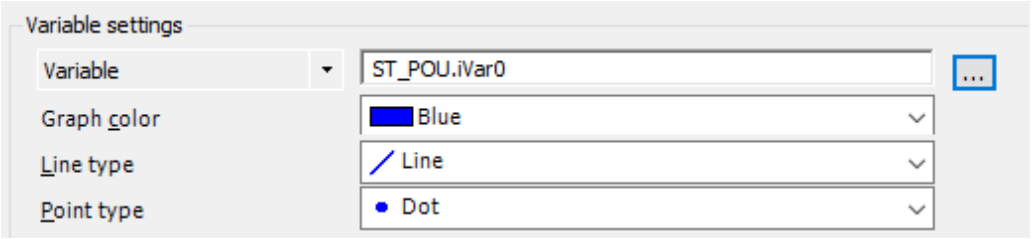
4. Click the [OK] button.
   The selected task will be registered in the trace object.

5. Click the "Add variable" link.
   The "Trace Configuration" dialog box will appear with the "Variable settings" pane displayed.



6. In **Variable settings**>**Variable** field, enter a variable to be traced.
   Either directly enter a variable to be traced or click the ... button to display the "Input Assistant" dialog box and then select a desired variable from the dialog box.
   In the Variable settings field, you can set a graph color, types, and other items for the variable.

Example: Entering variable iVar0 for ST_POU object



7. Click the [OK] button.
   The entered variable will be registered in the trace object.

8. Repeat steps "Step 5" to "Step 7" to register other variables to be traced.
   The entered variable will be registered in the trace object.

Example: Entering variables iVar1 and iVar2 for ST_POU object

This completes the procedure for registering variables to be traced in the trace object.

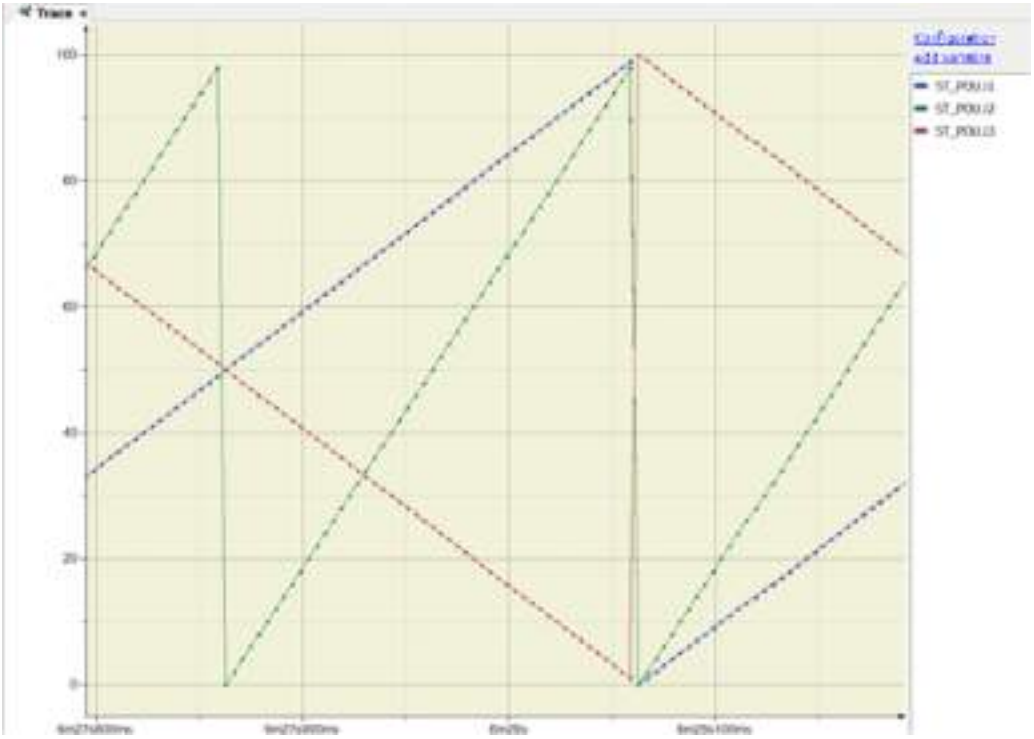
14.3.2 Executing Trace

By executing a trace, you can check the variables registered in the trace object.

Procedure

1. Connect the PC where GM Programmer is installed and the GM1 controller and log in to the GM1 controller.
   For details, refer to "8.5 Connecting to the GM1 Controller".

2. From the menu bar, select **Trace**>**Download Trace**.
   A trace will be started. After a cycle task is executed, the values of the variables registered with the trace object are recorded and displayed in the trace object.



3. From the menu bar, select **Trace**>**Stop Trace**.
   The trace will be stopped. To start a trace again, from the menu bar, select **Trace**>**Start Trace**.

Info.

- With respect to the trace display window, you can use "View" and other menus to zoom in or out, expand or shrink the time axis, or adjust variables. From the menu bar, select "Trace" and then select an appropriate menu item for the operation to be performed.
- Traced data can be saved. From the menu bar, select **Trace**>**Save Trace**. With "File type", you can select a file format for the trace file to be saved.

14.4 Recipe Manager Functions

With the Recipe Manager, you can add recipes and alos switch and control the recipe data.

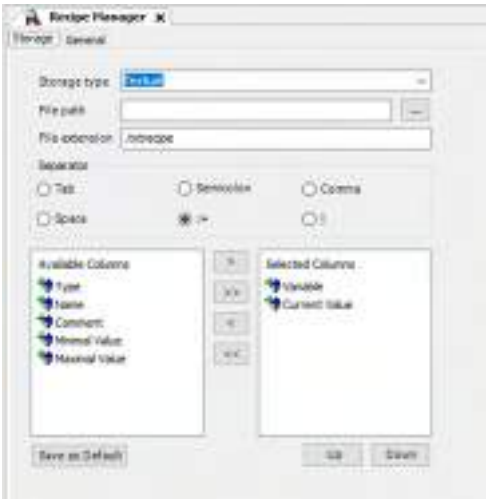
14.4.1 Setting the Recipe Manager

This section explains how to add and set Recipe Manager objects.

Procedure

1. Right-click the “Application” object in the navigation pane and select **Add Object** >**Recipe Manager** from the context-sensitive menu that is displayed.
   The “Add Recipe Manager” dialog box will be displayed.

2. Click the [Add]button.
3. The Recipe Manager setting window will be displayed.
   The default settings can be used as is. You can also make settings if necessary.



Name	Function
Storage type	Select a character string.
File path	Specify the path to store a recipe file.
File extension	Possible to set a desired name with up to 10 characters.
Separator	Specify the delimiter within the recipe file. Example) AAA (variable name): = 1 (value) BBB (variable name): = 12 (value)

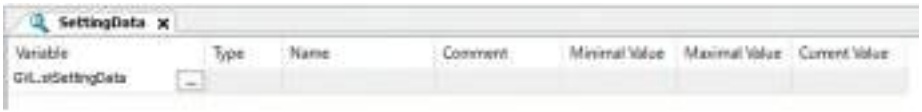
14.4.2 Setting the Recipe Definition

Procedure

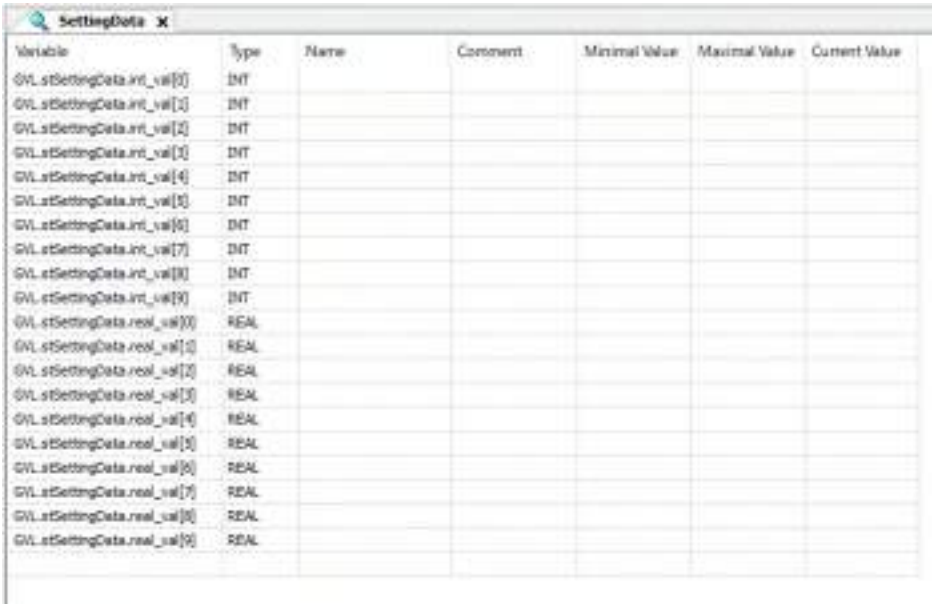
1. Right-click the “Recipe Manager” object and in the navigation pane and select **Add Object**>**Recipe Definition** from the context-sensitive menu that is displayed.
   The “Add Recipe Definition” dialog box will be displayed.

2. Enter a recipe definition name and click the [Add] button.
   Possible to set a desired name with up to 35 characters.

3. Move the cursor to below the variable, enter a variable name you want to add to the recipe definition.



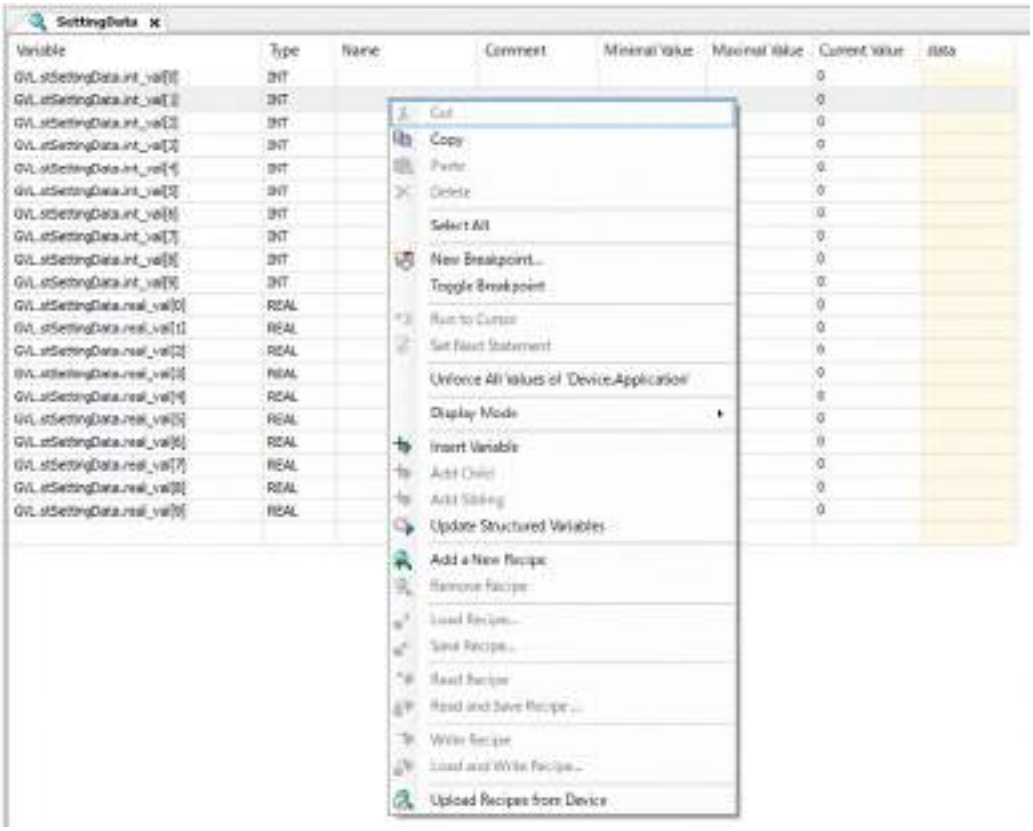
If you enter an array or structure, a list of developed variables is automatically registered.
However, since it takes time for the development if the number of elements is large, implement this only after saving the project in advance.



14.4.3 Recipe Operation Using the GM Programmer

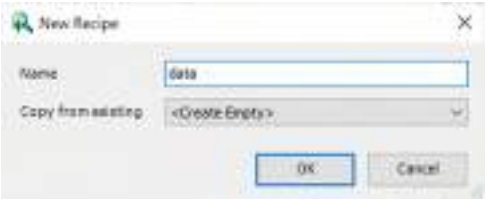
Procedure

1. After setting the recipe definition according to the procedures described in "14.4.2 Setting the Recipe Definition", log into the GM1 Controller and download the settings.
2. In the login state, the current value of the variable registered is displayed in the current value.
3. In the login state, move the cursor to the recipe definition and select “Add New Recipe”. The “New Recipe” dialog box will be displayed.

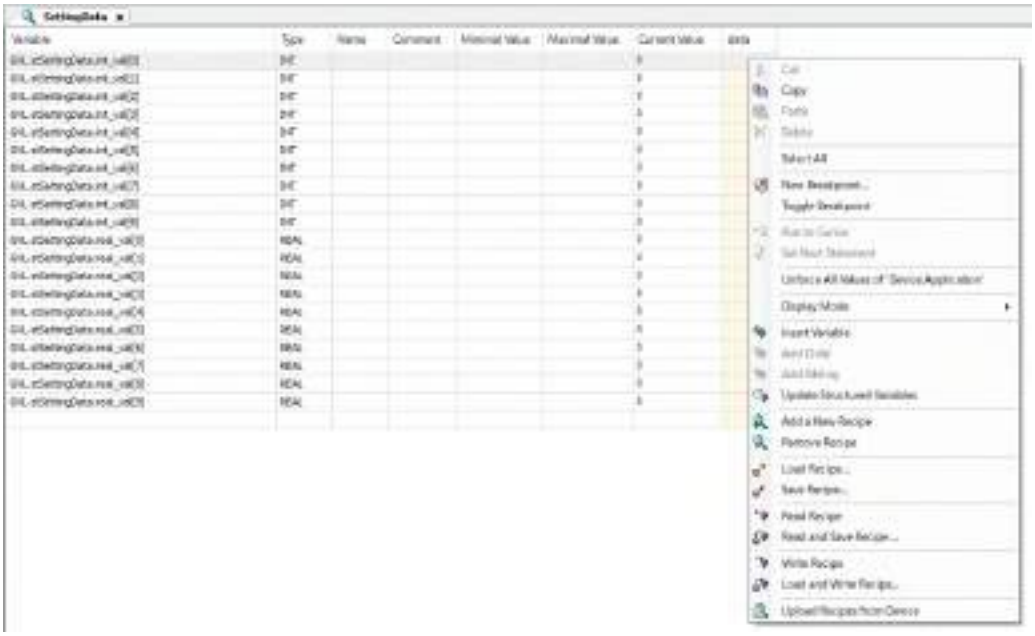


4. Enter a name for the new recipe and select OK to add the recipe as show in the above figure.

Possible to set a desired name with up to 35 characters.



5. Move the cursor to the added recipe and right-click to switch and control the recipe data.



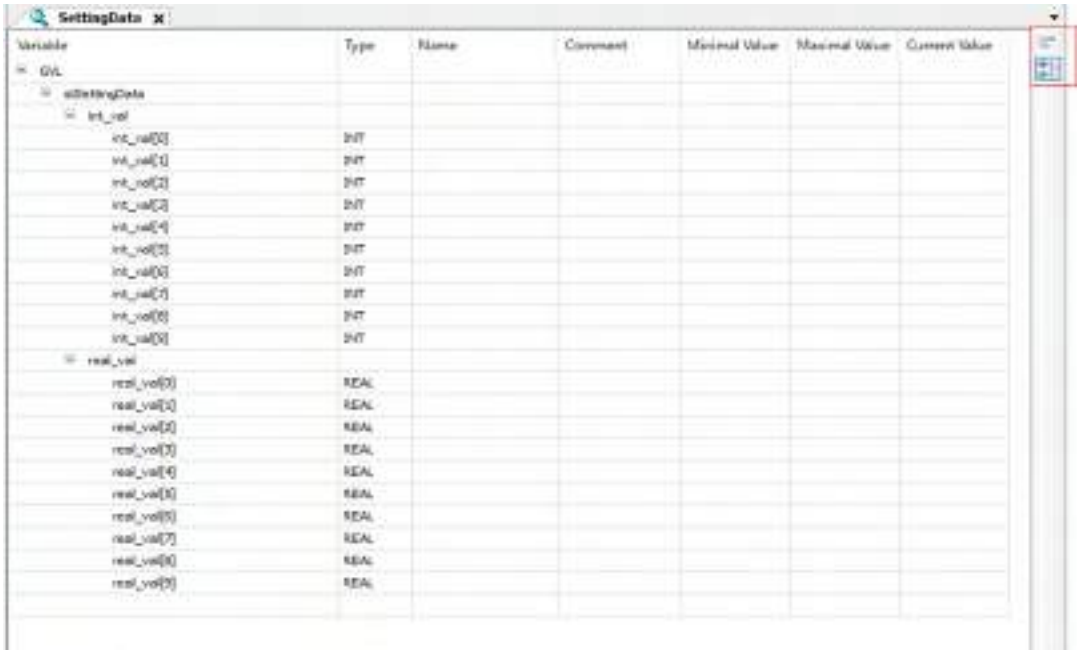
Info.

