M0020965

SANMOTION

MOTION CONTROLLER

C

SMC200-A, SMC200-B

Motion controller

Software manual

SANYO DENKI

1. Preface

1.1. Introduction

Thank you for purchasing the motion control "SANMOTION C" SMC200. This manual “Motion controller SMC200-A/SMC200-B" (hereinafter referred to as S200) describes the software including the important matters that must be aware of when using this product in order to protect customers' safety. Please read the documentation and related instruction manuals carefully before using, and ensure fully understand the function and performance of the product and use it properly.

The Products presented in this manual are meant to be used for general industrial applications. As this is designed and manufactured for general industrial applications. Therefore, we exclude the application for the followings such as equipment and systems for special applications.

- Do not use for medical devices and other equipment affecting people’s lives
- Do not use for that have significant effects on society and the general public
- Do not use in an environment where vibration is present, such as in a moving vehicle or shipping vessel
- Do not use for special applications related to aviation and space, nuclear power, electric power, submarine repeaters

However, even in the above-mentioned applications, we may allow products to be applied on conditions for such cases that limited specific usage or require no special quality (Quality no beyond general specification etc.). Please contact us beforehand.

- CODESYS® is a registered trademark of CODESYS GmbH.
- EtherCAT ® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH.
- Ethernet is a registered trademark of FUJIFILM Business Innovation Corp.
- 1-Wire is a registered trademark of Analog Devices, Inc.

1.2. Precautions related to these Instructions

In order to fully understand the functions of this product, please read this instruction manual thoroughly before using the product. After thoroughly reading the manual, keep it handy for reference.

Although the manufacturer has taken all possible measures to ensure the veracity of the contents of this manual, should you notice any error or omission, please notify your local sales office or the head office of your findings.

You are strictly prohibited to use (including, without limitation, copying, modifying, reproducing in whole or in part, uploading, transmitting, distributing) any part or all of the manual.

Carefully and completely follow the safety instructions outlined in this manual. Note that safety is not guaranteed for usage methods other than those specified in this manual or those methods intended for the original product.

Permission is granted to reproduce or omit a portion of the attached figures (as abstracts) for use.

The contents of this manual may be modified without prior notice as revisions or additions are created regarding the usage method of the product. Modifications are performed as per the revisions of this manual.

1.3. Documentation for further reading

The S200 is designed with various interfaces for configuring systems by combining necessary peripheral devices according to your functional requirements. For the details of the function, please refer to the instruction manual of the hardware as well.

No.	Title
M0020716	SMC200-A, SMC200-B Motion Controller Hardware manual
M0020996	SMC-USBW-01 Wireless adapter 3A Instruction Manual Combination with S200 series
M0020986	SMC200-A, SMC200-B Motion Controller Web Application Instruction Manual

1.4. About CODESYS

The S200 controller software is implemented based on CODESYS. Therefore, when using the S200, please also refer to the "CODESYS Online Help" below.

CODESYS Online Help ：「https://www.helpme-codesys.com/」

Henceforth, when "online help" is mentioned in this manual, it means CODESYS online help.

2. Safety notes

2.1. Representation

At various points in this manual you will see notes and precautionary warnings regarding possible hazards. The symbols used have the following meaning:

DANGER!

- indicates an imminently hazardous situation which will result in death or serious bodily injury if the corresponding precautions are not taken.

WARNING!

- indicates a potentially hazardous situation which can result in death or serious bodily injury if the corresponding precautions are not taken.

CAUTION!

- means that if the corresponding safety measures are not taken, a potentially hazardous situation can occur that may result in property injury or slight bodily injury.

CAUTION

- CAUTION used without the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in damage to property.

- This symbol reminds you of the possible consequences of touching electrostatically sensitive components.

Information

Useful practical tips and information on the use of equipment are identified by the "Information" symbol. They do not contain any information that warns about potentially dangerous or harmful functions.

3. Installation of development software

Please install SANMOTION C development software according to the following procedure.

Administrator authority is required for installation.

Contact your system administrator for more information.

3.1. PC environment

- CPU : 2.5 GHz or higher
- Memory : 8 GB or more
- Hard disk : 12 GB or more free space
- Ethernet port or USB port
- OS : Windows10 (32/64 Bit), Windows11

3.2. Run the installer

The development software installer has the following structure.

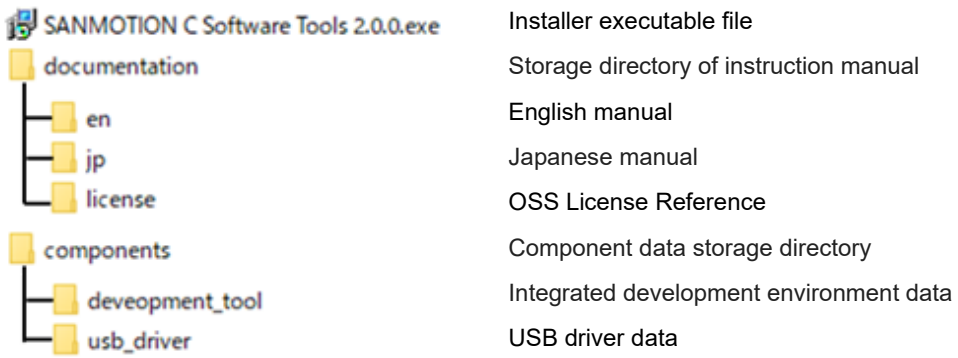


Fig 3.1 Installer configuration

1. Right-click "SANMOTION C Software Tools 2.0.0.exe" in the installer and click "Run as administrator".

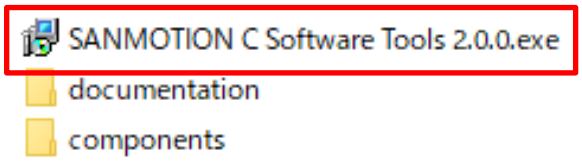


Fig 3.2 Run the installer

2. The language selection window will be displayed. Select “Japanese” or “English” and click “OK”.

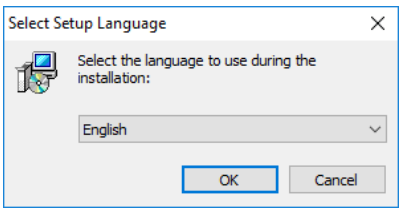


Fig 3.3 Select setup language window

3. The component selection window will be displayed. Select the component you want to install and click “Next (N)>“.

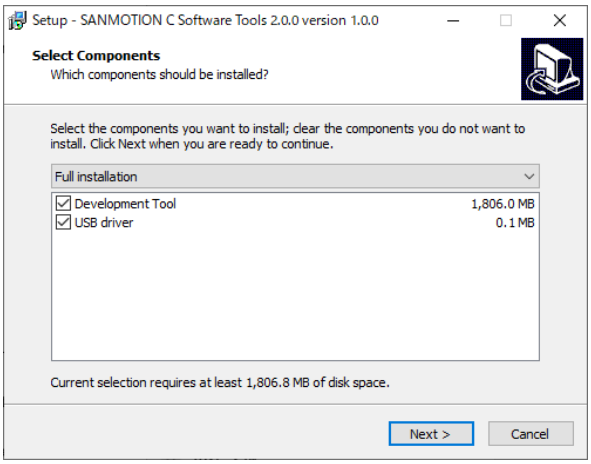


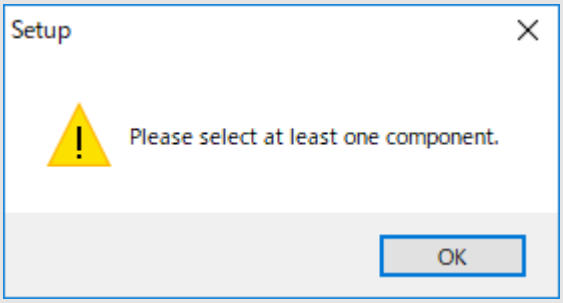
Fig 3.4 Select components window

Components can be selected with a dialog box at the top center or a check box at the top of each component name.

The following two types of dialog boxes are available. Selectable components are as follows.

dialog box	Integrated development environment	USB driver
Full installation	✓	✓
Custom installation	Any	Any

Please select at least one component. If it is not selected and you click “Next (N)>“, the following message will be displayed and stay in the component selection window



4. The confirmation window of installation setting is displayed. If the settings are correct, please click “Install”.

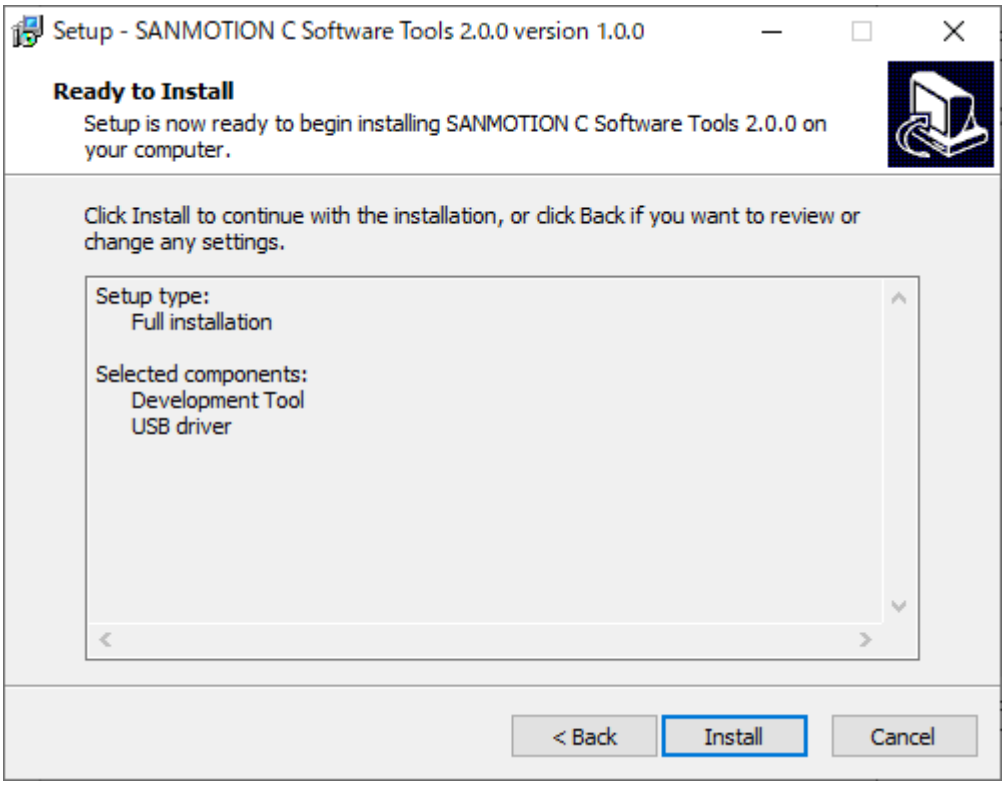


Fig 3.5 Installation start window

5. The installation will start.

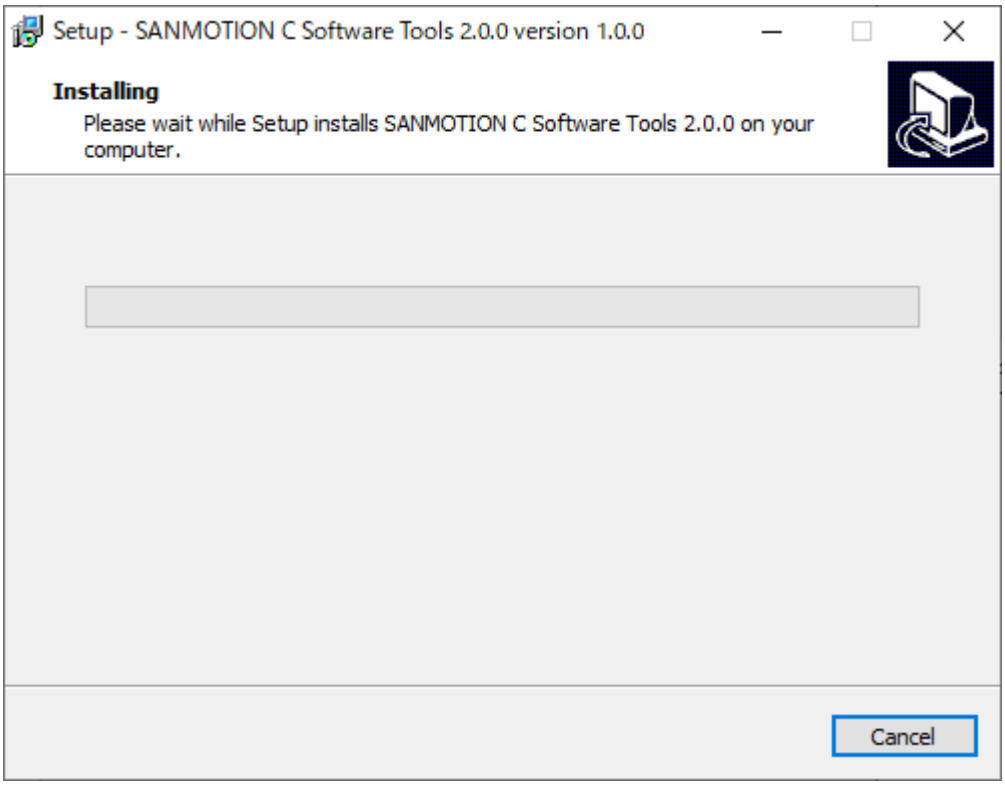


Fig 3.6 Windows during installation

6. Then, the installer of the integrated development tool starts up. Please click “NEXT>“ button.

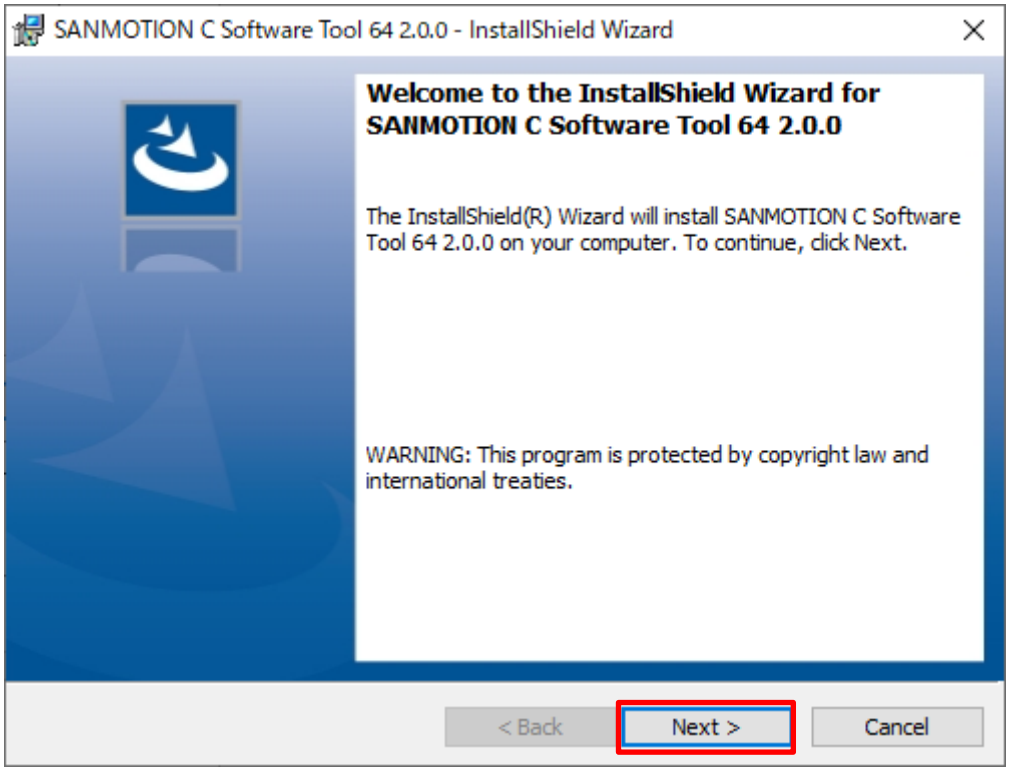


Fig 3.7 Integrated development tool installation start window

7. The license agreement screen is displayed. Check the contents, put a check in the “I accept the terms in the license agreement”, please click “NEXT>“.

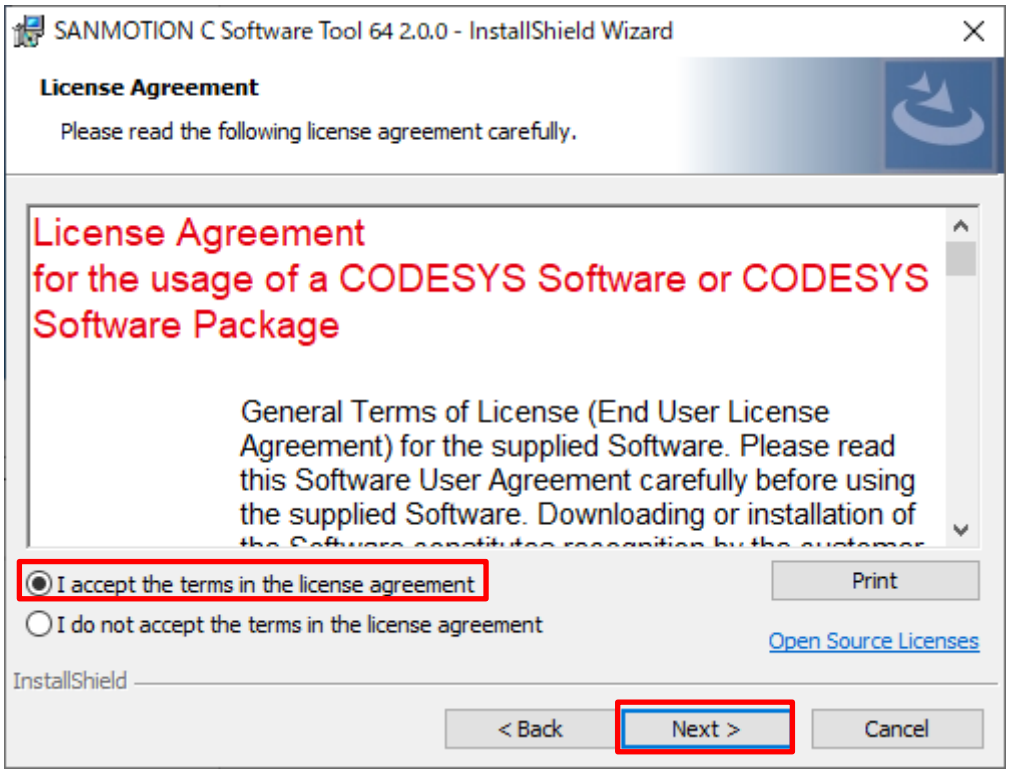


Fig 3.8 License Agreement

8. The Very important information screen is displayed. Please check “I have read the infomatin” after confirmation, and click “NEXT>“.

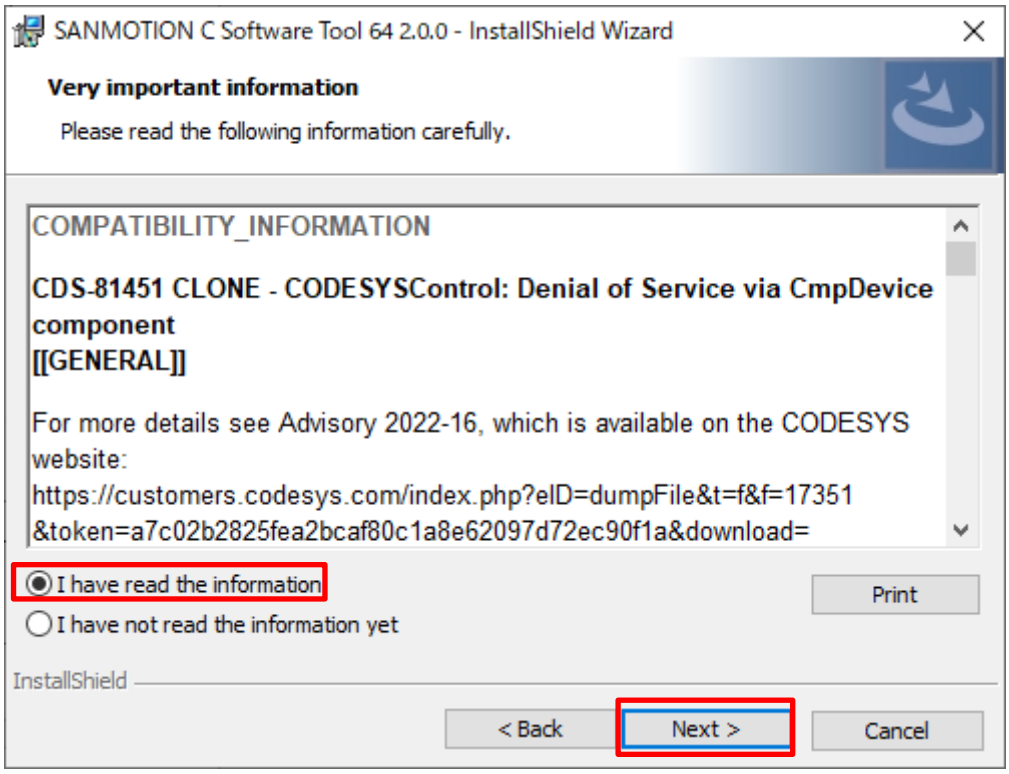


Fig 3.9 Very important information

9. As the screen for setting the save destination of the software is displayed, click “NEXT>“ after confirmation. If you want to change the save destination, you can change it from “Change ...”.

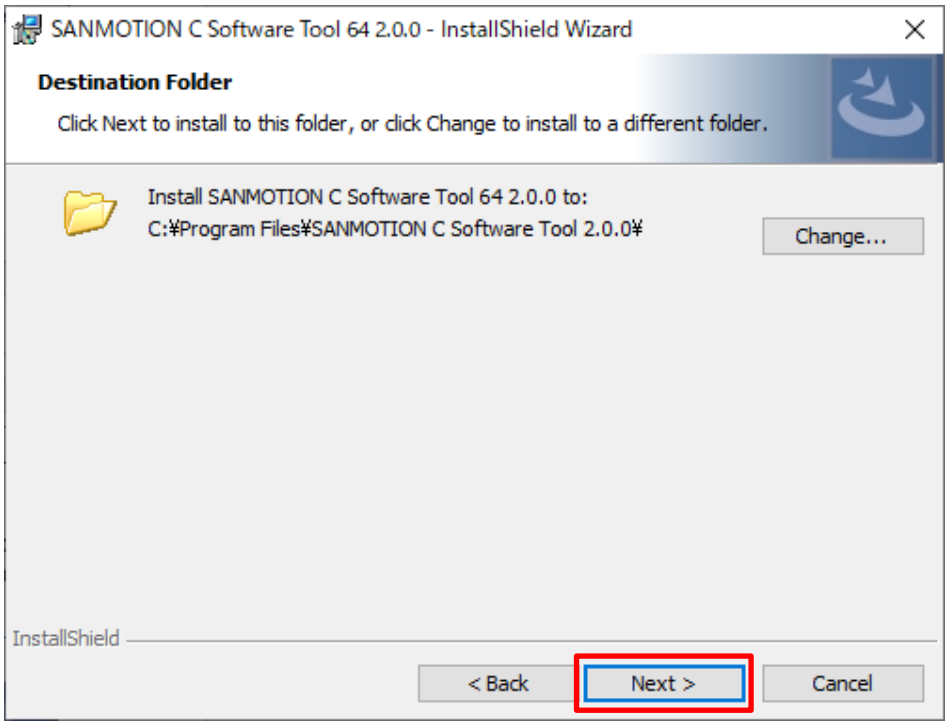


Fig 3.10 Save destination selection window

10. As the screen for setting the installation type of the component is displayed, please select “Complete” and click “NEXT>“.

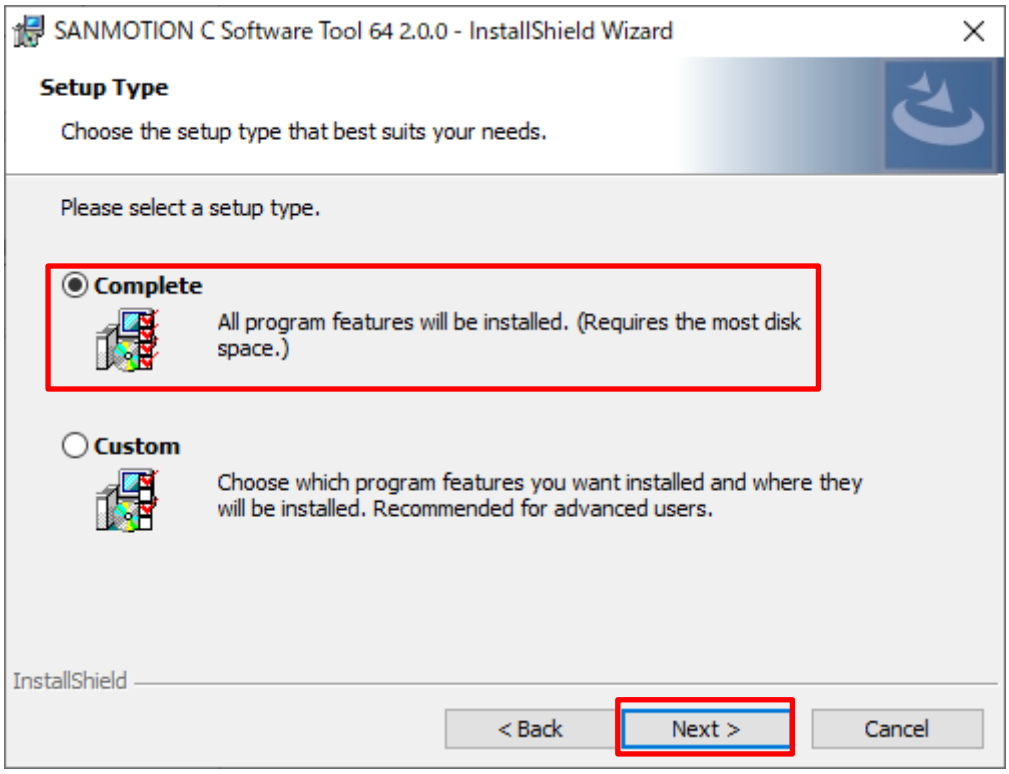


Fig 3.11 Installation type selection window

11. The confirmation screen of the installation setting is displayed. If the settings in steps 5 to 6 are correct, click “Install”.

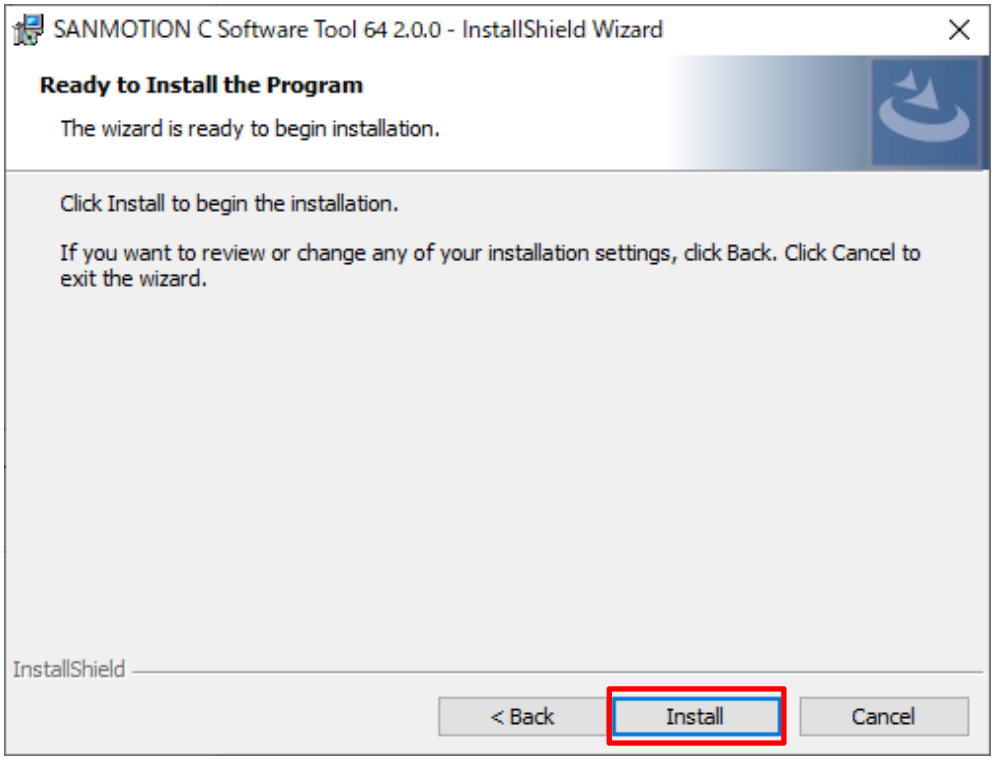


Fig 3.12 Installation start screen

12. Installation of the integrated development tool will start.

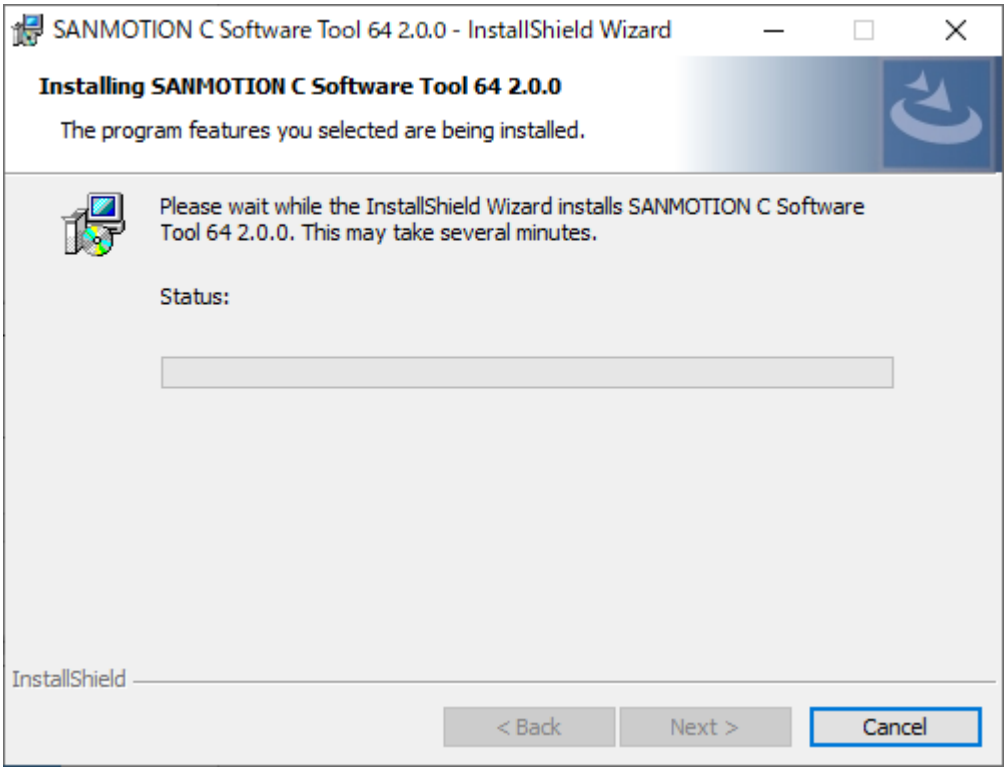


Fig 3.13 Screen during installation

13. When the installation of the integrated development tool is completed normally, the following window will be displayed, please click “Finish”.

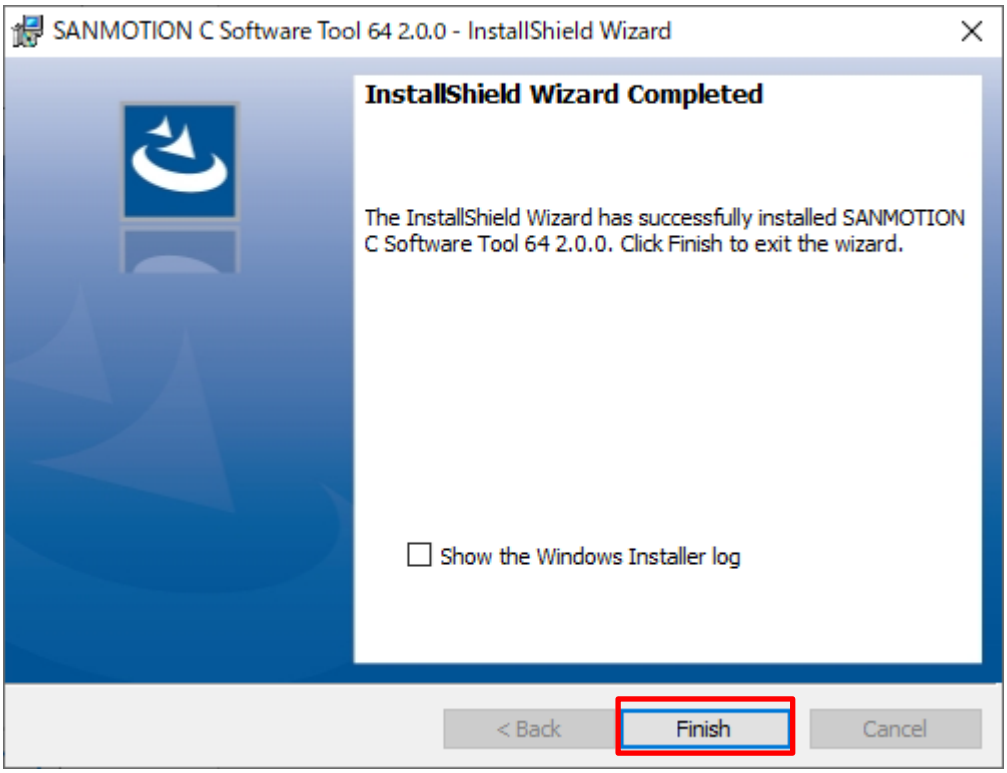


Fig 3.14 Installation complete screen

14. Installation of the USB driver is started after installation of the integrated development tool is completed.

15. When the installation of the USB driver is completed normally, the following window will be displayed. A computer restart is required to complete the installation. If there is no problem in restarting immediately, check "Restart immediately". If you want to manually restart later, check "Manual restart later". Then click "Finish".

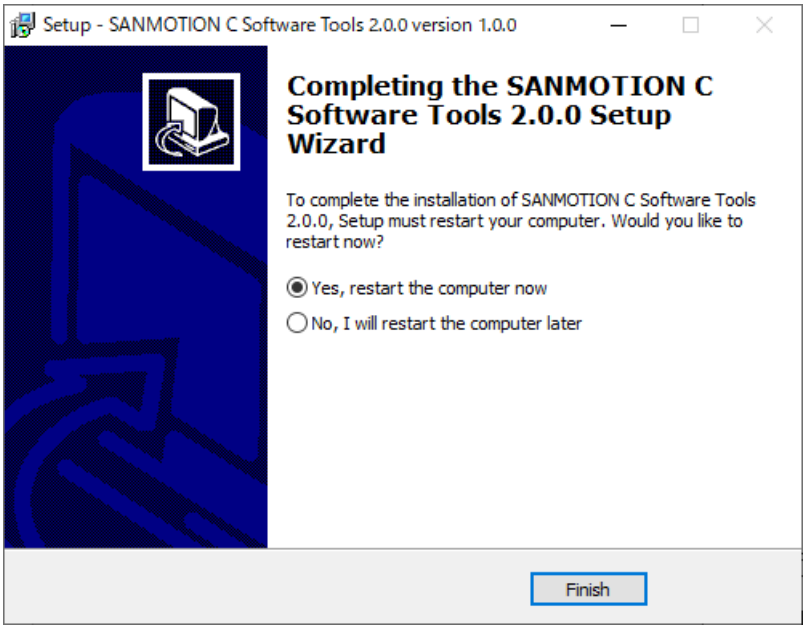


Fig 3.15 Screen when installation of USB driver is completed

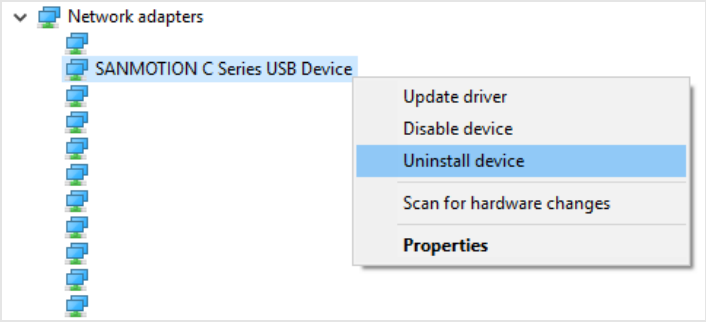
This completes the installation of the development software.

3.3. When it does not work

Please uninstall the “SANMOTION C Software Tools 2.0.0 version 1.0.0“ from “Uninstall the program” in the Windows control panel and install it again.

If you do not operate normally even after performing the above procedure please contact us.

USB driver is not completely uninstalled in “Uninstall a program”. To uninstall completely, please execute “uninstall device” from “Device Manager”.



