MITSUBISHI ELECTRIC

Programmable Automation Controllers

MELSEC MX Controller

MELSEC MX Controller Programming Technique Guide

-MXR300-16
-MXR300-32
-MXR300-64
-MXR500-128
-MXR500-256
-SW1DND-GXW3-J

SAFETY PRECAUTIONS

(Read these precautions before using this product.)

Before using this product, please read this manual and the relevant manuals introduced in this manual carefully and pay full attention to safety to handle the product correctly.

In this manual, the safety precautions are classified into two levels: "⚠WARNING" and "⚠ CAUTION".

⚠ **WARNING**	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
⚠ **CAUTION**	Indicates that incorrect handling may cause hazardous conditions, resulting in minor or moderate injury or property damage.

Under some circumstances, failure to observe the precautions given under "⚠CAUTION" may lead to serious consequences.

Observe the precautions of both levels because they are important for personal and system safety.

Make sure that the end users read this manual and then keep the manual in a safe place for future reference.

CONDITIONS OF USE FOR THE PRODUCT

(1) MELSEC programmable controller ("the PRODUCT") shall be used in conditions;
i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and
ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.
MITSUBISHI ELECTRIC SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI ELECTRIC USER'S, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT.
("Prohibited Application")
Prohibited Applications include, but not limited to, the use of the PRODUCT in;
- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

Notwithstanding the above restrictions, Mitsubishi Electric may in its sole discretion, authorize use of the PRODUCT in one or more of the Prohibited Applications, provided that the usage of the PRODUCT is limited only for the specific applications agreed to by Mitsubishi Electric and provided further that no special quality assurance or fail-safe, redundant or other safety features which exceed the general specifications of the PRODUCTs are required. For details, please contact the Mitsubishi Electric representative in your region.

(3) Mitsubishi Electric shall have no responsibility or liability for any problems involving programmable controller trouble and system trouble caused by DoS attacks, unauthorized access, computer viruses, and other cyberattacks.

INTRODUCTION

Thank you for purchasing the Mitsubishi Electric programmable controllers.
This manual describes effective programming techniques for using the following relevant products of MELSEC MX controllers (MX-R model).
Before using this product, please read this manual and the relevant manuals carefully and develop familiarity with the functions and performance of the controller to handle the product correctly.
When applying the program examples provided in this manual to an actual system, ensure the applicability and confirm that it will not cause system control problems.
Please make sure that the end users read this manual.

Relevant products

MXR300-16, MXR300-32, MXR300-64, MXR500-128, MXR500-256

CONTENTS

RELEVANT MANUALS

Manual name	Description
MELSEC MX Controller Programming Technique Guide [BCN-89999-9836] (this manual)	Programming techniques related to the MX controllers to improve the readability of the project and speed of program conversion, reduce the scan time, optimize motion control, improve efficiency of labels, and increase the speed of the CC-Link IE TSN communication
MELSEC MX Controller (MX-R Model) User's Manual [SH-082641ENG]	Procedures before operation, specifications, devices, memory, functions, parameters, and troubleshooting of the controller
MELSEC MX Controller (MX-R Model) Programming Manual [SH-082644ENG]	Programming language specifications, controller instructions, standard functions/function blocks, and specifications of motion control function blocks
GX Works3 Operating Manual [SH-081215ENG]	System configuration, parameter settings, and online operations of GX Works3

TERMS

Unless otherwise specified, this manual uses the following terms.

Term	Description
Global label	A label that is valid for all the program data when multiple program data are created in the project. There are two types of global labels: a module specific label (module label), which is automatically generated by GX Works3, and an optional label, which can be created for any specified device.
Device station	Stations other than the master station (local stations and remote stations)
Program block	A group of POUs that configure a program
Master station	A station that controls the entire network. This station can perform cyclic transmission and transient transmission with all stations. Only one master station can be used in a network.
Motion control FB	An FB that is related to motion control. Its name starts with "MC_" or "MCv_".
Module label	A label that represents one of memory areas (I/O signals and buffer memory areas) specific to each module in a given character string. For the module used, GX Works3 automatically generates this label, which can be used as a global label.
Label	A variable consisting of a specified string used in I/O data or internal processing
Remote station	A station that exchanges I/O signals (bit data) and I/O data (word data) with another station by cyclic transmission. This station can perform transient transmission.
Link device	Devices (RX, RY, RWr, RWw, LB, LW) in the CC-Link IE TSN function part or a network module
Local station	A station that performs cyclic transmission and transient transmission with the master station and other local stations
Virtual drive axis	A virtual axis that can generate a command virtually
Axis	A target to perform the motion control
Axis variable	An AXIS_* type variable instance including parameters and data related to the axis
Real encoder axis	An axis that generates a commanded position from the current position of the encoder connected to a device station. It is used for the master axis of the single axis synchronous control.
Real drive axis	An axis that is linked with a CC-Link IE TSN compatible device that supports the csp/csv/cst mode (sequential command) of the CiA402 drive profile
CC-Link IE TSN	An open network that uses "TSN (Time-Sensitive Networking)", which is an extension of the Ethernet standard, to ensure real-time control and handle information from other open networks simultaneously
CC-Link IE TSN Class	A group of devices and switching hubs compatible with CC-Link IE TSN, ranked according to the functions and performance by the CC-Link Partner Association. For CC-Link IE TSN Class, refer to the CC-Link IE TSN Installation Manual (BAP-C3007ENG-001) published by the CC-Link Partner Association.
FB	FB stands for function block, which is created by componentizing a circuit block that is used repeatedly within a sequence program so that it can be reused within the sequence program. This improves programming efficiency and reduces programming errors, improving program quality.
FB instance	A function block that is inserted into a sequence program
GX Works3	The product name of the software package for the MELSEC programmable controllers
RWr	A remote register of the link device. Word data (16-bit data) input from a device station to the master station. (For some areas in a local station, data are input in the opposite direction.)
RWw	A remote register of the link device. Word data (16-bit data) output from the master station to a device station. (For some areas in a local station, data are input in the opposite direction.)
RX	A remote input of the link device. Bit data input from a device station to the master station. (For some areas in a local station, data are input in the opposite direction.)
RY	A remote output of the link device. Bit data output from the master station to a device station. (For some areas in a local station, data are input in the opposite direction.)

GENERIC TERMS AND ABBREVIATIONS

Unless otherwise specified, this reference uses the following generic terms and abbreviations.

Term	Description
Engineering tool	A tool used for setting up controllers or servo amplifiers, programming, debugging, and maintenance
Controller	A generic term for the MXR300-16, MXR300-32, MXR300-64, MXR500-128, and MXR500-256

1 OVERVIEW

This document describes techniques for the MELSEC MX controllers (hereinafter referred to as controller) to structure and increase the speed of the project and program, optimize the motion control, improve efficiency of labels, and increase the speed of the CC-Link IE TSN communication.

For basic parameter settings and programming of the controller, refer to the following.

📖MELSEC MX Controller (MX-R Model) User's Manual

📖MELSEC MX Controller (MX-R Model) Programming Manual

This manual uses the following version of GX Works3 for explanation.

- GX Works3 Ver.1.115V

Techniques and effects

The following table lists the expected effects of each technique.

Technique		Effect				Reference
		Improved readability/ portability	Reduction of conversion time	Reduction of scan time	Improved synchronization performance	
Structuring programs	Componentizing programs using FBs and structures	○	—	—	—	☞ Page 10 IMPROVEMENT OF PROJECT READABILITY
	Arraying FBs	○	—	—	—	
	Grouping motion axes	○	—	—	—	
	Dividing global labels	○	○	—	—	
Increasing speed of program conversion	Dividing program files	○	○	—	—	☞ Page 27 INCREASING SPEED OF PROGRAM CONVERSION
	Converting programs into FBs	○	○	—	—	
	Reducing programs containing control syntax	○	○	—	—	
	Changing global label (structure) settings	—	○	—	—	
	Reducing consecutive references to the same array element	—	○	○	—	
Reduction of scan time	Replacing a BMOV instruction with multiple instructions	—	—	○	—	☞ Page 53 REDUCTION OF SCAN TIME
	Reducing rising edge execution instructions	—	—	○	—	
	Reducing module access instructions	—	—	○	—	
	Utilizing initial execution type programs	—	—	○	—	
	Omitting arguments and utilizing direct references	—	—	○	—	
	Dividing a circuit block with a large number of instructions	○	—	○	—	
	Utilizing label initial values	○	—	○	—	
Optimization of motion control	Increasing speed of specific axes using multiple cycle setting	—	—	○	—	☞ Page 79 OPTIMIZATION OF MOTION CONTROL
	Synchronizing user programs and motion operations	—	—	—	○	

Technique			Effect				Reference
			Improved readability/ portability	Reduction of conversion time	Reduction of scan time	Improved synchronization performance	
Improvement of label efficiency		Deleting global labels	—	○	—	—	☞ Page 91 IMPROVEMENT OF LABEL EFFICIENCY
		Deleting unused labels	—	○	—	—	
Increasing speed of CC-Link IE TSN communication	Reducing link refresh time	Reducing link refresh time	—	—	○	—	☞ Page 95 INCREASING SPEED OF CC-Link IE TSN COMMUNICATION
	Reducing communication cycle interval/ cyclic transmission time	Equalizing the number of device stations	—	—	○	—	
		Setting the upper limit of transmission size					
		Reducing the number of device stations					
		Deleting link device settings					
		Reducing the points of link device settings					
		Unifying network synchronization communication settings					
		Setting distribution of cyclic transmission					

2 IMPROVEMENT OF PROJECT READABILITY

The readability of a project can be improved by dividing it into small processing units (POUs) to make a hierarchical structure and unifying basic configurations and rules of projects. Improving the project readability makes it easier to identify the part to be modified when the project is changed. Also, it makes the project easier to understand for individuals other than the one who created it.
This chapter describes techniques for improving project readability.

List of techniques

The following table lists the techniques described in this chapter.

Item	Description	Reference
Componentizing programs using FBs and structures	Componentize the programs commonly used in each program block using FBs and structures.	☞ Page 10 Componentizing Programs Using FBs and Structures
Arraying FBs	Array the FBs that are executed at the same time and replace them with an FB combined with a FOR statement.	☞ Page 16 Arraying FBs
Grouping motion axes	Group the axis Nos. by axis type, such as the real drive axis, real encoder axis, and virtual drive axis.	☞ Page 20 Grouping Motion Axes
Dividing global labels	Group global labels into meaningful units and move them to different global label editors.	☞ Page 24 Dividing Global Labels

2.1 Componentizing Programs Using FBs and Structures

Overview

By componentizing FBs and structures that are commonly used in each program block, the amount of code can be reduced, resulting in more readable programs.
Componentization of programs has the following benefits.

Improvement of program portability and reusability

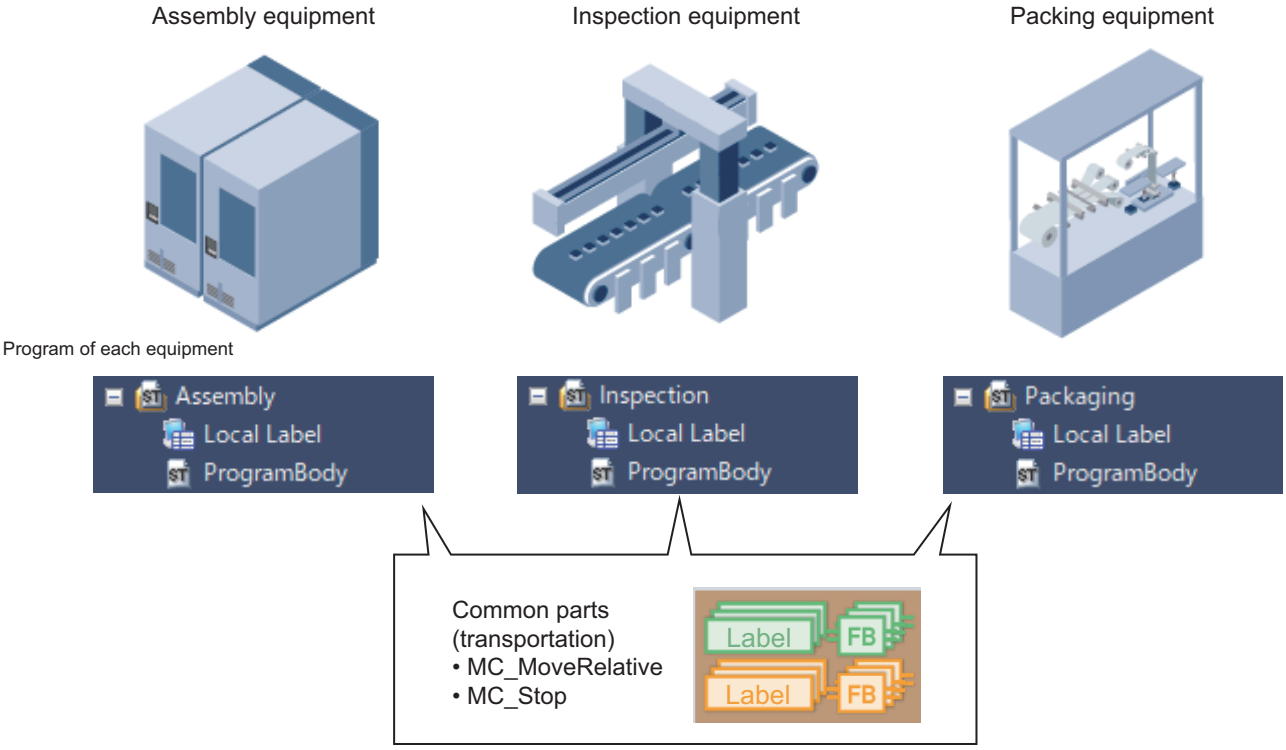
Componentizing programs makes it easier to port and reuse programs on a processing or process basis.

Improvement of program maintainability

By componentizing programs and using the components in multiple processing and processes, only the componentized structures and FBs need to be modified when processing is added or changed, eliminating the need to modify multiple programs individually. This improves the maintainability of programs.

Execution procedure

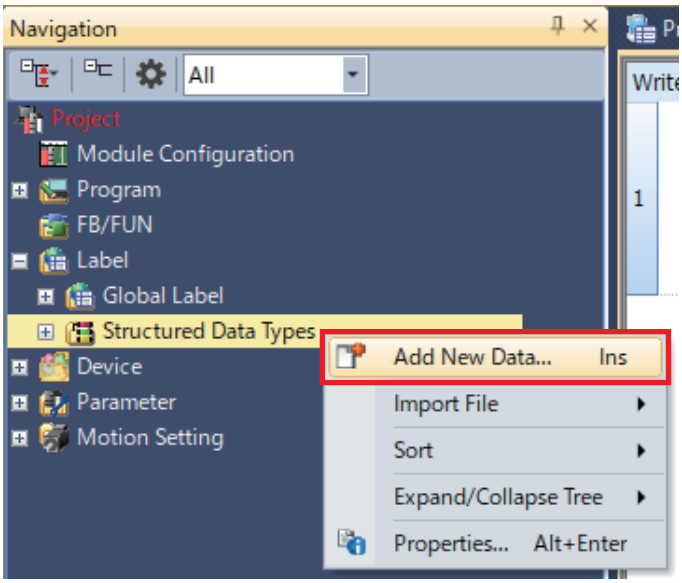
Using the following assembly equipment, inspection equipment, and packing equipment as examples, this section describes the procedure for creating common components for transportation and using them in programs for each equipment.
As the common components for transportation, motion control FBs "MC_MoveRelative" (relative value positioning), which transports workpieces, and "MC_Stop" (forced stop), which stops the operation, are used.



Operating procedure

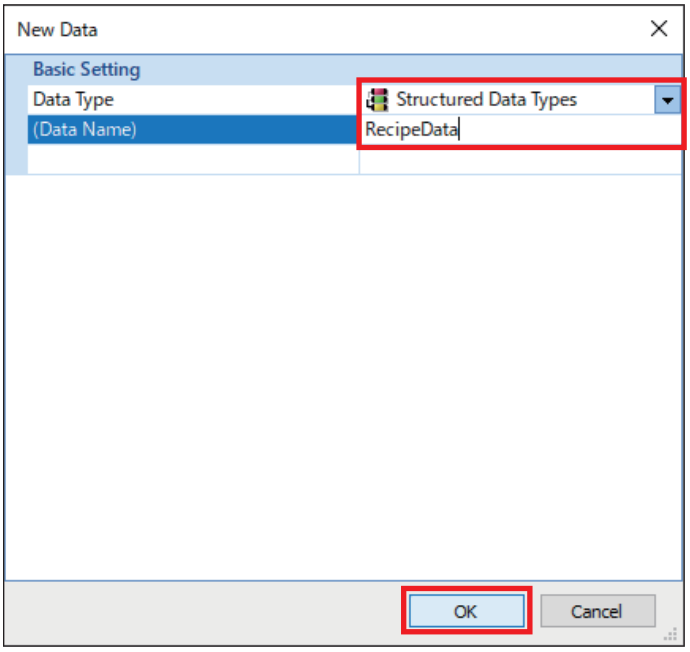
1. Newly create a structure to define the recipe data for transportation and stop in each process.

Navigation window ⇨ [Label] ⇨ Right-click [Structured Data Types] ⇨ [Add New Data]

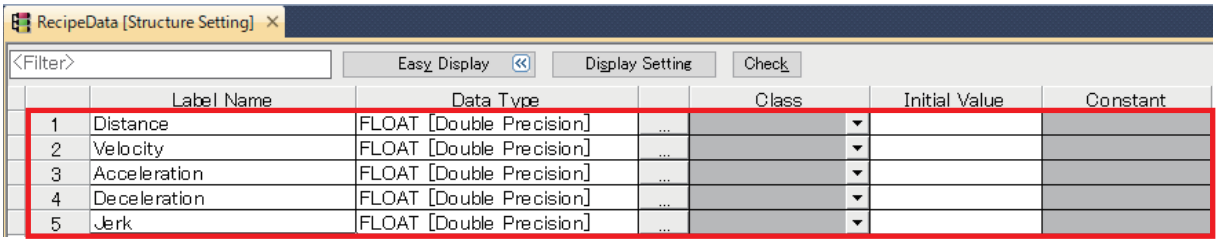


2. Configure the following settings, and then click the [OK] button.

- Data Type: Structured Data Types
- (Data Name): RecipeData



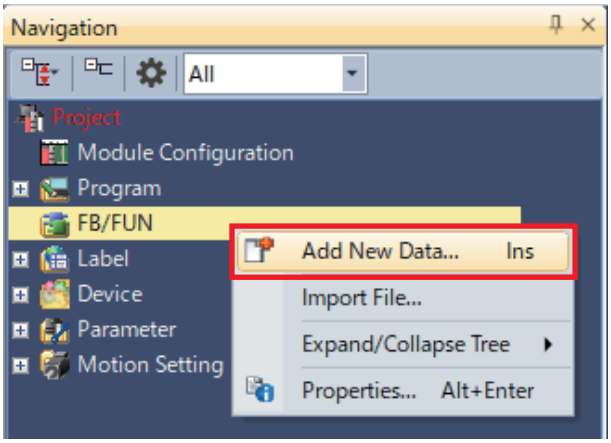
3. As members of the structure "RecipeData", define the following labels commonly used for "MC_MoveRelative" and "MC_Stop".



No.	Label name	Data type
1	Distance	FLOAT [Double Precision]
2	Velocity	FLOAT [Double Precision]
3	Acceleration	FLOAT [Double Precision]
4	Deceleration	FLOAT [Double Precision]
5	Jerk	FLOAT [Double Precision]

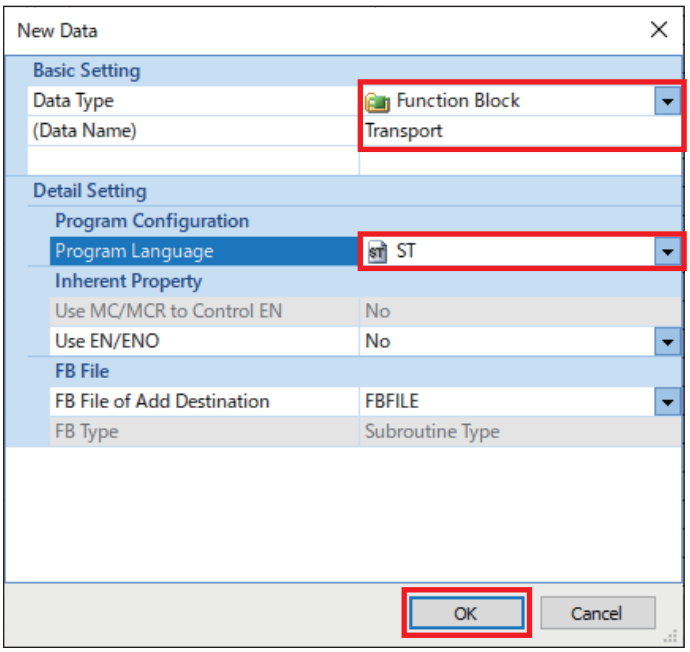
4. Newly create an FB for the common components.

Navigation window ⇨ Right-click [FB/FUN] ⇨ [Add New Data]

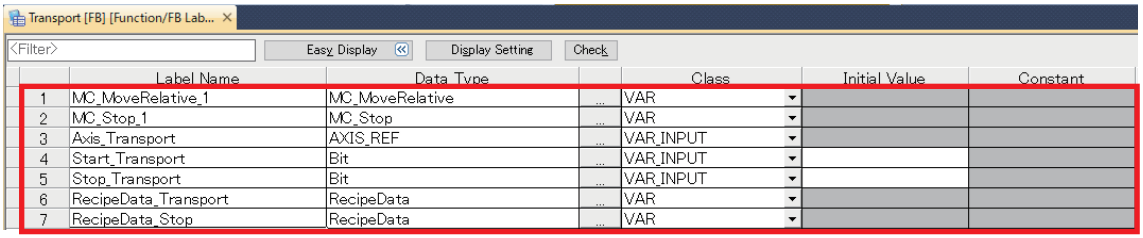


5. Configure the following settings, and then click the [OK] button.

- Data Type: Function Block
- (Data Name): Transport
- Program Language: ST



6. Open the local label editor of the FB "Transport" and define the following FB instances and labels required for transportation.



No.	Label name	Data type	Class	Remarks
1	MC_MoveRelative_1	MC_MoveRelative	VAR	An FB instance of the motion control FB "MC_MoveRelative"
2	MC_Stop_1	MC_Stop	VAR	An FB instance of the motion control FB "MC_Stop"
3	Axis_Transport	AXIS_REF	VAR_INPUT	The structure "AXIS_REF" is used as the data type.
4	Start_Transport	Bit	VAR_INPUT	—
5	Stop_Transport	Bit	VAR_INPUT	—
6	RecipeData_Transport	RecipeData	VAR_PUBLIC	The structure "RecipeData", which is created in steps 1 to 3, is used as the data type.
7	RecipeData_Stop	RecipeData	VAR_PUBLIC	

7. Open the program body of the FB "Transport" and write the following program that executes the motion control FBs "MC_MoveRelative" and "MC_Stop".

Transport [FB] [ST] *

```
1  //Workpiece transportation
2  MC_MoveRelative_1.Axis := Axis_Transport;
3  MC_MoveRelative_1(
4      Execute:= Start_Transport ,
5      Distance:= RecipeData_Transport.Distance ,
6      Velocity:= RecipeData_Transport.Velocity ,
7      Acceleration:= RecipeData_Transport.Acceleration ,
8      Deceleration:= RecipeData_Transport.Deceleration ,
9      Jerk:= RecipeData_Transport.Jerk
10     );
11
12 //Stop
13 MC_Stop_1.Axis := Axis_Transport;
14 MC_Stop_1(
15     Execute:= Stop_Transport ,
16     Deceleration:= RecipeData_Stop.Deceleration ,
17     Jerk:= RecipeData_Stop.Jerk
18     );
```

Row	Processing
1st to 3rd	Specify the label "Axis_Transpor" for the argument "Axis" of the motion control FBs "MC_MoveRelative" and "MC_Stop".
5th to 13th	Execute the motion control FB "MC_MoveRelative". Specify each member of the label "RecipeData_Transport" for the arguments "Distance", "Velocity", "Acceleration", "Deceleration", and "Jerk".
15th to 20th	Execute the motion control FB "MC_Stop". Specify each member of the label "RecipeData_Stop" for the arguments "Deceleration" and "Jerk".

8. Open the local label editor of the program block for the assembly equipment ("Assembly" in this example) and add the FB instances of the FB "Transport" created in steps 4 to 7.

- Label Name: Transport
- Data Type: Transport (FB "Transport")
- Class: VAR

Assembly [PRG] [Local Label Set...

<Filter> | Easy Display | Display Setting | Check

	Label Name	Data Type		Class	Initial Value	Constant
1	Transport	Transport	...	VAR		

9. Open the program body of each equipment and create a transportation program using the common component FB "Transport".

- Program of assembly equipment

Assembly [PRG] [ST] 28Byte

```
//Transportation processing
//Initial setting
Transport.Axis_Transport := Axis0001.AxisRef;
Transport.RecipeData_Transport.Distance := 10000.0;
Transport.RecipeData_Transport.Acceleration := 2000.0;
Transport.RecipeData_Transport.Deceleration := 2000.0;
Transport.RecipeData_Transport.Jerk := 300.0;

//Transportation start
IF  G_Process = G_Process_Assembly  THEN
    IF  NOT G_Stop_Assembly  THEN
        Transport.Start_Transport := TRUE ;
    ELSE
        Transport.Start_Transport := FALSE ;
        Transport.Stop_Transport := TRUE ;
    END_IF;
END_IF;
Transport();

//The following is the dedicated processing for assembly.
```

- Program of inspection equipment

Inspection [PRG] [ST] 28Byte

```
//Transportation processing
//Initial setting
Transport.Axis_Transport := Axis0001.AxisRef;
Transport.RecipeData_Transport.Distance := 20000.0;
Transport.RecipeData_Transport.Acceleration := 2000.0;
Transport.RecipeData_Transport.Deceleration := 2000.0;
Transport.RecipeData_Transport.Jerk := 300.0;

//Transportation start
IF  G_Process = G_Process_Inspection  THEN
    IF  NOT G_Stop_Inspection  THEN
        Transport.Start_Transport := TRUE ;
    ELSE
        Transport.Start_Transport := FALSE ;
        Transport.Stop_Transport := TRUE ;
    END_IF;
END_IF;
Transport();

//The following is the dedicated processing for inspection.
```

- Program of packing equipment

Packaging [PRG] [ST] 28Byte

```
//Transportation processing
//Initial setting
Transport.Axis_Transport := Axis0001.AxisRef;
Transport.RecipeData_Transport.Distance := 30000.0;
Transport.RecipeData_Transport.Acceleration := 2000.0;
Transport.RecipeData_Transport.Deceleration := 2000.0;
Transport.RecipeData_Transport.Jerk := 300.0;

//Transportation start
IF  G_Process = G_Process_Packaging  THEN
    IF  NOT G_Stop_Packaging  THEN
        Transport.Start_Transport := TRUE ;
    ELSE
        Transport.Start_Transport := FALSE ;
        Transport.Stop_Transport := TRUE ;
    END_IF;
END_IF;
Transport();

//The following is the dedicated processing for packing.
```

2.2 Arraying FBs

Overview

By arraying FBs that are executed multiple times simultaneously and executing them repeatedly using a FOR statement, the amount of code in a program is reduced, allowing you to create more readable programs.
Arraying FBs has the following benefit.

Improvement of program portability

Reduction of the amount of code in a program also reduces the amount of code when the program is ported, improving the program portability.

Execution procedure

Using the following motion control program as an example, this section describes the procedure for arraying the motion control FB "MC_Power", which is executed for multiple axis at the same time.

• ST program

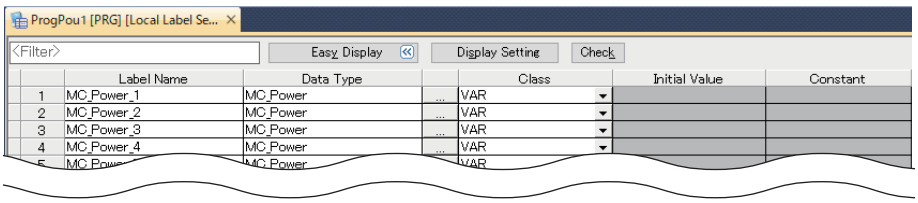
ProgPou1 [PRG] [ST]

```
MC_Power_1(
    Axis:= Axis0001.AxisRef ,
    Enable:= TRUE ,
    ServoON:= TRUE
//  Status=> ?BOOL? ,
//  ReadyStatus=> ?BOOL? ,
//  Busy=> ?BOOL? ,
//  Error=> ?BOOL? ,
//  ErrorID=> ?WORD?
    );

MC_Power_2(
    Axis:= Axis0002.AxisRef ,
    Enable:= TRUE ,
    ServoON:= TRUE
//  Status=> ?BOOL? ,
//  ReadyStatus=> ?BOOL? ,
//  Busy=> ?BOOL? ,
//  Error=> ?BOOL? ,
//  ErrorID=> ?WORD?
    );
```

The motion control FB "MC_Power" is executed for axis Nos. 1 to 100 at the same time.

• Local label



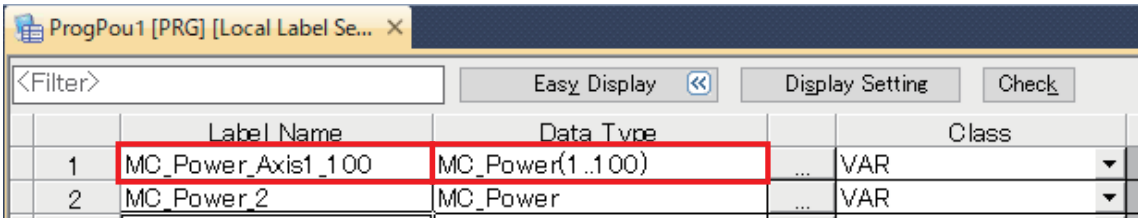
FB instances of "MC_Power_1" to "MC_Power_100"

Operating procedure

1. Set the instance of the FB to be arrayed to a one-dimensional array.

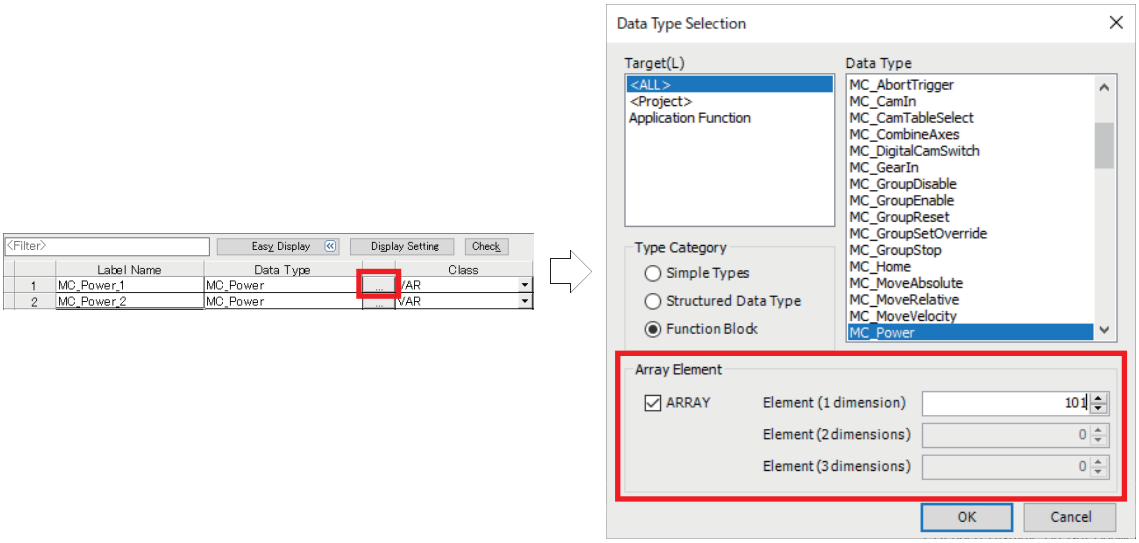
Open the local label editor, add "(1..100)" to the data type of the FB instance "MC_Power_1". Also, change the label name in accordance with the arraying.

- Label Name: MC_Power_Axis1_100
- Data Type: MC_Power (1..100)



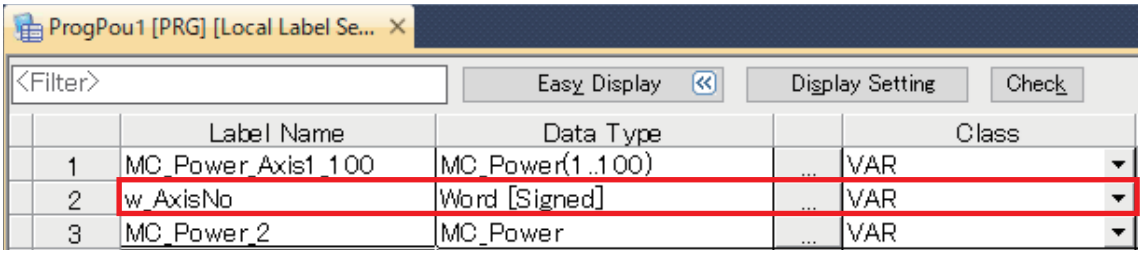
Point

- For the number of array elements, set the number of axes from the start to the end for which the FB will be executed.
- The data type of the label can also be changed in the "Data Type Selection" window displayed by clicking the [...] button on the right of the "Data Type" column. Select "ARRAY" in the "Array Element" section and set the number of arrays.



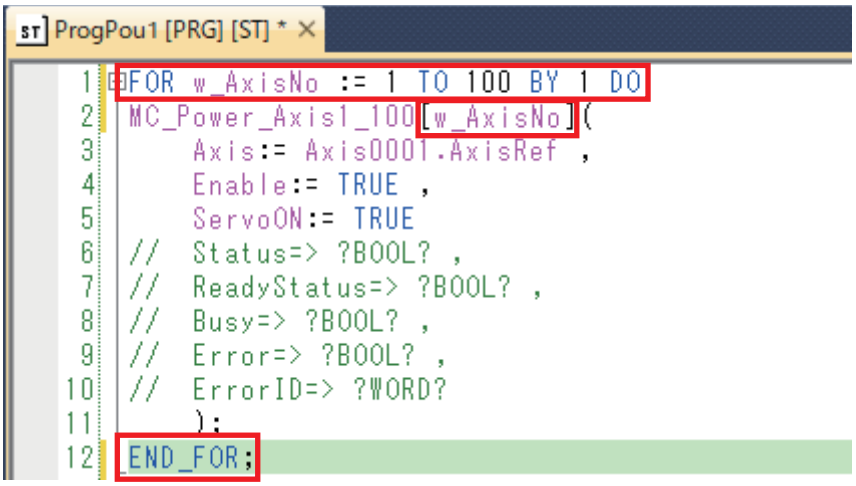
2. Newly create a local label (label for indicating the axis No. for which the motion control FB is executed) to be used for the iteration variable of the FOR statement.

- Label Name: w_AxisNo
- Data Type: Word [Signed]
- Class: VAR



3. Open the program and add the array element "w_AxisNo" to the FB instance "MC_Power_Axis1_100".
In addition, add the following FOR statement, which increments the array element "w_AxisNo" by one within the range of 1 to 100 and executes the FB for axis Nos. 1 to 100.

```
FOR w_AxisNo := 1 TO 100 BY 1 DO
MC_Power_Axis1_100[w_AxisNo](
Execution statement;
);
END_FOR;
```



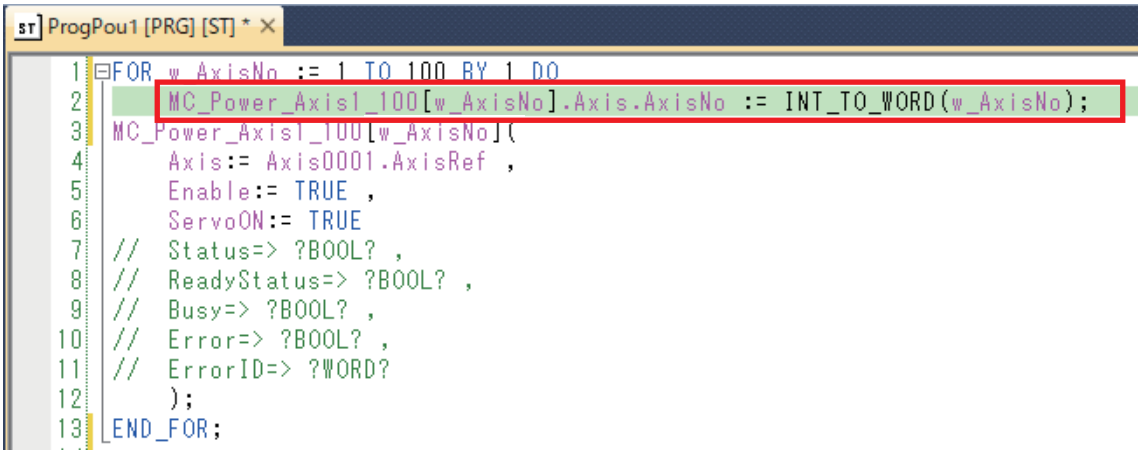
The FB "MC_Power_Axis1_100" is executed repeatedly for axis Nos. 1 to 100.

Point

For details on FOR statements, refer to the following manual.
MELSEC MX Controller (MX-R Model) Programming Manual

4. Using the axis variable "AxisName.AxisRef.AxisNo" (Axis No.), add the following statement, which directly specifies the label "w_AxisNo" (INT type) for the input variable "Axis" of the FB instance "MC_Power_Axis1_100".

```
MC_Power_Axis1_100[w_AxisNo].Axis.AxisNo := INT_TO_WORD(w_AxisNo);
```



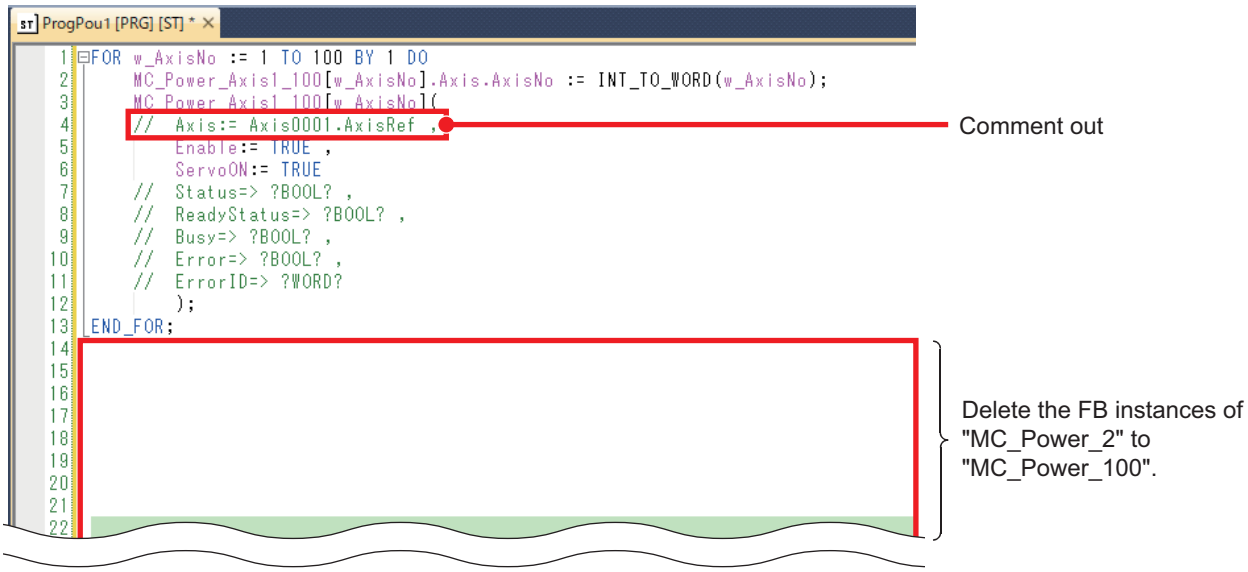
Point

The data type of the axis variable "AxisName.AxisRef.AxisNo" (Axis No.) is WORD (UINT). Specify WORD (UINT) type data for "AxisName.AxisRef.AxisNo".

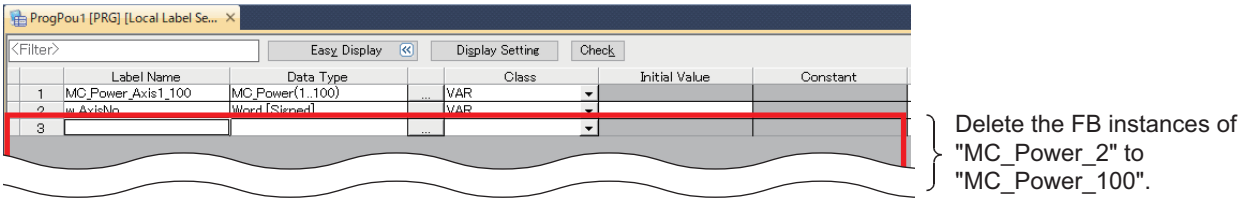
5. Comment out or delete the statement for setting the input variable "Axis" in the FB execution statement (fourth row in the example).

Also, delete unnecessary FB instances ("MC_Power_2" to "MC_Power_100") from the program and local labels.

• ST program



• Local label



Point

When describing FBs, you can omit I/O variables that are not used or not changed from the initial values.

2.3 Grouping Motion Axes

Overview

When multiple axis types (such as the real drive axis, real encoder axis, and virtual drive axis) are used, the axes used in the project can be clarified by grouping them by axis type or equipment or by determining the axis number range and start axis number for each axis type.
Grouping motion axes has the following benefit.

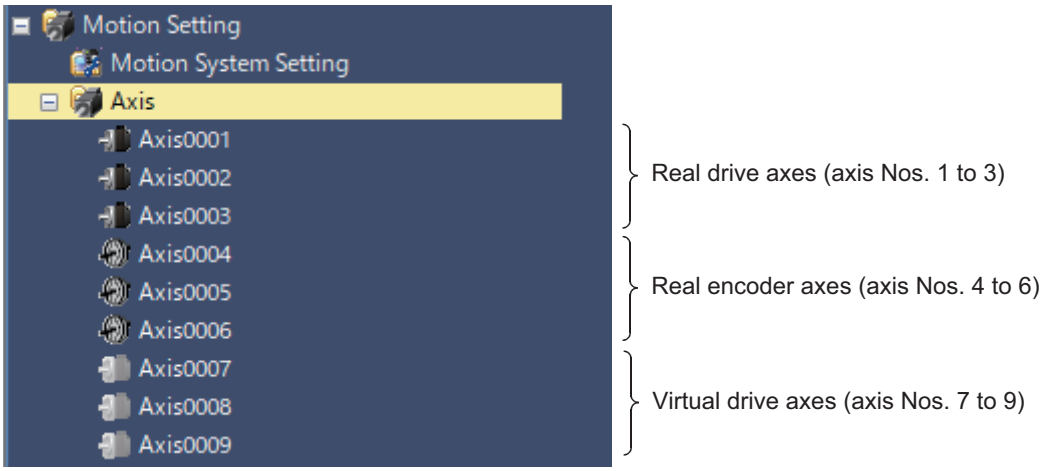
Improvement of project maintainability

Axes can be easily managed by dividing their folders according to the axis type and equipment. In addition, the maintainability is improved by determining the axis number range and start axis number for each axis type because doing so reduces program modifications when any axes are added/deleted due to a facility modification.

Execution procedure

Using the following project and axis number rules as examples, this section describes the procedure for grouping motion axes by axis type.

• Project



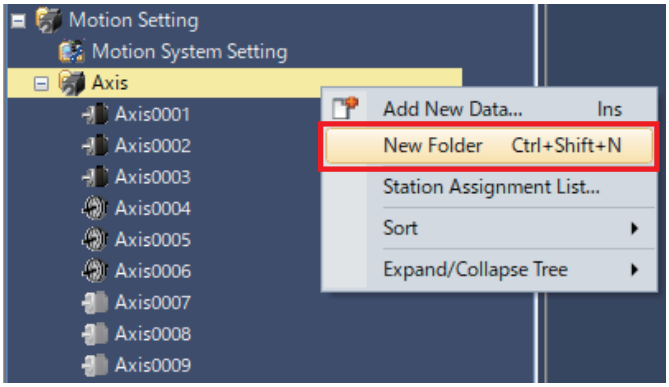
• Axis number rule

Axis type	Axis No.
Real drive axis	1 to 999
Real encoder axis	1000 to 1999
Virtual drive axis	2000 to 2999

Operating procedure

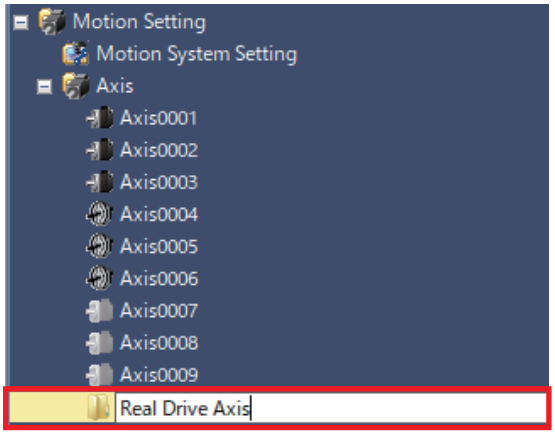
1. Newly create a folder for grouping the axes.

Navigation window ⇨ [Motion Setting] ⇨ Right-click [Axis] ⇨ [New Folder]

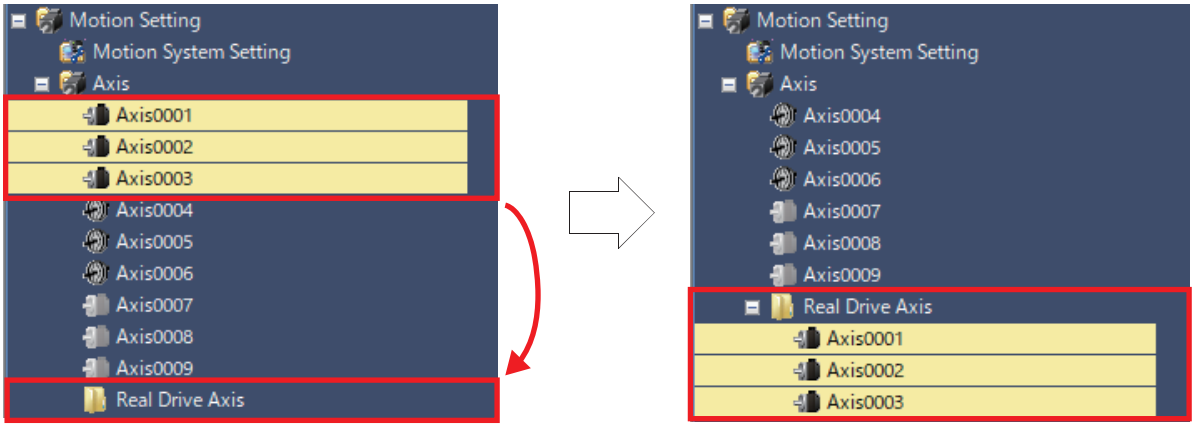


2. Name the folder.

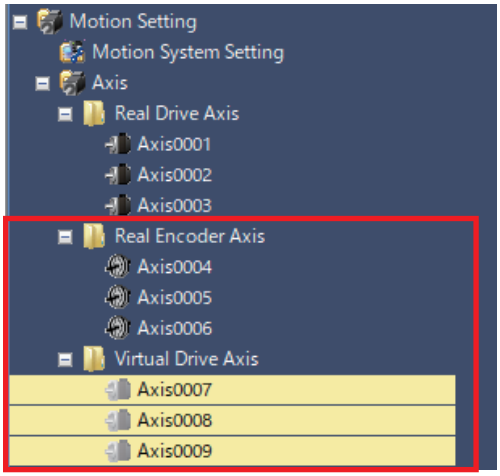
• Folder name: Real Drive Axis



3. Select the axes to be stored ("Axis0001" to "Axis0003") and drag and drop them into the folder created in step 2.

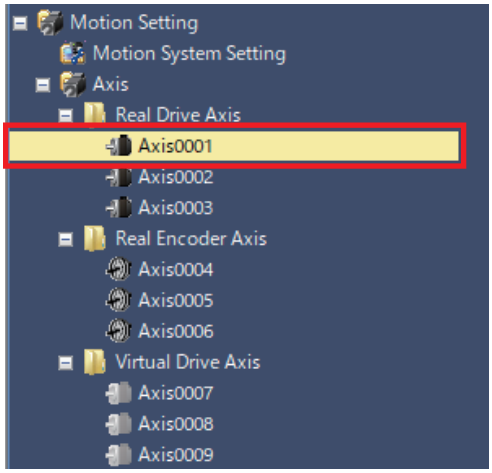


4. Repeat steps 1 to 3, group the real encoder axes and virtual drive axes in the same manner.



5. Open the window for setting axis parameters.

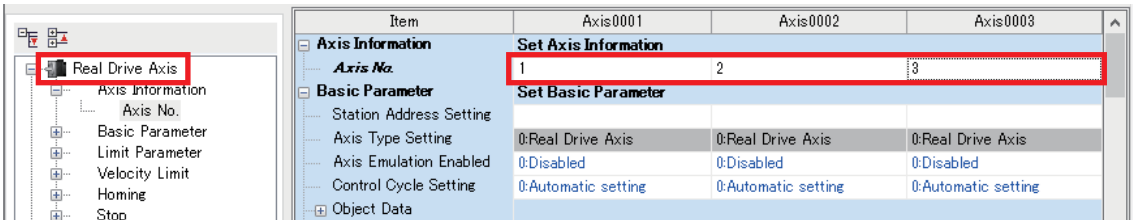
Navigation window ⇨ [Motion Setting] ⇨ [Axis] ⇨ Double-click any axis



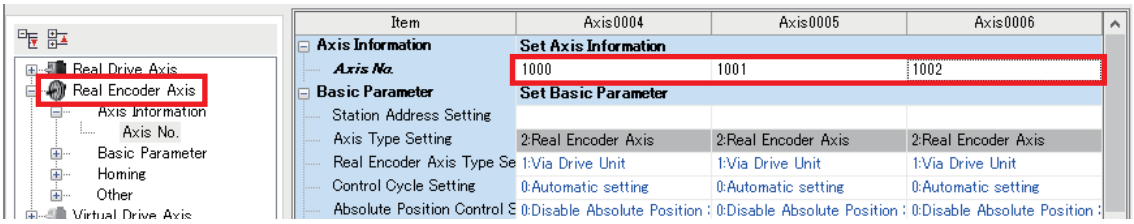
6. Set an axis number for each axis type.

Axis type to be set ⇨ [Axis Information] ⇨ [Axis No.]

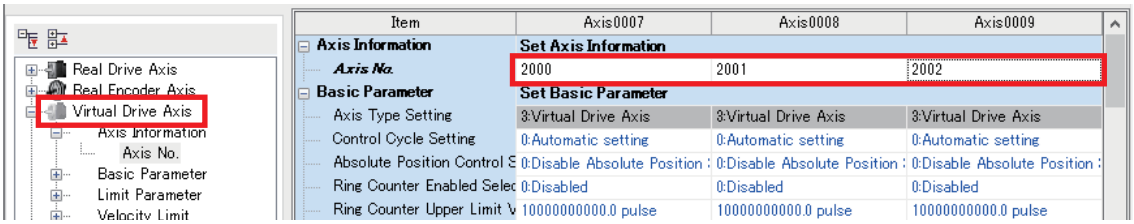
• Real drive axes: Axis Nos. 1 to 3



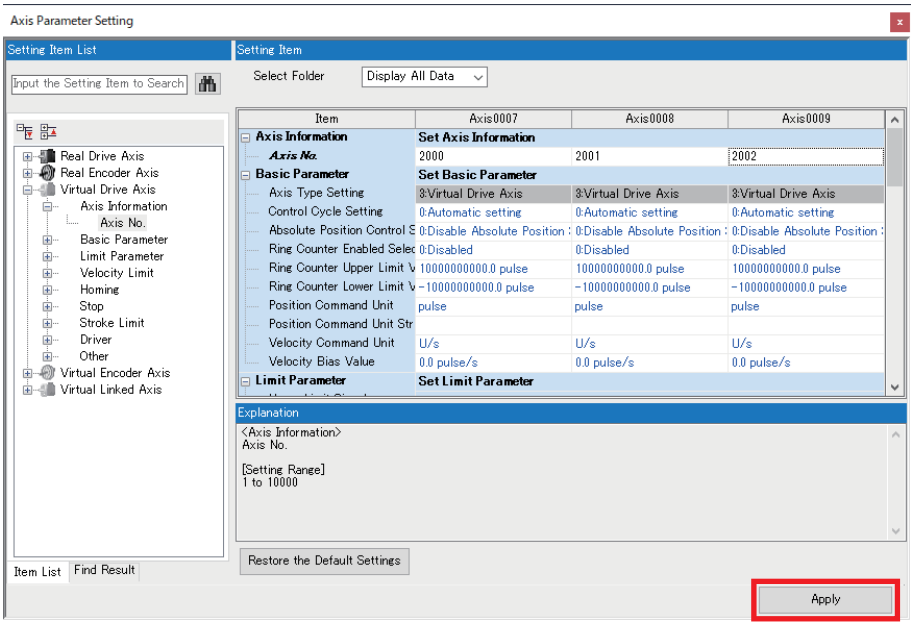
• Real encoder axes: Axis Nos. 1000 to 1002



• Virtual drive axes: Axis Nos. 2000 to 2002



7. Click the [Apply] button to apply the setting.



Precautions

When the station address (IP address) for each axis is not set, a warning occurs when the setting is applied.
When the warning occurs, set the station address in [Station Address Setting] under [Basic Parameter] in the parameter settings for each axis.

2.4 Dividing Global Labels

Overview

By dividing global labels into meaningful units (such as processes and equipment) and moving them to different global label editors, labels for each unit are clarified.
Dividing global labels has the following benefits.

Improvement of global label portability

Dividing global labels allows the labels to be ported on a meaningful unit (such as processes and equipment) basis.