- The following operations can be performed using the GM Programmer.Items that can be operated are different depending on whether the mode is offline or online. GM Programmer

Item	Offline	Online	Function
Insert Variable	O	O	Inerts a new variable in the recipe definition.
Add Child	O		You can add a child array that can be used in the structured view and that has not been registered in the recipe definition.
Add Sibling	O		You can add a brother array that can be used in the structured view and that has not been registered in the recipe definition.
Update Structured variable	O	O	If there are any items that are not registered in structure, they are reflected on the variable list.
Add a New Recipe	O	O	Adds a new recipe to the recipe definition.
Remove Recipe	O	O	Deletes the selected recipe.
Load Recipe	O	O	Loads the recipe file in the operating PC and write to the selected recipe.
Save Recipe	O	O	Saves the selected recipe in the operating PC as a recipe file.
Read Recipe		O	Reads the current value to the recipe data.
Read and Save Recipe		O	Reads the current value to the recipe (within the tool) and saves the recipe file in the operating PC.
Write Recipe		O	Writes the current value to the recipe data.
Load and Write Recipe		O	Loads data from the recipe file to the recipe (within the tool) and write it in the current value.

Item	Offline	Online	Function
Upload Recipes from device		○	Reads the recipe within the controller device and reflects it on the recipe within the tool.
Display mode (Decimal/ Hexadecimal)		○	-

- Switch to list view/structured view



15 Overview of PANATERM Lite for GM

15.1 System Requirements 15-3
15.1.1 Operating Environment of PANATERM Lite for GM 15-3
15.2 Installation and Uninstallation 15-4
15.2.1 Installing PANATERM Lite for GM 15-4
15.2.2 Uninstalling PANATERM Lite for GM 15-4
15.3 Basic Operations 15-5
15.3.1 How to Start 15-5
15.3.2 How to Exit 15-8
15.4 Component Names 15-9
15.4.1 Menu Bar 15-9
15.4.2 Toolbar 15-10
15.4.3 Navigation Pane 15-11
15.4.4 Main Pane 15-11
15.4.5 Status Field 15-12
15.5 Window Operations 15-13
15.5.1 Moving the Pane Location 15-13
15.5.2 Switching the Tab of the Main Pane 15-14
15.6 Selecting the Device to Connect 15-16
15.6.1 Configuring Servo Amplifier Communication Settings 15-16
15.6.2 Setting up the Servo Amplifier Connected to the GM1 Controller 15-18
15.6.3 Editing Settings without Connecting to the GM1 Controller 15-23
15.7 Parameter Window 15-25
15.7.1 Configuration of Parameters Window 15-25
15.7.2 Setting Parameters 15-27
15.7.3 Copying Parameters 15-29
15.7.4 Switching the Input Format of Parameter Values 15-30
15.7.5 Setting I/O Pin Assignment 15-31
15.8 Object Window 15-33
15.8.1 Configuration of Object Window 15-33
15.8.2 Setting Objects 15-36
15.9 MINAS Parameters for the GM1 Controller 15-38
15.10 Monitor Window 15-39
15.10.1 Configuration of Monitor Window 15-39
15.10.2 Checking the Monitor Window 15-41
15.11 Alarm Window 15-43
15.11.1 Configuration of Alarm Window 15-43

15.11.2 Checking Alarms 15-45
15.12 Other Functions 15-47
15.12.1 Language Setting Function 15-47
15.12.2 Help Function 15-47
15.12.3 Version Display Function 15-47
15.13 Troubleshooting for Servo Amplifiers and Motors 15-49
15.13.1 I Cannot Set up 15-49
15.13.2 I Cannot Communicate 15-49
15.13.3 I Cannot Print 15-49
15.13.4 I Cannot Set up Axes 15-50
15.13.5 PANATERM Lite for GM Does Not Behave Normally 15-50
15.13.6 The Parameter Window Does Not Behave Normally 15-50
15.13.7 The Object Window Does Not Behave Normally 15-51
15.13.8 The Monitor Window Does Not Behave Normally 15-51
15.13.9 The Alarm Window Does Not Behave Normally 15-51

15.1 System Requirements

15.1.1 Operating Environment of PANATERM Lite for GM

Programming software

Product name	Version	Applicable language
PANATERM Lite for GM	Ver.1.1	Japanese / English / Chinese

(Note 1) When GM Programmer is installed, MINAS setup support software "PANATERM Lite for GM" is installed at the same time.

Software operating environment

Item	Description
OS	Microsoft(R) Windows(R) 10: 32bit / 64bit
PC	PC with the following software installed: ● Microsoft.NET Framework 4.6.1 or later ● Microsoft Visual C++ 2010 SP1 Redistributable Package (x86) ● Microsoft Visual C++ 2010 SP1 Redistributable Package (x64) ● Microsoft Visual C++ 2013 Redistributable Package (x86) ● Microsoft Visual C++ 2013 Redistributable Package (x64) ● Microsoft Visual C++ 2015 Update 3 Redistributable Package (x86) ● Microsoft Visual C++ 2015 Update 3 Redistributable Package (x64)
HDD	At least 4 GB of free space
Memory	At least 8 GB
Communication port	LAN port (for Ethernet connection) USB 2.0 port (for USB connection)

15.2 Installation and Uninstallation

15.2.1 Installing PANATERM Lite for GM

When GM Programmer is installed, PANATERM Lite for GM is also installed at the same time.

15.2.2 Uninstalling PANATERM Lite for GM

When GM Programmer is uninstalled, PANATERM Lite for GM is also uninstalled at the same time.

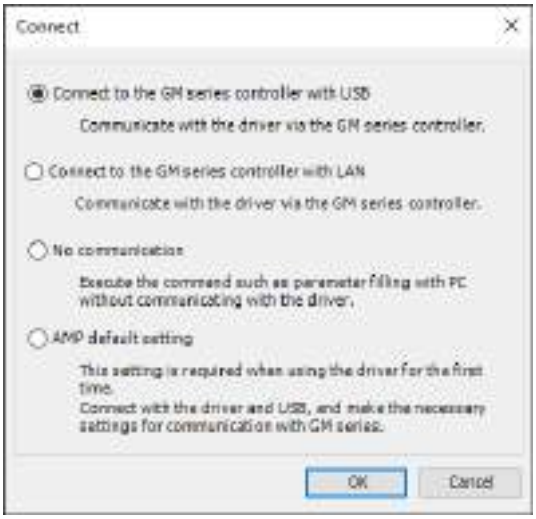
15.3 Basic Operations

This section explains how to start and exit PANATERM Lite for GM.

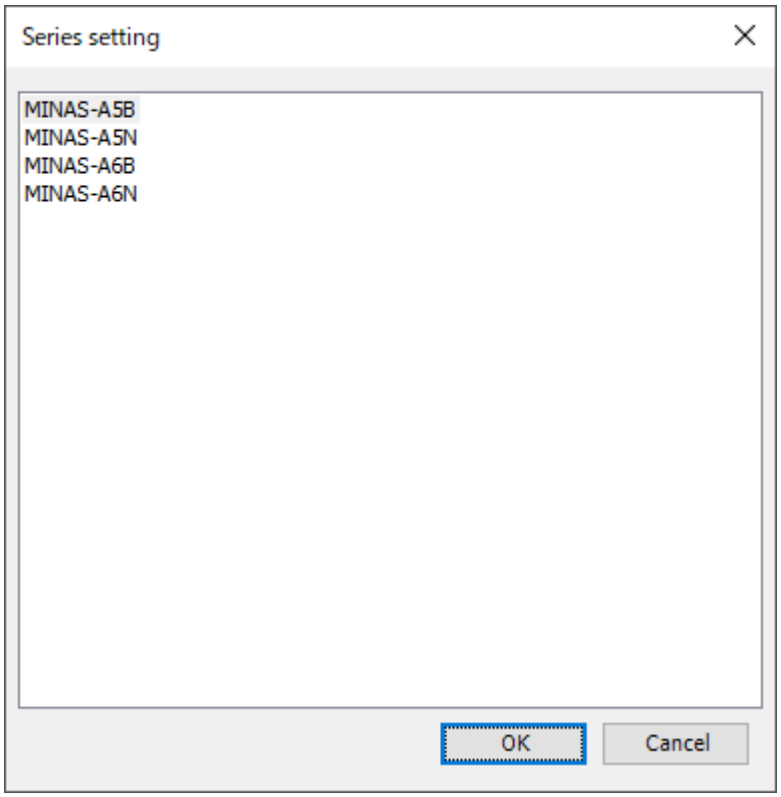
15.3.1 How to Start

1 2 Procedure

1. Click the [Start] button in the Windows task bar and select **Panasonic Corporation**>**PANATERM Lite for GM**.
2. The "Connect" dialog box will be displayed.
   Select a communication setting option and click [OK].

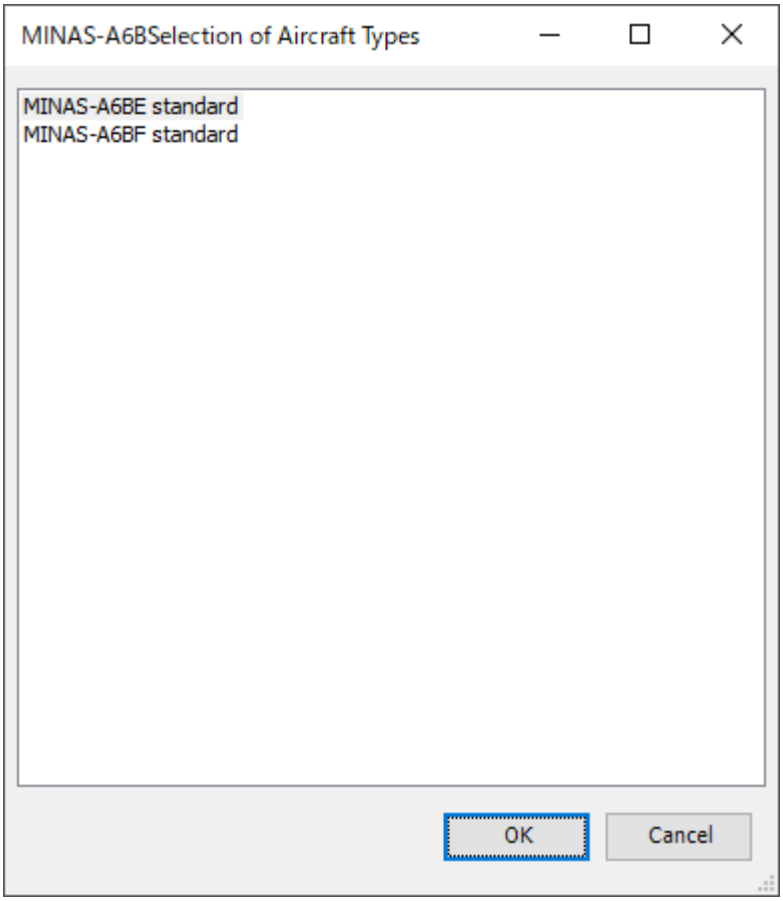


3. The "Series Setting" dialog box will be displayed.

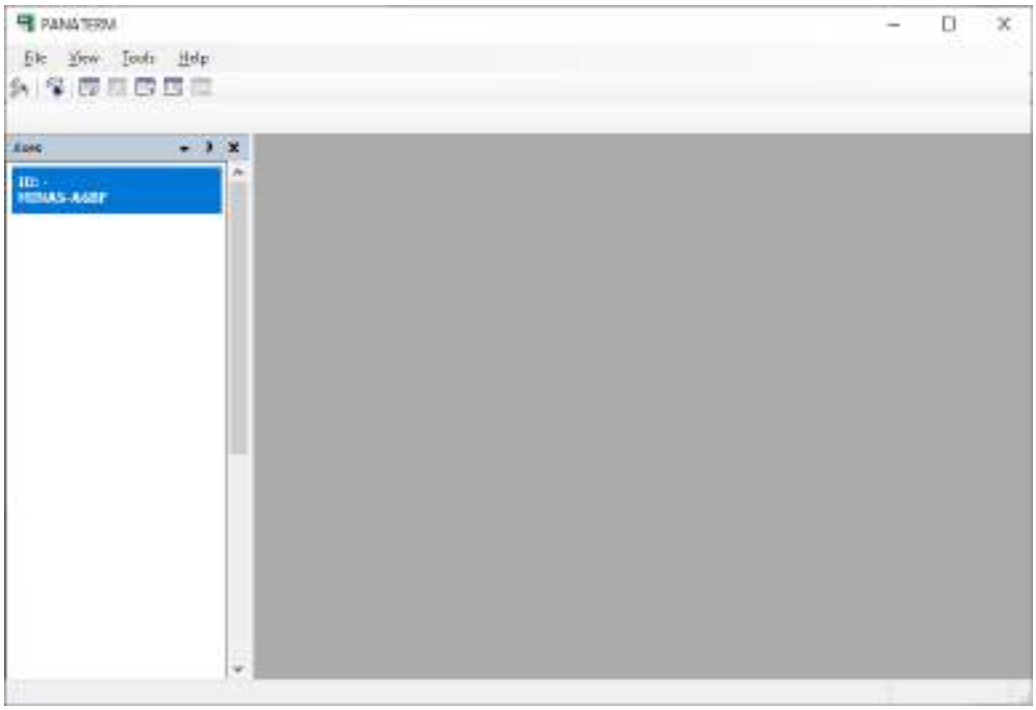

Series setting
MINAS-A5B
MINAS-A5N
MINAS-A6B
MINAS-A6N
OK
Cancel

Note

When the "Selection of Aircraft Types" dialog box is displayed, select a model and click the "OK" button.



4. PANATERM Lite for GM will be started.



15.3.2 How to Exit

- Note that all information will be lost if you close the program without saving settings, collected data, or other information.

Procedure

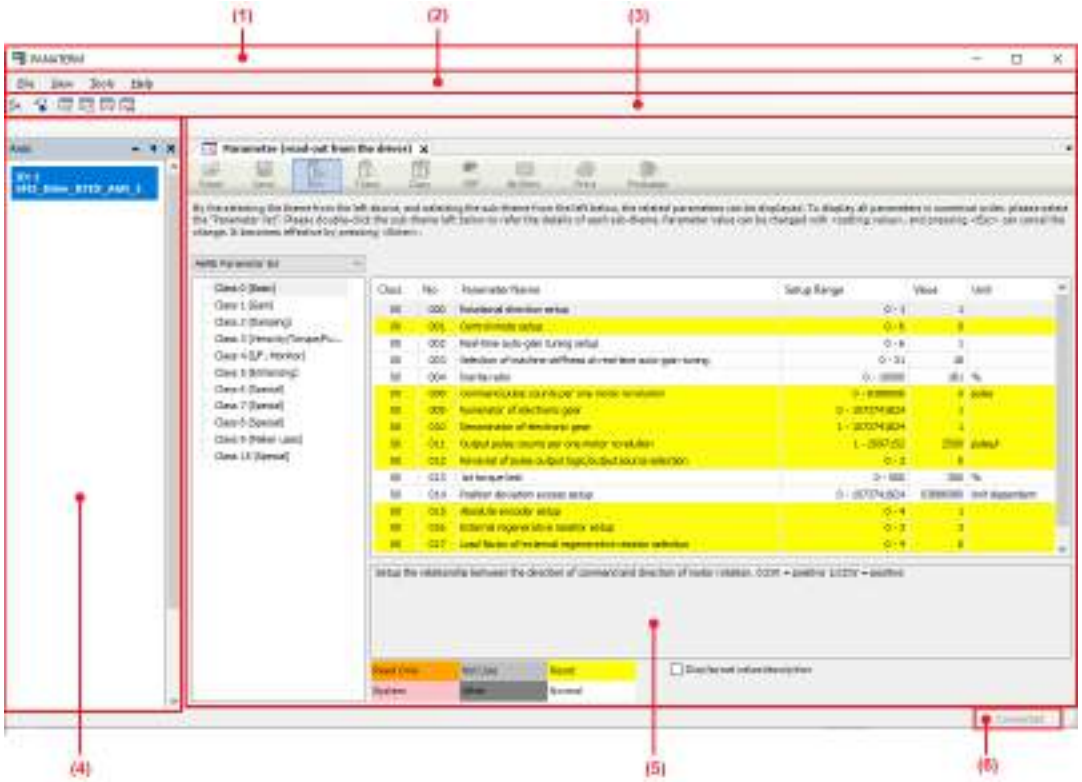
1. From the menu bar, select **File**>**Exit**.
   PANATERM Lite for GM will be closed.