4. SANMOTION C Software Tool 2.0.0

4.1. What is SANMOTION C Software Tool 2.0.0

SANMOTION C Software Tool 2.0.0 is an integrated development tool that can develop motion / PLC program, program debugging and hardware configuration conforming to international standard IEC 61131-3.

The program language supports six types (LD, FBD, CFC, ST, SFC, IL) conforming to IEC 61131-3, and it is possible to combine program languages suitable for control contents.

4.2. Template file

The SANMOTION C project is created from a template file. You can find the template file in the "SMC200-* Template Project" category in the "New Project" window. Select "PLC standard project" when creating a sequence program, and select "Motion standard project" when creating a motion program.

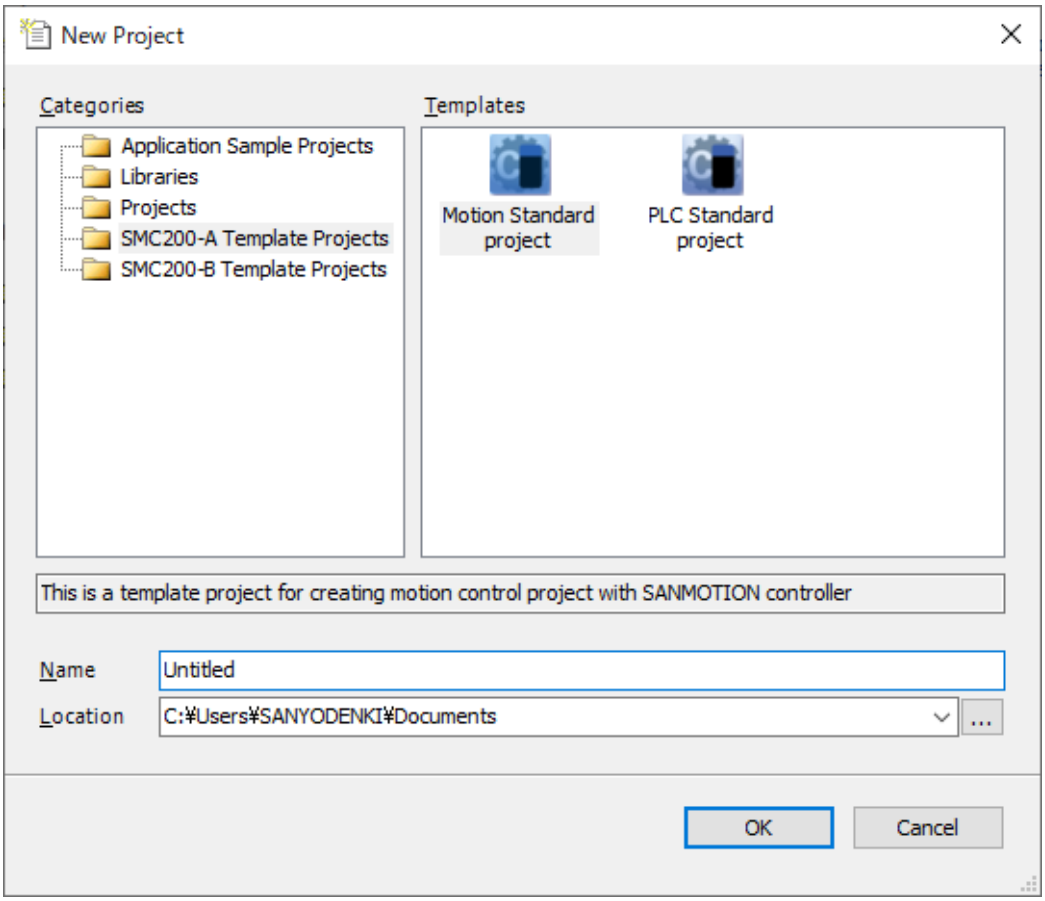


Fig.4.1 Template file selection window

In the template file, when using S200, the project with the minimum necessary setting is incorporated. Therefore, users can create projects without being conscious of EtherCAT settings, I/O control settings, etc.

4.3. Screen structure

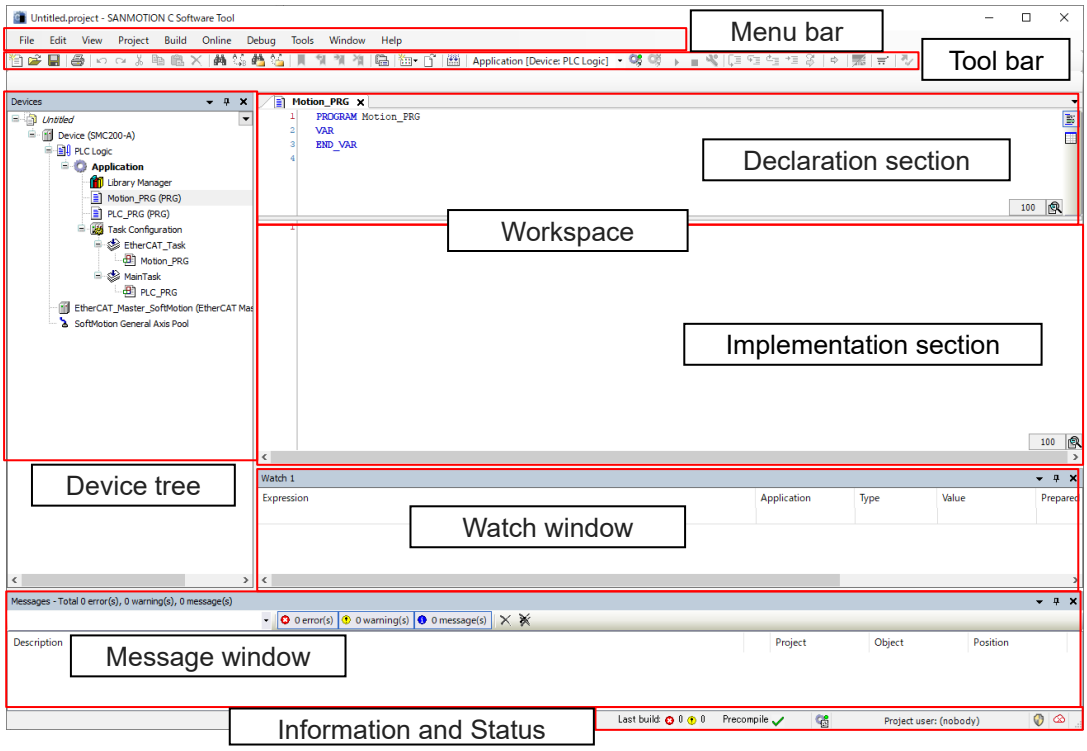


Fig.4.2 SANMOTION C Software Tool 2.0.0 basic screen

Item	Detail
Menu bar	edit the project.
Tool bar	supports project management, search, and input of programs.
Device tree	displays a list of POU, visualization, resources.
Workspace	Declaration section: Declare a program variable. Implementation section: Implement the program.
Message window	Displays logs at program compilation and search results of specified variables.
Watch window	Display variable monitoring or user defined list of watch expressions.
Information and Status	Provides information on current logged-in user, cursor position, PLC operation status, etc. RUN : During program execution. STOP : During program stop. HALT ON BP : During program stop by breakpoint.

If there are items that are not displayed in the window, they can be displayed by selecting the target item from "View" on the menu bar.

4.4. Project structure

The project will be managed in the device tree. In the device tree you manage all the objects necessary to run the project, such as hardware configuration and confirmation of the fieldbus system, hardware communication configuration and application.

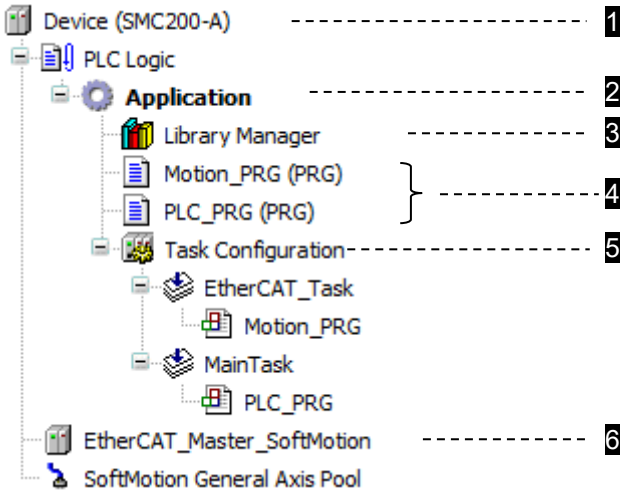


Fig.4.3 Project structure diagram

As shown above, one application needs at least one POU (program) and task. An application configuration example is shown below.

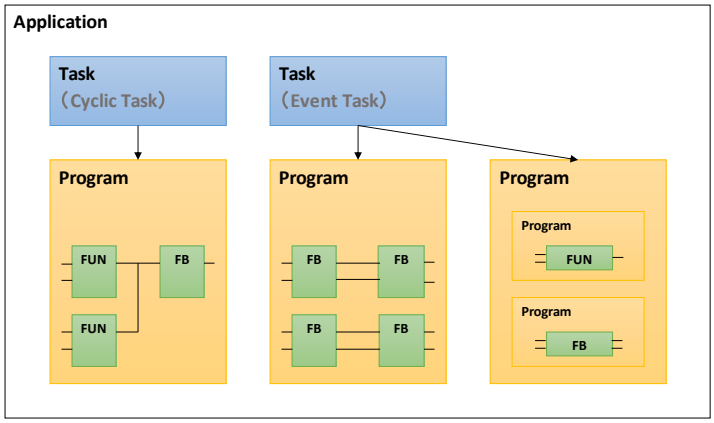


Fig.4.4 Application configuration example

No.	Item	Detail	Reference
1	Device	Target hardware to run the application	"4.5 Device"
2	Application	Set of objects required to execute PLC program ※Do not change the application name from "Application".	—
3	Library	Collection of reusable objects	—
4	POU	Program configuration unit such as PRG, FUN or FB	"4.6 POU"
5	Task	Process control of application program	"4.7 Task"
6	Drive	Drive configuration for motion control	"6.1.5.1 EtherCAT master setting" or later

4.5. Device

Configure the target hardware to run the application.

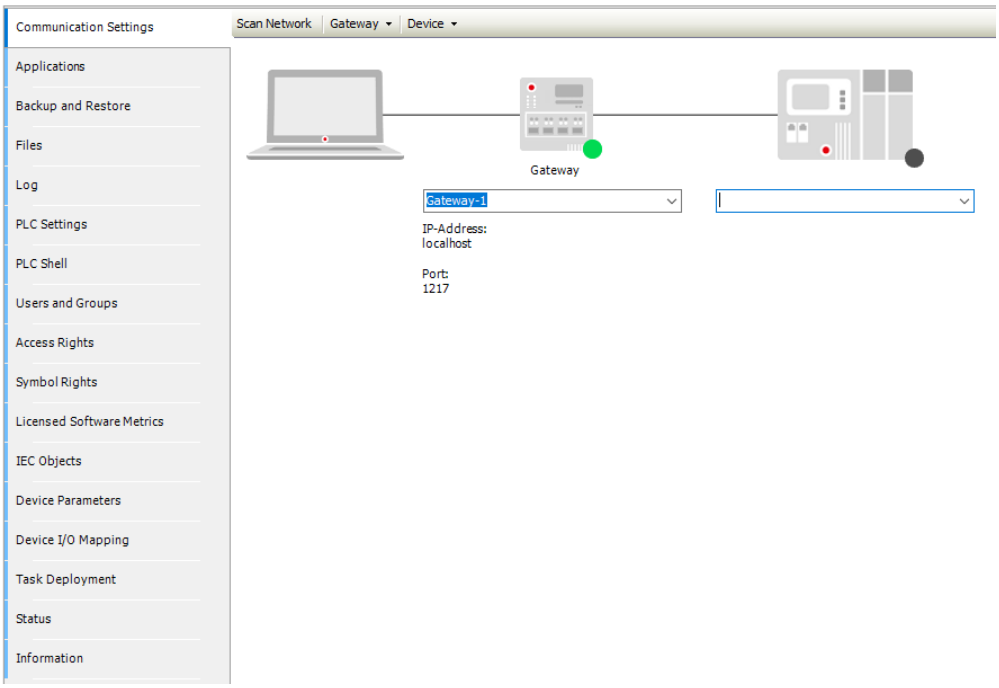


Fig.4.5 Device screen

Item	Detail
Communication Settings	Configuration of the connection between the development system and a S200. See "4.5.1 Communication Settings" for more information.
Applications	List of the applications on the S200.
Backup and Restore	Back up and restore applications currently running on the S200.
Files	Configuration of the file transfers between a host file system and the S200.
Log	Display of the S200 log file.
PLC Settings	Configuration of the handling of the I/Os: which application, behavior in the stop state, updating, bus cycle options, etc. See "4.5.3 PLC Settings" for more information.
PLC Shell	Text-based control monitor for interrogating certain information from the S200.
Users and Groups	User management with regard to the device at runtime.
Access Rrights	Rights for access to objects and files on the S200.
Symbol Rights	Access rights of individual user groups to symbols (symbol sets) on the S200.
Licensed Software Metrics	The status of each application is displayed in a tree structure.
IEC Objects	Displays objects that allow access to the device from IEC applications.
Device Parameters	Display and configuration of S200 parameters. See "4.5.4 Device Parameters" for more information.
Device I/O Mapping	Configure S200 I/O mapping. See "4.5.5 Device I/O Mapping" for more information.
Task Deployment	Overview of all inputs and outputs, which are assigned to tasks – useful for troubleshooting.
Status	Device-specific status and diagnostic messages.
Information	General information on the device (name, provider, version etc).

4.5.1. Communication Settings

Connect with the S200 with Ethernet cable or USB cable (Type-A to Type-miniB). The initial value of each IP address is as follows.

Interface(SILK)	IP address	Subnet mask
Ethernet（ETHERNET）	192.168.21.101	255.255.255.0
miniUSB（PC）	169.254.21.101	255.255.0.0

When connecting with Ethernet, please set the IP address of development PC to 192.168.21.XXX and the subnet mask to 255.255.255.0.

When connecting with USB, it is not necessary to set the IP address on the development PC. The IP address is automatically allocated by the APIPA function. If you have disabled the APIPA function, manually set the IP address within the range of [169.254.1.0] ~ [169.254.254.254].

The connection setting method with the S200 has automatic and manual setting.

■Automatic setting procedure

1. Click “Network Scan ...” in the communication settings to display the “Select Device” window.
2. If there is a connectable device, it will be displayed after Gateway.
3. When you click “OK” with the device you want to connect selected, the connection will be completed.

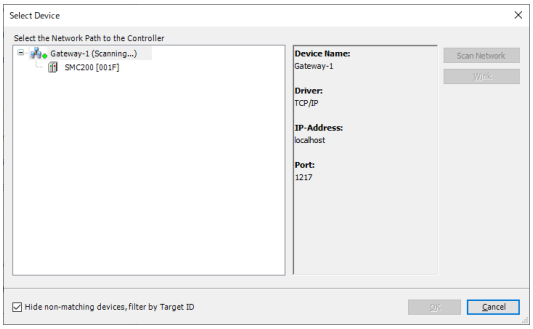


Fig.4.6 Device auto detection screen

If both the EHTERNET port and PC port (miniUSB) are connected between the S200 and the development PC and simultaneous access is possible, automatic search can not be performed correctly. In this case, specify the IP address in manual setting and connect.

■Manual setting procedure

Enter the IP address in the field of “<device not configured>“ in the communication setting. After input, press ENTER to start scanning. If there is a connectable device, the connection is automatically completed.

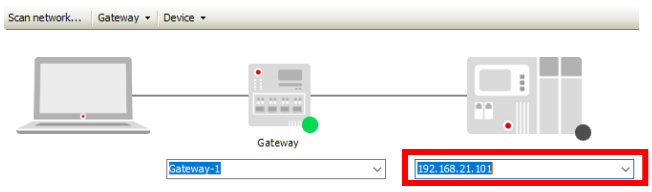


Fig.4.7 Device manual setting screen

If the PC can not connect to the S200, please confirm the environment by the following procedure.

① Confirm gateway

Make sure the status indicator lamp on the gateway is green on the connection screen (skip to the next confirmation item in the case of green). If it is red, it is probable that the gateway is not running or is stopped. If it is not running, please execute "CODESYS Gateway V3" from the Windows start screen. If it is stopped, right click on "CODESYS Gateway" in the notification area at the lower right of the Windows screen and select "Start Gateway".

② Confirm S200 startup status

Confirm that the status monitor (7 segment LED) of the S200 is in the RUN state ("O") (If it is not an error display and is not "O", skip to the next confirmation item). Please press the control button only once and confirm that the status monitor display changes. If the monitor does not change, it is probable that the S200 can not be detected due to the heavy CPU load by the running application. Since restarting the S200 and continuing to hold down the control button can stop the application startup, check the connection again after stopping.

③ Confirm unit type

Make sure that the unit type declared in the device tree matches the unit type of the target controller. You can confirm the unit type of the target controller by operating the control button (hold down for long when "u" is displayed). If the unit type is different, "Scan network ..." will not detect the S200. If the unit type matches and can not be detected, proceed to the next confirmation item.

④ Confirm IP addressIP

Please confirm that the S200 and the development PC are on the same network. The IP address of the Ethernet interface of the S200 can be confirmed by operating the control button (long press when "n" is displayed). The PING command is issued on the development PC, and if there is a response, the S200 is on the same network. If there is a response to the PING command and it can not be detected, proceed to the next confirmation item.

⑤ Confirm security software

When you use security software (firewall function) on development PC, communication of development tool may be blocked. Please allow communication of blocked development tools in your security software.

If there are no problems in all the above confirmation procedures, please contact us.

4.5.2. File

Files can be transferred between the development PC and the controller in the file tab in the device object.

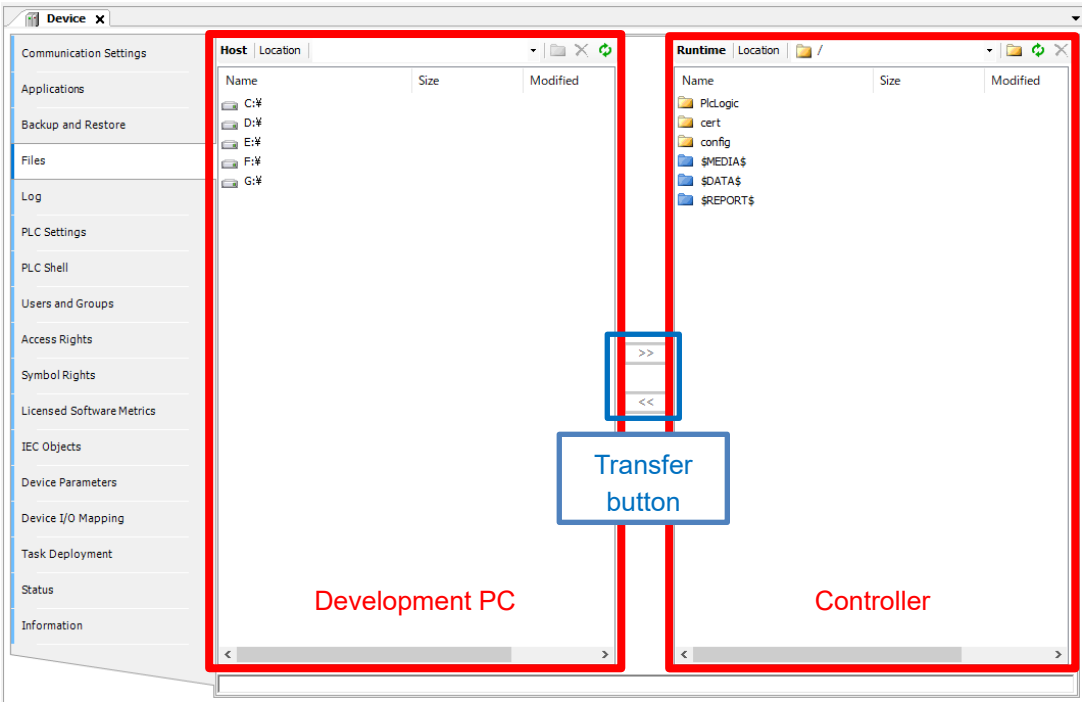


Fig 4.8 File transfer screen

The directories that can be accessed on this tab are the default path of the PLC application (yellow directory) and the defined path (blue directory). For the relationship between the PLC application path and the directory structure of the user area, refer to "6.4.2 Directory structure of user area".

Files can be transferred by selecting a file or directory and clicking the transfer button.

4.5.3. PLC Settings

In the PLC setting tab, make settings related to I/O update. External input data such as digital input is read at the beginning of the task and external output data such as digital output is written at the end of the task.

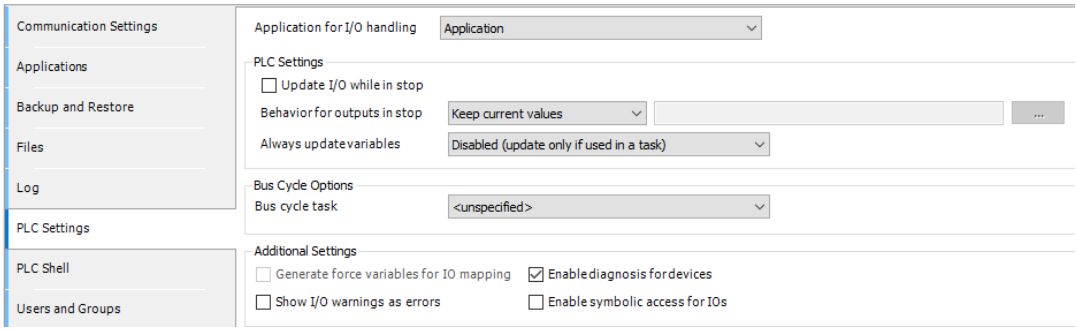
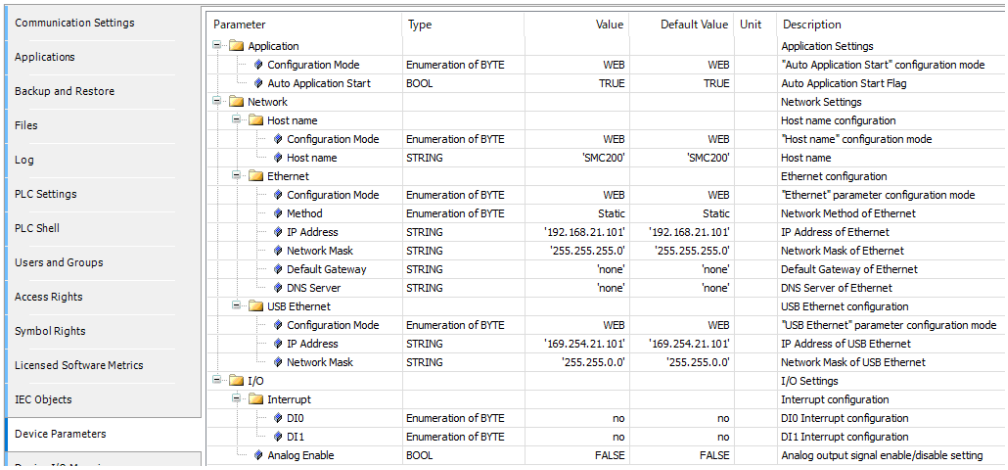


Fig.4.9 PLC setting screen

Item	Detail
Application for I/O handling	Select the application for I/O processing.
Update IO while in stop	Whether to update the I/O value even when the device is stopped is set. Valid: I/O is updated even while it is stopped Invalid: I/O is not updated while stopped
Behaviour for outputs in Stop	Handling of the output channels when the controller enters the stop state: ▪Keep current values: The current values are retained. ▪Set all outputs to default: The default values resulting from the I/O mapping are assigned. Set the default value in the "Default Value" column of "Device I / O Mapping". ▪Execute program: You can control the handling of the output values via a program contained in the project, which program executes at “STOP”. Enter the name of the program in the field on the right.
Always update variables	Global setting that defines whether or not CODESYS updates the I/O variables in the bus cycle task. This setting is effective for I/O variables of the slaves and modules only if “deactivated” is defined in their update settings. ▪Deactivated (update only if used in a task): CODESYS updates the I/O variables only if they are used in a task. ▪Activates 1 (use bus cycle task if not used in another task): CODESYS updates the I/O variables in the bus cycle task if they are not used in any other task.
Bus cycle task	Task that controls the bus cycle. By default the task defined by the device description is entered.

4.5.4. Device Parameters

In the "Device" parameter tab, you can make settings related to device functions. Parameters other than the I/O settings take effect after the power is turned on again.



Parameter	Type	Value	Default Value	Unit	Description
Application					Application Settings
Configuration Mode	Enumeration of BYTE	WEB	WEB		"Auto Application Start" configuration mode
Auto Application Start	BOOL	TRUE	TRUE		Auto Application Start Flag
Network					Network Settings
Host name					Host name configuration
Configuration Mode	Enumeration of BYTE	WEB	WEB		"Host name" configuration mode
Host name	STRING	'SMC200'	'SMC200'		Host name
Ethernet					Ethernet configuration
Configuration Mode	Enumeration of BYTE	WEB	WEB		"Ethernet" parameter configuration mode
Method	Enumeration of BYTE	Static	Static		Network Method of Ethernet
IP Address	STRING	'192.168.21.101'	'192.168.21.101'		IP Address of Ethernet
Network Mask	STRING	'255.255.255.0'	'255.255.255.0'		Network Mask of Ethernet
Default Gateway	STRING	'none'	'none'		Default Gateway of Ethernet
DNS Server	STRING	'none'	'none'		DNS Server of Ethernet
USB Ethernet					USB Ethernet configuration
Configuration Mode	Enumeration of BYTE	WEB	WEB		"USB Ethernet" parameter configuration mode
IP Address	STRING	'169.254.21.101'	'169.254.21.101'		IP Address of USB Ethernet
Network Mask	STRING	'255.255.0.0'	'255.255.0.0'		Network Mask of USB Ethernet
I/O					I/O Settings
Interrupt					Interrupt configuration
DI0	Enumeration of BYTE	no	no		DI0 Interrupt configuration
DI1	Enumeration of BYTE	no	no		DI1 Interrupt configuration
Analog Enable	BOOL	FALSE	FALSE		Analog output signal enable/disable setting

Fig 4.10 Device Parameters setting screen

Parameter			Detail	Default Value
Application	Configuration Mode		Configuration mode **WEB** : Web application settings are valid **PLC**: Project settings are valid	WEB
	Auto Application Start		Set the automatic startup of the application when the application exists at startup. **TRUE** :RUN state **FALSE** :STOP state	TRUE
network	Host name	Configuration Mode	Hostname **WEB** : Web application settings are valid **PLC**: Project settings are valid	WEB
		Host name	Set the host name.	‘SMC200’
	Ethernet	Configuration Mode	Configuration mode **WEB** : Web application settings are valid **PLC**: Project settings are valid	WEB
		Method	Set how to obtain an IP address. **DHCP** :Obtain from DHCP **Static** :Set manually	Static
		IP Address	Set the IP address for Ethernet port.	‘192.168.21.101‘
		Network Mask	Set netmask for Ethernet port	‘255.255.255.0‘
		Default Gateway	Default gateway	'none'
		DNS Server	DNS server	'none'
	USB Ethernet	Configuration Mode	Configuration mode **WEB** : Web application settings are valid **PLC**: Project settings are valid	WEB
		IP Address	Set the IP address for mini usb	‘169.254.21.101‘
		Network Mask	Set netmask for mini usb	‘255.255.0.0‘

Parameter			Detail	Default Value
I/O	Interrupt	DI0	Set the interrupt function of DI0 and DI1. **no** : Interrupt disabled **Rising edge** : An event occurs when a rising edge is detected. **Falling edge** : An event occurs when a falling edge is detected. **Both edge** : An event occurs when a rising or falling edge is detected.	no
		DI1		
	Analog Enable		Select whether the controller outputs current after starting. TRUE :Output when power on. FALSE :No output when power on。	FALSE

Be sure to set IP addresses of different segments (shown in red) for the IP addresses of ethernet, PC and Wireless adapter 3A.

Name	IP Address	Subnet mask
ethernet	192.168.21.101	255.255.255.0
PC	169.254.21.101	255.255.0.0
wireless	192.168.22.101	255.255.255.0

If "none" is set in the network settings, the currently set value will be deleted.

If an empty character (") is set, the currently set value is retained.

Only one default gateway can exist. If multiple default gateways are set, unexpected routing behavior may occur.

4.5.5. Device I/O Mapping

In the I/O mapping tab, you can assign the S200's digital input/output (input: 16 points, output: 8 points) and 2 analog outputs to variables. By assigning a variable name to the I/O mapping table as shown below, the set variable name can be used for programming.

You can assign either BOOL type or BYTE type.

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI0~7	%IB0	BYTE		
		Bit0	%IX0.0	BOOL		
		Bit1	%IX0.1	BOOL		
		Bit2	%IX0.2	BOOL		
		Bit3	%IX0.3	BOOL		
		Bit4	%IX0.4	BOOL		
		Bit5	%IX0.5	BOOL		
		Bit6	%IX0.6	BOOL		
		Bit7	%IX0.7	BOOL		
		DI8~15	%IB1	BYTE		
		Bit0	%IX1.0	BOOL		
		Bit1	%IX1.1	BOOL		
		Bit2	%IX1.2	BOOL		
		Bit3	%IX1.3	BOOL		
		Bit4	%IX1.4	BOOL		
		Bit5	%IX1.5	BOOL		
		Bit6	%IX1.6	BOOL		
		Bit7	%IX1.7	BOOL		

		DO0~7	%QB0	BYTE		
		Bit0	%QX0.0	BOOL		
		Bit1	%QX0.1	BOOL		
		Bit2	%QX0.2	BOOL		
		Bit3	%QX0.3	BOOL		
		Bit4	%QX0.4	BOOL		
		Bit5	%QX0.5	BOOL		
		Bit6	%QX0.6	BOOL		
		Bit7	%QX0.7	BOOL		
		AO0	%QW1	WORD(0..3999)		
		AO1	%QW2	WORD(0..3999)		

Fig.4.11 Setting of variable name to Device I/O (left: digital inputs, right: digital ・ analog outputs)

Updating the assigned variables can be set at the bottom of the I/O mapping tab.

Item	Detail
Always update variables	Definition for the device object about updating I/O variables. The default value is defined in the device description: User parent device setting: Update according to the setting of the PLC Settings tab. Enabled 1 (use bus cycle task if not used in any task): Runtime updates the I/O variables in the bus cycle task if they are not used in any other task.
Bus cycle task	Set the task for update the I/O. The drop-down list provides all tasks that are defined in the task configuration of the active application. **Use parent bus cycle setting** :Use the bus cycle task that set in PLC setting tab.

CAUTION

- Do not assign addresses used in Device I/O mapping to variables in AT declaration. If allocated, unexpected behavior may occur. Please refer to the online help for AT declaration

4.6. POU

IEC 61131-3 creates a program in a unit called POU (Program Organization Unit). Creating a program with this unit improves the readability and reusability of the program. Three types of functions (FUN), function block (FB), and program (PRG) are defined in the POU.

4.6.1. Program（PRG）

A program is a POU that provides one or more values during execution. After execution of the program, all values are retained until the next execution. The order of calling programs in the application is defined in the task object. It is possible to call PRG, FUN, FB from PRG.

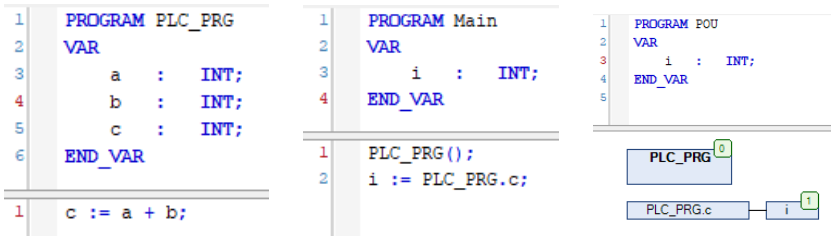


Fig.4.12 Program example (left: creation screen, center: ST, right: CFC)

4.6.2. Function block（FB）

The internal variable of FB holds its value from execution to next execution. Therefore, calling FB with the same argument does not necessarily return the same value. The output returns one or more data elements. Since the FB occupies memory, the instance name is required for execution.

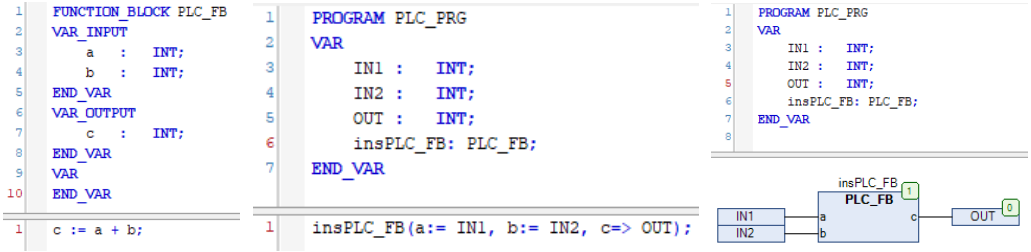


Fig.4.13 Example of function block (left: creation screen, center: ST, right: CFC)

4.6.3. Function（FUN）

FUN is a structural unit that does not hold internal state. Therefore, calling FUN with the same argument always returns the same value. The output returns one data element. It can not be instantiated. You can not call PRG, FB from FUN.

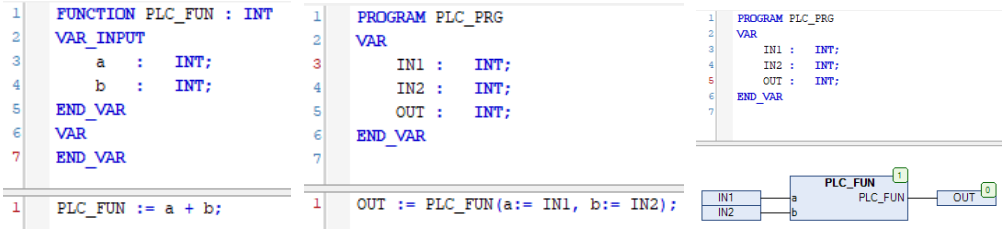


Fig.4.14 Example of function (left: creation screen, center: ST, right: CFC)

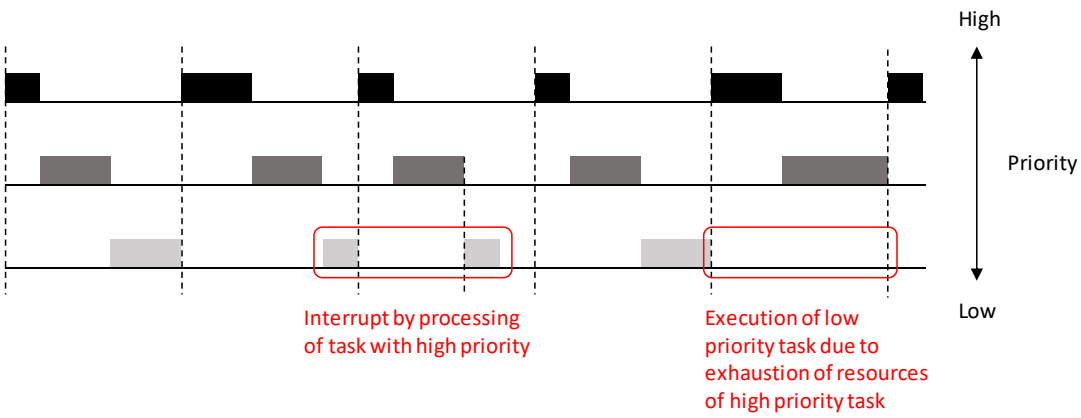
4.7. Task

In IEC 61131-3, a task indicates a function that specifies a condition for executing a user program. Tasks can define names, priorities, and triggering conditions for tasks. For each task, you can assign a program to be processed by the task. If the task is executed in the set cycle, these programs are processed in one cycle. The combination of priority and trigger condition determines the order in which tasks are executed.

The following rules apply to the execution of tasks:

- The task is executed if the condition is satisfied. That is, it is executed when the specified time elapses or after the rising edge of the set event variable.
- If there are valid conditions for multiple tasks, the task with the highest priority is executed.
- If there are valid conditions for multiple tasks and the same priority, the task with the longest wait time is executed.
- The calling process of the program is performed according to the order in the task configuration (top to bottom).

Because execution of tasks is performed in descending order of priority as described above, processing with lower priority tasks may be interrupted during execution. Also, if resources are allocated to tasks with higher priority, tasks with lower priority may not be processed.



The following parameters can be set for the task.