Reduction of program conversion time

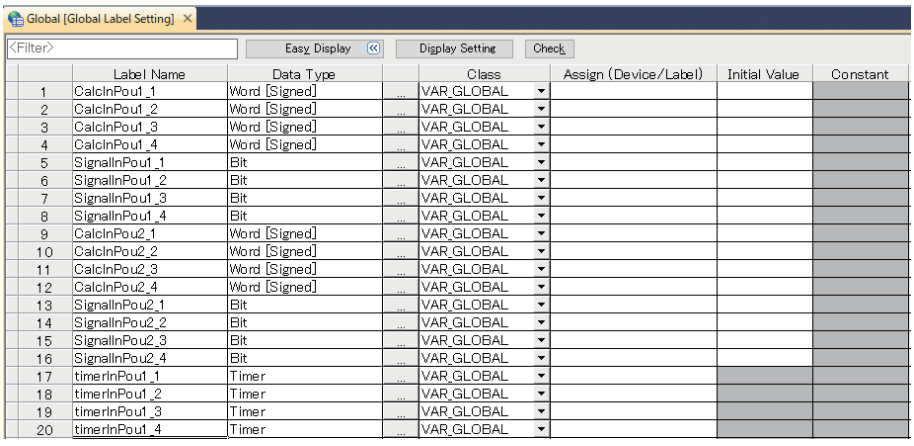
By dividing global labels, the range to be converted in the program can be localized, reducing the effect of the program conversion. This can reduce the program conversion time.

Improvement of operability for editing global labels

Dividing global labels into meaningful units (such as processes and equipment) makes it easier to find and modify the relevant global labels during editing.

Execution procedure

Using the following global labels as examples, this section describes the procedure for dividing them into the global labels used in equipment 1 and those used in equipment 2.

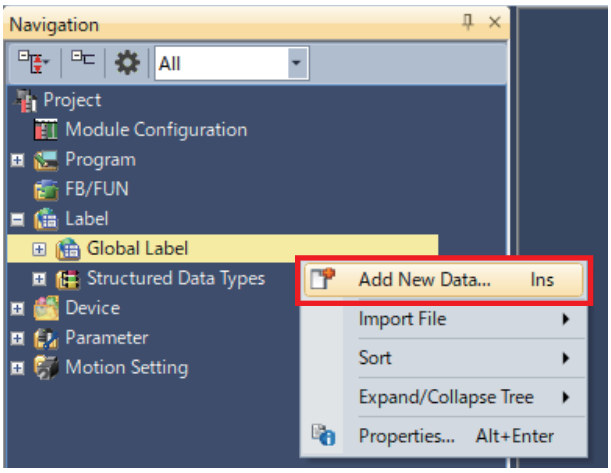


	Label Name	Data Type		Class	Assign (Device/Label)	Initial Value	Constant
1	CalcInPou1_1	Word [Signed]	...	VAR_GLOBAL			
2	CalcInPou1_2	Word [Signed]	...	VAR_GLOBAL			
3	CalcInPou1_3	Word [Signed]	...	VAR_GLOBAL			
4	CalcInPou1_4	Word [Signed]	...	VAR_GLOBAL			
5	SignalInPou1_1	Bit	...	VAR_GLOBAL			
6	SignalInPou1_2	Bit	...	VAR_GLOBAL			
7	SignalInPou1_3	Bit	...	VAR_GLOBAL			
8	SignalInPou1_4	Bit	...	VAR_GLOBAL			
9	CalcInPou2_1	Word [Signed]	...	VAR_GLOBAL			
10	CalcInPou2_2	Word [Signed]	...	VAR_GLOBAL			
11	CalcInPou2_3	Word [Signed]	...	VAR_GLOBAL			
12	CalcInPou2_4	Word [Signed]	...	VAR_GLOBAL			
13	SignalInPou2_1	Bit	...	VAR_GLOBAL			
14	SignalInPou2_2	Bit	...	VAR_GLOBAL			
15	SignalInPou2_3	Bit	...	VAR_GLOBAL			
16	SignalInPou2_4	Bit	...	VAR_GLOBAL			
17	timerInPou1_1	Timer	...	VAR_GLOBAL			
18	timerInPou1_2	Timer	...	VAR_GLOBAL			
19	timerInPou1_3	Timer	...	VAR_GLOBAL			
20	timerInPou1_4	Timer	...	VAR_GLOBAL			

Operating procedure

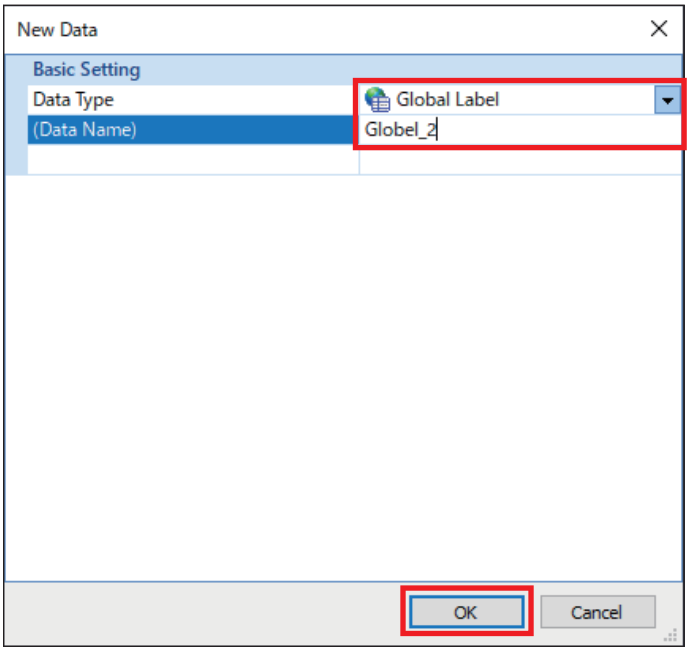
1. Newly create a global label to be the destination of the divided labels.

Navigation window ⇨ [Label] ⇨ Right-click [Global Label] ⇨ [Add New Data]

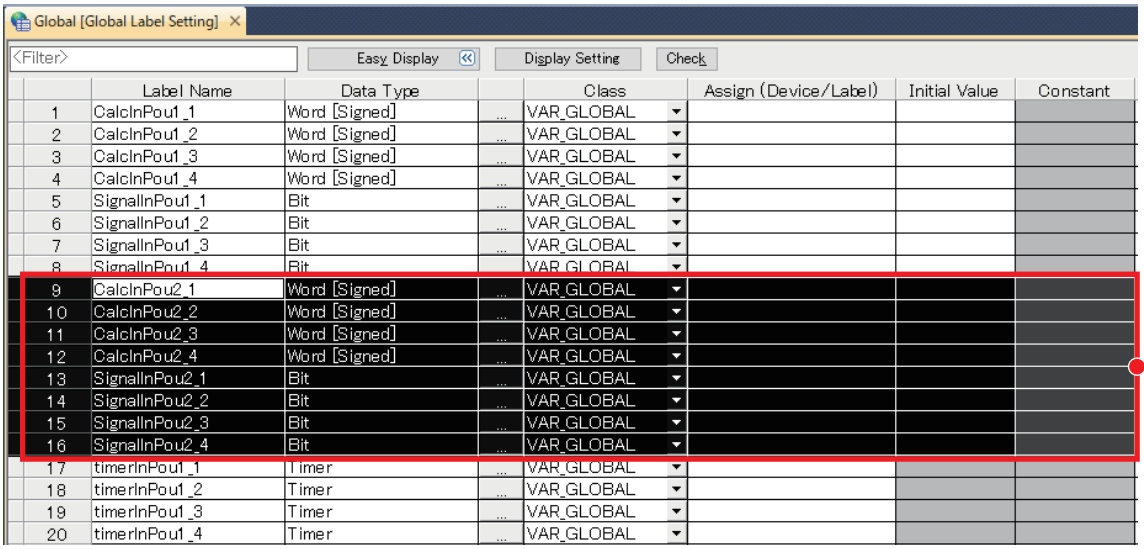


2. Configure the following settings, and then click the [OK] button.

- Data Type: Global Label
- (Data Name): Global_2



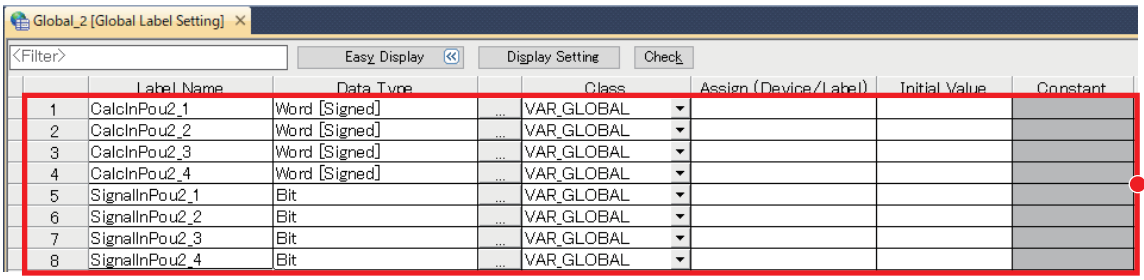
3. Open the global label editor before division, select and cut the global labels to be divided (Nos. 9 to 16 in this example). When selecting the rows to be cut, click the row numbers in the editor so that the entire rows are selected.



Point

- Otherwise, contents hidden in the label editor are not copied.
- It is recommended that you divide the global labels so that the number of labels within a single global label editor does not exceed 5000.

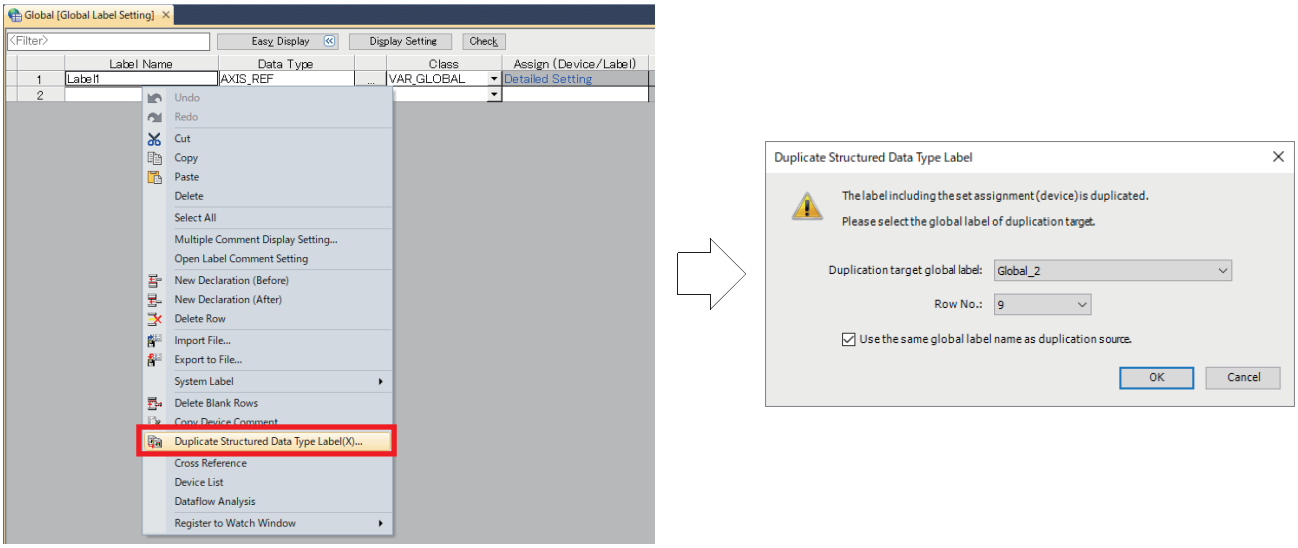
4. Open the global label created in steps 1 and 2 and paste the cut global labels.



Precautions

- If the data type of the global label to be moved to the division destination is a structure, the contents of the assigned device are not copied by this execution procedure.

Right-click the global label and select [Duplicate Structured Data Type Label] to copy the label and divide the global label. (After copying it, delete the global label before the division.)



3 INCREASING SPEED OF PROGRAM CONVERSION

By dividing programs or componentizing them using FBs, the area to be converted in the program can be minimized, increasing the speed of program conversion.
This chapter describes techniques for increasing the speed of program conversion.

List of techniques

The following table lists the techniques described in this chapter.

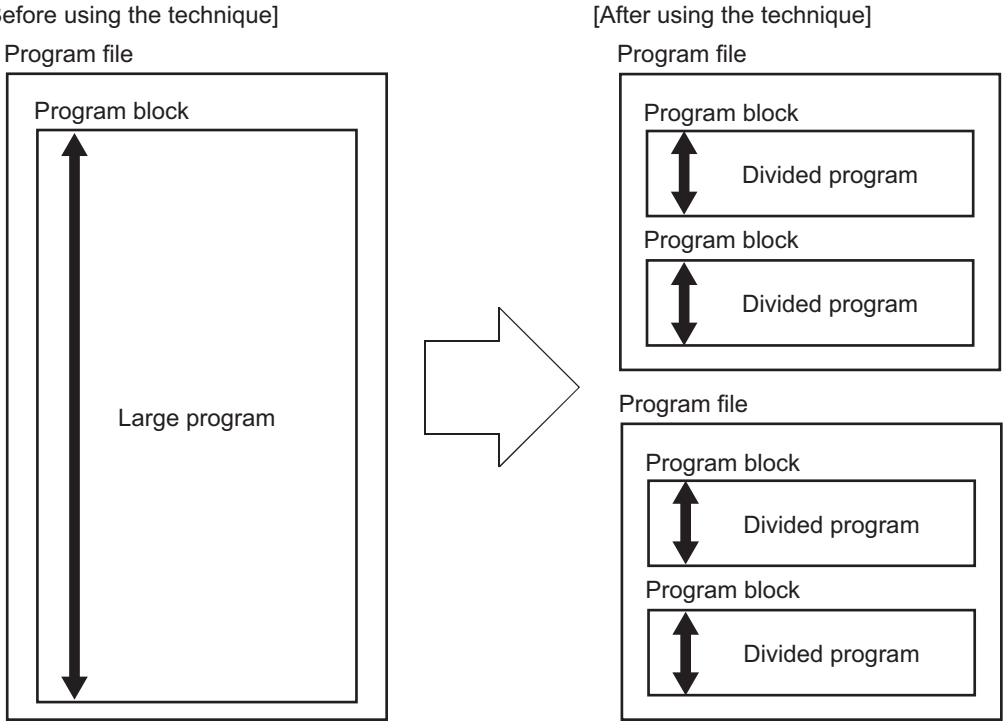
Item	Description	Reference
Dividing program files	Divide program files and program blocks into meaningful units and move them to different program files.	☞ Page 28 Dividing Program Files
Converting programs into FBs	Convert programs separated by pointers or meaningful units, such as similar programs in multiple locations and subroutine programs, into FBs.	☞ Page 37 Converting Programs into FBs
Shortening large-scale programs	Replace large programs in subroutines and control syntax with FBs. Or, move the programs outside syntax.	☞ Page 42 Shortening Large-Scale Programs
Changing global label (structure) settings	Delete unnecessary device assignment from structures. Or, modify the program so that only the necessary structure members are specified instead of passing structures to which devices are assigned directly to arguments of FBs.	☞ Page 45 Changing Global Label (Structure) Settings
Reducing consecutive references to the same array element	If the same array element is accessed multiple times, modify it so that it is accessed only once.	☞ Page 50 Reducing Consecutive References to the Same Array Element

3.1 Dividing Program Files

Overview

Structure and standardize the program by dividing program files and program blocks into meaningful units (such as equipment, processes, and purposes). This can localize the range to be converted in the program, reducing the program conversion time.

By setting the execution order of the divided program files, the programs can be operated in the same order as before the division.



Dividing program files has the following benefits.

Improvement of program readability

Processing of the equipment and processing for each process are clarified, making it easier to grasp the flow and structure of the program processing.

Improvement of program portability and reusability

Program files having program bodies and local labels for each equipment processing or process unit can be created, making it easier to port and reuse programs on a processing or process basis. This improves the program development efficiency.

Correlation between the number of program steps and conversion speed

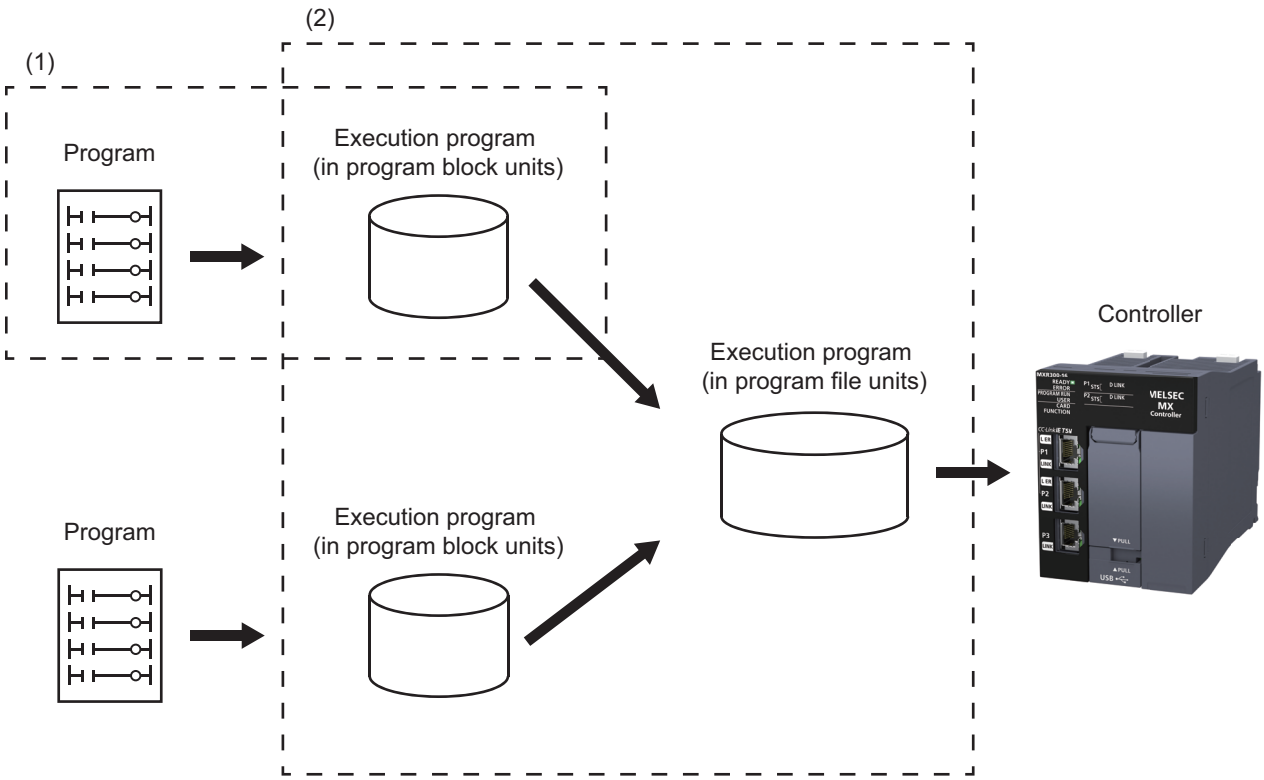
The following table shows the correlation between the number of program steps and conversion speed.
Refer to the table when dividing a program file or program block.

Number of program steps*1*2	Conversion speed (second)*3*4		
	No division (one program block in one program file)	Four program blocks in one program file	Two program blocks each in two program files
1000	4.2	4.2	4.2
10000	11.2	10.2	9.2
100000	125.2	83.1	79.7

*1 These are measured values with a simple program containing only contacts and coils for the number of steps.
*2 The conversion speed fluctuates depending on various factors such as project structure and program content. In particular, note that programs using pointers may require more time for conversion.
*3 These are measured values obtained on a personal computer equipped with an Intel® Core™ i7-10610U (1.80 GHz).
*4 These are conversion speeds when "Create Execution Programs after Conversion" is selected in the dialog displayed at the time of full conversion execution.

Relationship between the program conversion flow and this technique

The following shows the relationship between the program conversion flow and this technique.



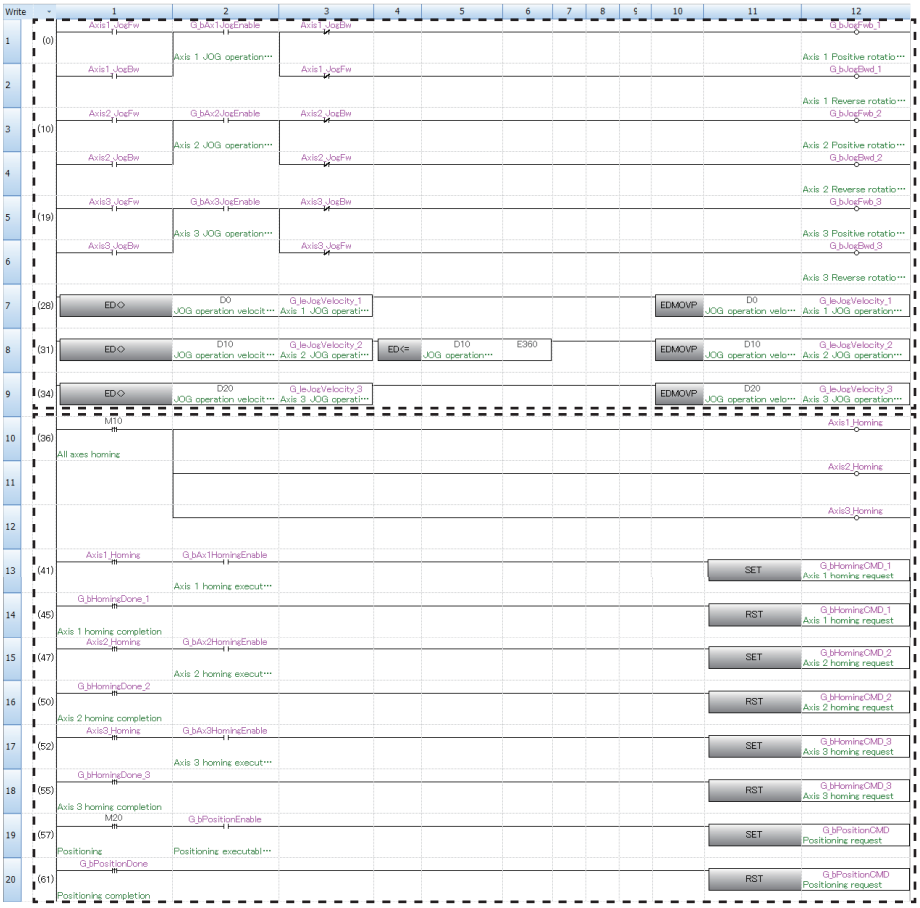
(1) Process for generating execution programs in program block units from the program described
(2) Process for combining execution programs in program block units into an execution program in a program file unit

When a program block is divided into multiple program blocks, the generation of execution programs in process (1) is parallelized, reducing the time for "Rebuild All", "Convert", and "Online Program Change".
Additionally, when a program file is divided into multiple program files, the size of the files written to the programmable controller in process (2) becomes smaller, reducing the time for "Convert" and "Online Program Change", which convert only the changed parts.

Execution procedure

Using the following program as an example, this section describes the procedure for dividing program files for each process.

- Program



- Local label

Motion [PRG] [Local Label Setti...

<Filter>

Easy Display | Display Setting | Check

	Label Name	Data Type		Class	Initial Value	Constant
1	Axis1_JogFw	Bit	...	VAR		
2	Axis1_JogBw	Bit	...	VAR		
3	Axis2_JogFw	Bit	...	VAR		
4	Axis2_JogBw	Bit	...	VAR		
5	Axis3_JogFw	Bit	...	VAR		
6	Axis3_JogBw	Bit	...	VAR		
7	Axis1_Homing	Bit	...	VAR		
8	Axis2_Homing	Bit	...	VAR		
9	Axis3_Homing	Bit	...	VAR		

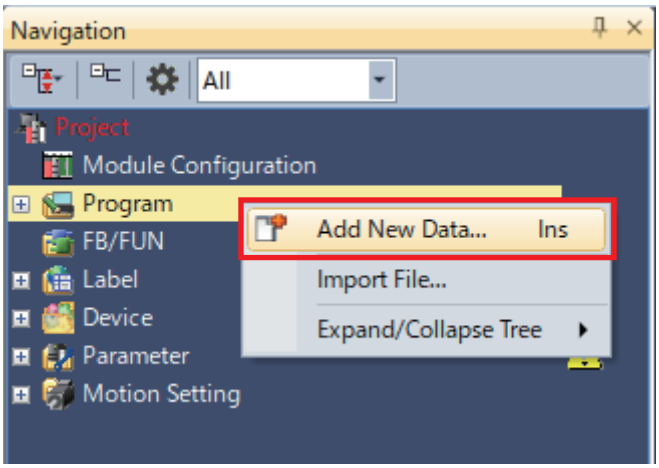
Used for process 1

Used for process 2

Operating procedure

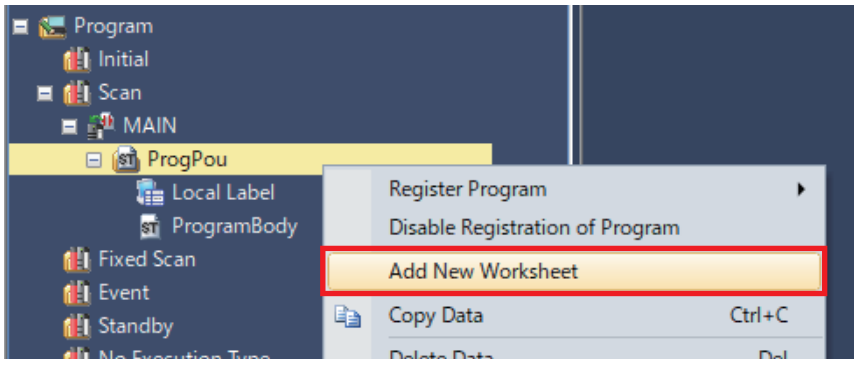
1. Newly create a program file to be the destination of the divided programs.

Navigation window ⇨ Right-click [Program] ⇨ [Add New Data]



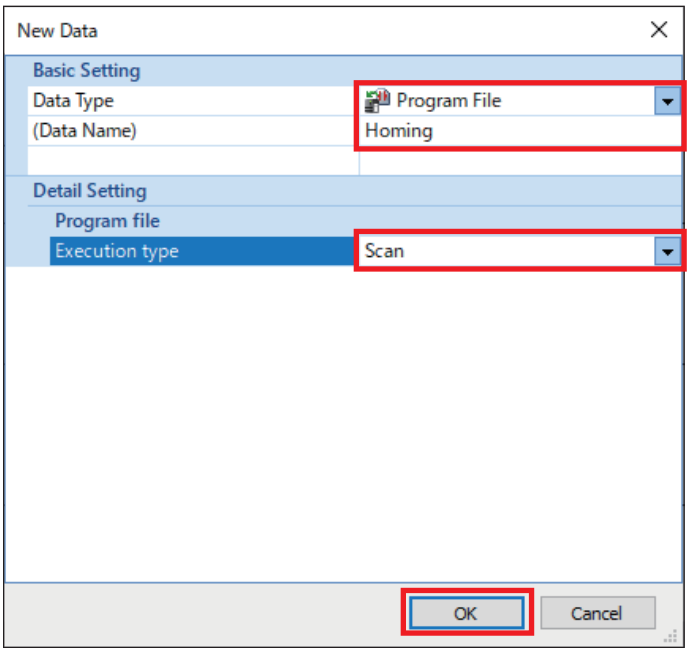
Point

- The smaller the amount of code per program file is, the shorter the program conversion time is. Therefore, it is recommended that the divided programs be moved to a new program file. (There will be no problem in operation if they are moved to an existing program file.)
- ST programs and FBD programs can be divided by adding a new worksheet in the same POU. This improves the program readability and portability. (The program conversion time is not reduced.)

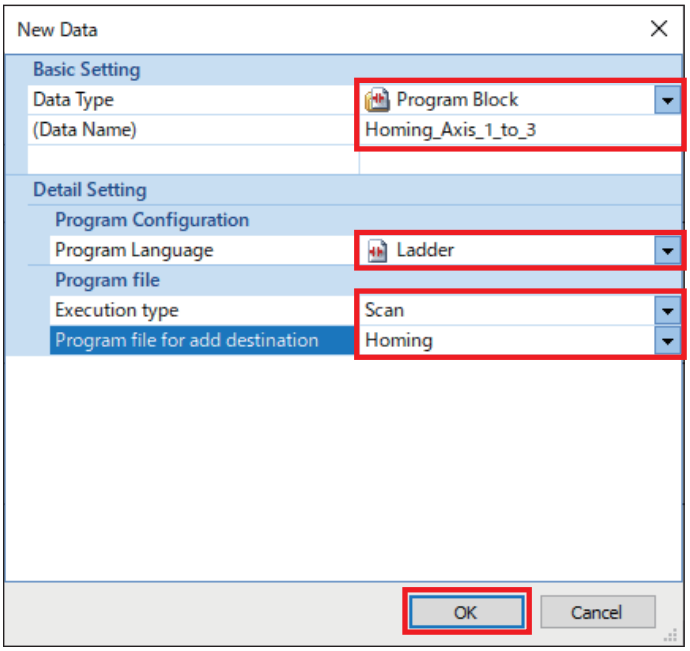


2. Configure the following settings, and then click the [OK] button.

- Data Type: Program File
- (Data Name): Homing
- Execution type: Scan

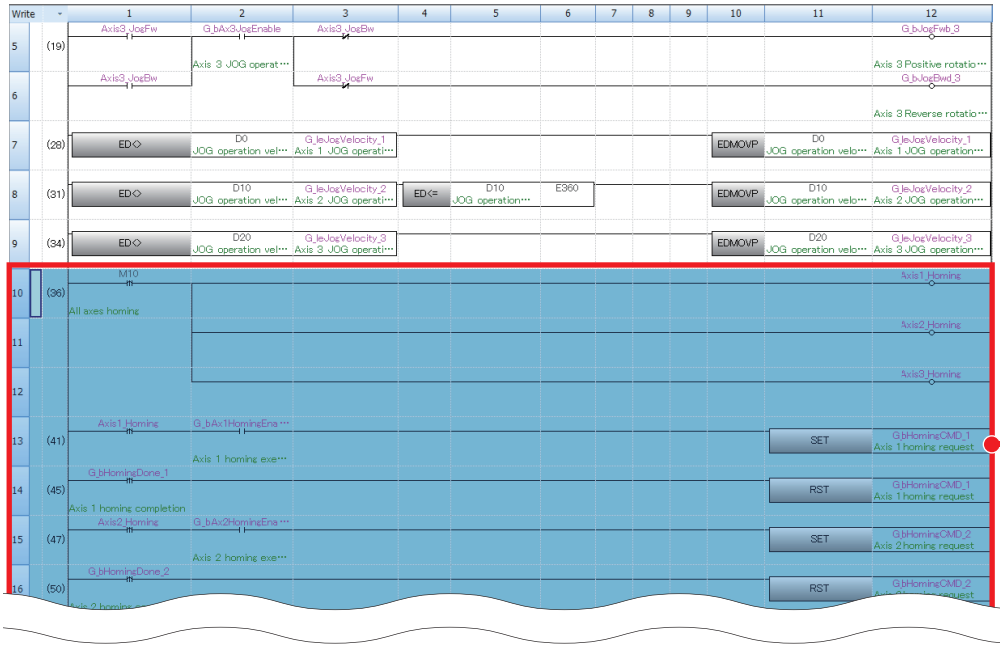


3. Open the "New Data" window in the same manner as step 1 and create a new program block.
Configure the following settings, and then click the [OK] button.



Item	Setting	Remarks
Data Type	Program Block	—
(Data Name)	Homing_Axis_1_to_3	—
Program Language	Ladder	—
Execution type	Scan	Set the same execution type as that of the program file created in step 2.
Program file for add destination	Homing	Set the program file created in step 2.

4. Open the program body of the program file to be divided, determine the position from which the file is divided (start of process 2 in this example), select and cut all the programs after the position.



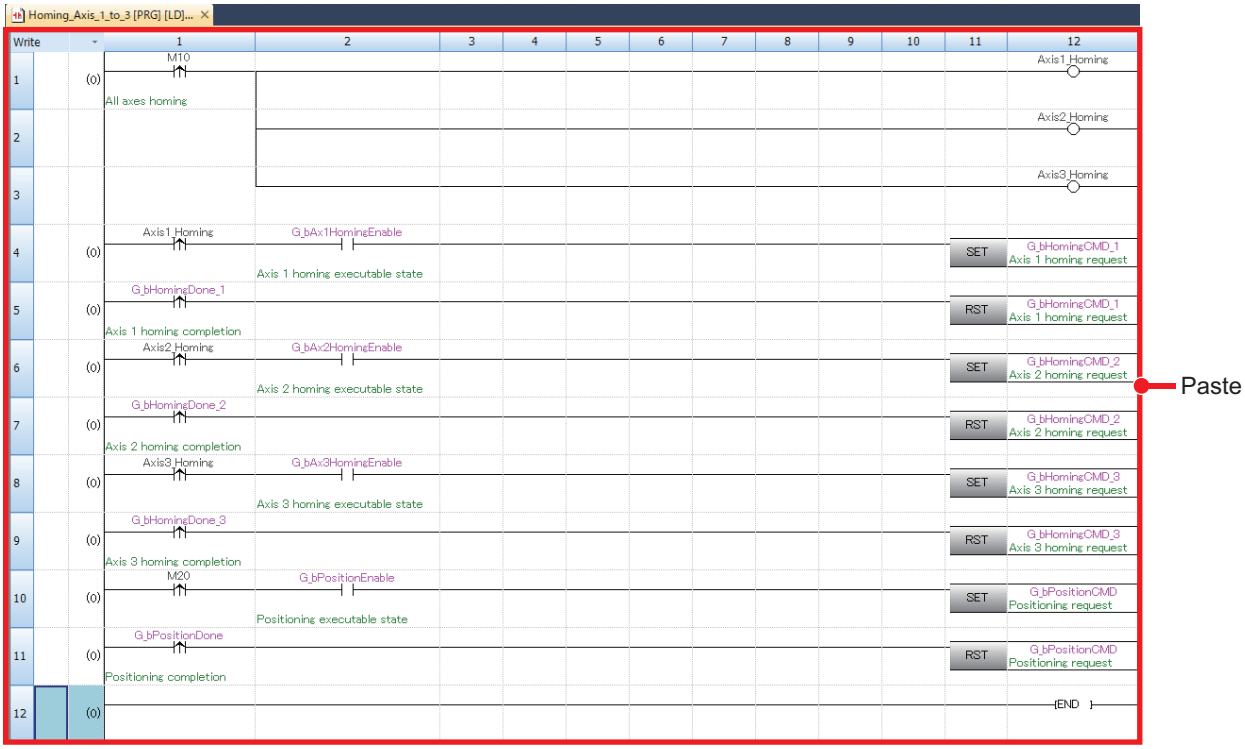
Point

Divide the program so that the local labels and local devices used in the program before the division are used only in either one of the programs after the division. When dividing the file in this manner is difficult, it is recommended that the program be converted into FBs.

For converting programs into FBs, refer to the following.

☞ Page 37 Converting Programs into FBs

5. Open the program body of the program block created in step 3 and paste the cut program.

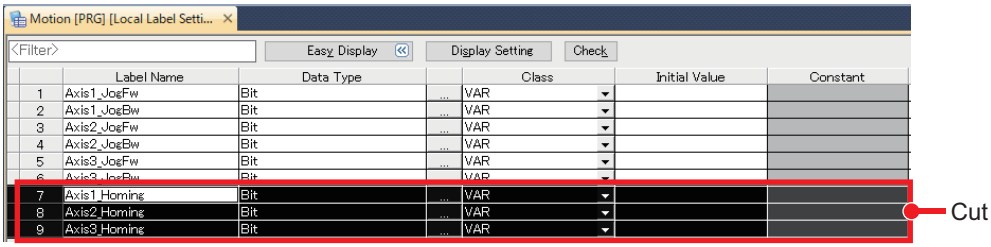


Point

If there are any program blocks whose execution orders are after the divided program block, move them together to the new program file.

6. Open the local label editor of the division source program file, and cut the local labels used in the division destination program (No. 7 to 9 in the example).

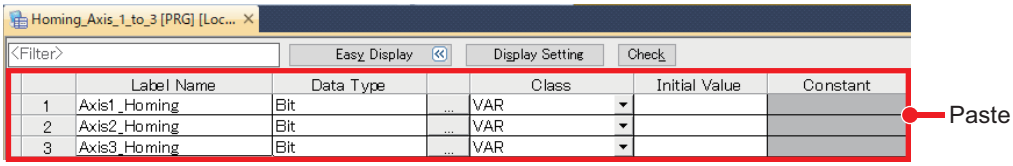
When selecting the rows to be cut, click the row numbers in the editor so that the entire rows are selected.



Point

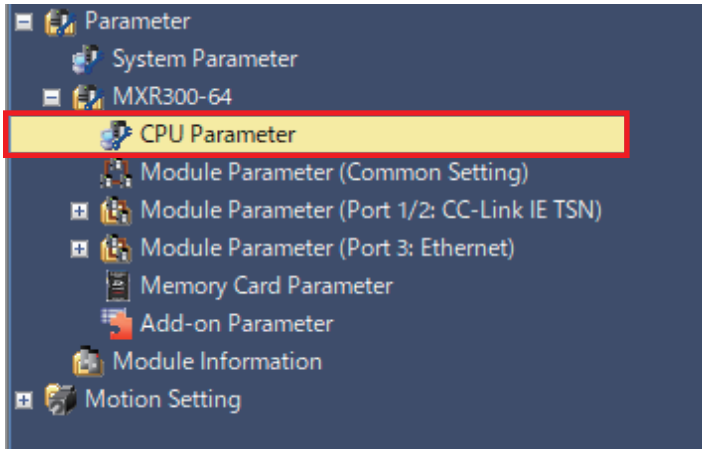
Otherwise, contents hidden in the label editor are not copied.

7. Open the local label editor of the program block created in step 3, paste the cut local labels.



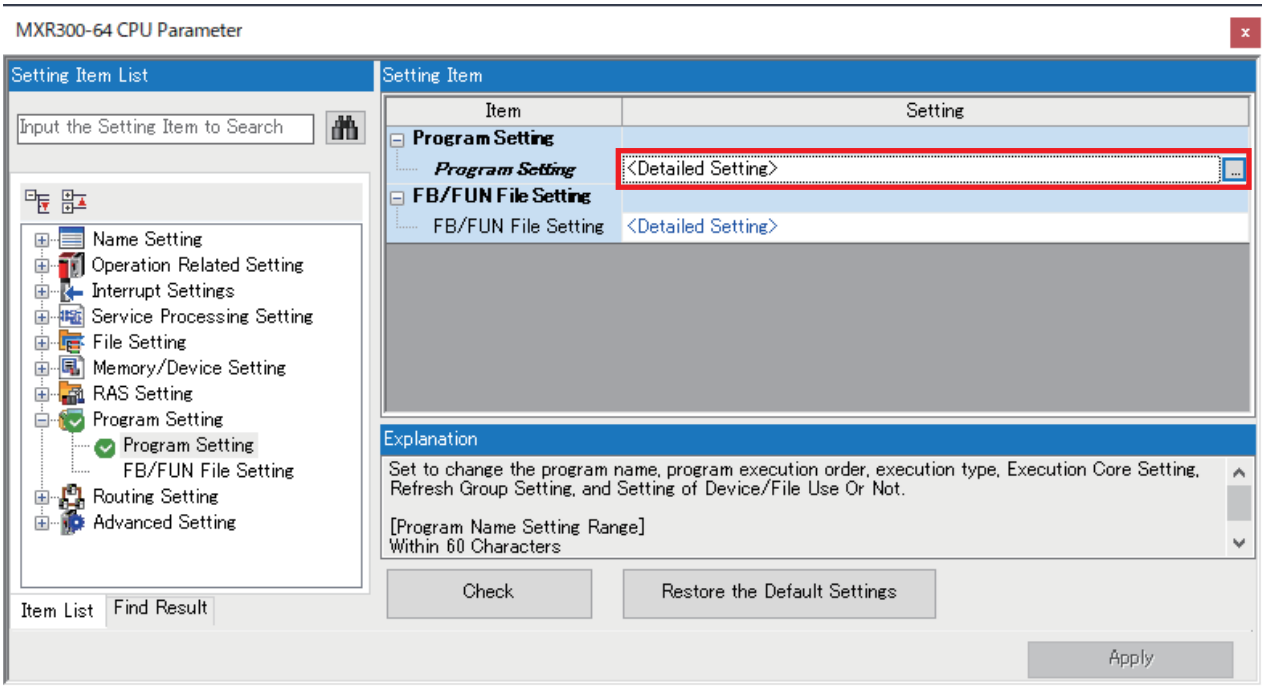
8. Open CPU parameters.

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ Double-click [CPU Parameters]

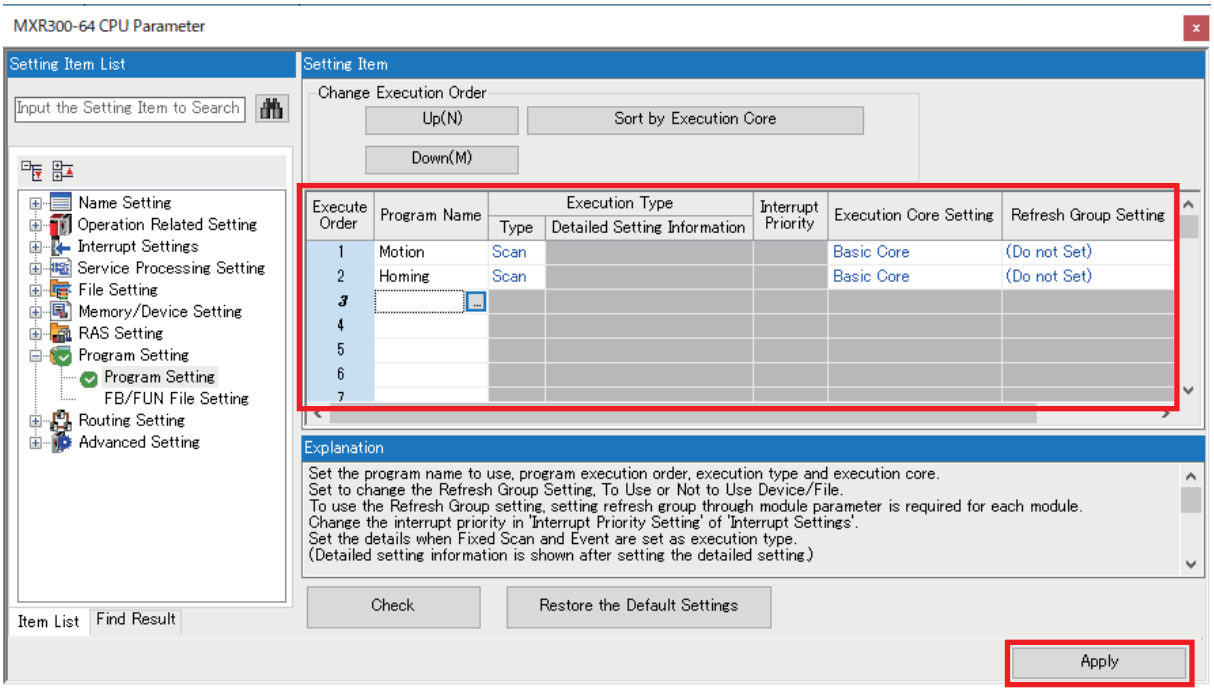


9. To set the program execution order, open the program setting.

[Program Setting] ⇨ Double-click <Detailed Setting> of [Program Setting]

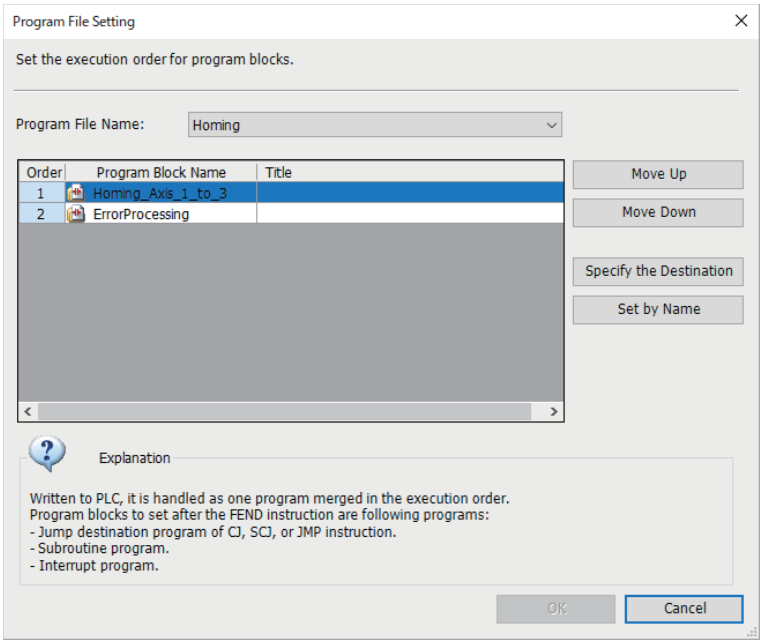


10. Sort the programs in the execution order, and then click the [Apply] button to apply the setting.



Point

The execution order of program blocks can be changed in the "Program File Setting" window.
For details, refer to the following manual.
GX Works3 Operating Manual



Precautions

If dividing program files separates the jump source and destination of an instruction using a pointer in different program files or it makes the elements that make up subroutines, control syntax, etc. no longer exist in the same program, a conversion error occurs.

3.2 Converting Programs into FBs

Overview

By converting programs separated by meaningful units, such as similar programs in multiple locations and subroutine programs separated by pointers, into FBs, the range to be converted in the program can be localized, reducing the effect of the program conversion.
In programs using pointers, replacing the pointers with FBs can structure the range to be converted in the program, reducing the program conversion time.
In ladder programs, FBs can be created using existing circuits. By calling the created FBs from the positions where the programs before the conversion into FBs were originally located and giving necessary devices and labels as arguments, the same program operation as the one before the conversion into FBs can be implemented.
Converting programs into FBs has the following benefits.

Improvement of program readability

By converting a program into FBs, it becomes a simple program consisting only of "boxes" (FBs), inputs, and outputs only, allowing you to create a program that is easy to read.

Improvement of program portability and reusability

By commonly used programs into FBs, the need to copy programs and modify devices at the time of porting programs is eliminated. In addition, saving programs converted into FBs as libraries allows you to use common libraries in various projects. This improves the program reusability.

Improvement of program quality

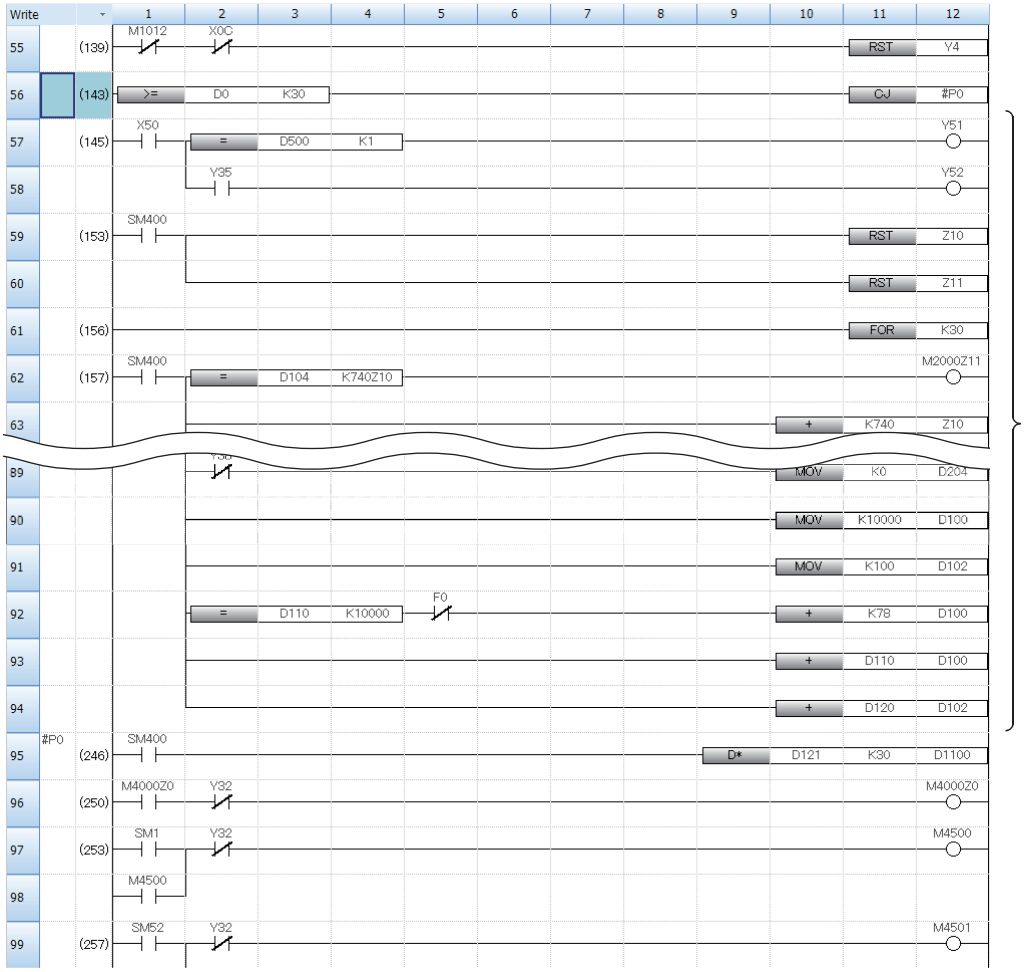
By componentizing programs as FBs and reusing them, the same FB can be used for common processing even if the developers are different. This allows equalization of program quality independent of program developers' skill levels.

Point

In programs other than ladder programs (ST and FBD/LD), FBs can be created by cutting the program to be converted into an FB and local labels and pasting them into a new FB.
For newly creating FBs, refer to the following manual.
GX Works3 Operating Manual

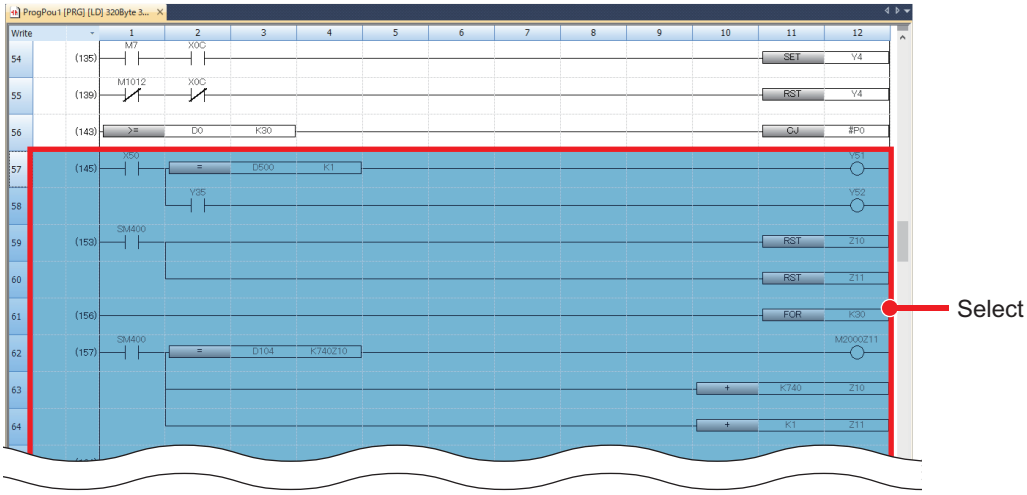
Execution procedure

Using the following ladder program as an example, this section describes the procedure for converting a program separated by pointers into an FB.



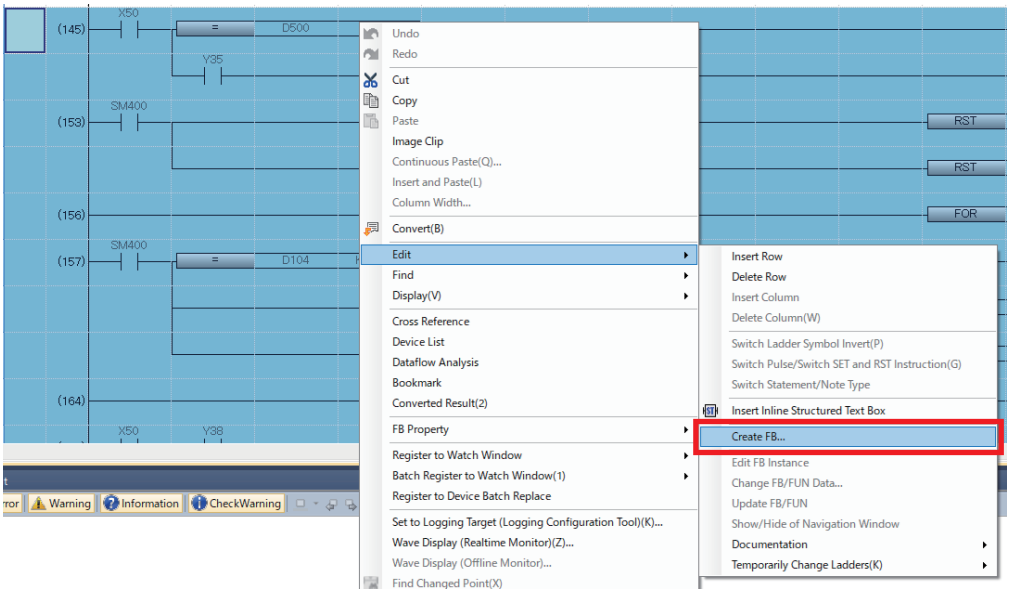
Operating procedure

1. Open the program to be converted to an FB and select the range to convert.



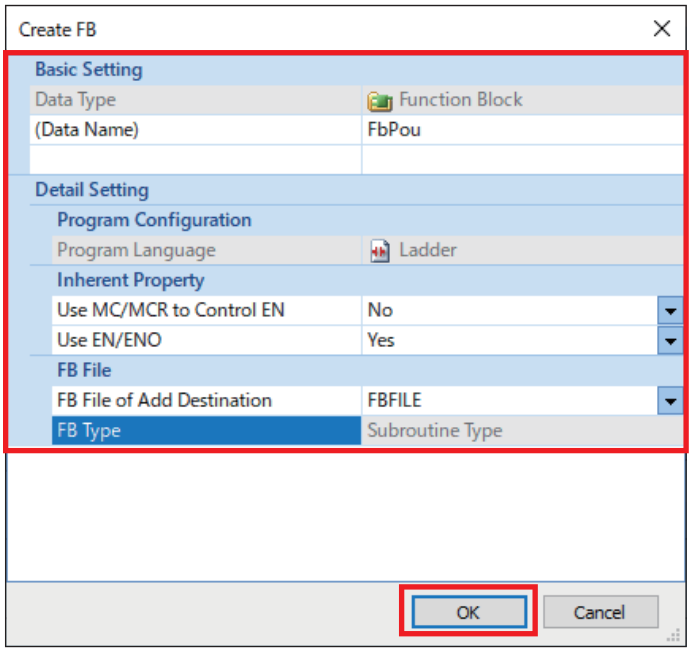
2. Create an FB from the selected range.