Info.

- You can also close PANATERM Lite for GM by clicking the [×] button on the title bar.

15.4 Component Names

This section explains the components and displays of PANATERM Lite for GM.



No.	Name	Description
(1)	Title bar	The title bar displays the project file name, [minimize] button, [maximize] button, and [close] button.
(2)	Menu bar	The menu bar displays the menu commands for each purpose in list format.
(3)	Toolbar	The toolbar displays each command as an icon.
(4)	Navigator pane	This pane displays a list of axes.
(5)	Main pane	This pane displays the Parameter window, Monitor window, Alarm window, and other windows. The window can be switched by selecting a desired tab.
(6)	Status field	This field displays the status of connection to the GM1 controller.

15.4.1 Menu Bar

The menu bar displays the following menus:

File

Item	Function
Settings	Used to set up an amplifier with the amplifier connected or with a model selected. Select Model: Select an amplifier to be connected. Connect Amplifier: Select either the connection via PC or the direct connection for connecting the GM1 controller and the amplifier order to set up an amplifier.
Exit	Closes PANATERM Lite for GM

View

Item	Function
Axes	Displays a list of axes
Parameter	Displays the Parameter window
Monitor	Displays the Monitor window
Alarm	Displays the Alarm window

Tools

Item	Function
English	Switches the display language of GM Programmer to English
日本語	Switches the display language of GM Programmer to Japanese
中文(簡体)旧版	Switches the display language of GM Programmer to Chinese

Help

Item	Function
About	Displays version information
PANATERM Lite for GM Help	Displays the manual

15.4.2 Toolbar

The toolbar displays the following icons:

Name	Icon	Function
Select the drive series		Sets up an amplifier with the amplifier unconnected
Opens the connrct dialog		Select either the connection via PC or the direct connection for connecting the GM1 controller and the amplifier order to set up an amplifier.
Opens the Axis view		Displays a list of axes
Opens the Alarm view		Displays the Alarm view

Name	Icon	Function
Opens the Parameter view		Displays the Parameter view
Opens the Monitor view		Displays the Monitor view

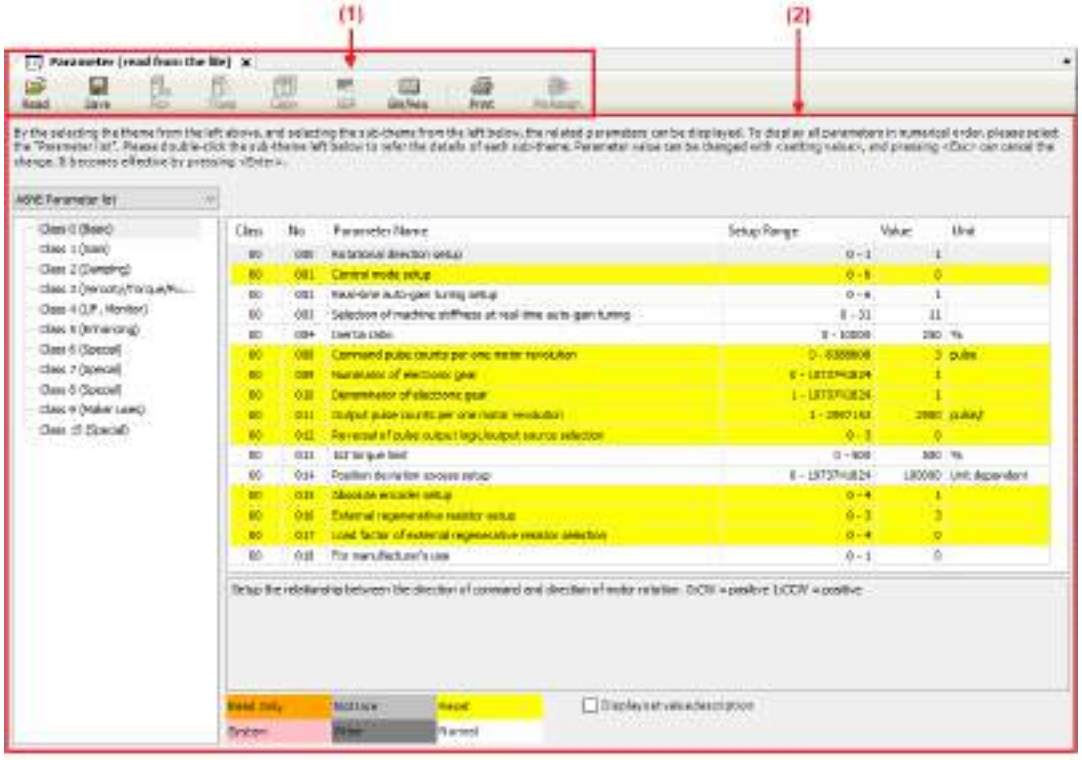
15.4.3 Navigation Pane

The navigation pane displays the following tree:

No.	Name	Icon	Function
(1)	Auto Hide		Always shows the navigation pane.
			Minimizes and hides the navigation pane.
	Close		Closes the navigation pane.
(2)	Axes		Displays a list of axes downloaded to the GM1 controller

15.4.4 Main Pane

The main pane displays the following sub-panes:



No.	Name	
(1)	Toolbar	In each window, the toolbar displays commands as icons.
(2)	Main view	The main view displays the Parameter window, Alarm window, Monitor window, and other windows.

15.4.5 Status Field

The status field displays the current communication status.

Display	Description
	Indicates that there is no communication with the GM1 controller.
Connected	Indicates that there are communications with the GM1 controller.

15.5 Window Operations

This section explains common window operations for PANATERM Lite for GM.

15.5.1 Moving the Pane Location

You can freely change the layout of each pane of PANATERM Lite for GM.

Example: Moving the navigator pane from the left edge to the right edge of the window

1 2 Procedure

1. Click the title bar of the navigator pane and then drag it to the main pane.
   The navigator pane will stay in a floating state and arrows indicating movable directions will be displayed.

2. Drag the navigator pane in the direction in which you want to move it.
   The relocation destination will be displayed in light blue.

3. Release the left mouse button.

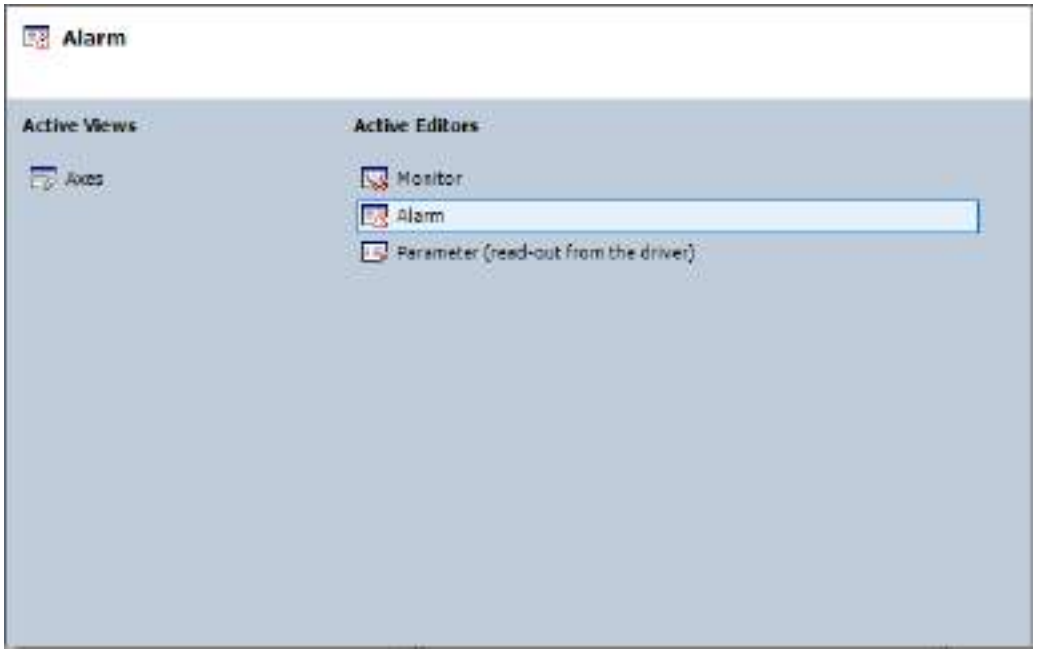
The navigator pane will be docked into the existing pane and the relocation will be completed.

15.5.2 Switching the Tab of the Main Pane

You can switch the tab of the main pane.

Procedure

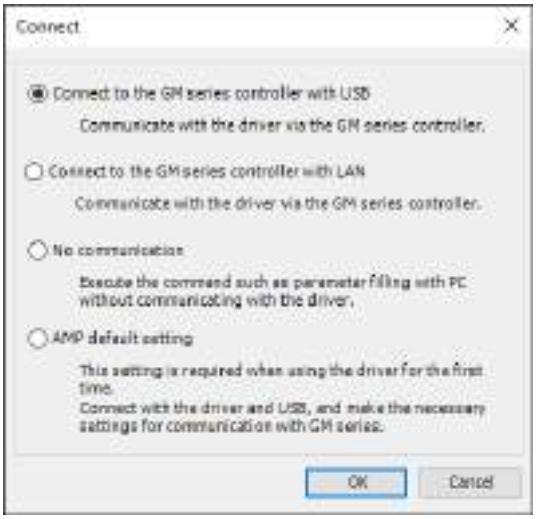
1. Press the <Ctrl> key + <Tab> key simultaneously.
   The window for switching the tab of the main pane will be displayed.



2. While holding down the <Ctrl> key, press the <Tab> key until the desired tab is selected.
3. Release the <Ctrl> key.
   The window corresponding to the selected tab will be displayed.

15.6 Selecting the Device to Connect

After you start PANATERM Lite for GM or when you select **File**>**Settings**>**Connect Amplifier**, a dialog box for selecting the device to be connected will be displayed.



The following options will be displayed.

- Connect to the GM series controller with USB
- Connect to the GM series controller with LAN
- No communication
- AMP default setting

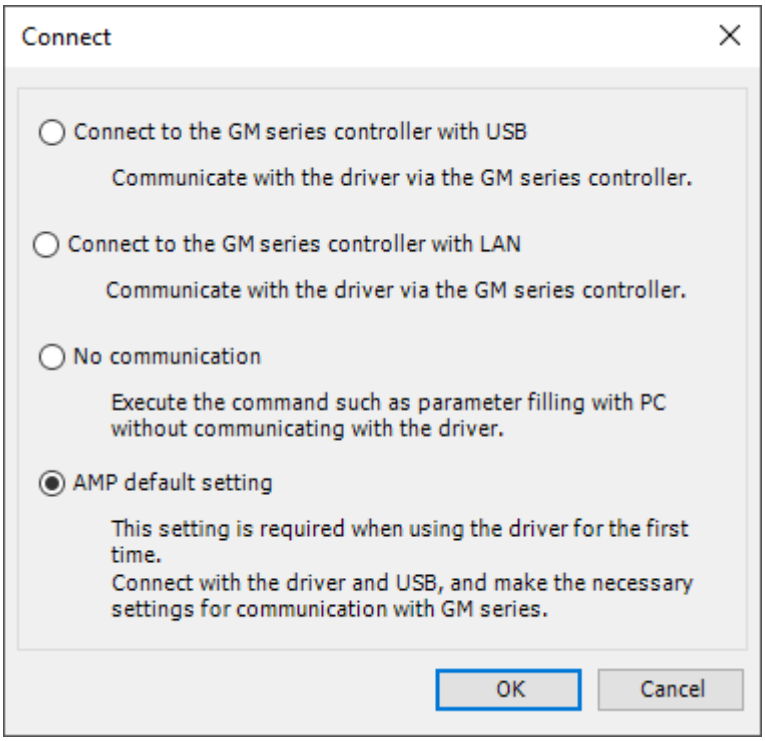
The following section describes operations when each option is selected.

> When you use the servo amplifier for the first time after it was purchased, you must establish a communication between the GM1 Controller and the servo amplifier. Connect the PC and the servo amplifier with a USB cable and execute "Configure amplifier communication settings".
>
> Then, with the servo amplifier and GM1 Controller connected, set up the servo amplifier.

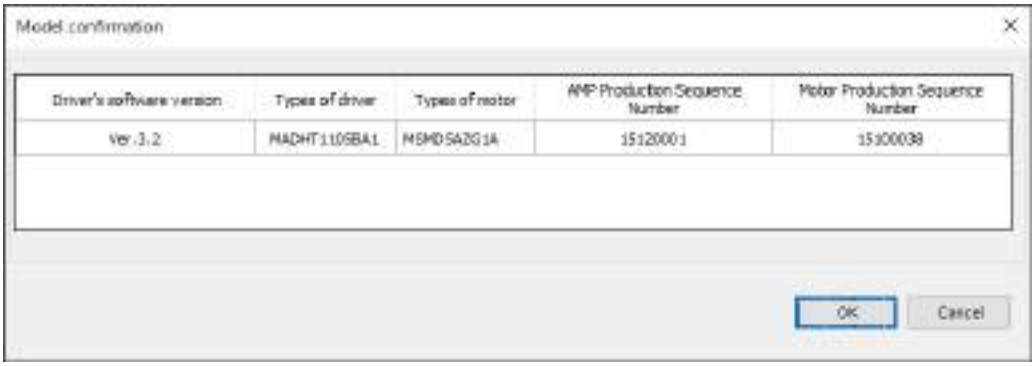
15.6.1 Configuring Servo Amplifier Communication Settings

Procedure

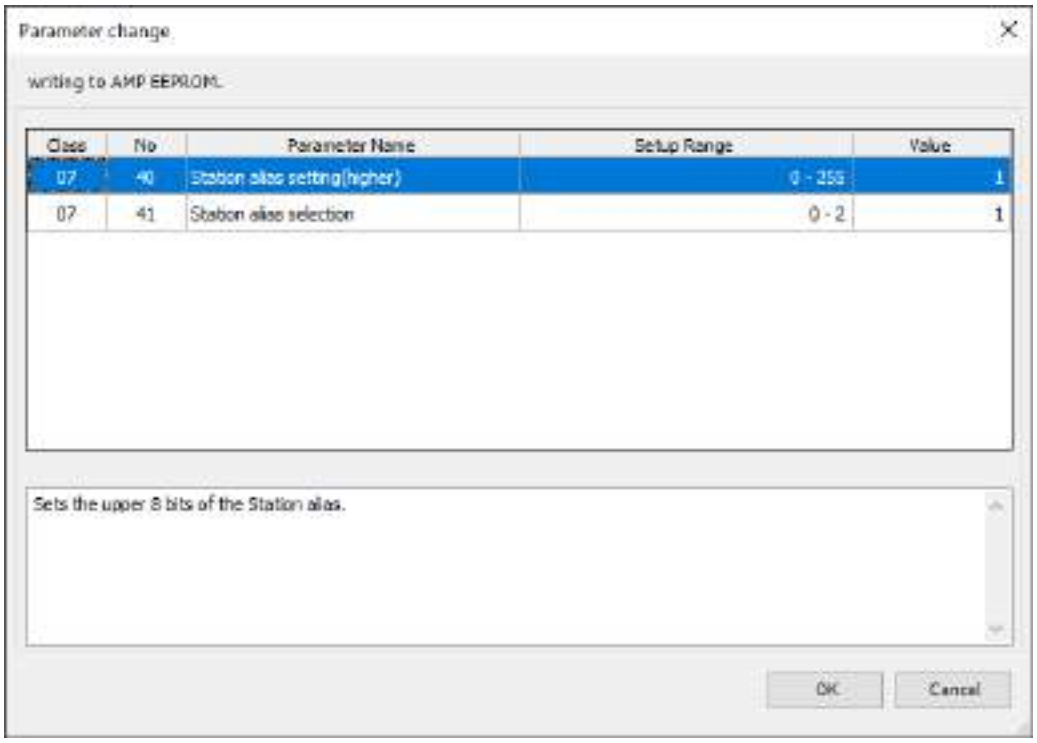
1. Start PANATERM Lite for GM.
   The "Connect" dialog box will be displayed.



2. Select "AMP default setting" and click [OK].
The "Model confirmation" dialog box will be displayed.



3. Check the software version of the servo amplifier for which settings are to be changed and then click [OK].
The "Parameter change" dialog box will be displayed.



When Pr7.41 is set to 0, set value of the rotary switch on the front panel of the servo amplifier and the set value of Pr7.40 are set as “Station alias”.

Station alias	
Higher 8 bits	**Lower 8 bits**
Set value of 3740h	Set value of the rotary switch

When Pr7.41 is set to 1, the value of the SII area (0004h) is regarded as Station alias.

The setting of the SII area (0004h) can be made using EtherCAT slave device object of the GM Programmer.

4. Click the [OK] button.

The "Setting Complete" dialog box will be displayed.

5. Click the [OK] button.

The main pane will be displayed. Start the servo amplifier.