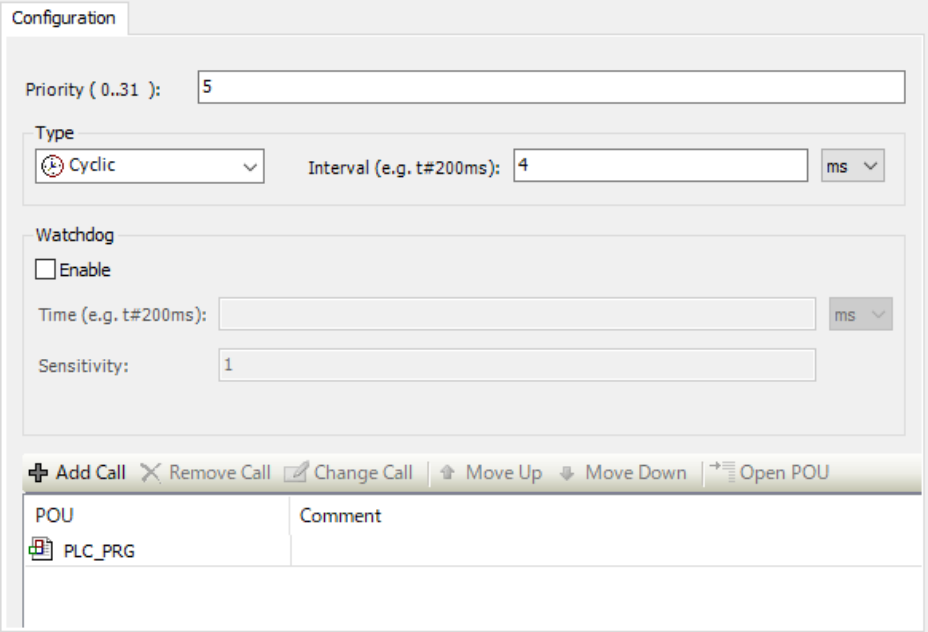


Fig.4.15 Task setting screen

Item	Detail
Priority	Sets the task priority. Priority is set between [0 and 31], 0 is the highest priority, 31 is the lowest priority.
Type	There are four types of tasks, and the program is executed at the following timing. **Cyclic** : It is executed at the “Interval” specified as the interval. **Event** : It executes when acquiring the rising edge of the flag specified for the event. **Freewheeling** : Since the program is continuously executed from the beginning to the end of one program as one cycle, the cycle can not be set. Since Freewheeling is executed continuously, it is necessary to set the lowest priority. Do not set Freewheeling for tasks that call programs that perform motion control. **Staus** : Execute when the flag specified for the event is TRUE. **External** : Executes when an interrupt event of digital input occurs. Refer to “4.5.4 Device Parameters” for setting interrupts.
Watchdog	You can set the monitoring time for each task. When the watchdog validity is checked, if the task processing exceeds the time set in “time” and it exceeds the number set in “sensitivity”, it ends in error state.
Add / Remove POU	POU can be added with “Add Call” and “Remove Call” can be deleted. You can also add it by dragging POU to the task on the tree.

In this S200, set the task priority as follows.

EtherCAT_Task: Priority 0

Other tasks: Priority 5 or less

4.8. Variable

In SANMOTION C Software Tool 2.0.0, it is possible to program by variable, not conscious of memory map.

4.8.1. Data type

It is necessary to define how to handle the assigned values for the variables in the program. This is called data type. There are two types of data types: standard data types already defined and user-defined data types defined by the user. The data type assigned to each identifier (variable name) specifies how much memory space is reserved and what type of value to store.

4.8.1.1. Standard data type

An example of the data type defined in IEC 61131-3 is shown below.

Reserve	Data type	lower limit	upper limit	Number of bits
BOOL	Boolean	0	1	1
INT	16 bit integer	-32768	32767	16
DINT	32 bit integer	-2147483648	2147483647	32
UINT	Unsigned 16 bit integer	0	65535	16
UDINT	Unsigned 32 bit integer	0	4294967295	32
REAL	Real number	-3.402823e+38	3.402823e+38	32
LREAL	64 bit real number	-1.7976931348623157e+308	1.7976931348623157e+308	64
STRING	Variable length single-byte character string	1 character	255 characters	8
BYTE	8-bit bit string	0	255	8
WORD	16-bit bit string	0	65535	16
DWORD	32-bit bit string	0	4294967295	32
TIME	Duration	0ms	4194967295ms	32
DATE	date	1970-00-00	2106-02-06	32

4.8.1.2. User-defined data type

An example of the data type defined in IEC 61131-3 is shown below.

Data type	detail	Declaration example
Array	Data structure in which plural data are arranged consecutively. It supports array of 1 to 3 dimensions.	ARRAY [0..9] OF INT
Pointer	It is a variable that holds the position information of the place where the content of a variable is stored. For details, see "9.7 Pointer"	POINTER TO INT
Enumeration type	User-defined data type consisting of string constants and numbers. Create enumerated objects as objects in the Object Types tab of the Object Organizer.	TYPE Access :(*Read:= 0,* *Write := 1*); END_TYPE
Structure	It combines various data types into one data type. Create the structure as an object in the Data Types tab of the Object Organizer.	TYPE *Coordinate*: STRUCT *X : REAL;* *Y : REAL;* *Z : REAL;* END_STRUCT END_TYPE

4.8.2. Declarative syntax

When using variables in a program, it is necessary to declare variables according to how variables are used. In IEC 61131-3, declaration of a variable is defined as follows.

Reserve	How to use variables
VAR	Used inside the configuration unit (local variable).
VAR_INPUT	Input from the outside and can not be changed within the constituent unit
VAR_OUTPUT	Output from the constituent unit to the external constituent.
VAR_IN_OUT	Input and output by external unit element. Can be changed in the configuration unit.
VAR_GLOBAL	Accessible from any configuration unit.
RETAIN	Define a hold variable.
CONSTANT	Constant variable.

4.8.3. Initial value setting

For the variable, you can set the value to set at the start of the project. When initial value is not set, “0” is set for numeric data and “” is set for character string.

An example of setting is as follows.

```
(*INT type initialization example*)
iNumber      : INT := 10;
(*Array initialization example*)
arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3]          (* 0,0,4,4,4,4,2,3 *)
(*Initialization example of structure*)
CartPos      : Coordinate := (X:=100, Y:=100, Z:=100);
(*Initialization example of structure array*)
PointData    : ARRAY [0..9] OF Coordinate := [ (X:=100, Y:=100, Z:=100),
                                              (X:=200, Y:=200, Z:=200),
                                              (X:=300, Y:=300, Z:=300) ];
```

If an array element is not explicitly set with a default value, it is initialized with the default value of the basic type. In the above example, the remaining elements are set to 0.

4.8.4. Input Assistant function

SANMOTION C Software Tool 2.0.0 has the following functions as program input assistant function.

1. Input Assistant

“Edit” on the menu bar ⇒ “Input Assistant” displays a dialog for selecting objects that can be entered at the current cursor position in the editor window.

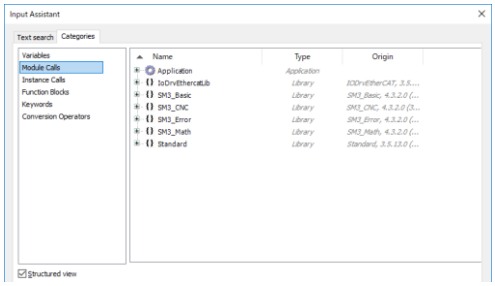


Fig.4.16Input Assistant dialog box

2. Display variable candidate

If you insert a dot “.” Instead of an identifier, a selection box will be displayed and a list of all the elements such as global variables and local variables in the project will be displayed. If you insert a dot “.” After a POU or structure variable, a list of variables declared in that POU or structure is displayed. Bits other than BOOL type can be accessed by adding dots and numbers at the end.

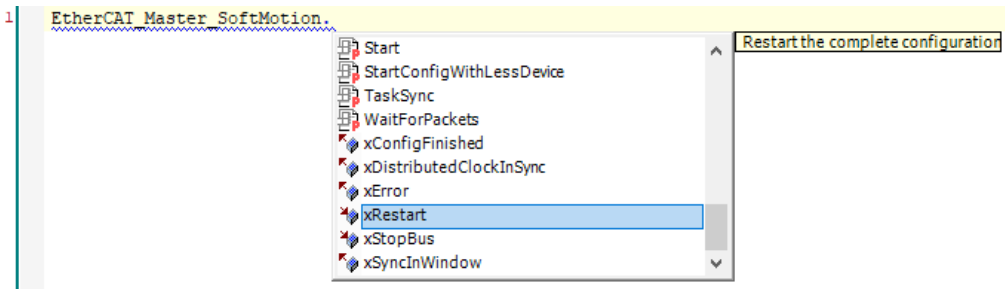


Fig.4.17 Examples of using variable candidates

3. Completion candidate display

If press <Ctrl> + <Space> key after entering an arbitrary character string, a list of all POUs, global variables, and available local variables in the project starting from the input character string will be displayed.

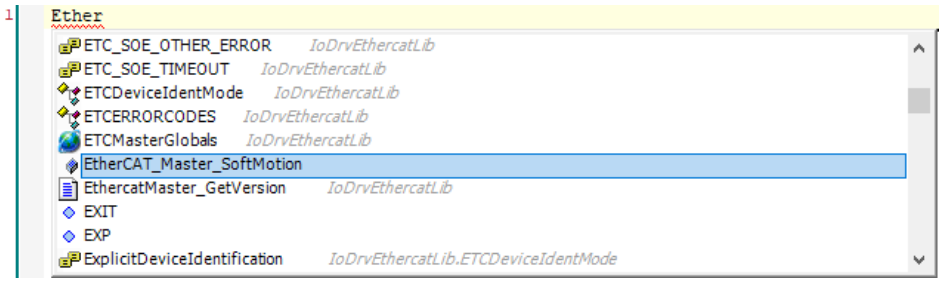


Fig.4.18 Completion candidate display usage example

4. Automatic declaration of variables

When you enter an undeclared variable in the implementation section, the "Auto Declare" dialog is displayed. "Type" is automatically entered with the data type inferred by the development environment. Clicking "OK" inserts the variable into the declaration section.

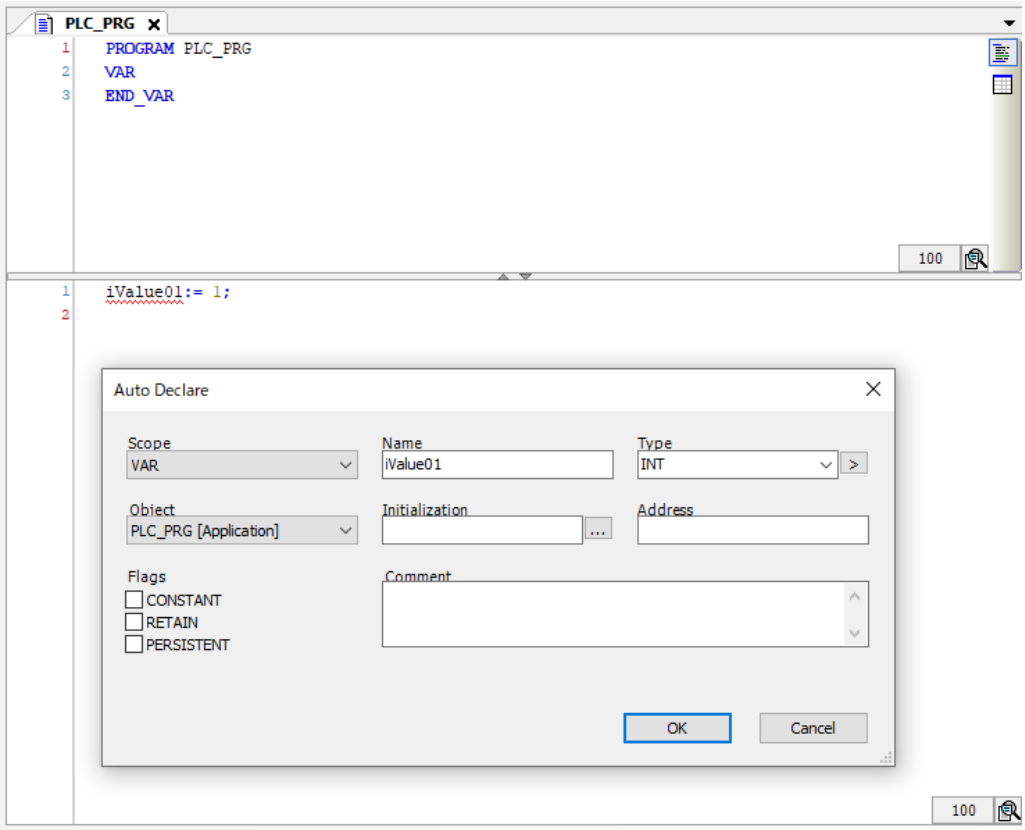


Fig.4.19 Automatic declaration of variable usage example

By default, the automatic variable declaration function is disabled only for the ST language.

Activation of the automatic declaration function in ST language and change of editor-related settings can be performed from "Tools" ⇒ "Options" ⇒ "SmartCoding" on the menu bar.

4.9. Programming language

SANMOTION C Software Tool 2.0.0 supports six programming languages (LD, IL, FBD, CFC, ST, SFC) conforming to IEC 61131-3, and it is possible to combine program languages suited to the control contents.

For details on how to use each programming language, refer to the online help.

4.9.1. LD (Ladder Diagram)

The ladder diagram developed from the electrical circuit control fig used in the automobile industry to describe the relay control circuit. This language consists of a combination of multiple contacts and coils on the current line (line) flowing from left to right. The “ON” and “OFF” states of the contacts placed on each line are transferred to the connected coils. It is easy to understand because the structure is simple.

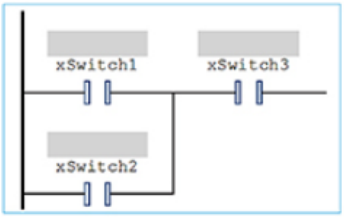


Fig.4.20 LD Description example

4.9.2. IL (Instruction List)

It is a low-level language similar to an assembler. It consists of one command, one operand, and one optional label on each line. It is a language that is often used for prioritizing processing speed.

Fig.4.21 IL Description example

4.9.3. FBD (Function Block Diagram)

FBD is a graphic oriented programming language. It has an input variable and an output variable and combines multiple functions and function blocks. The language of the feature is easy to grasp the flow of signals and data.

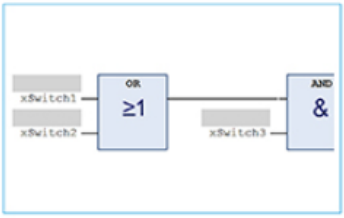


Fig.4.22 FBD Description example

4.9.4. CFC (Continuous Function Chart)

CFC is a graphic oriented programming language. Unlike FBD, there is no compulsion such as line execution, feedback loop etc. are possible, and various functions can be arranged freely. It is a language easy to grasp the flow of signals and data.

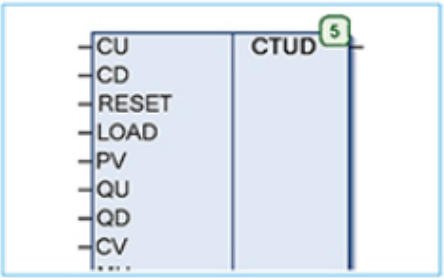


Fig.4.23 CFC Description example

4.9.5. ST (Structured Text)

It is a high level text language that supports structured programming and has language syntax very similar to BASIC. Since it is easy to describe calculation formulas and logical expressions, it is a language that is often used for creating complicated programs.

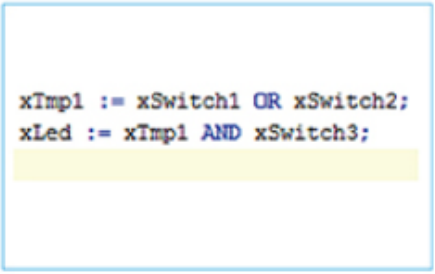


Fig.4.24 ST Description exampleST

4.9.6. SFC (Sequential Function Chart)

A graphic language for describing the sequence control of the control system. A step describing processing under the state, a transition describing the state transition condition, and a link that is a line connecting the step and the transition. It is used to describe time and event driven control sequences.

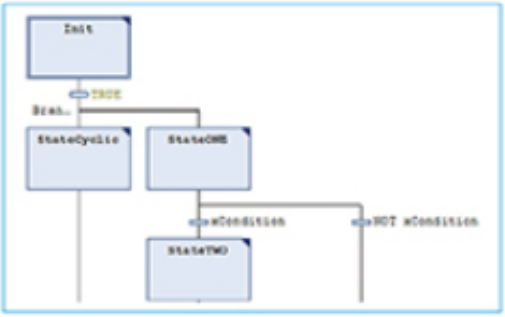


Fig.4.25 SFC Description example

4.9.7. Program language features

Each programming language has good, weak control contents. Therefore, by selecting the programming language according to the control contents, readability and processing performance can be improved.

A table summarizing the good and weak control contents of each programming language is shown below.

Processing details	LD	IL	FBD	CFC	ST	SFC
logical operation	△	×	○	○	△	×
Formula manipulation	△	×	△	△	○	×
Simple relay sequence processing	○	×	△	△	△	×
Sequence processing by state transition	△	×	×	×	△	○
Complex control process	△	×	△	△	○	×
In case of restrictions on program memory	△	○	△	△	△	×
In case of requiring processing speed	△	○	△	△	△	×
Representation that is easy to handle with control flow	×	×	△	△	△	○
In case of want to check the control operation visually.	○	×	○	○	×	△
＜Meaning of symbols＞ ○ : Suitable, △ : Sometimes it is not suitable, × : Unsuitable						

IL language is disabled at default setting. please put a check mark the following item to display IL language. Menu bar→"Tools"→"Options…"→"FBD, LD, and IL editor"→"IL"→"Enable IL"

4.10. Add device configuration file

If the configuration file for the EtherCAT slave or EtherNet / IP adapter is not installed in the development environment, installation must be performed using the “Device Repository”. Please, obtain the files to be installed from the device manufacturer you are using.

Type	File	Extension
EtherCAT Slave	ESI Fail	.xml
EtherNet/IP Adapter	EDSFail	.eds

The following shows the installation procedure.

1. Please select "Device repository ... (D)" from "Tool" on the menu bar.

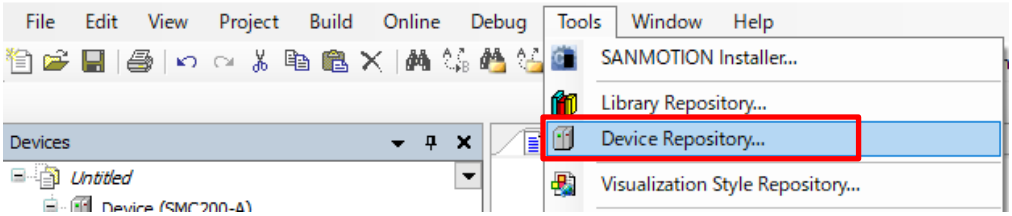


Fig 4.26 Device Repository

2. The Device Repository window will open. Click on "Install (I)".

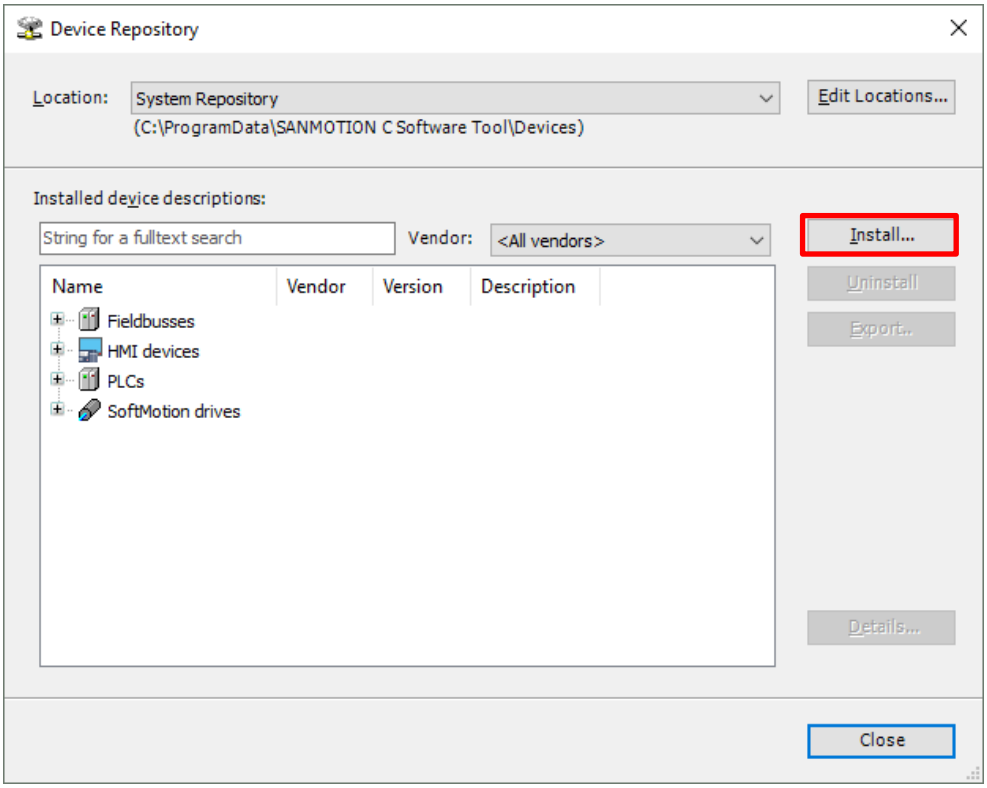


Fig 4.27 Device repository window

3. The Device Description Installation window will be displayed, so select the file to be installed. Note that only the file format set in the red frame at the bottom of the image is displayed in this window. Change according to the format of the file to be installed.

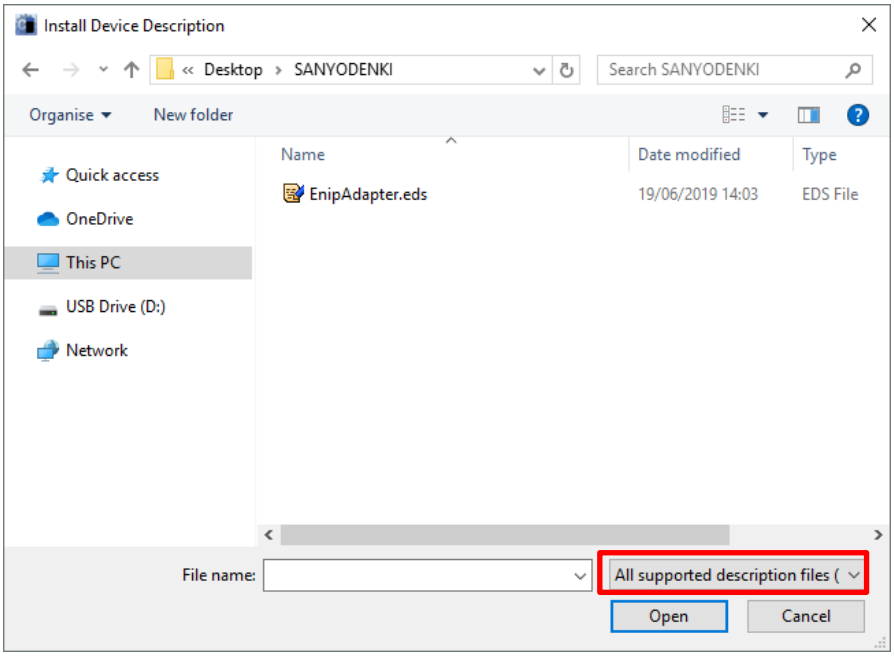


Fig 4.28 Install Device Description window

4. If it is successfully installed, the display will be as follows.

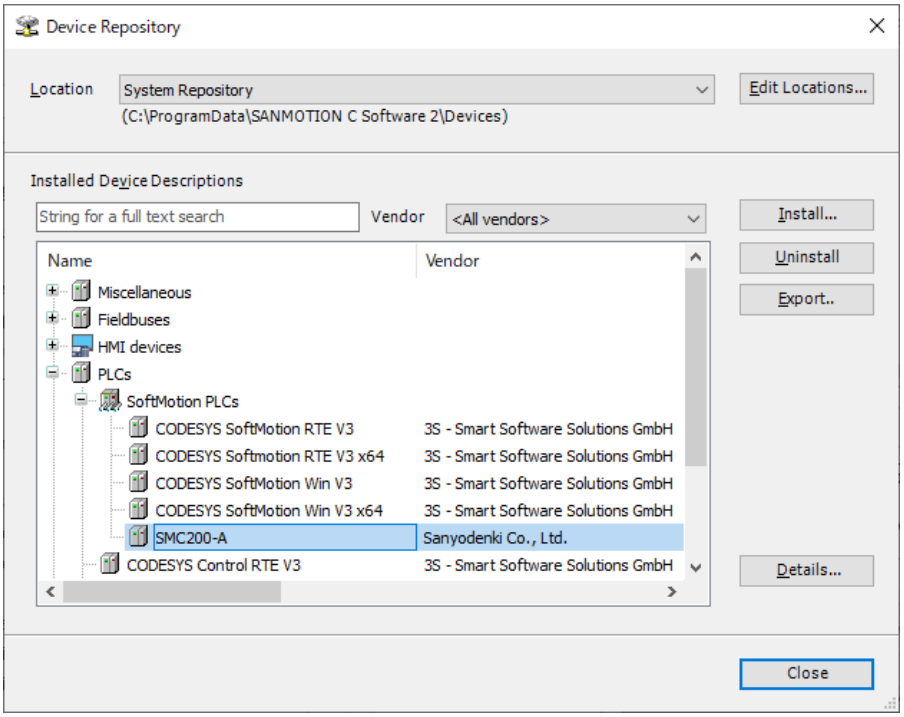


Fig 4.29 Device repository window when installation is complete

4.11. Library

4.11.1. Add library

The library provides modules and functions for use with the SANMOTION C Software Tool 2.0.0 application. Only the minimum necessary library is added to the project created from the template file. Therefore, you need to add libraries as needed.

The representative library is described below.

Library	Detail
File Access	Used for file control
Network	Used for network control
Serial Communication	Used when performing serial communication control
Memory	Used for memory control.

Library is added from "Library Manager". The following shows the procedure for adding a library.

1. Double-click "Library Manager" in the device tree and select "Add library".

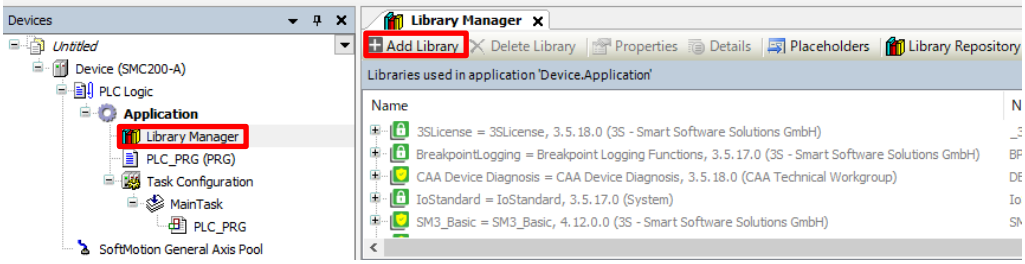


Fig 4.30 Library Manager

2. The Add Library window will be displayed, so select the library you want to add. Note that only the basic library is displayed in the window that is displayed first.

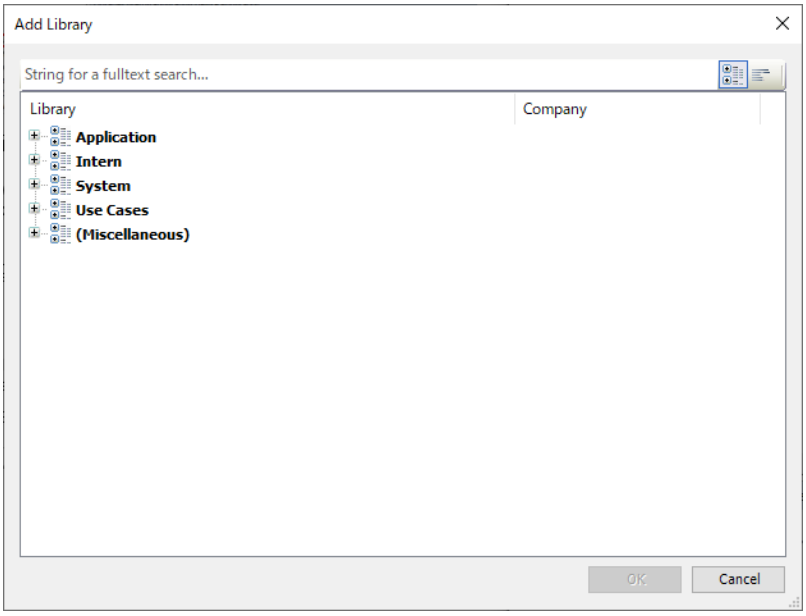


Fig 4.31 Add Library window

4.11.2. Create library

You can create a new library. This allows you to use your own POU in multiple projects. The following is an example of the procedure for creating a library.

Please refer to CODESYS Online Help for details on each item.

1. The library is created from the template file. There is a template file in the “Libraries” category in the “New Project” window. This time, select the standard template "CODESYS library".

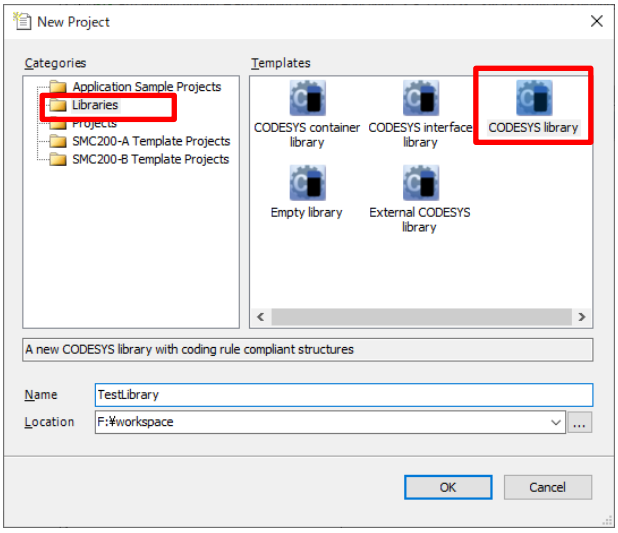


Fig 4.32 Create a new library

2. Set up the library. Click the POU at the bottom of the device tree to switch tabs and display the template POU. You can change the creator of the library, namespace, etc. by clicking "Project Information".

Be sure to set unique values for the default namespace and placeholder.

You can edit the default namespace in "Overview" and the placeholder in "Properties".

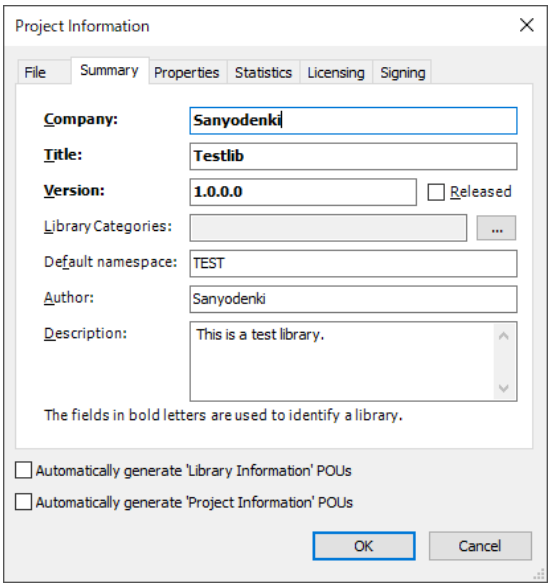


Fig 4.33 Project Information

3. Create a new function. Right click the “Functions” folder of the template and select “Add Object” → “POU…”.

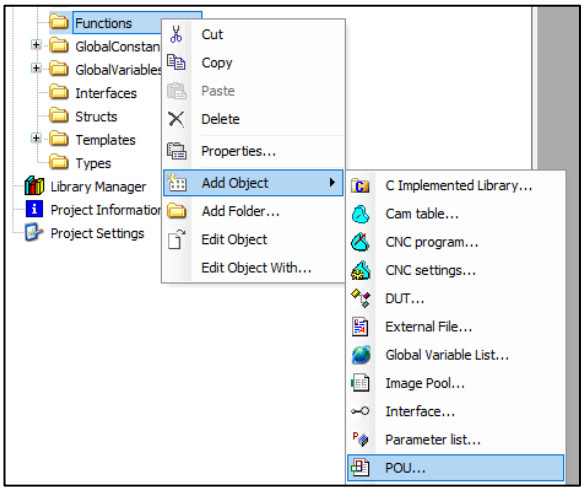


Fig 4.34 Create a new function

4. The "Add POU" window will be displayed. Set up the function.

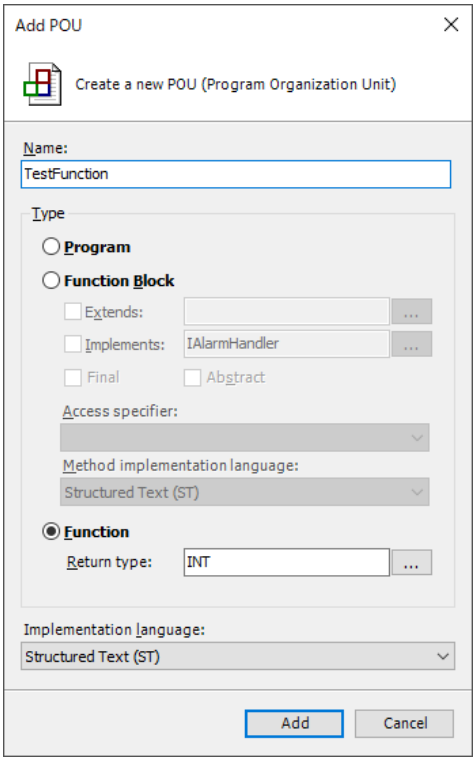


Fig 4.35 Add POU

5. Describe the processing content in the added function. This time, we will create a function that receives two INT type variables as inputs and returns the sum.

```
TestFunction
1  FUNCTION TestFunction : INT
2  VAR_INPUT
3      Data1: INT;
4      Data2: INT;
5  END_VAR
6  VAR
7  END_VAR
8

1  TestFunction := Data1 + Data2;
```

Fig 4.36 Function description

6. Compile the library. Select "Build"-> "Check all Pool Objects" from the menu bar to start compiling. When the compilation is completed normally, the message "0 errors, 0 warnings" is displayed.

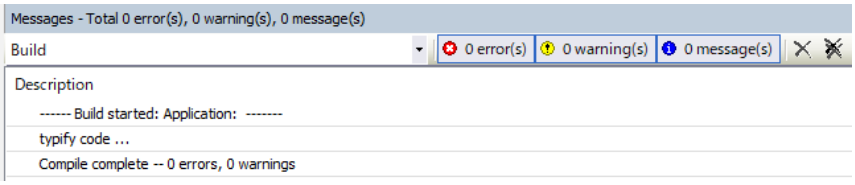


Fig 4.37 Compile result

7. When the compilation is complete, save the library from "File" on the menu bar. There are two types of library file formats.

   (1) Select "Save Project" or "Save Project As" to save it as a library file (.library). This format allows you to re-edit the library. You can also refer to the source code from the project.

   (2) Select "Save Project As Compiled Library" to save it as a compiled library file (.compiled-library). Files of this format cannot be re-edited and the source code cannot be referenced from the project.

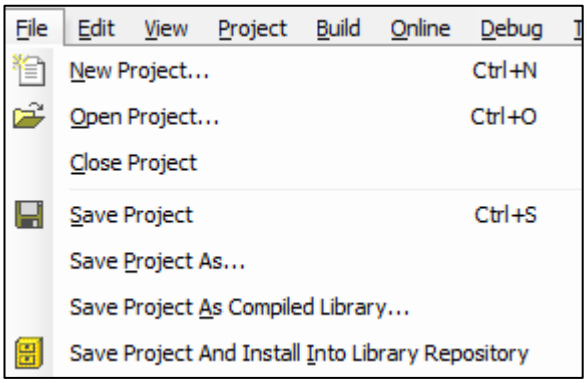


Fig 4.38 Saving the library

4.11.3. Install library

The newly created library can be called by installing it in the SANMOTION C Software Tool 2.0.0 application.

The following shows how to install the library.

Select “Tools” → “Library Repository” from the menu bar to display the “Library Repository” window.

Click “Install” and select the library file you want to install.

When the installation is complete, the new library will be displayed in "Installed libraries".