Right-click ⇨ [Edit] ⇨ [Create FB]

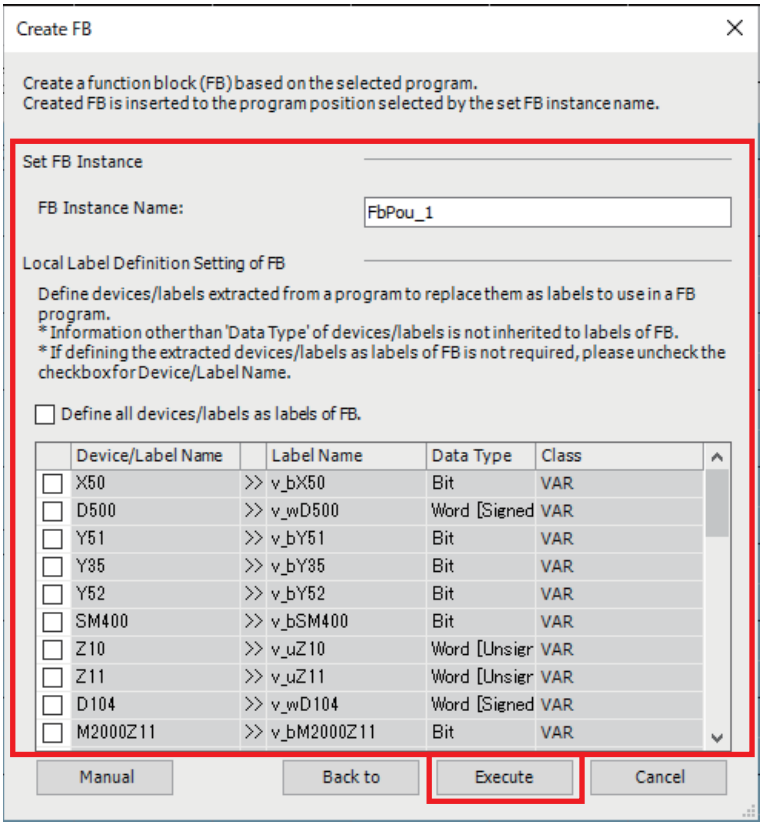


3. Configure the following settings for the FB to be created according to the program contents, and then click the [OK] button.

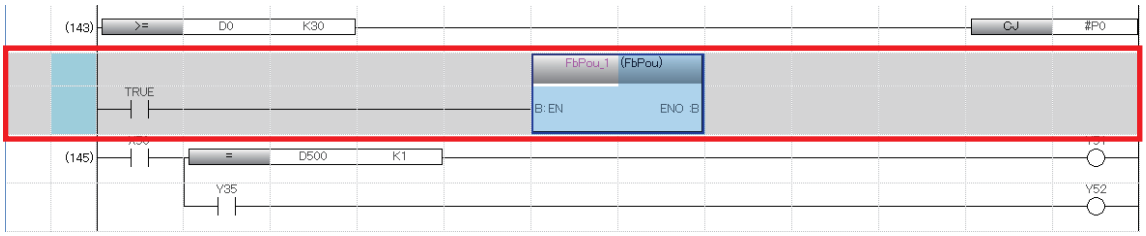
In this example, "Use EN/ENO" is set to "Yes".



4. Set the instance name and local label definition of the FB, and click the [Execute] button.
In the example, the "Define all devices/labels as labels of FB." checkbox is deselected.



The created FB is inserted into the start position of the FB conversion range.



Precautions

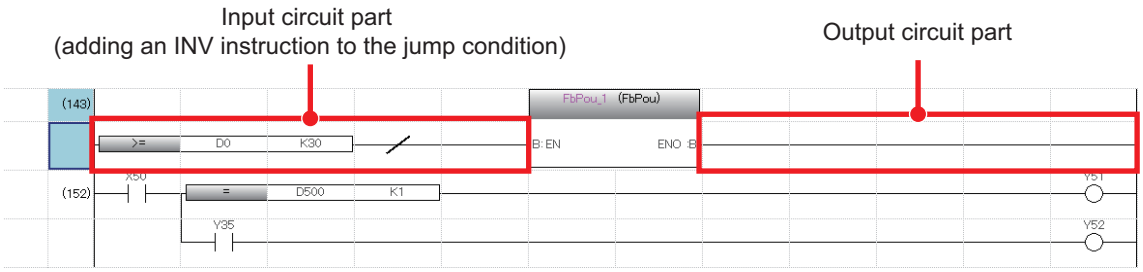
When setting the local label definition of the FB, note the following points.

- Local devices cannot be set as FB arguments. In "Local Label Definition Setting of FB", clear the checkboxes of the corresponding device/label names.
- When the same local device/local label is used both inside and outside the FB program, define the local device/local label as the label (argument) of the FB.

For details on the local label definition setting of FBs, refer to the following manual.
📖GX Works3 Operating Manual

5. Create input and output circuit parts according to the input and output labels of the FB so that the program is equal to the one before the conversion to the FB.

In this example, an INV instruction is added to the jump condition of the pointer to modify it to be an FB execution condition.



Precautions

- If converting a program into an FB makes the jump source and destination of an instruction using a pointer no longer exist in the same program or it makes the elements that make up subroutines, control syntax, etc. no longer exist in the same program, a conversion error occurs.
- The larger the amount of data passed as FB arguments is, the longer the scan time may be.

3.3 Shortening Large-Scale Programs

Overview

If there is a large program within a control syntax such as the FOR to NEXT instructions or subroutines in a ladder program, or the IF and FOR statements in an ST program, the program can be shortened by converting the program into an FB or by moving programs that do not need to be written within the control syntax to the outside of the syntax. This can reduce the amount of code, reducing the program conversion time.

The following table lists the control syntax that can be shortened for each program type.

Type	Control syntax
Ladder program	• FOR to NEXT instructions • Circuit block groups separated by pointers such as subroutine programs
ST program	• IF statements • CASE statements • FOR statements • WHILE statements • REPEAT statements
FBD/LD program	• FOR to NEXT instructions • Circuit groups separated by jump labels

Shortening programs has the following benefit.

Improvement of program readability

The amount of code is reduced, improving the program readability.

Point

For the overview and benefits of converting programs into FBs, refer to the following.

☞ Page 37 Overview

Correlation between the number of program steps in control syntax and conversion speed

The following table shows the correlation between the number of program steps in control syntax and conversion speed. Refer to the table when moving programs to the outside of control syntax.

Number of program steps*1*2	Conversion speed (second)*3*4	
	Program in control syntax*5	Program outside control syntax
1000	5.6	4.2
10000	19.7	11.2
100000	447.5	125.2

*1 These are measured values with a simple program containing only contacts and coils for the number of steps.

*2 The conversion speed fluctuates depending on various factors such as project structure and program content. In particular, note that programs using pointers may require more time for conversion.

*3 These are measured values obtained on a personal computer equipped with an Intel® Core™ i7-10610U (1.80 GHz).

*4 These are conversion speeds when "Create Execution Programs after Conversion" is selected in the dialog displayed at the time of full conversion execution.

*5 These are measured values when the program is described within FOR to NEXT instructions.

Execution procedure

This section describes the procedure for shortening programs.

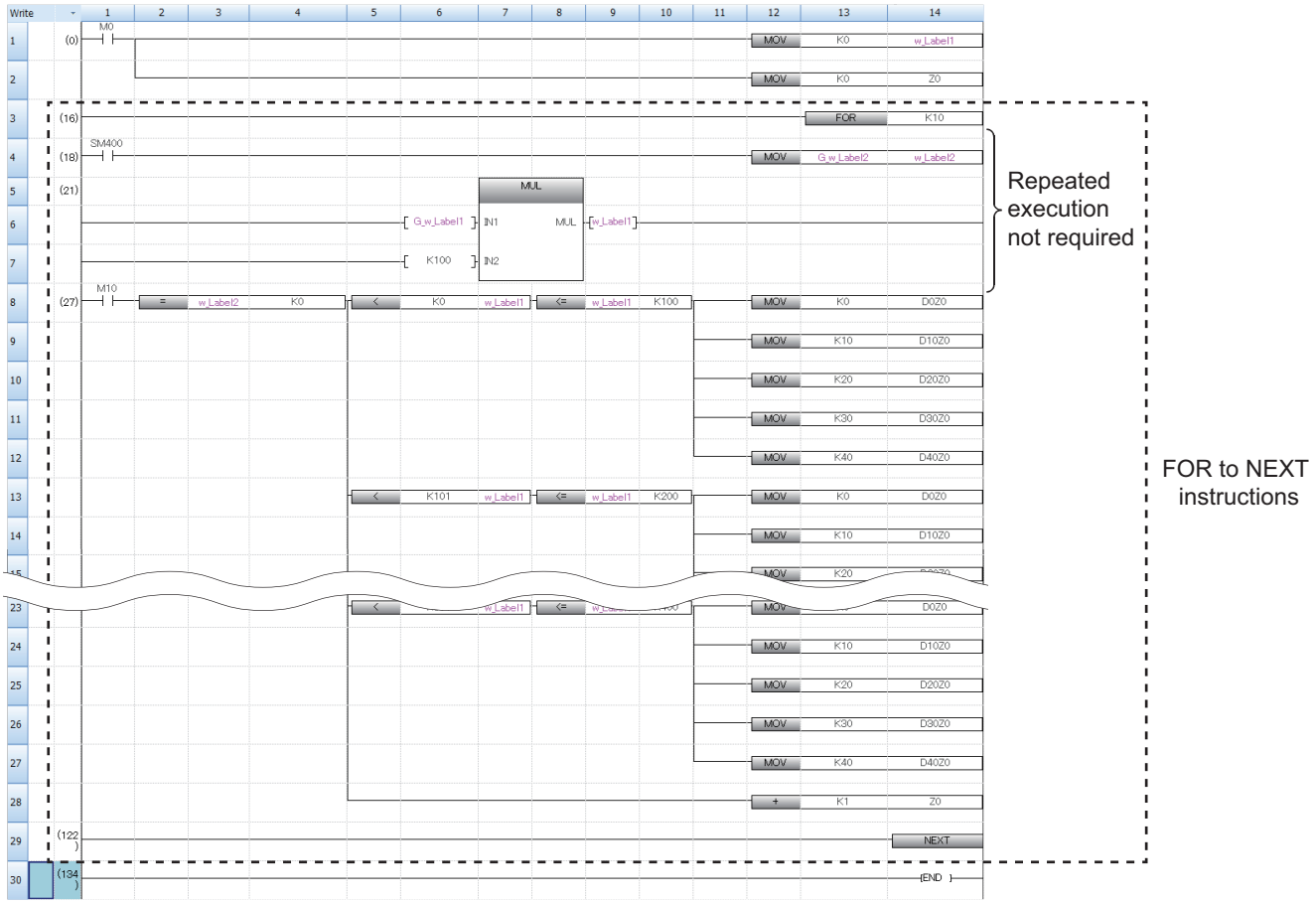
Converting programs in control syntax into FBs

For converting programs in control syntax into FBs, refer to the following procedure.

☞ Page 38 Execution procedure

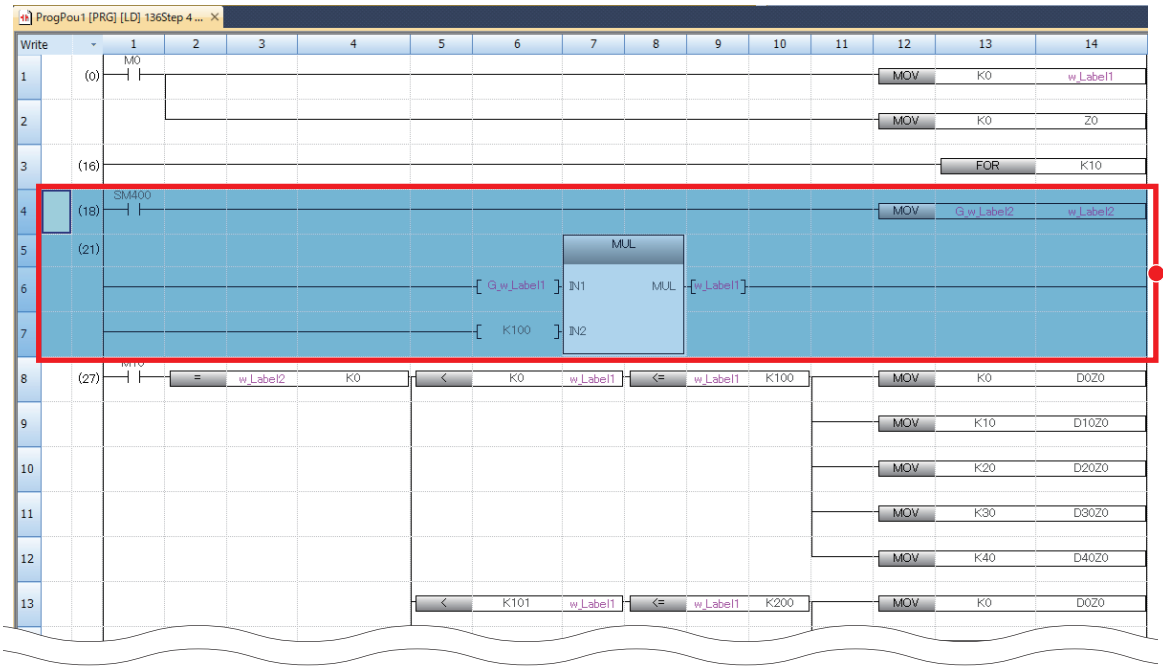
Moving unnecessary programs in control syntax

Using the following program as an example, this section describes the procedure for moving programs not required to be repeatedly executed to the outside of the FOR to NEXT instructions.

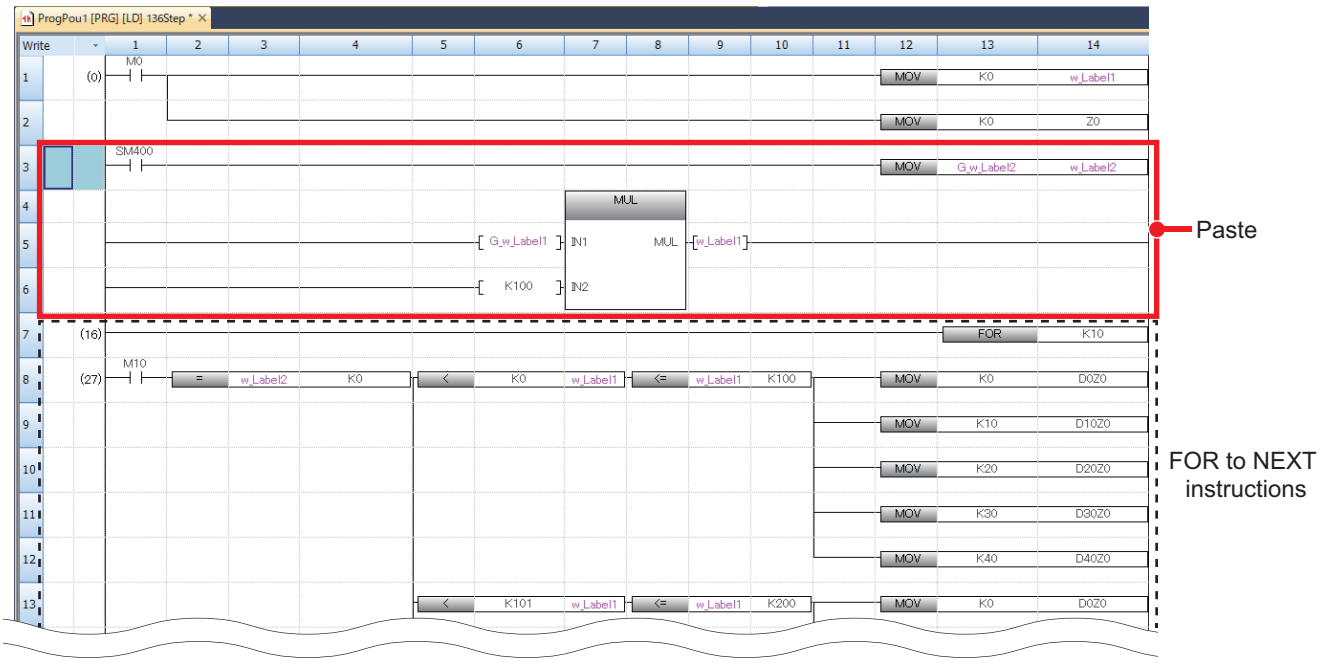


Operating procedure

1. Open the program and select and cut the programs not required to be repeatedly executed in the FOR to NEXT instructions.



2. Paste the cut programs to the outside of the control syntax. Paste them in the correct position according to the execution order.



3.4 Changing Global Label (Structure) Settings

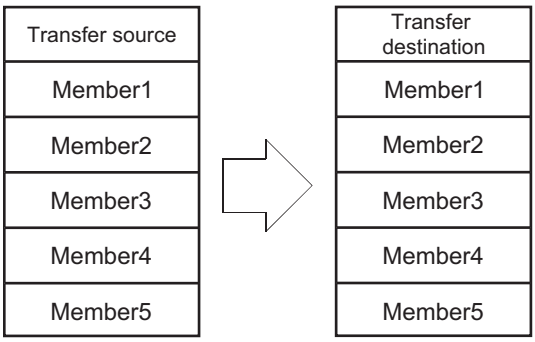
Overview

When structures to which devices are assigned are used as the following arguments and assignment statements, values are transferred for each member of the structure. This generates many transfer instructions internally.

- Arguments of subroutine type FBs
- Arguments of MOVE instructions
- Assignment statements of ST programs and FBD programs

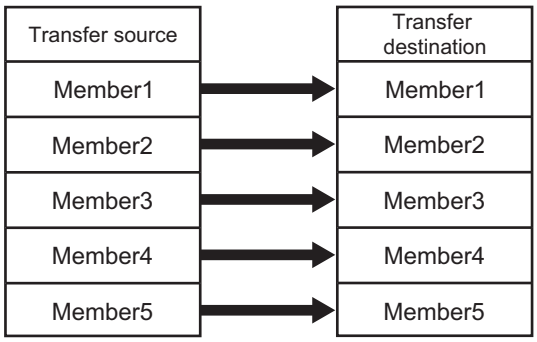
The following shows a transfer example where a structure without devices assigned is used as a MOVE instruction argument and another transfer example where a structure with devices assigned is used as a MOVE instruction argument.

< Without devices assigned >



Transferring all the members at once
(Number of steps: Small)

< With devices assigned >



Transferring each member
(Number of steps: Large)

The generation of transfer instructions can be suppressed by deleting unnecessary device assignment to structure instances. For subroutine type FBs, the generation of transfer instructions is suppressed by using only necessary structure members as arguments. Reducing the number of instructions internally generated can reduce the program conversion time.
Modifying structures has the following benefit.

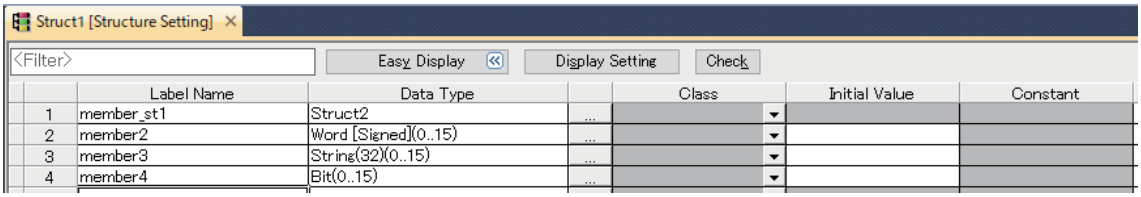
Reduction in program size

By modifying structures, the internal generation of transfer instructions can be suppressed, reducing the program size.

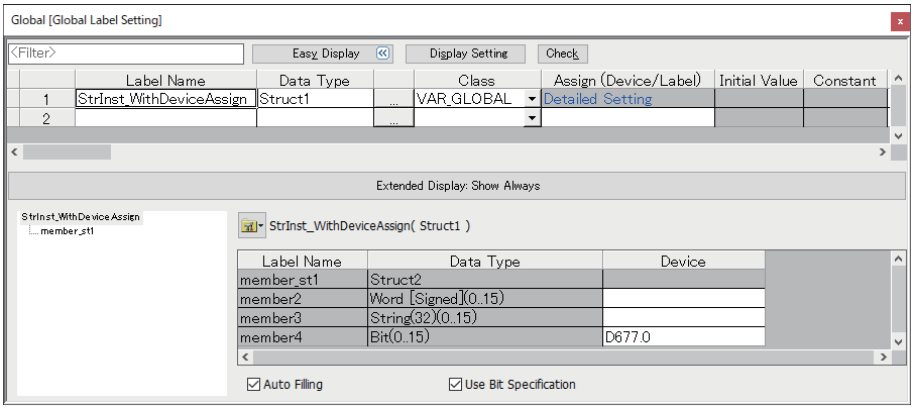
Execution procedure

Using the following program as an example, this section describes the procedure for deleting a device assigned to the instance "StrInst_WithDeviceAssign" of the structure "Struct1". In addition, it describes the procedure for modifying the program in which the subroutine type FB "FBPou_1" whose input argument is "StrInst_WithDeviceAssign" is used.

- Structure "Struct1"

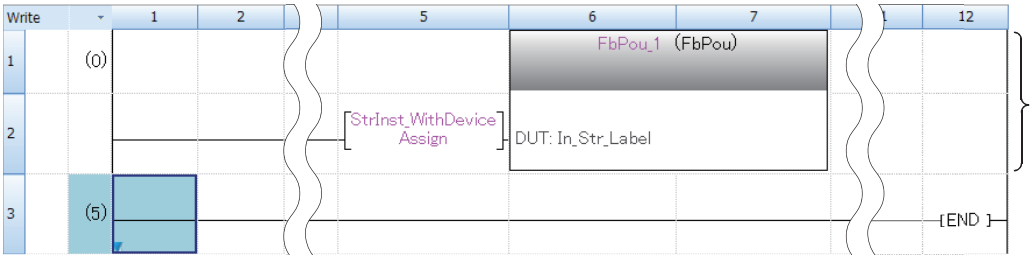


- Structure instance "StrInst_WithDeviceAssign" (global label)



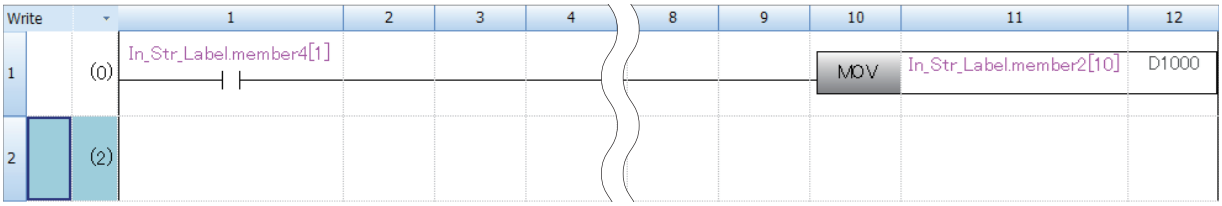
Assign the device (D677.0) to the member "member4".

- Program where the subroutine type FB "FBPou_1" is used

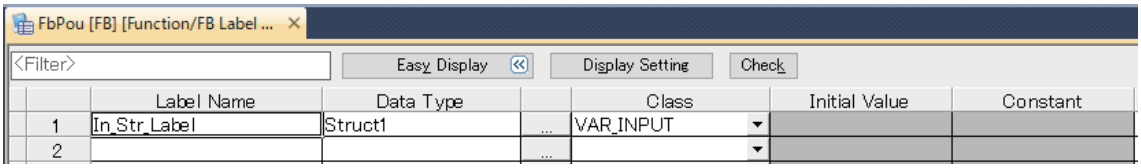


Specify the structure instance in the input argument of the FB.

- Program of the subroutine type FB "FBPou_1"



- Local label of the subroutine type FB "FBPou_1"

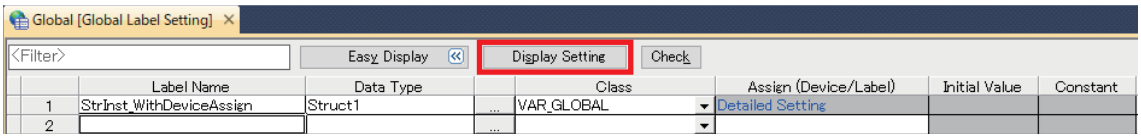


Deleting unnecessary device assignment to a structure instance

The following shows the procedure for deleting unnecessary device assignment to the structure instance "StrInst_WithDeviceAssign".

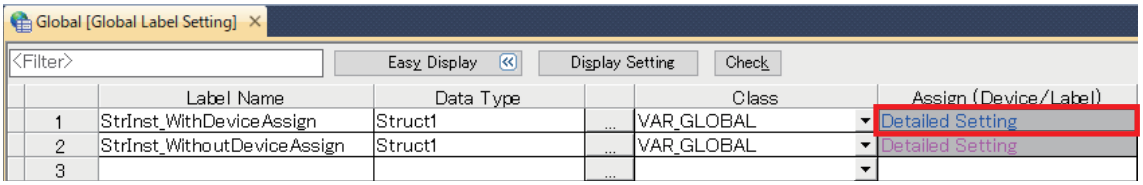
Operating procedure

1. Open the global label editor where the structure instance "StrInst_WithDeviceAssign" is registered and click the [Display Setting] button.

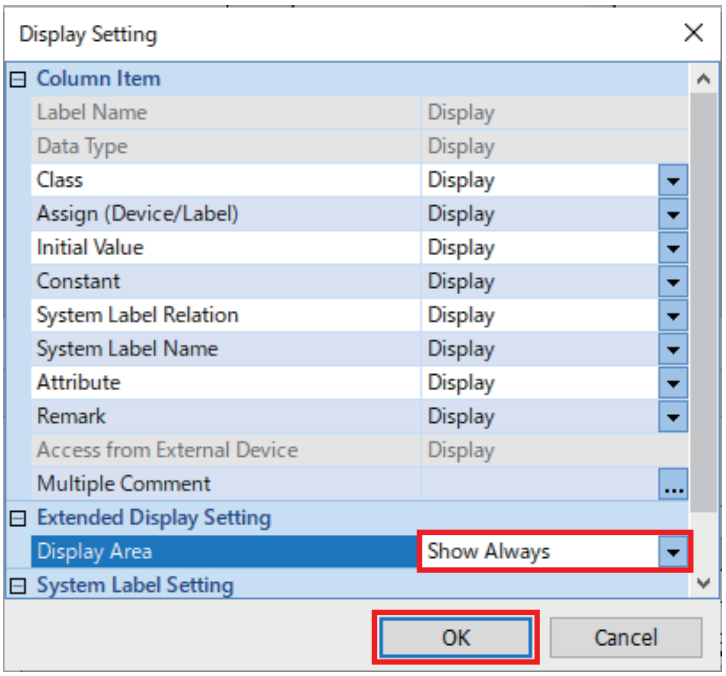


Point

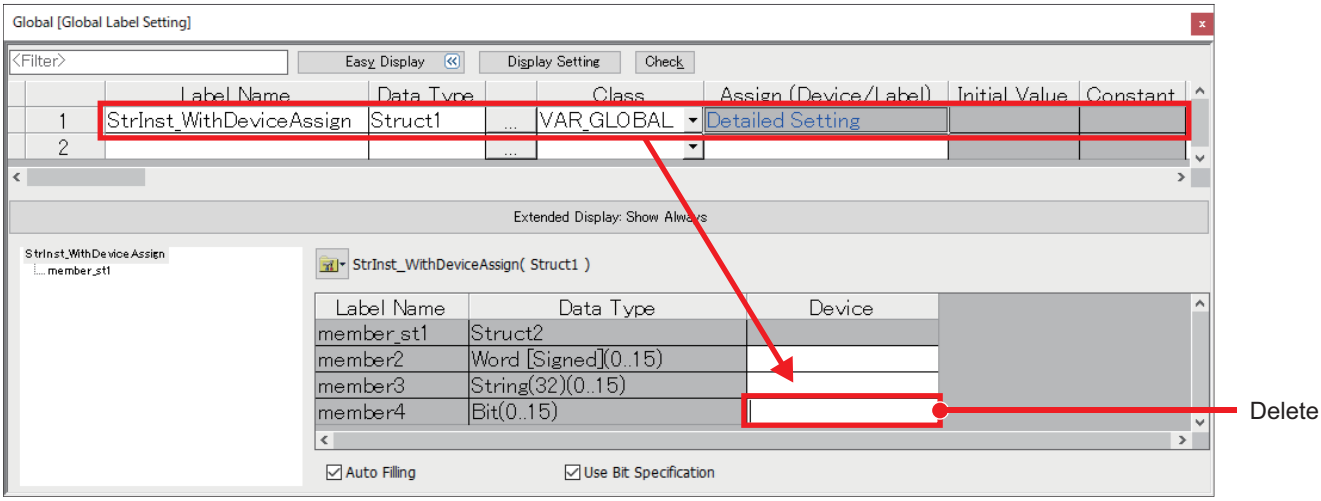
For structure instances to which devices are assigned, "Detailed Setting" in the "Assign (Device/Label)" column is displayed in blue.



2. Change [Display Area] under [Extended Display Setting] to "Show Always", and then click the [OK] button.



3. Select the structure instance to which any devices are assigned and delete the device "D677.0" assigned to member4 from the "Device" column in the extended display area.

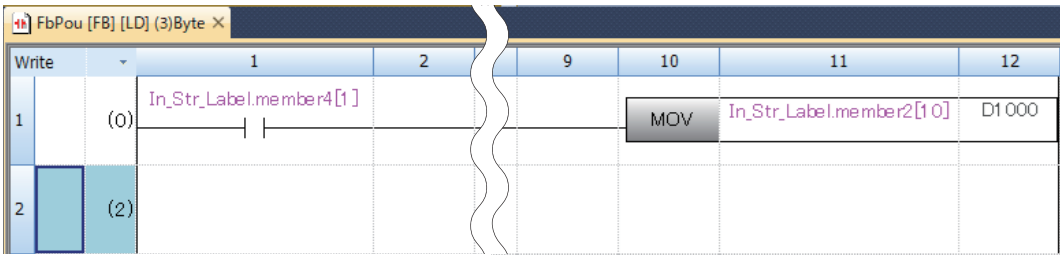


Modifying arguments of the subroutine type FB

The following shows the procedure for modifying the input argument "StrInst_WithDeviceAssign" of the subroutine type FB "FBPou_1" to be necessary structure members only.

Operating procedure

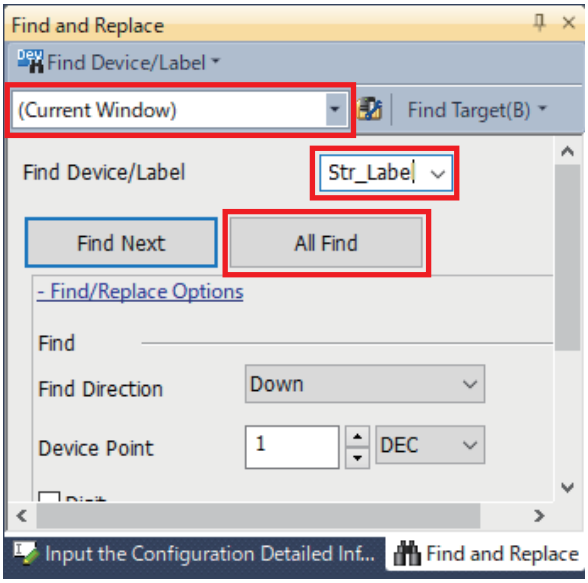
1. Open the program of the subroutine type FB "FBPou_1".



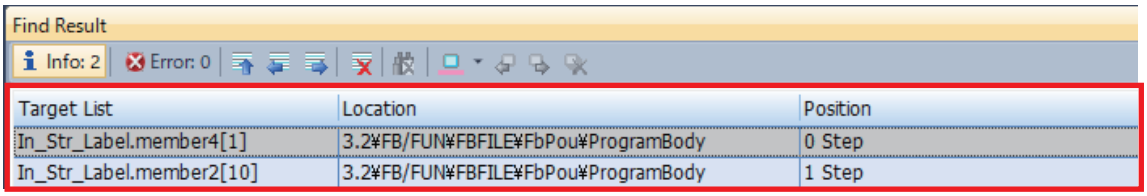
2. Search for the structure instance "In_Str_Label" used in the FB "FBPou_1".

In the "Find and Replace" window, set each item as follows, and then click the [All Find] button.

- Search range: (Current Window)
- Find Device/Label: In_Str_Label



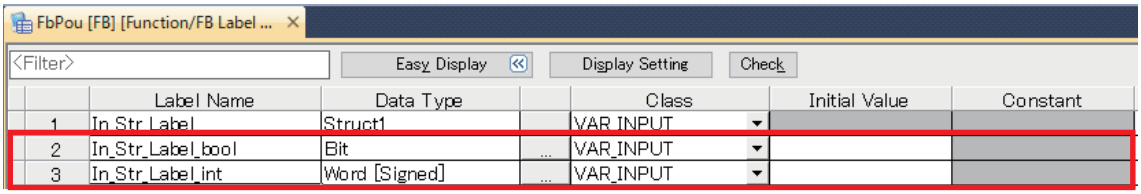
3. In the "Find Result" window, the list of items where the structure instance "In_Str_Label" is used is displayed.



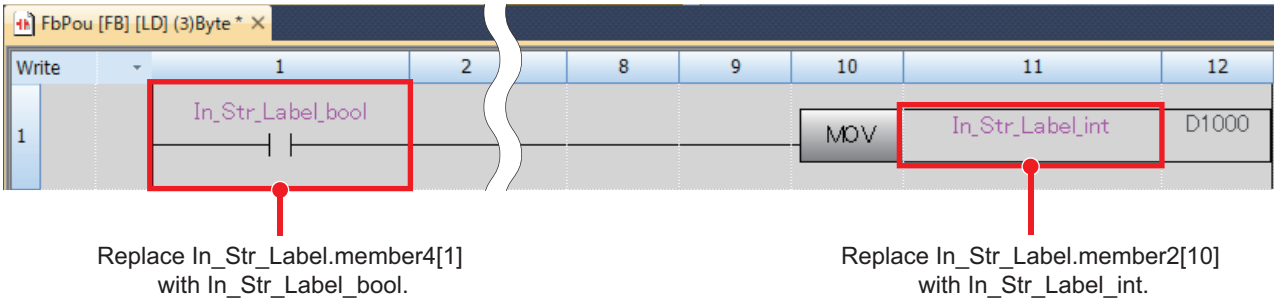
4. Open the local label editor of the FB "FBPou_1", create the following labels that can be used instead of the structure members searched in step 3.

Set the data type and class so that they match those for the structure member and structure instance.

- Label used instead of In_Str_Label.member4[1]: In_Str_Label_bool (Data Type: Bit, Class: VAR_INPUT)
- Label used instead of In_Str_Label.member2[10]: In_Str_Label_int (Data Type: Word [Signed], Class: VAR_INPUT)

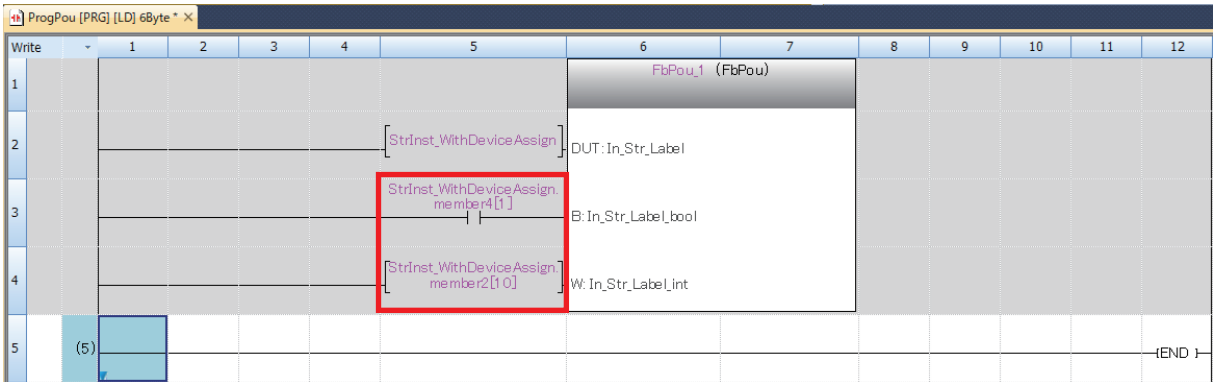


5. Open the program of the FB "FBPou_1", replace the structure members searched in step 3 with the labels created in step 4.

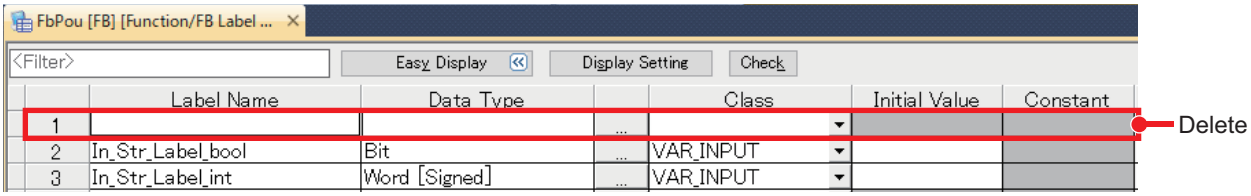


6. Open the program where the FB "FBPou_1" is used, set the members of the structure instance "StrInst_WithDeviceAssign" for each argument.

Set "StrInst_WithDeviceAssign.member4[1]" for the argument "In_Str_Label_bool" and set "StrInst_WithDeviceAssign .member2[10]" for the argument "In_Str_Label_int".



7. Open the local label editor of the FB "FBPou_1" and delete the unnecessary structure instance "In_Str_Label".



3.5 Reducing Consecutive References to the Same Array Element

Overview

If array elements are specified dynamically (if a device or label is specified as the array index), multiple instructions to access the memory are generated when the program is executed.
Therefore, if there are multiple arrays that access the same element, copying it to a different memory once can reduce the number of instructions to access the memory output when the program is executed. This can reduce the number of instructions internally generated, reducing the program conversion time and scan time.

Execution procedure

Using the following program as an example, this section describes the procedure for reducing consecutive references to the same array element.

• Program

ProgPou1 [PRG] [ST]

```
1  IF  (targetStLabel[D1, D2] > 5)  THEN
2      destLabel1 := targetStLabel[D1, D2] ;
3      destLabel2 := targetStLabel[D1, D2] + K10 ;
4      destLabel3 := targetStLabel[D1, D2] * K2 ;
5      destLabel4 := targetStLabel[D1, D2] + targetStLabel[D10, D20];
6  //  ELSE
7  //   ?Statement? ;
8  END_IF;
9
```

targetStLabel[D1, D2] is accessed multiple times.

• Local label

ProgPou1 [PRG] [Local Label Se...

<Filter> | Easy Display | Display Setting | Check

	Label Name	Data Type		Class	Initial Value	Constant
1	targetStLabel	Word [Signed](0..2,0..2)	...	VAR		
2	destLabel1	Word [Signed]	...	VAR		
3	destLabel2	Word [Signed]	...	VAR		
4	destLabel3	Word [Signed]	...	VAR		
5	destLabel4	Word [Signed]	...	VAR		

Operating procedure

1. Open the program and determine the part to be replaced (part where the same array element is consecutively referenced).

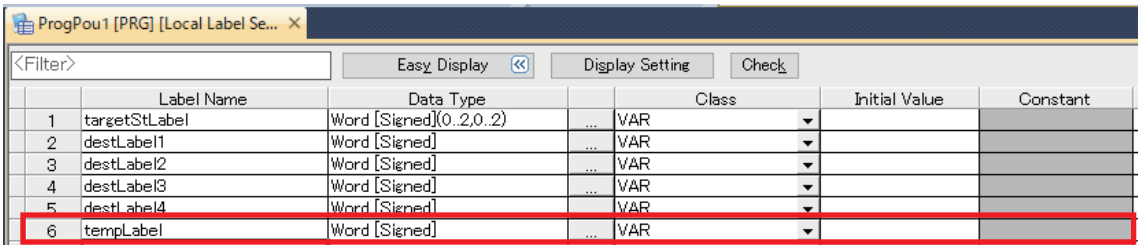
In this example, "targetStLabel[D1, D2]", which is used five times in the program, is the target to be replaced.

ProgPou1 [PRG] [ST]

```
1  IF  targetStLabel[D1, D2] > 5)  THEN
2      destLabel1 := targetStLabel[D1, D2] ;
3      destLabel2 := targetStLabel[D1, D2] + K10 ;
4      destLabel3 := targetStLabel[D1, D2] * K2 ;
5      destLabel4 := targetStLabel[D1, D2] + targetStLabel[D10, D20];
6  //  ELSE
7  //   ?Statement? ;
8  END_IF;
9
```

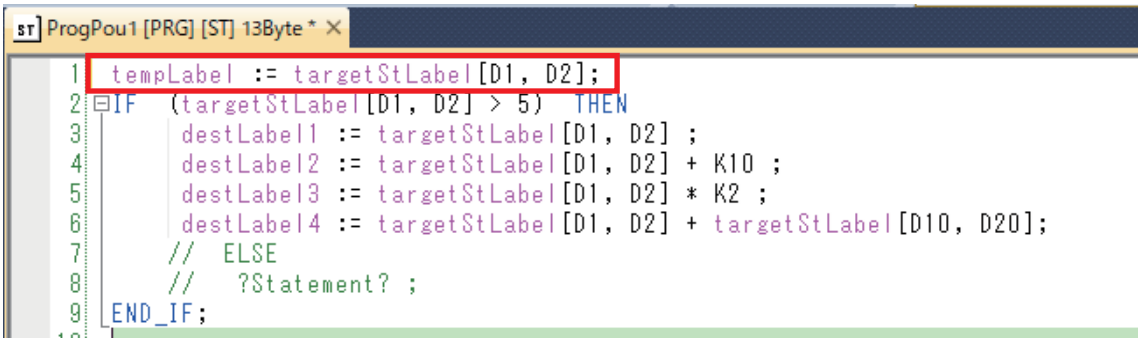
2. Open the local label editor of the program, and add the local label "tempLabel", which is used instead of the label to access the array element.

- Label Name: tempLabel
- Data Type: Word [Signed]
- Class: VAR



3. Open the program and add the following syntax in the first row of the program to evacuate "targetStLabel[D1, D2]" to the local label "tempLabel" created in step 2.

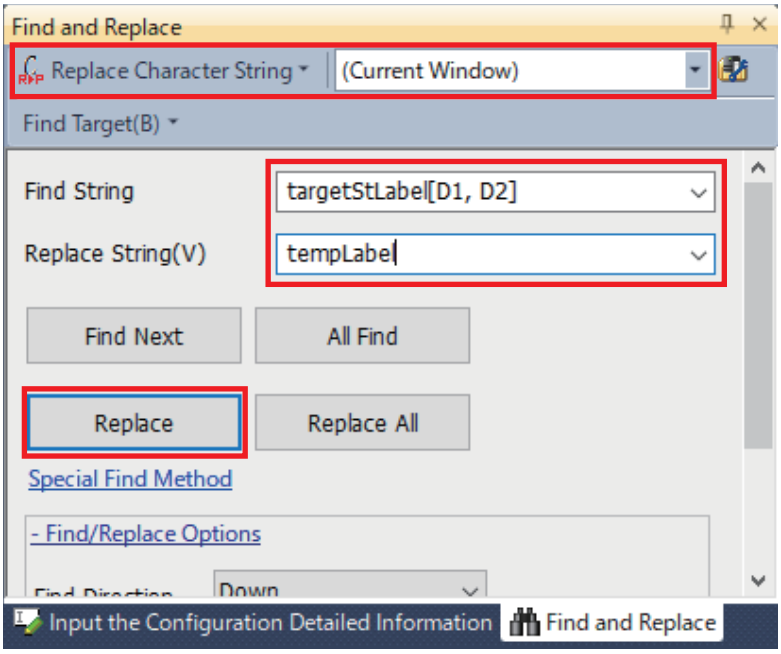
- Evacuation syntax: tempLabel := targetStLabel[D1, D2];



4. Replace "targetStLabel[D1, D2]" with the local label "tempLabel" created in step 2.

In the "Find and Replace" window, set each item as follows, and then click the [Replace] button. (Repeat this step until all the target labels are replaced.)

- Search/replacement method: Replace Character String
- Search range: (Current Window)
- Find String: targetStLabel[D1, D2]
- Replace String: tempLabel



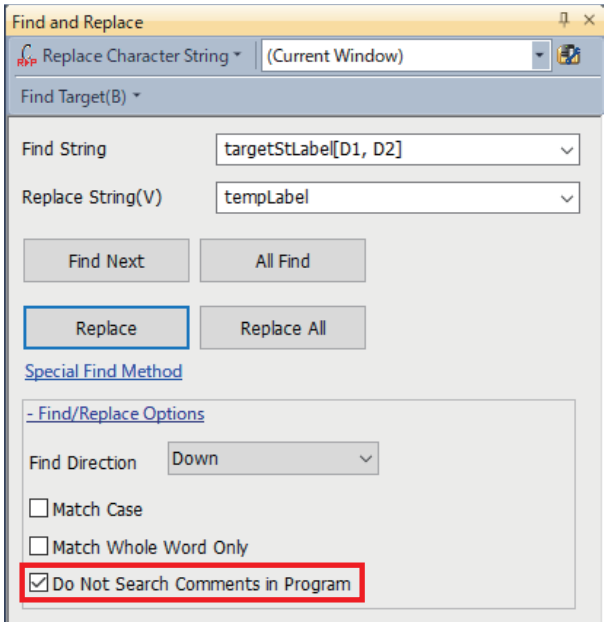
After the replacement, the program will look as follows.

ProgPou1 [PRG] [ST] 13Byte *

```
 1  tempLabel := targetStLabel[D1, D2];
 2  IF (tempLabel > 5) THEN
 3      destLabel1 := tempLabel ;
 4      destLabel2 := tempLabel + K10 ;
 5      destLabel3 := tempLabel * K2 ;
 6      destLabel4 := tempLabel + targetStLabel[D10, D20];
 7      //  ELSE
 8      //    ?Statement? ;
 9  END_IF;
```

Point

- Although batch replacement is available with the [Replace All] button, it is recommended that all the found strings be checked one by one when replaced.
- When not replacing comments in the program, select the "Do Not Search Comments in Program" checkbox in the "Find/Replace Options" section.



Precautions

- This technique is effective when the array index is not a constant.
- Depending on the contents of the program (for example, if the device or label used to dynamically specify the array element is rewritten during the program), the label used to access the array elements may not be replaced. Check the program contents before performing the replacement.

The D1 value of "targetStLabel[D0, D1]" is rewritten during the program.

ProgPou2 [PRG] [ST] *

```
 1  IF (targetStLabel[D1, D2] > 5) THEN
 2      destLabel1 := targetStLabel[D1, D2] ;
 3      destLabel2 := targetStLabel[D1, D2] + K10 ;
 4      D1 := 0 ;
 5      destLabel3 := targetStLabel[D1, D2] * K2 ;
 6      destLabel4 := targetStLabel[D1, D2] + targetStLabel[D10, D20];
 7      //  ELSE
 8      //    ?Statement ;
 9  END_IF;
```

4 REDUCTION OF SCAN TIME

The scan time can be reduced by adjusting the program contents and parameter settings. By reducing the scan time, the processing time of the program can be reduced.
This chapter describes techniques for reducing the scan time.

List of techniques

The following table lists the techniques described in this chapter.

Item	Description	Reference
Replacing a BMOV instruction with multiple instructions	Replace a BMOV instruction with multiple MOV instructions using FOR to NEXT instructions.	☞ Page 54 Replacing a BMOV Instruction with Multiple Instructions
Reducing rising edge execution instructions	Reduce the number of rising edge execution instructions (instructions executed only at the rising edge).	☞ Page 59 Reducing Rising Edge Execution Instructions
Reducing module access instructions	Change the part where a buffer memory area of the intelligent module is directly accessed to automatic refresh or link refresh of the intelligent module.	☞ Page 61 Reducing Module Access Instructions
Utilizing initial execution type programs	In ST programs, move the processes that are executed only once to the initial execution type programs.	☞ Page 64 Utilizing Initial Execution Type Programs
Omitting arguments and utilizing direct references	In ST programs, omit or delete the assignment processes to the I/O arguments of the motion control FB and any unnecessary assignments.	☞ Page 70 Omitting Arguments and Utilizing Direct References
Dividing a circuit block with a large number of instructions	In ladder programs, reduce the number of instructions contained in one circuit block to 100 or less.	☞ Page 74 Dividing a Circuit Block with a Large Number of Instructions
Utilizing label initial values	Use label initial values for setting the initial values of labels or FBs that will not be changed.	☞ Page 77 Utilizing Label Initial Values

How to check the scan time

The scan time value can be checked in the following special registers.
To check the scan time when using each technique, use them.

No.	Name	Description	Details	Setting
SD518	Initial scan time	Initial scan time (unit: ms)	The initial scan time is stored into SD518 and SD519 (it is measured in increments of μ). SD518: The ms part is stored. (Storage range: 0 to 65535) SD519: The μ part is stored. (Storage range: 0 to 999) These areas are cleared to 0 when the operating status of the controller is switched from STOP → RUN.	Set by the system every time the END processing is performed
SD519		Initial scan time (unit: μs)		
SD520	Current scan time	Current scan time (unit: ms)	The current scan time is stored into SD520 and SD521 (it is measured in increments of μ). SD520: The ms part is stored. (Storage range: 0 to 65535) SD521: The μs part is stored. (Storage range: 0 to 999) These areas are cleared to 0 when the operating status of the controller is switched to STOP.	
SD521		Current scan time (unit: μs)		
SD522	Minimum scan time	Minimum scan time (unit: ms)	The minimum value of the scan time other than one for the initial execution program is stored into SD522 and SD523 (it is measured in increments of μ). SD522: The ms part is stored. (Storage range: 0 to 65535) SD523: The μ part is stored. (Storage range: 0 to 999) These areas are cleared to 0 when the operating status of the controller is switched from STOP → RUN.	
SD523		Minimum scan time (unit: μs)		
SD524	Maximum scan time	Maximum scan time (unit: ms)	The maximum value of the scan time other than one for the initial execution program is stored into SD524 and SD525 (it is measured in increments of μ). SD524: The ms part is stored. (Storage range: 0 to 65535) SD525: The μ part is stored. (Storage range: 0 to 999) These areas are cleared to 0 when the operating status of the controller is switched from STOP → RUN.	
SD525		Maximum scan time (unit: μs)		

For ladder programs, the scan time measurement function can be used.
For details on the scan time measurement function, refer to the following manual.
📖GX Works3 Operating Manual

4.1 Replacing a BMOV Instruction with Multiple Instructions

Overview

Because the execution speed of MOV instructions is faster than that of BMOV instructions, replacing BMOV instructions with MOV instructions can reduce the scan time.
There are the following two methods for replacing BMOV instructions: replacing them with multiple MOV instructions or using FOR to NEXT instructions to execute the MOV instruction multiple times.

Performance values of MOV and BMOV instructions

The following table lists the performance value examples of MOV (INC) and BMOV instructions in the MXR500-256.
Refer to the table when replacing BMOV instructions with MOV instructions.

Instruction	Execution speed (μs)
MOV instruction	0.00050
INC instruction	0.00361
BMOV instruction (when the number of transfers is 1)	0.076
BMOV instruction (when the number of transfers is 1024)	0.191

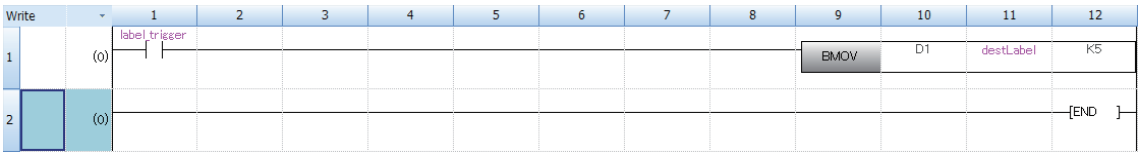
Precautions

The performance values differ depending on the model.

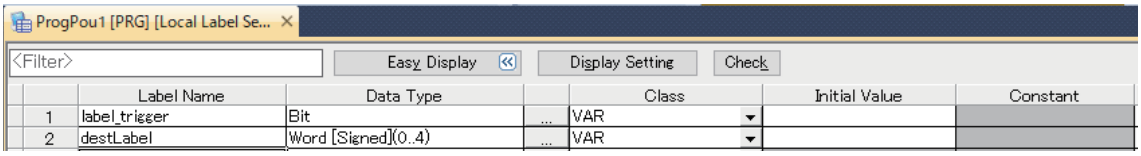
Execution procedure

Using the following program as an example, this section describes the procedure for replacing a BMOV instruction with MOV instructions.

- Program



- Local label

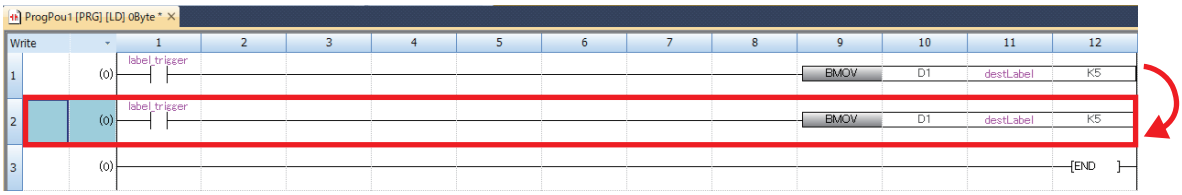


Replacement with multiple MOV instructions

The following shows the procedure for replacing a BMOV instruction with multiple MOV instructions.

Operating procedure

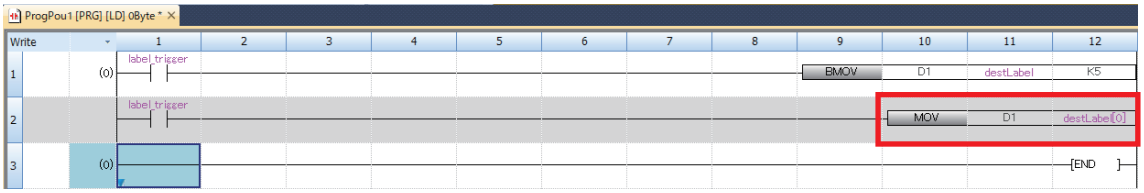
1. Open the program where the BMOV instruction is used, copy the row of the BMOV instruction, and then paste it in the next row.