15.6.2 Setting up the Servo Amplifier Connected to the GM1 Controller

This section explains how to set up MINAS series servo amplifiers connected to the GM1 Controller.

For other devices, refer to their instruction manuals when setting up them.

The PC communicates with the servo amplifier connected to the GM1 Controller.

Connect the PC and GM1 Controller with an Ethernet cable or USB cable. With the GM1 Controller and servo amplifier connected with a Cat5e shielded cable, set up the servo amplifier.

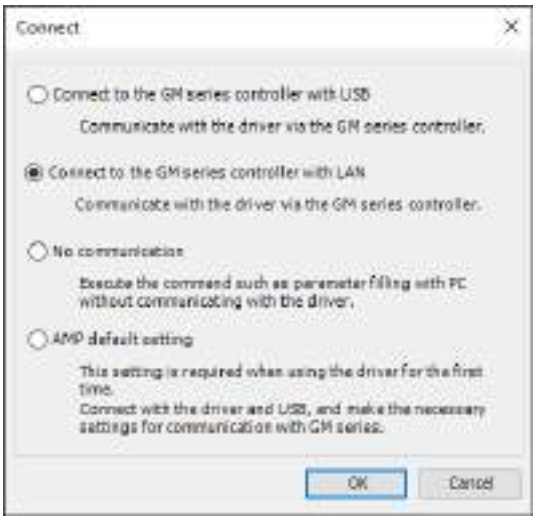
- Perform this setup work only after the connection between the GM1 Controller and the servo amplifier has been established.

If Connected with Ethernet Cables

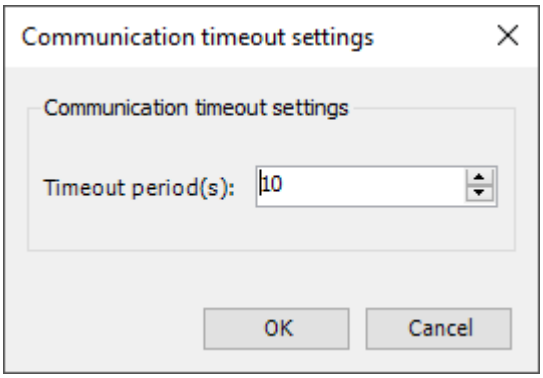
If connected with Ethernet cables, use the following procedure.

Procedure

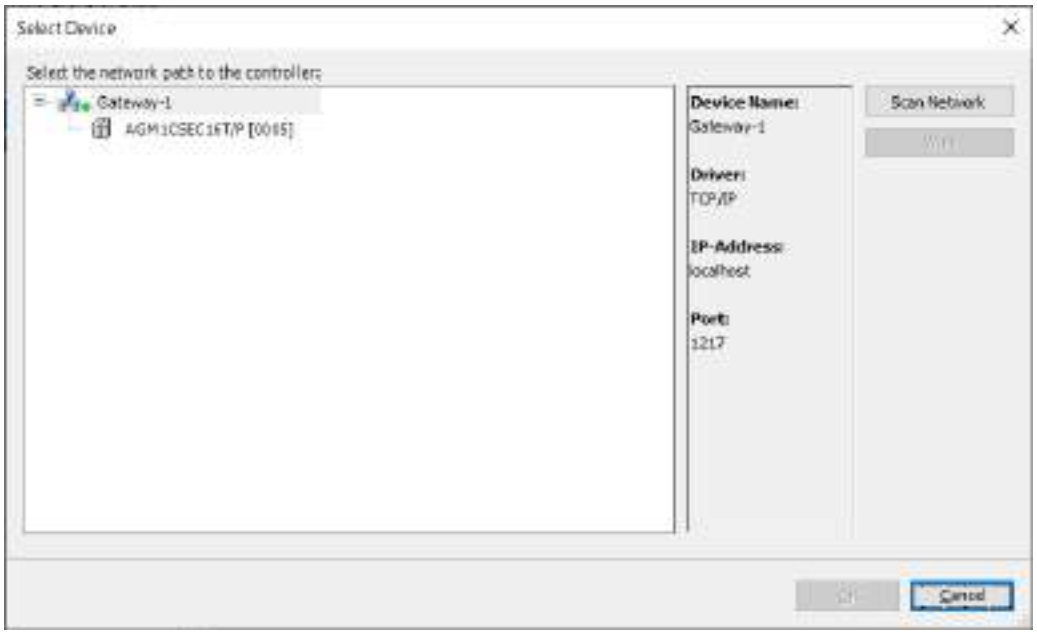
1. Start PANATERM Lite for GM.
   The "Connect" dialog box will be displayed.



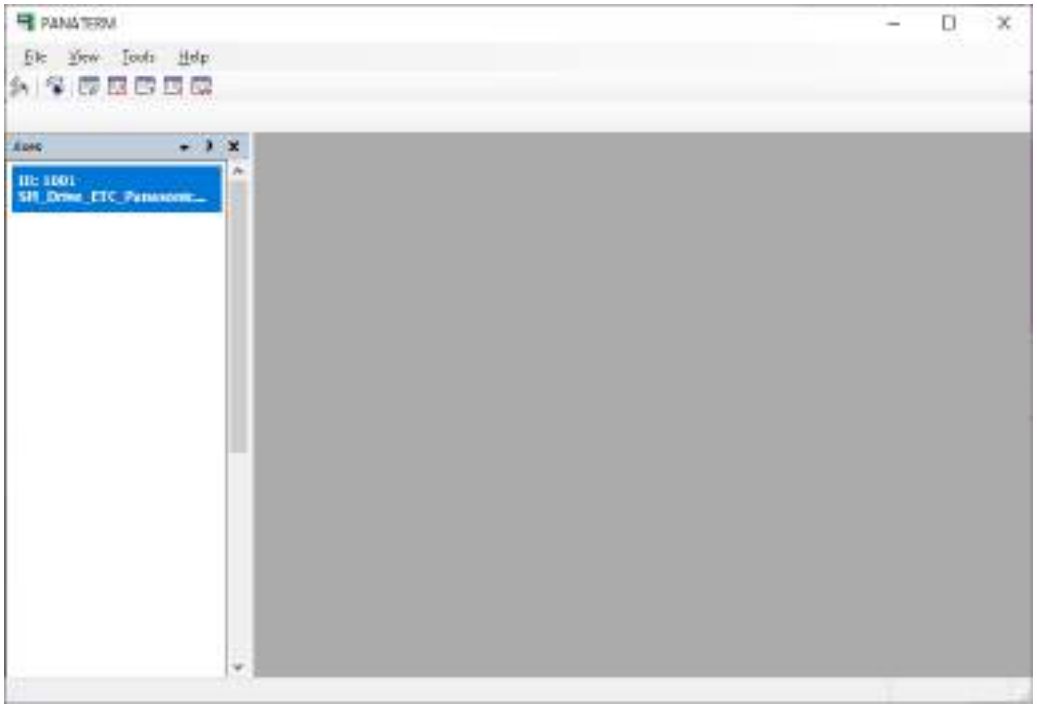
2. Select "Connect to the GM series controller with LAN" and click the [OK] button.
   The "Communication timeout settings" dialog box will be displayed.



3. Change the timeout period and click the [OK] button.
The "Select Device" dialog box will be displayed.



4. Click the [Search Network] button, select the GM1 Controller, and click the [OK] button.
The main pane will be displayed.

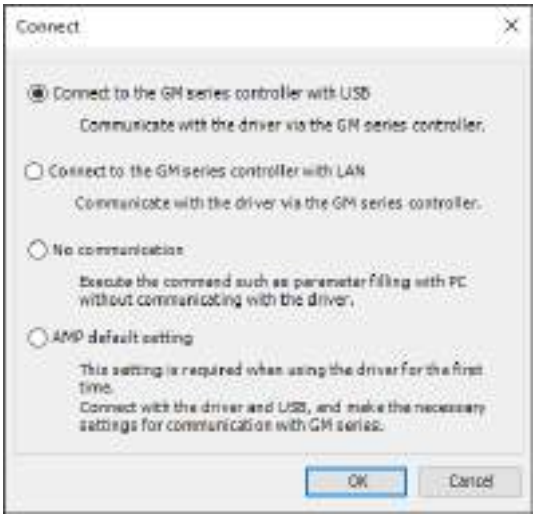


If Connected with USB Cables

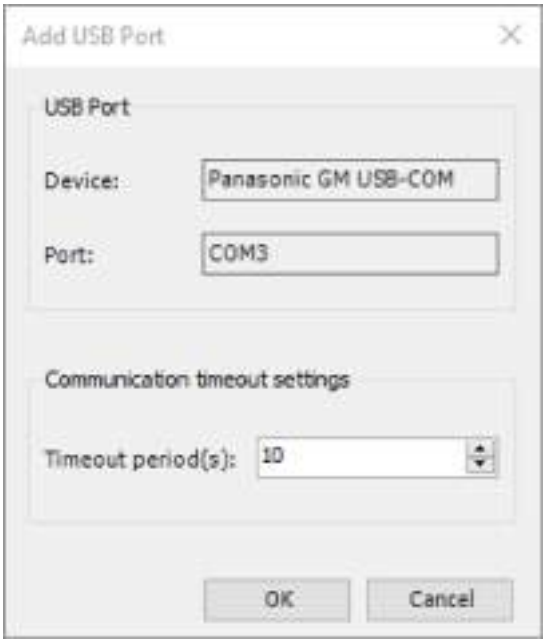
If connected with USB cables, use the following procedure.

1 2 Procedure

1. Start PANATERM Lite for GM.
   The "Connect" dialog box will be displayed.

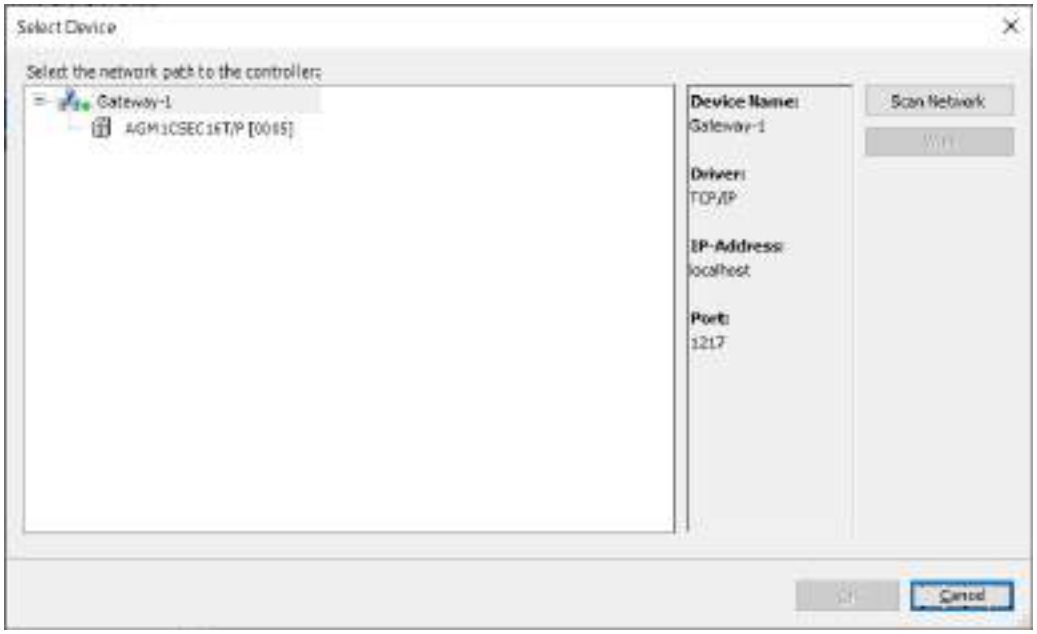


2. Select "Connect to the GM series controller with USB" and click the [OK] button.
   The "Add USB Port" dialog box will be displayed.

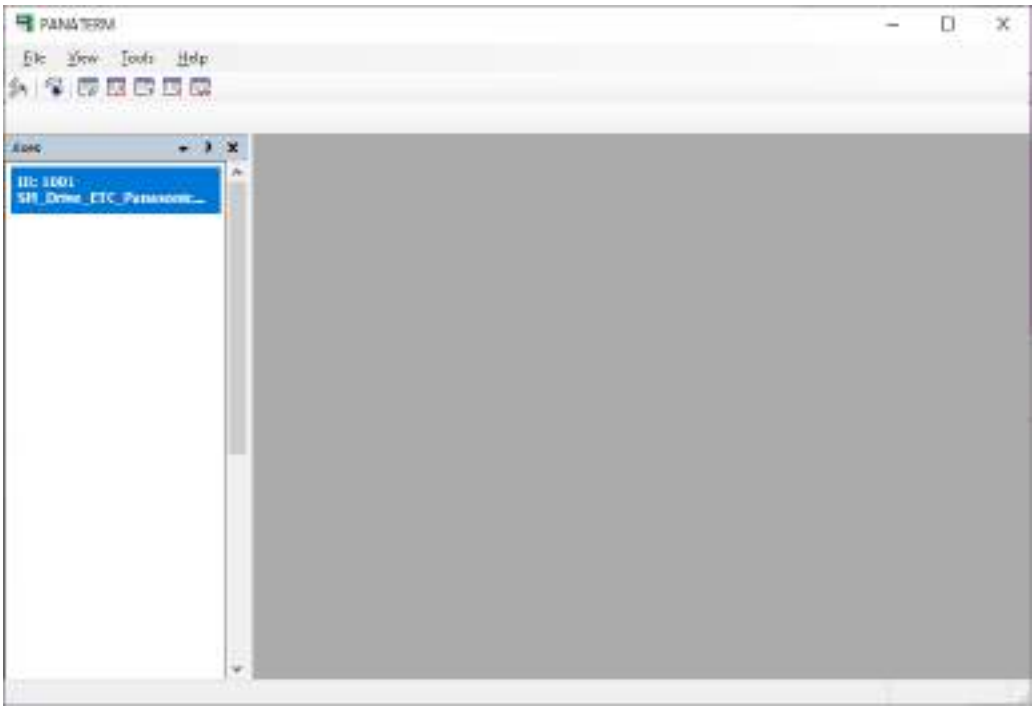


3. Change the timeout period and click the [OK] button.
   A message window will be displayed, asking whether to add a USB port and restart the Gateway.

4. Click the [OK] button.
   The "Select Device" dialog box will be displayed.



5. Click the [Search Network] button, select the GM1 Controller, and click the [OK] button.
   The main pane will be displayed.

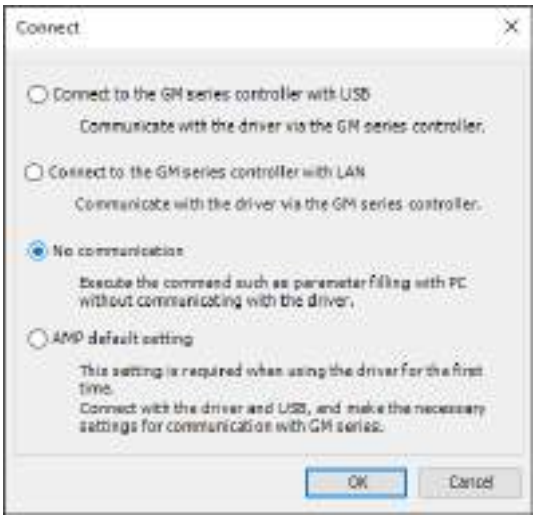


15.6.3 Editing Settings without Connecting to the GM1 Controller

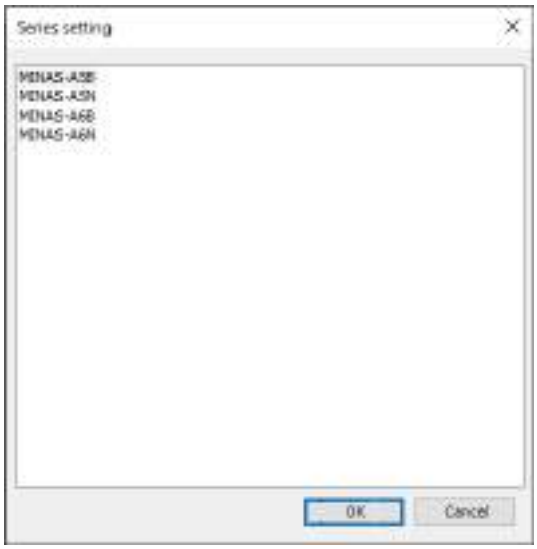
You can freely edit parameters and other data saved in files without connecting to the GM1 controller. The edited contents are not written to the servo amplifier.