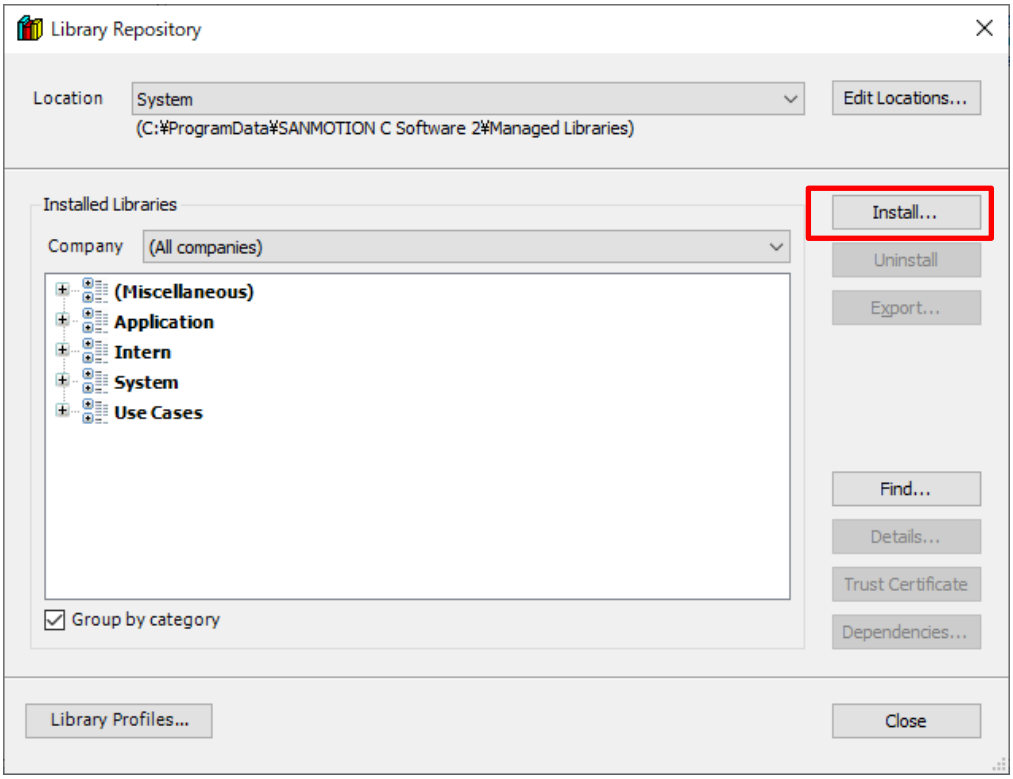


Fig 4.39 Install the library

The installed library can be used in the project by calling it in the same procedure as "4.11.1. Add library".

4.11.4. Use library

Shows how to use the added library.

The POU in the added library can be used by using a unique namespace.

You can check the namespace for each library in the library manager.

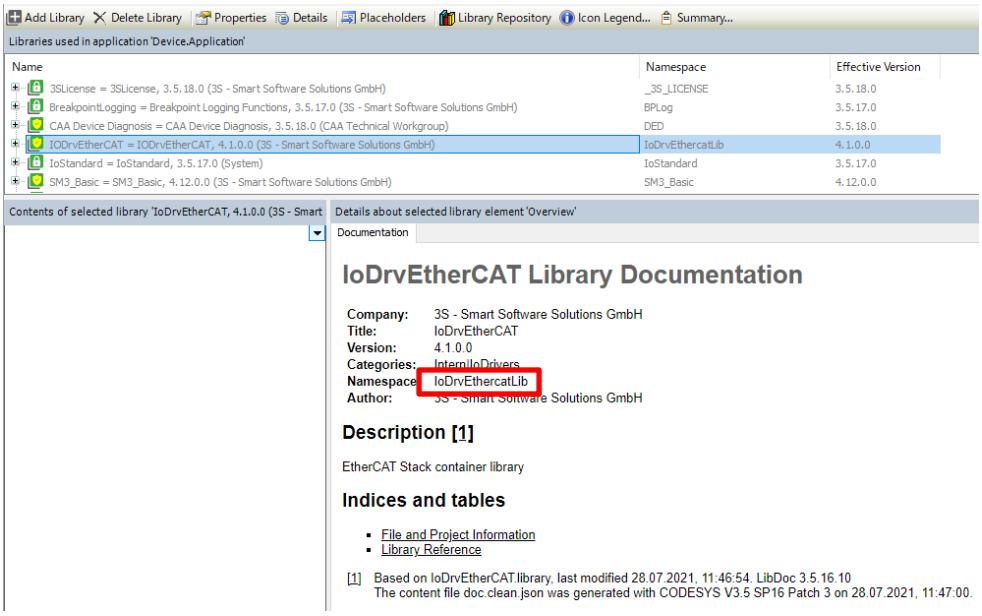


Fig 4.40 Check the namespace

As an example, use the function that returns the sum of two INT type variables created in "4.11.2. Create library". You can use the function in the added library by writing "<namespace>.<function name>". In this example, the process is to store the sum of “a” and “b” in the variable “c”.

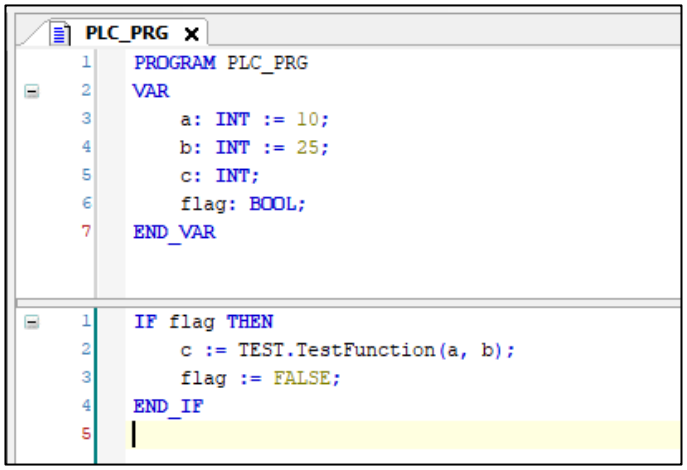


Fig 4.41 Use a function

4.12. Application transfer

There must be no error compiling the program to transfer the application to the controller. The controller connection settings must be set.

4.12.1. Transfer from the integrated development environment via the network

The procedure for transferring an application from the integrated development environment via the network is described below.

1. Select the controller in the device tree to display the communication settings tab screen.

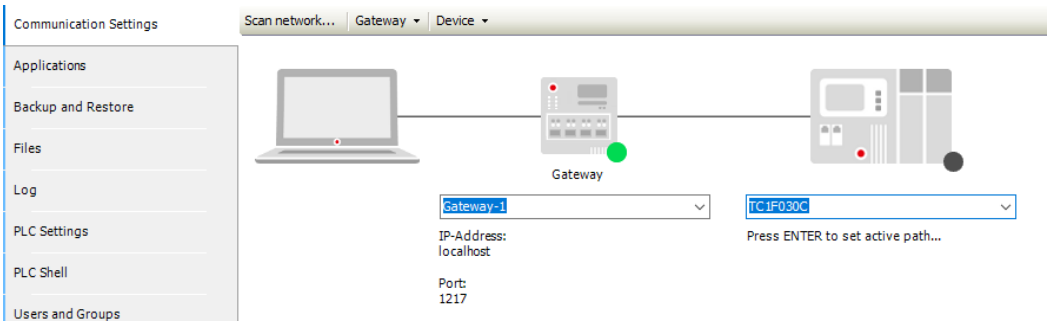


Fig 4.42 Controller communication setting screen

2. Please refer to "4.5.1 Communication Settings" for the connection method.

3. After connecting, you can transfer the application by selecting "Online"→"Login" from the menu bar.

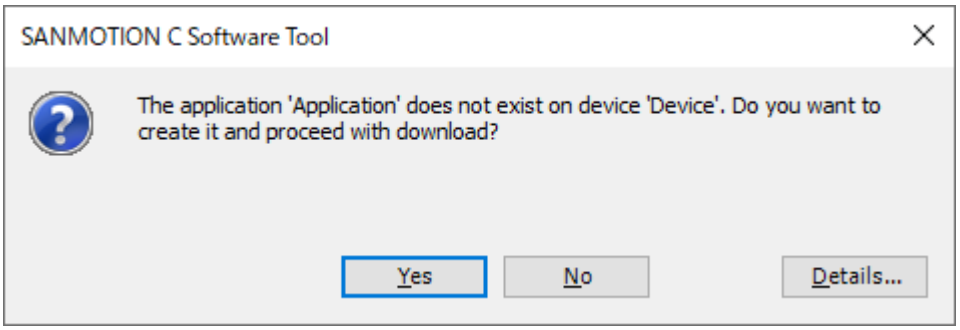


Fig 4.43 Login message when application does not exist

If the application already exists on the controller and you make changes related to the system configuration (For example, changing the number of control axes), the following message will be displayed. Select "Yes" to transfer.

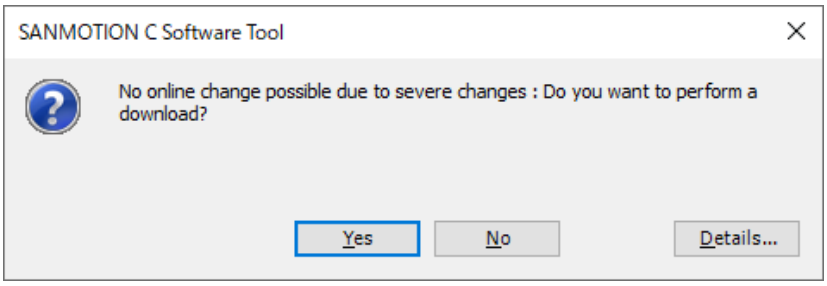


Fig 4.44 Login message when the application exists and there is a severe change

If the application already exists on the controller and you change anything other than the parts related to the system configuration, the following message is displayed. Select an option according to the transfer content, and then click OK to start the transfer process.

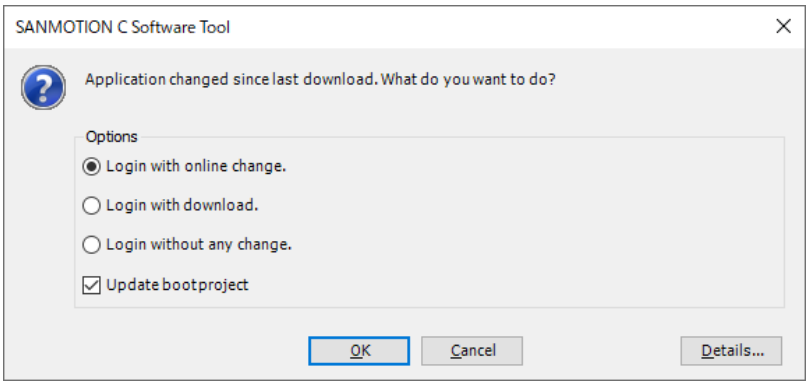


Fig 4.45 Login message when the application exists and is not a severe change

Parameter name	Detail
Login with online change.	Download only the changes in the running application. Online changes keep the project running. No application initialization is performed.
Login with download.	Generates the code for the entire application and downloads it. This command initializes all variables except retain data.
Login without any change.	It transitions to the login state without transferring the application.
Update bootproject	A boot project is a project that is loaded when the controller starts. Therefore, if this option is not enabled, the project that does not reflect the changes will be loaded when the power is turned on again.

If you transfer the application by "Login with online change." when changing the visualization, the application will not be initialized. Screen changes may not be reflected. Therefore, when changing the visualization, use " Login with download." to transfer the application.

4.12.2. Source code downloads and upload

The integrated development environment has a function to transfer the project source code to the controller as a project archive. You can then load this project archive from the controller into your integrated development environment as needed.

4.12.2.1. Source download(Development PC → Controller)

The command to download the source code differs depending on the login status.

[Login]

You can transfer the source code to the connected controller by selecting "Online"→ "Source Download to Connected Device" from the menu bar.

[Logging out]

You can transfer the source code by selecting "File"→ "Source Download…" from the menu bar and selecting the controller you want to download in the following window.

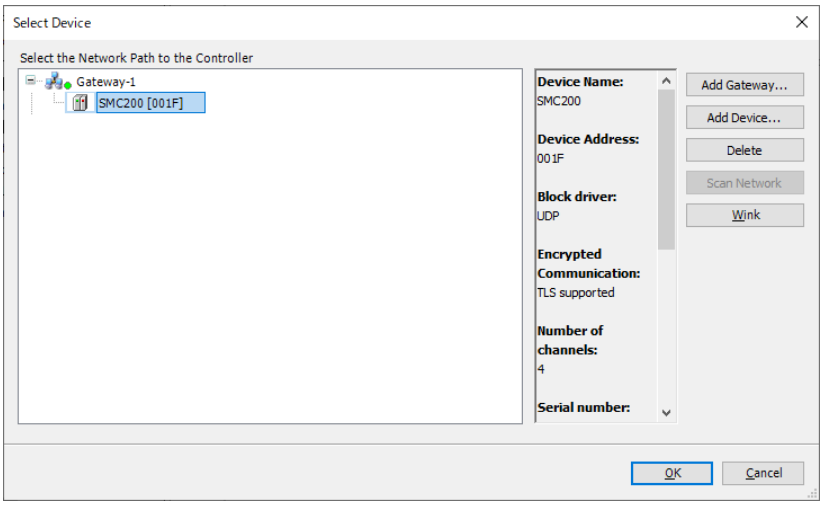


Fig 4.46 Source code download destination selection screen

If the source code is successfully downloaded, the project archive Archiv.prj will be saved in the user area.

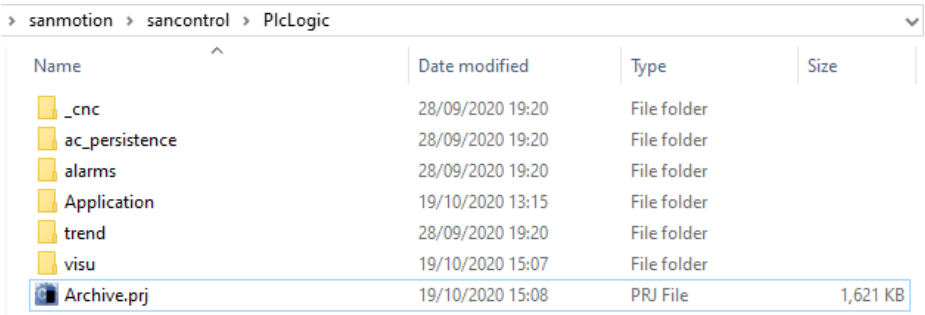


Fig 4.47 Project archive creation directory

4.12.2.2. Source upload(Controller → Development PC)

You can upload the source code by selecting "File"→ "Source upload…" from the menu bar and selecting the controller you want to upload in the following window.

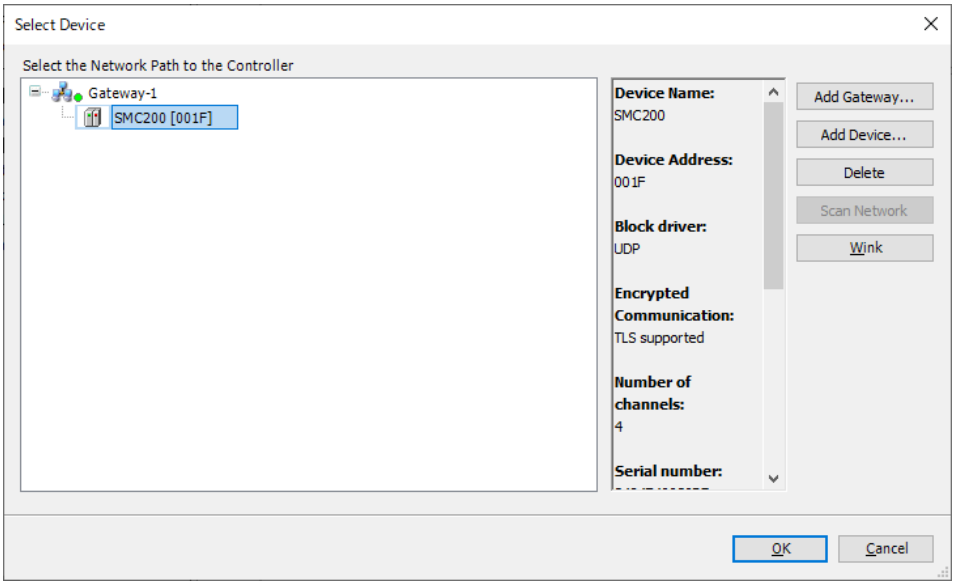


Fig 4.48 Source code upload source selection screen

If a valid project archive exists for the selected controller, the following window will be displayed. You can open the project by setting the deployment destination of the project and clicking "Extract".

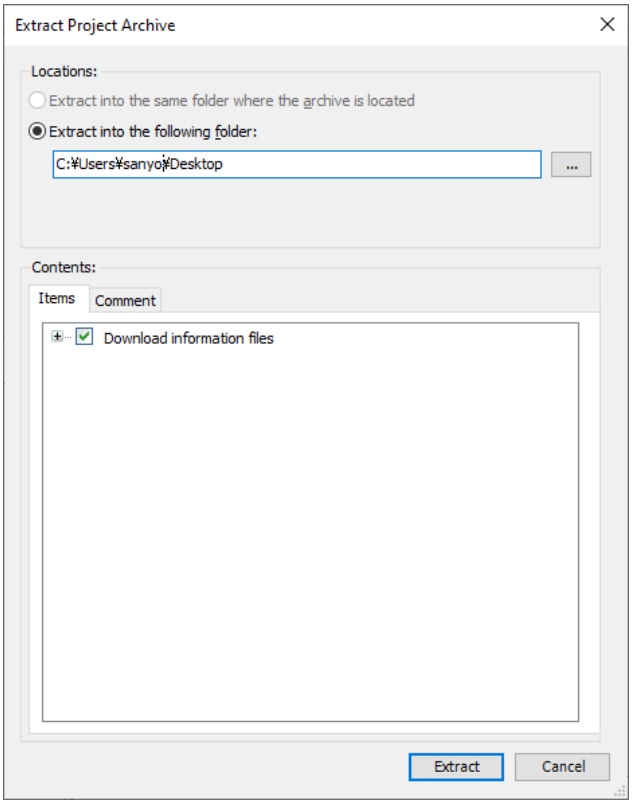


Fig 4.49 Source code expansion setting screen

4.13. Debug function

SANMOTION C Software Tool 2.0.0 provides various options for testing your application and detecting errors. Debug is to find and correct errors in programs. Using breakpoints and stepping commands, you can examine specific parts of a program. By writing values to variables, you can influence the running program.

4.13.1. Monitoring

When the application is running on the S200, in the SANMOTION C Software Tool 2.0.0 there are some features for monitoring and changing the values of the variables as well as for recording and storing the value charts.

Device.Application.Motion_PRG

Expression	Type	Value	Prepared value	Address	Comment
Drive1_Power	MC_Power				For servo on/off contro
Axis	REFERENCE TO AXI...	16#B6520BE8			Reference to axis
Enable	BOOL	TRUE			.. _MC_Power.Enable:
bRegulatorOn	BOOL	TRUE			``TRUE``: Enables the power stage.
bDriveStart	BOOL	TRUE			``TRUE``: Disables the quickstop mechanism.
Status	BOOL	FALSE			.. _MC_Power.Status:
bRegulatorRealState	BOOL	FALSE			``TRUE``: The power...ge has been switch...
bDriveStartRealState	BOOL	FALSE			``TRUE``: Drive is no...ocked by the quickst...
Busy	BOOL	TRUE			``TRUE``: Execution ...he function block ha...
Error	BOOL	FALSE			``TRUE``: Error has ...rred within the func...
ErrorID	SMC_ERROR	SMC_NO_ERROR			Error identification
strInstancePath	STRING	'Device.Applicati...			
Drive1_Home	SMC200.SanHome				For homing
Drive1_Move	MC_MoveAbsolute				For moving
MainStep	INT	0			Main operation step management variable
Trigger	R_TRIG				Execution trigger detection FB

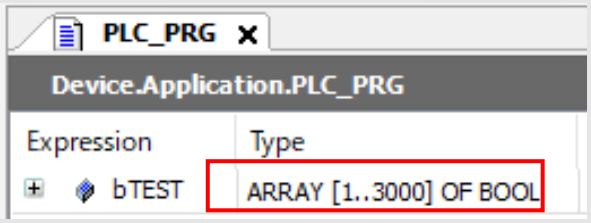
```
1   Drive1_Power(Axis := Drive1, Enable TRUE := TRUE);
2   Drive1_Home(Axis := Drive1);
3   Drive1_Move(
4       Axis:= Drive1,
5       Velocity 100 := 100,
6       Acceleration 1E+04 := 10000,
7       Deceleration 1E+04 := 10000);
8
9   CASE MainStep 0 OF
10
11   0 : (* S-ON *)
12     Drive1_Power.bDriveStart TRUE := TRUE;
13     Drive1_Power.bRegulatorOn TRUE := TRUE;
14     IF Drive1_Power.Status FALSE THEN
15      MainStep 0 := 1;
16     END_IF
17
18   1 : (* Homing *)
19     Drive1_Home.Execute FALSE := TRUE;
20     IF Drive1_Home.Done FALSE THEN
21      MainStep 0 := 2;
22     END_IF
23
```

Fig.4.50 Example of monitoring display

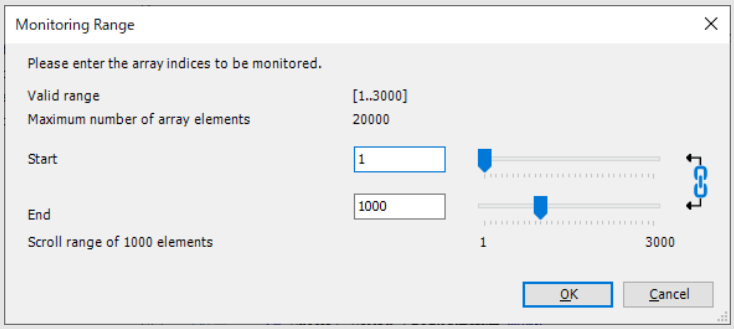
From the menu bar "Debug" ⇒ "Display Mode" you can switch the display of the variable value to binary, decimal, hexadecimal.

The maximum number of arrays that can be displayed is 1000. If you want to check variables with a sequence number of 1000 or more, change the setting in "Monitoring range". The procedure is shown below.

1. Log in to S200 and double-click the type field.



2. A window for setting the monitoring range will be displayed. Change it to any value.



4.13.2. Breakpoint

Breakpoints are commonly used for debugging programs. You can set breakpoints at specific positions in the program to force an execution stop and to monitor variable values. The following table shows an overview of all defined breakpoints for an application.

Item	Detail
Enable/disable breakpoint（F9）	Toggles the status of the breakpoint or execution point between “enabled” and “disabled”
Start（F5）	Move to the next breakpoint.
Step into（F8）	Execution of each statement one at a time; also in called POUs.
Step over（F10）	Execution of statements in one step; called programs are processed.

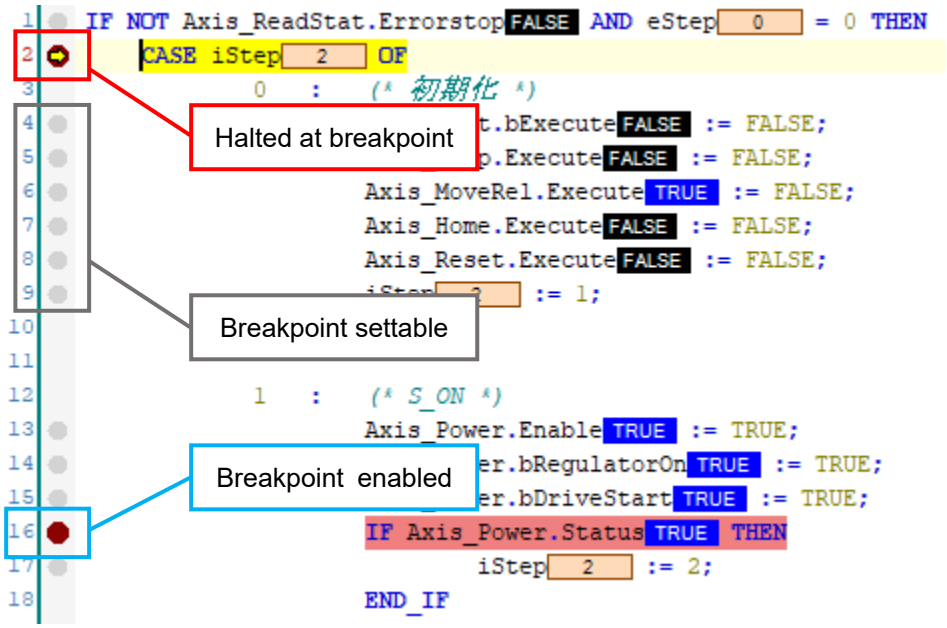


Fig.4.51 Using Breakpoints

CAUTION

- If you place a breakpoint in the POU being executed with EtherCAT_Task and stop the program, EtherCAT communication also stops. Be aware that it may cause unexpected behavior.

4.13.3. Forcing and Writing Variables in online

Variable values in the PLC can be changed in online mode. The value that can be written depends on the variable type. You can select as many variables as you want to write. It can be used to input triggers and edit variables. By using the dialog box, you can perform variable input at once. The command of substitution function is as follows.

Item	Detail
Write Values（Ctrl + F7）	This command sets a predefined value to a variable on the controller one time.
Force Values（F7）	This command sets a permanent predefined value to a variable on the controller.
Unforce Values（Shift + F7）	This command resets the forcing of all variables.

Expression	Type	Value	Prepared value
bStart	BOOL	TRUE	FALSE
iStep	INT	2	0
eStep	INT	0	
iErrorCnt	INT	0	

Fig.4.52 Example of writing value

4.13.4. Flow Control

With flow control, you can monitor the processing of the application program. With an activated flow control, IDE displays the variable values and results from function calls and operations at the respective processing location and time. In this way, the exact lines of code and networks that run through the current cycle are marked in colors.

It can be set from “Debug” menu bar “Toggle Flow Control Mode”.

```
1  Drivel_Power(Axis := Drivel, Enable TRUE := TRUE);
2  Drivel_Home(Axis := Drivel);
3  Drivel_Move(
4      Axis:= Drivel,
5      Velocity 100 := 100,
6      Acceleration 1E+04 := 10000,
7      Deceleration 1E+04 := 10000);
8
9  CASE MainStep 0 OF
10
11   0 : (* S-ON *)
12     Drivel_Power.bDriveStart TRUE := TRUE;
13     Drivel_Power.bRegulatorOn TRUE := TRUE;
14     IF Drivel_Power.Status FALSE THEN
15      MainStep 0 := 1;
16     END_IF
17
18   1 : (* Homing *)
19     Drivel_Home.Execute FALSE := TRUE;
20     IF Drivel_Home.Done FALSE THEN
21      MainStep 0 := 2;
22     END_IF
23
24   2 : (* Set Wait *)
25     Trigger(CLK FALSE := xSet FALSE);
```

Fig.4.53 Example of flow control display

4.13.5. Trace

You can use a Trace to follow the value history of variables on the controller in a similar way as a digital sampling oscilloscope.

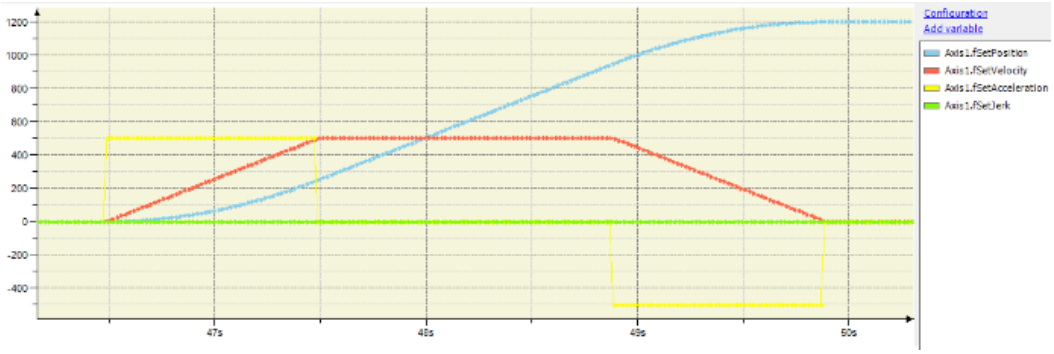


Fig.4.54 Trace screen

The following is procedure for using Tarce.

This sample program counts up execution of PLC_PRG from 0 to 100. And it is recorded in iCounter.

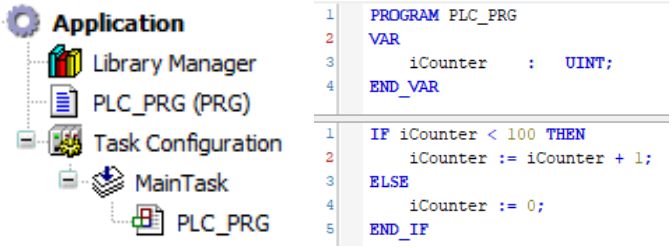


Fig.4.55 Structure of sample project

1. Create a trace object

   Right click on “Application” and select “Add Object” → “Trace”, add trace object. You can set the object name when adding, so please set any name.

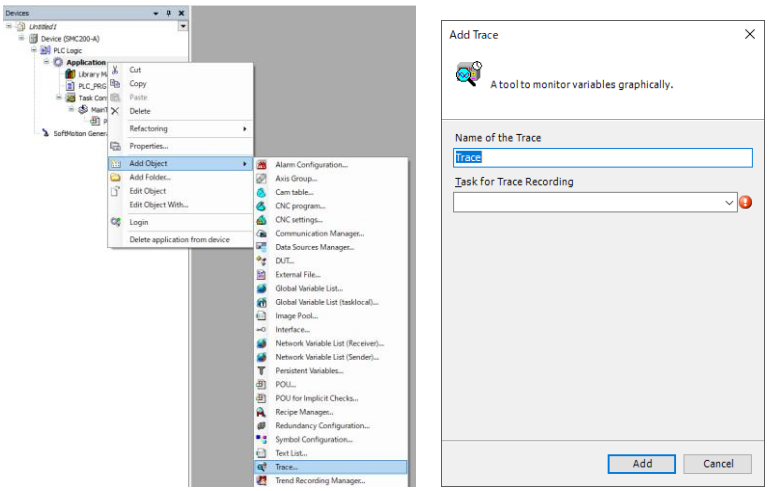


Fig.4.56 Create a trace object

2. Double click on the created trace object. The following trace editor opens.

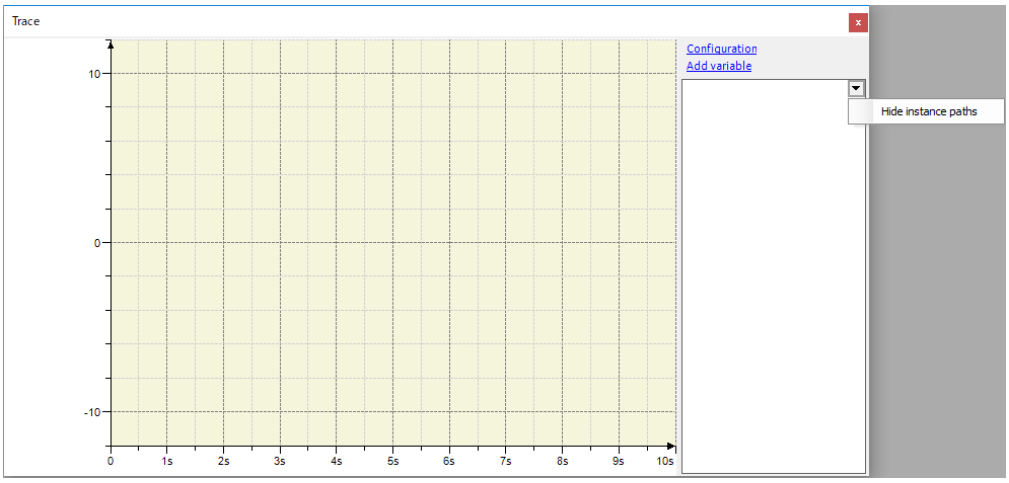


Fig.4.57 Trace

Item	Detail
Configuration	Opens the Trace configuration dialog. The Variable settings are displayed on the right.
Add variable	Adds a new trace variable and opens the Trace configuration dialog with its variable settings. Select a variable in the input field of the Variable setting to trace its value curve.
Hide instance paths	Display of the variable name in the list Example : PLC_PRG.iCounter **Enable** : iCounter **Disable** : PLC_PRG.iCounter

3. Click “Configuration” to set the trace. The task sets MainTask. This is because the sample project uses only MainTask.

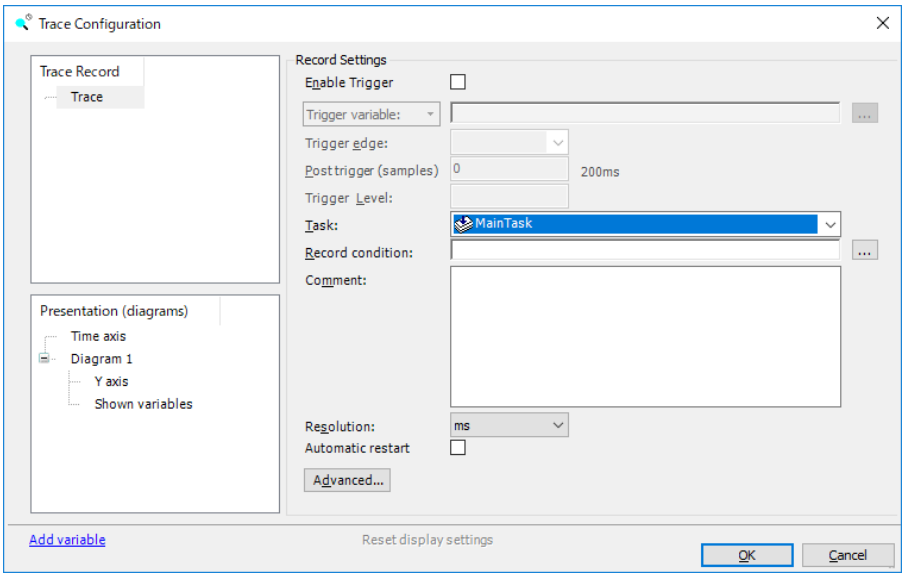


Fig.4.58 Trace Configuration(Task)

Item	Detail
Enable trigger	**Enable**: Triggering is activated. The trace data is buffered in runtime mode only when a trigger signal has been sent. **Disable:** Continuous display of current records
Trigger Variable	Signal that is used as a trigger. A complete instance path is required.
Trigger edge	Defined the edge detection for triggering: **positive** : ▪For Boolean trigger variables, triggering occurs when the values changes from FALSE to TRUE. ▪For analog trigger variables, triggering occurs when the value as defined in Trigger Level is reached from below. **negative** : ▪For Boolean trigger variables, triggering occurs when the values changes from TRUE to FALSE. ▪For analog trigger variables, triggering occurs when the value as defined in Trigger Level is reached from above. **both** : ▪For Boolean trigger variables, triggering occurs when the values changes. ▪For analog trigger variables, triggering occurs when the value as defined in Trigger Level is reached.
Post trigger	Number of records per trace variable that are buffered after triggering. Preset: 50. Value range: 0 to (2^{32} - 1)
Trigger Level	Value that is reached for triggering
Task	Task where data was recorded
Recording condition	In runtime mode, the application checks the recording condition. If it is fulfilled, then the trace data is buffered.
Comment	Comment of Recording
Resolution	Measure for the time stamp that is recorded per data set
Automatic restart	After the device is restarted, the trace is started automatically if the trigger has not occurred yet.
Advanced trace settings	This dialog provides extended settings for recording data.