2. Change the BMOV instruction pasted in step 1 to a MOV instruction.

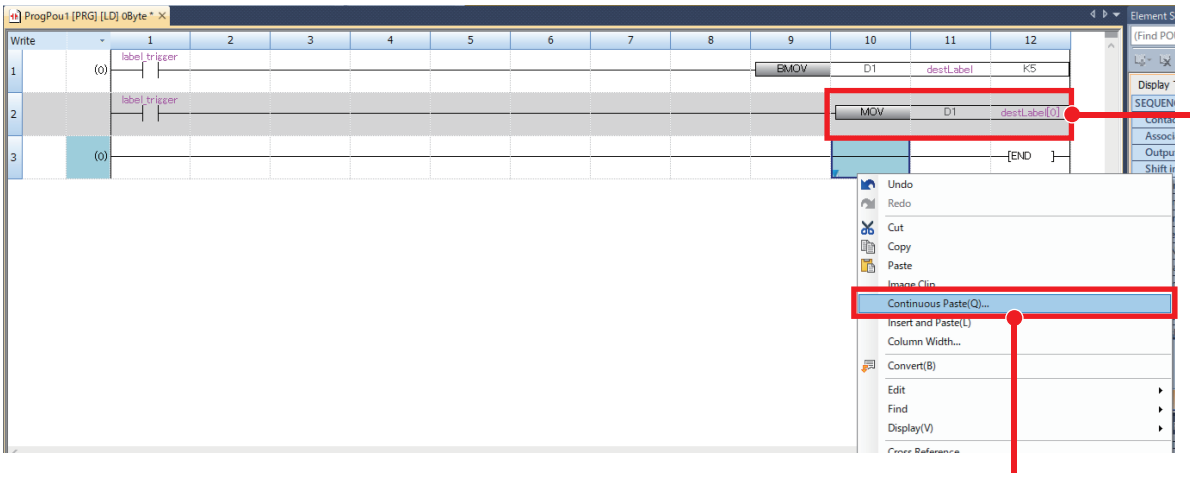
Change the argument so that the "D1" value is transferred to the local label "destLabel[0]".

- MOV D1 destLabel[0]



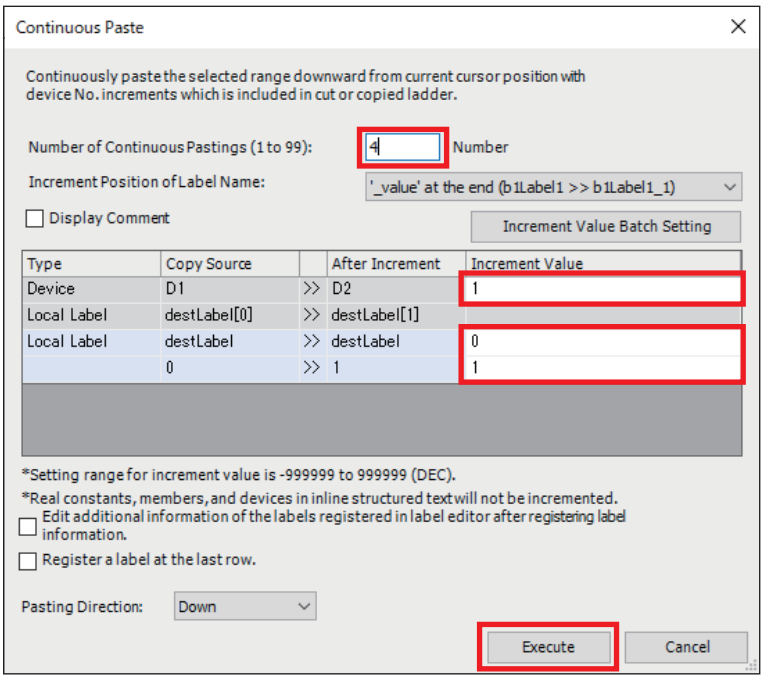
3. Copy the MOV instruction modified in step 2 and perform continuous pasting at the position where the next MOV instruction is to be inserted.

Right-click the position where the MOV instruction is to be inserted ⇨ [Continuous Paste]

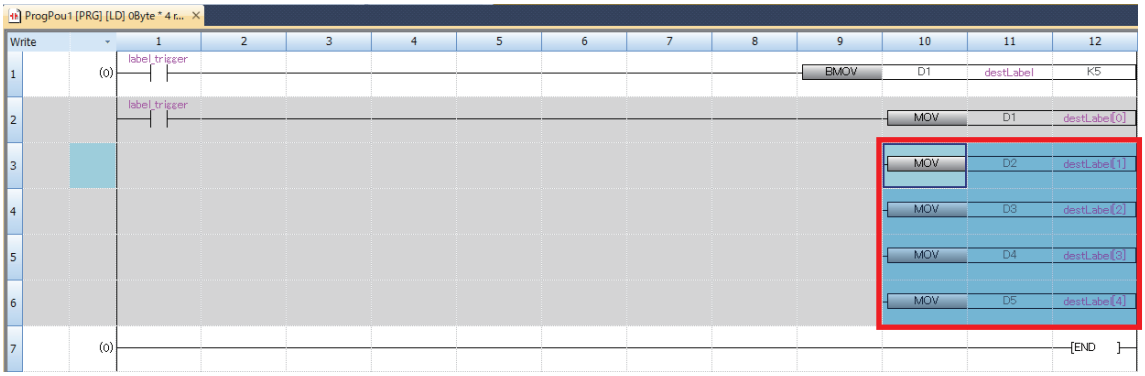


4. Set the number of continuous pastings and increment values, and then click the [Execute] button.

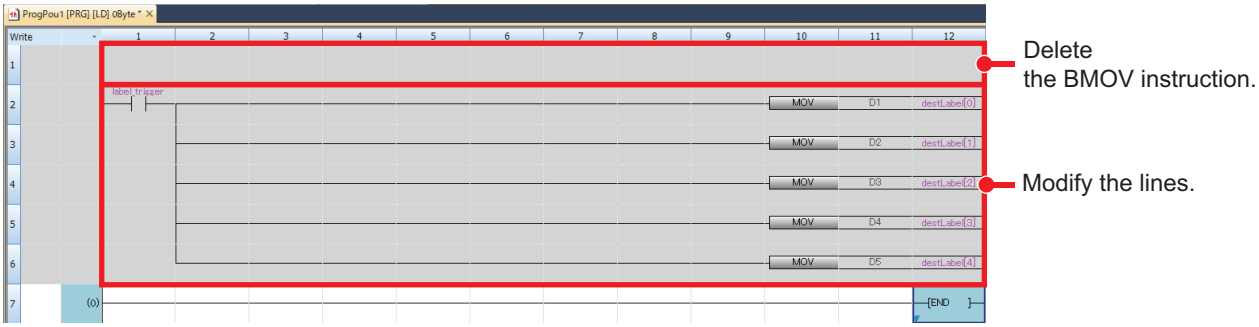
- Number of Continuous Pastings: 4 Number
- Number of increment values of the device "D1": 1
- Number of increment values of the array element of the local label "destLabel": 1



Continuous pasting is performed.



5. Modify the lines so that each MOV instruction is connected to the contact of the row added in step 1 and delete the unnecessary line of the BMOV instruction.



Replacement with FOR to NEXT instructions and MOV instruction

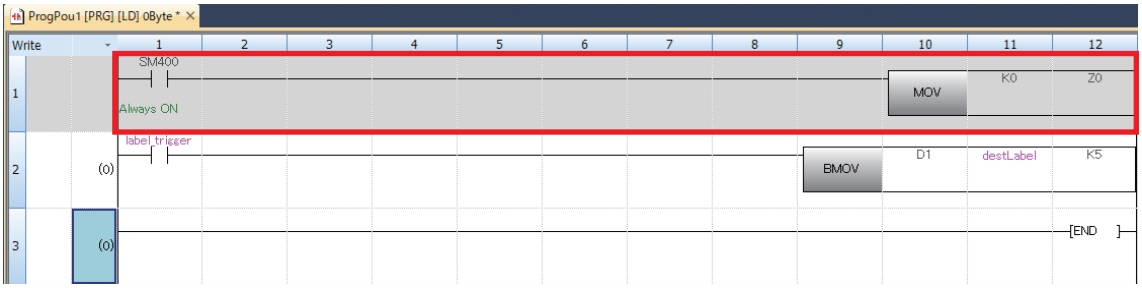
The following shows the procedure for replacing a BMOV instruction with a program in which a MOV instruction is executed multiple times using FOR to NEXT instructions.

Operating procedure

1. Open the program where the BMOV instruction is used, insert a new row into the first row, and add a circuit to initialize the index register "Z0".

Use the special relay SM400 (always ON) as the contact and use the following MOV instruction to transfer K0 to the index register "Z0".

- MOV K0 Z0



Precautions

Index registers are used to indirectly specify device numbers. If any other index registers are used in the other sections of the program, be careful not to use the same index register number.

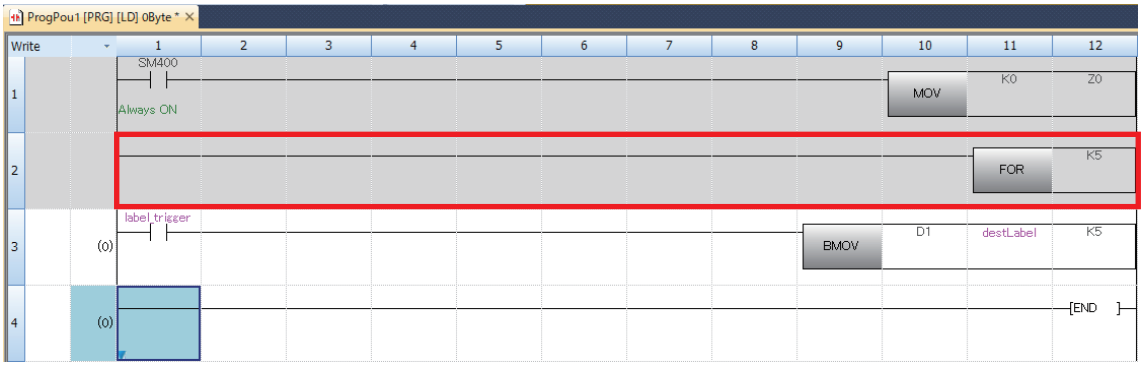
For details on index registers, refer to the following manual.

MELSEC MX Controller (MX-R Model) Programming Manual

2. Insert a new row into the second row and add a FOR statement.

To repeat the MOV instruction for the number of executions (number of transfers), set "K5" in the argument.

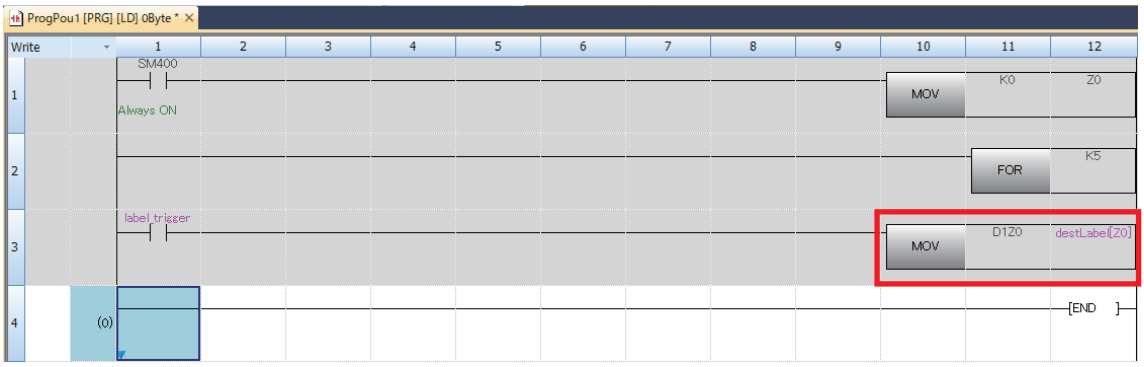
- FOR K5



3. Change the BMOV instruction to a MOV instruction.

Change the argument so that the "D1Z0" value is transferred to the local label "destLabel[Z0]" using the index register initialized in step 1.

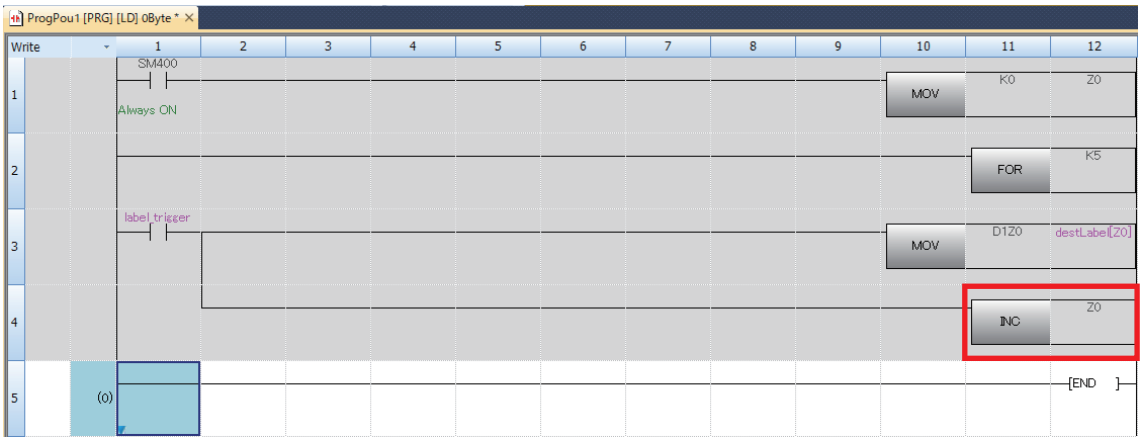
- MOV D1Z0 destLabel[Z0]



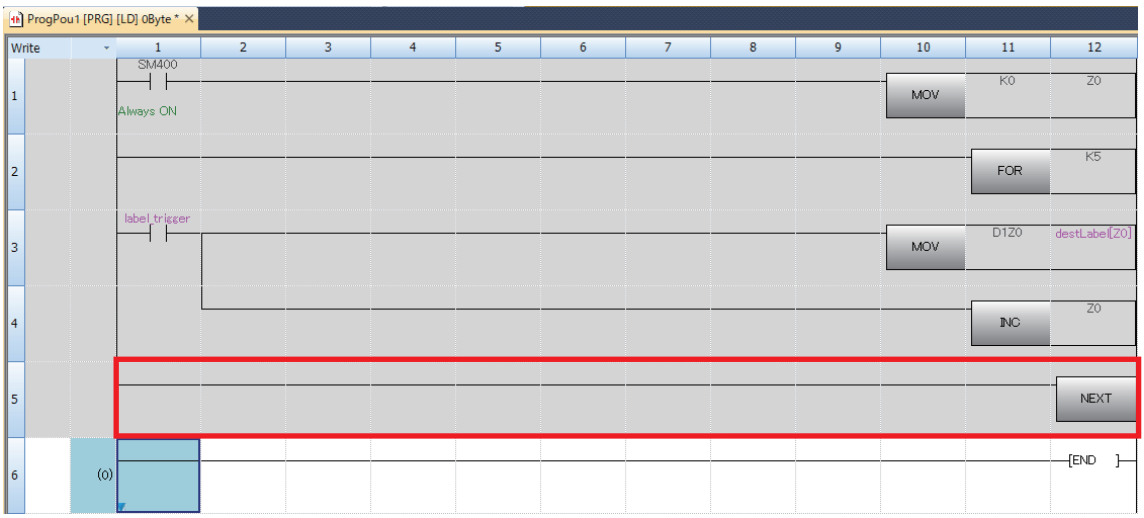
4. In the next row of the MOV instruction, add the following INC instruction to increase the index register "Z0" by one.

Connect the added INC instruction to the contact of the MOV instruction.

- INC Z0



5. Add a NEXT statement in the next row of the INC instruction.



Precautions

- When a BMOV instruction is replaced with multiple MOV instructions, the program readability may decrease.
- If the number of transfers of the BMOV instruction is large, replacing it with MOV instructions may not reduce the scan time. Before replacing it, calculate and check the execution speed of the MOV and BMOV instructions. For the execution speed, refer to the following.

☞ Page 54 Performance values of MOV and BMOV instructions

4.2 Reducing Rising Edge Execution Instructions

Overview

Since rising edge execution instructions (instructions with "P" at the end of the instruction symbol executed only at the rising edge) have slower processing performance compared to ON execution instructions, reducing the number of rising edge execution instructions can reduce the scan time.

The number of rising edge execution instructions can be reduced by either of the following methods.

- Replace unnecessary rising edge execution instructions with ON execution instructions.
- If multiple rising edge execution instructions are used, change the contact to a rising edge execution instruction and change the other rising edge execution instructions to ON execution instructions.

Performance values of rising edge execution instructions

When the execution condition is a rising edge instruction, additional time is added to the processing time of the ON execution instruction. (Example: Processing time of MOVP = Processing time of MOV + added time)

The following table lists the added time for each model.

Refer to the table when replacing or changing rising edge instructions.

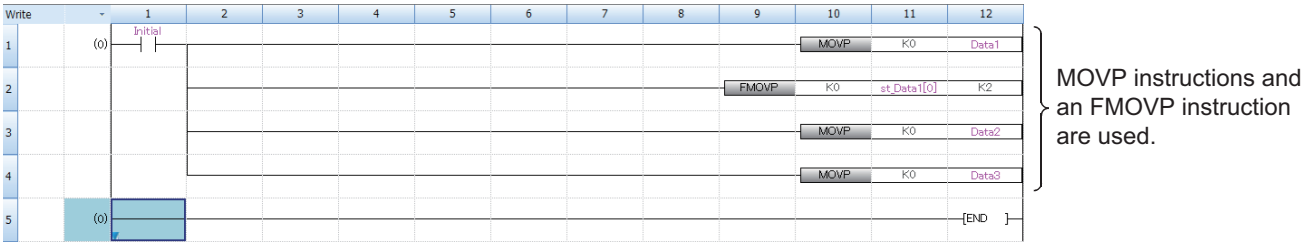
Model	Added time (n seconds)
MXR300-16, MXR300-32, MXR300-64	28.62
MXR500-128, MXR500-256	20.81

Precautions

The performance values differ depending on the model.

Execution procedure

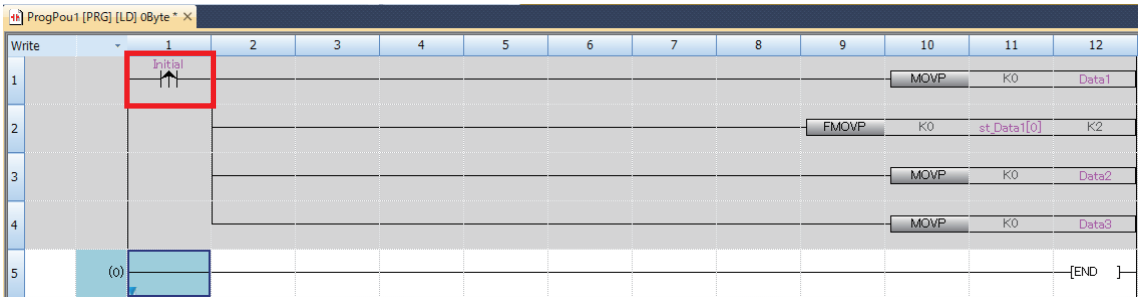
Using the following program as an example, this section describes the procedure for reducing the number of rising edge execution instructions "MOVP" and "FMOVP".



Operating procedure

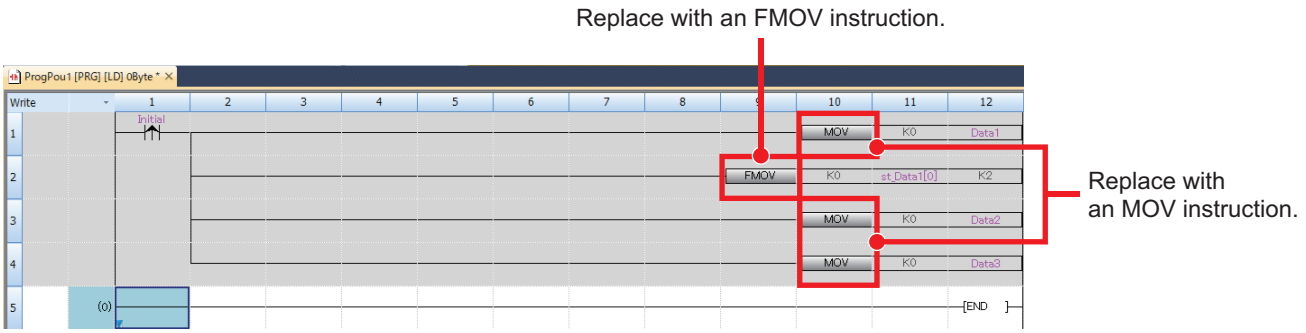
1. Open the program and change the contact with multiple connected rising edge execution instructions to a rising edge execution instruction.

In this example, the normally open contact operation start instruction (LD) is changed to the rising edge pulse operation start instruction (LDP).



2. Change the rising edge execution instructions connected to the contact described in step 1 to ON execution instructions.

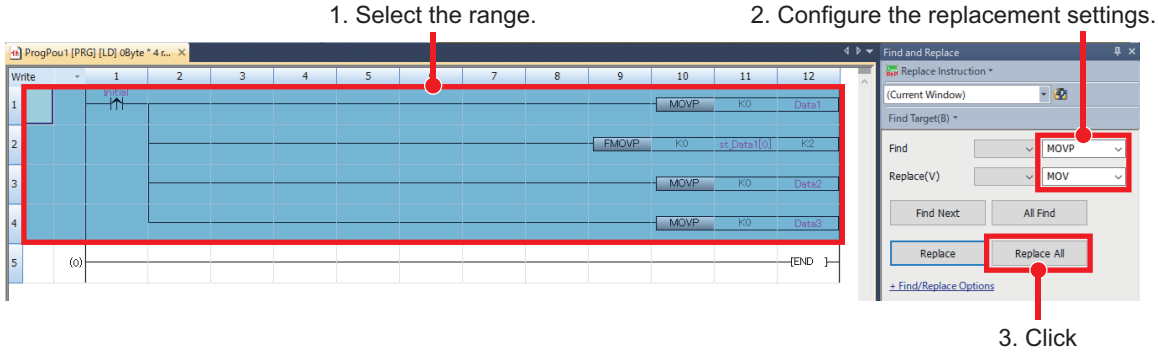
- Changing MOVP instruction → MOV instruction
- Changing FMOVP instruction → FMOV instruction



Point

If there are multiple instructions of the same instruction type, batch replacement can be performed under "Replace Instruction" in the "Find and Replace" window.

Set the searching target instruction and the replacement destination instruction with the replacement range selected and click the [Replace All] button.



4.3 Reducing Module Access Instructions

Overview

When the module buffer memory is accessed, the scan time can be reduced by consolidating direct access to the buffer memory using module access instructions such as FROM and TO instructions, or by changing the access to the one using the automatic refresh function or link refresh function of the intelligent module.

Performance values of module access instructions and link refresh

The following table lists the performance value examples of the module access instructions, automatic refresh function, and link refresh function in the MXR500-128 and MXR500-256.
Refer to the table when replacing module access instructions.

Item	Minimum execution performance value per word(n seconds)		
	Number of transfers: 1 word	Number of transfers: 1024 words	Number of transfers: 32768 words
FROM instruction	5470	40.42	—
TO instruction	3740	23.43	—
Automatic refresh (read)*1	10730	30.45	20.32
Automatic refresh (write)*1	7360	17.17	10.22
Link refresh (CC-Link IE TSN module)*1	52012	62.78	13.58

*1 It is calculated using the formula provided in the manual of the controller. For details, refer to the following manual.
MELSEC MX Controller (MX-R Model) User's Manual

Precautions

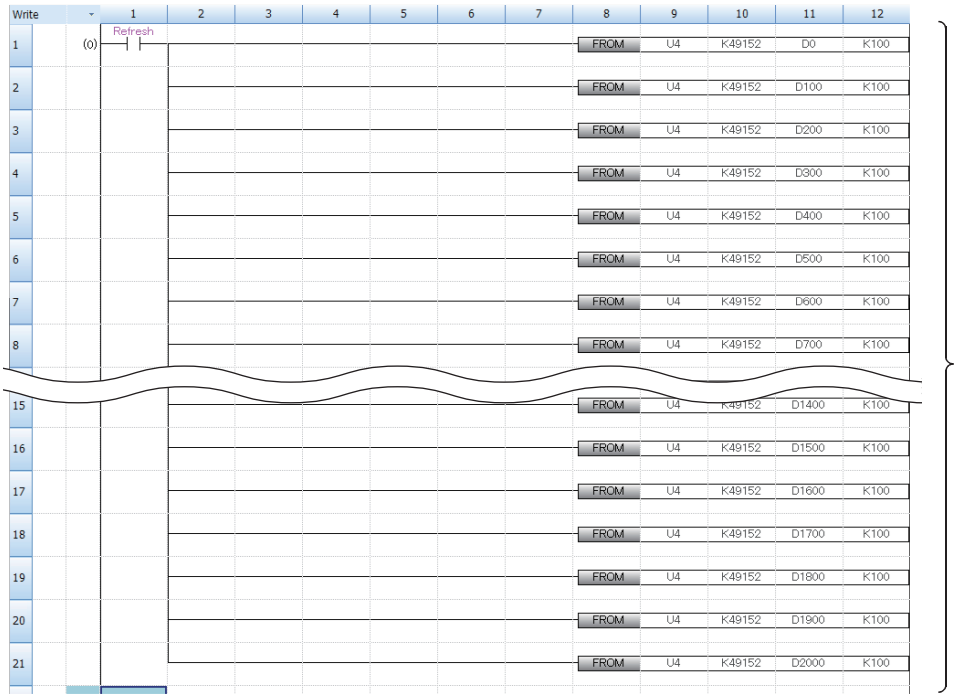
The performance values differ depending on the model.

Execution procedure

The following describes the procedure for reducing the number of module access instructions.

Consolidating FROM and TO instructions

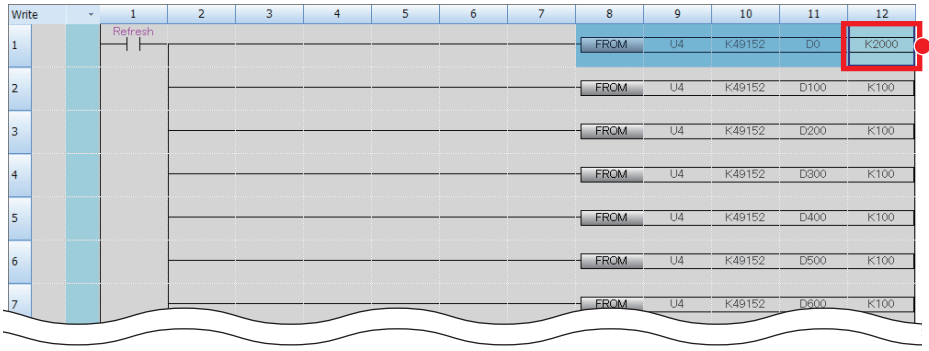
Using the following program, which reads 2000 points (2000 words) of values from the buffer memory area starting from the I/O address 0x0040 of the intelligent function module to the D device, as an example, this section describes the procedure for consolidating 20 FROM instructions into a single FROM instruction.



2000 words of data are read by using 20 FROM instructions, each reading 100 words.

Operating procedure

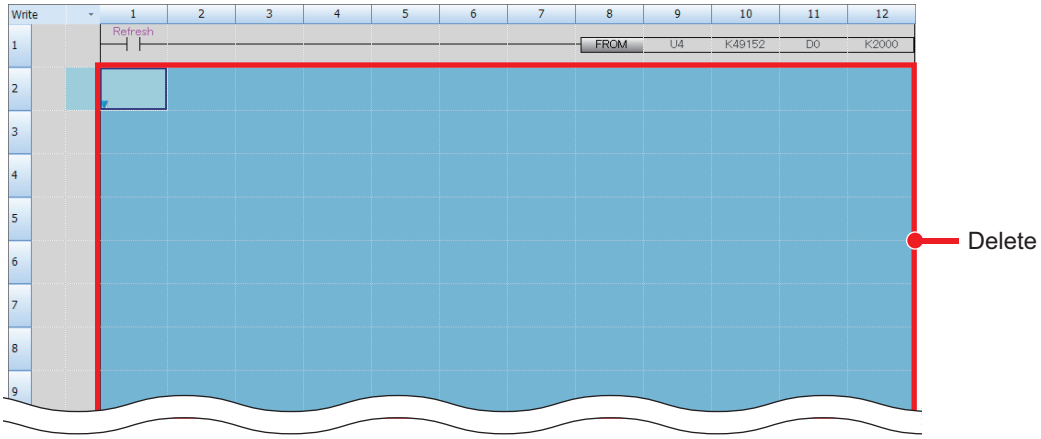
1. Open the target program file and change the number of read data of the FROM instruction in the first row to K2000 (2000 words).



Change to K2000.

2. Delete the FROM instructions in the second and subsequent rows.

This consolidates 20 FROM instructions, which read 2000 words of data in unit of 100 words, into a single FROM instruction which reads 2000 words of data at once.



Replacement with the automatic refresh and link refresh functions

Replace module access instructions with the automatic refresh and link refresh functions.

For how to use the automatic refresh and link refresh functions, refer to the manual of each module.

4.4 Utilizing Initial Execution Type Programs

Overview

A scan execution type program is executed once a scan.
If there are processes, which need to be executed only once after the programmable controller is switched to RUN, in a scan execution type ST program (motion control program), moving them to an initial execution type ST program can reduce the scan time.

Execution procedure

Using the following ST program as an example, this section describes the procedure for moving processes that need to be executed only once after the programmable controller is switched to RUN in the program to an initial execution type program.

• ST program

ProgPou [PRG] [ST] 52Byte

```
1  WdRollWidth := 300.0;        //Sheet length [mm]
2  WdSpeedMax := 600.0;         //Maximum velocity [r/min]
3  WdMaxDia := 120.0;           //Roll diameter maximum value [mm]
4  MotorInertia := 0.0777e-4;   //Inertia [kg·m^2]
5
6  G_MCv_AllPower_1(
7  //  Axis:= ?AXIS_REF? ,
8     Enable:= TRUE ,
9     ServoON:= TRUE
10 //  Busy=> ?BOOL? ,
11 //  Error=> ?BOOL? ,
12 //  ErrorID=> ?WORD?
13     );
14
```

Processes executed only once after the PLC switches to RUN

• Local label

ProgPou [PRG] [Local Label Sett...

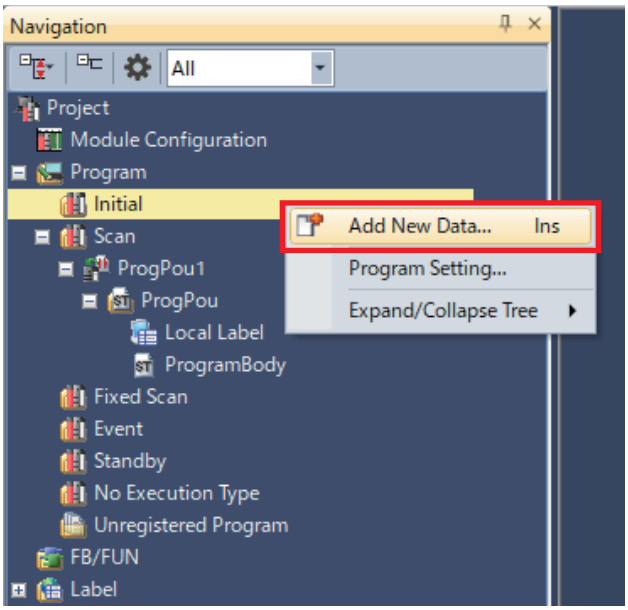
<Filter> | Easy Display | Display Setting | Check

	Label Name	Data Type		Class	Initial Value	Constant
1	WdRollWidth	FLOAT [Single Precision]	...	VAR		
2	WdSpeedMax	FLOAT [Single Precision]	...	VAR		
3	WdMaxDia	FLOAT [Single Precision]	...	VAR		
4	MotorInertia	FLOAT [Single Precision]	...	VAR		
5	MCv_Jog_1	MCv_Jog	...	VAR		

Operating procedure

1. Newly create an initial execution type program file.

Navigation window ⇨ [Program] ⇨ Right-click [Initial] ⇨ [Add New Data]

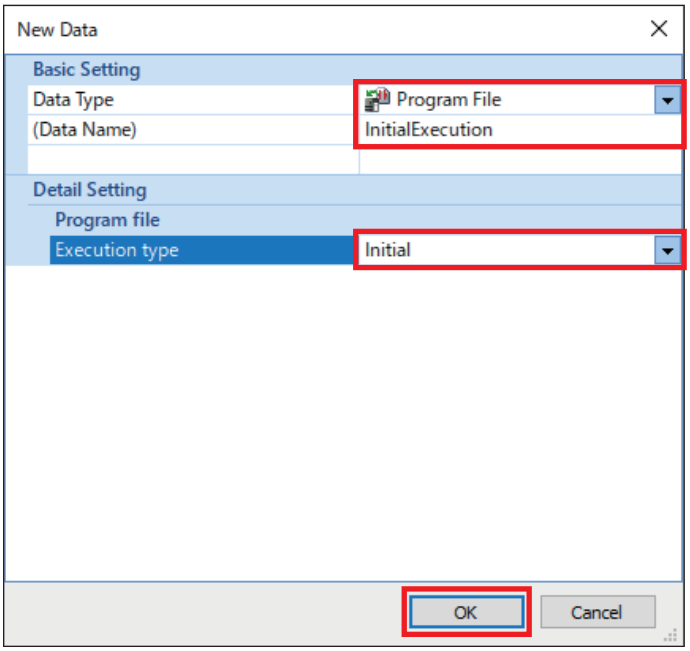


Point

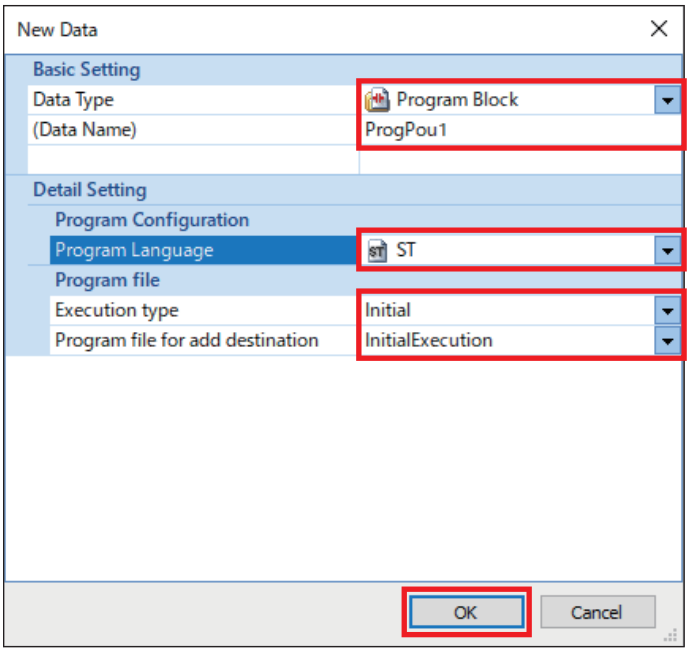
The smaller the amount of code per program file is, the shorter the program conversion time is. Therefore, it is recommended that processes executed only once after the programmable controller is switched to RUN be moved to a new initial execution type program file. (There will be no problem in operation if they are moved to an existing initial execution type program file.)

2. Configure the following settings, and then click the [OK] button.

- Data Type: Program File
- (Data Name): InitialExecution
- Execution type: Initial

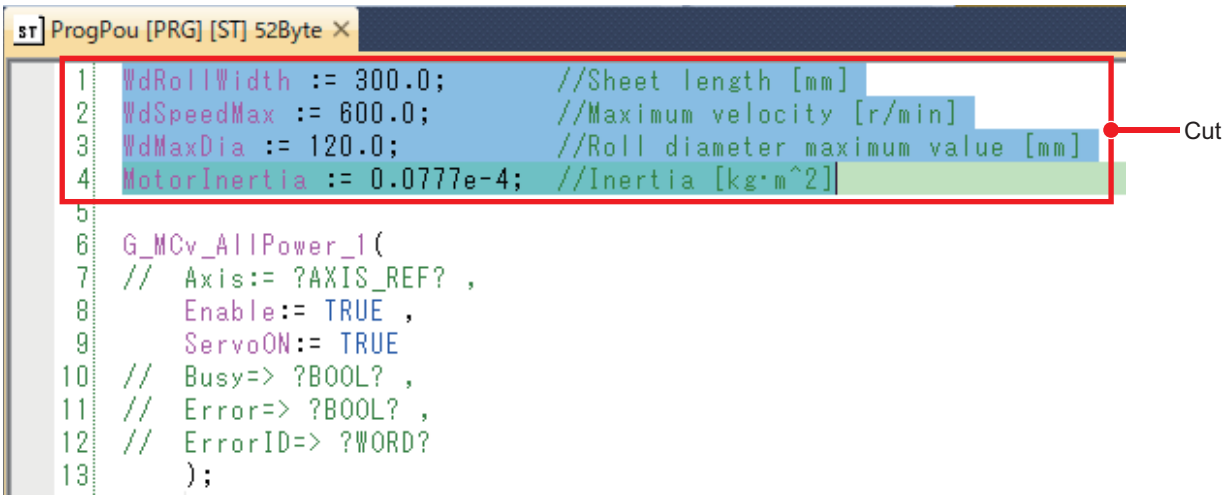


3. Open the "New Data" window in the same manner as step 1 and create a new program block. Configure the following settings, and then click the [OK] button.

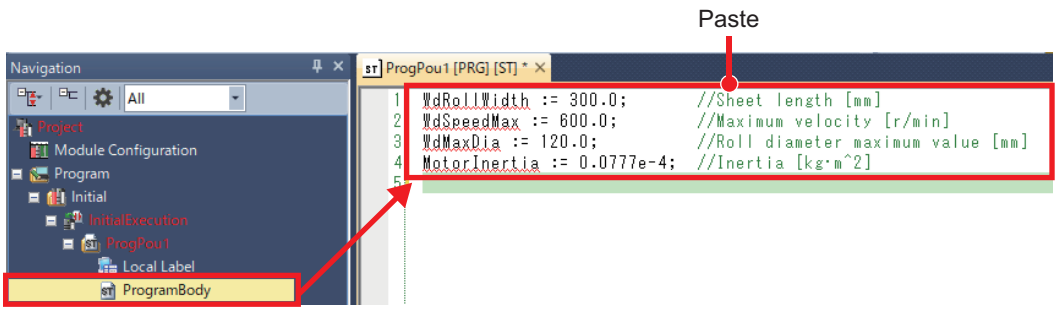


Item	Setting	Remarks
Data Type	Program Block	—
(Data Name)	ProgPou1	—
Program Language	ST	—
Execution type	Initial	Set the same execution type as that of the program file created in step 2.
Program file for add destination	InitialExecution	Set the program file created in step 2.

4. Open the ST program before process movement and cut the processes that set the initial values for each label in first to fourth rows.

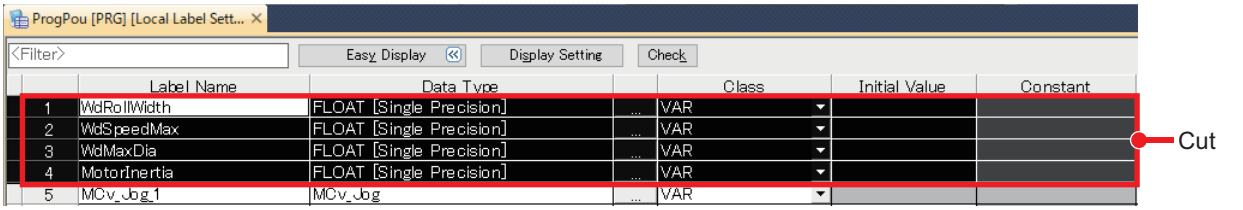


5. Open the initial execution type program created in step 3, and paste the processes cut in step 4.



6. Open the local label editor of the ST program before process movement, and cut the local labels used in the initial execution type program (No. 1 to 4 in the example).

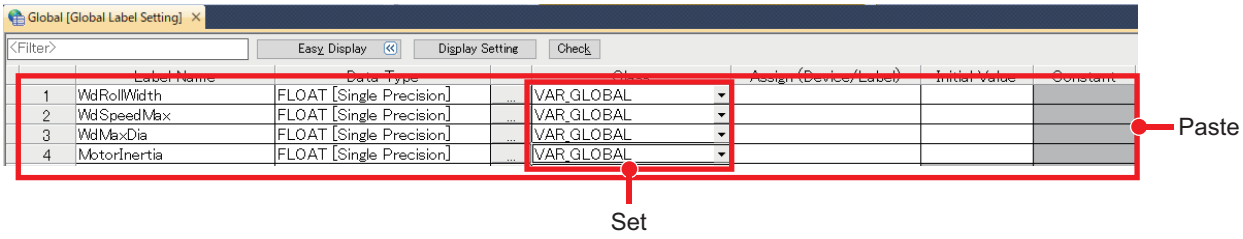
When selecting the rows to be cut, click the row numbers in the editor so that the entire rows are selected.



Point

Otherwise, contents hidden in the label editor are not copied.

7. Open the global label, paste the local labels cut in step 6, and set the classes.



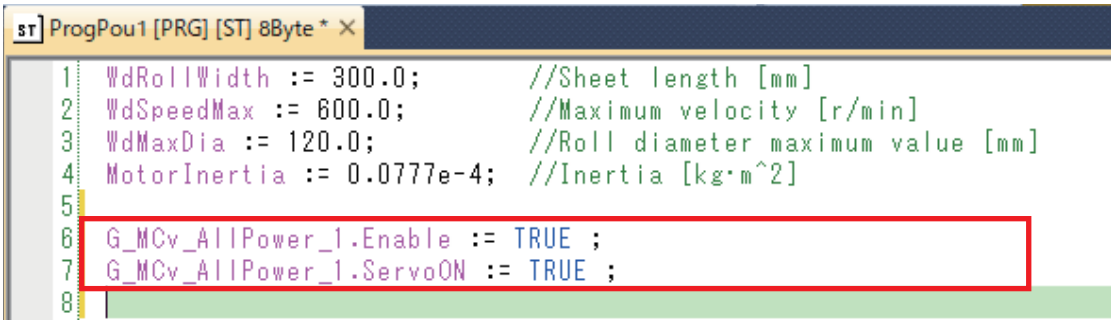
Point

In this example, the cut local labels are pasted into a global label. However, the paste destination is not necessarily a global label.

Check the label reference status and select a local label or global label as the paste destination.

8. Open the initial execution type program and write the processes for specifying "TRUE" in the input variables "Enable" and "ServoON" of the FB instance "G_MCv_AllPower_1".

- G_MCv_AllPower_1.Enable := TRUE;
- G_MCv_AllPower_1.ServoON := TRUE;

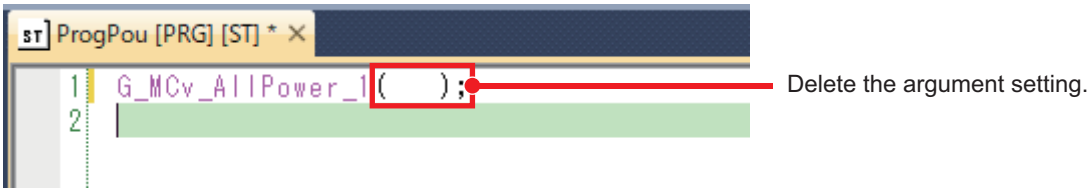


Point

- The instance of the motion control FB must be a global label since it is used in both the program before process movement and the initial execution type program. If it is a local label, change it to a global label.
- Inputting a "." (period) after the instance name of the motion control FB displays a list of label name candidates, and selecting a candidate automatically inputs the label name. If no candidate is displayed, it can be set by selecting the following items from the menu: [Tool] ⇨ [Options] ⇨ "Edit" ⇨ "Instruction/Device/Label Candidacy Display" ⇨ "Operational Setting" ⇨ "Instruction/Device/Label name Prediction" For details, refer to the following manual.

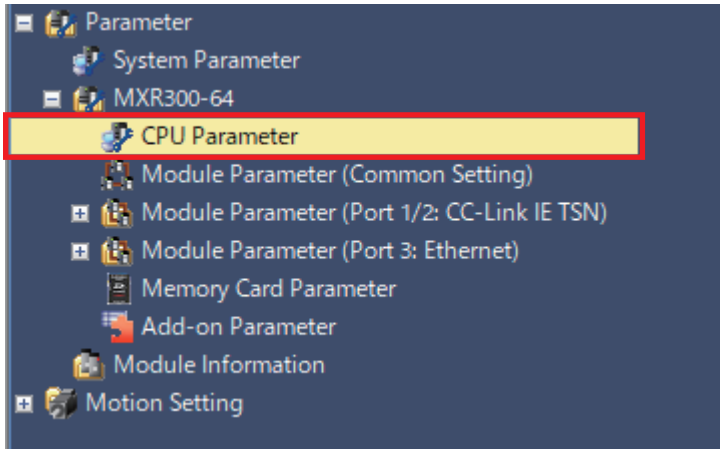
GX Works3 Operating Manual

9. Open the ST program before process movement and delete the argument setting of the FB instance "G_MCv_AllPower_1".



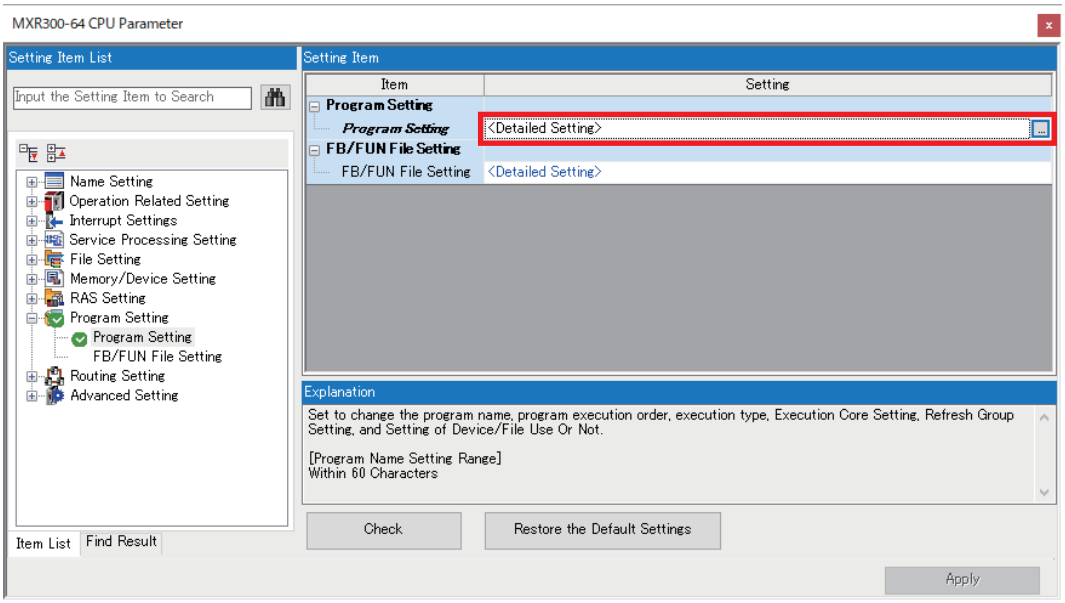
10. Open CPU parameters.

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ Double-click [CPU Parameters]

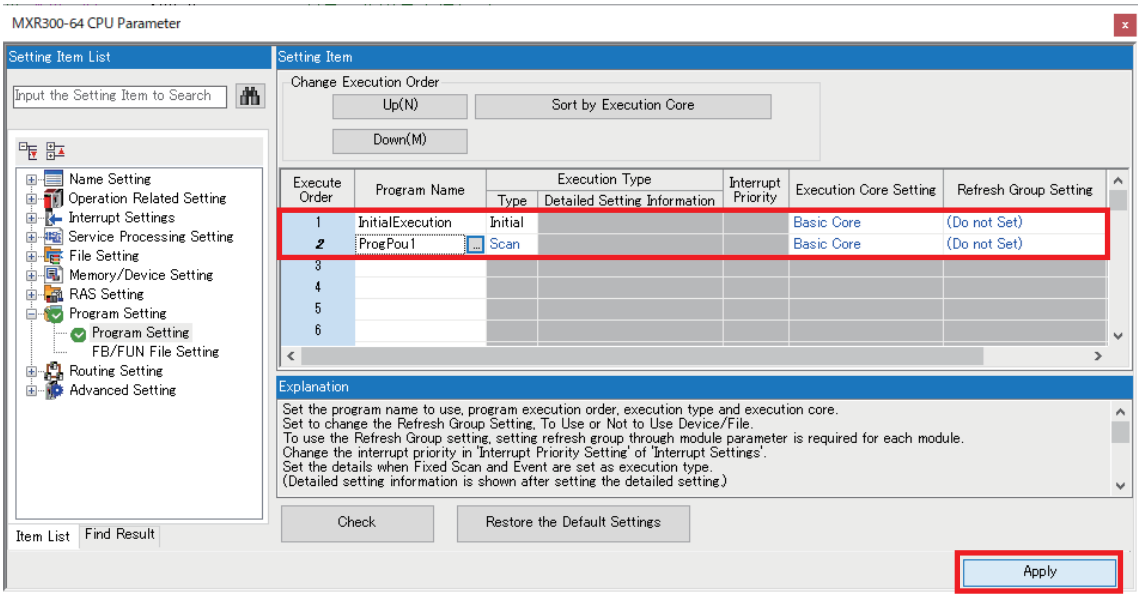


11. To set the program execution order, open the program setting.

[Program Setting] ⇨ Double-click <Detailed Setting> of [Program Setting]



12. Sort the programs so that the initial execution type program is executed first, and then click the [Apply] button to apply the setting.



4.5 Omitting Arguments and Utilizing Direct References

Overview

Since a motion control FB within an ST program (motion control program) has numerous arguments, assigning other labels to I/O arguments can increase the load on assignment/reference processing.
Therefore, by using the following methods to avoid using other labels, the generation of assignment/reference processing can be reduced, and the scan time can be reduced.

- Directly inputting values to I/O arguments without using labels
- Omitting assignment of values/labels to unnecessary FB arguments
- Directly referencing FB labels in condition evaluations within IF statements
- Directly assigning FB arguments when assigning FB values to other labels

Execution procedure

Using the following program as an example, this section describes the procedure for modifying I/O argument settings for the motion control FB instance "MCv_Jog_1".

ProgPou [PRG] [ST]

```
1  bLabel6[0] := TRUE ;
2
3  //Axis 1: JOG operation execution
4  MCv_Jog_1(
5      Axis:= Axis0001.AxisRef ,
6      JogForward:= bLabel6[0] ,
7      JogBackward:= bLabel6[1] ,
8  //  Velocity:= ?LREAL? ,
9  //  Acceleration:= ?LREAL? ,
10 //  Deceleration:= ?LREAL? ,
11 //  Jerk:= ?LREAL? ,
12 //  Options:= ?DWORD? ,
13     Done=> bLabel7[0] ,
14     Busy=> bLabel7[1] ,
15     Active=> bLabel7[2] ,
16     CommandAborted=> bLabel7[3] ,
17     Error=> bLabel7[4] ,
18     ErrorID=> uLabel8
19     );
20
21 //Axis 1: Error occurrence (bLabel7[4] turns on.)
22 IF  bLabel7[4]  THEN
23     //Store the error ID (uLabel8) in uLabel11.
24     uLabel11:= uLabel8 ;
25 END_IF;
26
```

The label "bLabel6[0]" is assigned to the argument "JogForward".
(direct specification available)

Unused arguments are remaining.
(omittable)

Assigning labels is not required because these arguments are not used.
(omittable)

The labels "bLabel7[4]" and "uLabel8" are assigned to the arguments "Error" and "ErrorID".
(direct specification available)

Operating procedure

1. Without using the label "bLabel6[0]", directly specify "TRUE" in the input variable "JogForward" of the FB instance "MCv_Jog_1".

Change the label "bLabel6[0]" in the first row to "MCv_Jog_1.JogForward". In addition, comment out or delete the argument "JogForward" setting of "MCv_Jog_1".

ProgPou [PRG] [ST] *

```
1  MCv_Jog_1.JogForward := TRUE ;
2
3  //Axis 1: JOG operation execution
4  MCv_Jog_1(
5      Axis:= Axis0001.AxisRef ,
6  //  JogForward:= bLabel6[0] ,
7      JogBackward:= bLabel6[1] ,
8  //  Velocity:= ?LREAL? ,
9  //  Acceleration:= ?LREAL? ,
10 //  Deceleration:= ?LREAL? ,
11 //  Jerk:= ?LREAL? ,
12 //  Options:= ?DWORD? ,
13     Done=> bLabel7[0] ,
14     Busy=> bLabel7[1] ,
15     Active=> bLabel7[2] ,
16     CommandAborted=> bLabel7[3] ,
17     Error=> bLabel7[4] ,
18     ErrorID=> uLabel8
19     );
```

Change to "MCv_Jog_1.JogForward".

Comment out

Point

Inputting a "." (period) after the instance name of the motion control FB displays a list of label name candidates, and selecting a candidate automatically inputs the label name.

If no candidate is displayed, it can be set by selecting the following items from the menu: [Tool] ⇨ [Options] ⇨ "Edit" ⇨ "Instruction/Device/Label Candidacy Display" ⇨ "Operational Setting" ⇨ "Instruction/Device/Label name Prediction". For details, refer to the following manual.