1 2 Procedure

1. Start PANATERM Lite for GM.
   The "Connect" dialog box will be displayed.



2. Select "No communication" and click the [OK] button.
   The "Select Series " dialog box will be displayed.



3. Select a servo amplifier to be connected and click the [OK] button.
   The main pane will be displayed.

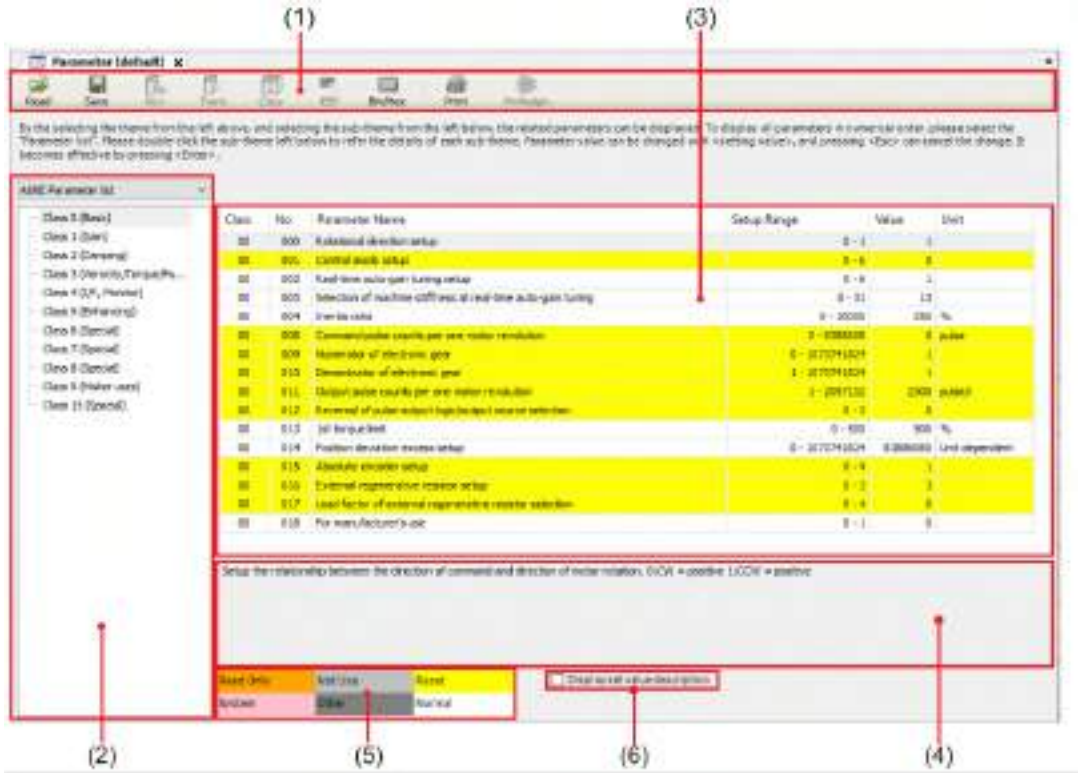
Note

When the "Select Model" dialog box is displayed, select a model and click the "OK" button.

15.7 Parameter Window

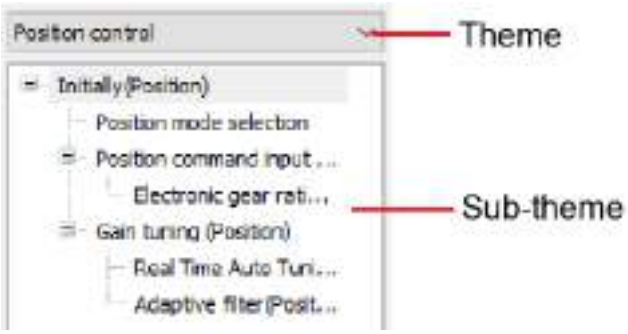
The Parameter window allows the user to check and rewrite the values of servo amplifier parameters, save them to parameter files, and perform other parameter-related operations.

15.7.1 Configuration of Parameters Window



No.	Name	Function
(1)	Toolbar	The toolbar consists of basic operation commands related to parameters, such as save and read.

Icon	Name	Function
Read	Read	Reads parameters from file ".prm5". When this button is enabled, you can specify a parameter file also by drag-and-drop operation.
Save	Save	Writes parameters to file ".prm5".
Rcv	Rcv	Receives parameters from the servo amplifier.

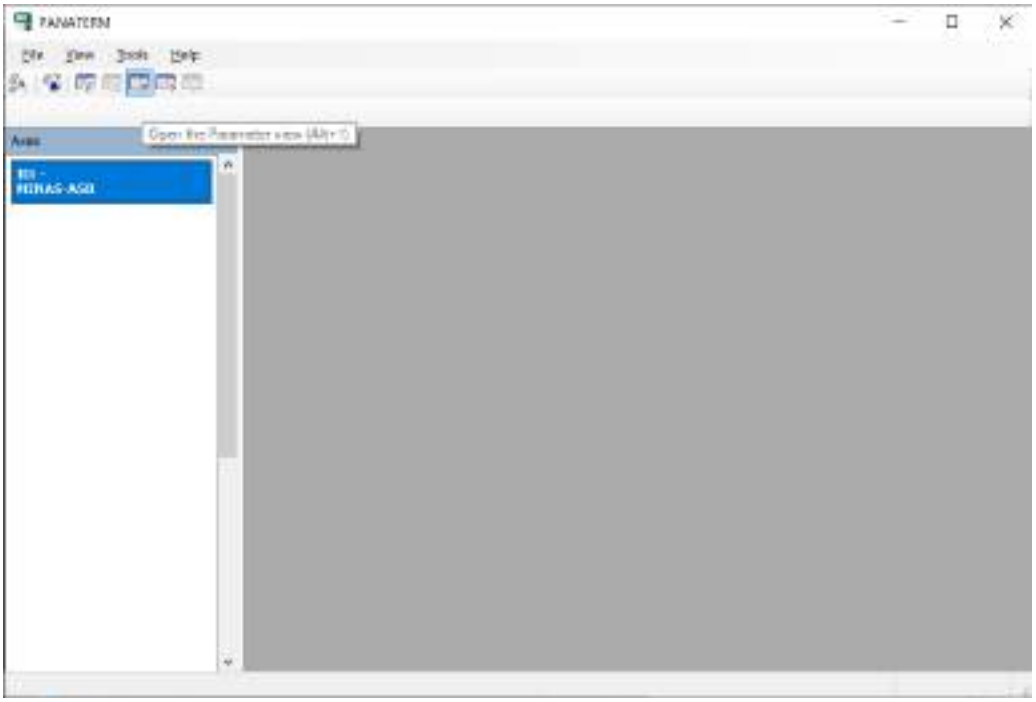
<table>
<tr><th>No.</th><th>Name</th><th>Function</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Icon</th><th>Name</th><th>Function</th></tr>
<tr><td>  </td><td>Trans</td><td>Transmits parameters to the servo amplifier.</td></tr>
<tr><td>  </td><td>Copy</td><td>Copies the parameters of a servo amplifier to servo amplifiers for other axes.</td></tr>
<tr><td>  </td><td>EEP</td><td>Writes parameters to EEPROM of the servo amplifier.</td></tr>
<tr><td>  </td><td>Bin / Hex</td><td>Inputs the selected settings in binary or hexadecimal format.</td></tr>
<tr><td>  </td><td>Print</td><td>Prints parameters.</td></tr>
<tr><td>  </td><td>Pin assignment setting</td><td>Sets I/O pin assignment.</td></tr>
</table>
</td></tr>
<tr><td>(2)</td><td>Theme selection pane</td><td>After a theme is selected, if a parameter category is selected from a sub-theme, related parameters will be displayed in the parameter setting area.
 
For details on each parameter, refer to the instruction manual and other technical references for the servo amplifier.</td></tr>
<tr><td>(3)</td><td>Parameter setting area</td><td>Allows the user to set or edit parameters.
<table>
<tr><th>Name</th><th>Function</th></tr>
<tr><td>Class</td><td>Displays parameter categories</td></tr>
<tr><td>No.</td><td>Displays parameter numbers</td></tr>
<tr><td>Parameter Name</td><td>Displays parameter names</td></tr>
<tr><td>Setup Range</td><td>Displays the maximum and minimum allowable values of parameter settings</td></tr>
<tr><td>Value</td><td>Displays parameter values. Values can be changed. For parameters provided with a ▼ button beside the set value, a desired value can be selected from the combo box. After selecting a value from the combo box, press the <Enter> key. For parameters without a ▼ button beside the set value, either directly enter a value using <numerical> keys or click "▲""▼" to edit the value by increasing or decreasing it. To set a</td></tr>
</table>
</td></tr>
</table>

No.	Name	Function	
		Name	**Function**
			value, press the <Enter> key. To return a value to its original value, press the <Esc> key.
		Unit	Displays the unit of parameter settings.
(4)	Text display area	Displays a description related to the selected parameter.	
(5)	Parameter attribute description area	Displays a description of parameter attributes. The background color of each parameter in the parameter setting area represents an attribute.	
(6)	"Display-set value description" check box	Selecting the check box displays combo boxes and decimal points in the "Value" column of the parameter setting area. To display parameter set values in an easy-to-understand manner, select the check box.	

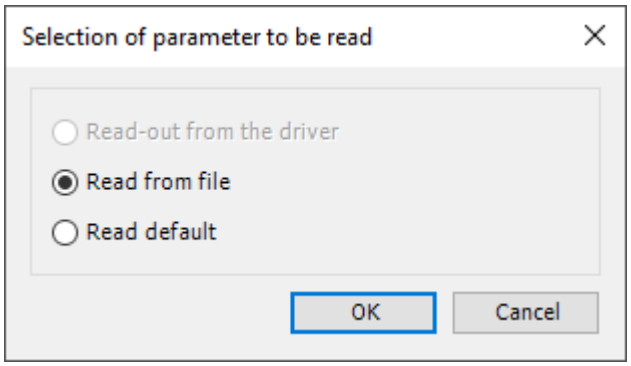
15.7.2 Setting Parameters

Procedure

1. From the menu bar on the main pane, select **View**>**Parameter**. Alternatively, on the toolbar, click the "Open the Parameter view" icon.



The "Selection of parameter to be read" dialog box will be displayed.



☐Read-out from the driver

Communicates with the connected servo amplifier and reads the parameter settings from the servo amplifier. If this mode is selected, parameter values will be reflected in the servo amplifier as soon as they are changed.

☐Read from file

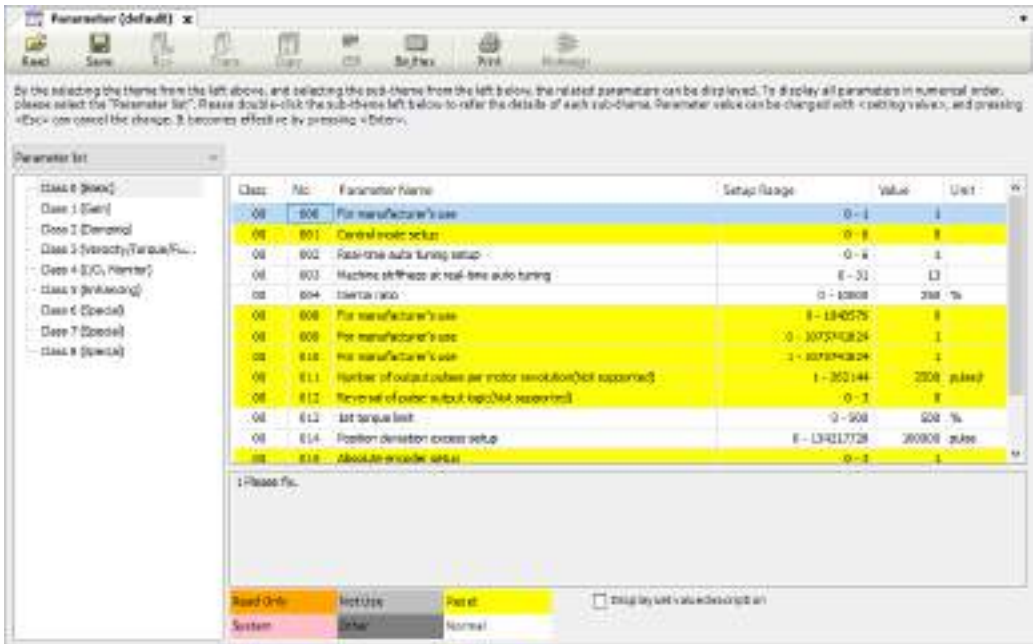
Reads the parameter file (".prm5") that was edited previously. If communication is performed with the servo amplifier, parameter values will be reflected in the servo amplifier as soon as they are changed.

☐Read default

Reads the factory default settings of the servo amplifier that were saved during installation. If communication is performed with the servo amplifier, parameter values will be reflected in the servo amplifier as soon as they are changed.

2. Select one of the three options above and click the [OK] button.

The Parameter window will be displayed.



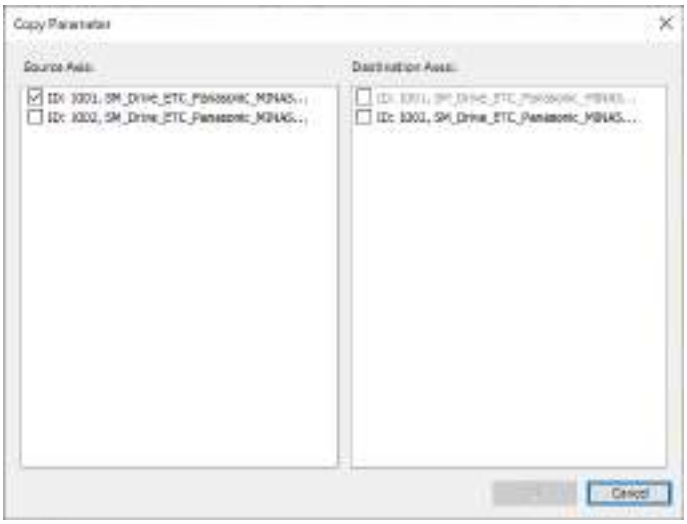
3. After changing the parameter settings, click the [EEP] button to write the parameter settings to the EEPROM of the servo amplifier.
4. Click the [×] button on the Parameter window to close the Parameter window.

15.7.3 Copying Parameters

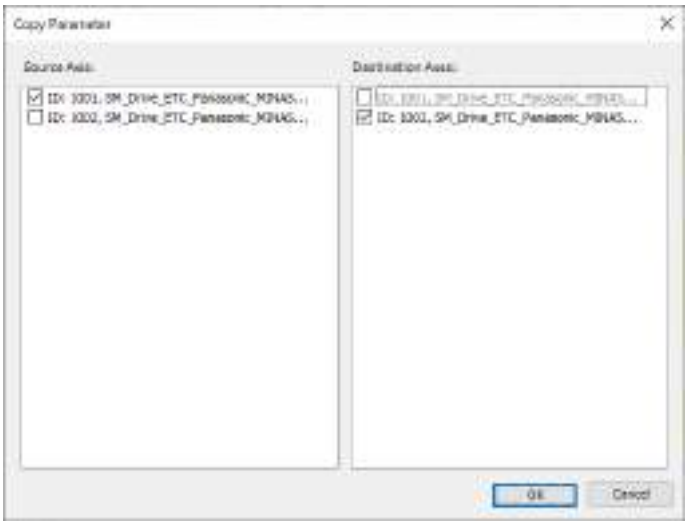
The parameters of a servo amplifier can be copied to servo amplifiers for other axes.
It is not possible to copy from A6B type to A5B type or from A5B type to A6B type.

Procedure

1. Click the "Copy" icon on the toolbar.
   The Copy Parameter window will be displayed.



2. In the "Copy source" area, select an axis from which parameters are to be copied.
3. In the "Copy destination" area, select an axis to which parameters are to be copied.



4. Click the [OK] button.
5. The Parameter window for the copy destination axis will be displayed.
6. Click the "Trans" icon on the toolbar.
   The parameters will be written to the servo amplifier.
7. Click the [EEP] icon on the toolbar.
   The parameters will be written to the EEPROM of the servo amplifier.

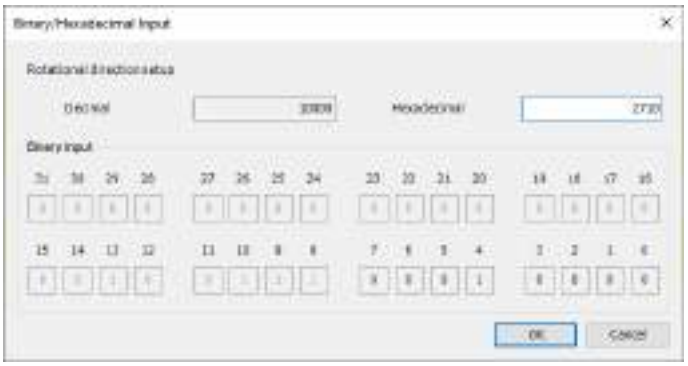
15.7.4 Switching the Input Format of Parameter Values

Selected parameter values can be entered in binary or hexadecimal format.

Procedure

1. Click the "Bin / Hex" icon on the toolbar.
   The Binary / Hexadecimal Input window will be displayed.

2. To enter parameter values in hexadecimal format, enter a value and then press the <Enter> key. To enter parameter values in binary format, click the button corresponding to each bit to switch between "0" and "1".