4. Click “Add Variable” and set the variable to be recorded. In the sample project, the variable sets PLC_PRG.iCounter.

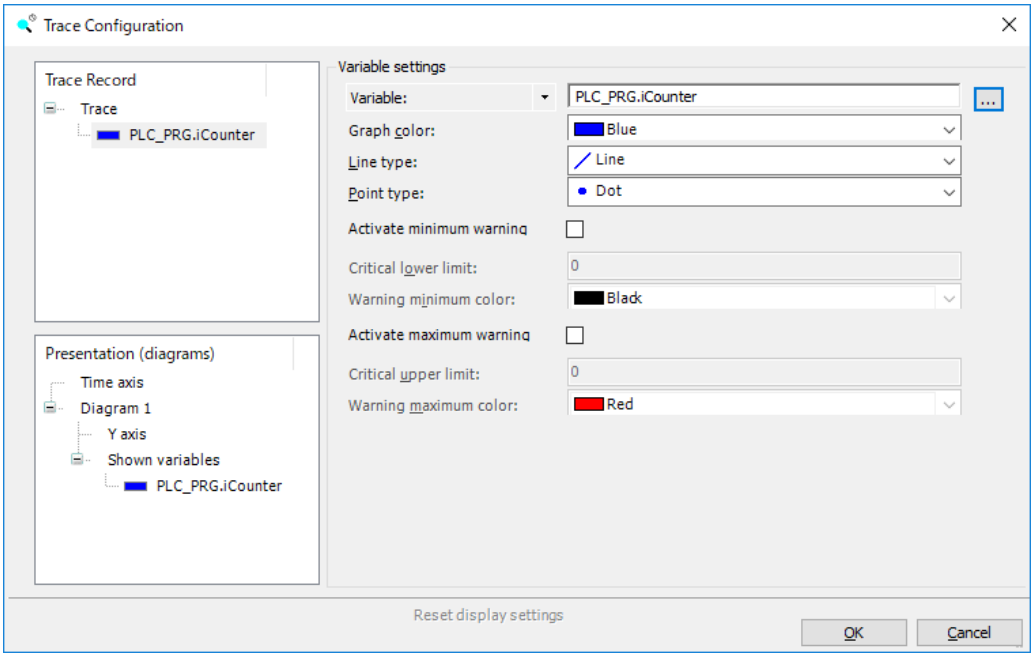


Fig.4.59 Trace Configuration(Variable)

Item	Detail
Variable	Set the variable.
Graph color	Color of the variable in the trace diagram
Line Type	Display as line chart **Line** : Values are linked to form a line. **Step** : Values are linked in the form of steps **None** : Values are not linked
Point type	Display as scatter chart **Dot** : Value is displayed as a dot **Cross** : Value is displayed as a cross. **None** : value is not displayed
Activate minimum warning	Warning when less than the lower limit
Cretical lower limit	If the value of the trace variable falls below the limit, the variable is displayed in the warning color.
Warning minimum color	Warning color on falling below the limit
Activate maximum warning	Warning when exceeding the upper limit
Cretical upper limit	If the value of the trace variable exceeds the upper limit, the variable is displayed in the warning color.
Warning maximum color	Warning color on exceeding the limit

5. The following commands can be used with the trace function. To start tracing, execute “Download trace “.

Symbol	Name	Detail
	Download trace	This command transfers the trace configuration on the controller to the associated application, and starts the data recording. The recorded data is transferred back to the development system. The trace diagram shows the current samples and continues.
	Start Trace	This command starts the data recording on the S200 when it is stopped.
	Stop Trace	This command stops the data recording of a trace.
	Reset Trigger	This command resets the trace configuration after a triggered data recording. Then the application can record new data and react to a trigger again.
	Mouse Zooming	This command enables and disables mouse zooming in the trace diagram.
	Reset View	This command resets the trace diagram to the default view.
	AutoFit	This command scales the y-axis of the trace diagram for optimum display of all graphs, making sure that the y-values fit in the visible region of the diagrams. The command works with both single-channel and multi-channel displays.
	Cursor	This function ▪inserts a trace cursor into the trace diagram when no trace cursor is available ▪inserts a second trace cursor into the trace diagram when 1 trace cursor is available ▪removes the trace cursors when 2 trace cursors are available
	Compress	This command compresses the trace graph by zooming into the displayed time range by a fixed percentage.
	Stretch	This command stretches the trace graph by zooming out of the displayed time range by a fixed percentage.

Trace data is displayed as shown below.

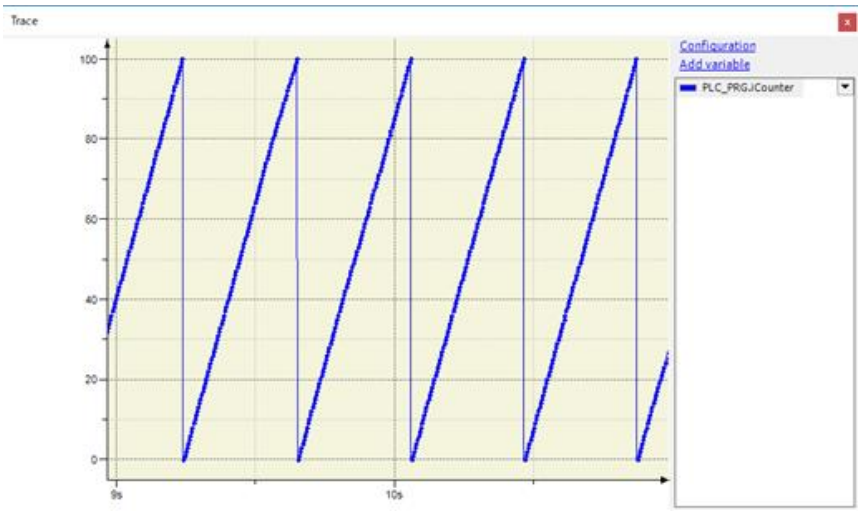


Fig.4.60 Trace editor during execution

4.13.6. Simulation

If you do not have a target device (controller), you can debug the program using simulation. In this case, the application runs on the simulated controller.

To use the simulation function, it is necessary that there are no errors in compiling the program and you are not logged in to the controller.

The following shows how to debug using the simulation function.

1. Select "Online" → "Simulation" from the menu bar, or right click the controller in the device tree and select "Simulation" to enable the simulation mode. When simulation mode is enabled, the controller's name in the device tree is displayed in italics.

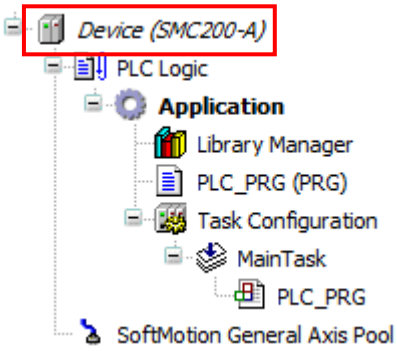


Fig.4.61 Simulation mode

2. Select "Online" → "Login" from the menu bar.

3. The first time you log in as an active application, you will be asked if you want to create and load the application "Sim. <Device name>. <Application name>". Select Yes to continue.

4. Log in to the controller in which the application was simulated. If you log in in simulation mode, a warning symbol is displayed on the controller in the device tree.

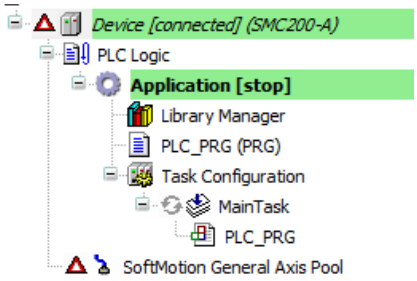


Fig.4.62 Login in simulation mode

5. You can now start debugging. After debugging is complete, you can log out of the controller and exit simulation mode in the same way as enabling.

Debugging in simulation mode is only possible for 30 minutes in a row. A license error will occur 30 minutes after login. To debug again, select "Online"-> "Reset warm" from the menu bar and re-execute the application.

EtherCAT communication cannot be simulated in simulation mode. Also, only "Model number: SMC200-A" can control the EtherCAT slave axis as a virtual axis.

5. Settings in the Web application

5.1. Web application

The Web application is a standard application installed in the S200 that allows you to perform information and settings for the S200 using a web browser.

This section describes network and service setting parameters.

For details of other items, refer to "M0020986 Web Application Instruction Manual".

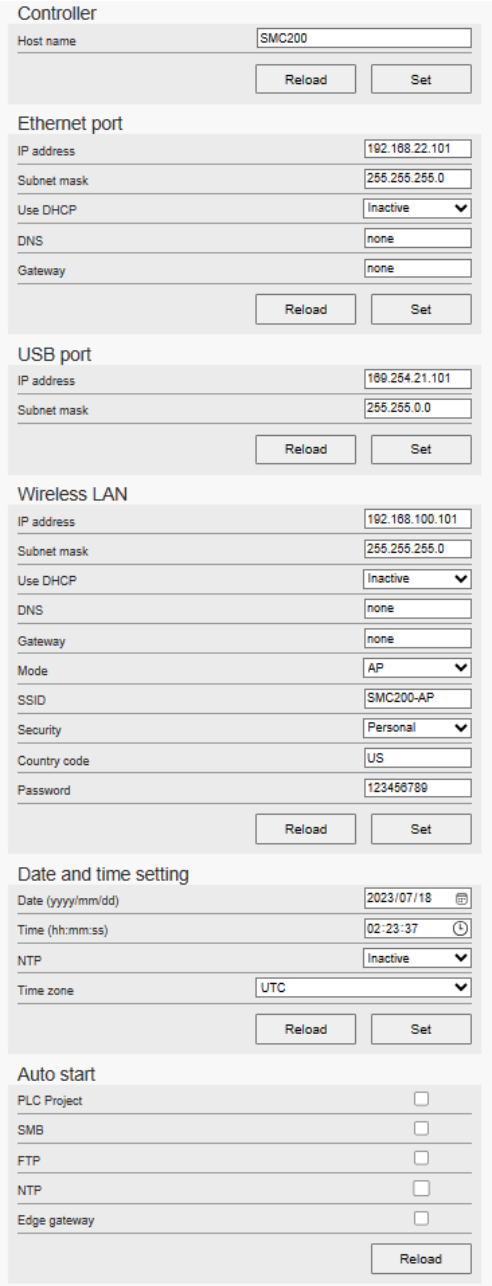


Fig.5.1 Web application setting items

Item	Detail
Controller	Host name
Ethernet port	IP address
	Subnet mask
	Use DHCP（Active / Inactive）
	DNS
	Gateway
USB port	IP address
	Subnet mask
Wireless LAN	IP address
	Subnet mask
	Use DHCP（Active / Inactive）
	DNS
	Gateway
	Mode (AP (Access Point) / STA (Station))
	SSID
	Security (Personal / None)
	Country code（Specify country of use）
	Password
Date and time setting	Current date and time setting
	NTP（Active / Inactive）
	Time zone
Auto start (Automatic startup settings for each service)	PLC Project: application program
	SMB: Samba Server
	FTP: File Transfer Protocol Server
	NTP: Network Time Protocol Server
	Edge gateway: Connection service with cloud services

【Buttons for each item】

Reload: Reads the currently set data from the controller.

Set: Sets the entered value to the controller.

6. Communication function

6.1. EtherCAT

EtherCAT is an open network communication between master and slave using real-time Ethernet. In transmission, when a frame transmitted from the master passes through the slave, the Output data is taken out and the Input data is inserted in the same manner. The EtherCAT slave device can reduce the frame delay time by reading / writing data while passing the frame in that node.

For the EtherCAT communication of SMC200, the daisy chain is adopted as the topology configuration, and the category 5 or more of the cable is recommended.

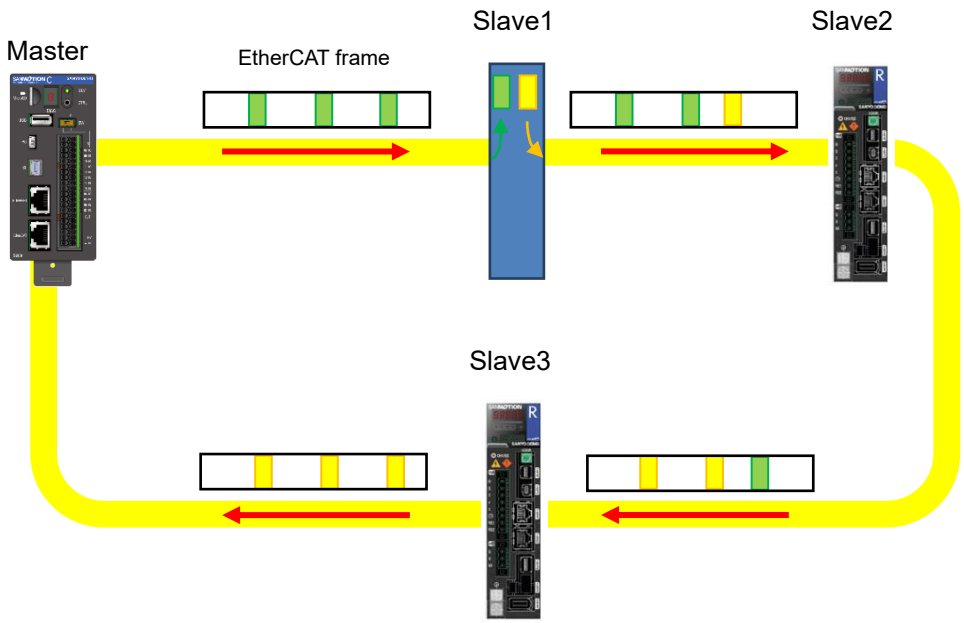


Fig 6.1 EtherCAT frame flow

EtherCAT provides the cyclic communication to transfer the process data periodically, and a mailbox communication for reading / writing of data to any slave at any time.

RS3 EtherCAT amplifier supports CoE (CANopen over EtherCAT). Following two methods are available for accessing from slaves to SANMOTION C as the master.

- PDO（Process Data Object）：Cyclic communication
- SDO（Service Data Object）：Mailbox communication

By accessing the above-mentioned ways, we can change or receive a variety of information.

6.1.1. Supported operation mode

The EtherCAT-CoE specification has various operation modes called operation modes.

The corresponding operation mode for SMC200-A or SMC200-B is shown below.

Operation Mode	SMC200-A	SMC200-B
Profile Position Mode	○	○
Profile Velocity Mode	○	○
Homing Mode	○	○
Cycle Sync. Position Mode	○	×
Cycle Sync. Velocity Mode	○	×
Cycle Sync. Torque (force) Mode	○	×

In profile position mode and profile velocity mode (hereinafter PTP control), the slave generates trajectory. For example, in the case of profile position mode, the target position, profile speed, and profile acceleration / deceleration are passed from the master to the slave, and the slave operates by generating a trajectory. As a result, the CPU load on the S200 can be reduced.

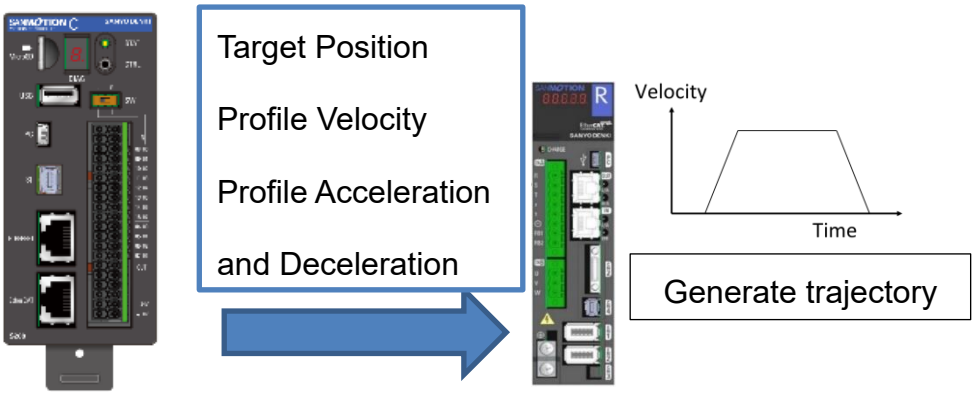


Fig 6.2 Generate trajectory in profile position mode

In Cycle Sync. Position Mode and Cycle Sync. Velocity Mode, EtherCAT master performs generates trajectory. For example, in Cycle Sync. Position Mode, the master performs start generation. After that, the generated trajectory is periodically supplied to the slave as the target position. This allows multiple axes to operate in synchronization.

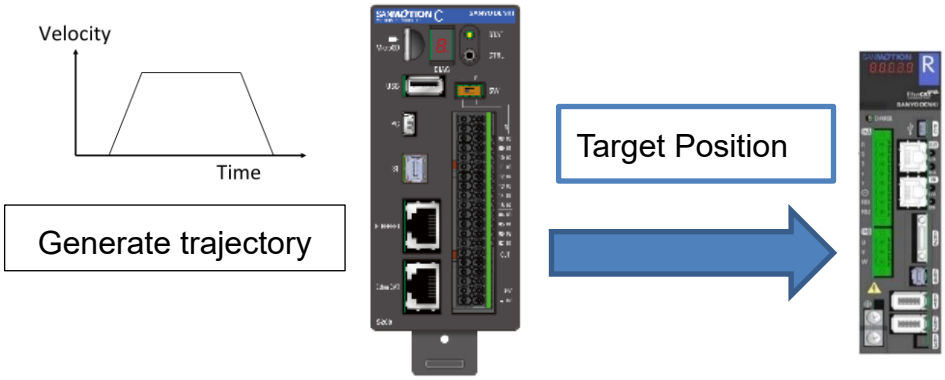


Fig 6.3 Generate trajectory in profile position mode

6.1.2. Object Dictionary

The Object dictionary is one of the features of CANopen, it has a role as an interface between the communication and application.

All of the objects in the Object dictionary consists of 16bit index represented by four hexadecimal digits with a sub-index by 8bit. The summary of the Object dictionary defined by the CoE is as below.

Index(Hex)	Object
0x0000～0x0FFF	Data type Area
0x1000～0x1FFF	Communication Profile Area
0x2000～0x5FFF	Manufacturer Specific Profile Area
0x6000～0x9FFF	Standardized Device Profile Area
0xA000～0xFFFF	Reserved

Each object has the following parameters:

- Data type ： Data type（BOOLEAN，Usigned32 etc.） of objects
- Access rights ： Access restrictions to object（RW，WO，RO，CONST）
- PDO mapping ： PDO mapping of objects enable / disable (Possible, No)
- Update ： Effective timing of the writing of the data in the SDO communication (with immediate effect, ESM transition requirements, effectiveness in the control power is restored)

When accessing objects, those parameters need to be cehceked. Refer to the instruction manual of the servo amplifier for the parameters of each object.

By read / ride to the object dictionary entries, various parameters of the slave amplifier such as device settings, monitoring are controlled.

6.1.3. Process Data Object(PDO)

EtherCAT real-time communication uses the PDO communication. PDO communication is high priority message sent by the broadcast. Therefore, PDO communication is suitable for the transmission of real-time data (control of I/O modules, the measured values of sensors, etc.). Setting the object to cyclic communication data is called PDO mapping.

The default PDO mapping is set according to the contents of the EtherCAT slave ESI file to be added. Refer to【Expert Process Data】in “6.1.5.2 EtherCAT slave setting” for how to edit PDO mapping.

6.1.4. Service Data Object(SDO)

SDO is the method to access all of the entries of the object dictionary using the request and response messages. SDO communication is asynchronous communication. That can't read/write parameters in real time because executed only if it is possible to reply in the intervals of the PDO communication by the command from the controller.

SDO communication is a way of communication that aims to get or to set slave parameters before starting PDO communication, or to check state of slaves during interval of PDO communication. There is Start-up parameter functions as one of the SDO communication setting way before starting PDO communication.（See【Startup Parameters】in ”6.1.5.2 EtherCAT slave setting”。SDO communication in interval of PDO communication can be achieved by using of the function blocks.（See “6.1.6 Function block for SDO communication”）.

6.1.5. EtherCAT device editor

6.1.5.1. EtherCAT master setting

The setting items of the EtherCAT master are shown below. Since EtherCAT NIC settings are set in the template project, please do not change.

"Auto config Master / Slave" automatically configures the master and slave. This configuration is activated by default and is strongly recommended for standard applications as it is usually sufficient for auto-configured content. Deactivating this option requires you to have expert knowledge, as you will have to do all the configuration of the master and slave manually.

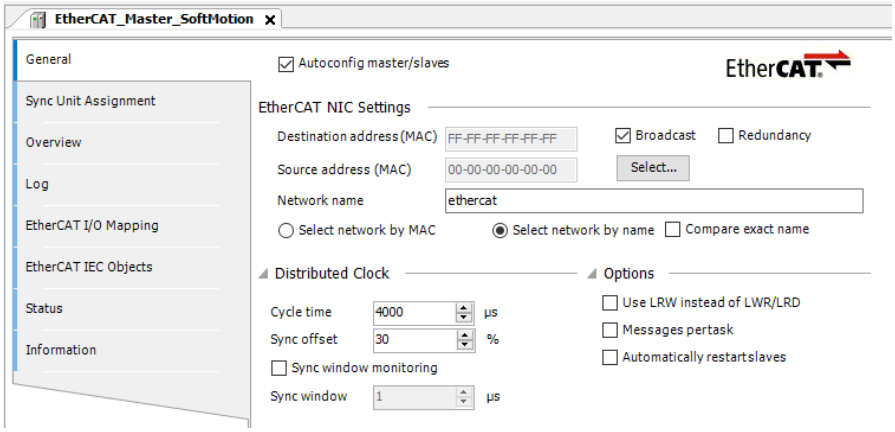


Fig 6.4 EtherCAT master setting screen

Item	Detail
Cycle time	Set the cycle time of EtherCAT. It is possible to set 2000µs, 4000µs, 8000µs, 16000µs as the cycle time. Recommend more than 2000µs. Purpose: Motion control 1-4 Axis / Motion control 5-8 Axis / Robot control 4 Axis Cycle time: **2000µs～** / **8000µs～** / **8000µs～**
Sync Offset	Enables the time delay of the sync interrupt of the EtherCAT slave to be adjusted to the cycle time of the PLC. Please set 30%.
Use LRW instead of LWR/LRD	If this option is enabled, LRW (read/write command) is used for EtherCAT communication. If disabled, LRD (read command) and LWR (write command) are sent in separate commands.
Enable messages per task	If this option is enabled, PDO-mapped variables will be updated in the task of the POU where the variable is used. If disabled, PDO mapped variables will be updated in EtherCAT_Task. [Recommended: Disabled]
Automatic Restart Slaves	With this option enabled, if communication with a slave is interrupted, the master will restart the slave and try again. If disabled, the slave will not restart even if communication is interrupted. [Recommended: Disabled]

If you connect a slave different from the configured EtherCAT slave with 'Enable slave auto start' enabled, reconnection processing is always performed, so the S200's CPU load may become large and it may not operate properly.

When enabling this option, please check the connection status carefully.

6.1.5.2. EtherCAT slave setting

The following items can be set in the slave setting. The slave setting screen is displayed by double clicking on the added slave.

【General】

On the "General" tab, you can set slave parameters.

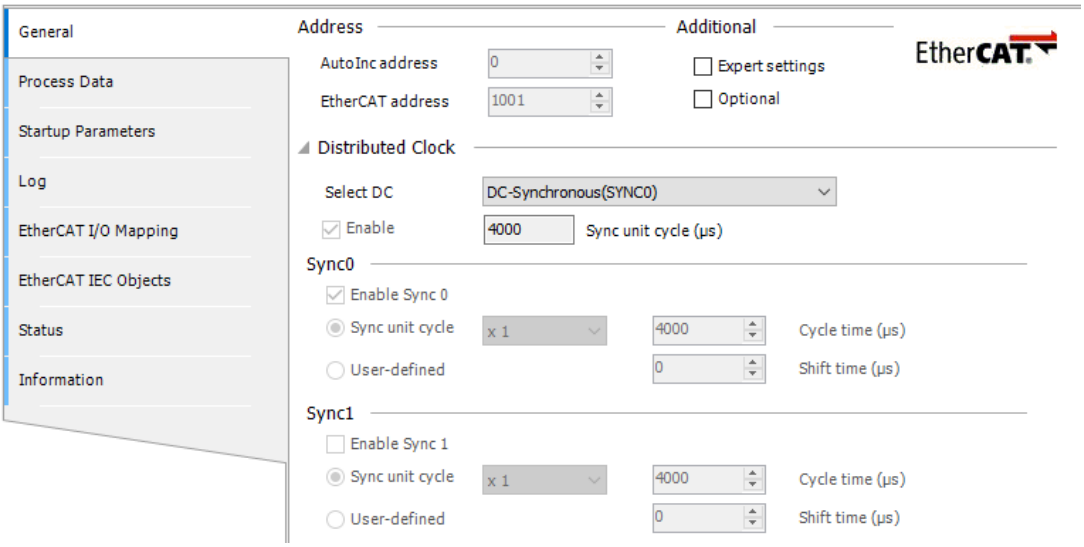


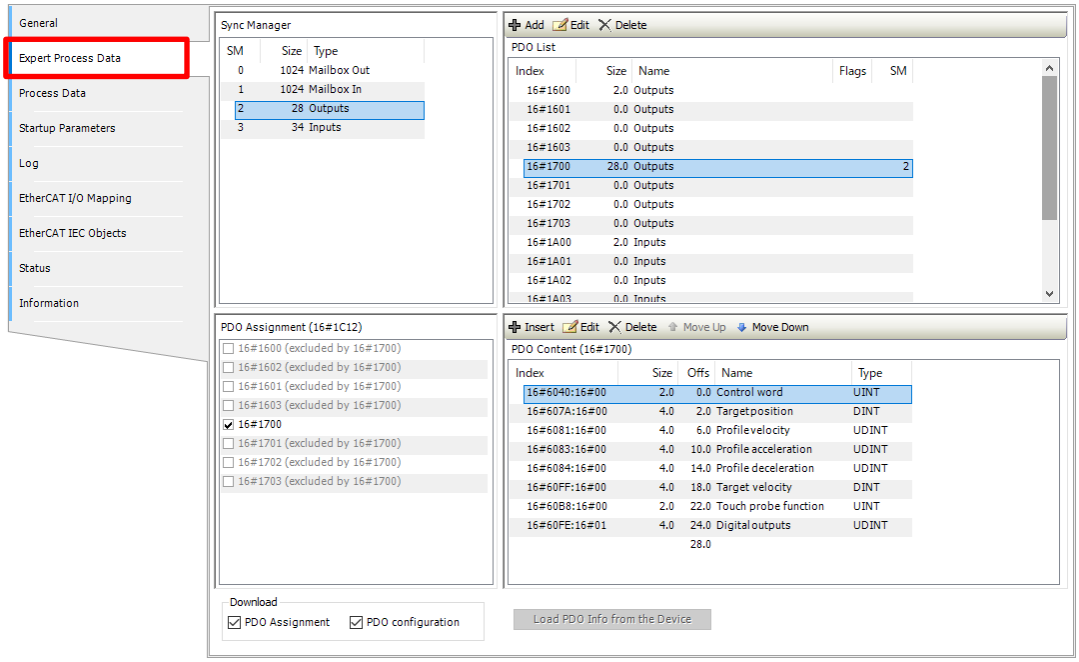
Fig 6.5 Slave setting screen (general)

Item	Detail
Address	Sets the address used for EtherCAT communication. Fields can be edited only when the auto-configuration mode of the EtherCAT master is deactivated.
Distributed Clock	Drop-down list containing settings for distributed clocks as described in the slave's device description file.
Additional	**Expert settings**: If this option is enabled, special settings such as check items for slave startup and timeout are added. In addition, the "Expert Process Data" tab is displayed and the PDO mapping can be edited. **Optional**: If this option is enabled, the station alias address can be set for EtherCAT slaves that support the station alias function.

For details on each parameter, refer to "Fieldbus Support"⇒ "EtherCAT Configurator"⇒ "EtherCAT Slave"⇒ "Tab 'EtherCAT-Slave - General' in the online help.

【Expert Process Data】

"Expert Process Data" tab is only displayed if "Enable Expert Settings" is active and allows you to edit the PDO configuration.



item	detail
Sync Manager	A list of data sizes allocated to Sync Manager by type is displayed, and the PDO mapping list allocated to the selected Sync Manager is displayed in "PDO Assignment".
PDO Assignment	You can select the PDO mapping to be assigned to Sync Manager from the list. The same settings can be made on the "Process Data" tab.
PDO List	The total size of the objects entered in each PDO mapping and the assigned Sync Manager channel number are displayed, and the objects entered in the selected PDO mapping are displayed in "PDO Content".
PDO Content	You can edit the PDO mapping selected in "PDO List".

The method of mapping "0x2103.01 warning status" in RS3 EtherCAT amplifier using "Expert Process Data" is described below.

1. Since "Warning status" is a read-only object, select the PDO mapping assigned to SM3 in the "PDO List", and select "Insert" in "PDO Content".

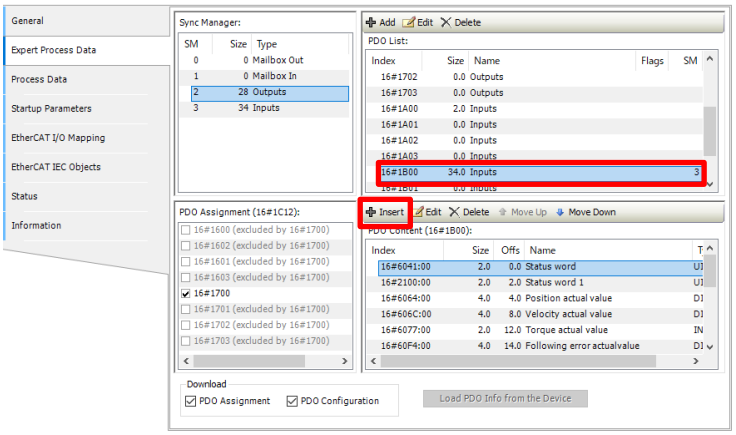


Fig 6.6 Select PDO Mapping to Edit

2. A list of object dictionaries defined in the ESI file will be displayed, so select “0x2103.01 Warning status” and click OK.

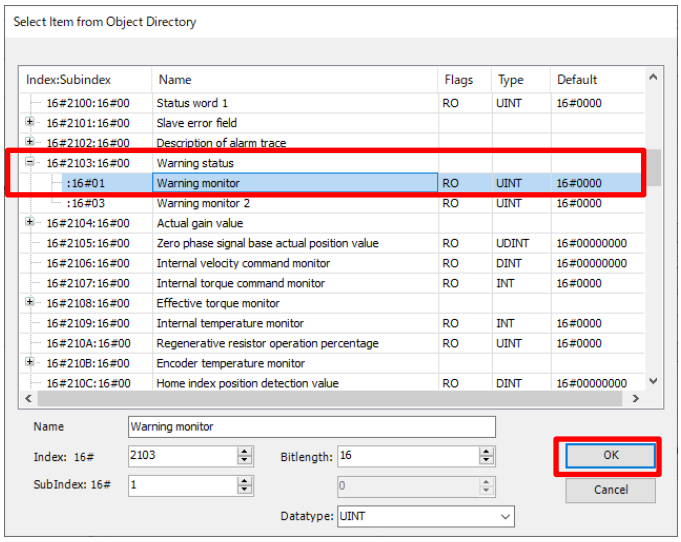


Fig 6.7 Add 0x2103.01 Warning status

3. Confirm that the warning status has been added in the "Process Data" tab.

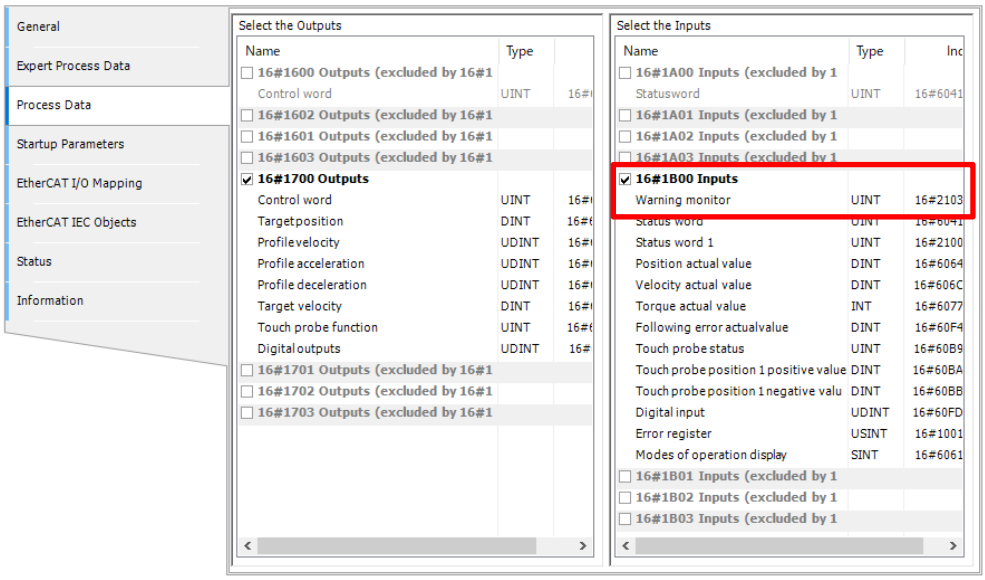


Fig 6.8 Confirmation 0x2103.01 Warning status

【Startup Parameters】

On the "Startup Parameters" tab, you can set parameters of the slave using SDO.

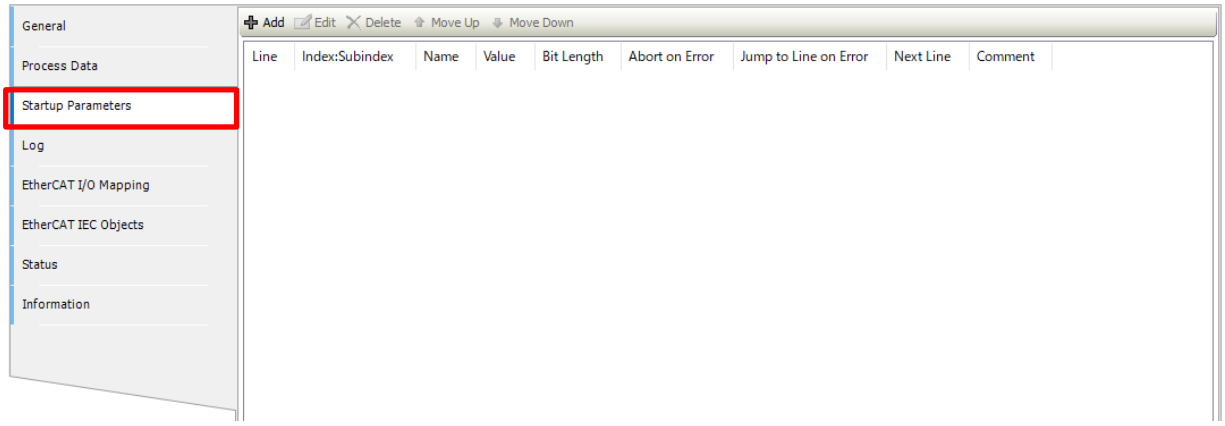


Fig.6.9 Slave setting screen (Startup Parameter)

I will explain how to change the homing mode by using "startup parameter". Please set according to the following procedure.

1. Please click "Add".

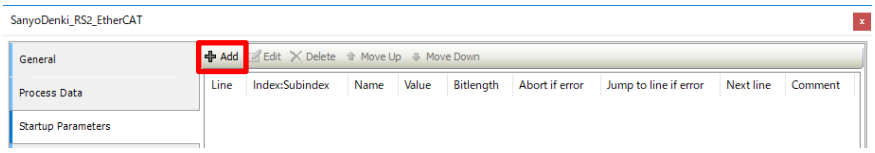


Fig.6.10 Add startup parameters

2. Please select the object dictionary you want to change. For this time, please select "16 # 6098: 16 # 00 Homing method". Enter the value of the homing method you want to set to the value. Then click "OK".

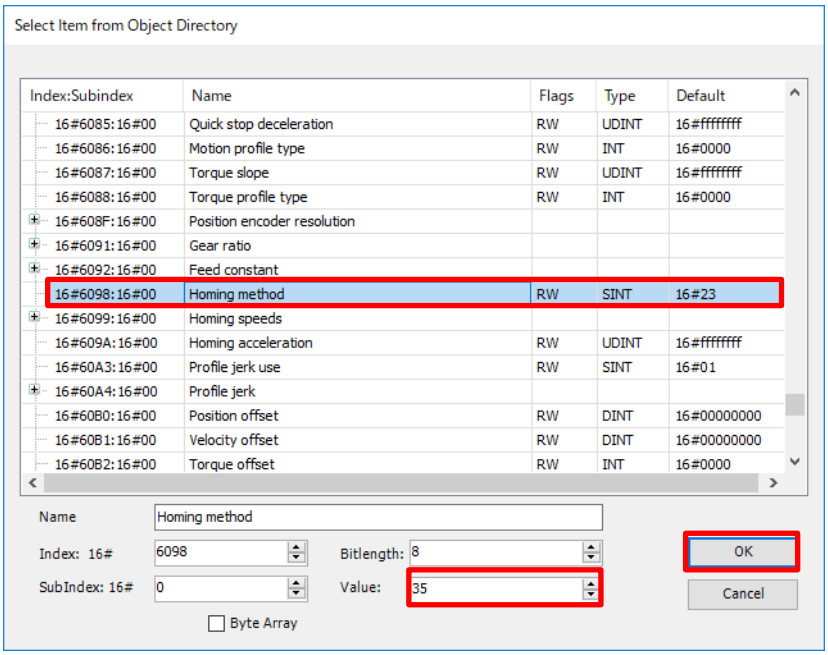


Fig.6.11 Selection of items from target object dictionary

3. After the addition, the display will be as follows.

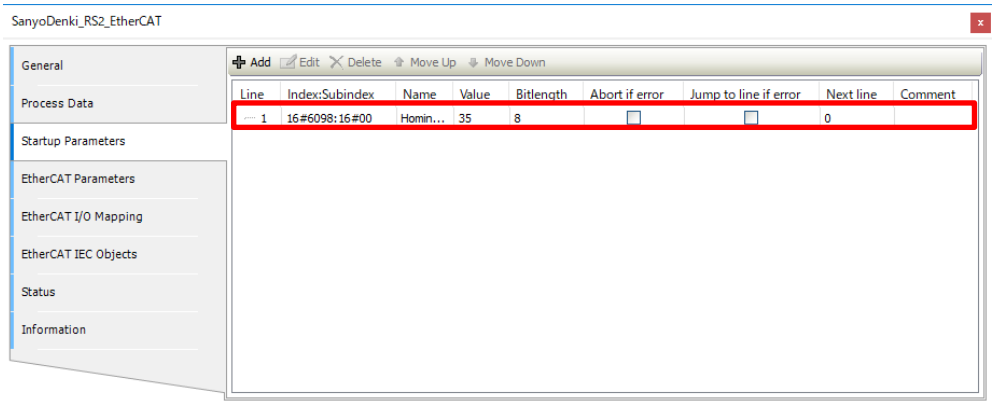


Fig.6.12 Slave setting screen

6.1.6. Function block for SDO communication

6.1.6.1. ETC_CO_SdoRead

This function block is used to read amplifier parameters in SDO communication.