GX Works3 Operating Manual

2. Comment out or delete the unused argument settings in the 8th to 16th rows.

ProgPou [PRG] [ST] *

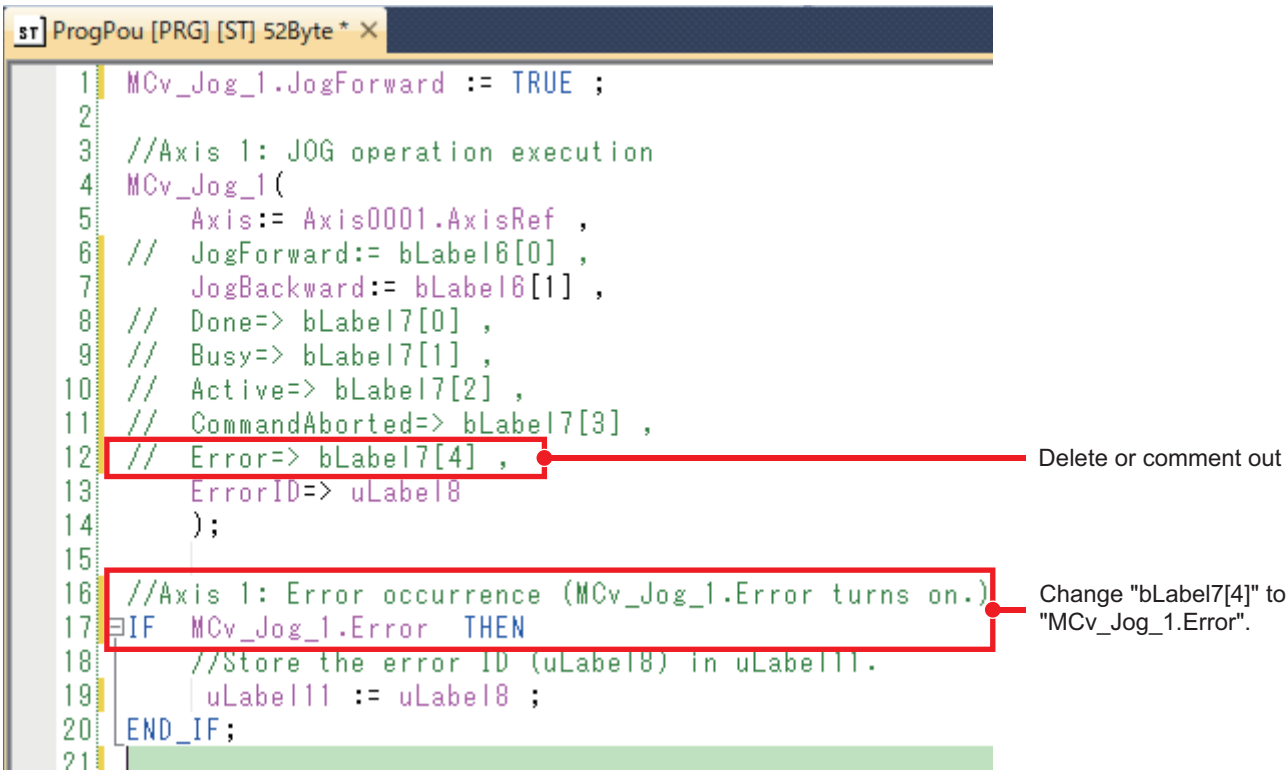
```
1  MCv_Jog_1.JogForward := TRUE ;
2
3  //Axis 1: JOG operation execution
4  MCv_Jog_1(
5      Axis:= Axis0001.AxisRef ,
6  //  JogForward:= bLabel6[0] ,
7      JogBackward:= bLabel6[1] ,
8
9
10
11
12
13 //  Done=> bLabel7[0] ,
14 //  Busy=> bLabel7[1] ,
15 //  Active=> bLabel7[2] ,
16 //  CommandAborted=> bLabel7[3] ,
17     Error=> bLabel7[4] ,
18     ErrorID=> uLabel8
19     );
```

Delete or comment out

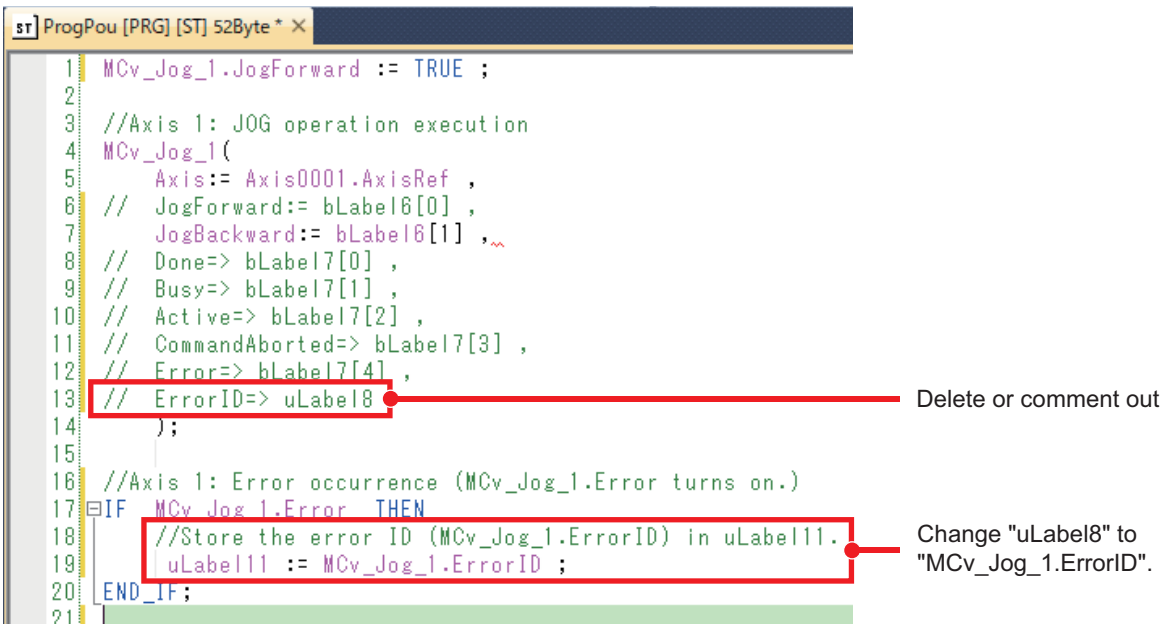
Point

You can omit variables that are not used or not changed from the initial values.

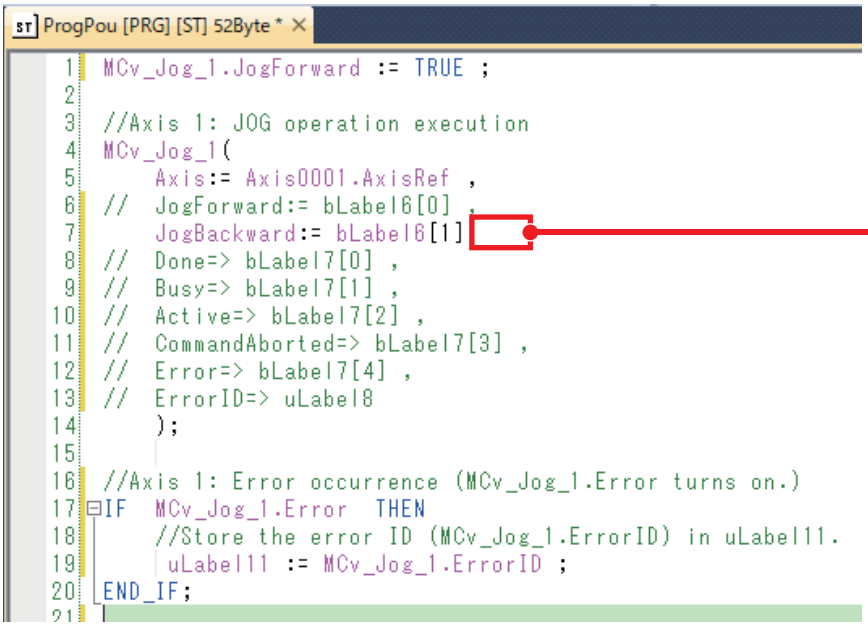
3. Without using the label "bLabel7[4]", directly specify the input variable "Error" of the FB instance "MCv_Jog_1". Change the label "bLabel7[4]" in the 16th and 17th rows to "MCv_Jog_1.Error". In addition, comment out or delete the argument "Error" setting of "MCv_Jog_1".



4. Without using the label "uLabel8", directly specify the input variable "ErrorID" of the FB instance "MCv_Jog_1". Change the label "uLabel8" in the 18th and 19th rows to "MCv_Jog_1.ErrorID". In addition, comment out or delete the argument "ErrorID" setting of "MCv_Jog_1".



5. According to commenting out and deletion in steps 1 to 4, modify the program so that there is no error in syntax.
In this example, an unnecessary comma is deleted from the FB argument setting.



Point

If there are any labels that are no longer used in the entire project due to the direct specification of I/O arguments or the removal of unnecessary label assignments, delete the unused labels from the label editor.

4.6 Dividing a Circuit Block with a Large Number of Instructions

Overview

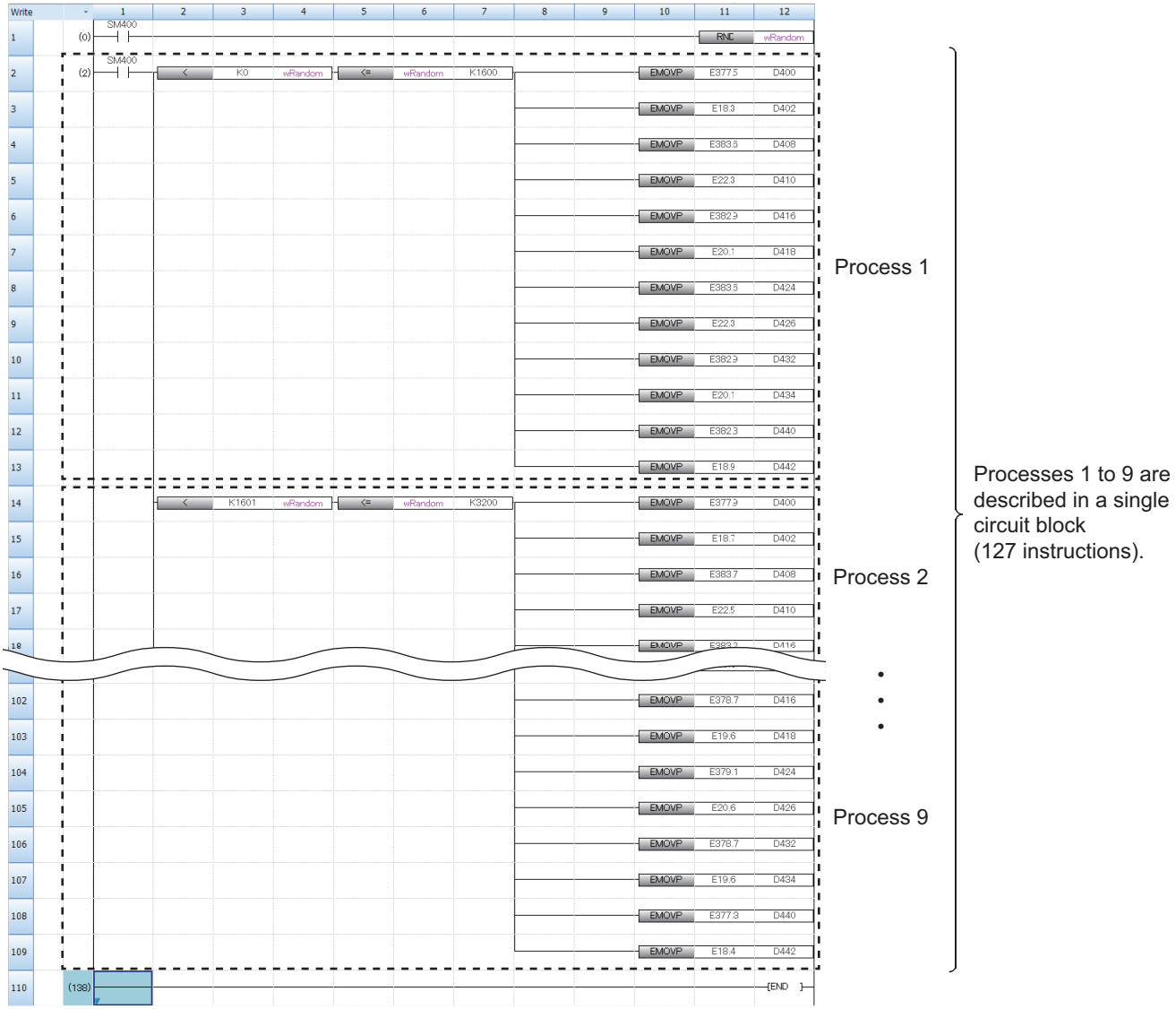
If a single circuit block contains more than 100 instructions, dividing the instructions into multiple circuit blocks so that each circuit block contains 100 or less instructions can reduce the instruction processing time. This can reduce the scan time.
Dividing a circuit block has the following benefit.

Improvement of program readability

By dividing a circuit block into multiple circuit blocks, improving the program readability.

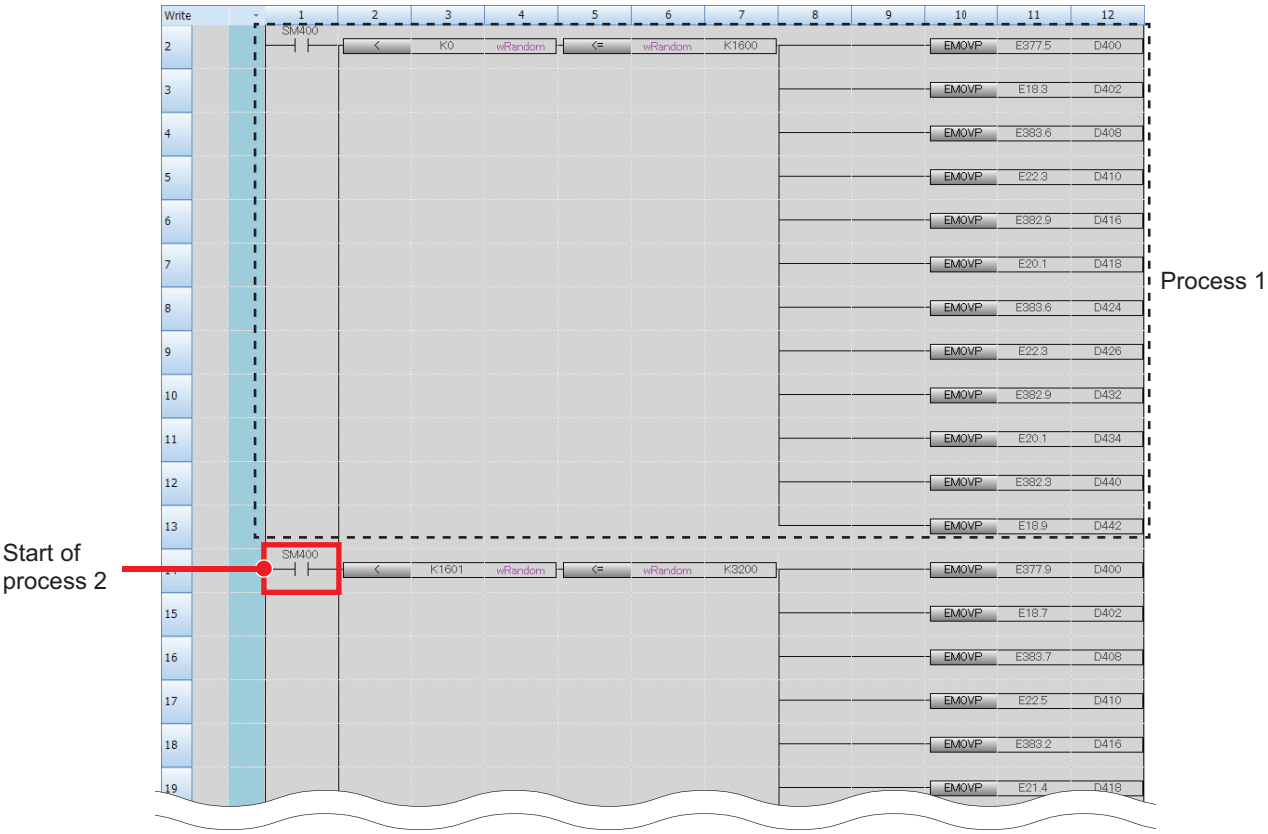
Execution procedure

Using the following program as an example, this section describes the procedure for dividing a circuit block which contains instructions for processes 1 to 9 (127 instructions) into multiple circuit blocks by process.

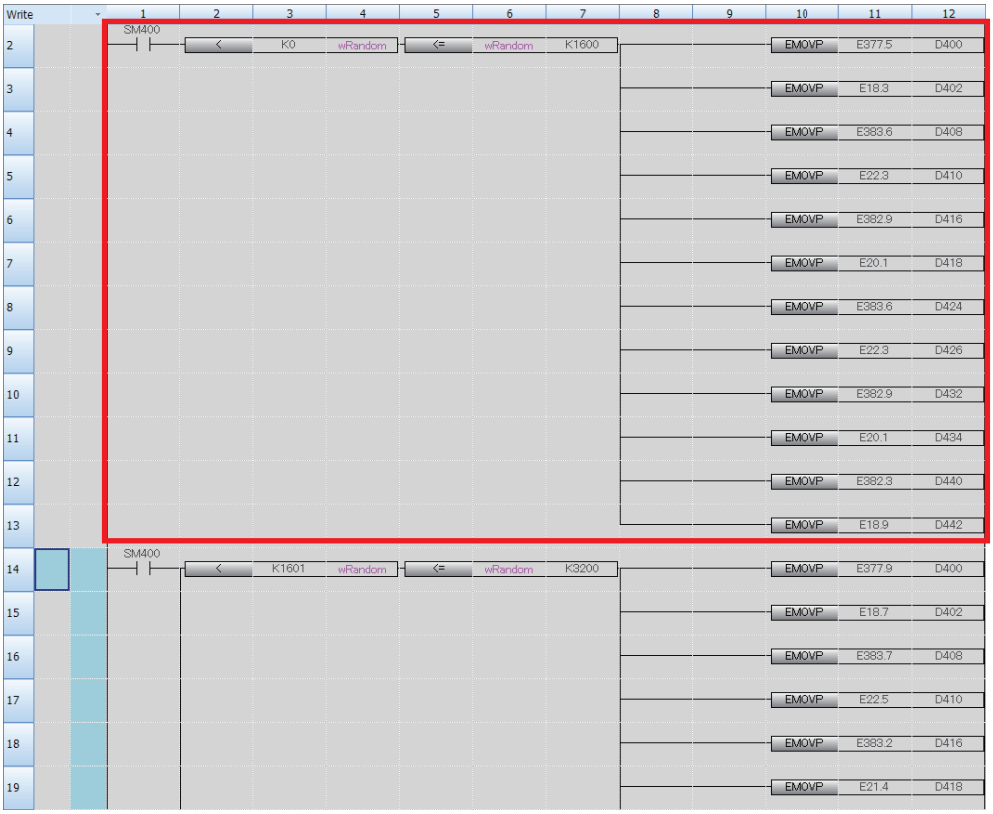


Operating procedure

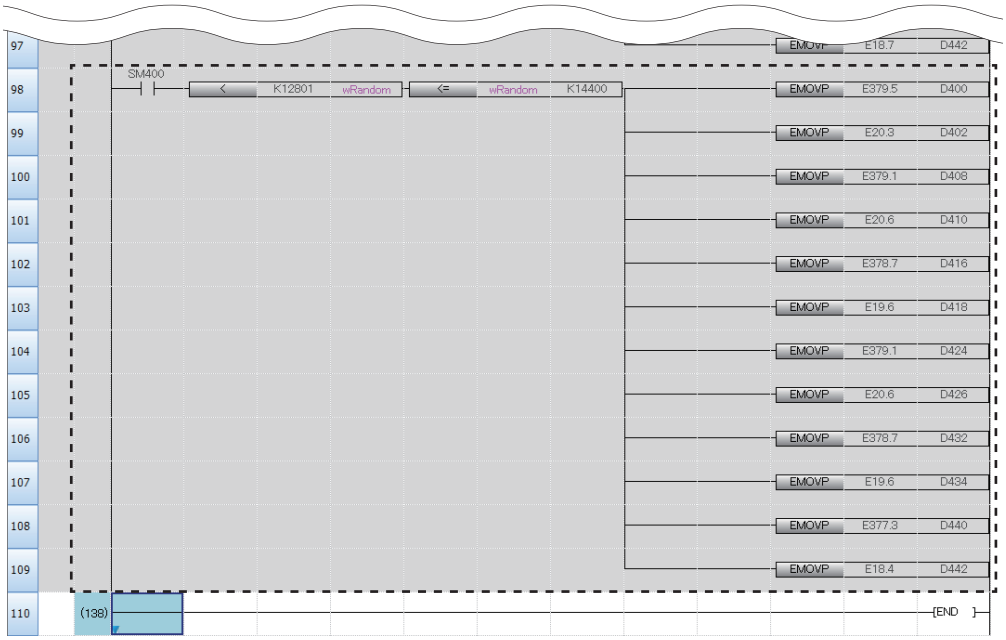
1. Open the program, and add the contact "SM400", which is the same as the first row of process 1, at the position from which the circuit block is divided (the beginning of process 2).



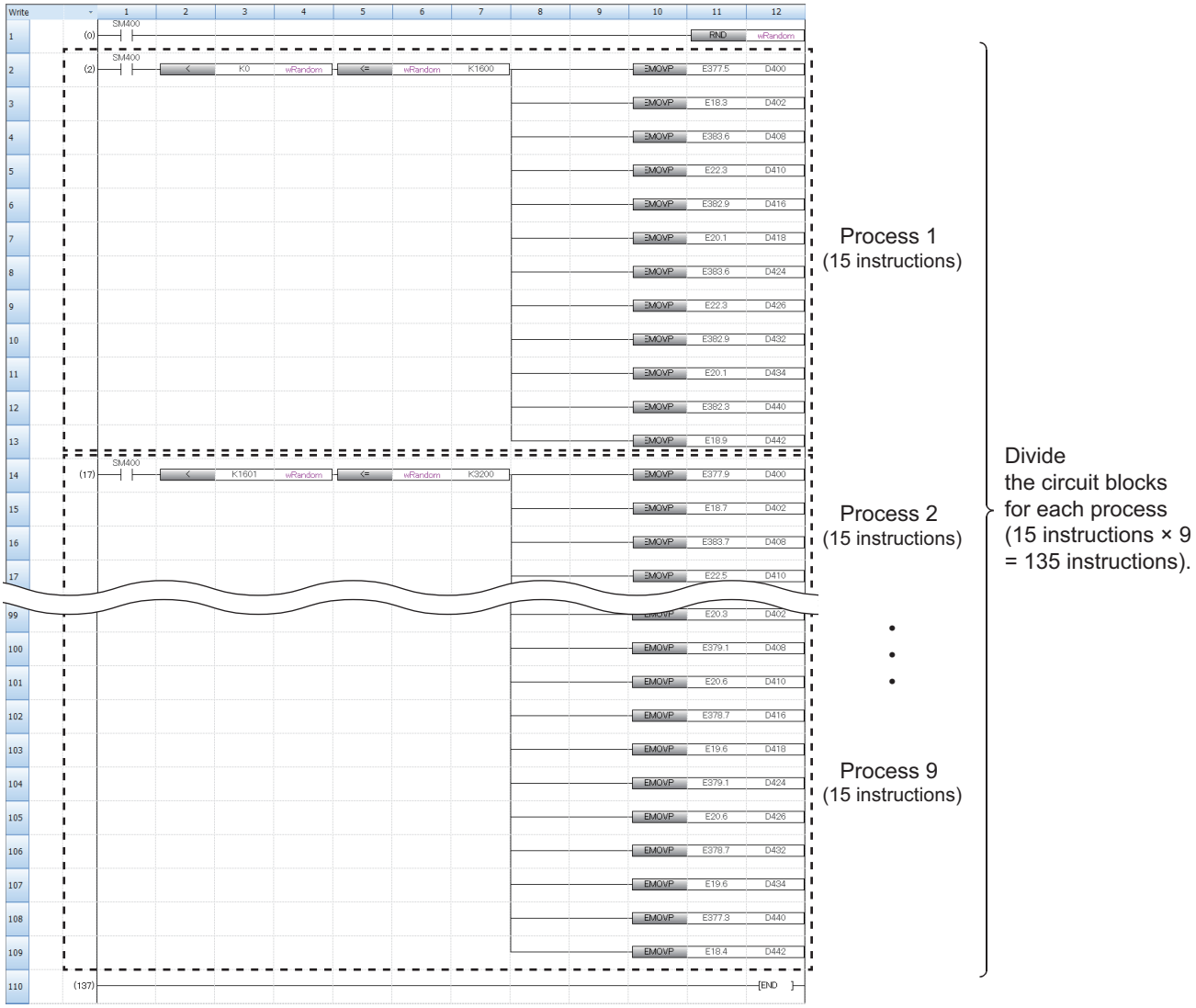
2. Modify the line of the process 1 contact.



3. Repeat steps 1 and 2, and divide the circuit blocks by process up to process 9.



After steps 1 to 3 are performed, the program will consist of nine blocks, each containing one process (15 instructions).



4.7 Utilizing Label Initial Values

Overview

For labels or FB initial values that do not require constant modification in the program, setting the initial values in the label editor instead of specifying them within the program can reduce the scan time.
Utilizing label initial values has the following benefit.

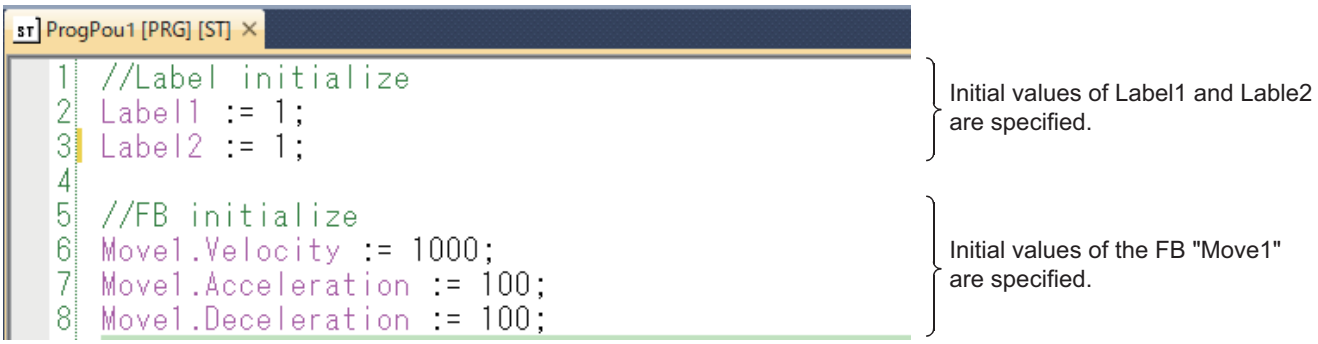
Improvement of program readability

The processes for specifying initial values within the program cab be omitted, improving the program readability.

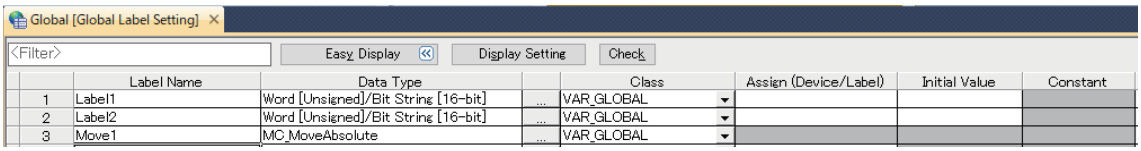
Execution procedure

Using the following existing initial execution type program as an example, this section describes the procedure for setting the initial values of global labels or FBs, which are currently specified within the program, in the label editor.

- Initial execution type program



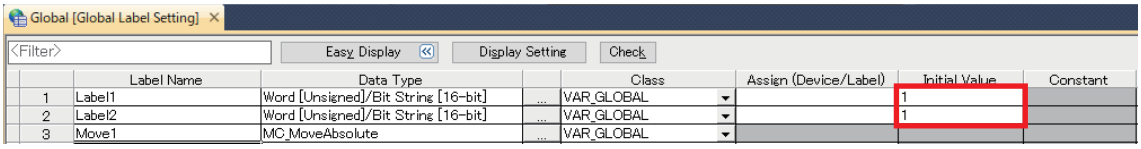
- Global label



Operating procedure

1. Open the global label editor and set the same initial values as those specified in the initial execution type program in "Label1" and "Label2".

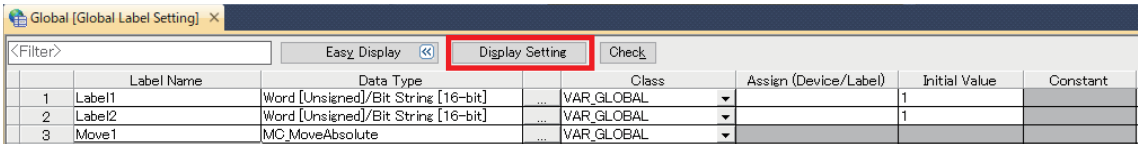
- Label1: 1
- Label2: 1



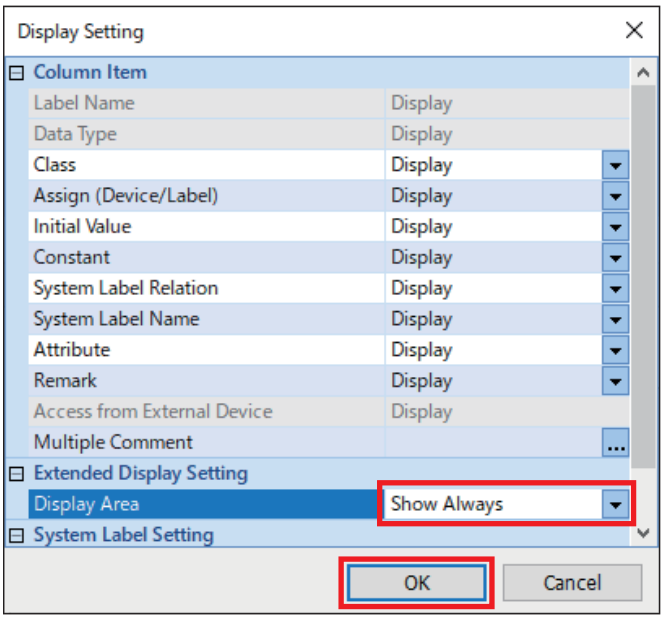
Point

If the "Initial Value" column is not displayed, click the [Show Details] button to display it.

2. Click the [Display Setting] button.

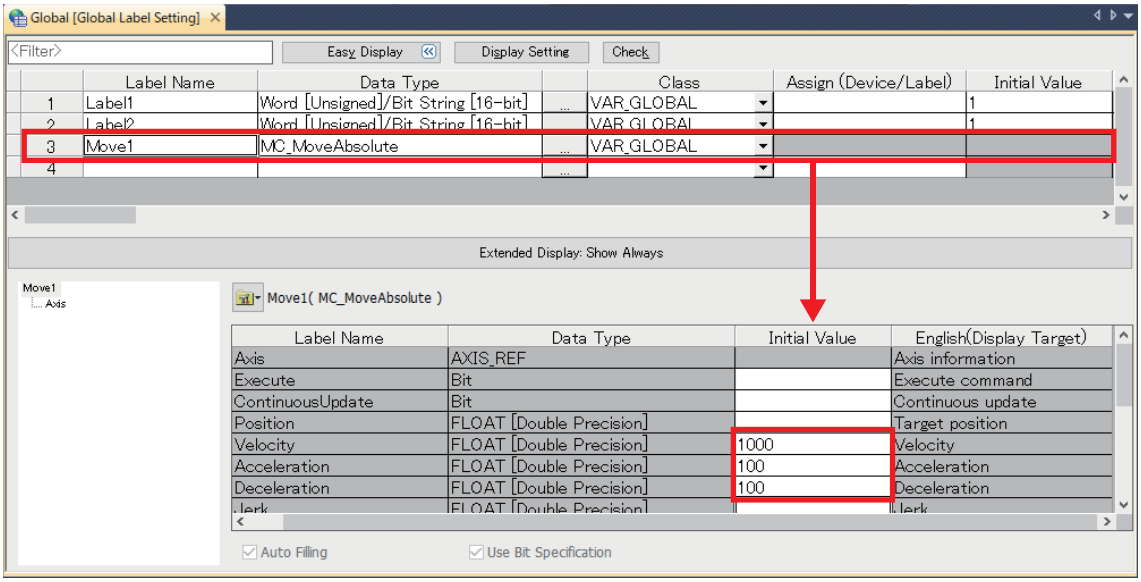


3. Change [Display Area] under [Extended Display Setting] to "Show Always", and then click the [OK] button.

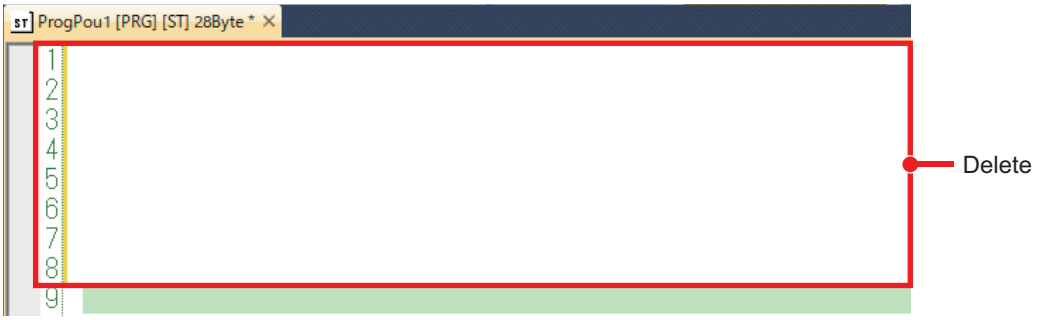


4. Select the global label "Move1" and set the same initial values as those specified in the initial execution type program in the "Initial Value" column in the extended display area.

- Velocity: 1000
- Acceleration: 100
- Deceleration: 100



5. Open the initial execution type program in which the initial values are specified and delete the processes that specify the initial values.



5 OPTIMIZATION OF MOTION CONTROL

Motion control can be optimized by accelerating specific axes in multi-axis control or by synchronizing user programs with motion control.
This chapter describes techniques for optimizing motion control.

List of techniques

The following table lists the techniques described in this chapter.

Item	Description	Reference
Increasing speed of specific axes using multiple cycle setting	Using the multiple cycle setting function, accelerate only the axes that require high-speed control in multi-axis control.	☞ Page 79 Increasing Speed of Specific Axes Using Multiple Cycle Setting
Synchronizing user programs and motion operations	Synchronize the cycles of user programs and motion operations in multiple cores to synchronize the update timing of variables.	☞ Page 84 Synchronizing User Programs and Motion Operations

5.1 Increasing Speed of Specific Axes Using Multiple Cycle Setting

Overview

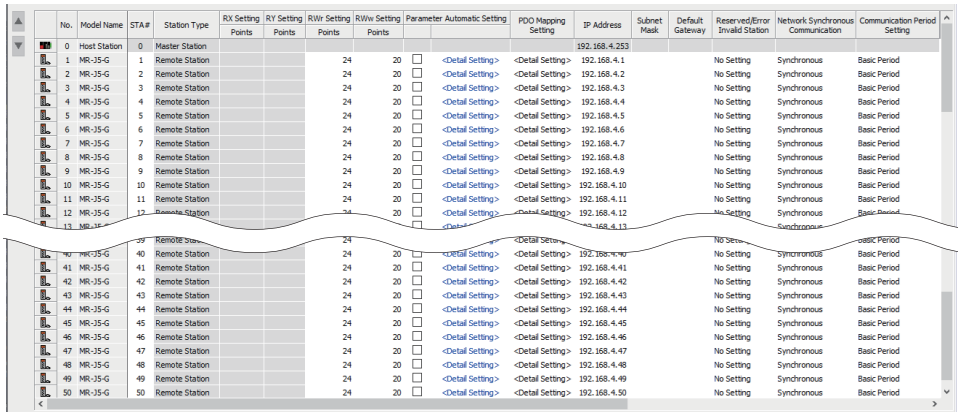
If certain axes require high-speed control in a motion program for performing multi-axis control, the multiple cycle setting function can be used to set the cycle for each axis. This allows the axes that require high-speed control to be controlled with shorter operation cycles. Additionally, in a multi-axis configuration, the overall control load can be adjusted.

Point

This technique is more effective when combined with the techniques for accelerating CC-Link IE TSN. For accelerating CC-Link IE TSN, refer to the following.
☞ Page 95 INCREASING SPEED OF CC-Link IE TSN COMMUNICATION

Execution procedure

Using the following axis configuration as an example, this section describes the procedure for accelerating specific axes with the multiple cycle setting function.



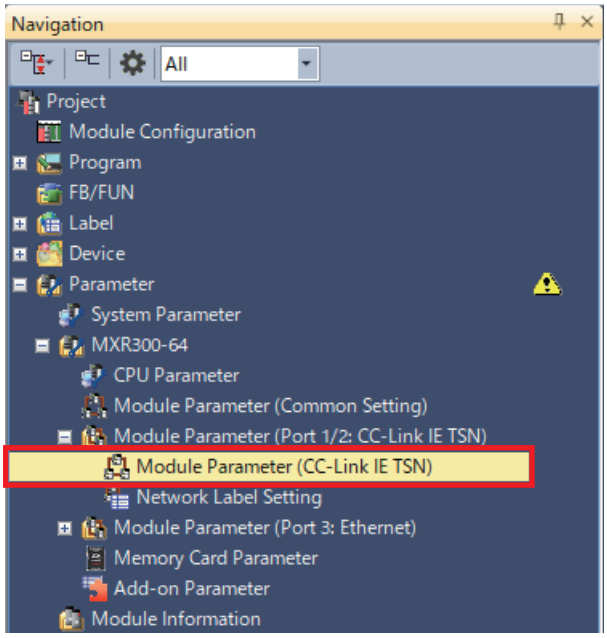
In this example, the communication cycles of the MR-J5-G are set as follows.

Station No.	Communication cycle
1 to 5	Basic Period
6 to 20	Normal-speed
21 to 50	Low-speed

Operating procedure

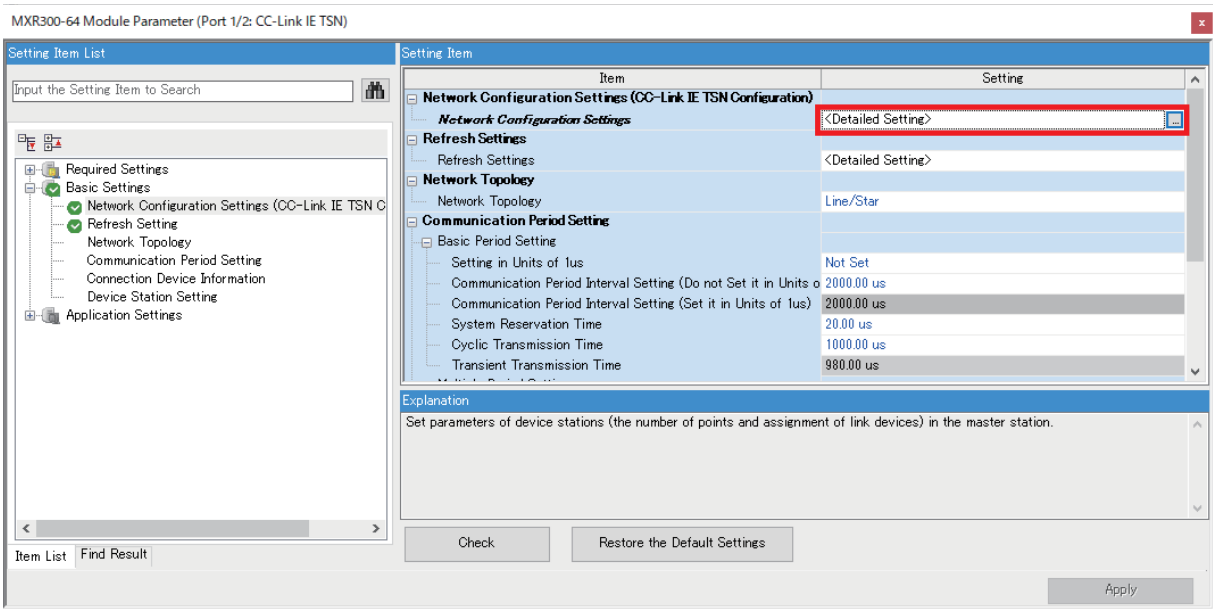
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]

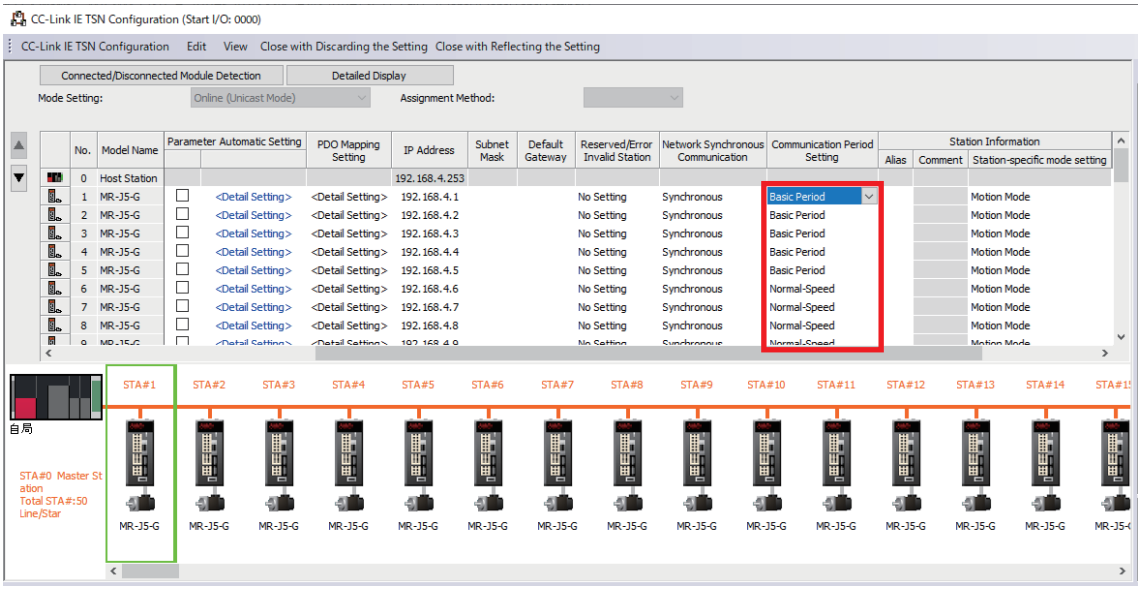


2. In the basic settings, configure the network configuration settings.

[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



3. Set the communication cycle of each axis in the "Communication Period Setting" column in the "CC-Link IE TSN Configuration" window.



Station No.	Communication cycle
1 to 5	Basic Period
6 to 20	Normal-speed
21 to 50	Low-speed

Point

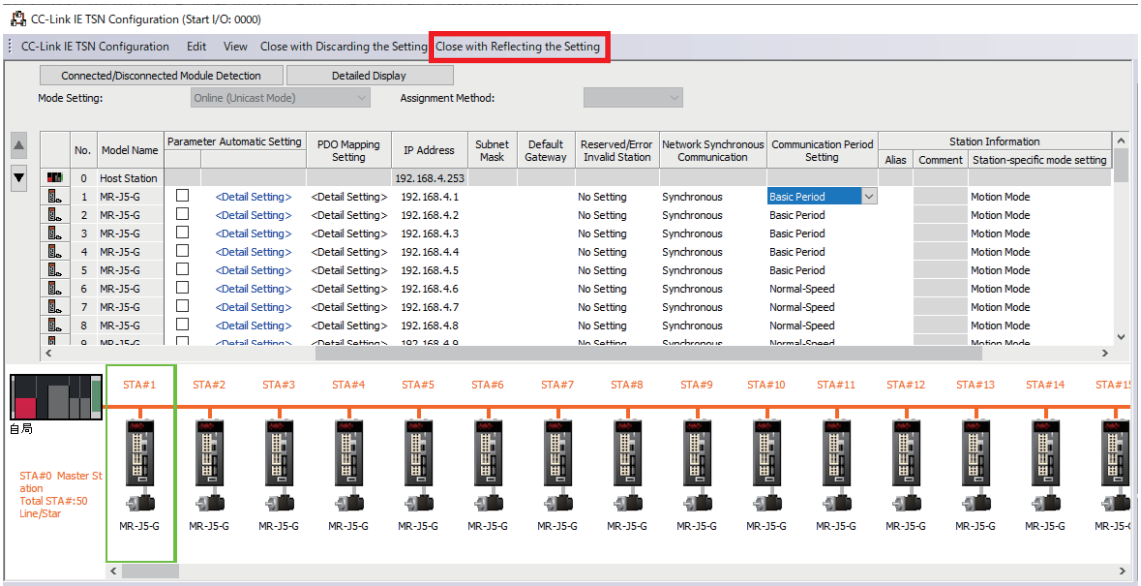
The fixed scan interval for the basic period and multipliers for the normal-speed and low-speed can be set in [Communication Period Setting] displayed by selecting the following items: [Module Parameter (CC-Link IE TSN)] ⇨ [Basic Settings]

Precautions

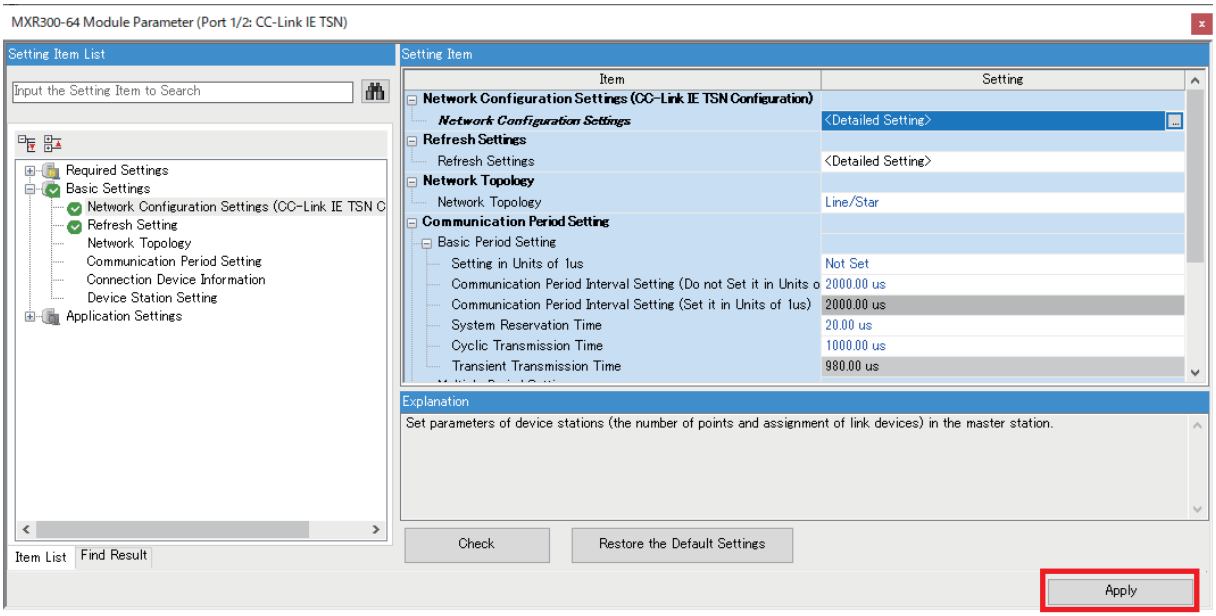
Depending on the communication speeds of the master station and remote stations, specific communication cycles cannot be set. For the communication speed settings for the master station and remote stations, refer to the following manual.

📖MELSEC MX Controller (MX-R Model) User's Manual

4. Click the [Close with Reflecting the Setting] button.

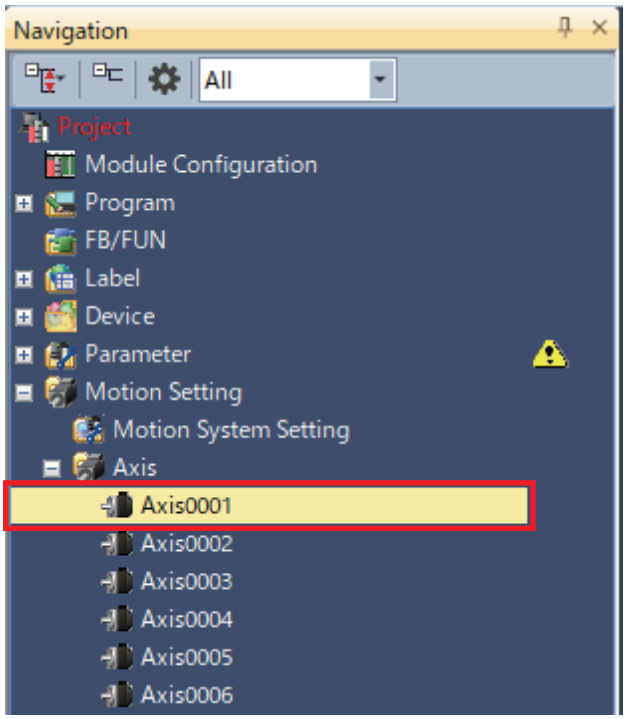


5. Click the [Apply] button to apply the setting.



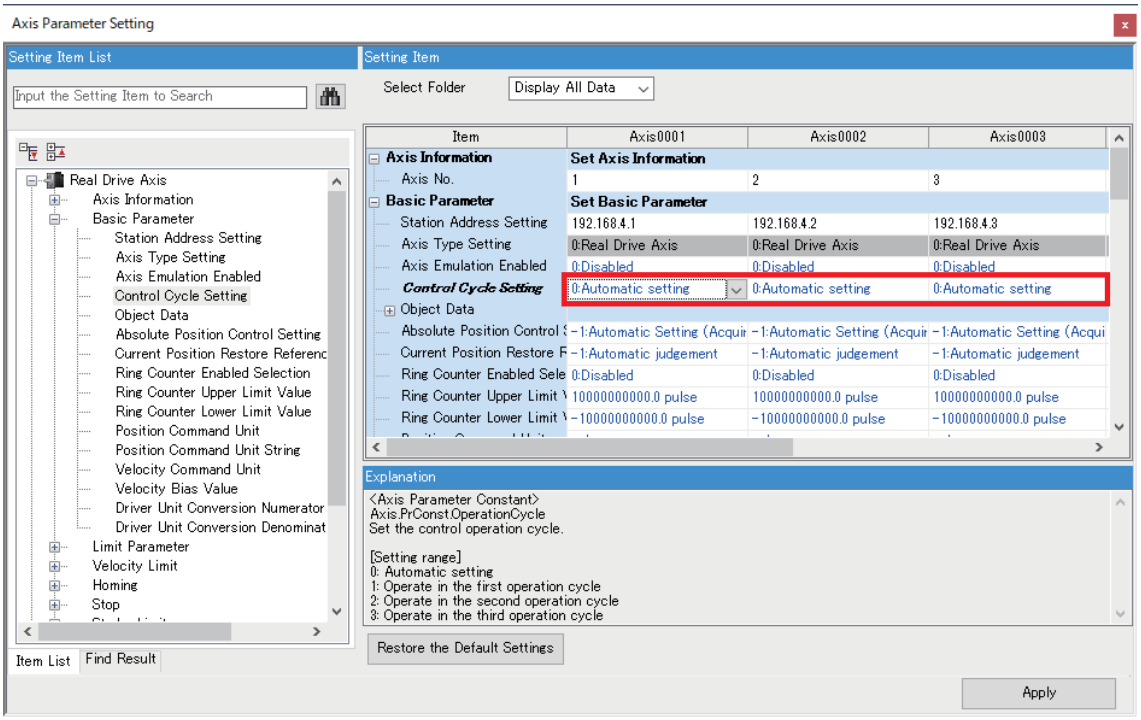
6. Set parameters for each axis.

Navigation window ⇨ [Motion Setting] ⇨ [Axis] ⇨ Double-click any axis

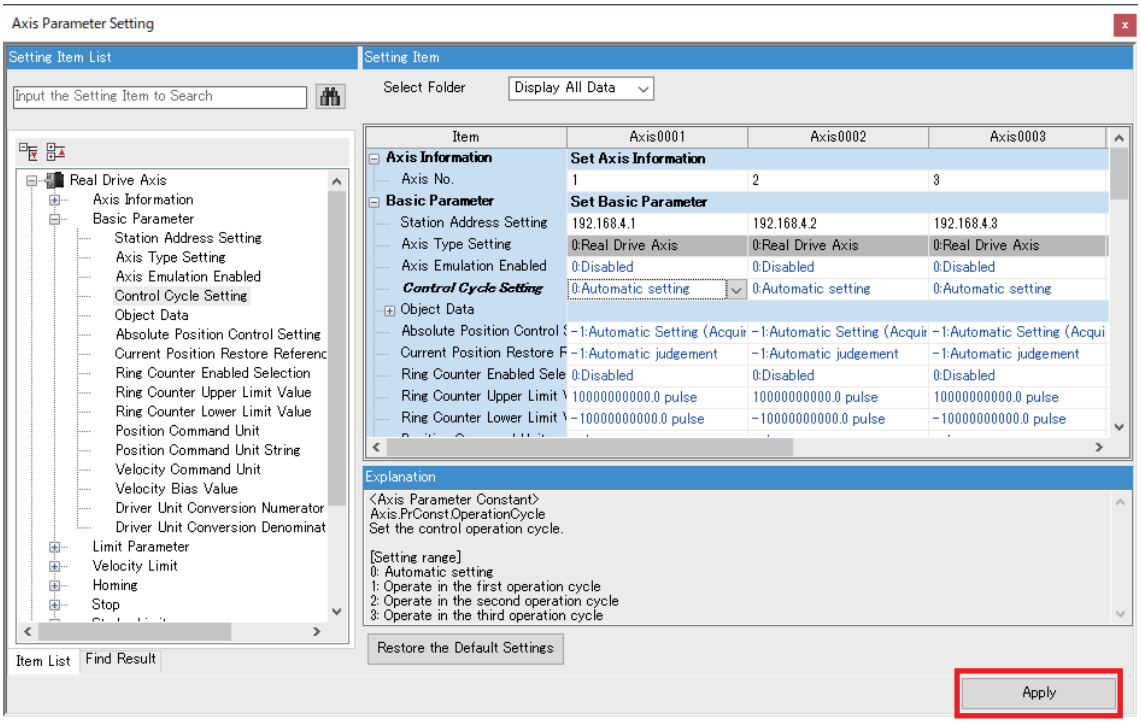


7. Set the control cycle setting of each axis to "0: Automatic setting" (default).

Axis type to be set ⇨ [Basic Parameter] ⇨ [Control Cycle Setting]



8. If you change any setting in step 7, click the [Apply] button to apply the setting.



Precautions

When performing synchronization control between axes with different cycles, there are some restrictions. For details, refer to the following manual.

MELSEC MX Controller (MX-R Model) User's Manual

5.2 Synchronizing User Programs and Motion Operations

Overview

The controller executes user programs and motion operations in parallel on multiple cores. Therefore, accessing variables used in motion operations (such as AxisName.Md.SetPosition updated by the motion function part) from user programs with execution cycles different from the motion operation cycle may lead to unstable variable update timing.
When the variable update timing is important, create a program synchronized with the motion operation cycle to synchronize the variables used in the user program and motion operations.