Note: If the entered value is outside the setting range of the parameter, the allowable range will be displayed below the decimal display section.

3. After the above input operation is complete, click the [OK] button.

15.7.5 Setting I/O Pin Assignment

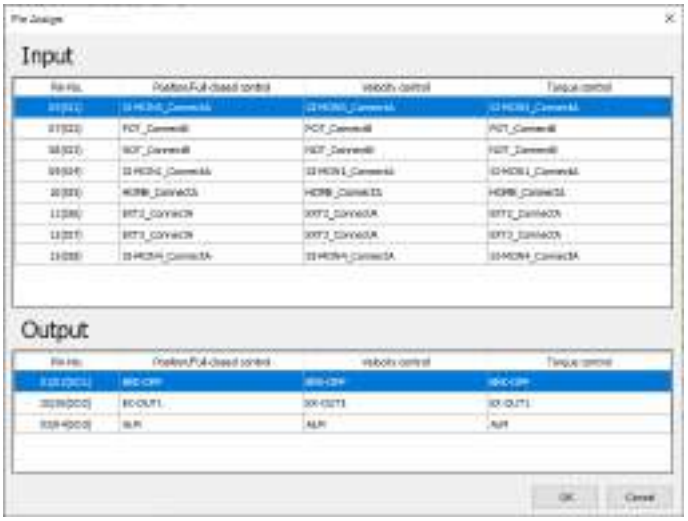
I/O pin assignment can be set.

The assignment can be set for the input and output related parameter "Class 4 (I/F, monitor)".

1 2 Procedure

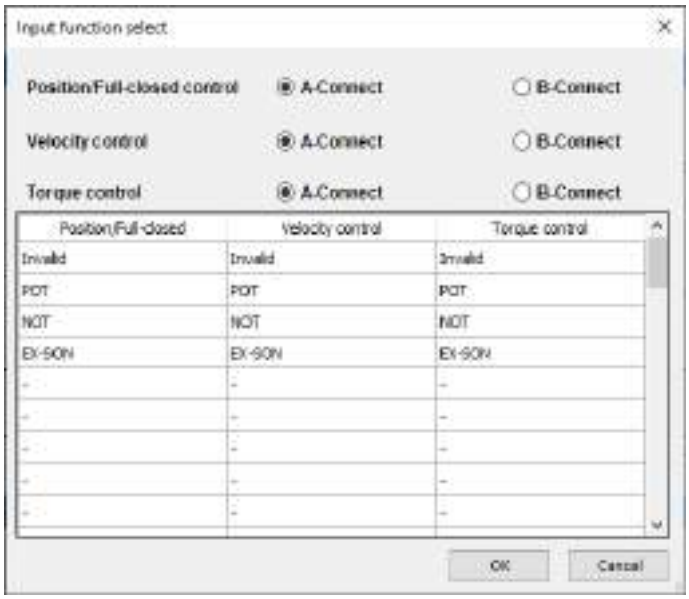
1. Click the "Pin assignment " icon on the toolbar.

The pin assignment setting input window will be displayed.



2. Double-click the row of the pin number to be set.

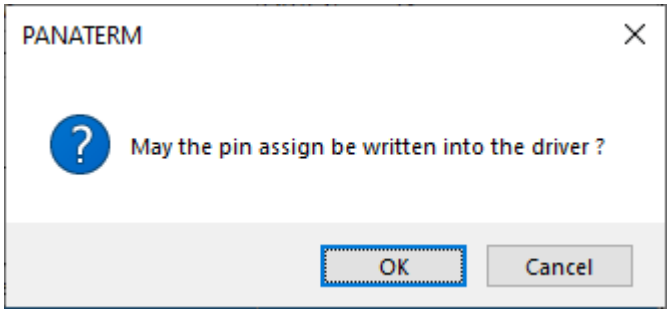
The function selection window will be displayed.



3. Select a function to be assigned to the pin for each control mode and a contact method (only when an input function is selected).

4. In the function selection window, click the [OK] button.
The display will be returned to the pin assignment setting input window.

5. Click the [OK] button.
A confirmation window will be displayed, asking whether to write to the servo amplifier.



Click the [OK] button to write the parameter settings to the EEPROM of the servo amplifier.
Click the [Cancel] button to close the window without writing the parameter settings to the EEPROM of the servo amplifier.

Note: Pin assignment settings do not take effect until the servo amplifier is restarted.

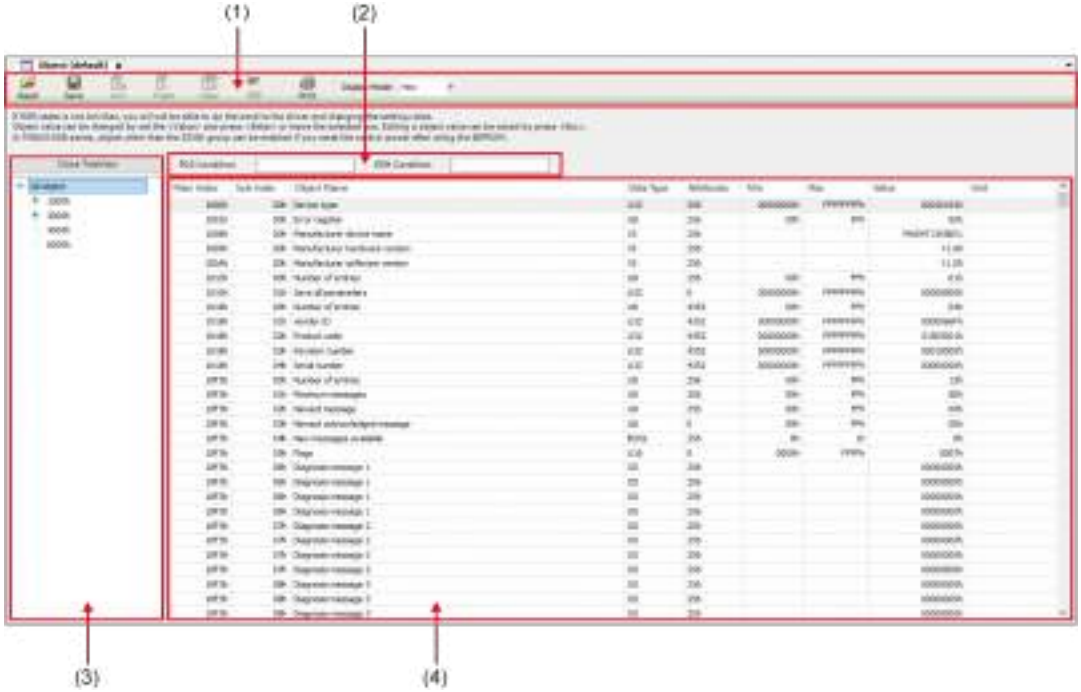
15.8 Object Window

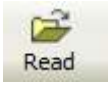
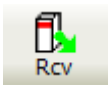
Troubleshooting can be performed more easily by displaying and editing the list of objects on the amplifier side without using a host device.

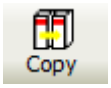
Info.

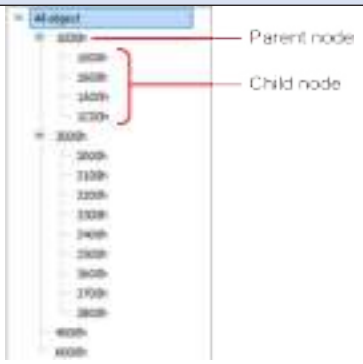
- Some objects affect the behaviors of servo amplifiers or motors. Therefore, before changing the objects, carefully read the instruction manual and other technical references for the servo amplifier and pay careful attention when changing them.

15.8.1 Configuration of Object Window



No.	Name	Function		
(1)	Toolbar	The toolbar consists of basic operation commands related to objects, such as save and read.		
		Icon	Name	Function
			Read	Reads objects from the file (.obj5). When this button is enabled, you can specify a file by drag-and-drop operation.
			Save	Writes objects to the file (.obj5).
			Rcv	Receives objects from the servo amplifier.

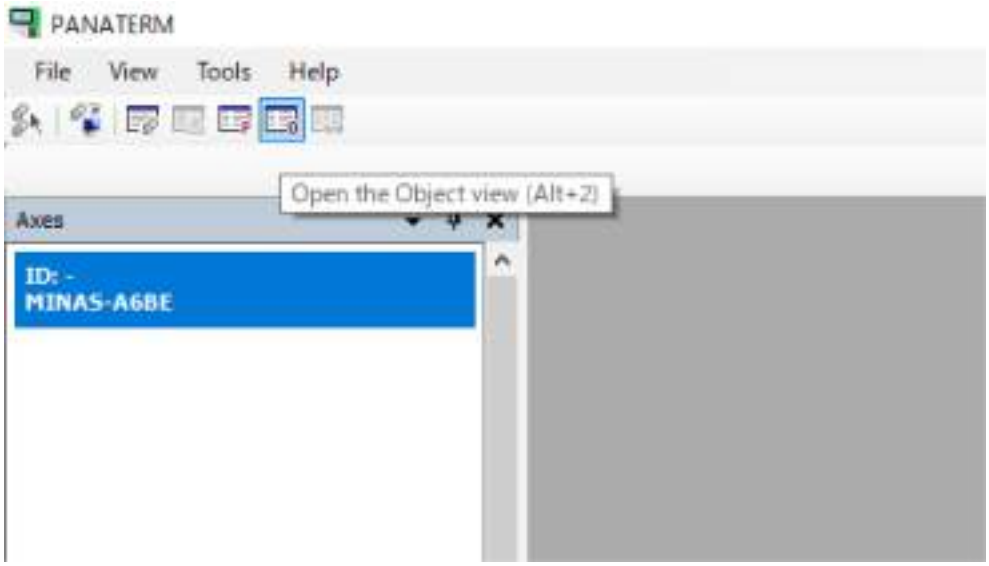
<table>
<tr><th>No.</th><th>Name</th><th colspan="3">Function</th></tr>
<tr><td rowspan="5"></td><td rowspan="5"></td><td>  </td><td>Trans</td><td>Transmits objects to the servo amplifier.
If you turn OFF the power supply to the servo amplifier without writing to EEPROM, the object will be reverted back to the value before the change was made.</td></tr>
<tr><td>  </td><td>Copy</td><td>Copies the objects of a servo amplifier to servo amplifiers for other axes.</td></tr>
<tr><td>  </td><td>EEP</td><td>Writes objects to EEPROM of the servo amplifier.
Do not turn OFF the power supply to the servo amplifier and the PC while data is being written to EEPROM.
If the power supply is cut off while data is being written, the data content is not warranted.</td></tr>
<tr><td>  </td><td>Print</td><td>Prints objects.</td></tr>
<tr><td>  </td><td>Display mode</td><td>Changes the numerical display of the object being displayed.
Hex: Displayed in hexadecimal number and “h” is placed at the end of the number.
Dec: Displayed in decimal number and a sign is set.
Bin: Displayed in binary number and “b” is placed at the end of the number.
The values in the Min-Max column are displayed in hexadecimal number.</td></tr>
<tr><td rowspan="2">(2)</td><td rowspan="2">Condition monitor</td><td colspan="3">PDS Condition
Displays the PDS condition on the servo amplifier side.
It varies depending on the object value of 6041h-00h.</td></tr>
<tr><td colspan="2">ESM Condition
Displays the condition that indicates whether the object can be rewritten on the servo amplifier side.
Rewriting is possible when “Operational” is displayed.</td><td>Full display</td></tr>
<tr><td>(3)</td><td>Object tree</td><td colspan="3">When a node is selected from the object tree, related objects are displayed in the object setting field.
Operate Close TreeView or Open TreeView to hide or display the object tree.
Select “All object” to display all objects.
When a parent node is selected, all child node objects under the selected node are displayed.
When a child node is selected, objects of the selected node are displayed.
For details of each object, refer to the instruction manual and other technical references for the servo amplifier.</td></tr>
</table>

<table>
<tr><th>No.</th><th>Name</th><th colspan="2">Function</th></tr>
<tr><td></td><td></td><td colspan="2"> 
</td></tr>
<tr><td rowspan="11">(4)</td><td rowspan="11">Object setting field</td><td colspan="2">Allows the user to edit and set an object.</td></tr>
<tr><td>Name</td><td>Function</td></tr>
<tr><td>"Main Index"</td><td>Displays the Main Index of an object.</td></tr>
<tr><td>Sub Index</td><td>Displays the Sub Index of an object.</td></tr>
<tr><td>Object Name</td><td>Displays the name of an object.</td></tr>
<tr><td>Data Type</td><td>Displays the data type of an object.
I8: Integer 8 type
I16: Integer 16 type
I32: Integer 32 type
U8: Unsigned 8 type
U16: Unsigned 16 type
U32: Unsigned 32 type
Bool: Boolean type
OS: Octet String type
VS: Visible String type</td></tr>
<tr><td>Attributes</td><td>Displays the attributes of an object.
RO: Read-only attribute, exclusively for reading.
RW: Read/write attribute, possible to read or write.</td></tr>
<tr><td>Min</td><td rowspan="2">Displays the setting range of an object.
The setting range is not displayed for an object whose Data Type is “OS” or “VS”.</td></tr>
<tr><td>Max</td></tr>
<tr><td>Value</td><td>Displays the value of an object. The value can be changed if the object attribute is “RW” and the set value is a numerical value.
Note that there are input restrictions for each Display mode setting. The following values can be input.
Hex: 0 to 9, A to F (After editing, “"h"” is automatically placed at the end of the number.)
Dec: 0 to 9 and minus sign “-”
Bin: 0 and 1 (“"b"” is automatically placed at the end of the number.)
After changing the value, press the <ENTER> key or click the [Trans] button.
To return a value to its original value press the <ESC> key.</td></tr>
<tr><td>Unit</td><td>Displays the unit of an object set value.</td></tr>
</table>

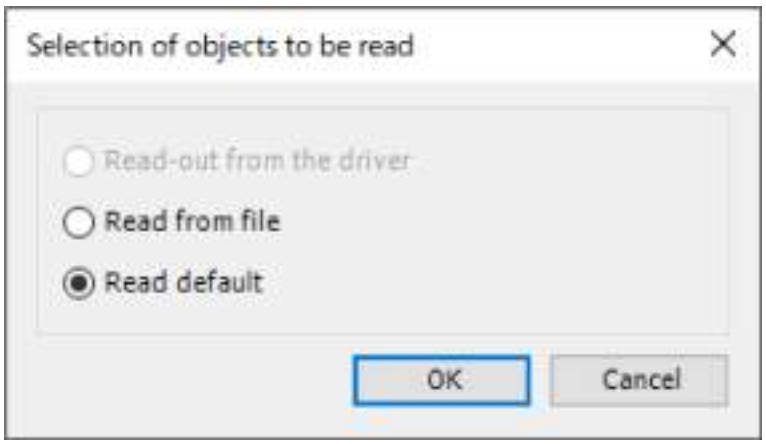
15.8.2 Setting Objects

Procedure

1. From the menu bar on the main pane, select **View>Object**. Alternatively, on the toolbar, click the "Open the Object view" icon.



The "Selection of objects to be read" dialog box will be displayed.



☐**Read-out from the amplifier.**

Communicates with the connected servo amplifier and reads objects set in the servo amplifier.

If this mode is selected, object values will be reflected in the servo amplifier as soon as they are changed.

☐Read from file

Reads the parameter file ("obj5") that was edited previously. If communication is performed with the servo amplifier, object values will be reflected in the servo amplifier as soon as they are changed.

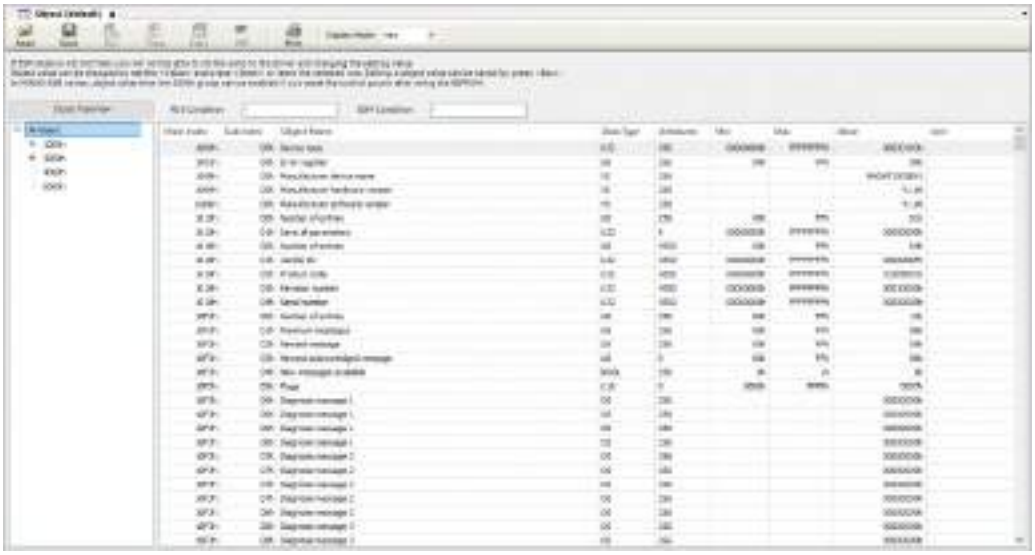
☐Read default

Reads the standard default settings of the servo amplifier that were saved during installation.