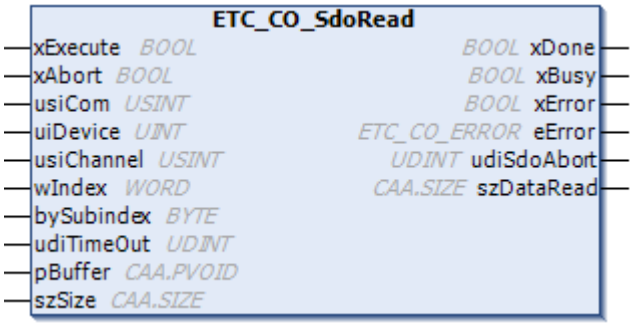


Fig 6.13 ETC_CO_SdoRead

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the reading of the slave parameters. In order to release the internal channel again afterwards, the instance must be called at least once with xExecute: FALSE.
xAbort	BOOL	TRUE: The current read process is aborted.
usiCom	USINT	Number of the EtherCAT master: usiCom is always 1 if only one EtherCAT master is used. If several masters are used, 1 designates the first, 2 the second and so on.
uiDevice	UINT	Physical address of the slave. If the auto-configuration mode is deactivated in the master, the slave can be given its own address. This address must be specified here. If the auto-configuration mode is activated, the first slave is given the address 1001. The current address of a slave is always located in the Slave dialog of the slave in the EtherCAT address field.
usiChannel	USINT	Reserved for future extensions
wIndex	WORD	Index of the parameter in the object directory.
bySubindex	BYTE	Subindex of the parameter in the object directory.
udiTimeOut	UDINT	Definition of the watchdog time in milliseconds. If the reading of the parameters is not yet complete on expiry of this time, an error message is output.
pBuffer	CAA.PVOID	Pointer to a data buffer in which the data are stored after a successful parameter transfer
szSize	CAA.SIZE	Size of the data buffer (pBuffer) in bytes

VAR_OUTPUT		
xDone	BOOL	TRUE: Reading of the parameter was completed without error.
xBusy	BOOL	TRUE: Reading is not yet completed.
xError	BOOL	TRUE: An error occurred during reading.
eError	ETC_CO_ERROR	Information about the cause of the error that was displayed by xError, e.g. ETC_CO_TIMEOUT in case of a timeout
udiSdoAbort	UDINT	If an error has occurred in the device, this output provides further information about it
szDataRead	CAA.SIZE	Number of bytes read; maximally szSize (input).

6.1.6.2. ETC_CO_SdoWrite

This function block is used to write amplifier parameters in SDO communication.

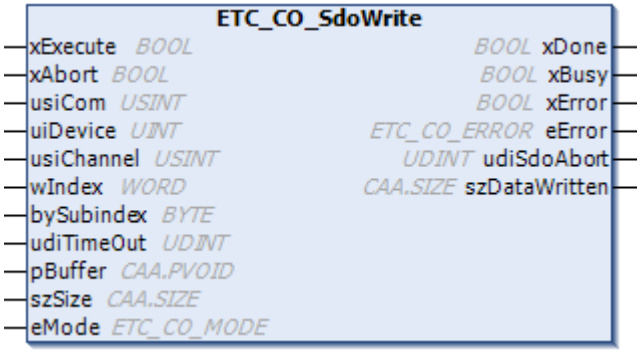


Fig 6.14 ETC_CO_SdoWrite

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the reading of the slave parameters. In order to release the internal channel again afterwards, the instance must be called at least once with xExecute: FALSE.
xAbort	BOOL	TRUE: The current read process is aborted.
usiCom	USINT	Number of the EtherCAT master: usiCom is always 1 if only one EtherCAT master is used. If several masters are used, 1 designates the first, 2 the second and so on.
uiDevice	UINT	Physical address of the slave. If the auto-configuration mode is deactivated in the master, the slave can be given its own address. This address must be specified here. If the auto-configuration mode is activated, the first slave is given the address 1001. The current address of a slave is always located in the Slave dialog of the slave in the EtherCAT address field.
usiChannel	USINT	Reserved for future extensions
wIndex	WORD	Index of the parameter in the object directory.
bySubindex	BYTE	Subindex of the parameter in the object directory.

VAR_INPUT		
udiTimeOut	UDINT	Definition of the watchdog time in milliseconds. If the reading of the parameters is not yet complete on expiry of this time, an error message is output.
pBuffer	CAA.PVOID	Pointer to a data buffer containing the data to be written.
szSize	CAA.SIZE	Size of the data buffer (pBuffer) in bytes
eMode	ETC_CO_MODE	AUTO mode is usually set and the mode suitable for the length is thus automatically used.
VAR_OUTPUT		
xDone	BOOL	TRUE: Reading of the parameter was completed without error.
xBusy	BOOL	TRUE: Reading is not yet completed.
xError	BOOL	TRUE: An error occurred during reading.
eError	ETC_CO_ERROR	Information about the cause of the error that was displayed by xError, e.g. ETC_CO_TIMEOUT in case of a timeout
udiSdoAbort	UDINT	If an error has occurred in the device, this output provides further information about it
szDataWritten	CAA.SIZE	Number of bytes written; maximally szSize (input).

6.1.7. PDO communication

To send and receive data by PDO communication, assign variables to the EtherCAT object according to the following procedure.

6.1.7.1. Assign variables

Follow the procedure below to assign variables to EtherCAT objects.

1. Double-click the EtherCAT device in the device list to display the setting screen and select "EtherCAT I / O Mapping".

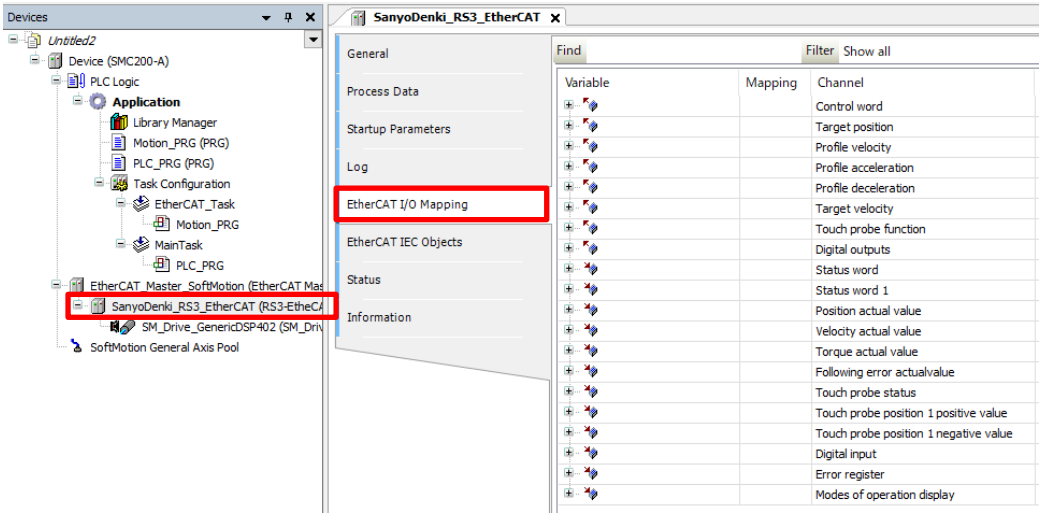


Fig 6.15 EtherCAT I/O Mapping

2. Select EtherCAT I / O Mapping to display a list of objects currently assigned to PDO. Enter any variable name in the "Variable" column of the data you want to use.

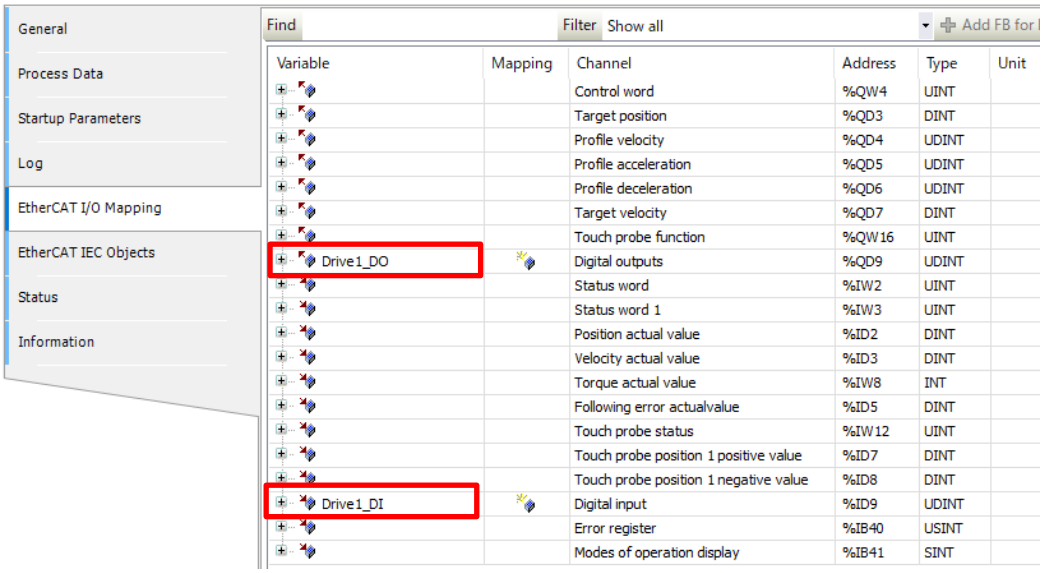


Fig 6.16 Assign variables

6.1.7.2. Use variables

The variables assigned in "6.1.7.1. Assign variables" can be used in the program. The figure below is an example of a program that uses the assigned variables "Drive1_DO" and "Drive1_DI".

EtherCAT objects include read-only objects and read-write objects. For details on the objects, refer to the instruction manual of each device.

```
PLC_PRG
1  PROGRAM PLC_PRG
2  VAR
3      udiDrive1_DigitalOutput: UDINT;
4      udiDrive1_DigitalInput: UDINT;
5  END_VAR

1  udiDrive1_DigitalOutput := Drive1_DO;
2  udiDrive1_DigitalInput := Drive1_DI;
3
```

Fig 6.17 Use variables

For EtherCAT devices used as axes, the system performs PDO communication for motion control.

Objects used by the system cannot be written, they can only be read.

6.2. EtherNet/IP

This S200 supports EtherNet/IP scanners and adapters. EtherNet/IP is an industrial protocol using Ethernet. By using EtherNet/IP, data can be exchanged between multi-vendor products. This S200 supports data exchange via cyclic communication (Implicit communication) and data exchange via message communication (Explicit communication).

6.2.1. Basic specifications

Item		Detail
Scanner	Maximum number of connected devices	4
	Minimum communication cycle	50ms
	Task Interval	IOTask :50ms
		ServiceTask :20ms
	Priority	IOTask :1
		ServiceTask :30
Adapter	Device Type	12
	Maximum number of communication data	Output:508byte , Input:504byte (Recommended is 128 bytes for output and 128 bytes for input)
	Minimum communication cycle	50ms
	Task Interval	IOTask :25ms
		ServiceTask :20ms
	Priority	IOTask :1
		ServiceTask :30
	Corresponding CIP class object	Identity Object Message Router Object Assembly Object Connection Manager Object TCP/IP Interface Object EtherNet Link Object
	Configurable data types	BYTE (1byte) WORD (2byte) DWORD (4byte) REAL (4byte) Big (A set of BYTE type data for the maximum number of communication data)

EtherNet/IP task is generated automatically when EtherNet/IP device is added. Do not change the priority of automatically generated tasks.

It is not possible to execute the scanner function and the adapter function at the same time on one S200.

6.2.2. Adapter setting procedure

6.2.2.1. Adapter addition procedure

Describes the procedure for adding an adapter. Use "PLC standard project" as a template.

1. Right-click "Device" and select "Add Device…".

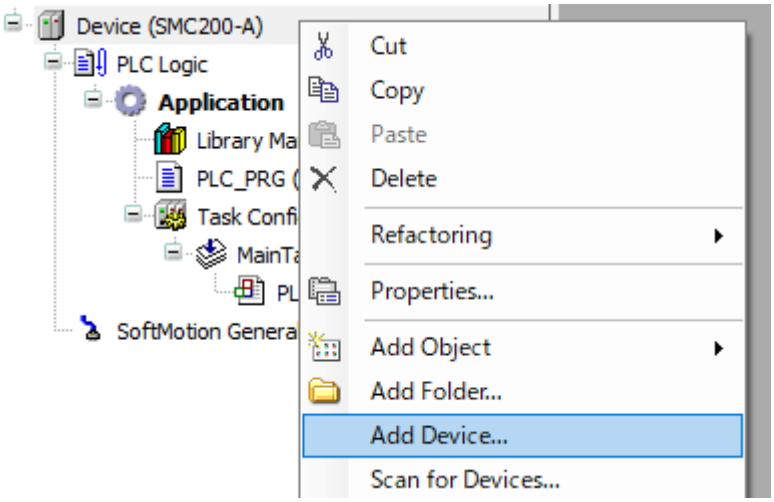


Fig 6.18 Add Device

2. The Add Device window will open. Double-click on "Ethernet".

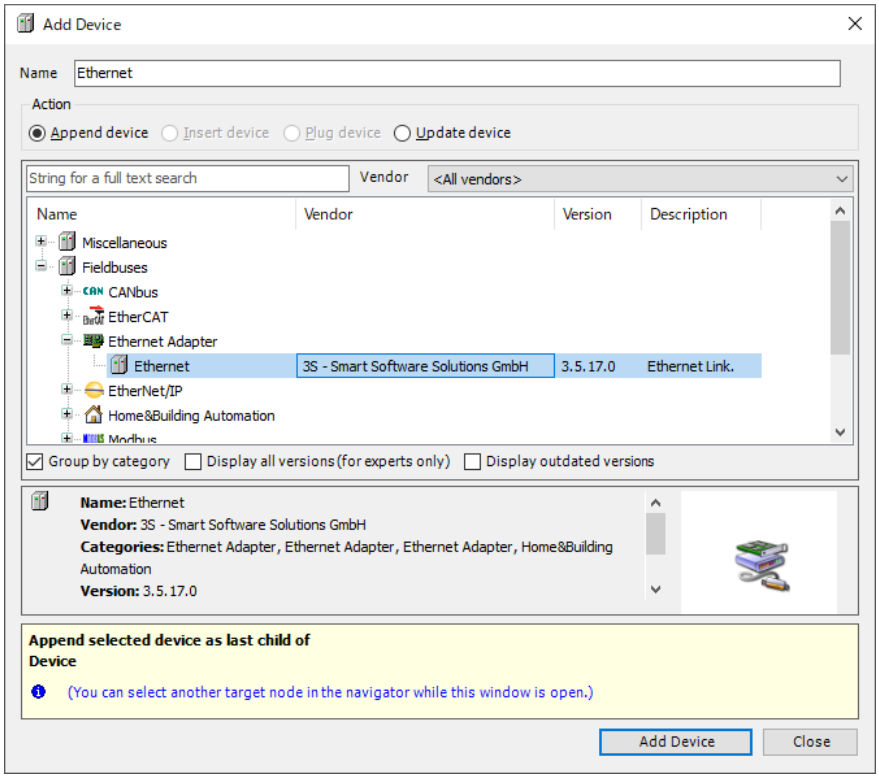


Fig 6.19 Add Device window

3. With "Ethernet" selected, double-click "EtherNet/IP Adapter".

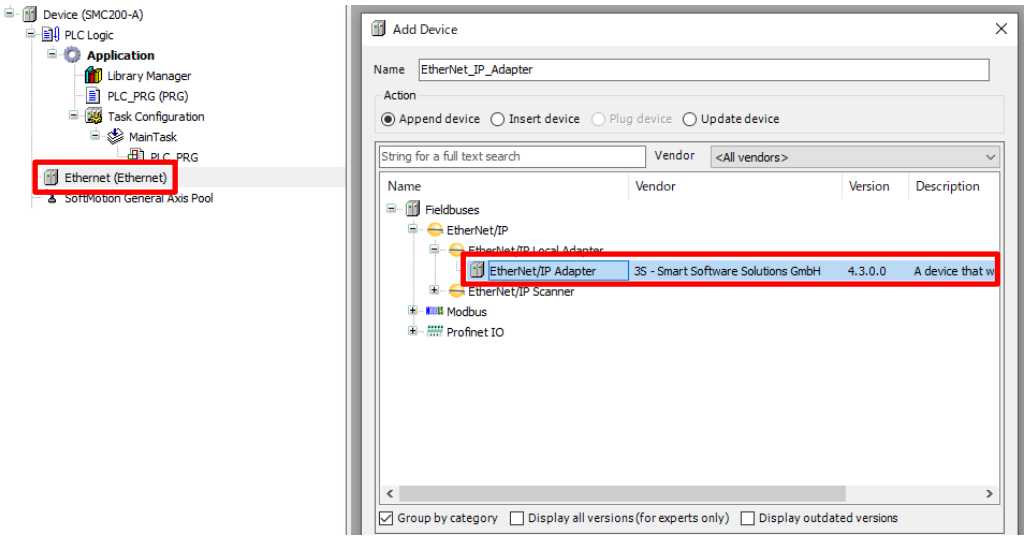


Fig 6.20 Add Adapter

4. Add "EtherNet/IP Module" with "EtherNet_IP_Adapter" selected.

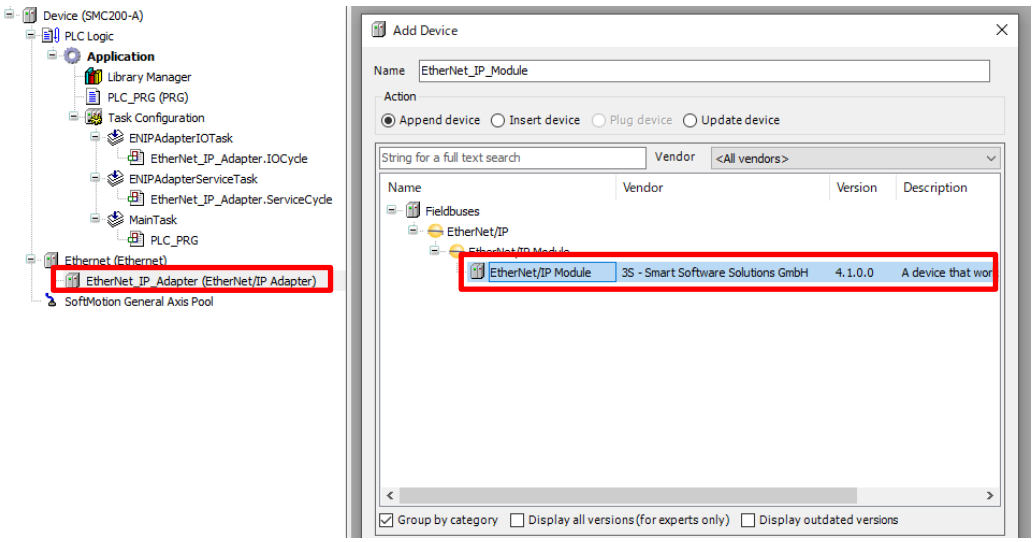


Fig 6.21 Add Module

6.2.2.2. Ethernet settings

You can open the setting screen by double-clicking "Ethernet" in the device tree. The following items can be set on the Ethernet setting screen.

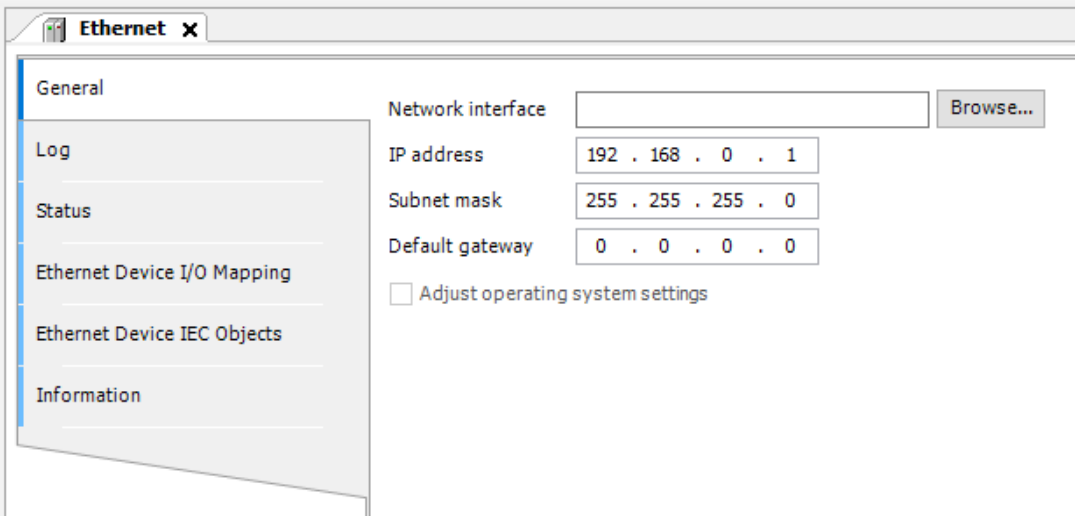
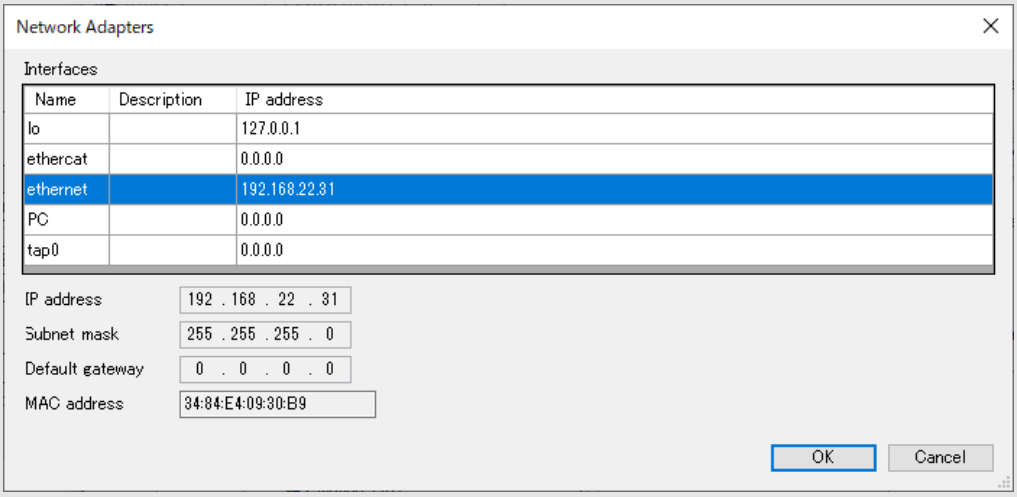


Fig 6.22 Ethernet setting screen

Item	Detail
Network Interface	Set the name of the interface (ethernet) to be used.
IP Address	Set the IP address currently set to ETHERNET port.
Subnet Mask	Set the Subnet Mask currently set to ETHERNET port.
Default Gateway	Set the Default Gateway currently set to ETHERNET port.
Adjust Operating System Settings	Unavailable for manufacturer maintenance.

If the target S200 is selected by network scan, you can set all items automatically by selecting "Browse..." and selecting "ethernet".



6.2.2.3. Adapter settings

Double-click the device tree “EtherNet_IP_Adapter” to open the adapter settings screen. The following items can be set on the adapter setting screen.

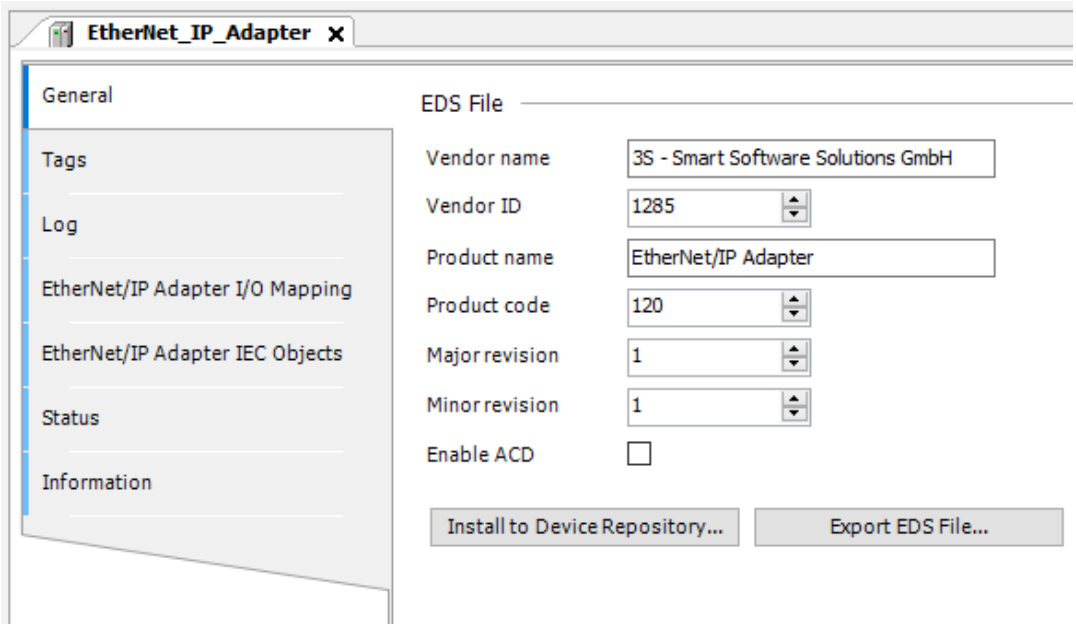


Fig 6.23 Adapter setting screen

Item	Detail
Vendor Name	Vendor name set in EDS file (do not change from the initial setting)
Vender ID	Vendor ID set in EDS file (do not change from the initial setting)
Product Name	Product Name set in EDS file (do not change from the initial setting)
Product Code	Product Code set in EDS file (do not change from the initial setting)
Major Revision	Major Revision set in EDS file (do not change from the initial setting)
Minor Revision	Minor Revision set in EDS file (do not change from the initial setting)
Install to Device Repository…	Install the adapter to the device repository with the settings you have made.
Export EDS File…	You can output an EDS file that reflects the current settings. The module settings performed in "6.2.2.4Module settings" are also reflected.

*Please do not change the default settings for vendor name, vendor ID, product name, product code, major revision, and minor revision.

6.2.2.4. Module settings

You can open the module setting screen by double-clicking "EtherNet_IP_Module" in the device tree.

【Assemblies】

The Assembly tab allows you to set the data types to use.

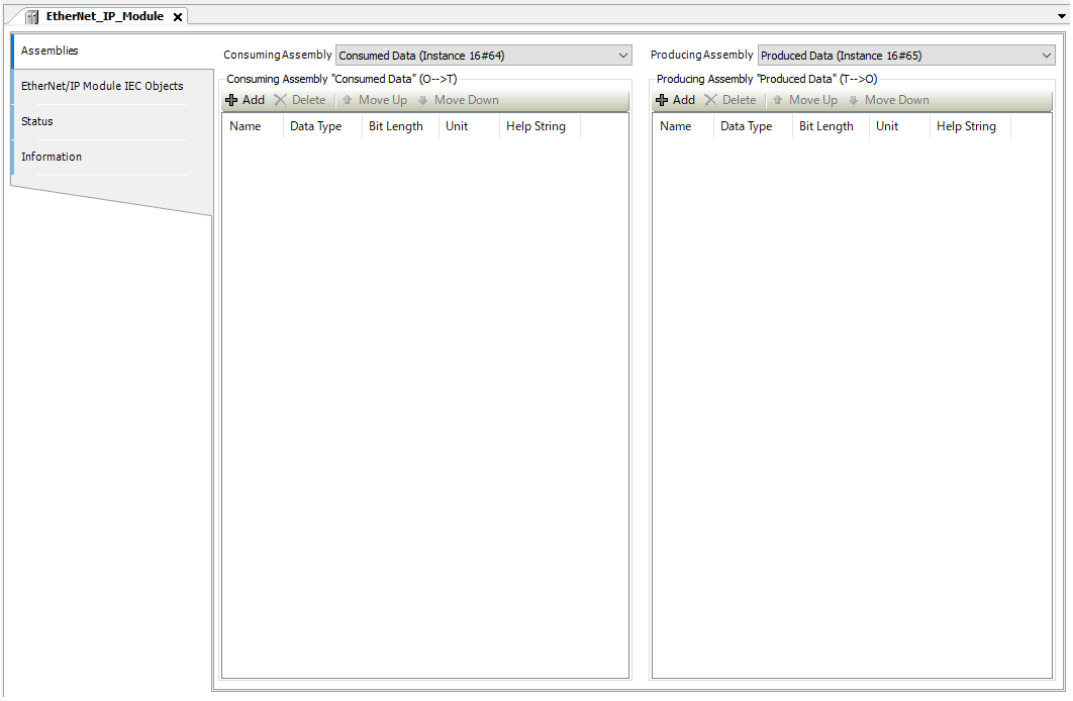


Fig 6.24 Assemblies tab

【EtherNet/IP Module I/O Mapping】

The EtherNet/IP Module I/O Mapping tab allows you to assign variables to cyclically updating data on EtherNet/IP. The allocated variables can be used within the program as global variables.

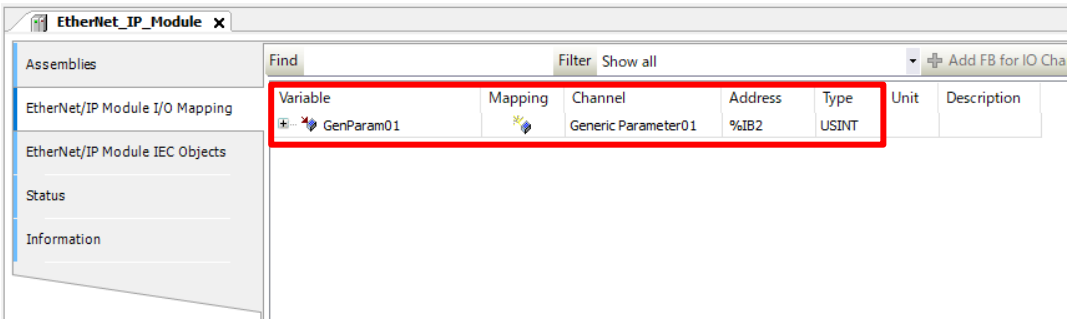


Fig 6.25 EtherNet/IP Module I/O Mapping tab

6.2.3. CIP object

The CIP objects mainly used in the EtherNet/IP adapter of this S200 are listed below.

6.2.3.1. Identity Object (Class Code : 0x01)

【Supported service code】

Service code	Name	Description	Instance ID
0x01	Get_Attributes_All	Get attribute value.	0x01
0x0E	Get_Attribute_Single	Returns the contents of the specified attribute.	0x01
0x05	Reset	Reset EtherNet/IP communication.	0x01

【Instance Attribute (Instance ID : 0x01)】

Attr. ID	Name	Data Type	Access	Description
1	Vendor ID	UINT	Read Only	Manufacturer identification (Default value：179)
2	Device Type	UINT	Read Only	Indication of general type of product (Default value：12)
3	Product Code	UINT	Read Only	Identification of the particular product (Default value SMC200-A：120)
4	Revision	STRUCT of:{	Read Only	Revision
	↳Major Revision	UINT	Read Only	Major Revision (Default value：1)
	↳Minor Revision	USINT }	Read Only	Minor Revision (Default value：1)
5	Status	WORD	Read Only	Current status
6	Serial Number	UDINT	Read Only	Serial number (the value after the third octet of the MAC address)
7	Product Name	STRING	Read Only	Product name (Default value Sanyodenki_ SMC200-A)

6.2.3.2. TCP/IP Interface Object (Class Code : 0xF5)

【Supported service code】

Service code	Name	Description	Instance ID
0x0E	Get_Attribute_Single	Returns the contents of the specified attribute.	0x00, 0x01
0x10	Set_Attribute_Single	Modifies a single attribute.	0x01

【Class Attribute (Instance ID : 0x00)】

Attr. ID	Name	Data Type	Access	Description
1	Revison	UINT	Read Only	Revision of the object (Default value：4)

【Instance Attribute (Instance ID : 0x01)】

Attr. ID	Name	Data Type	Access	Description
1	Status	DWORD	Read Only	Interface status
2	Configuration Capability	DWORD	Read Only	Interface capability flags (Fixed value：0x20)
3	Configuration Control	DWORD	Read Only	Interface S200 flags (Fixed value：0x00)
4	Physical Link Object	STRUCT of:{	Read Only	Path to physical link object
	↳Path size	UINT	Read Only	Size of Path
	↳Path	Padded EPATH }	Read Only	Logical segments identifying the physical link object
5	Interface Configuration	STRUCT of: {	Read Only	TCP/IP network interface config
	↳IP Address	UDINT	Read Only	IP address
	↳Network Mask	UDINT	Read Only	Network mask
	↳Gateway Address	UDINT	Read Only	Default gateway
	↳Name Server	UDINT	Read Only	Primary name server
	↳Name Server 2	UDINT	Read Only	Secondary name server
	↳Domain Name	UDINT }	Read Only	Default domain name
6	Host Name	STRING	Read Only	Host name (Fixed value：SMC200)
13	Encapsulation Inactivity Timeout	UINT	Read/Write	Number of seconds of inactivity before TCP connection is closed. Maximal value is 3600. (Default value：120)

6.2.3.3. Assembly Object (Class Code : 0x04)

【Supported service code】

Service code	Name	Description	Instance ID
0x0E	Get_Attribute_Single	Returns the contents of the specified attribute.	0x100, 0x101

【Instance Attribute】

■Output assembly (Instance ID : 0x100)

Attr. ID	Name	Data Type	Access	Description
3	Output assembly	By setting module data type	Read Only	Current value of write only data

■Input assembly (Instance ID : 0x101)

Attr. ID	Name	Data Type	Access	Description
3	Input assembly	By setting module data type	Read Only	Current value of read only data

6.2.4. Scanner setting procedure

6.2.4.1. Add scanner procedure

Describes the procedure for adding a scanner. Use "PLC standard project" as a template.

1. Right-click "Device" and select "Add Device…".

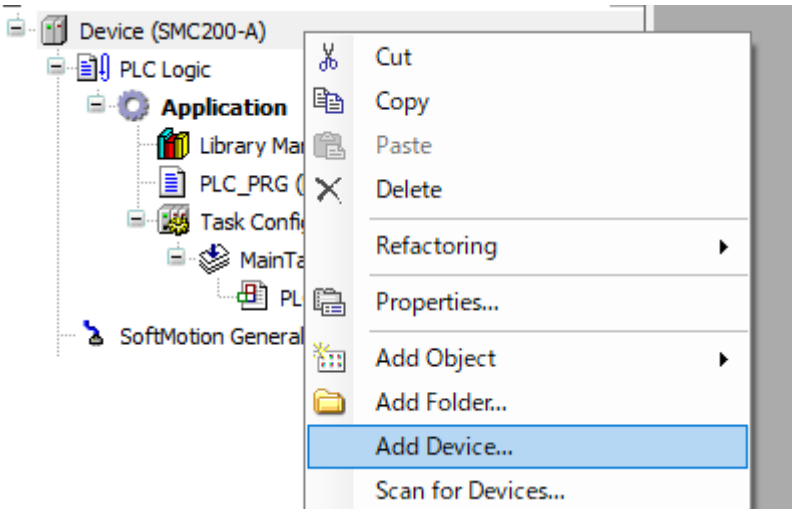


Fig 6.26 Add Device

2. The Add Device window will open. Double-click on "Ethernet".

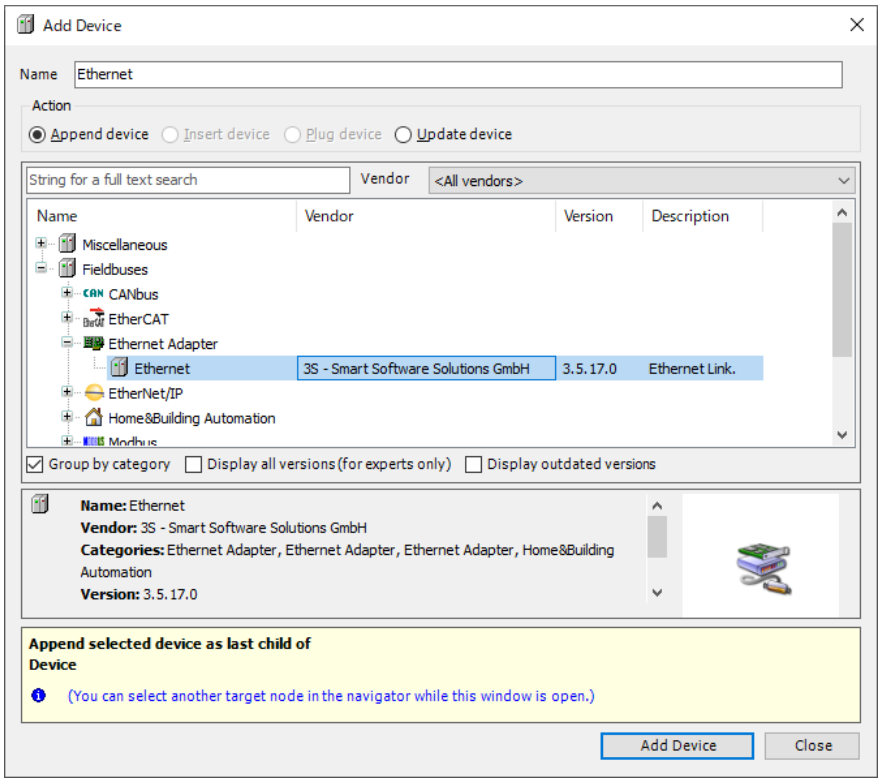


Fig 6.27 Add Device window

3. Add "EtherNet/IP Scanner" with "Ethernet" selected.

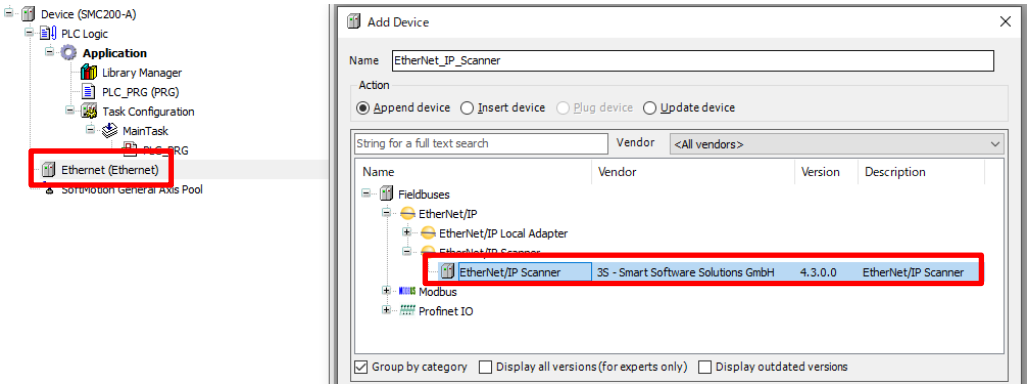


Fig 6.28 Add Scanner

6.2.4.2. Scanner settings

The setting screen can be opened by double-clicking "EtherNet_IP_Scanner" in the device tree.