Program synchronized with the motion operation cycle

By using the start of the network communication cycle as the reference point, the program execution cycle can always be synchronized with the motion operation cycle in the execution order of program → motion operation.
The following table lists the combinations of network communication cycles and programs to be synchronized with the motion operation cycle.

Motion operation cycle	Network communication cycle to be synchronized	Program to be synchronized		
		Execution type	Trigger type	Execution cycle
1st operation cycle	Basic cycle	Event	Network communication cycle synchronization	Basic cycle
2nd operation cycle	Normal-speed cycle			Normal-speed cycle
3rd operation cycle	Low-speed cycle			Low-speed cycle

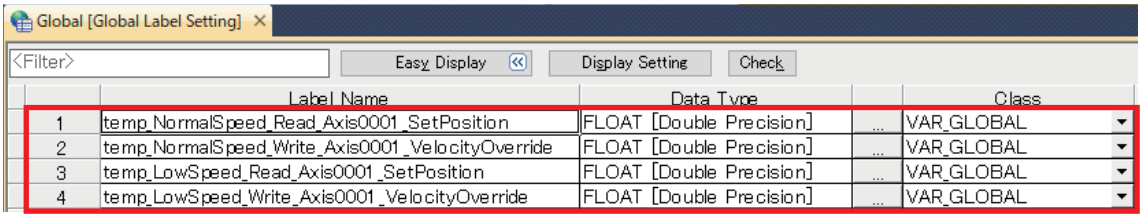
Execution procedure

Using the variables "AxisName.Md.SetPosition" (Command current position) and "AxisName.Cd.VelocityOverride" (Velocity override factor) as examples, this section describes the procedure for synchronizing the execution timings of these variables with the program.

Operating procedure

1. Open a global label editor and define global labels for saving the variables "AxisName.Md.SetPosition" and "AxisName.Cd.VelocityOverride" (hereafter referred to as temporary labels).

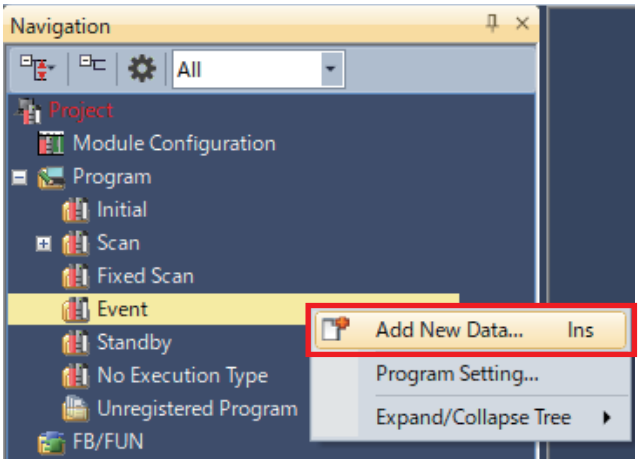
In this step, temporary labels are defined as follows.



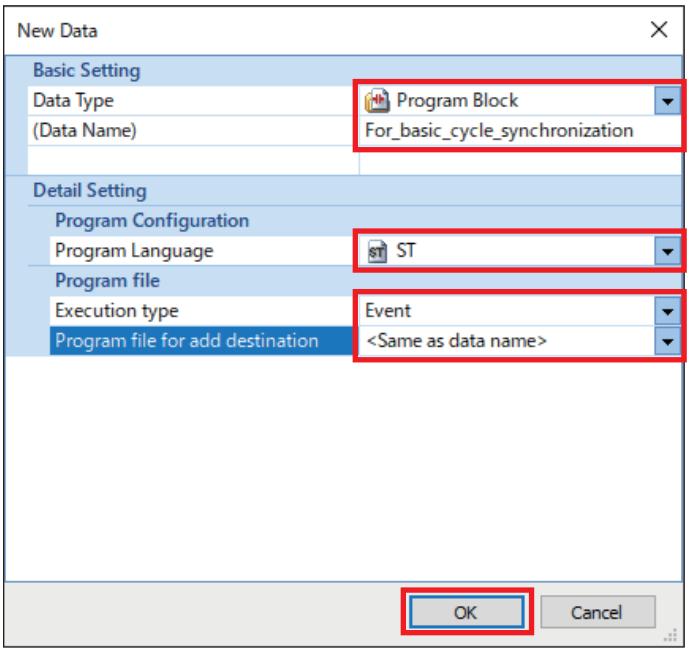
No.	Label name	Data type	Class	Remarks
1	temp_NormalSpeed_Read_Axis0001_SetPosition	FLOAT [Double Precision]	VAR_GLOBAL	A command current position temporary label for accessing the normal-speed cycle
2	temp_NormalSpeed_Write_Axis0001_VelocityOverride	FLOAT [Double Precision]	VAR_GLOBAL	A velocity override temporary label for accessing the normal-speed cycle
3	temp_LowSpeed_Read_Axis0001_SetPosition	FLOAT [Double Precision]	VAR_GLOBAL	A command current position temporary label for accessing the low-speed cycle
4	temp_LowSpeed_Write_Axis0001_VelocityOverride	FLOAT [Double Precision]	VAR_GLOBAL	A velocity override temporary label for accessing the low-speed cycle

2. Newly create an event execution type program block whose basic cycle is to be synchronized with the motion operation.

Navigation window ⇨ [Program] ⇨ Right-click [Event] ⇨ [Add New Data]



3. Configure the following settings, and then click the [OK] button.

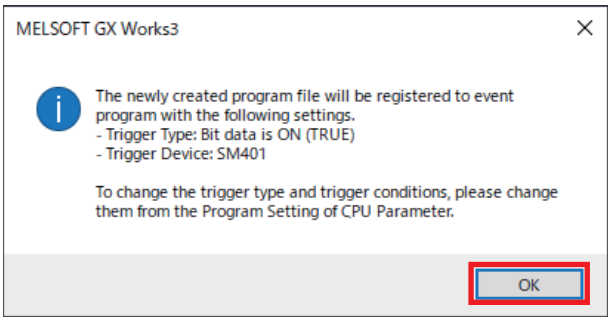


Item	Setting
Data Type	Program Block
(Data Name)	For_basic_cycle_synchronization
Program Language	ST
Execution type	Event
Program file for add destination	<Same as data name>

Point

To match the program file name and program block name when creating a new program file and program block, setting "Program file for add destination" to "<Same as data name>" when creating a program block creates a program file with the same name.
This can omit the creation of a new program file.

4. Click the [OK] button.



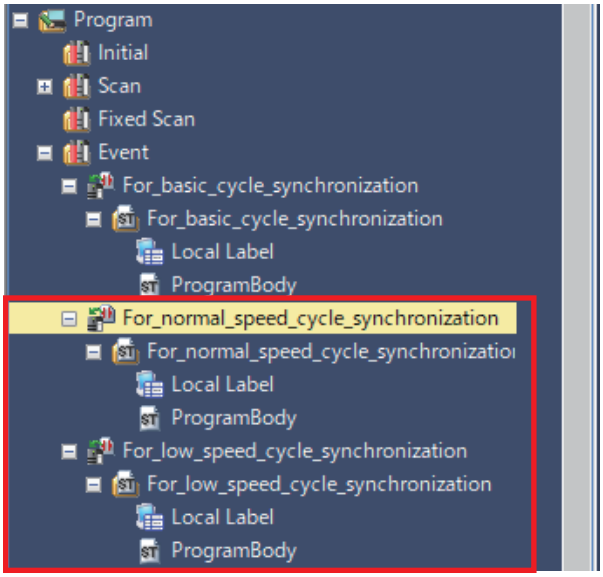
5. Open the program body of the program block created in step 3 and write the following processes for synchronizing the variables with the basic cycle.

```
For_basic_cycle_synchronizatio...
 1  //Basic cycle  Saving processing for reading/writing
 2
 3  //Save the variable in the temporary label of the medium speed (cycle time: four times the basic cycle) cycle.
 4  IF MotionSystem.Md.SystemBaseCycle_Counter MOD 4 = 1 THEN
 5      //Read
 6      temp_NormalSpeed_Read_Axis0001_SetPosition := Axis0001.Md.SetPosition;
 7      //Write
 8      Axis0001.Cd.VelocityOverride := temp_NormalSpeed_Write_Axis0001_VelocityOverride;
 9  END_IF;
10
11  //Save the variable in the temporary label of the low speed (cycle time: 16 times the basic cycle) cycle.
12  IF MotionSystem.Md.SystemBaseCycle_Counter MOD 16  = 1 THEN
13      //Read
14      temp_LowSpeed_Read_Axis0001_SetPosition := Axis0001.Md.SetPosition;
15      //Write
16      Axis0001.Cd.VelocityOverride := temp_LowSpeed_Write_Axis0001_VelocityOverride;
17  END_IF;
```

Row	Processing
3rd to 9th	When accessing the axis label of an axis controlled in the 1st operation cycle from a program with a normal-speed cycle (4 times the basic cycle), execute the saving and restoration processes in cycles in which the remainder obtained by dividing the system basic cycle counter*1 by 4 is 1. • At reading: Save the value of the variable "AxisName.Md.SetPosition" in the temporary label. • At writing: Restore the value of the temporary label in the variable "AxisName.Cd.VelocityOverride".
11th to 17th	When accessing the axis label of an axis controlled in the 1st operation cycle from a program with a low-speed cycle (16 times the basic cycle), execute the saving and restoration processes in cycles in which the remainder obtained by dividing the system basic cycle counter*1 by 16 is 1. • At reading: Save the value of the variable "AxisName.Md.SetPosition" in the temporary label. • At writing: Restore the value of the temporary label in the variable "AxisName.Cd.VelocityOverride".

*1 The system basic cycle counter (System.Md.SystemBaseCycle_Counter) is a counter that increases its value by 1 at basic cycle intervals of the network and returns to 1 when the cycle exceeds the least common multiple of each motion operation cycle. This counter can be used to check the execution timing of each motion operation cycle process.
For details, refer to the following manual.
MELSEC MX Controller (MX-R Model) User's Manual

6. Referring to steps 2 to 4, create event execution type program blocks to be synchronized with the normal-speed cycle and low-speed cycle.



7. Open the event execution type program blocks to be synchronized with the normal-speed cycle and low-speed cycle created in step 6, write the following processes for setting a value in the variable "AxisName.Cd.VelocityOverride" in accordance with the value in "AxisName.Md.SetPosition".

• Event execution type program to be synchronized with the normal-speed cycle

For_normal_speed_cycle_synchr...

```
1  //If Md.SetPosition of Axis0001 that operates in the first operation cycle is 1000.0 or less
2  //Set Cd.VelocityOverride of Axis0001 that operates in the first operation cycle to 10.0.
3  IF temp_NormalSpeed_Read_Axis0001_SetPosition <= 1000.0 THEN
4      temp_NormalSpeed_Write_Axis0001_VelocityOverride := 10.0;
5  END_IF;
```

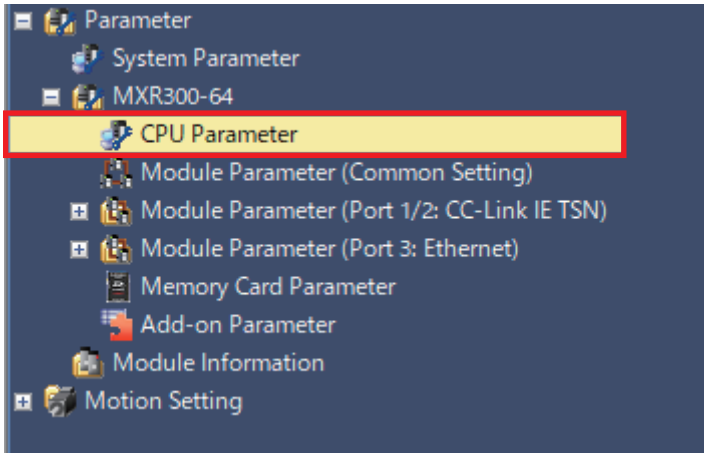
• Event execution type program to be synchronized with the low-speed cycle

For_low_speed_cycle_synchroni...

```
1  //If Md.SetPosition of Axis0001 that operates in the first operation cycle exceeds 1000.0
2  //Set Cd.VelocityOverride of Axis0001 that operates in the first operation cycle to 5.0.
3  IF temp_NormalSpeed_Read_Axis0001_SetPosition > 1000.0 THEN
4      temp_NormalSpeed_Write_Axis0001_VelocityOverride := 5.0;
5  END_IF;
```

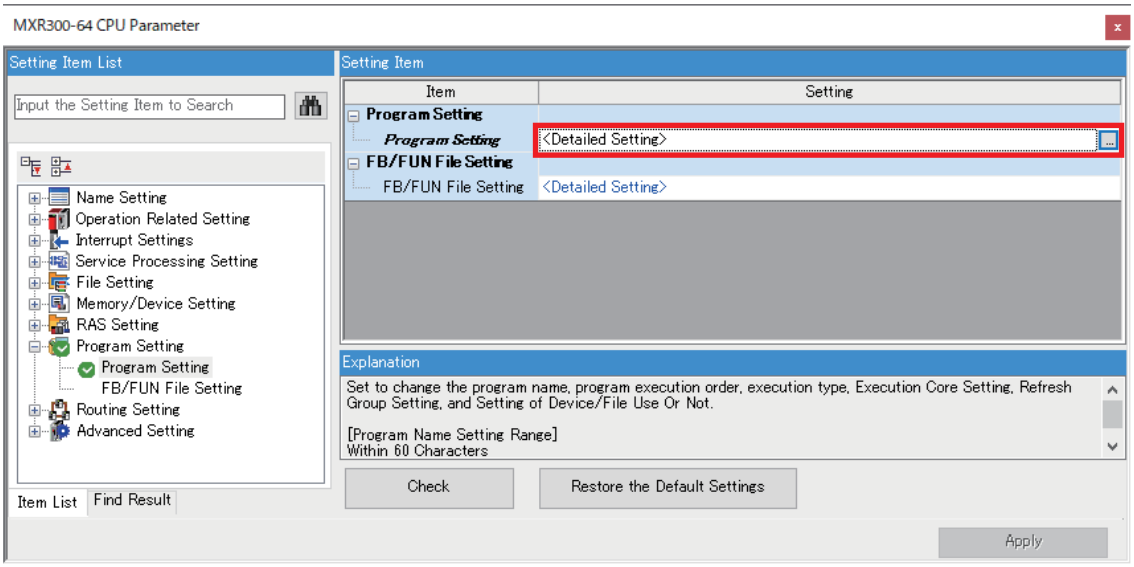
8. Open CPU parameters.

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ Double-click [CPU Parameters]

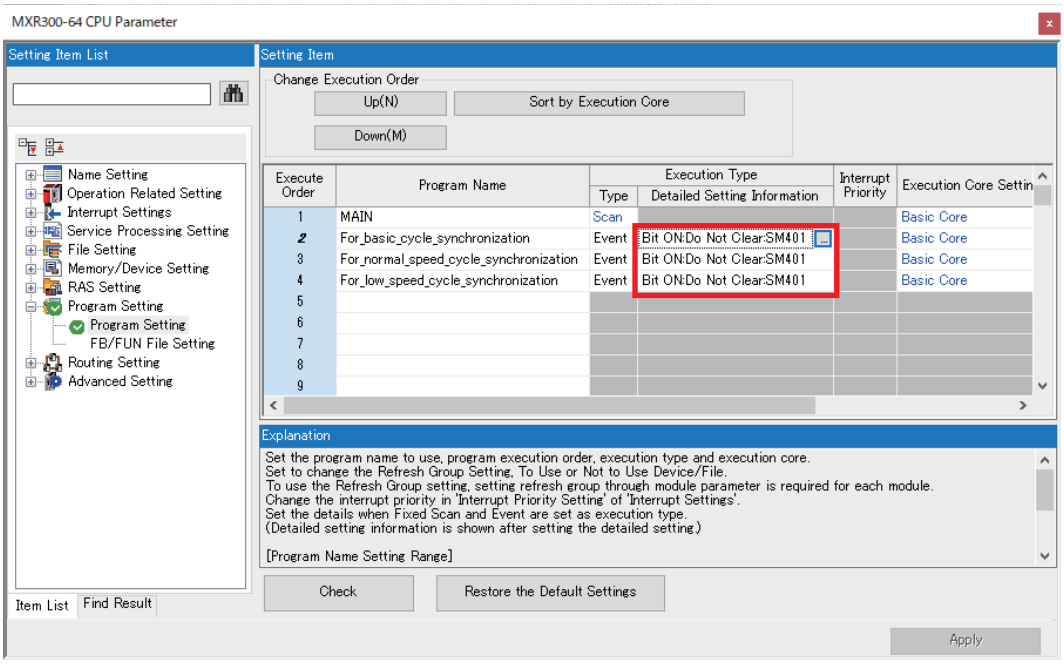


9. To change the trigger setting for the event execution type programs, open the program setting.

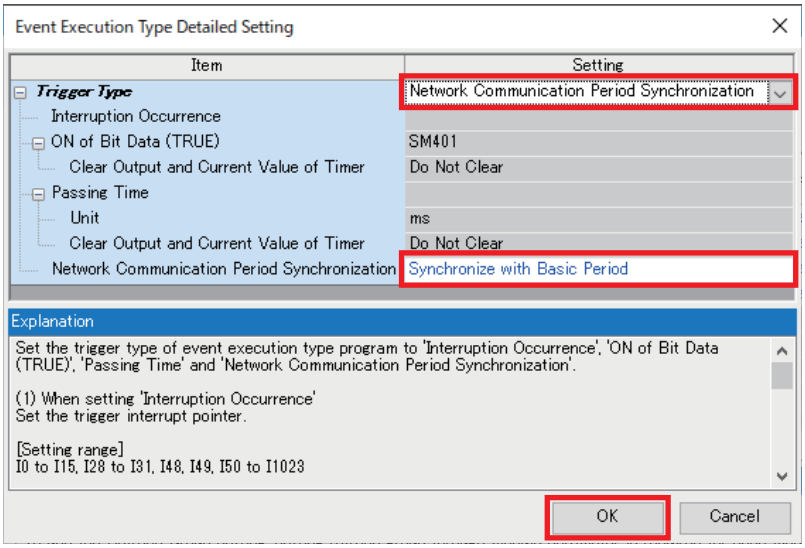
[Program Setting] ⇨ Double-click <Detailed Setting> of [Program Setting]



10. Double-click the "Detailed Setting Information" column of the event execution type programs created in this procedure.

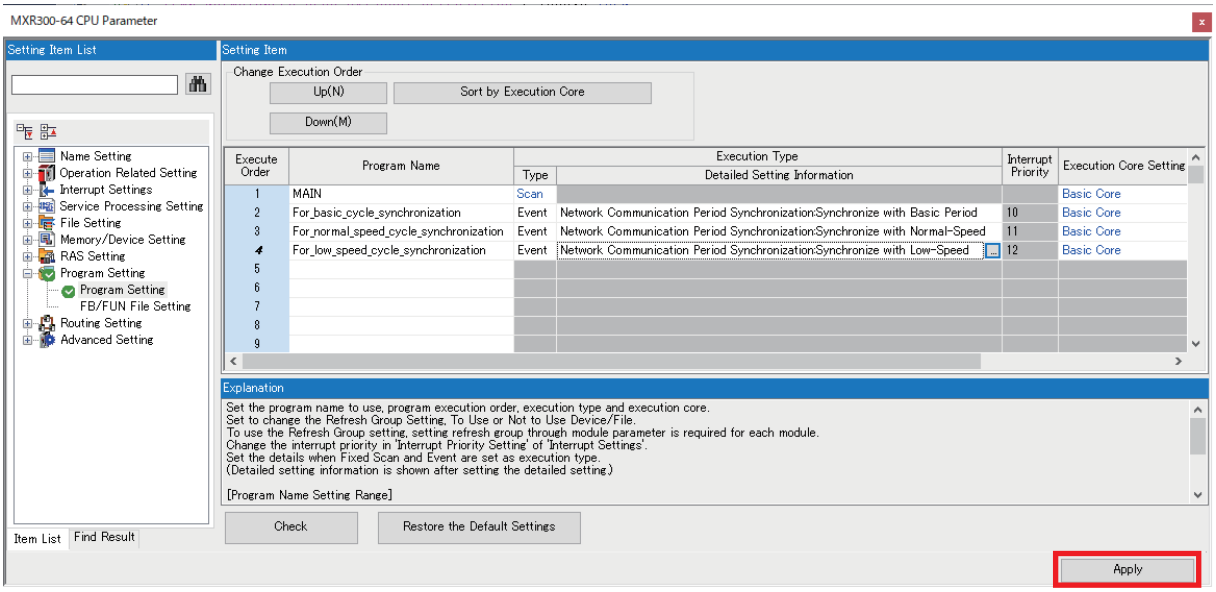


11. Set "Trigger Type" and "Network Communication Period Synchronization" as follows in accordance with the program contents, and then click the [OK] button.



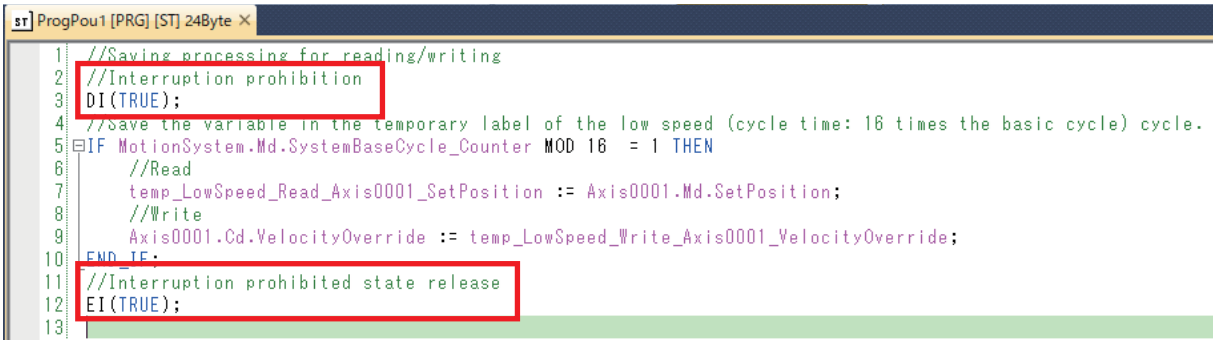
Program	Trigger type	Network communication cycle synchronization
Event execution type program to be synchronized with the basic cycle ("For_basic_cycle_synchronization" in this procedure)	Network Communication Period Synchronization	Synchronize with Basic Period
Event execution type program to be synchronized with the normal-speed cycle ("For_normal_speed_cycle_synchronization" in this procedure)		Synchronize with Normal-Speed
Event execution type program to be synchronized with the low-speed cycle ("For_low_speed_cycle_synchronization" in this procedure)		Synchronize with Low-Speed

12. Click the [Apply] button to apply the setting.



Precautions

- When the temporary labels used for saving are accessed from a program with a higher priority than the program performing the saving and restoration processes, use the DI and EI instructions to execute the saving and restoration processes in the interrupt disabled state.



- Due to the addition of the saving and restoration processes to/from the temporary labels, this technique may increase program complexity or scan time.

6 IMPROVEMENT OF LABEL EFFICIENCY

By changing labels that do not need to be global labels to local labels or by deleting unnecessary labels, label efficiency can be improved. Improvement of label efficiency also improves the program readability.
This chapter describes techniques for improving label efficiency.

List of techniques

The following table lists the techniques described in this chapter.

Item	Description	Reference
Reducing the number of global labels	Change labels that do not need to be global labels to local labels.	☞ Page 91 Reducing the Number of Global Labels
Deleting unused labels	Search the project for unused labels and delete them in a batch.	☞ Page 93 Deleting Unused Labels

6.1 Reducing the Number of Global Labels

Overview

A global label is used when the label needs to be used in multiple programs or when a device needs to be assigned to the label. If global labels are used in other cases, they can be reduced by changing them to local labels.
Reducing the number of global labels has the following benefits.

Reduction of program malfunction risk

The scope of label influence on the program can be localized, reducing the risk of malfunctions.

Reduction of program conversion time

The scope of label influence on the program can be localized, which can minimize the influence during program conversion, resulting in reduction of the program conversion time.

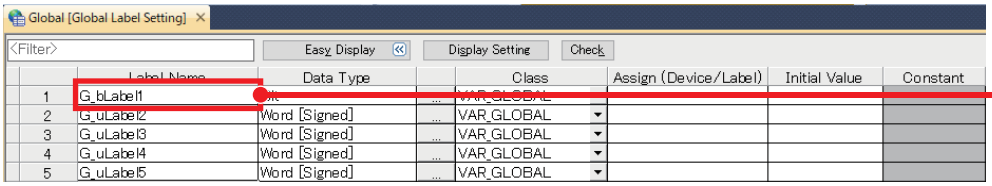
Execution procedure

This section describes the procedure for searching for unnecessary global labels, deleting them, and changing them to local labels.

Operating procedure

1. Check the usage of each global label by using cross reference.

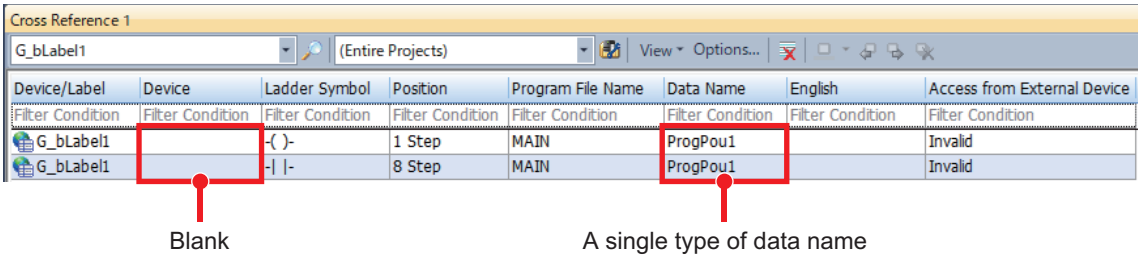
Open the global label, select the label to be checked, and then press the Ctrl and E keys.



Press the Ctrl and E keys with the label selected.

2. The cross reference result is displayed.

It can be determined that the label does not need to be a global label if the "Device" column is empty and only one data name is displayed in the "Data Name" column.

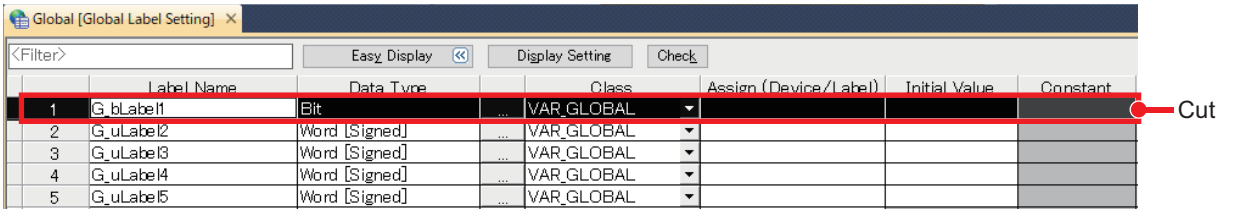


Point

By double-clicking the global label in the cross reference result, the program and position where the label is used can be checked. Check them before deleting the global label.

3. Cut the global label to be deleted from the global label editor.

When selecting the rows to be cut, click the row numbers in the editor so that the entire rows are selected.



Point

Otherwise, contents hidden in the label editor are not copied.

4. Open the local label of the program where the global label to be deleted is used, and paste the label cut in step 3.

After pasting it, set a class. If there are naming rules for local labels and global labels, change the label name as necessary.

Point

- When a label copied from the global label is pasted into the local label, the class setting is not pasted. Set a class again.
- The label name used in the program is automatically changed according to changes in steps 1 to 4. Therefore, modifying the program is not required. When it is not changed, it can be set by selecting the following items from the menu: [Tool] ⇨ [Options] ⇨ "Other Editor" ⇨ "Label Editor Common" ⇨ "Operational Setting" ⇨ "Track label name automatically in program editor" For details, refer to the following manual.

GX Works3 Operating Manual

5. For other global labels, perform steps 1 to 4.

Precautions

Global labels to be deleted (labels to be moved to the local label) cannot be accessed from external devices or used in other programs. Before deleting a global label, check the program where the label is used to determine if it can be deleted.

6.2 Deleting Unused Labels

Overview

The cross reference function of GX Works3 can be used to search the project for unused labels and delete them, improving label efficiency.
Deleting unused labels has the following benefits.

Increase in available label usage capacity

Deleting unused labels increases label usage capacity that can be used.

Reduction of program search time

Deleting unused labels reduces the number of search targets within the project, reducing the search time.

Reduction of program conversion time

Deleting unused labels reduces the label data amount, resulting in reduction of the program conversion time.

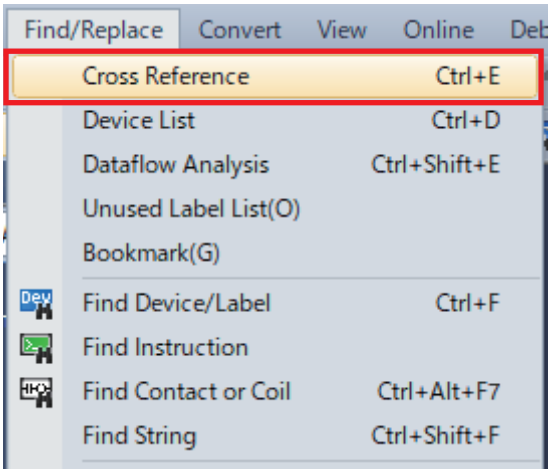
Execution procedure

This section describes the procedure for deleting unused labels.

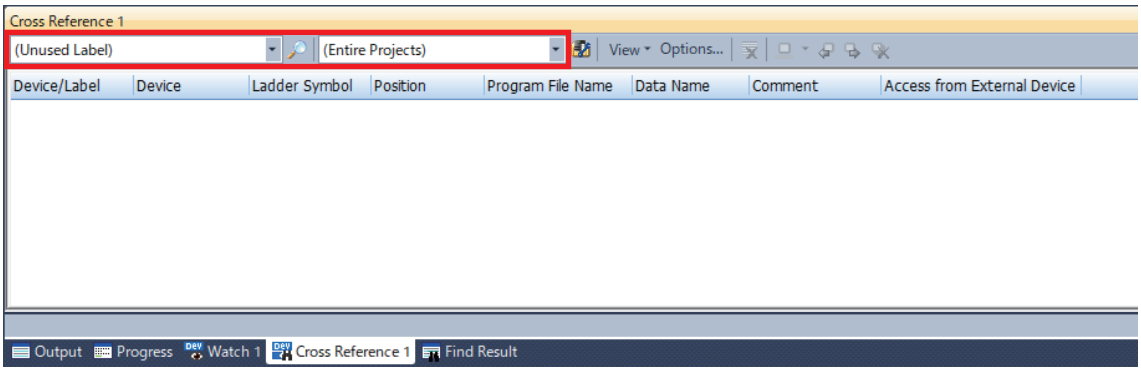
Operating procedure

1. Open cross reference to search for unused labels.

Click [Find/Replace] ⇨ [Cross Reference] from the menu.

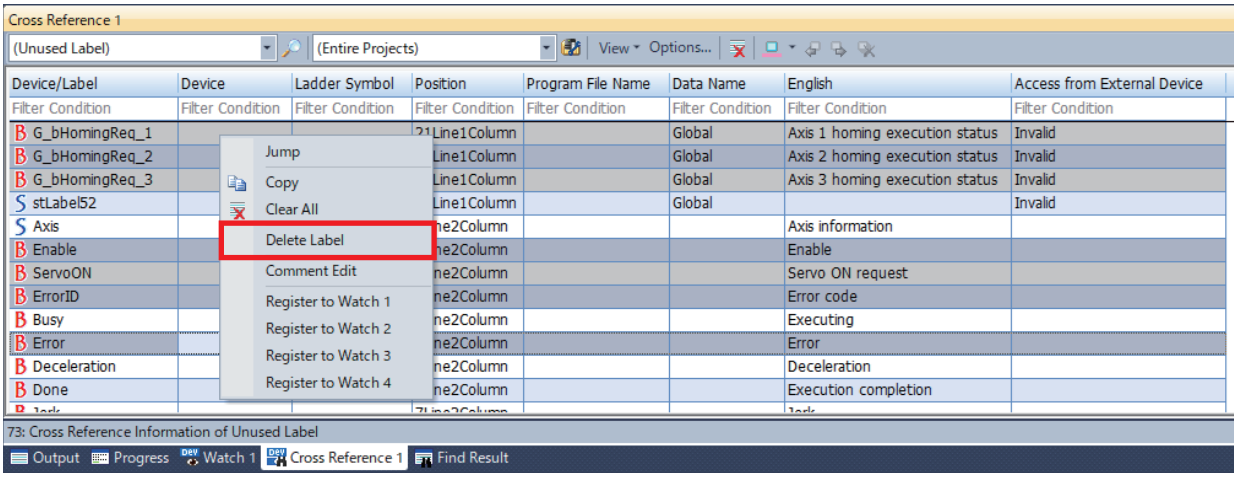


2. Set the search range to "(Unused Label)" and "(Entire Project)", and click the [ ] button.



3. A list of unused labels in the project is displayed.

Select labels to be deleted while pressing the Ctrl key, right-click them, and delete them by selecting [Delete Label].



Point

- Although all the labels can be deleted in a batch by selecting them with the Ctrl and A keys, deleting them one by one while checking the contents is recommended.
- By searching for unused labels again after deleting them, you can check that unused labels have been deleted.

7 INCREASING SPEED OF CC-Link IE TSN COMMUNICATION

The speed of the CC-Link IE TSN communication can be improved by modifying the link refresh setting and communication cycle setting of the device stations to reduce the link refresh time, communication cycle interval, and cyclic transmission time. By improving the speed of the CC-Link IE TSN communication, faster control can be achieved, improving the equipment takt time.

This chapter describes techniques for improving the speed of the CC-Link IE TSN communication.

List of techniques

The following table lists the techniques described in this chapter.

Item		Description	Reference
Reducing link refresh time		Reassign link device Nos. so that the device ranges in the refresh setting are sequential.	☞ Page 96 Reducing Link Refresh Time
Reducing communication cycle interval/ cyclic transmission time	Equalizing the number of device stations	Equalize the number of device stations to be connected to each port of the master station.	☞ Page 103 Equalizing the Number of Device Stations
	Setting the upper limit of transmission size	For device stations whose CC-Link IE TSN Class is "Class A" and communication cycle is set to "Low-Speed", set a small upper limit of the transmission size in the CC-Link IE TSN Class A group.	☞ Page 104 Setting the Upper Limit of Transmission Size
	Reducing the number of device stations	If there are any unnecessary device stations, reduce the number of device stations connected to the master station.	☞ Page 106 Reducing the Number of Device Stations
	Deleting link device settings	Delete unnecessary link device settings from each device station.	☞ Page 110 Deleting Link Device Settings
	Reducing the points of link device settings	Reduce the points of link device settings in each device station.	☞ Page 115 Reducing the Points of Link Device Settings
	Unifying network synchronous communication settings	If device stations whose network synchronous communication setting is set to "Asynchronous" and "Synchronous" coexist in device stations whose CC-Link IE TSN Class is "Class B", unify the setting to either of them.	☞ Page 120 Unifying Network Synchronous Communication Settings
	Setting distribution of cyclic transmission	For device stations whose CC-Link IE TSN Class is "Class B" and communication cycle is set to "Normal-Speed" or "Low-Speed", set the cyclic transmission distribution.	☞ Page 123 Setting Distribution of Cyclic Transmission

Checking communication cycle interval and cyclic transmission time

The actual measured value of the communication cycle interval and the calculation value of the cyclic transmission time can be checked in the following buffer memory areas.

Use them to check the communication cycle interval and cyclic transmission time when each technique is executed.

- Communication cycle interval (actual measured value): (U3E0\G10200 to U3E0\G10201)
- Cyclic transmission time (calculation value): (U3E0\G10272 to U3E0\G10273)

Since the communication cycle interval is an actual measured value, the displayed value may change slightly when the controller is reset.

7.1 Reducing Link Refresh Time

Overview

By reassigning the device numbers of the RX setting/RY setting and the RWr setting/RWw setting for device stations configured in the network configuration settings using "Communication Period Interval Shortening Assignment", the refresh destination device range is set sequentially. This reduces the number of refresh settings, simplifying the refresh settings. Additionally, it reduces the link refresh time, communication cycle interval, and cyclic transmission time, enabling faster CC-Link IE TSN communication.

Communication period interval shortening assignment

The device number reassignment function in the network configuration settings enables sequential assignment of device numbers to the link devices of the specified target stations.
When "Communication Period Interval Shortening Assignment" is selected as the assignment method for the device number reassignment function, device number assignment is executed for all stations to reduce the cyclic receive processing time.
In the communication period interval shortening assignment, the device numbers for the RX setting, RY setting, RWr setting, and RWw setting of device stations configured in the network configuration settings are assigned sequentially in the following order. (The device numbers within each setting are assigned in the station number order.)

Assignment order	Network Synchronous Communication	Communication Period Setting
1	Asynchronous	Basic Period
2	Asynchronous	Normal-Speed
3	Asynchronous	Low-Speed
4	Synchronous	Basic Period
5	Synchronous	Normal-Speed
6	Synchronous	Low-Speed

For details on the communication period interval shortening assignment, refer to the following manual.
MELSEC MX Controller (MX-R Model) User's Manual

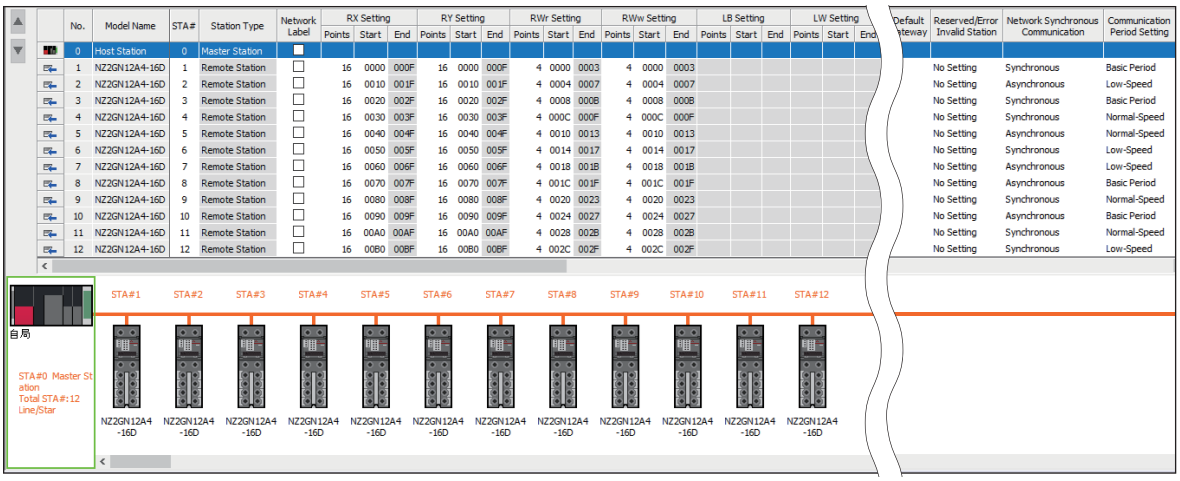
Precautions

- When the communication period interval shortening assignment is performed on link devices of a remote station that already have device numbers assigned, the order of existing link device numbers is changed.
- The communication period interval shortening assignment is effective when there are device stations with the network synchronous communication setting set to "Synchronous".

Execution procedure

Using the following network configuration and refresh settings as examples, this section describes the procedure for reassigning the link device numbers.

- Network configuration



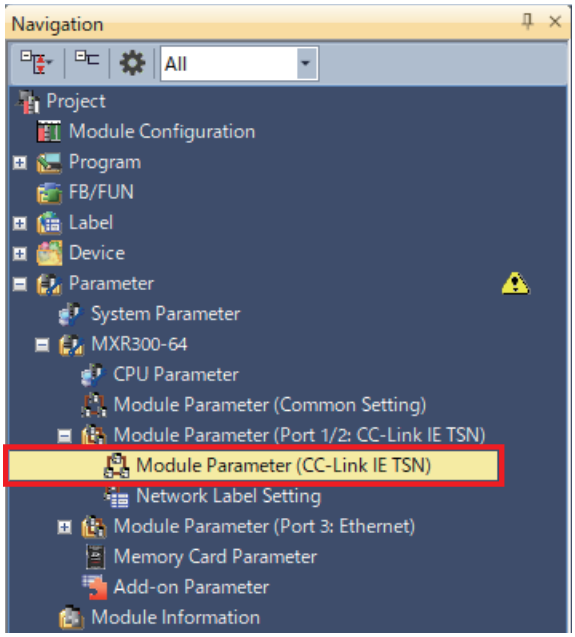
- Refresh setting

No.	Link Side					CPU Side				
	Device Name	Points	Start	End		Target	Device Name	Points	Start	End
-	SB									
-	SW									
1	RX	192	00000	000BF		Specify Device	X	192	01000	010BF
2	RY	192	00000	000BF		Specify Device	Y	192	01000	010BF
3	RWr	48	00000	0002F		Specify Device	W	48	00000	0002F
4	RWw	48	00000	0002F		Specify Device	W	48	01000	0102F

Operating procedure

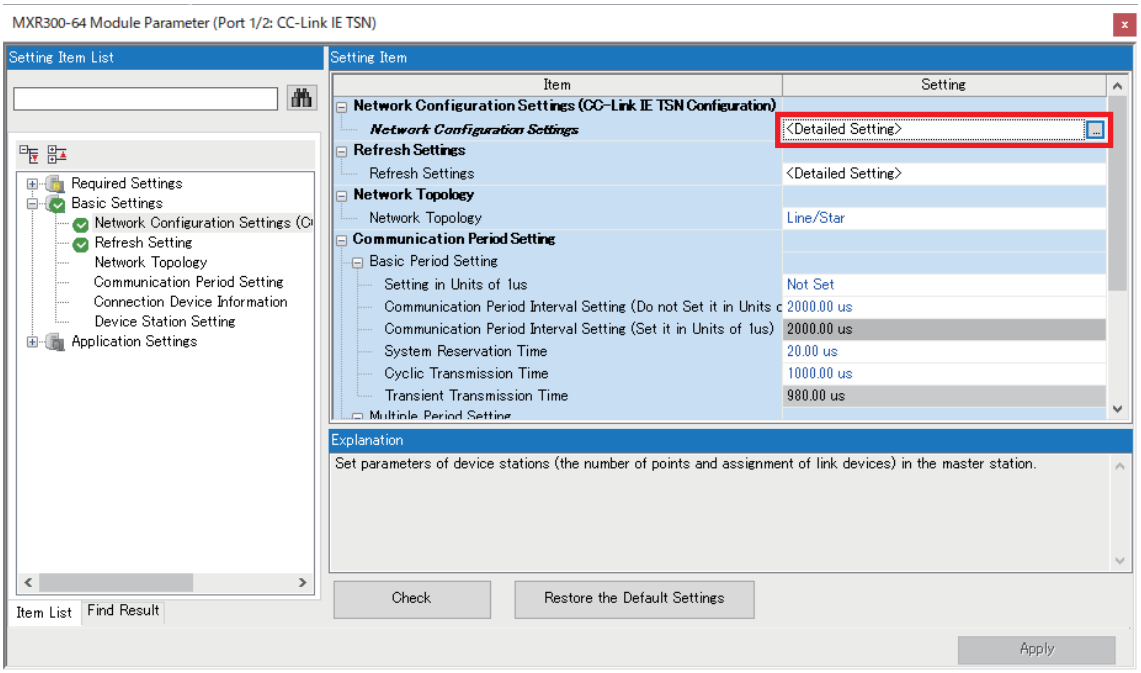
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]



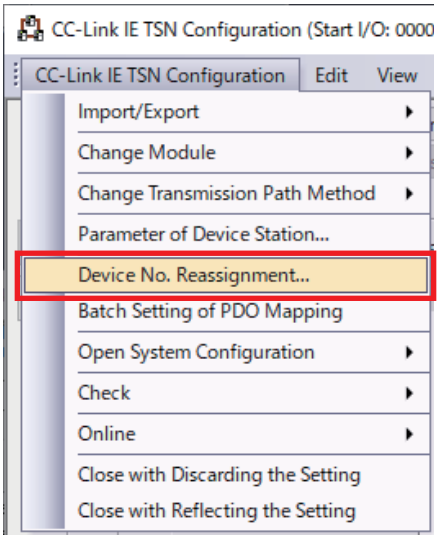
2. In the basic settings, open the network configuration settings.

[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



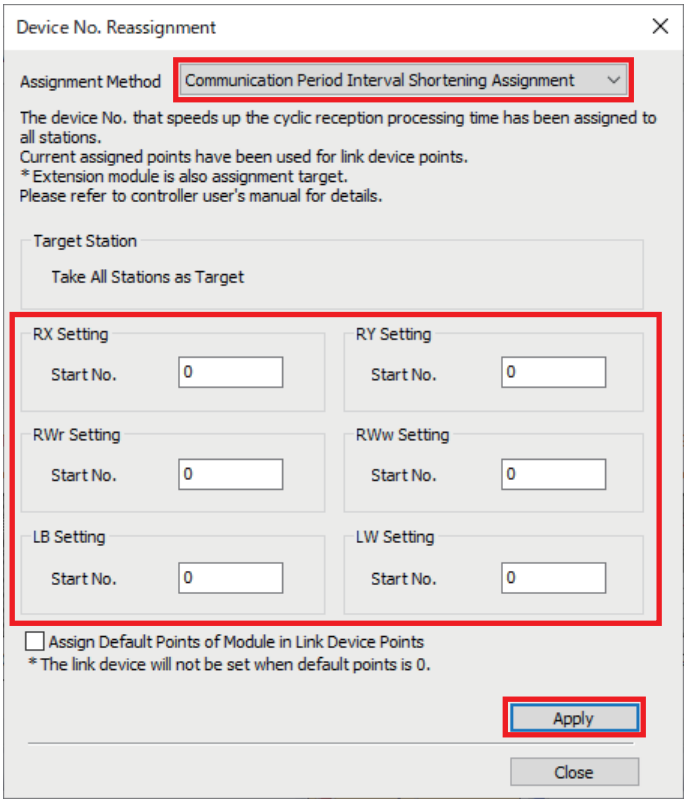
3. Reassign the device numbers.

Click [CC-Link IE TSN Configuration] ⇨ [Device No. Reassignment] from the menu.



4. Select "Communication Period Interval Shortening Assignment" for "Assignment Method", set the start number of each link device, and then click the [Apply] button.

• Start No. of each link device: 0

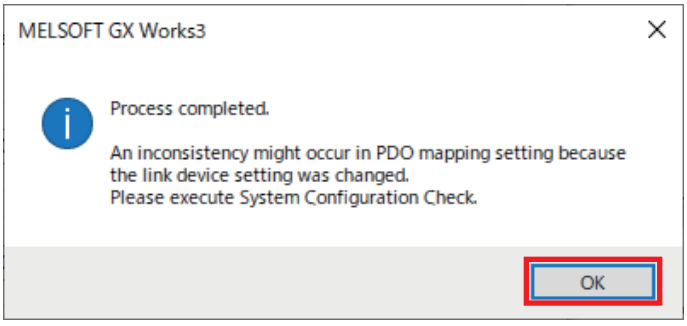


Precautions

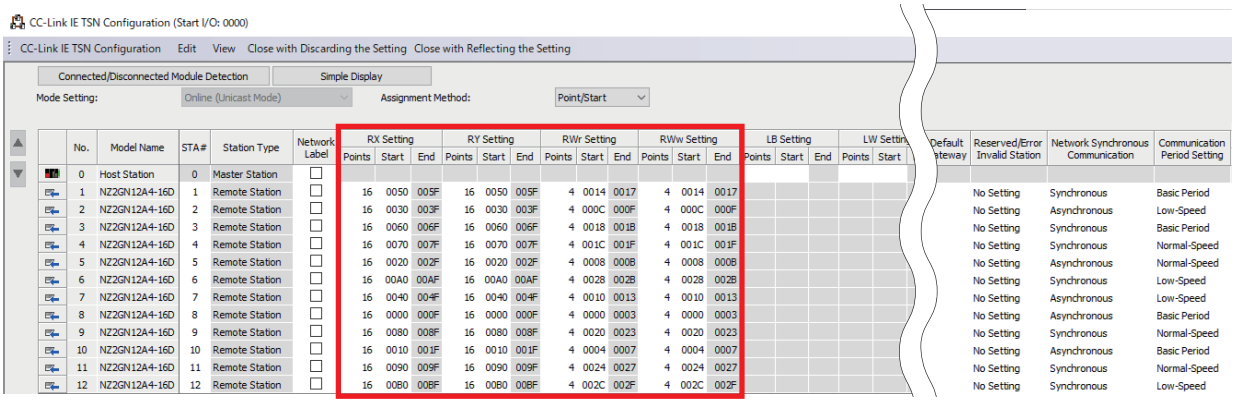
If device numbers have already been assigned to each link device, the order of the numbers is changed. For details on the assignment orders of devices, refer to the following.

☞ Page 96 Communication period interval shortening assignment

5. Click the [OK] button.



The device numbers are reassigned.

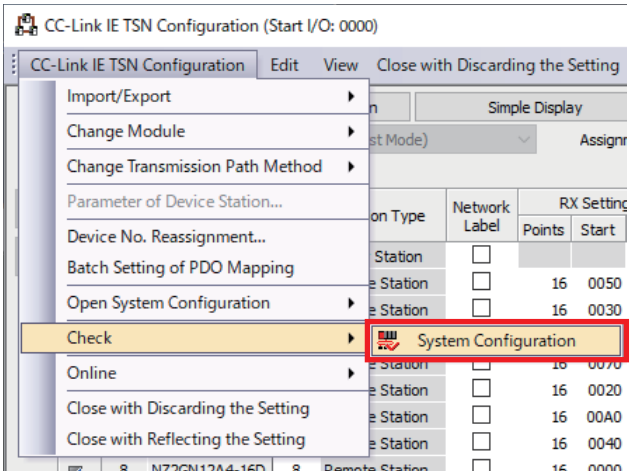


No.	Model Name	STA#	Station Type	Network Label	RX Setting Points	RX Setting Start	RX Setting End	RY Setting Points	RY Setting Start	RY Setting End	RWr Setting Points	RWr Setting Start	RWr Setting End	RWw Setting Points	RWw Setting Start	RWw Setting End	LB Setting Points	LB Setting Start	LB Setting End	LW Setting Points	LW Setting Start	Default Gateway	Reserved/Error Invalid Station	Network Synchronous Communication	Communication Period Setting
0	Host Station	0	Master Station																						
1	NZ2GN12A4-16D	1	Remote Station		16	0050	005F	16	0050	005F	4	0014	0017	4	0014	0017							No Setting	Synchronous	Basic Period
2	NZ2GN12A4-16D	2	Remote Station		16	0030	003F	16	0030	003F	4	000C	000F	4	000C	000F							No Setting	Asynchronous	Low-Speed
3	NZ2GN12A4-16D	3	Remote Station		16	0060	006F	16	0060	006F	4	0018	001B	4	0018	001B							No Setting	Synchronous	Basic Period
4	NZ2GN12A4-16D	4	Remote Station		16	0070	007F	16	0070	007F	4	001C	001F	4	001C	001F							No Setting	Synchronous	Normal-Speed
5	NZ2GN12A4-16D	5	Remote Station		16	0020	002F	16	0020	002F	4	0008	000B	4	0008	000B							No Setting	Asynchronous	Normal-Speed
6	NZ2GN12A4-16D	6	Remote Station		16	00A0	00AF	16	00A0	00AF	4	0028	002B	4	0028	002B							No Setting	Synchronous	Low-Speed
7	NZ2GN12A4-16D	7	Remote Station		16	0040	004F	16	0040	004F	4	0010	0013	4	0010	0013							No Setting	Asynchronous	Low-Speed
8	NZ2GN12A4-16D	8	Remote Station		16	0000	000F	16	0000	000F	4	0000	0003	4	0000	0003							No Setting	Asynchronous	Basic Period
9	NZ2GN12A4-16D	9	Remote Station		16	0080	008F	16	0080	008F	4	0020	0023	4	0020	0023							No Setting	Synchronous	Normal-Speed
10	NZ2GN12A4-16D	10	Remote Station		16	0010	001F	16	0010	001F	4	0004	0007	4	0004	0007							No Setting	Asynchronous	Basic Period
11	NZ2GN12A4-16D	11	Remote Station		16	0090	009F	16	0090	009F	4	0024	0027	4	0024	0027							No Setting	Synchronous	Normal-Speed
12	NZ2GN12A4-16D	12	Remote Station		16	00B0	00BF	16	00B0	00BF	4	002C	002F	4	002C	002F							No Setting	Synchronous	Low-Speed

6. Check the system configuration.

If any error or warning occurs, review the configuration and settings.

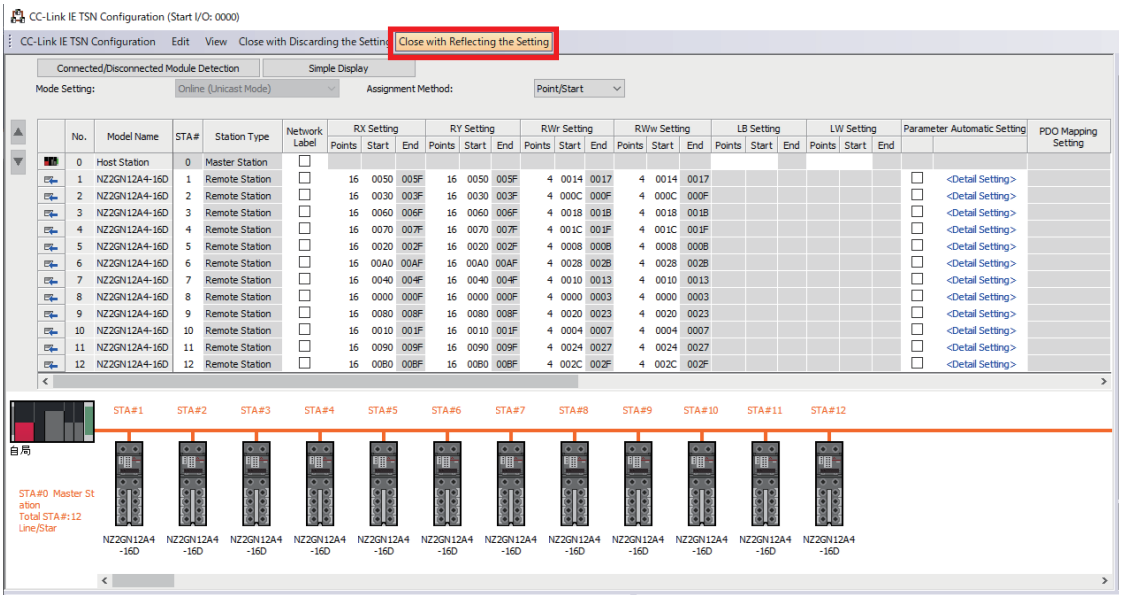
Click [CC-Link IE TSN Configuration] ⇨ [Check] ⇨ [System Configuration] from the menu.