If communication is performed with the servo amplifier, object values will be reflected in the servo amplifier as soon as they are changed.

2. Select one of the three options above for reading the object settings and click the [OK] button.

The Object window will be displayed.



3. After changing the object settings, click the [EEP] button to write the object settings to the EEPROM of the servo amplifier.

4. Click the [×] button on the Object window to close the Object window.

15.9 MINAS Parameters for the GM1 Controller

Some parameters for servo amplifiers on the A5B / A6B side affect the behaviors of the EtherCAT-compatible GM1 controller. Use the following parameters.

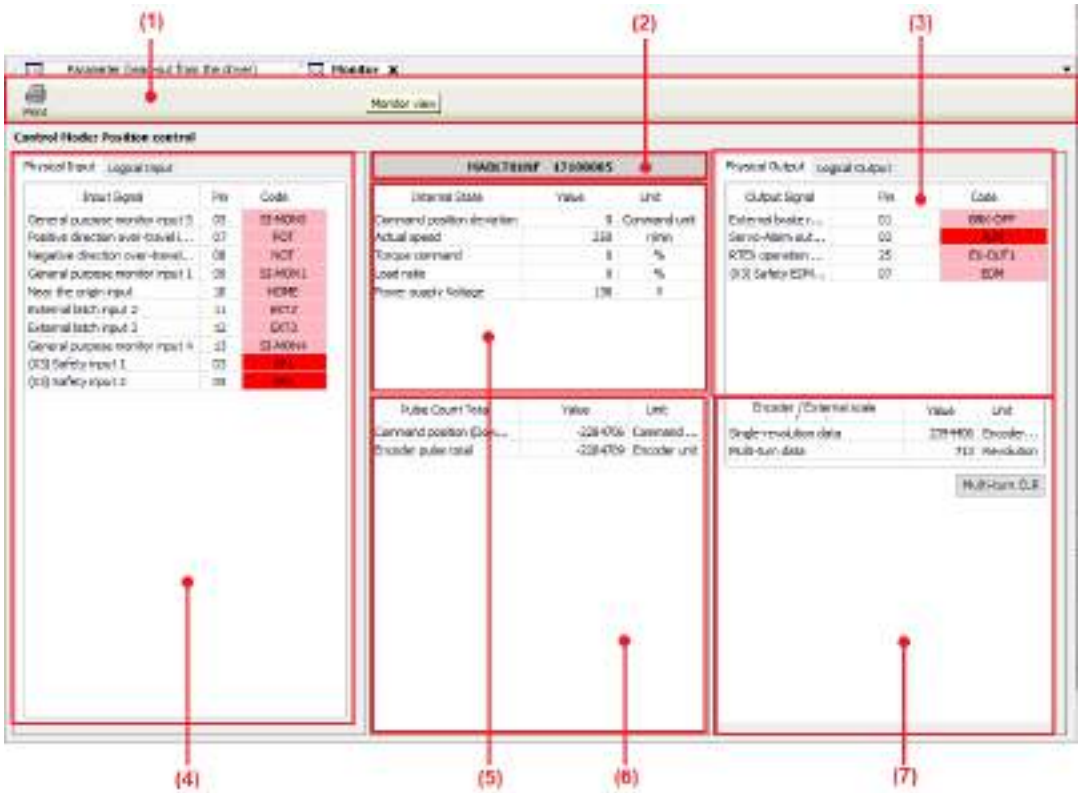
No.	Name	Settings	Standard factory default setting
Pr5.04	Over-travel inhibit input setup	Use setting value 1. (Recommended)	1 (Note 1)

(Note 1) We recommend that the set value should not be changed judging from the characteristics of the GM1 and MINAS.

15.10 Monitor Window

The Monitor window displays the operating states of servo amplifiers and motors, I/O signals, internal statuses, and other information and also allows the user to check them.

15.10.1 Configuration of Monitor Window



NO.	Name	Description
(1)	Toolbar	The toolbar consists of basic operation commands related to parameters. Icon: Print — Name: Print — Function: Prints the contents of the Monitor window.
(2)	Amplifier model name and serial number	Displays the model name and serial number of the servo amplifier.
(3)	Output signal status monitor	Displays the status of each output signal. The tab can be switched between "Physical Output" and "Logical Output". Physical Output – Displays the status of output signals from the servo amplifier. Red: Indicates that output transistor is ON Pink: Indicates that output transistor is OFF

<table>
<tr><th>NO.</th><th>Name</th><th>Description</th></tr>
<tr><td></td><td></td><td>Logical Output – Displays the status of signals within the servo amplifier.
Red: Indicates that signal status is active
Pink: Indicates that signal status is inactive</td></tr>
<tr><td>(4)</td><td>Input signal status monitor</td><td>Displays the status of input signals. The tab can be switched between "Physical Input" and "Logical Input".
Physical Input – Displays the status of input signals to the servo amplifier.
Red: Indicates that COM- is connected
Pink: Indicates that signal status is open
Logical Input – Displays the status of signals within the servo amplifier.
Red: Indicates that signal status is active
Pink: Indicates that signal status is inactive</td></tr>
<tr><td>(5)</td><td>Internal status monitor</td><td>Displays the internal status of the servo amplifier.
<table><tr><th>Name</th><th>Function</th></tr><tr><td>Commanded position deviation</td><td>Displays the position deviation of a command unit.</td></tr><tr><td>Actual speed</td><td>Displays the monitor speed</td></tr><tr><td>Torque command</td><td>Displays the torque command.</td></tr><tr><td>Load factor</td><td>Displays the ratio relative to the rated load. Adjust the operation pattern so that 100% is not exceeded.</td></tr><tr><td>Power supply voltage value</td><td>Displays the voltage (voltage between the P and N terminals) of power supply to the servo amplifier.</td></tr></table></td></tr>
<tr><td>(6)</td><td>Pulse sum monitor</td><td>Displays the sum of command and encoder pulses received by the servo amplifier.</td></tr>
<tr><td>(7)</td><td>Encoder information monitor</td><td>Displays encoder information.
<table><tr><td>Single-turn data</td><td>Displays an absolute position when the motor makes no more than a single turn.</td></tr><tr><td>Multi-turn data</td><td>Displays how many turns the motor made after "Clear" operation.</td></tr></table>Clicking "Clear Multi-turn" resets the multi-turn data stored in the encoder to "0" and clears all encoder errors.
Note: Before using "Clear Multi-turn", check the precautions on use. To clear encoder errors, you may need to restart the servo amplifier.</td></tr>
</table>

(Note 1) Because Ethernet communication is used to transfer data between the servo amplifier and PC, there is a difference or delay between the value displayed on the screen and the actual value of the servo amplifier.

(Note 2) When the polarity is "+", symbol "+" is not displayed.

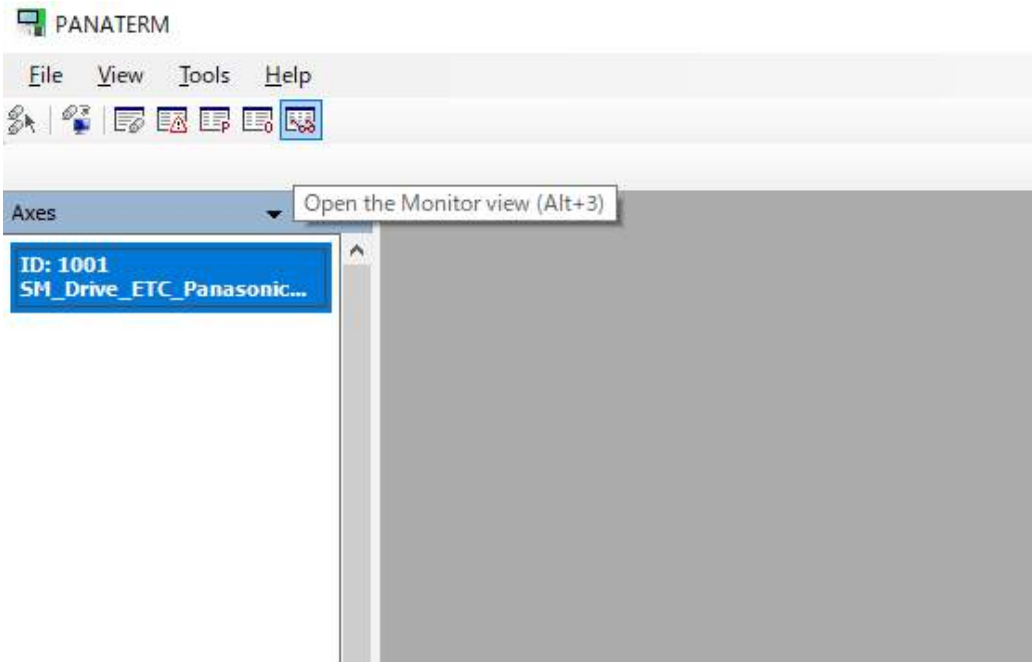
(Note 3) The monitor function is not a measuring instrument. Use the values displayed in the Monitor window as a guide.

(Note 4) If the servo amplifier outputs "Error 40.0 Error protection from absolute system failure" or "Error 42.0 Error protection from absolute overspeed", execute "Clear Multi-turn". Unless the absolute encoder is reset, the alarm cannot be cleared.

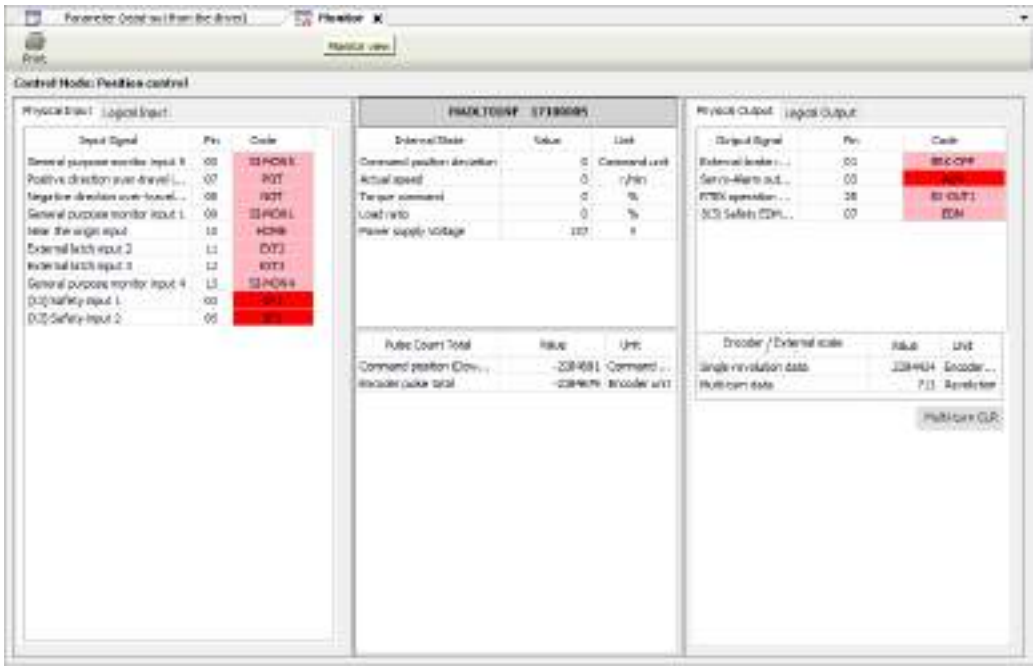
15.10.2 Checking the Monitor Window

Procedure

1. From the menu bar on the main pane, select **View**>**Monitor**. Alternatively, on the toolbar, click the "Open the Monitor view" icon.



The Monitor window will be displayed.



2. Check each item.
   Check the input signal state, output signal state, and the internal status of the servo amplifier.

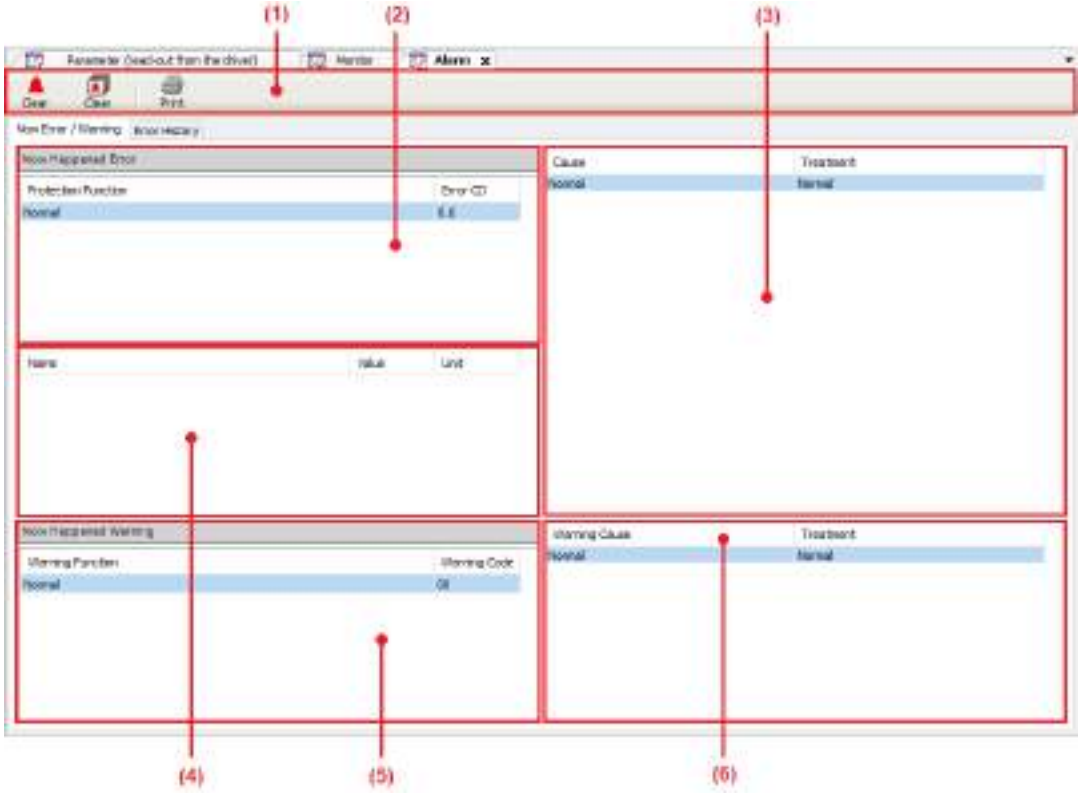
3. Click the [×] button on the Monitor window.
   The Monitor window will be closed.

15.11 Alarm Window

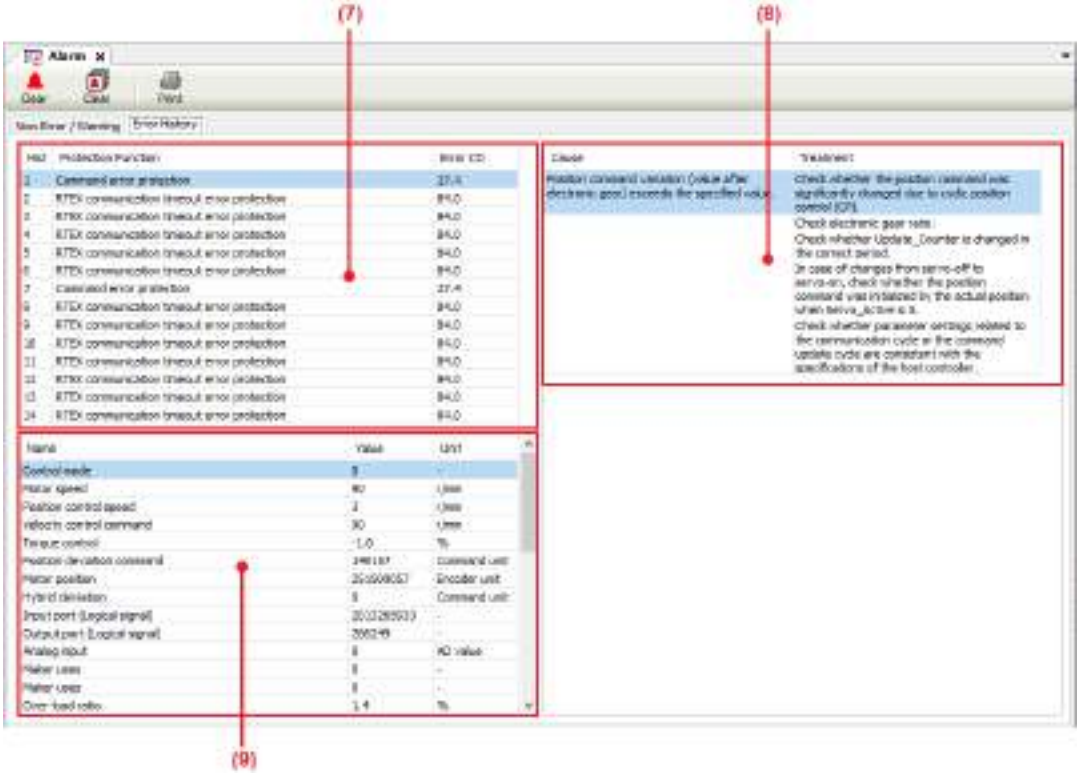
The Alarm window allows the user to check error status when the front panel of the servo amplifier is blinking due to motor operation failure or for some other reason.