If "Automatically-reestablish Connections" is checked, communication is automatically reestablished when communication with the adapter at the connection destination is broken.

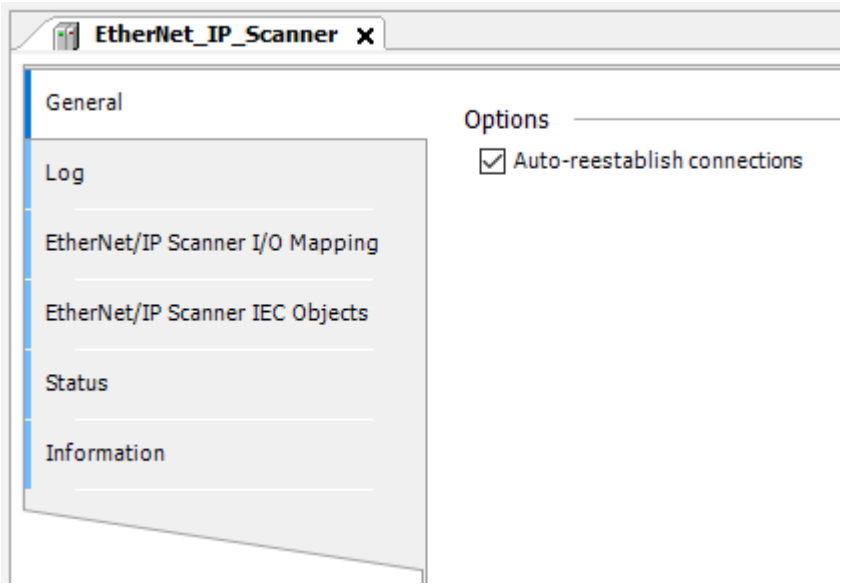


Fig 6.29 Scanner setting screen

6.2.4.3. Add remote adapter

The following is the procedure for adding a remote adapter.

1. Right-click “EthernNet_IP_Scanner” and select “Add Device…”.

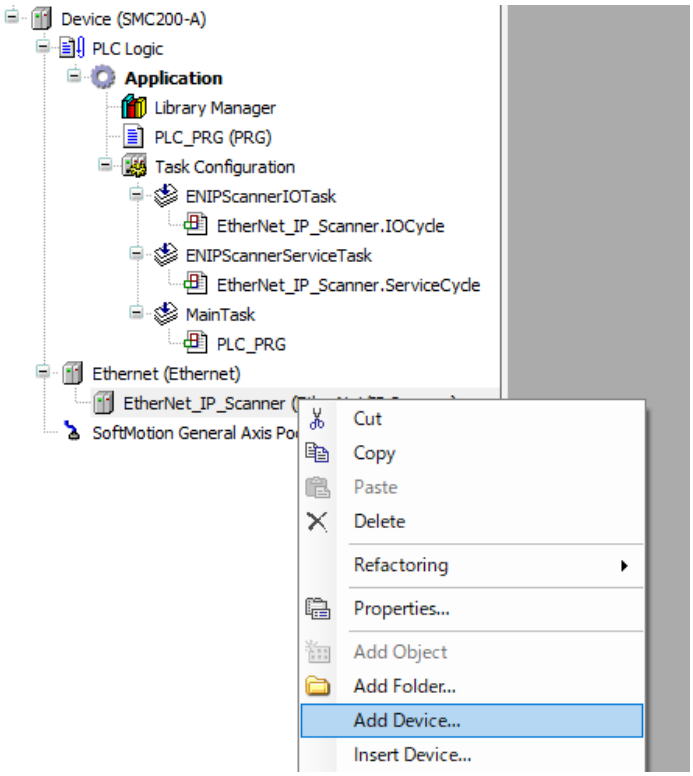


Fig 6.30 Add Device

2. The add device window will open. Add "Generic EtherNet/IP device".

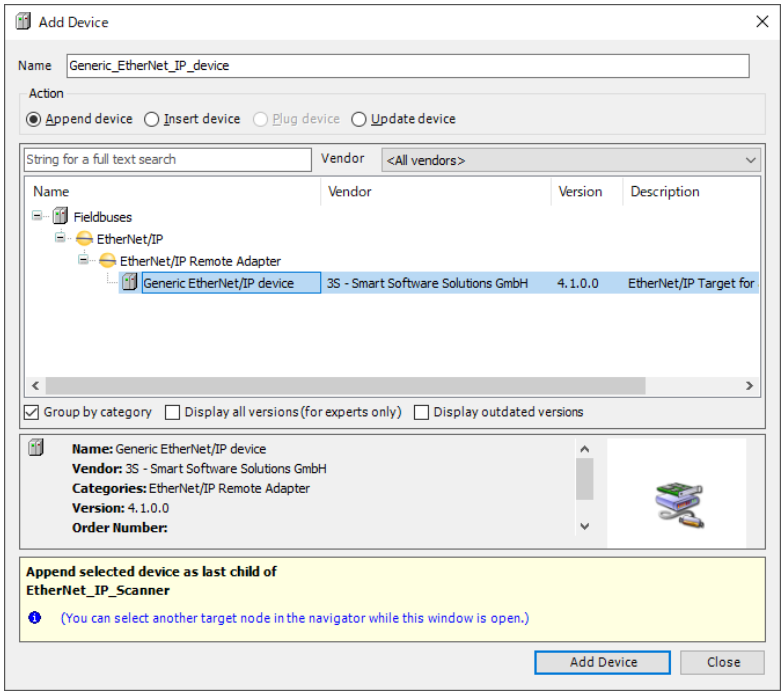


Fig 6.31 Add Device window

6.2.4.4. Remote Adapter Configuration

The setting screen of the remote adapter is displayed by double-clicking "Generic EtherNet/IP device" in the device tree. The following items can be set on the remote adapter setting screen.

【General】

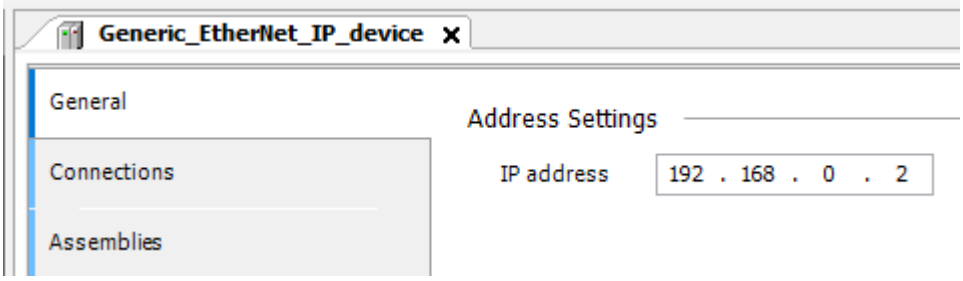


Fig 6.32 General tab

Item	Detail
IP Address	Set the IP address of the connection destination adapter.

【Connections】

The connection tab has the following setting items.
Click "Add Connection..." or double-click the name of an existing connection to display the Edit Connection window.

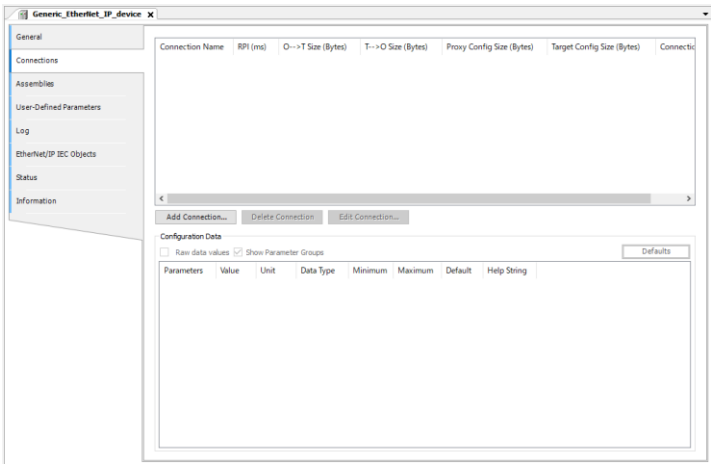


Fig 6.33 Connections tab

You can set the following items in the Edit connection window.

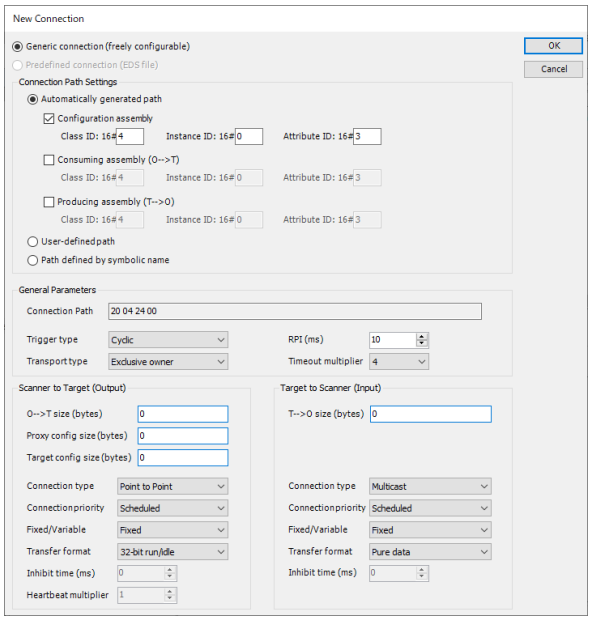


Fig 6.34 Edit connection window

*Basically, the contents of the EDS file are automatically reflected.

Connection Path Settings:

Item	Detail
Automatically generated path	A connection path is automatically generated from the values of the Constituting Assembly, Consuming Assembly, and Producing Assembly.
User-defined path	The connection path is specified manually in the corresponding input field.
Path defined by symbolic name	Paths are specified by symbolic names. Condition: Device must support symbolic connect path.

General Parameters:

Item	Detail
Connection Path	Set the IP address of the adapter to connect to.
Trigger type	**Cyclic**: Data exchange takes place cyclically at intervals set by the RPI. **Change of State**: Data is automatically exchanged when the scanner output or adapter input changes. **Application**: Not Implemented
Transport type	For details, check the specifications of CIP Volume 1 and Volume 2.
RPI (ms)	Request Packet Interval. The time interval (in ms) at which the sending application requests the target application to send data. This value must be a multiple of bus cycle tasks.
Timeout multiplier	If a device fails, there is a time delay (RPI * timeout multiplier) before the device state switches to 'error'.

Scanner to Target (Output):

Item	Detail
O-->T size (bytes)	Amount of data from scanner to adapter
Proxy Config size (bytes)	Size of proxy configuration data
Target Config size (bytes)	Size of adapter configuration data
Connection type	**Multicast**: A network connection is established. Connection data can be received by multiple consumers. **Unicast**: A network connection is established. Connection data can only be received by one consumer **Null**: No network connection established.
Connection priority	Conflicts can occur when two scanners use different priorities for an adapter. Adjusting the connection priority solves this problem.
Fixed / Variable	For details, check the specifications of CIP Volume 1 and Volume 2.
Transfer format	Not compatible
Inhibit time (ms)	Not compatible
Heartbeat multiplier	Not compatible

Target to Scanner (Input):

Item	Detail
T-->O size (bytes)	Amount of data from scanner to adapter
Connection type	**Multicast**: A network connection is established. Connection data can be received by multiple consumers. **Unicast**: A network connection is established. Connection data can only be received by one consumer **Null**: No network connection established.
Connection priority	Conflicts can occur when two scanners use different priorities for an adapter. Adjusting the connection priority solves this problem.
Fixed / Variable	For details, check the specifications of CIP Volume 1 and Volume 2.
Transfer format	Not compatible
Inhibit time (ms)	Not compatible

6.2.5. Explicit message communication function block

EtherNet / IP has two communication functions. There are Implicit message communication, which communicates at a fixed cycle, and Explicit message communication, which communicates at an arbitrary timing.

Implicit message communication is a function that performs data communication in the communication cycle set by RPI (Requested Packet Interval). Communication settings can be made on the "Connection" tab in "6.2.4.4Remote Adapter Configuration".

Explicit message communication is a function that sends and receives data to and from a specified node at any time. Communication can be performed using the Explicit message communication function block.

This function block supports only Explicit message communication from the scanner. Therefore, set the device name of the destination node in the input variable itfEtherNetIPDevice.

6.2.5.1. Apply_Attributes

This function block is used for calling the “Apply_Attributes” service of a specific instance of a CIP object.

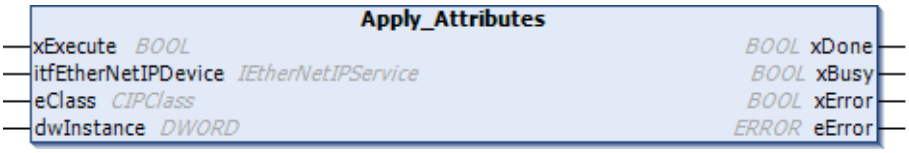


Fig 6.35 Apply_Attributes

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.2. NOP

This function block is used for calling the NOP service of a specific instance of a CIP object.

Fig 6.36 NOP

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.3. Reset

This function block is used for calling the reset service of a specific instance of a CIP object.

Fig 6.37 Reset

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.4. Start

This function block is used for calling the “Start” service of a specific instance of a CIP object.

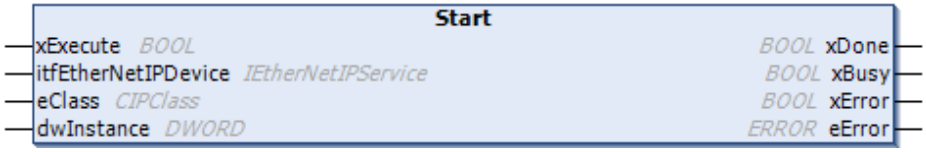


Fig 6.38 Start

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.5. Stop

This function block is used for calling the “Stop” service of a specific instance of a CIP object.

Fig 6.39 Stop

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.6. Get_Attributes_All

This function block is used for querying the attribute of a specific instance of a CIP object.

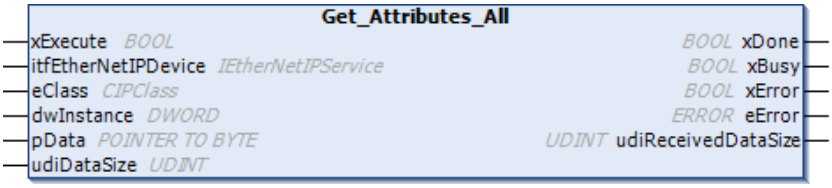


Fig 6.40 Get_Attributes_All

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
pData	POINTER TO BYTE	Data buffer
udiDataSize	UDINT	Size of buffer
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification
udiReceivedDataSize	UDINT	Size of the received data

6.2.5.7. Get_Attribute_Single

Use this function block for querying the attribute of a specific instance of a CIP object.

Fig 6.41 Get_Attributes_ Single

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
wAttribute	WORD	Attribute the services is addressed to. Leave 0 if this service does not address an attribute.
pData	POINTER TO BYTE	Data buffer
udiDataSize	UDINT	Size of buffer
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification
udiReceivedDataSize	UDINT	Size of the received data

6.2.5.8. Set_Attributes_All

This function blocks is used for setting the attribute of a specific instance of a CIP object.

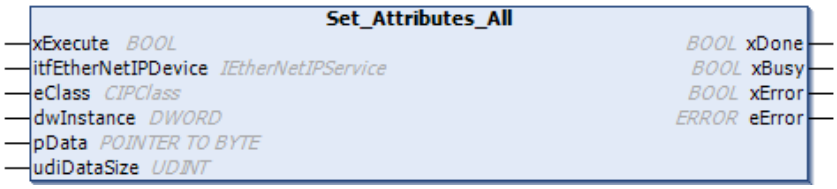


Fig 6.42 Set_Attributes_All

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
pData	POINTER TO BYTE	Error identification
udiDataSize	UDINT	Size of the received data
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.9. Set_Attribute_Single

This function block is used for setting the attribute of a specific instance of a CIP object.

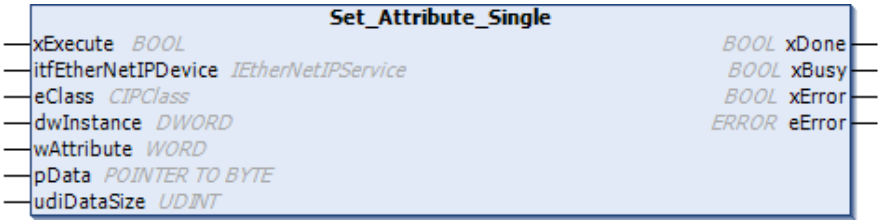


Fig 6.43 Set_Attributes_ Single

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
wAttribute	WORD	Attribute the services is addressed to. Leave 0 if this service does not address an attribute.
pData	POINTER TO BYTE	Data buffer
udiDataSize	UDINT	Size of buffer
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification

6.2.5.10. Generic_Service

This function block performs a generic service at an EtherNet/IP Adapter.

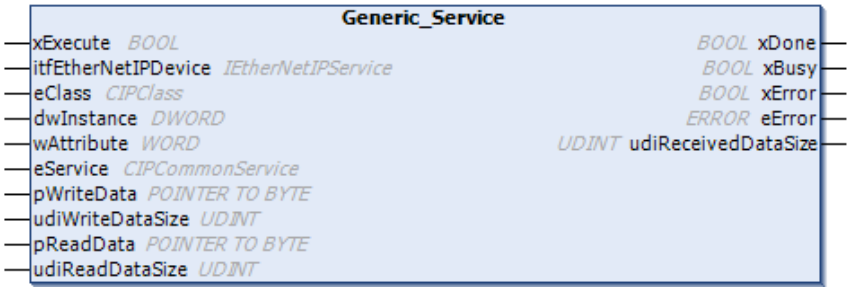


Fig 6.44 Generic_Service

VAR_INPUT		
xExecute	BOOL	Rising edge: Starts the execution of the FB.
itfEtherNetIPDevice	IEtherNetIPService	EtherNet/IP Device which implements the EtherNet/IP Services interface
eClass	CIPClass	Class which shall perform the service
dwInstance	DWORD	Instance which shall perform the service (0: Class level, 1..x: Instance level)
wAttribute	WORD	Attribute the services is addressed to. Leave 0 if this service does not address an attribute.
pWriteData	POINTER TO BYTE	Data to write to the EtherNet/IP Adapter. Leave 0 if no data should be sent to the EtherNet/IP Adapter.
udiWriteDataSize	UDINT	Size of data to write to the EtherNet/IP Adapter. Leave 0 if no data should be sent to the EtherNet/IP Adapter.
pReadData	POINTER TO BYTE	Data expected to receive from the EtherNet/IP Adapter. Leave 0 if no data is expected to be received from the EtherNet/IP Adapter
udiReadDataSize	UDINT	Size of data expected to receive from the EtherNet/IP Adapter. Leave 0 if no data is expected to be received from the EtherNet/IP Adapter.
VAR_OUTPUT		
xDone	BOOL	Function block execution complete
xBusy	BOOL	The FB is not finished
xError	BOOL	Signals that an error has occurred within the function block
eError	ERROR	Error identification
udiReceivedDataSize	UDINT	Size of the received data

For details on each service, refer to the specifications of the EtherNet/IP adapter to be connected.

6.3. OPC UA

This S200 has OPC UA server function. OPC UA is a data exchange standard for safe and reliable industrial communication that enables data exchange between multi-vendor products and across different operating systems.

The OPC UA setting is made with the symbol configuration object. The procedure for using OPC UA communication is described below.

1. Create symbol configuration object

   Right-click on "Application" and select "Add Object" → "Symbol Configuration" to add the symbol configuration object. Support for object name and OPC UA function can be set when adding, please enable support and set arbitrary name.

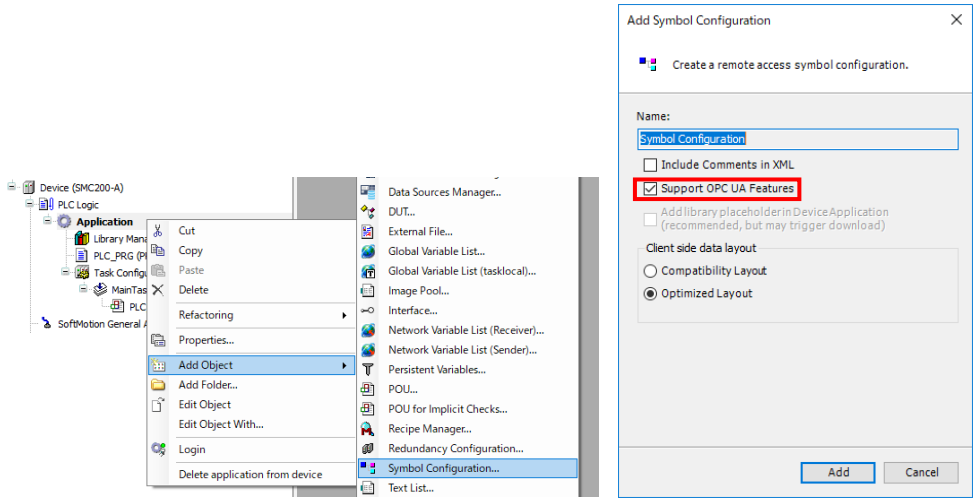


Fig.6.45 Symbol Configuration Object

2. Double clicking on the created symbol configuration object opens the following symbol configuration editor.

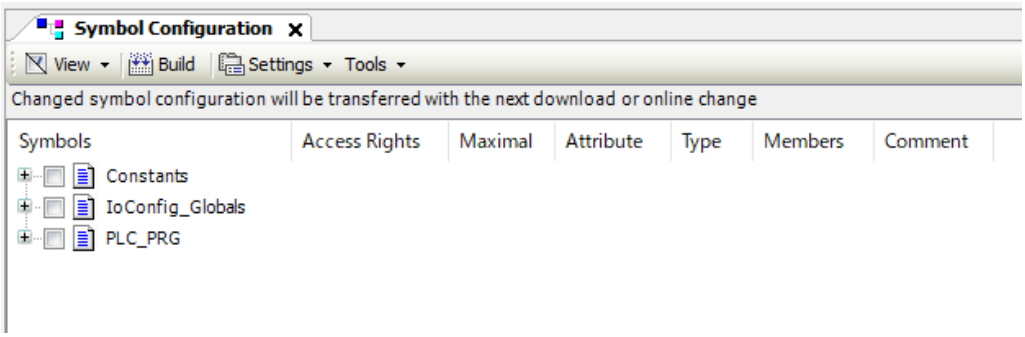


Fig.6.46 symbol configuration editor

3. Variable setting for remote access

   The symbol configuration editor displays a list of variables included in the application. Use the check box to the left of the variable name to enable / disable remote access.

   If you enable remote access, you can change the access rights for the symbol by clicking the symbol in the access right column.

   The list of access rights symbols is shown below.

Symbol	Detail
	Read only
	Write only
	Readable / Writable

   This completes the OPC UA setting. By downloading the project, you can access the specified variables from the OPC UA client.

When connecting from the OPC UA client, you can connect by using the URL or IP address and port number output in the log.

**********	CmpOPCUAServer
All available networkadapters are used.	CmpOPCUAServer
Loopbackadapter activated.	CmpOPCUAServer
URL: opc.tcp://SMC200:4840	CmpOPCUAServer
Hostname: SMC200, Port: 4840	CmpOPCUAServer
OPC UA Server Started:	CmpOPCUAServer
**********	CmpOPCUAServer

6.4. File sharing service

The S200 implements an FTP server and a Samba server as file sharing services. By enabling each server from the Web application, you can share files in the user area.

By enabling the file sharing service, you can perform the following operations on a PC where development environment is not installed.

- Confirmation/acquisition of files generated in the PLC application.
- Confirmation and acquisition of PLC application logs.
- Get status report.
- Operation of connected media (USB memory, microSD).

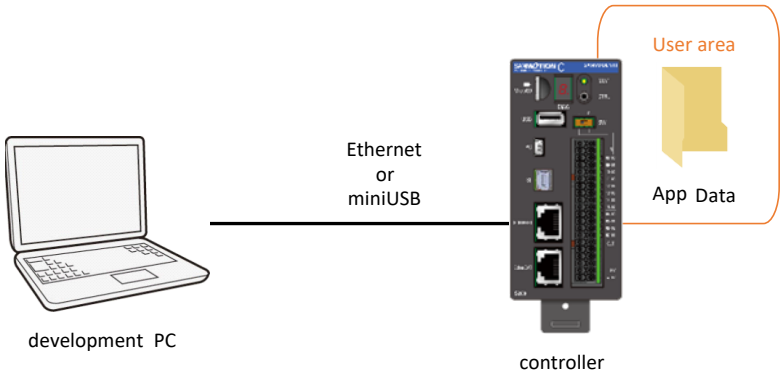


Fig.6.47 File sharing

6.4.1. Enable server from web application

You can control each server from the "File sharing" tab of the web application.

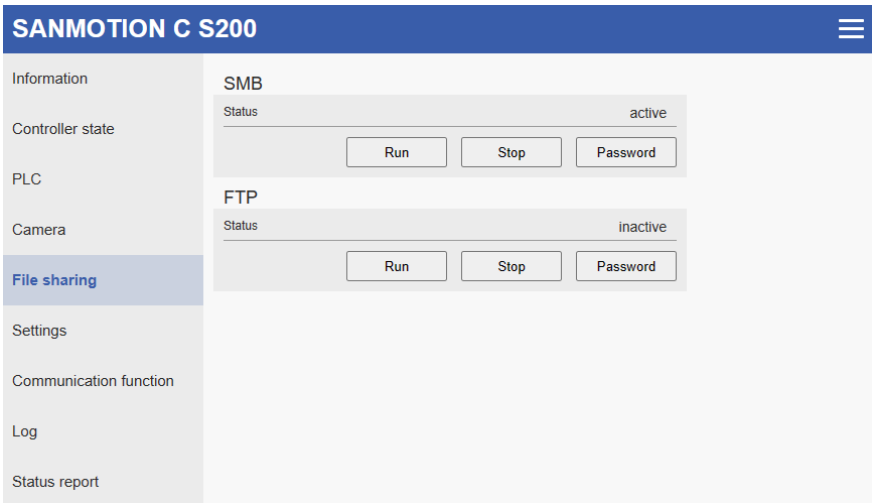


Fig.6.48 Enable server from web application

Item	Detail
Status	Show the current service running state
RUN button	Start server
STOP button	Stop server
Password button	Show server password change window

The status of each server at factory shipment is as follows.

FTP server : inactivate

Samba server : activate

Usernames and passwords are sent in plain text with FTP. We recommend disabling the FTP server if OT security is important.

6.4.2. Directory structure of user area

The structure of the user area is shown below.

Directory	File path in PLC application	Detail
/	-	User area top directory
└sancontrol	./	Default path for PLC application
└ PlcLogic	./PlcLogic	Directory where PLC application is stored
└data	$DATA$	Data storage area (20GB)
└report	$REPORT$	Directory for storing status reports
└ tmp	-	Volatile directory
└ media	$MEDIA$	Media links
└ usb_p_	$MEDIA$/usb_p_	Link destination of USB memory
└ microsd_p_	$MEDIA$/microsd_p_	Link destination of microSD
└ log	-	Directory containing CODESYS runtime logs
└ image	-	Storage destination for still images saved by camera control

6.4.3. Connection method

6.4.3.1. FTP

You can connect to the FTP server using the command prompt that is standard installed in Windows. When connecting, you need to enter the connection destination (host name or IP address), user name, and password in the red frame in the figure below.

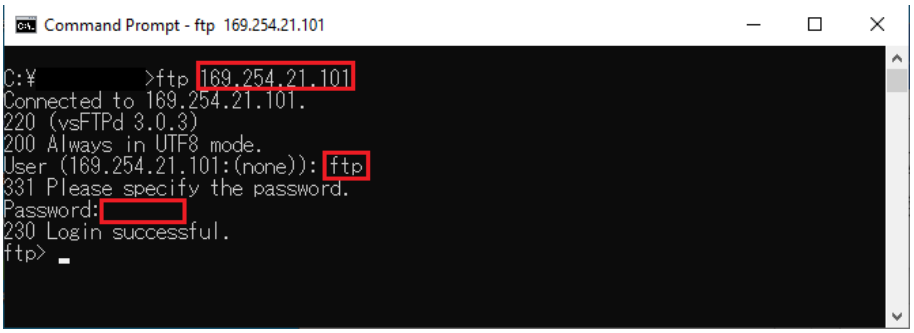


Fig.6.49 FTP server connection

User name	Password
ftp	ftp (default value)

6.4.3.2. Samba

You can connect to the Samba server from File Explorer. You can access it by entering “\\<hostname>" or IP address in the explorer path setting field.

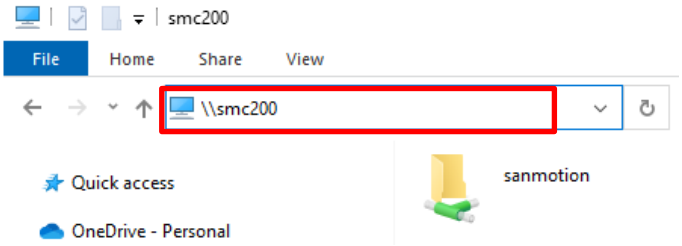


Fig.6.50 Samba server connection

To access the shared directory, you need to enter the following user ID and password.

Fig.6.51 Samba server login screen

User ID	Password
sanmotion	sanmotion (default value)

6.5. Wireless communication

Wireless functionality can be added by connecting a wireless adapter 3A (model number: SMC-USBW-01) to the USB port of the S200. For details, please refer to "M0020996 Wireless Adapter 3A Instruction Manual (Combination with S200 Series)".

WARNING!

- Do not use this function when someone with a cardiac pacemaker is nearby.
- Wireless is greatly affected by the surrounding radio wave environment, such as noise and crosstalk between users using the same frequency band, so communication may become unstable. Therefore, use the wireless communication function for purposes such as monitoring or file operations unrelated to motion control.

CAUTION!

- This function uses radio waves in the 2.4 GHz frequency band. Radio wave interference may occur when using this product near the following devices or radio stations.
  ・Industrial/scientific/medical equipment (microwave ovens, wireless LAN equipment, security equipment, cardiac pacemakers, etc.)
  ・Radio stations that do not require a license (specific power-saving radio stations)
  ・Radio stations requiring a license (on-premises radio stations for mobile identification used in factory production lines, amateur radio stations)
- Do not use near a microwave oven, in a place where static electricity or radio interference occurs, or in a room shut off by a metal door. Radio waves may not reach depending on the usage environment.

7. Control programming

7.1. I/O control programming

Create an I / O control program. Use "PLC standard project" as a template.

7.1.1. I/O assignment

Assign variable names to the I/O mapping table. You can use this variable name in subsequent programming. Follow the procedure below to make the setting.

1. Double-click "Device (SMC200-A)" and select "Device I/O mapping".

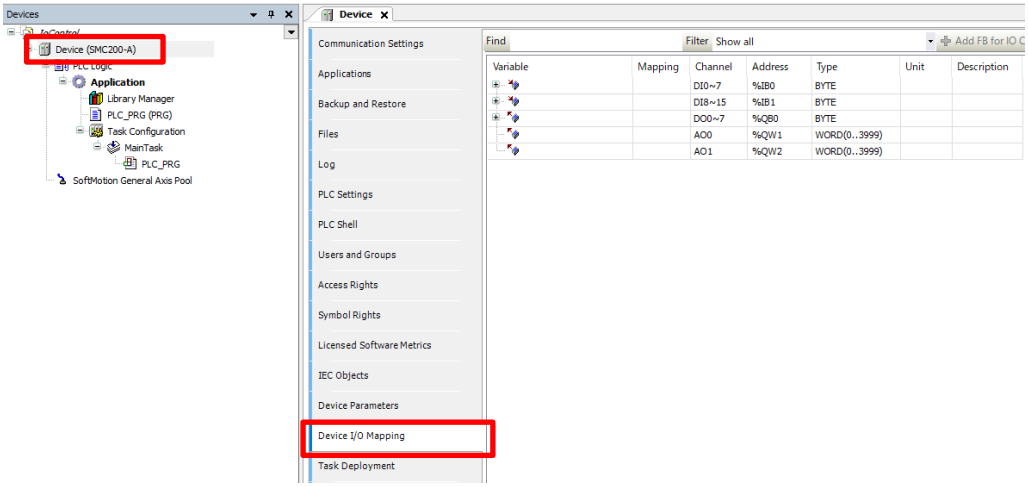


Fig.7.1 Device I/O mapping screen

2. Enter the following variable name in the variable field.

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI0~7	%IB0	BYTE		
DI0		Bit0	%IX0.0	BOOL		
DI1		Bit1	%IX0.1	BOOL		
DI2		Bit2	%IX0.2	BOOL		
DI3		Bit3	%IX0.3	BOOL		
DI4		Bit4	%IX0.4	BOOL		
DI5		Bit5	%IX0.5	BOOL		
DI6		Bit6	%IX0.6	BOOL		
DI7		Bit7	%IX0.7	BOOL		
		DI8~15	%IB1	BYTE		
DI8		Bit0	%IX1.0	BOOL		
DI9		Bit1	%IX1.1	BOOL		
DI10		Bit2	%IX1.2	BOOL		
DI11		Bit3	%IX1.3	BOOL		
DI12		Bit4	%IX1.4	BOOL		
DI13		Bit5	%IX1.5	BOOL		
DI14		Bit6	%IX1.6	BOOL		
DI15		Bit7	%IX1.7	BOOL		
		DO0~7	%QB0	BYTE		
DO0		Bit0	%QX0.0	BOOL		
DO1		Bit1	%QX0.1	BOOL		
DO2		Bit2	%QX0.2	BOOL		
DO3		Bit3	%QX0.3	BOOL		
DO4		Bit4	%QX0.4	BOOL		
DO5		Bit5	%QX0.5	BOOL		
DO6		Bit6	%QX0.6	BOOL		
DO7		Bit7	%QX0.7	BOOL		
		AO0	%QW1	WORD(0..3999)		
		AO1	%QW2	WORD(0..3999)		

Fig.7.2 Setting variable names to Device I/O

7.1.2. Creation of I/O control program

Create a simple program that uses the assigned variable name. Please follow the procedure below. In addition, in order to check the operation after program creation, it is necessary to turn digital input on and off. Refer to "Hardware Manual" for wiring method.