Point

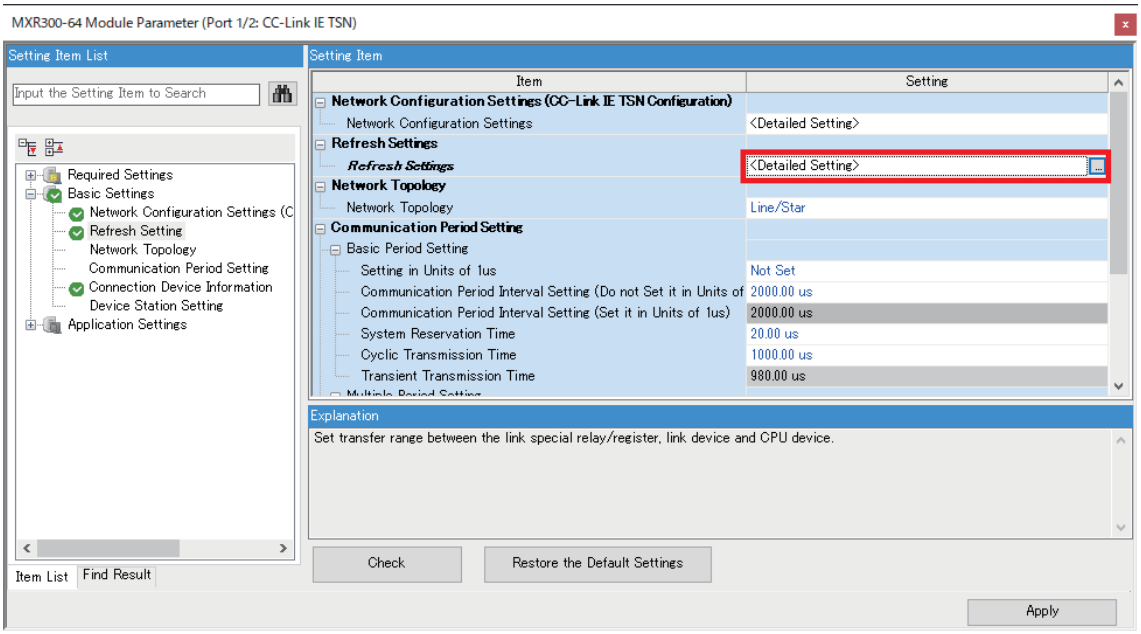
If there is any error or warning, it is displayed in the "Output" docking window.

7. Click the [Close with Reflecting the Setting] button.



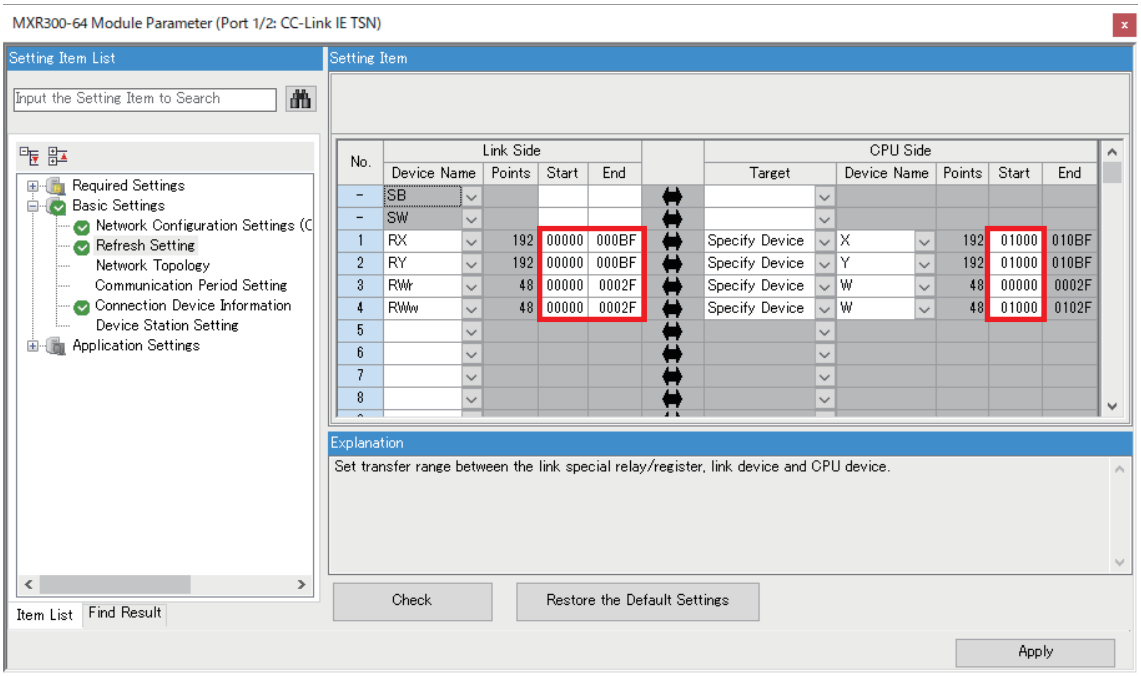
8. Configure the refresh settings according to the link refresh setting.

[Basic Settings] ⇨ [Refresh Setting] ⇨ Double-click <Detailed Setting> of [Refresh Settings]

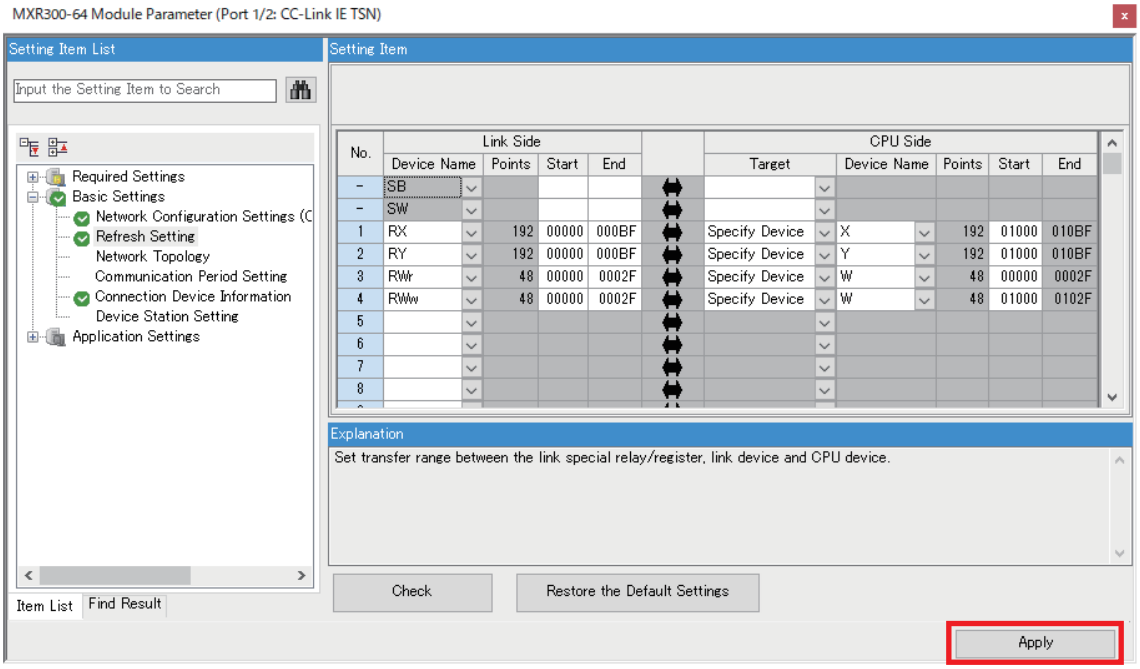


9. Check if the start and end numbers of the link side devices have changed due to the reassignment of link devices, and change them if necessary.

If the link side settings have been changed, review the CPU side settings as well.



10. Click the [Apply] button to apply the setting.



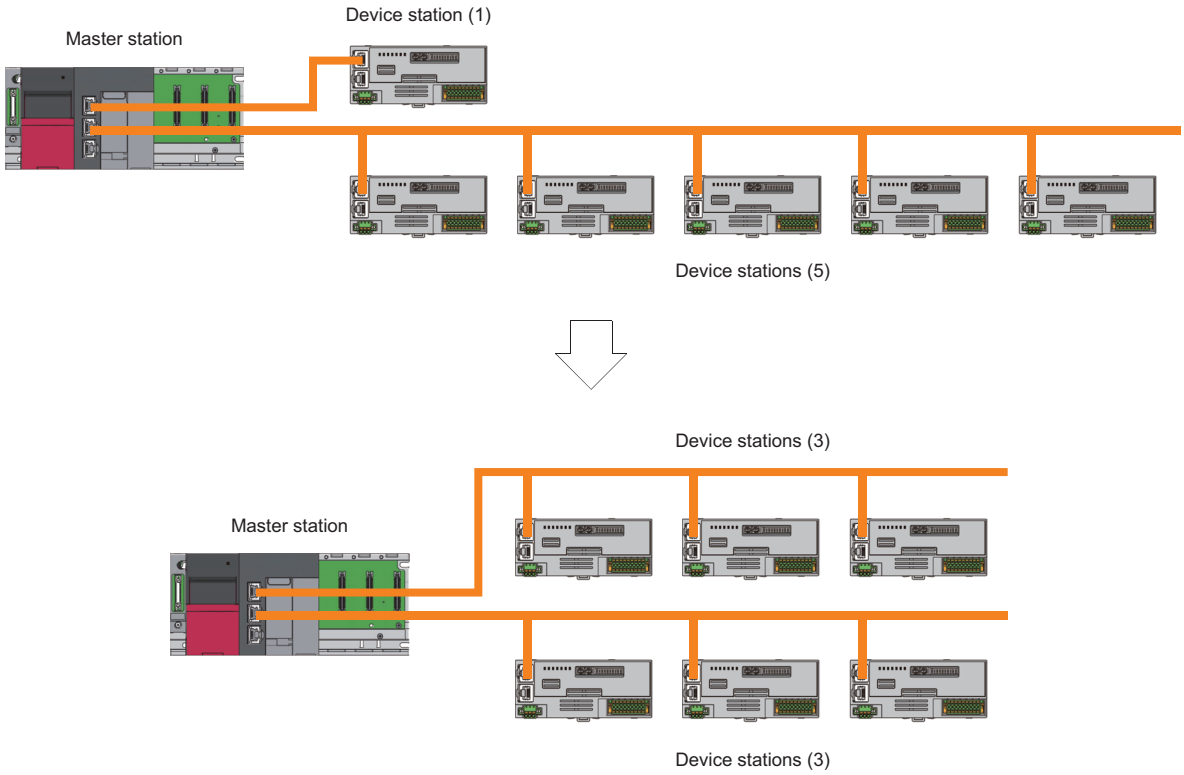
7.2 Equalizing the Number of Device Stations

Overview

By equalizing the number of device stations connected to each port of the master station, the communication cycle interval and cyclic transmission time can be reduced.

Execution procedure

If the number of device stations connected to each port is not equal as shown below, modify the connection so that it becomes equal.



Precautions

Equalizing the number of device stations may impose restrictions on the placement of device stations and the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.

7.3 Setting the Upper Limit of Transmission Size

Overview

For device stations whose CC-Link IE TSN Class is "Class A" and communication cycle is set to "Low-Speed" connected to the master station, the communication cycle interval and cyclic transmission time can be reduced by setting a small upper limit of the transmission size in the CC-Link IE TSN Class A group.

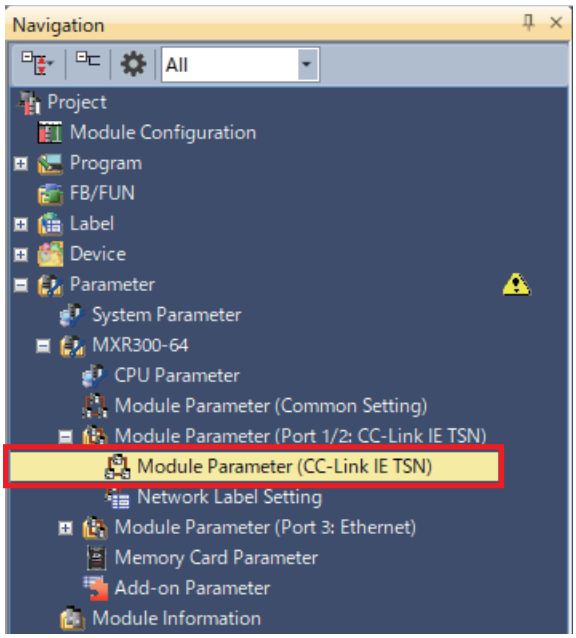
Execution procedure

Using the case in which a device station whose CC-Link IE TSN Class is "Class A" and communication cycle is set to "Low-Speed" is connected to the master station as an example, this section describes the procedure for setting the upper limit of the transmission size in the CC-Link IE TSN Class A group.

Operating procedure

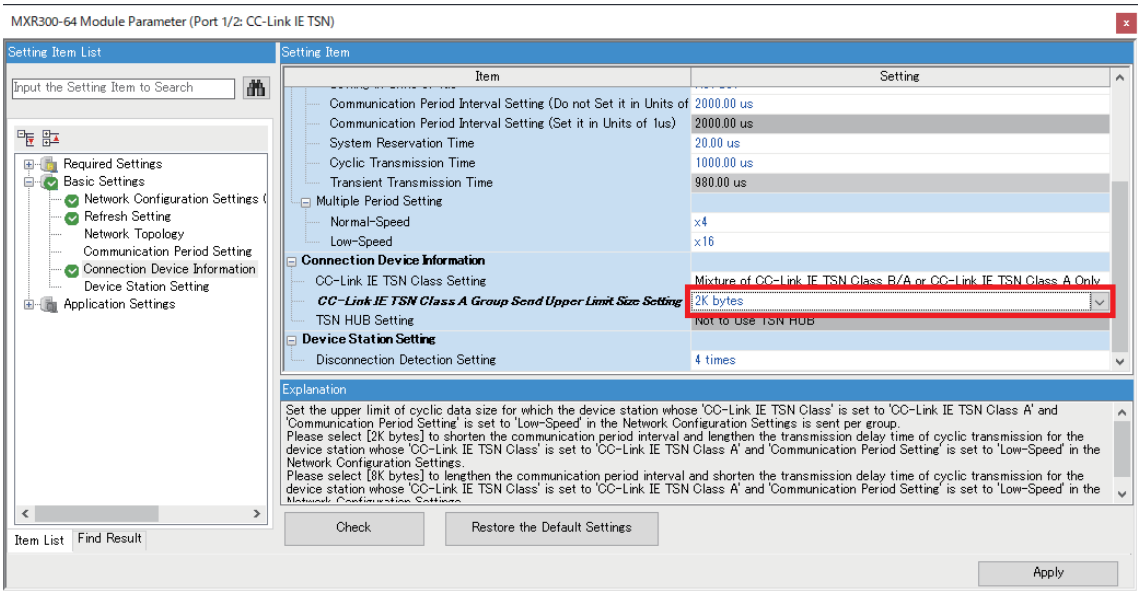
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]

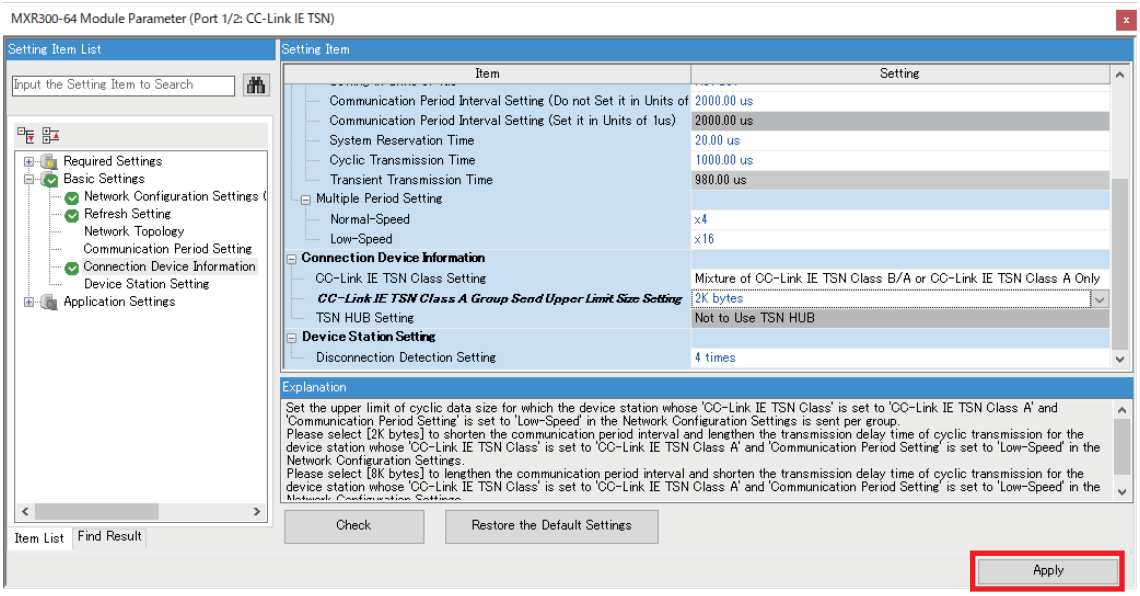


2. Set the upper limit of the transmission size in the CC-Link IE TSN Class A group to "2K bytes" (default).

[Basic Settings] ⇨ [Connection Device Information] ⇨ [CC-Link IE TSN Class A Group Send Upper Limit SIze Setting]



3. Click the [Apply] button to apply the setting.



Precautions

- Setting the upper limit of the transmission size may impose restrictions on the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.
- When "CC-Link IE TSN Class A Group Send Upper Limit Size Setting" is set to "2K bytes", communication is performed with one group per communication cycle. As a result, the transmission delay time for device stations whose CC-Link IE TSN Class is "Class A" and communication cycle is set to "Low-Speed" will increase. For details, refer to the following manual.

MELSEC MX Controller (MX-R Model) User's Manual

7.4 Reducing the Number of Device Stations

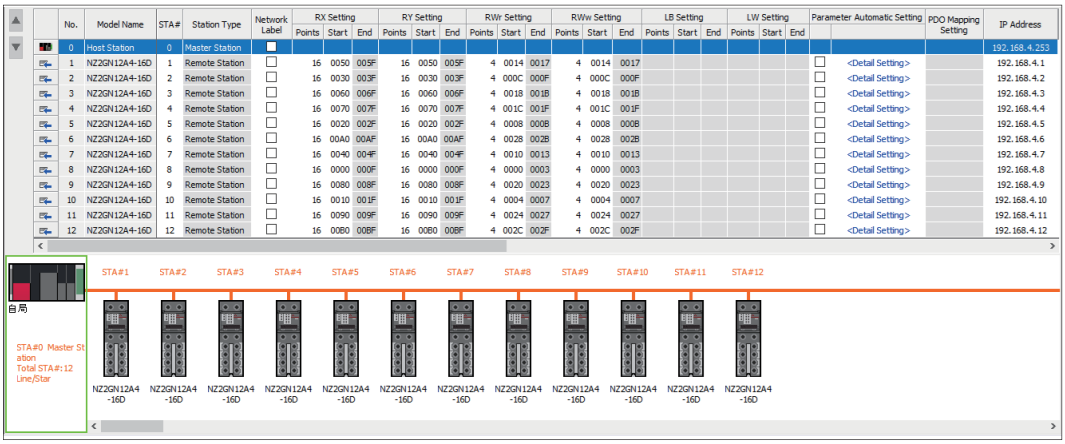
Overview

By reducing the number of device stations connected to the master station, the communication cycle interval and cyclic transmission time can be reduced.

Execution procedure

Using the following network configuration and refresh settings as examples, this section describes the procedure for reducing the number of device stations.

- Network configuration



Device stations with station Nos. 10 to 12 (not used)

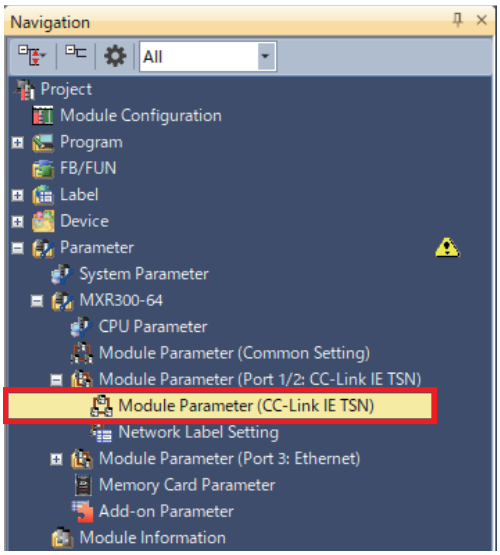
- Refresh setting

No.	Link Side					CPU Side				
	Device Name	Points	Start	End		Target	Device Name	Points	Start	End
-	SB									
-	SW									
1	RX	192	00000	000BF		Specify Device	X	192	01000	010BF
2	RY	192	00000	000BF		Specify Device	Y	192	01000	010BF
3	RWr	48	00000	0002F		Specify Device	W	48	00000	0002F
4	RWw	48	00000	0002F		Specify Device	W	48	01000	0102F

Operating procedure

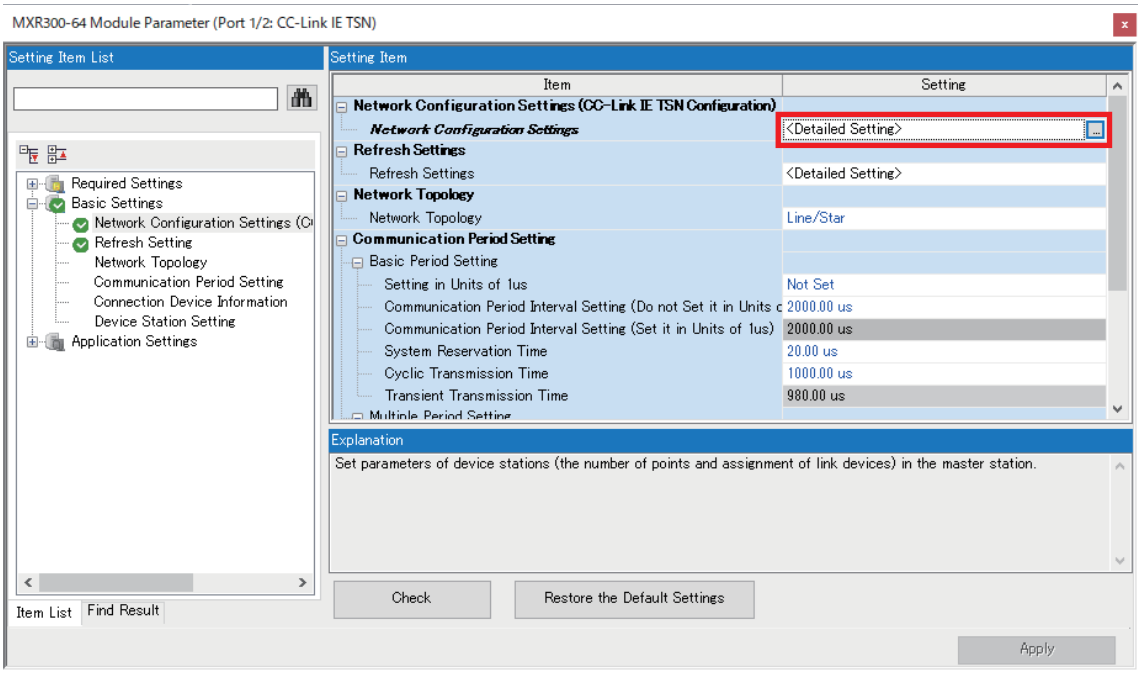
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]



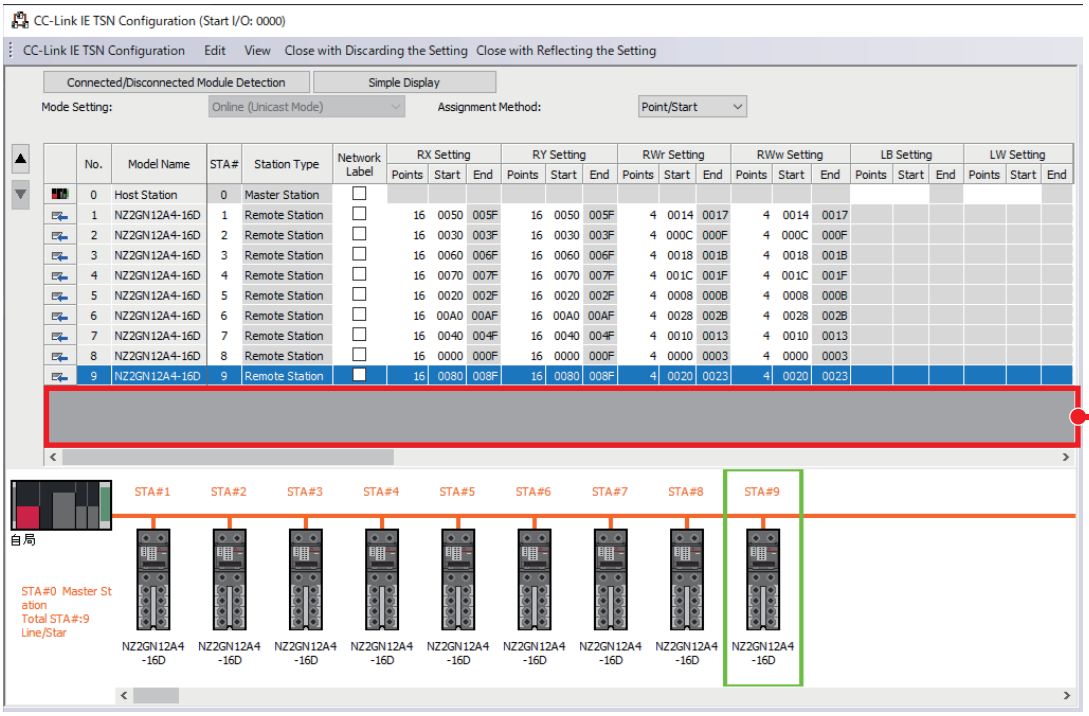
2. In the basic settings, open the network configuration settings.

[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



3. Delete unused and unnecessary device stations.

- Device stations with station Nos. 10 to 12: Delete



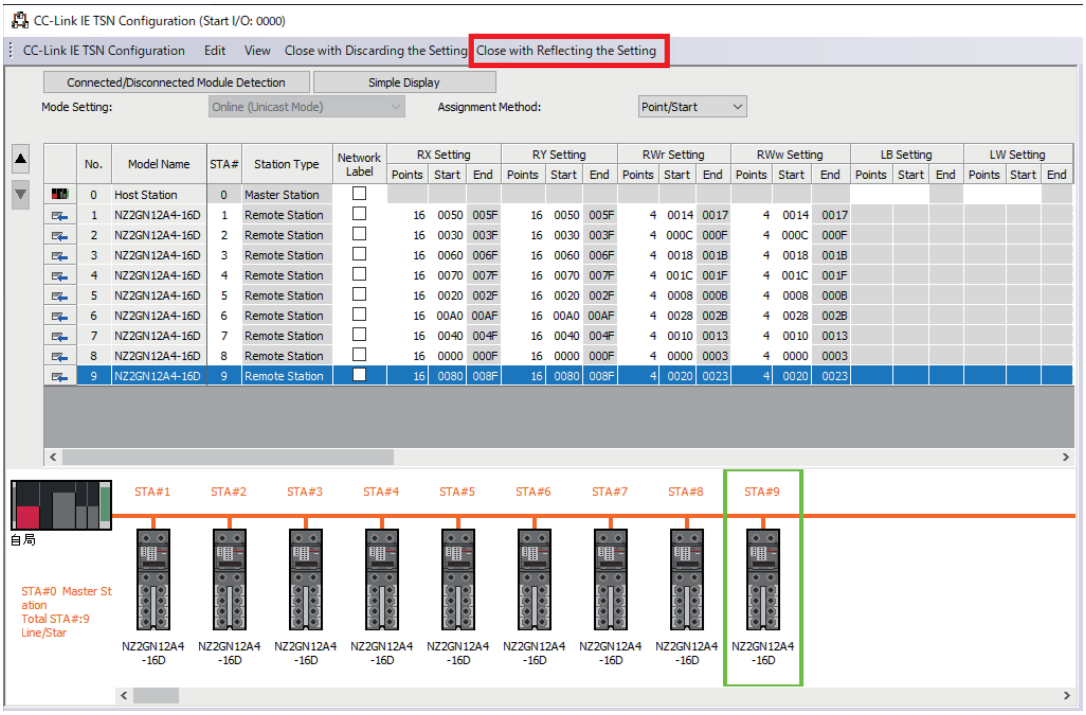
Point

After unnecessary device stations are deleted, the link refresh time, communication cycle interval, and cyclic transmission time may be reduced further by reassigning the link device numbers.

For details, refer to the following.

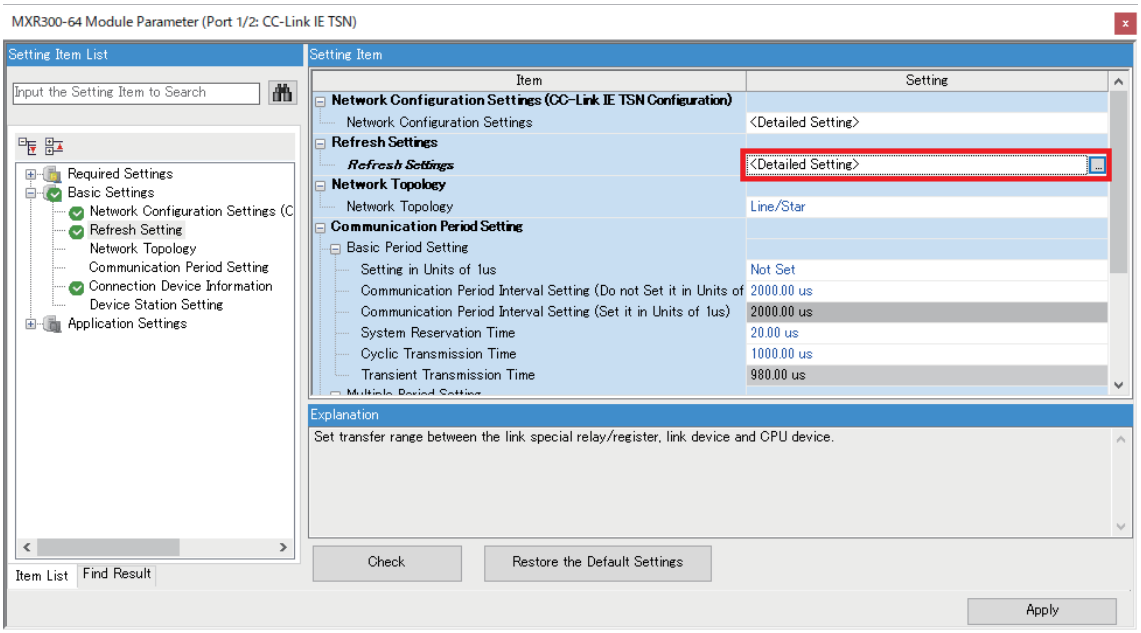
☞ Page 96 Reducing Link Refresh Time

4. Click the [Close with Reflecting the Setting] button.



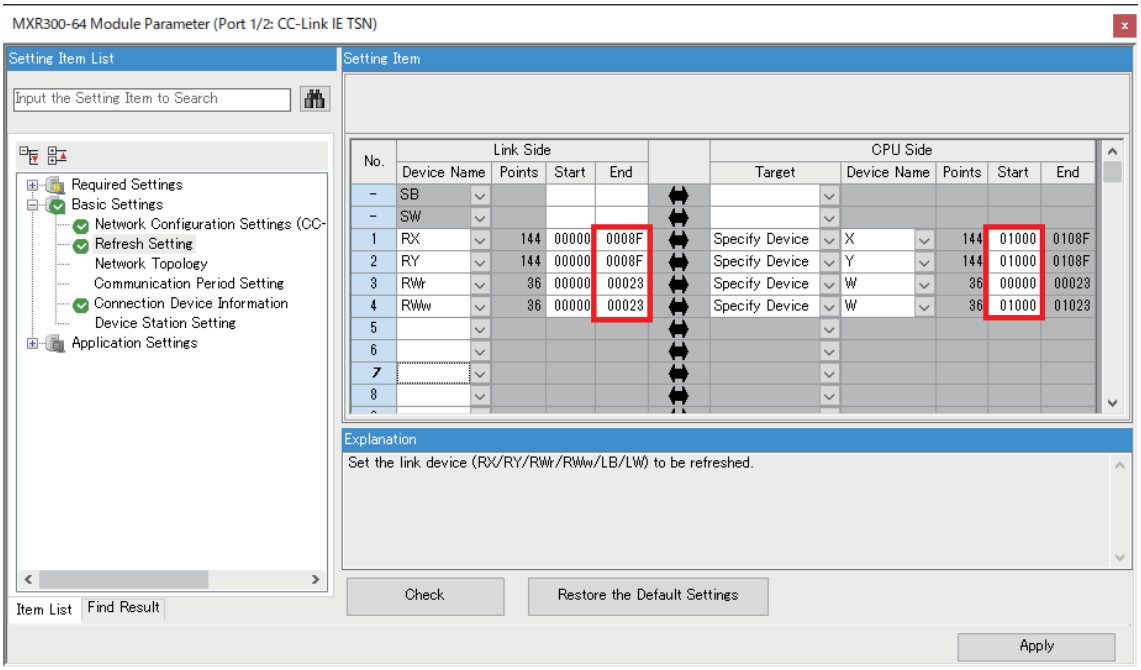
5. Configure the refresh settings according to the link refresh setting.

[Basic Settings] ⇨ [Refresh Setting] ⇨ Double-click <Detailed Setting> of [Refresh Settings]

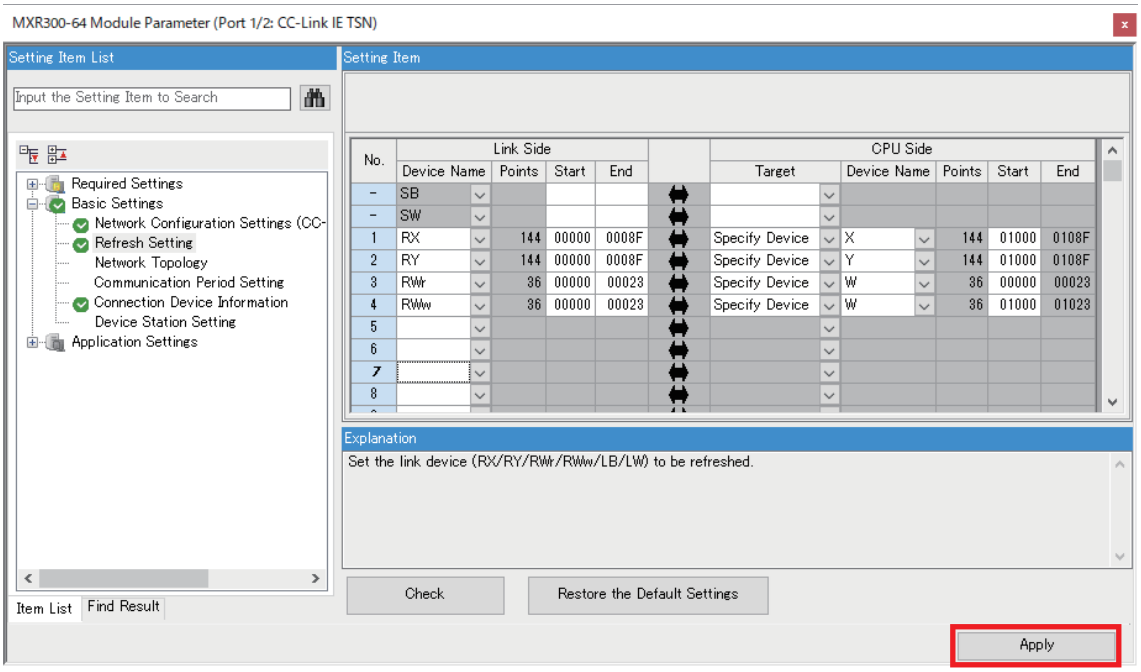


6. Change the end numbers of the link side devices according to the deletion of unnecessary devices in step 3. In addition, review the CPU side settings according to the changes in the link side settings.

- End numbers of the RX and RY settings: 0008F
- End numbers of the RWr and RWw settings: 00023



7. Click the [Apply] button to apply the setting.



Precautions

- Deleting device stations reduces the number of connected device stations and may impose restrictions on the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.
- Deleting device stations whose CC-Link IE TSN Class is "CC-Link IE TSN Class A" and communication cycle is set to "Low-Speed" does not have any effect on accelerating CC-Link IE TSN. Check the settings of each device station in the "Network Configuration Settings" window.

7.5 Deleting Link Device Settings

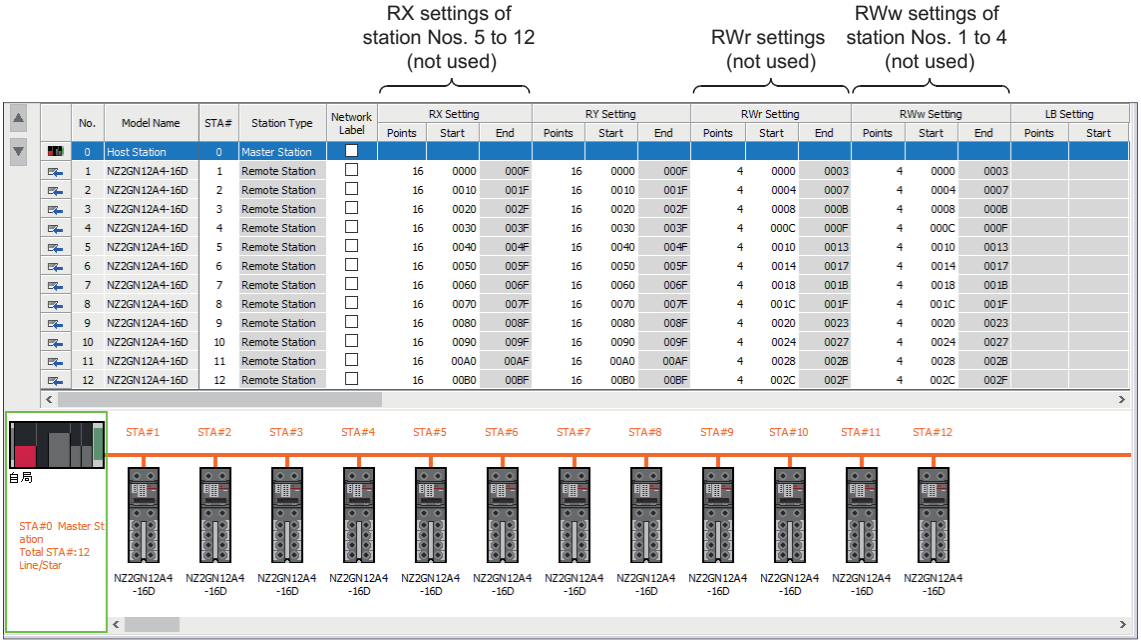
Overview

By deleting unnecessary link device settings of each device station, the communication cycle interval and cyclic transmission time can be reduced.

Execution procedure

Using the following network configuration and refresh settings as examples, this section describes the procedure for deleting unnecessary link device settings.

- Network configuration



No.	Model Name	STA#	Station Type	Network Label	RX Setting Points	RX Setting Start	RX Setting End	RY Setting Points	RY Setting Start	RY Setting End	RWr Setting Points	RWr Setting Start	RWr Setting End	RWw Setting Points	RWw Setting Start	RWw Setting End	LB Setting Points	LB Setting Start
0	Host Station	0	Master Station															
1	NZ2GN12A4-16D	1	Remote Station		16	0000	000F	16	0000	000F	4	0000	0003	4	0000	0003		
2	NZ2GN12A4-16D	2	Remote Station		16	0010	001F	16	0010	001F	4	0004	0007	4	0004	0007		
3	NZ2GN12A4-16D	3	Remote Station		16	0020	002F	16	0020	002F	4	0008	000B	4	0008	000B		
4	NZ2GN12A4-16D	4	Remote Station		16	0030	003F	16	0030	003F	4	000C	000F	4	000C	000F		
5	NZ2GN12A4-16D	5	Remote Station		16	0040	004F	16	0040	004F	4	0010	0013	4	0010	0013		
6	NZ2GN12A4-16D	6	Remote Station		16	0050	005F	16	0050	005F	4	0014	0017	4	0014	0017		
7	NZ2GN12A4-16D	7	Remote Station		16	0060	006F	16	0060	006F	4	0018	001B	4	0018	001B		
8	NZ2GN12A4-16D	8	Remote Station		16	0070	007F	16	0070	007F	4	001C	001F	4	001C	001F		
9	NZ2GN12A4-16D	9	Remote Station		16	0080	008F	16	0080	008F	4	0020	0023	4	0020	0023		
10	NZ2GN12A4-16D	10	Remote Station		16	0090	009F	16	0090	009F	4	0024	0027	4	0024	0027		
11	NZ2GN12A4-16D	11	Remote Station		16	00A0	00AF	16	00A0	00AF	4	0028	002B	4	0028	002B		
12	NZ2GN12A4-16D	12	Remote Station		16	00B0	00BF	16	00B0	00BF	4	002C	002F	4	002C	002F		

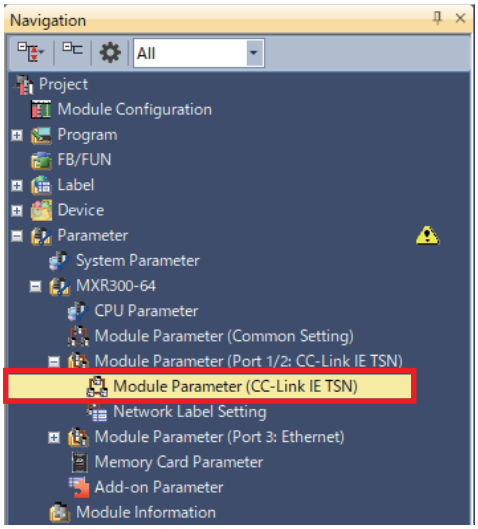
- Refresh setting

No.	Link Side Device Name	Points	Start	End		CPU Side Target	Device Name	Points	Start	End
-	SB									
-	SW									
1	RX	192	00000	000BF		Specify Device	X	192	01000	010BF
2	RY	192	00000	000BF		Specify Device	Y	192	01000	010BF
3	RWr	48	00000	0002F		Specify Device	W	48	00000	0002F
4	RWw	48	00000	0002F		Specify Device	W	48	01000	0102F

Operating procedure

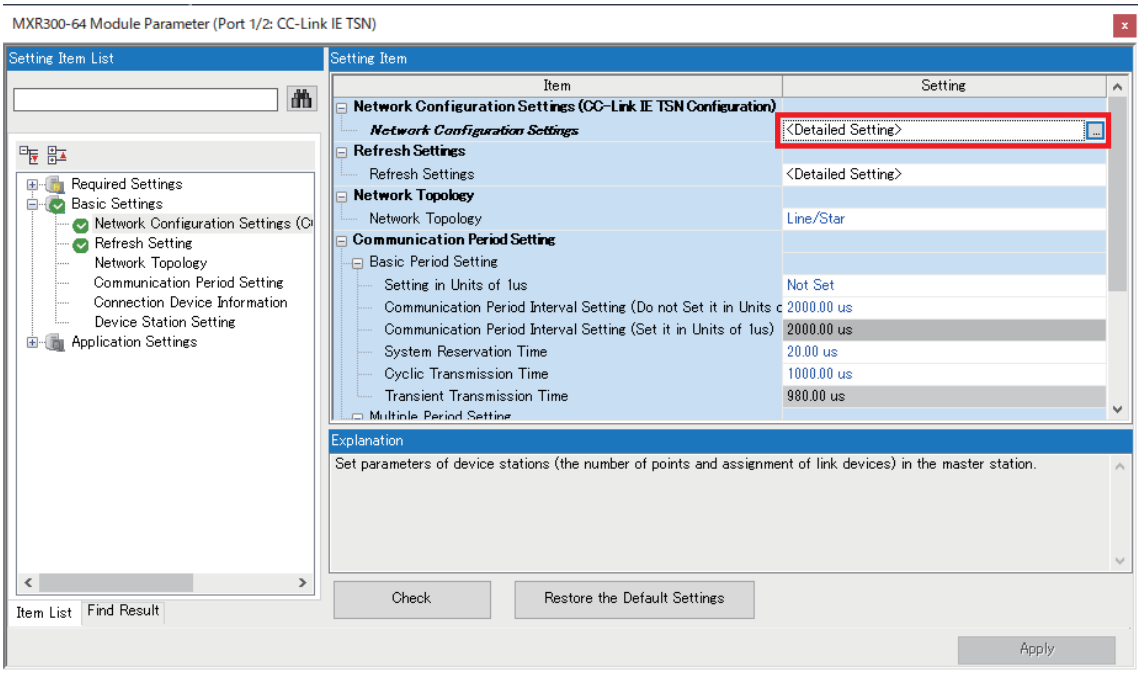
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]



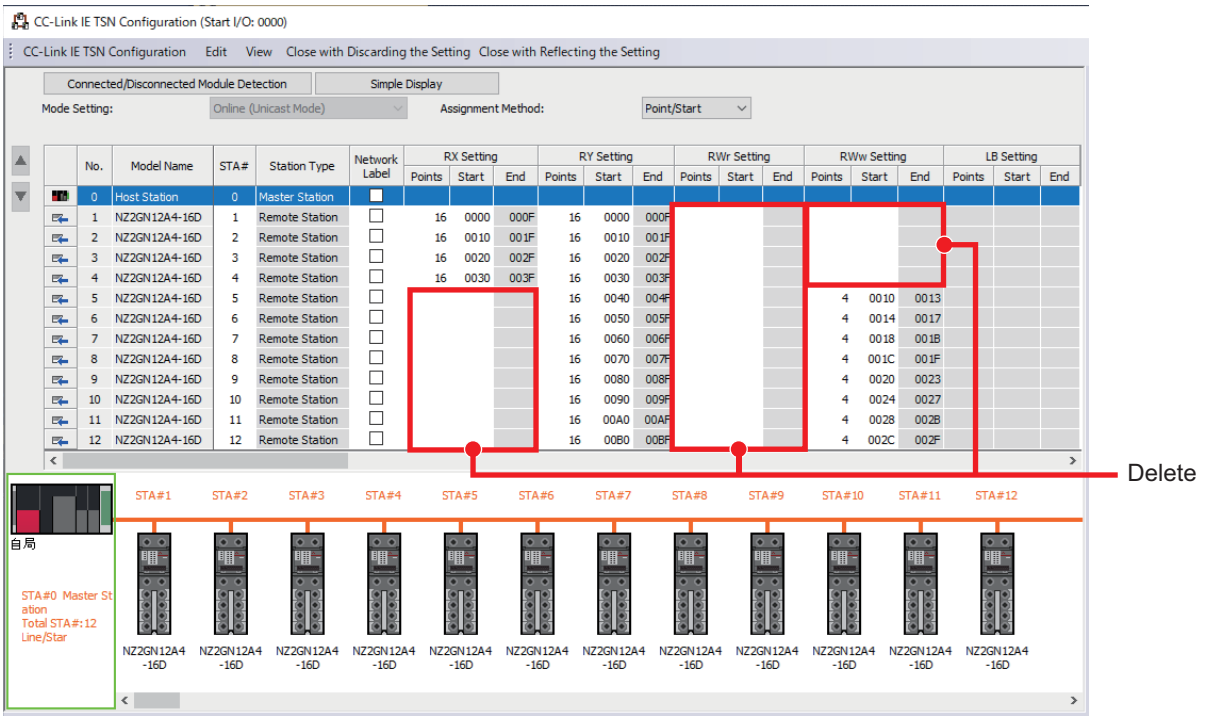
2. In the basic settings, open the network configuration settings.

[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



3. Delete unnecessary RX, RY, RWr, and RWw settings of each device station (state where no device is assigned).

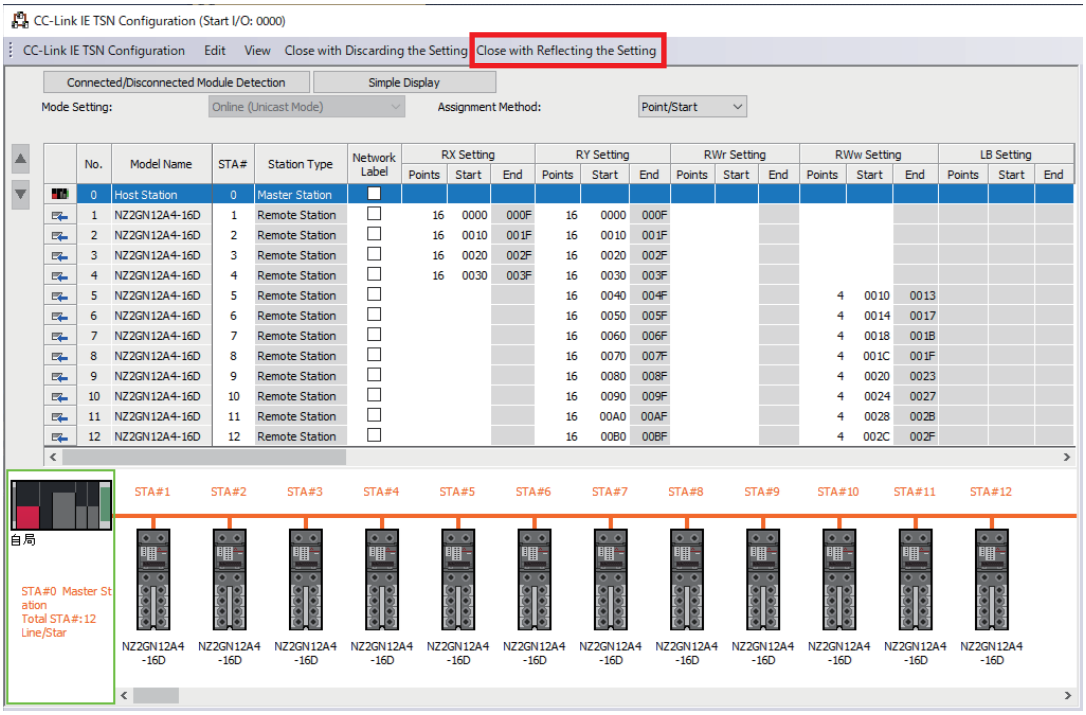
- RX settings of station Nos. 5 to 12: Delete
- RWr settings: Delete
- RWw settings of station Nos. 1 to 4: Delete



Point

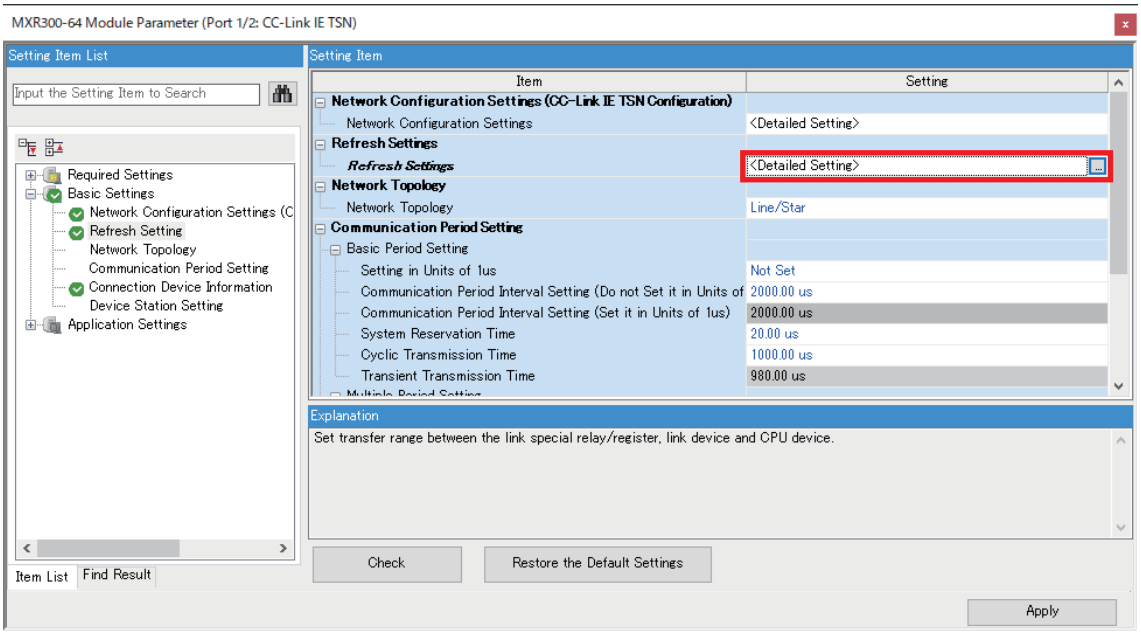
After unnecessary link device settings are deleted, the link refresh time, communication cycle interval, and cyclic transmission time may be reduced further by reassigning the link device numbers.
For details, refer to the following.
☞ Page 96 Reducing Link Refresh Time

4. Click the [Close with Reflecting the Setting] button.



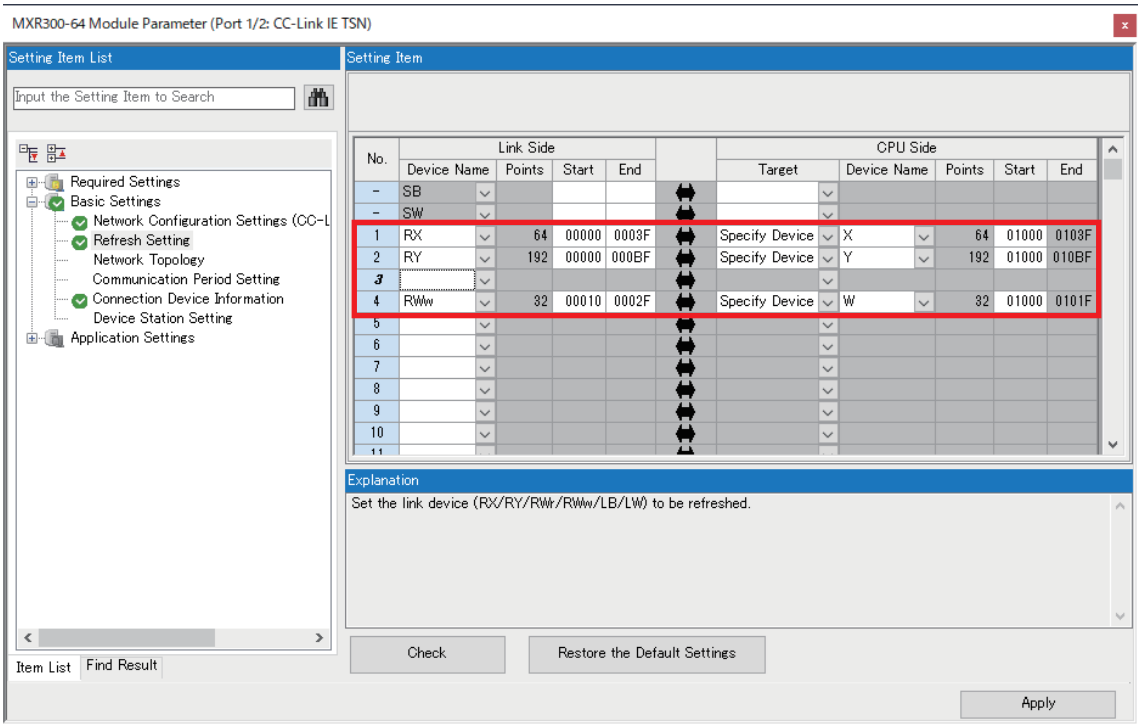
5. Change the refresh settings according to the changes in the link refresh settings.