15.11.1 Configuration of Alarm Window

Display of the current errors and warnings (only during communication with servo amplifier)



Display of error histories



No.	Name	Description
(1)	Toolbar	<table><tr><th>Icon</th><th>Name</th><th>Function</th></tr><tr><td>  </td><td>Clear</td><td>Allows the user to clear the current alarm history. If you click this button after eliminating the cause of the alarm, the current alarm will be cleared and normal operation will be performed. However, you cannot clear any alarms that cannot be cleared by alarm clear input signals of servo amplifiers. In such a case, turn off the servo amplifier, eliminate the cause of the alarm, and then turn the power on again.</td></tr><tr><td>  </td><td>Clear</td><td>Allows the user to clear error histories.</td></tr><tr><td>  </td><td>Print</td><td>Prints error-related information.</td></tr></table>
(2)	Current error display area	Displays the alarm numbers and names of all errors that are currently occurring.

No.	Name	Description
		The alarm displayed on the top of the list is the alarm displayed on the front panel of the servo amplifier.
(3)	Error cause / treatment display area	Displays the cause and treatment of the selected error.
(4)	Motor internal status display area	Displays the motor internal status in the event of an alarm.
(5)	Current warning display area	Displays the warning numbers and names of all warnings that are currently occurring.
(6)	Warning cause / treatment display area	Displays the cause and treatment of the selected warning.
(7)	Error history display area	Displays the order of error histories, alarm numbers, and error names.
(8)	Error cause / treatment display area	Displays the cause and treatment of the selected error.
(9)	Motor internal status display area	Displays the motor internal status in the event of an alarm.

(Note 1) Some alarms cause tripping as errors but are not recorded in error histories. For alarms that are not recorded in error histories, refer to the instruction manual of the servo amplifier.

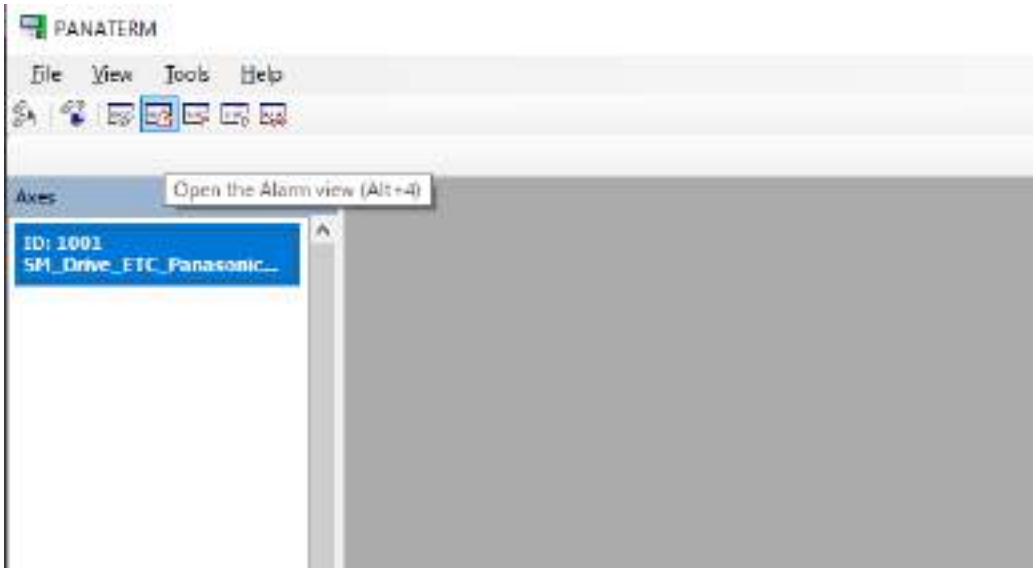
(Note 2) Up to 14 error histories are stored. When more than 14 errors occur, error histories are erased in chronological order (the oldest error history is erased first).

(Note 3) Up to three histories of motor internal status in the event of an alarm are stored. If an alarm occurs immediately after the power is turned on, motor internal status may not be captured normally.

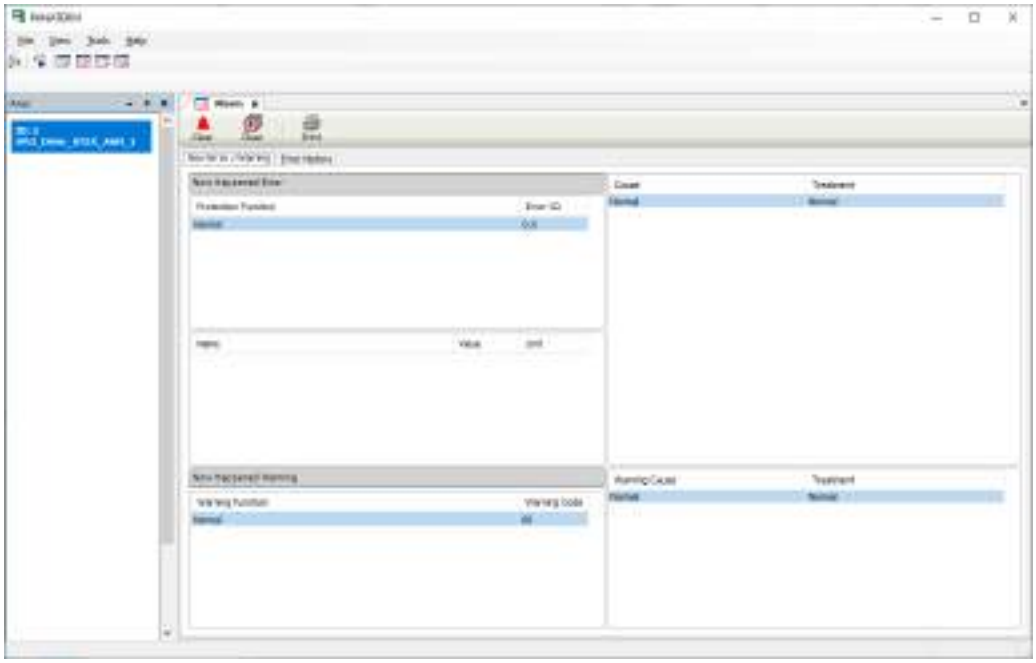
15.11.2 Checking Alarms

Procedure

1. From the menu bar on the main pane, select **View**>**Alarm**. Alternatively, on the toolbar, click the "Open the Alarm view" icon.



The Alarm window will be displayed.



2. Check for any errors that are currently occurring.
   Click the "Now Error / Warning" tab and check for any errors that are currently occurring.
3. Check for any errors that occurred in the past.
   Click the "Error History" tab and check for any errors that occurred in the past.
4. Click the [×] button on the Alarm window.
   The Alarm window will be closed.

15.12 Other Functions

15.12.1 Language Setting Function

This function allows the user to set the display language of PANATERM Lite for GM.
The default setting is the same language as the one set in GM Programmer.

Procedure

1. Select a language from the menu bar tool.
   The language set in PANATERM Lite for GM will be switched.

Info.

- The display language setting of PANATERM Lite for GM is linked with that of the GM Programmer.

15.12.2 Help Function

While performing operation in PANATERM Lite for GM, you can start the Help function to check information such as operating methods.

Procedure

1. From the menu bar, select **Help**>**PANATERM Lite for GM Help**.
   "PANATERM Lite for GM Operation Guide" will be started.

15.12.3 Version Display Function

This function allows the user to check the version, license, and other information for PANATERM Lite for GM.

Procedure

1. From the menu bar, select **Help**>**About**.

2. Click a desired button at the bottom of the window.

Button	Description
Version Info	Displays information about the plug-ins that have been applied and the operating system of the PC that is used.
License Info	Displays license information for the software used by PANATERM Lite for GM.

15.13 Troubleshooting for Servo Amplifiers and Motors

This section explains how to resolve problems.

15.13.1 I Cannot Set up

Symptom	Action method
I cannot set up	● Refer to the chapter related to the system configuration that you need, and check that the PC in which you install the software satisfies the necessary conditions. In particular, note that all the necessary service packs for the operating system must have been applied. ● The installer may have been damaged due to download failure. Clear the browser cache and then download again.

15.13.2 I Cannot Communicate

Symptom	Solution
After PANATERM Lite for GM is started, the servo amplifier is not recognized in the window for connecting to the servo amplifier.	● Check that the power to the control circuit of the servo amplifier is turned on. ● Check the USB communication cable for loose connections or breakage or check whether a correct cable is used. ● Check whether the USB port on the PC is functioning normally. (Refer to the instruction manual of the PC.) ● Check whether the USB driver is installed correctly.
After PANATERM Lite for GM is started, the GM1 controller is not recognized in the window for connecting to the GM1 controller.	● Check that the power to the GM1 controller is turned on. ● Check the USB communication cable or Ethernet cable for loose connections or breakage or check whether a correct cable is used. ● Check whether the USB port on the PC is functioning normally. (Refer to the instruction manual of the PC.) ● Check whether the LAN adapter on the PC is functioning normally. (Refer to the instruction manual of the PC.) ● Check whether the USB driver is installed correctly. ● Check whether Gateway is running.

15.13.3 I Cannot Print

Symptom	Action method
I cannot print	● Check whether the printer is connected correctly. Check that the printer driver is operating normally, by printing a test page, for example. ● You may be unable to print any comment that consists of too many characters in a single line. Split such a comment into multiple lines so that the number of characters per line falls within the printable range.

15.13.4 I Cannot Set up Axes

Symptom	Solution
The number of servo amplifiers connected does not match the number of servo amplifiers checked by performing a search.	● Check whether the axis name (ID) of the servo amplifier connected to the PC is 0, the respective axis names (IDs) of other servo amplifiers are 1 to 15, and there are any duplicate axis names (IDs). ● Check the communication cable for loose connections or breakage or check whether a correct cable is used.

15.13.5 PANATERM Lite for GM Does Not Behave Normally

Symptom	Solution
PANATERM Lite for GM responds or acts slowly	● Close any windows that are not used. All windows communicate with the servo amplifier in certain cycles even if they are hidden under other windows. ● If USB devices other than servo amplifiers are connected, reduce USB communication load by pausing their operations or taking some other action.
The window does not open or icons appear garbled	● The PC is running out of memory. Temporarily close PANATERM Lite for GM and other applications that are not used. Alternatively, turn the PC off and then on, and start PANATERM Lite for GM again.
PANATERM Lite for GM does not respond	● Press the <Ctrl> key + <Alt> key + <Delete> key simultaneously to invoke the window for forcibly terminating programs, and terminate PANATERM Lite for GM.
PANATERM Lite for GM suddenly terminates	● Start PANATERM Lite for GM again.
PANATERM Lite for GM does not start	● Microsoft .NET Framework Ver. 4.6.1 may have failed to be installed. Refer to the website of Microsoft and install Microsoft .NET Framework Ver. 4.6.1 directly on the PC that you use. After installing Microsoft .NET Framework Ver. 4.6.1, run the PANATERM Lite for GM installer again.

15.13.6 The Parameter Window Does Not Behave Normally

Symptom	Solution
The changed parameter value returns to its original value	● After changing the parameter value, press the <Enter> key or click the [Trans] button. If you move to another parameter or make changes in the window without performing either of these operations, any change to the parameter value will be canceled. ● When the values read from a file are displayed in the window, changed parameter values are not sent to the servo amplifier. To send changed parameter values to the servo amplifier, click the [Trans] button.

15.13.7 The Object Window Does Not Behave Normally

Symptom	Solution
The object window does not open.	● The object editor window can be displayed only when a series that uses the EtherCAT communication is selected. (Example: MINAS-A5B, MINAS-A6B)
The object set value cannot be changed or transmitted.	● Check if the ESM Condition is set to “Operational”.
The object set value returns to its original value.	● After changing the object value, press the <Enter> key or click the [Trans] button. If you move to another object or make changes in the window without performing either of these operations, any change to the object value will be canceled. ● When the values read from a file are displayed in the window, changed object values are not sent to the servo amplifier. To send changed object values to the servo amplifier, click the [Trans] button.

15.13.8 The Monitor Window Does Not Behave Normally

Symptom	Solution
Monitor display does not change	● There is a possibility that communication with the servo amplifier has been disrupted and the PC is in an offline state. Check the connection status of the servo amplifier.

15.13.9 The Alarm Window Does Not Behave Normally

Symptom	Action method
Error histories are not displayed	● If no error has occurred before or error histories have been cleared, error histories are not displayed. ● Supplementary information about errors is displayed only when the most recent, the second most recent, or the third most recent error history is selected. Select an error history number again. ● Errors that are not recorded in error histories are not displayed in error histories even if they occur.

(MEMO)

Appendix Warranty / Cautions for Proper Use

Warranty App-2
Warranty Period App-2
Warranty Scope App-2
Cautions for Proper Use App-3

Warranty

Warranty Period

Warranty period shall be 12 months from the ex-factory date or 18 months from the date of manufacturing.

This Warranty shall be exempted in the following cases,

1. Defects resulting from misuse and/or repair or modification by the customer.
2. Defects resulting from drop of the Product or damage during transportation.
3. Defects resulting from improper usage of the Product beyond the Specifications.
4. Defects resulting from fire, earthquake, lightening, flood, damage from salt, abnormal voltage or other Act of God, or other disaster.
5. Defects resulting from the intrusion of foreign material to the Product, such as water, oil or metallic particles.

   Parts exceeding their standard lifetime specified in this document are excluded.

Warranty Scope

Panasonic warrants the replacement of the defected parts of the Product or repair of them when the defects of the Product occur during the Warranty Period, and when the defects are under Panasonic responsibility.This Warranty only covers the Product itself and does not cover any damage incurred by such defects.

Panasonic in accordance with 'Warranty Period' records,in any case,the machine state is poor,and cause damage to your company and the third party,all liability,Panasonic is not responsible.

1. The machines are not assembled in accordance with the instructions or precautions noted in this specification.
2. When the machine does not match the product assembled in the machine.
3. This specification does not depend on your company.
4. When the machine condition is not caused by Panasonic reasons.

Cautions for Proper Use

- Practical considerations for exporting the product or assembly containing the product
  When the end user of the product or end use of the product is associated with military affair or weapon, its export may be controlled by the Foreign Exchange and Foreign rade Control Law. Complete review of the product to be exported and export formalities should be practiced.
- This product is intended to be used with a general industrial product, but not designed or manufactured to be used in a machine or system that may cause personal death when it is failed.
- Installation, wiring, operation, maintenance, etc., of the equipment should be done by qualified and experienced personnel.
- Install a safety equipments or apparatus in your application, when a serious accident or loss of property is expected due to the failure of this product.
- This product is designed for general industrial equipments.Don´t use this product under special conditions such as nuclear energy control, aerospace equipments, transportation, medical equipment, various safety equipments or special equipments.
- The wiring condition(earth wire method and cables length and shield cable condition of signal lines) may affect the noise resistance,please confirm the noise resistance of the machine.
- Failure of this product depending on its content, may generate smoke of about one cigarette. Take this into consideration when the application of the machine is clean room related.
- Product overload can cause the goods to fall,please follow the marking.
- Do not use benzine, thinner, alcohol, acidic cleaner and alkaline cleaner because theycan discolor or damage the exterior case.
- This product shall be treated as industrial waste when you dispose.
- This product related standards,laws and the user is responsible for matching between machine and components in terms of configuration, dimensions, life expectancy, characteristics, when installing the machine or changing specification of the machine. The user is also responsible for complying with applicable laws and regulations.
- The product will not be guaranteed when it is used outside its specification limits.
- Parts are subject to minor change to improve performance.

(MEMO)

Revision History

The manual code is shown at the bottom of the cover page.

Date of issue	Manual code	Revision details
August 2021	WUME-GM1ETCOP-01	First edition
February 2022	WUME-GM1ETCOP-02	2nd edition ● Updated the CNC control (G codes, instructions). ● Newly added the Recipe Manager.
April 2022	WUME-GM1ETCOP-03	3rd edition ● Changed the Company name

(MEMO)

(MEMO)

Please contact .....

Industrial Device Business Division,
Panasonic Industry Co., Ltd.
7-1-1 Morofuku, Daito City, Osaka, 574-0044, Japan
industrial.panasonic.com/ac/e/

WUME-GM1ETCOP-03