[Basic Settings] ⇨ [Refresh Setting] ⇨ Double-click <Detailed Setting> of [Refresh Settings]

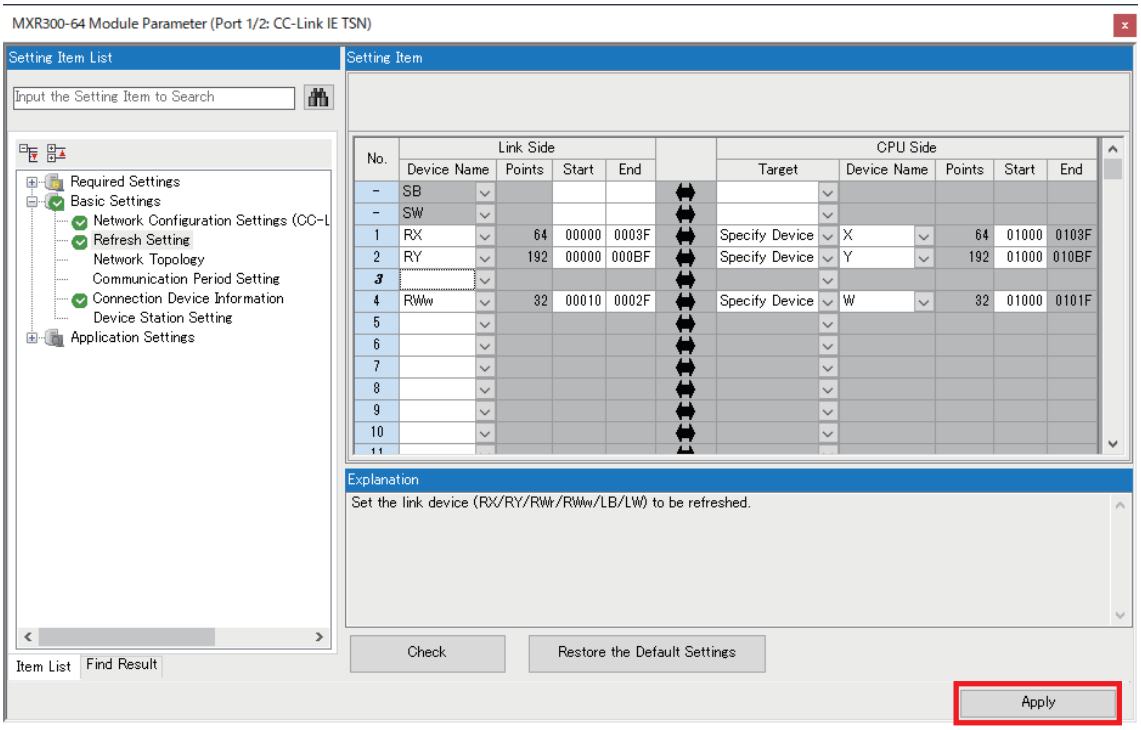


6. Change the start and end numbers of the link side devices according to the link device numbers set in steps 3 and 4. In addition, review the CPU side settings according to the changes in the link side settings.

- Number range of RX settings: 00000 to 0003F
- RWr settings: Delete
- Number range of RWw settings: 00010 to 0002F



7. Click the [Apply] button to apply the setting.



Precautions

- Deleting link device settings may impose restrictions on the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.
- Deleting device stations whose CC-Link IE TSN Class is "CC-Link IE TSN Class A" and communication cycle is set to "Low-Speed" does not have any effect on accelerating CC-Link IE TSN. Check the settings of each device station in the "Network Configuration Settings" window.

7.6 Reducing the Points of Link Device Settings

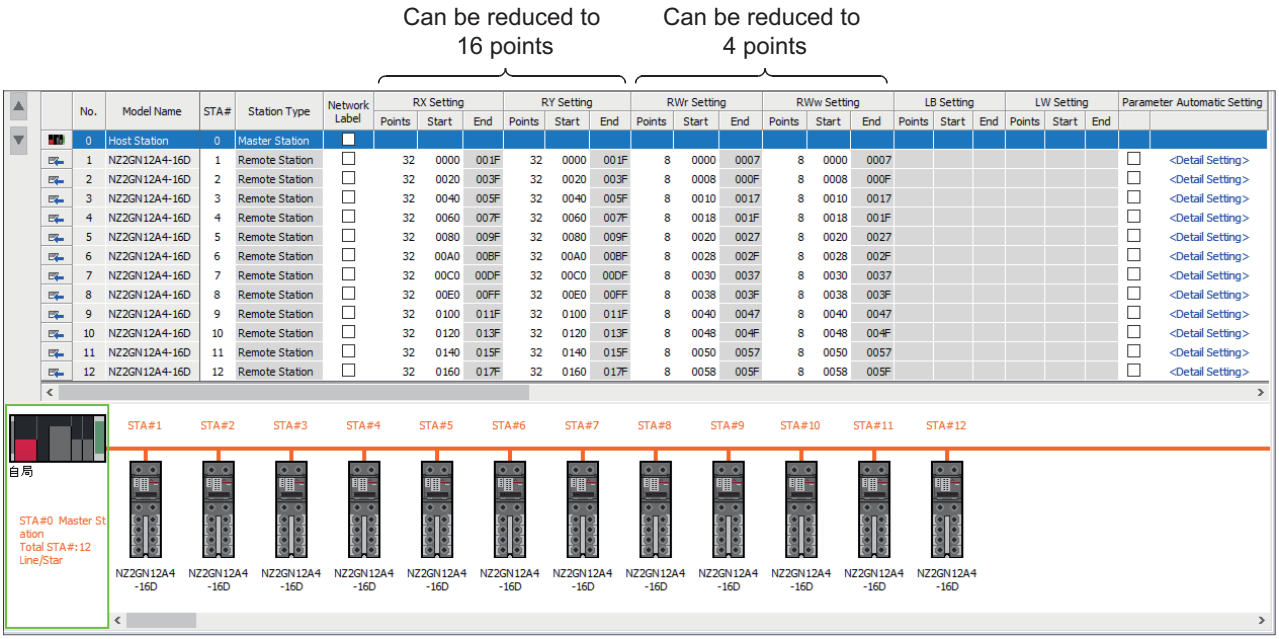
Overview

By significantly reducing the points of link device settings of each device station, the communication period interval and cyclic transmission time can be reduced.

Execution procedure

Using the following network configuration and refresh settings as examples, this section describes the procedure for reducing the points of link device settings.

- Network configuration



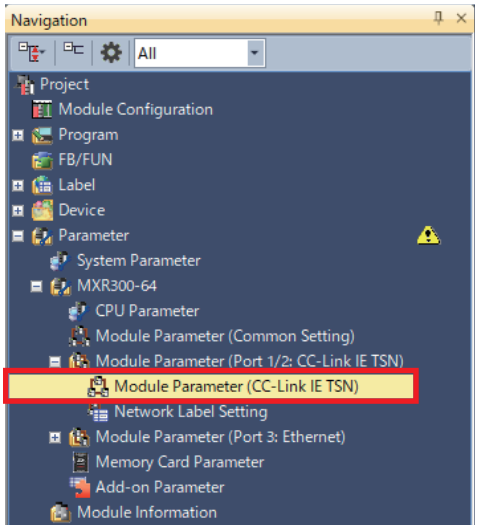
- Refresh setting

No.	Link Side					CPU Side				
	Device Name	Points	Start	End		Target	Device Name	Points	Start	End
-	SB									
-	SW									
1	RX	384	00000	0017F		Specify Device	X	384	01000	0117F
2	RY	384	00000	0017F		Specify Device	Y	384	01000	0117F
3	RWr	96	00000	0005F		Specify Device	W	96	00000	0005F
4	RWw	96	00000	0005F		Specify Device	W	96	01000	0105F

Operating procedure

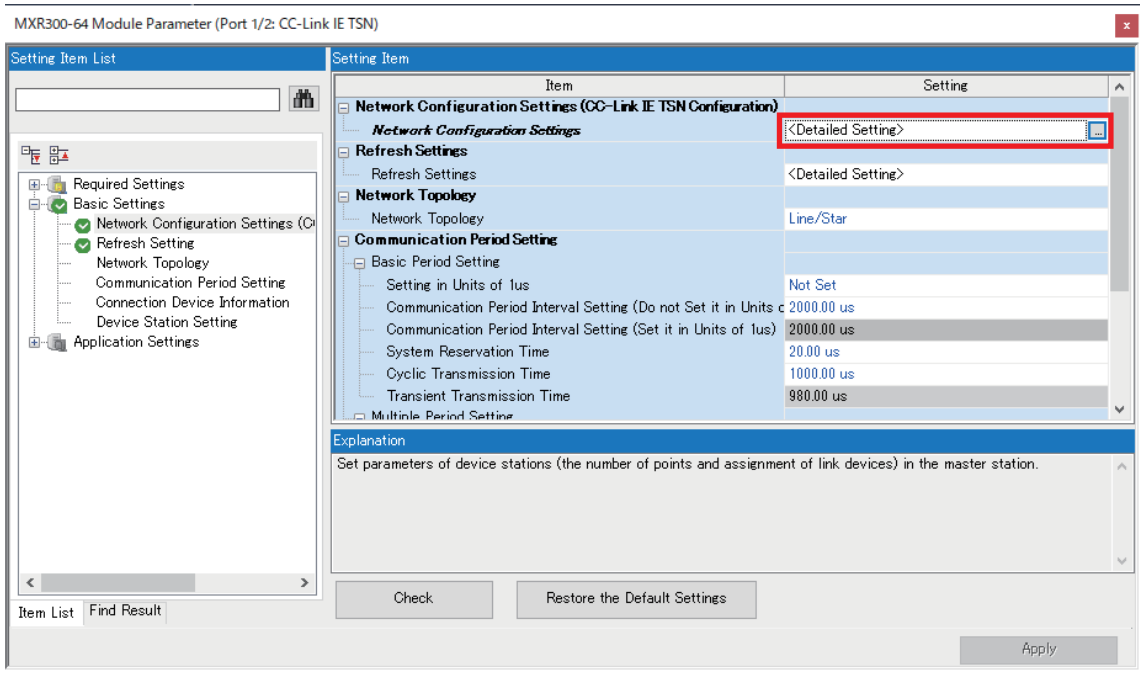
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]



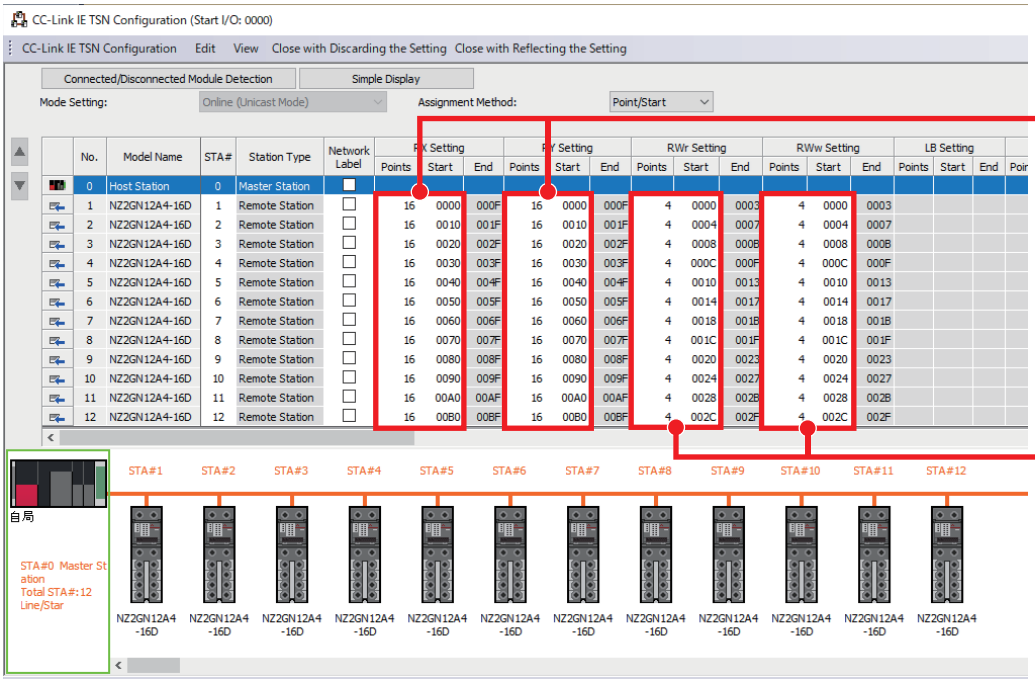
2. In the basic settings, open the network configuration settings.

[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



3. Reduce the points of the RX, RY, RWr, and RWw settings of each device station, and change the start addresses according to the deleted points.

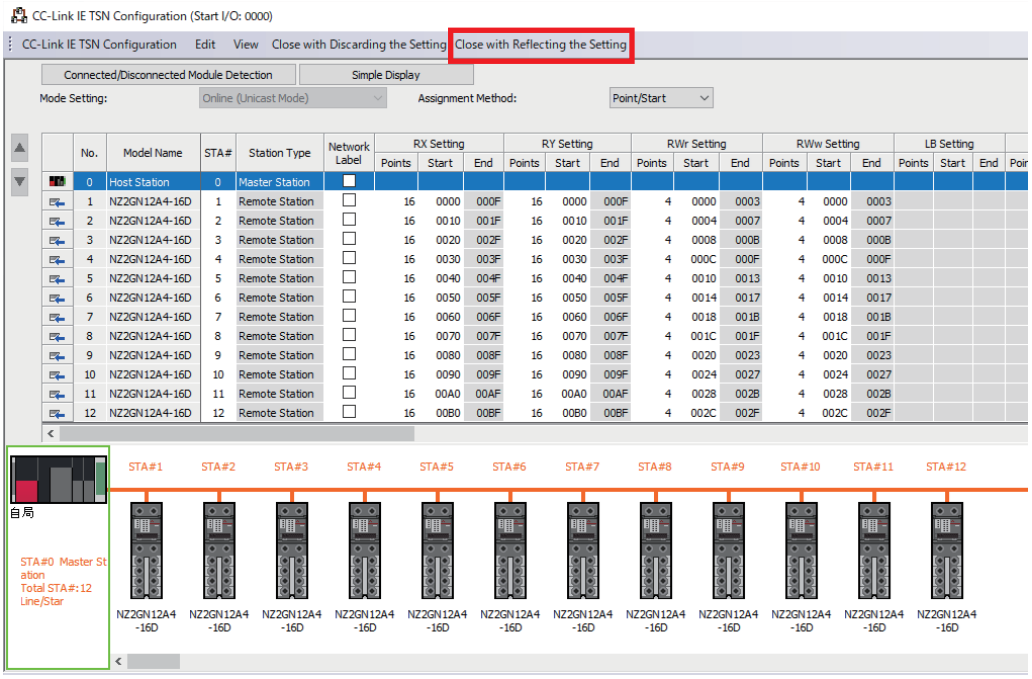
- Points of the RX and RY settings: 16
- Points of the RWr and RWw settings: 4
- Start addresses of the RX, RY, RWr, and RWw settings: Change according to the points



Point

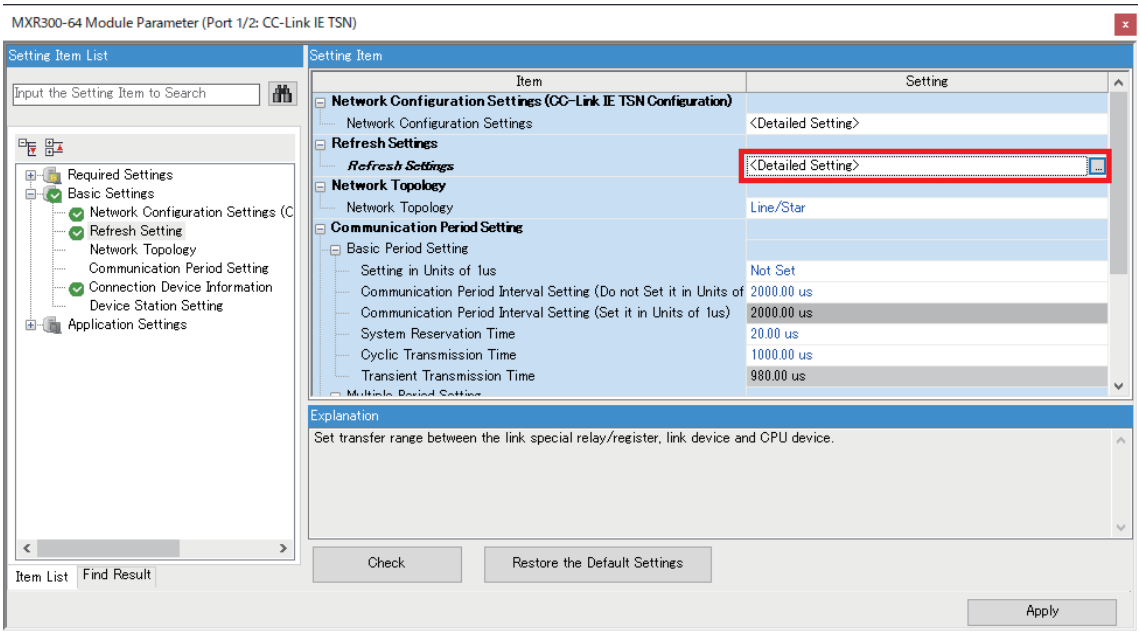
After the points of link device settings are reduced, the link refresh time, communication cycle interval, and cyclic transmission time may be reduced further by reassigning the link device numbers.
For details, refer to the following.
☞ Page 96 Reducing Link Refresh Time

4. Click the [Close with Reflecting the Setting] button.



5. Change the refresh settings according to the changes in the link refresh settings.

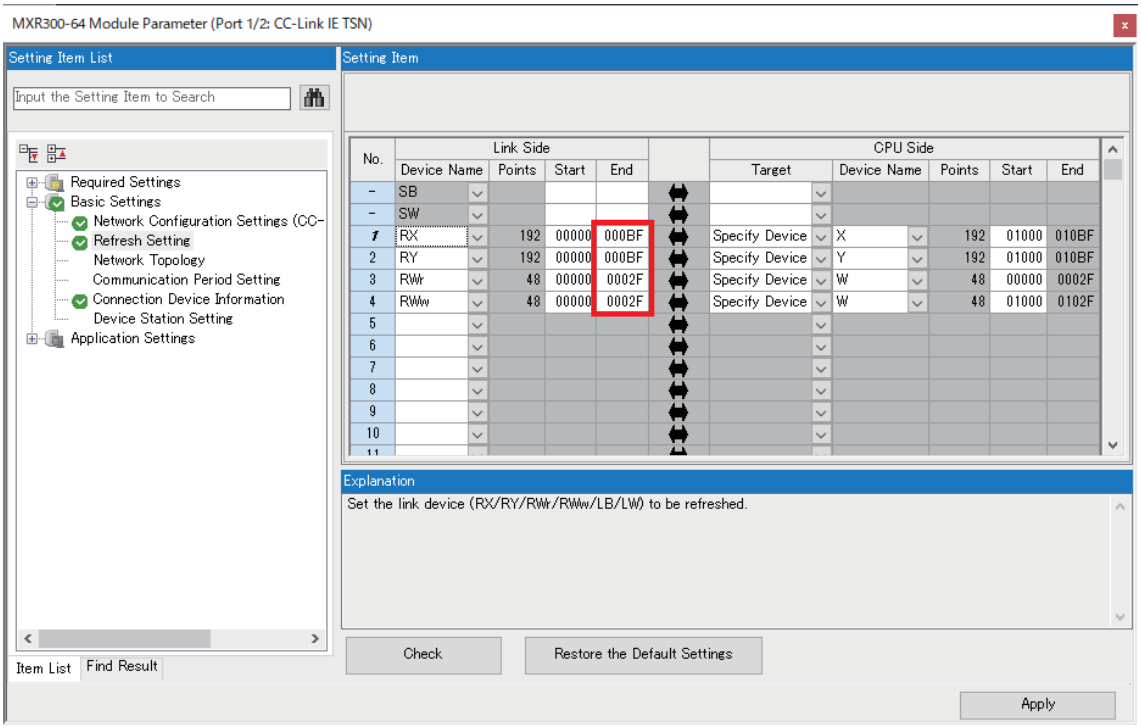
[Basic Settings] ⇨ [Refresh Setting] ⇨ Double-click <Detailed Setting> of [Refresh Settings]



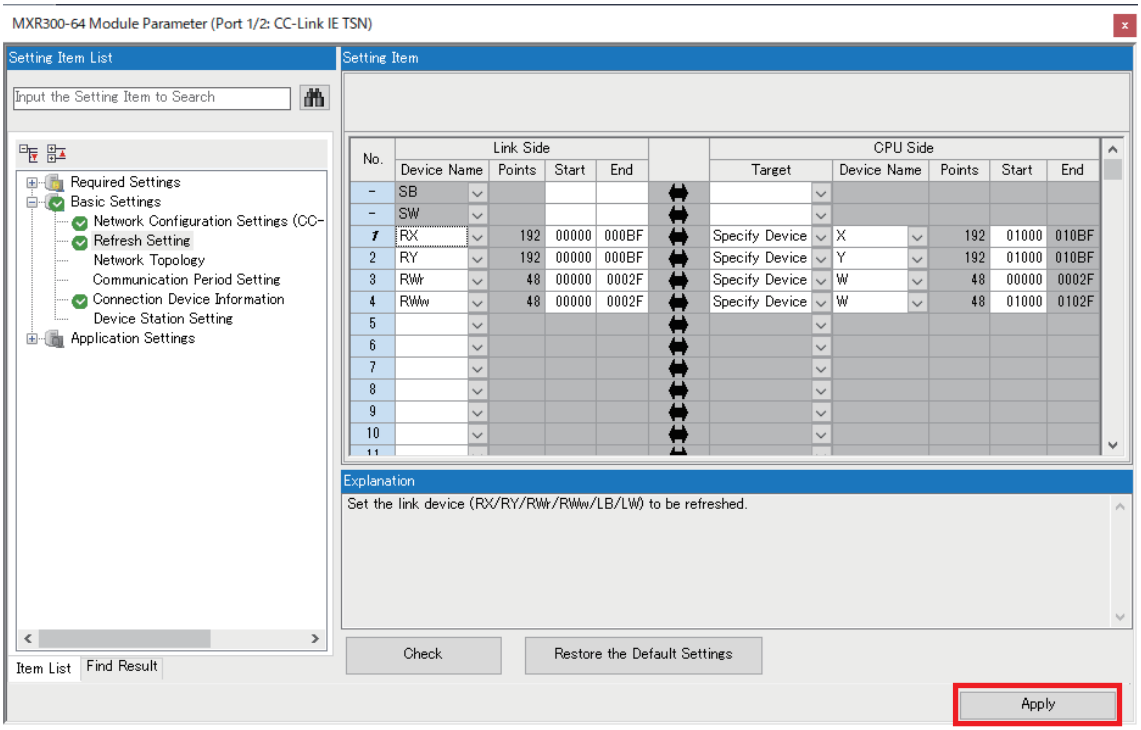
6. Change the end numbers of the link side devices according to the link device numbers set in step 3.

In addition, review the CPU side settings according to the changes in the link side settings.

- End numbers of the RX and RY settings: 000BF
- End numbers of the RWr and RWw settings: 0002F



7. Click the [Apply] button to apply the setting.



Precautions

- Deleting link device settings may impose restrictions on the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.
- Deleting device stations whose CC-Link IE TSN Class is "CC-Link IE TSN Class A" and communication cycle is set to "Low-Speed" does not have any effect on accelerating CC-Link IE TSN. Check the settings of each device station in the "Network Configuration Settings" window.

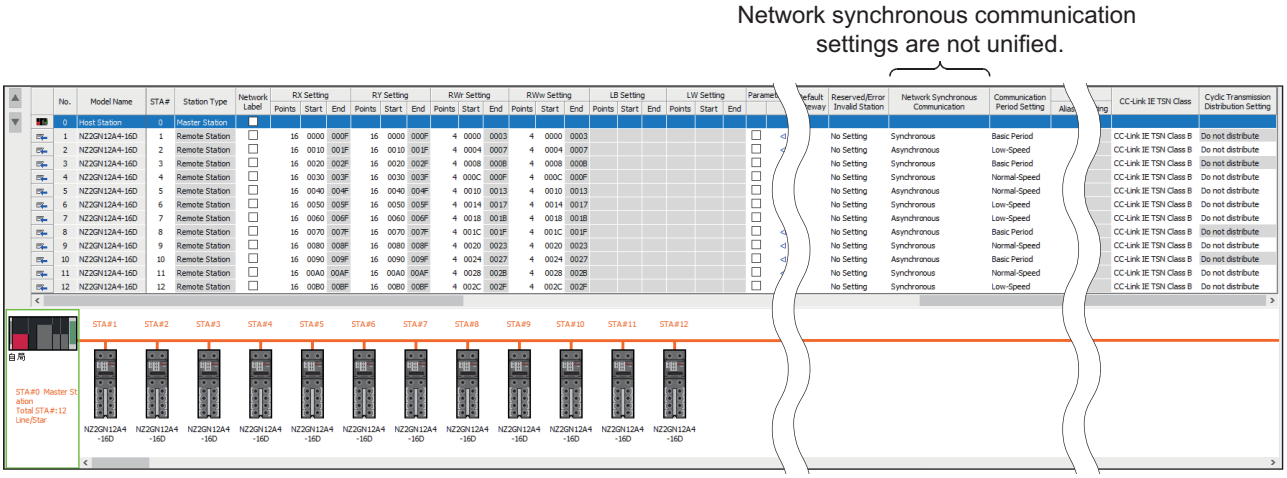
7.7 Unifying Network Synchronous Communication Settings

Overview

If device stations whose network synchronous communication setting is set to "Synchronous" and "Asynchronous" coexist in device stations whose CC-Link IE TSN Class is "Class B", the communication period interval and cyclic transmission time can be reduced by unifying the setting to either of them.

Execution procedure

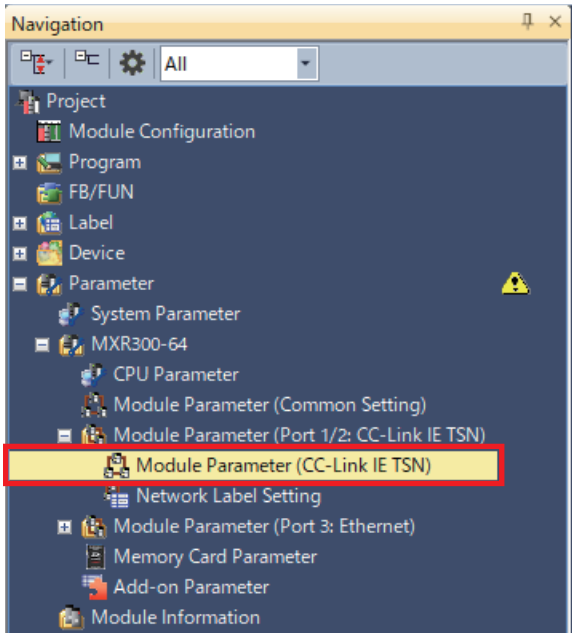
Using the following network configuration as an example, this section describes the procedure for unifying the network synchronous communication settings of device stations whose CC-Link IE TSN Class is "Class B".



Operating procedure

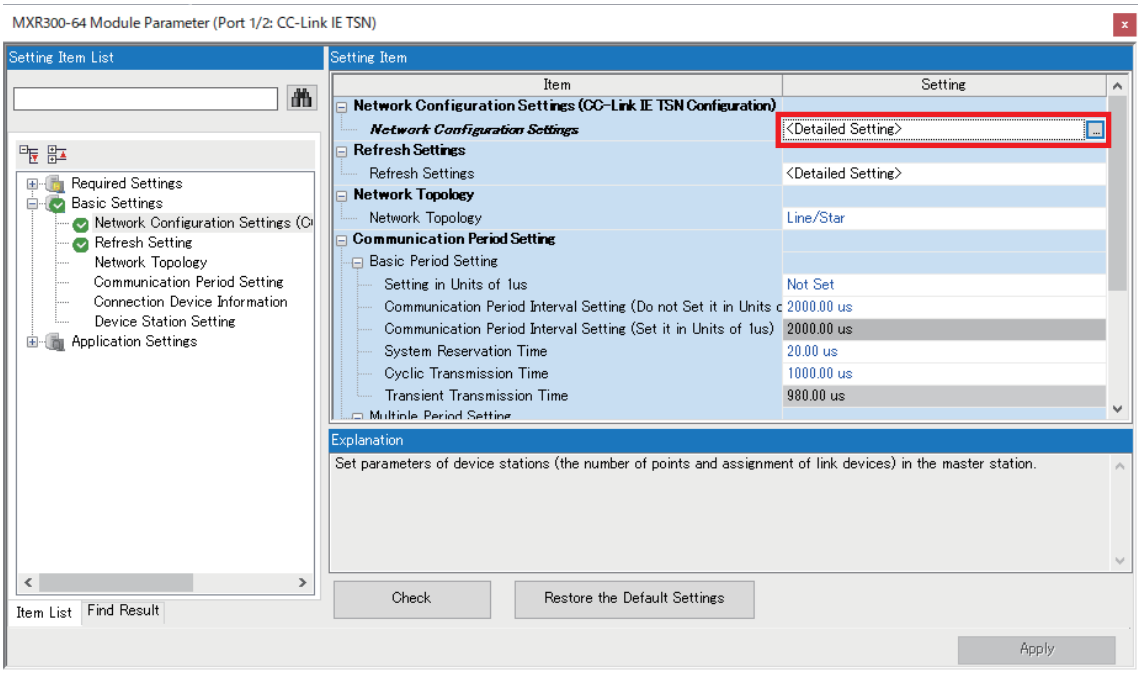
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]

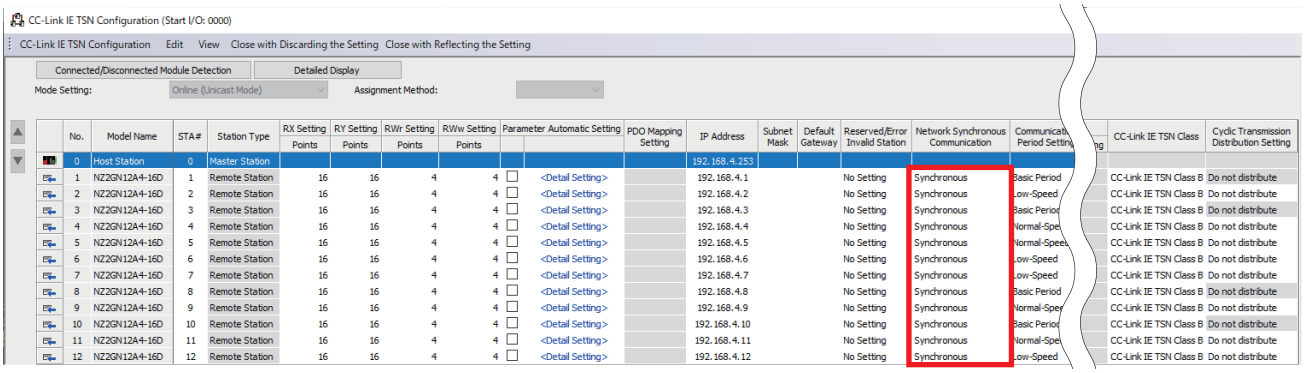


2. In the basic settings, open the network configuration settings.

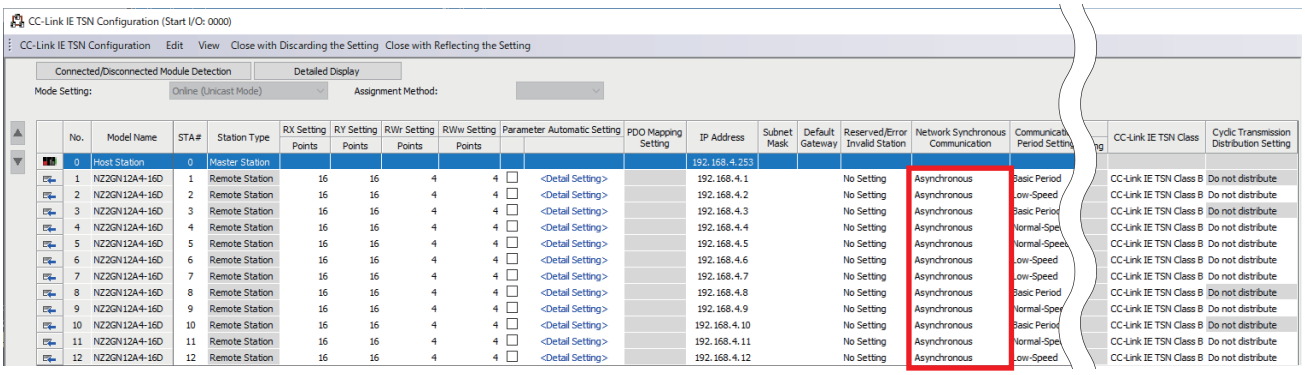
[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



3. Set "Network Synchronous Communication" of device stations whose CC-Link IE TSN Class is "Class B" to "Synchronous" or "Asynchronous".



Or

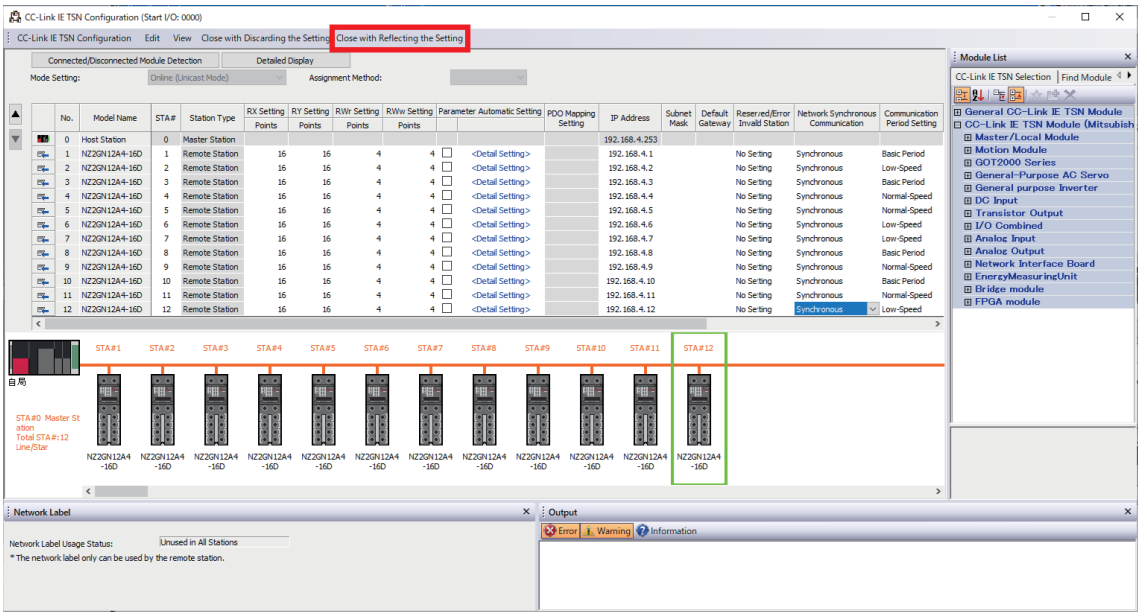


Point

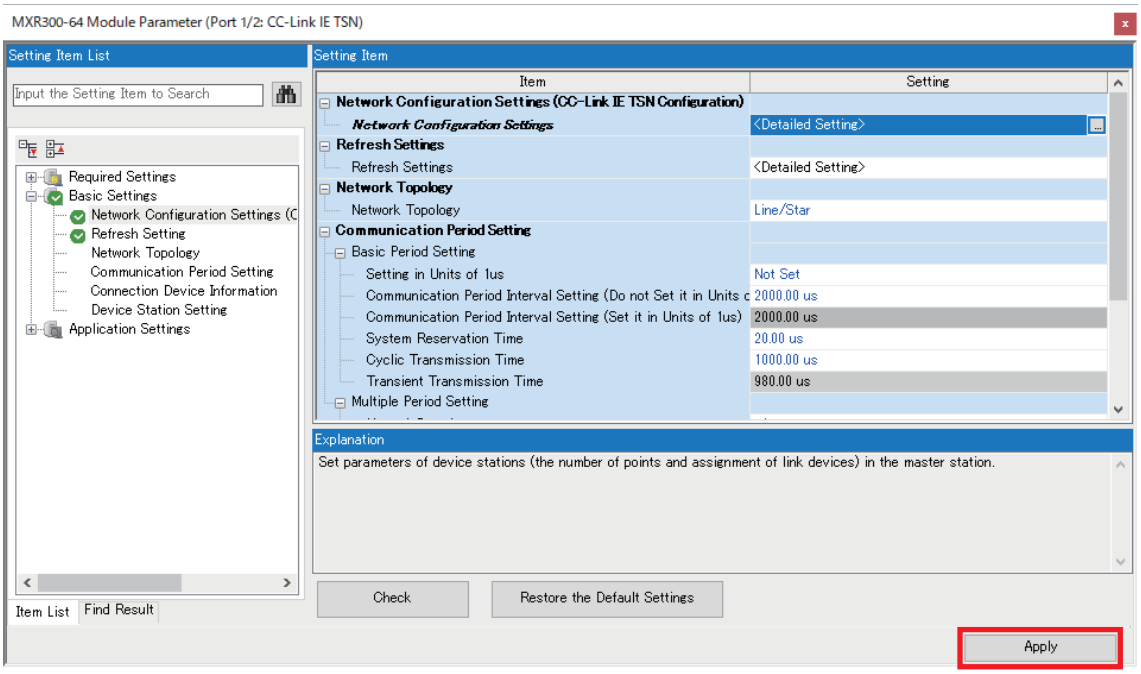
After the network synchronous communication settings are changed, the link refresh time, communication cycle interval, and cyclic transmission time may be reduced further by reassigning the link device numbers. For details, refer to the following.

☞ Page 96 Reducing Link Refresh Time

4. Click the [Close with Reflecting the Setting] button.



5. Click the [Apply] button to apply the setting.



Precautions

Since communication with the network synchronous communication setting set to "Synchronous" can be performed only by device stations whose CC-Link IE TSN Class is "Class B", setting this item may impose restrictions on the device stations used and the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.

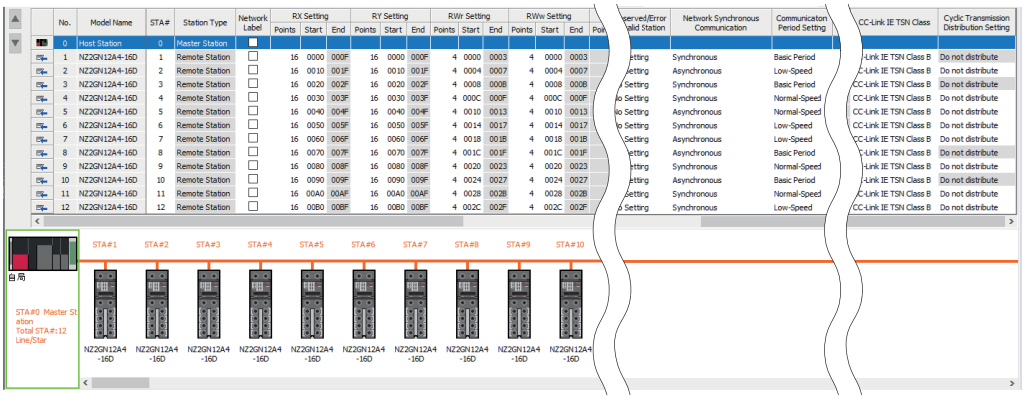
7.8 Setting Distribution of Cyclic Transmission

Overview

For device stations whose CC-Link IE TSN Class is "Class B" and communication cycle is set to "Normal-Speed" or "Low-Speed", the communication period interval for the basic cycle and cyclic transmission time can be reduced by distributing the timing of cyclic transmissions for each station within the communication cycle.

Execution procedure

Using the following network configuration as an example, this section describes the procedure for distributing the timing of cyclic transmissions for device stations whose CC-Link IE TSN Class is "Class B" and communication cycle is set to "Normal-Speed" or "Low-Speed".

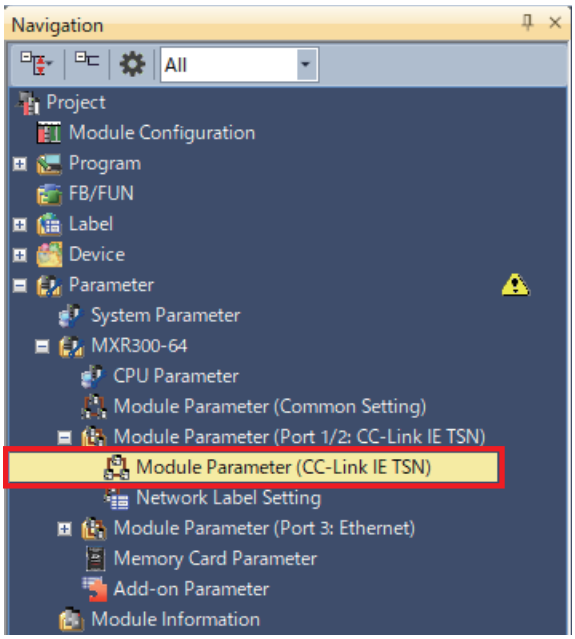


The cyclic transmission distribution setting is set to "Do not distribute".

Operating procedure

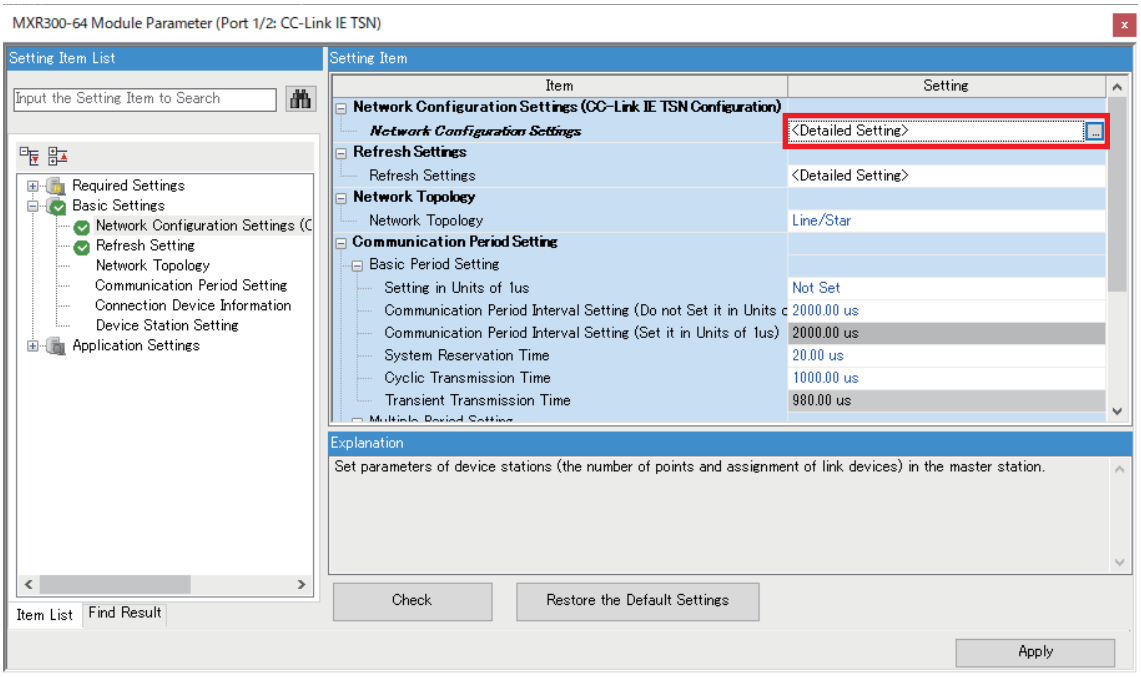
1. Open module parameters (CC-Link IE TSN).

Navigation window ⇨ [Parameter] ⇨ Model name ⇨ [Module Parameter (Port 1/2: CC-Link IE TSN)] ⇨ Double-click [Module Parameter (CC-Link IE TSN)]

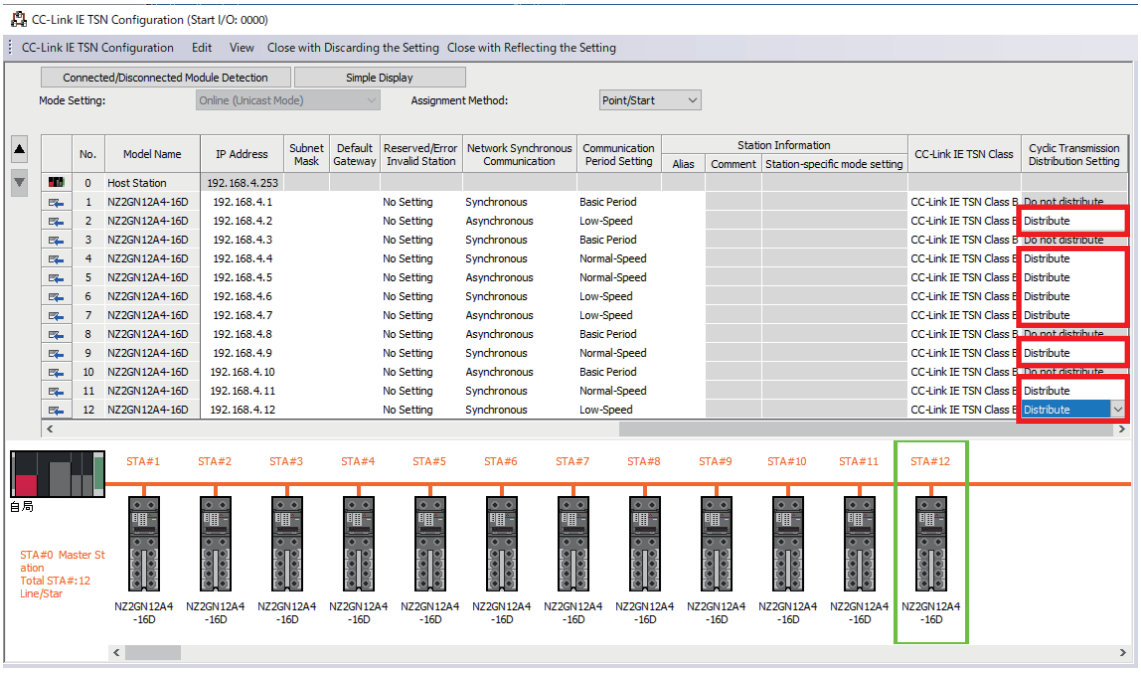


2. In the basic settings, open the network configuration settings.

[Basic Settings] ⇨ [Network Configuration Settings (CC-Link IE TSN Configuration)] ⇨ Double-click <Detailed Setting> of [Network Configuration Settings]



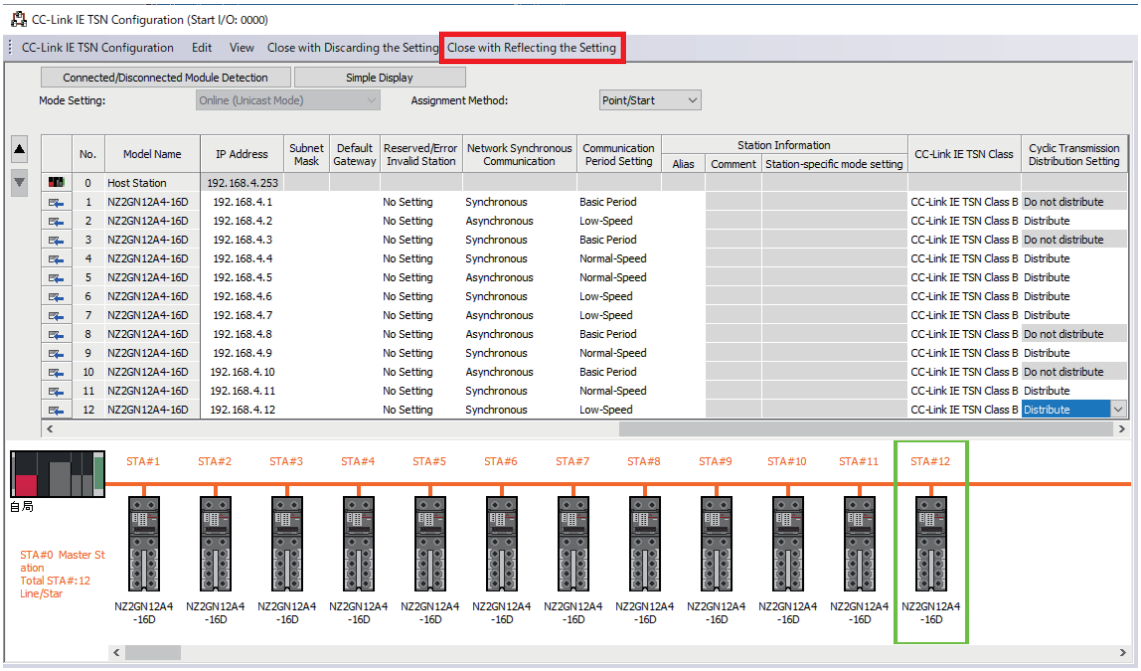
3. Set "Cyclic Transmission Distribution Setting" to "Distribute" for device stations whose CC-Link IE TSN Class is "Class B" and communication cycle is set to "Normal-Speed" or "Low-Speed".



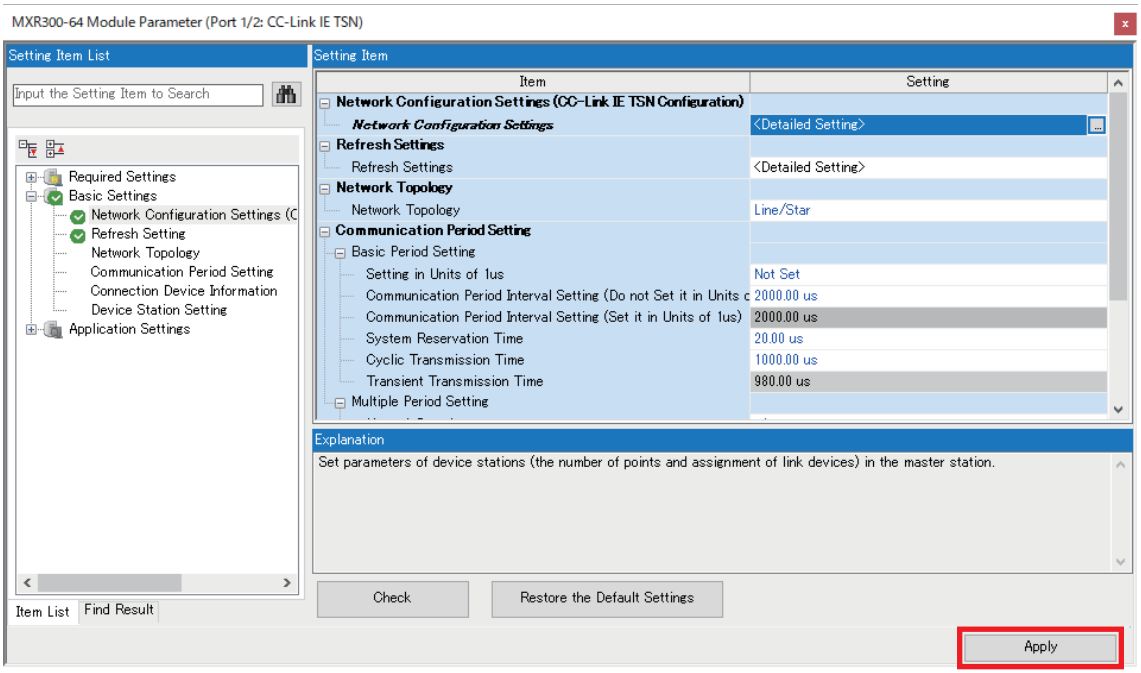
Point

If the "Cyclic Transmission Distribution Setting" column is not displayed, click the [Detail Display] button to display it.

4. Click the [Close with Reflecting the Setting] button.



5. Click the [Apply] button to apply the setting.



Precautions

Since the cyclic transmission distribution can be set only for device stations whose CC-Link IE TSN Class is "Class B" and communication cycle is set to "Normal-Speed" or "Low-Speed", setting this item may impose restrictions on the device stations used and the feasible system configuration. Determine whether this technique can be used depending on the system to be configured.

MEMO

REVISIONS

*The manual number is given on the bottom left of the back cover.

Revision date	*Manual number	Description
June 2025	BCN-89999-9836-A	First edition

Japanese manual number: BCN-89999-9684-B

INFORMATION AND SERVICES

For further information and services, please contact your local Mitsubishi Electric sales office or representative.
Visit our website to find our locations worldwide.

MITSUBISHI ELECTRIC Factory Automation Global Website
Locations Worldwide
www.MitsubishiElectric.com/fa/about-us/overseas/

TRADEMARKS

The company names, system names, and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.
In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

BCN-89999-9836-A

BCN-89999-9836-A(2506)

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: TOKYO BLDG., 2-7-3, MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS: 1-14, YADA-MINAMI 5-CHOME, HIGASHI-KU, NAGOYA 461-8670, JAPAN