1. Please open "PLC_PRG". And describe the following in the mounting section.

   【Implementation section】

```
DO0 := DI0 AND DI1;
DO1 := DI2 AND DI3;
DO2 := DI4 AND DI5;
DO3 := DI6 AND DI7;
DO4 := DI8 AND DI9;
DO5 := DI10 AND DI11;
DO6 := DI12 AND DI13;
DO7 := DI14 AND DI15;
```

2. After creating the program, click  to log in.

3. After login it will be like the following screen. Please click the operation and execute the program.

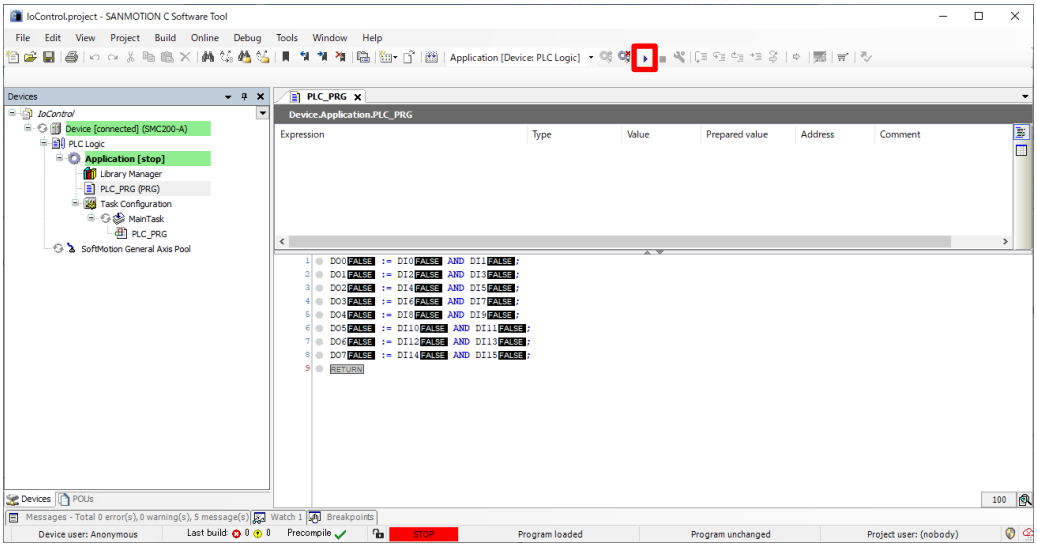


Fig.7.3 Screen after login

4. When the program is normally executed, the following screen will appear.

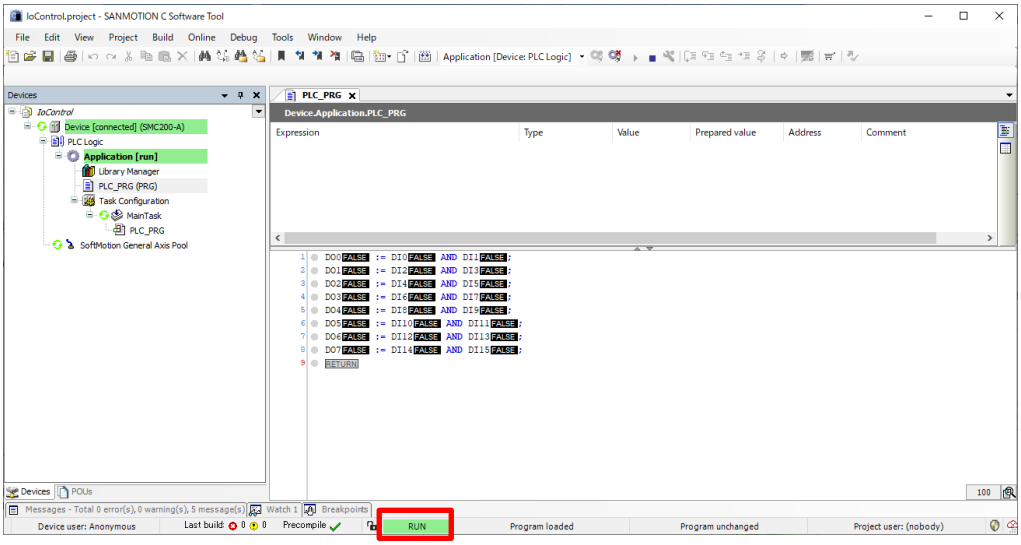


Fig.7.4 Screen after driving

5. Check the operation of the program. For the input 2 points of the program created this time, 1 output point is set to TRUE. Please input the signal to the digital input and confirm.

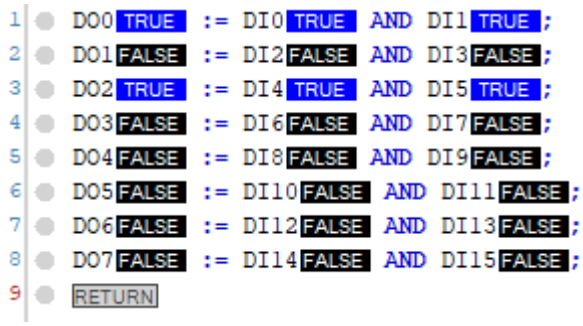


Fig.7.5 Screen during program execution

7.2. Manual drive program

Please open the "Motion Standard project" to create a program to control the servo amplifier connected to the EtherCAT port of SMC200-A.

This sample program uses the automatic variable declaration function.

For details of the function, please refer to "4.8.4 Input Assistant function".

7.2.1. Sample program summary

Create a program to perform JOG operation.

In the following description, we will assume the case where it is combined with SMC200-A and Sanyo Denko servo amplifier RS3 series (thereafter RS3). Please connect RS3 and S200 with reference to the following figure.

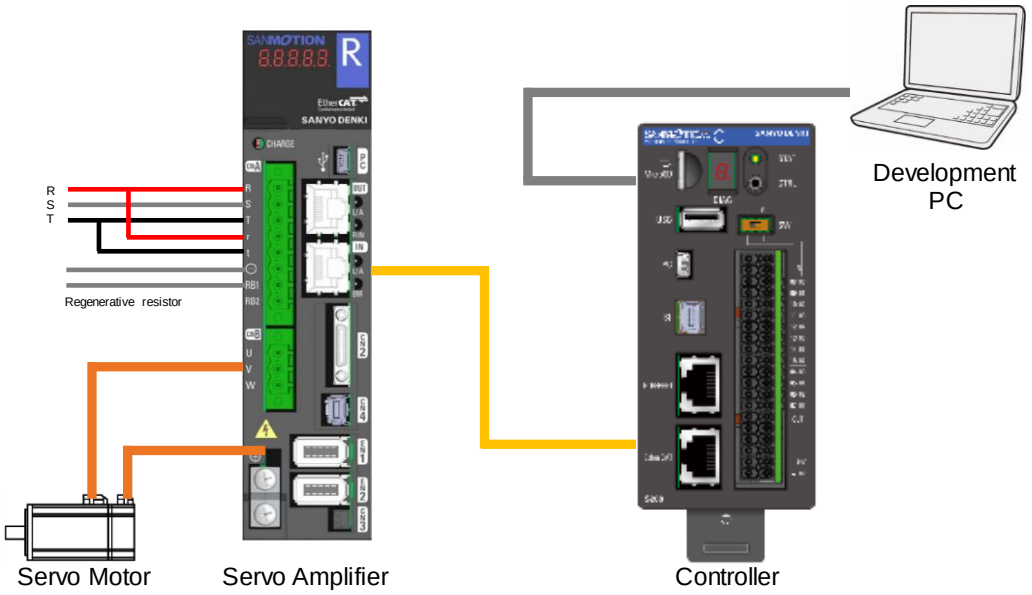


Fig 7.6 Connection diagram with RS3

7.2.2. Configuration

7.2.2.1. Add slave

Add and configure slaves. There are two ways to add slaves. It is a method to add manually and a method to search for and add slaves connected to the master.

【Manual setting】

1. Right-click "EtherCAT_Master_SoftMotion (EtherCAT Master Soft Motion)" and click "Add Device ...".

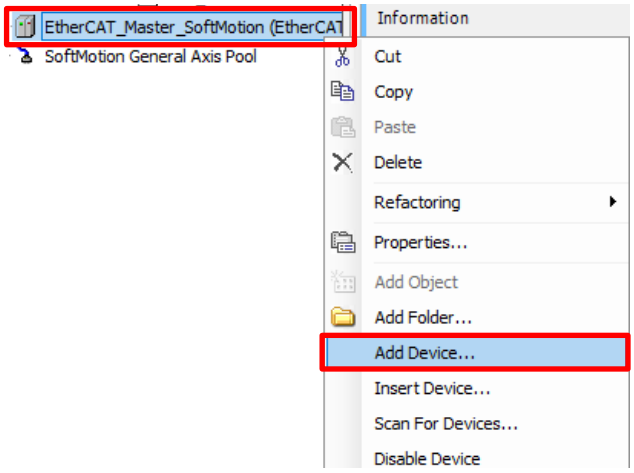


Fig.7.7 Add device

2. Select the slave to use and click "Add Device". Please close this window after adding the slave.

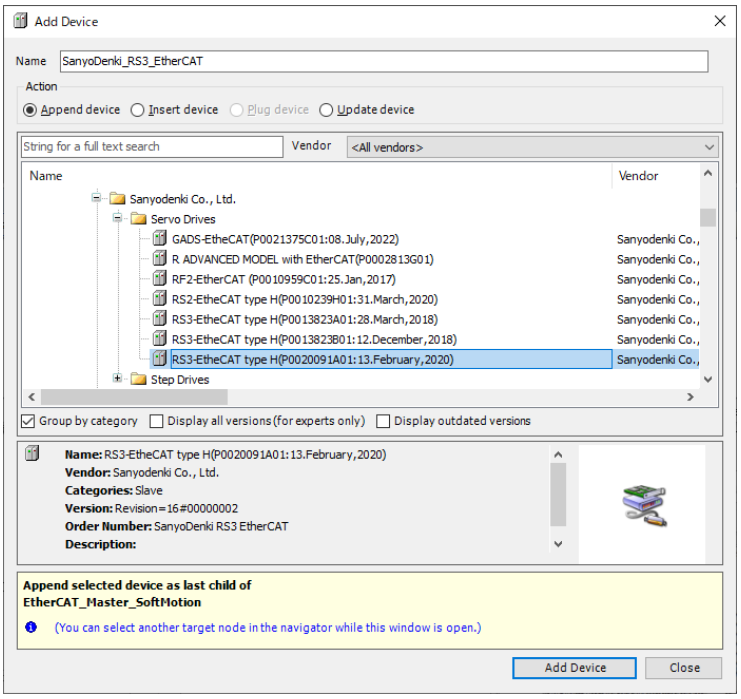


Fig.7.8 Add Device Window

【Search setting】

Search and add the slave connected to the master. Please add according to the following procedure. For the S200 to be used for the first time, it is possible to search the slave by adding the EtherCAT master and logging in once and logging out. This is because a stack is not created unless an EtherCAT master is added.

1. Right-click "EtherCAT_Master_SoftMotion (EtherCAT Master Soft Motion)" and select "Search for devices ...".

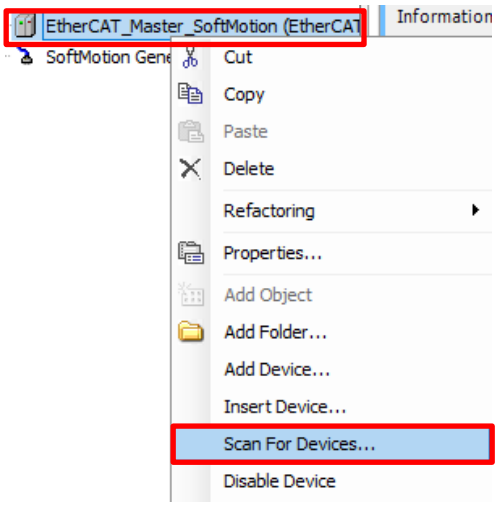


Fig.7.9 Scan for devices

2. Since the slave connected to the master is displayed, select the slave to be added and click "Copy to Project".

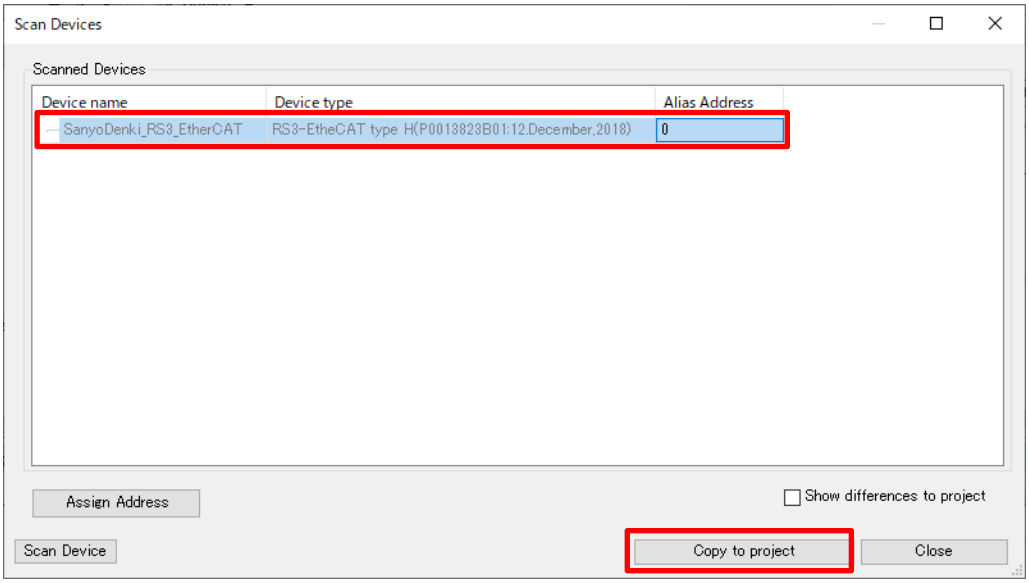


Fig.7.10 Scan Device window

7.2.2.2. Add axis

Add axes. Please add according to the following procedure.

1. Right-click on the added slave ("SanyoDenki_RS3_EtherCAT") and click "Add SoftMotion CiA 402 Axis".

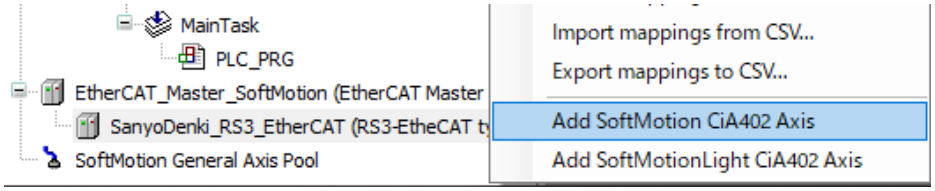


Fig.7.11 Add axis

2. Change the name of the axis. The name of the axis can be changed to arbitrary name, and you can use this name in subsequent programming. Please select again the axis you want to change and click again. Please enter an arbitrary name and press the Enter key.

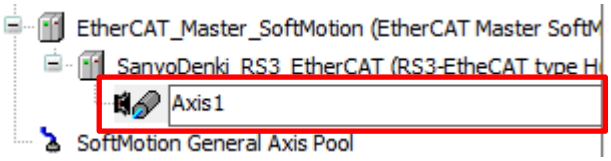


Fig.7.12 Change axis name

3. If you change the axis name, a window will be displayed asking if you want to adapt the object name change for all references in the project. Please click "Yes".

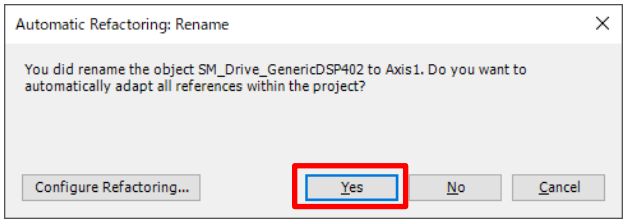


Fig.7.13 Automatic Refactoring: Rename window

4. A list of matched objects will be displayed, so click "OK".

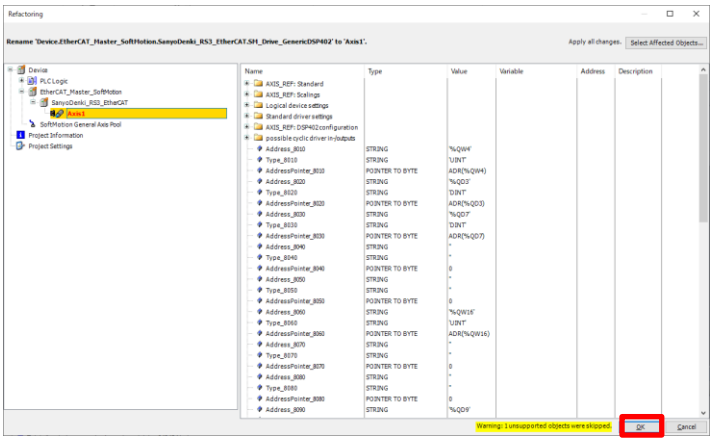


Fig.7.14 Refactor window

7.2.2.3. Axis settings

The following items can be set in the axis setting. The axis setting screen is displayed by double clicking on the added slave. This section explains the setting items on the "General" and "Scaling / Mapping" screens.

【General】

In the " General " tab you can set the parameters of the axis.

Set the axis type to "Modulo" this time.

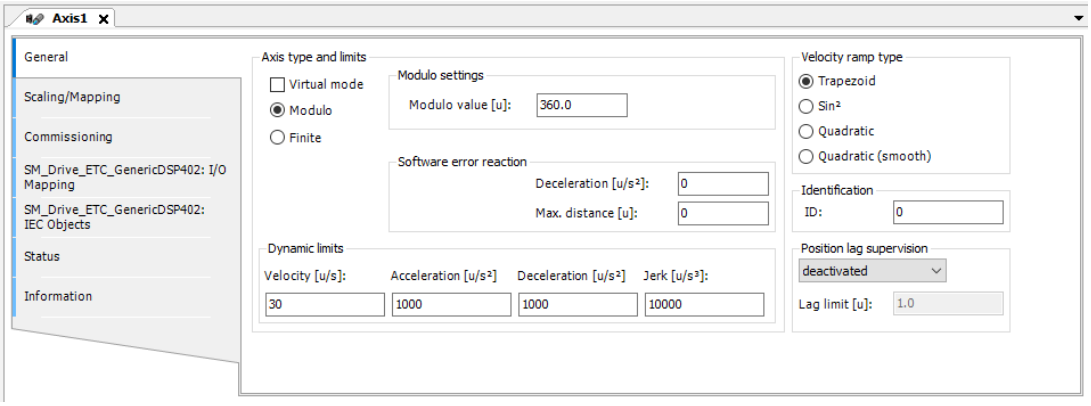


Fig.7.15 Axis setting screen (general)

Item	Detail
Virtual mode	The drive is replaced by a simulation that is similar to a virtual drive unit.
Modulo	The drive turns endlessly without limiting the traversing range
Finite	The drive has a fixed work area
Modulo settings	Sets the modulo maximum value. (Appears when the axis type is Modulo.)
Software limits	Position values are restricted by the lower limit Negative and an upper limit Positive. (Appears when the axis type is Finite)
Software error reaction	Deceleration value when reaching the limit switch.
Dynamic limits	It applies to CNC and robot control.
Velocity ramp type	Defines the velocity profile for motion-generating single-axis and master/slave modules. Trapezoid, Sin^2, Quadratic, Quadratic (smooth)
Identification	Integer identifier. Should be unique for each drive. For example, this identifier is used in the PLC log in order to identify the drive when an error occurs.
Position lag supervision	A drag error is detected when the difference between the set position and the compensated actual position exceeds the drag error limit.

【Scaling/Mapping】

On the "Scaling / Mapping" tab, you set the user unit system.

In this sample program, we assume an axis with an encoder resolution of 17 bits. At this time, set the “increments” in the axis parameter to "131072". Also, set the “unit in application“ to "360".

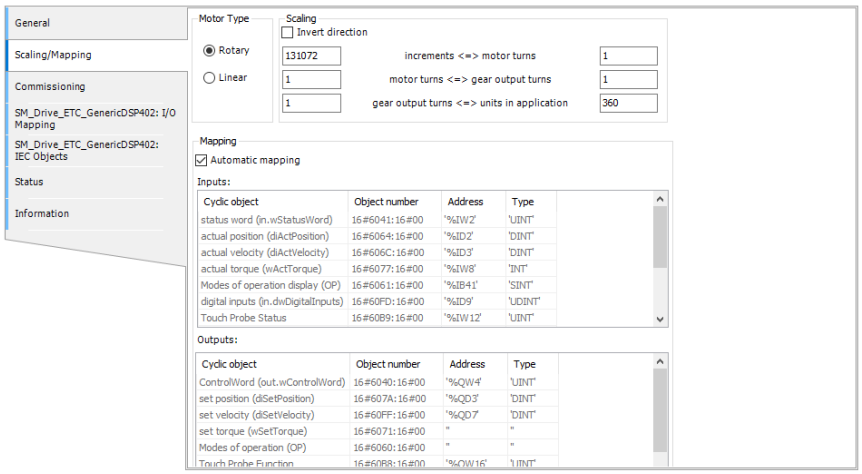


Fig.7.16 Axis setting screen (Scaling / Mapping)

Item	Detail
Motor Type	Set the type of connected motor. When the linear type is set, the only scaling setting item is "increments <=> units in application".
Invert direction	The direction of rotation is reversed. The motor receives the specified values with inversed signs.
increments<=> motor turns	Number of increments that correspond to a given number of motor rotations. If the number of pulses per revolution is 131072 (17 bits), set increment to 131072 and motor rotation to 1.
motor turns <=> gear output turns	Number of motor rotations that correspond to a given number of gear output rotations. If the reduction ratio is 1/10, please set the motor rotation to 10 and the gear output rotation to 1.
gear output turns <=> units in application	Number of gear output rotations that correspond to a unit in the application. For one gear revolution at 360 degrees, set the gear output rotation to 1 and the unit within the application to 360.
Automatic mapping	IEC parameters that affect the drive are automatically mapped to the corresponding inputs and outputs of the device.

CAUTION

- Unexpected behavior may occur if the parameter "Increments" is set differently from the device.

  Set the following values correctly. For details, refer to the instruction manual for each driver.

  Servo amplifier : Encoder resolution

  (If the scale setting is changed, resolution after scale conversion)

  PB Driver : Value set for object 0x6092:01 'feed'

If an object that is not defined in the object dictionary, such as a dummy object, is mapped to PDO, the automatic mapping function may not be associated correctly. In that case, do not use dummy objects or associate them manually.

7.2.2.4. The state diagram

The initial state of an axis is disabled. State transitions due to issued motion commands are shown by full arrows. Dashed arrows are used for state transitions that are caused by the system

Motion commands listed above the states transit the axis to the corresponding state. In the states DiscreteMotion, ContinuousMotion, and SynchronizedMotion these motion commands may also be issued when the axis is already in the according motion state.

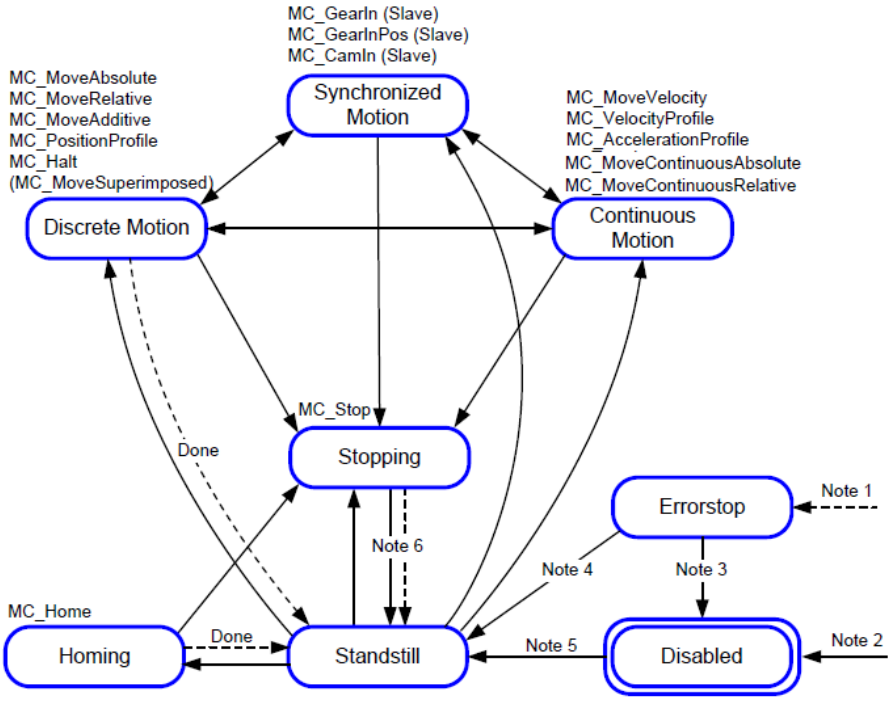


Fig.7.17 State diagram of an axis

Note.	Detail
1	From any state. An error in the axis has occured.
2	From any state. MC_Power.Enable = FALSE . There is no error in the axis.
3	MC_Reset and MC_Power.Status = FALSE
4	MC_Reset and MC_Power.Enable=TRUE, MC_Power.bRegulatorOn=TRUE, MC_Power.bDriveStart=TRUE and MC_Power.Status = TRUE
5	MC_Power.Enable=TRUE, MC_Power.bRegulatorOn=TRUE, MC_Power.bDriveStart=TRUE and MC_Power.Status = TRUE
6	MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

7.2.3. Sample program

Jog operation is performed using MC_StartupDrive. Follow the procedure below to perform jog operation. SMC_StartupDrive is a set of representative function blocks (FB) below. In addition, please refer to the help for explanation of each FB.

1. Select the program "Motion_PRG". Add FB. Click the mounting part and press the F2 key.

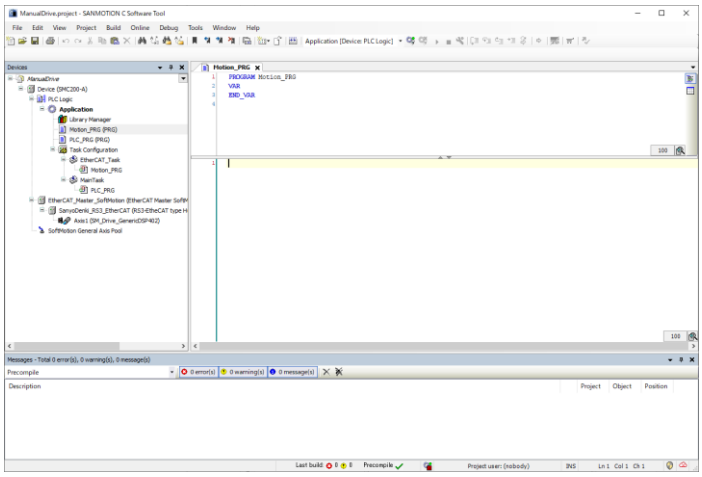


Fig.7.18 Program creation screen

2. The input assistant window will be displayed. Click "Function Blocks" and select "SMC_StartupDrive" in "SM3_Basic" and click OK.

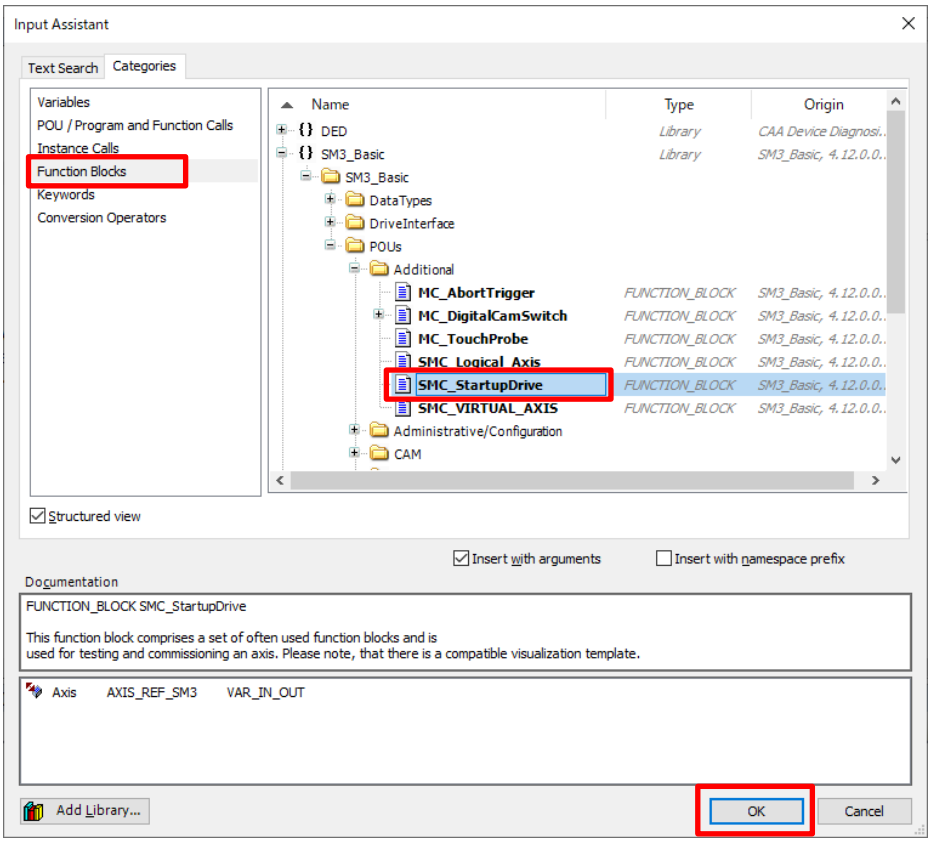


Fig.7.19 Input assistant window

3. Set the name of FB. Please enter "StartUp Drive" and click "OK".

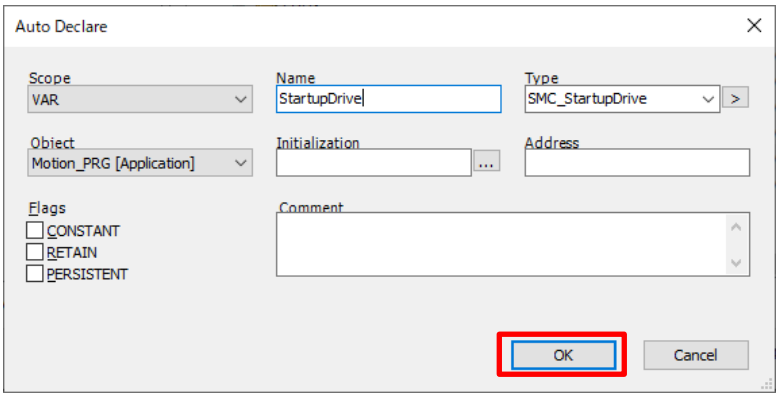


Fig.7.20 Automatic declaration window

4. Allocate the axis to be controlled to FB. Please set the axis to FB input.

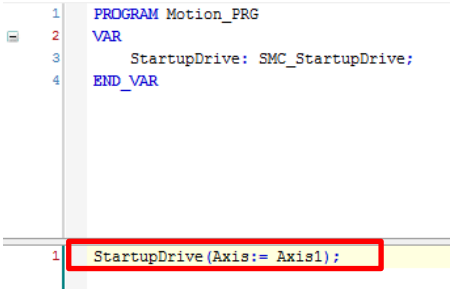


Fig.7.21 Assignment of axes to FB

5. After setting please click on  to login. After login it will be like the following screen. Please click the operation and execute the program.

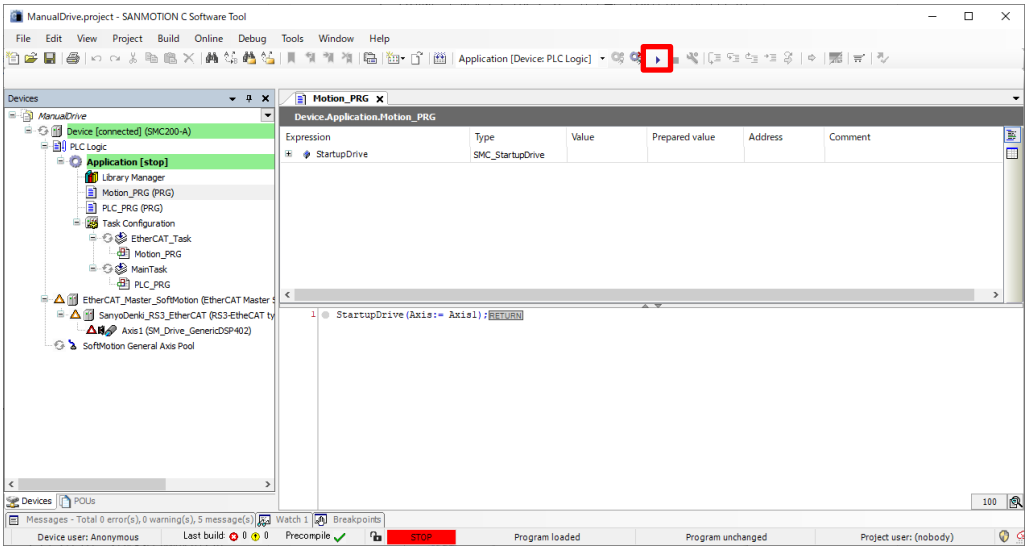


Fig.7.22 Screen after login

6. When the program is normally executed, the following screen appears.

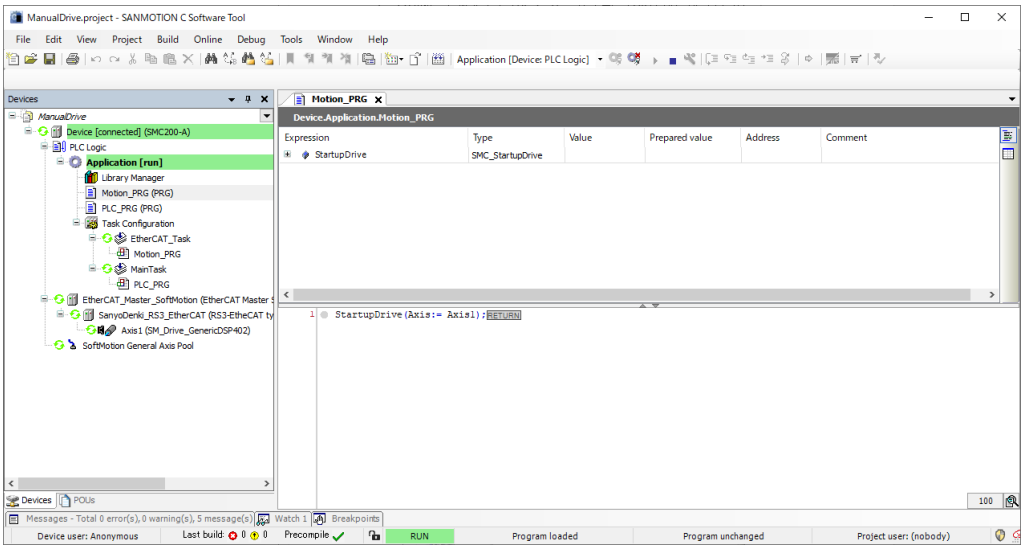


Fig.7.23 Screen after execution

7. Servo on. Please click on the "+" to the left of SMC_StartupDrive displayed in the declaration section and expand it. Next, expand MC_Power. The following screen will be displayed. For MC_Power, refer to “9.2.1.1 MC_Power”.

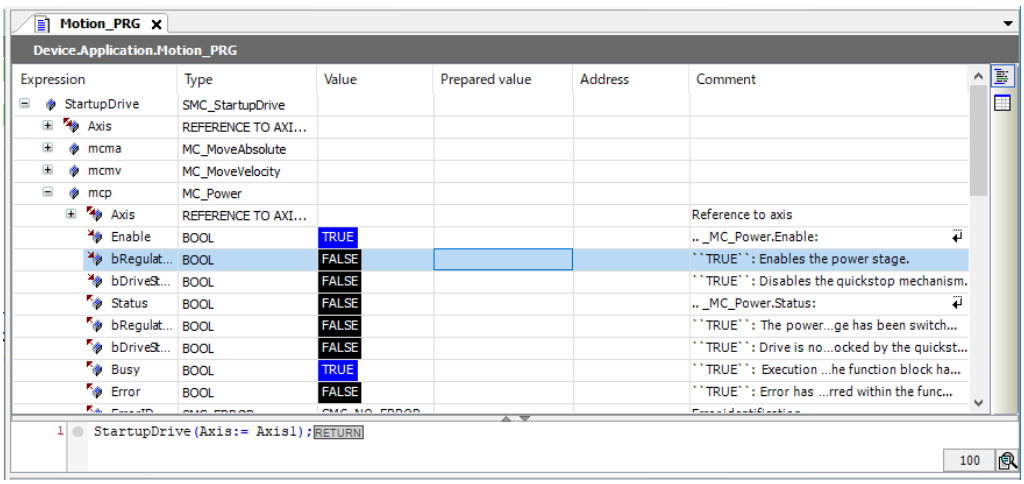


Fig.7.24 After declaration department's MC_Power expansion

8. Set the value. Click "Set value" column of "bRegulatorOn" and "bDriveStart" and set it to TRUE. After that, by pressing Ctrl + F7, the value is set to the FB input and the servo is turned on.

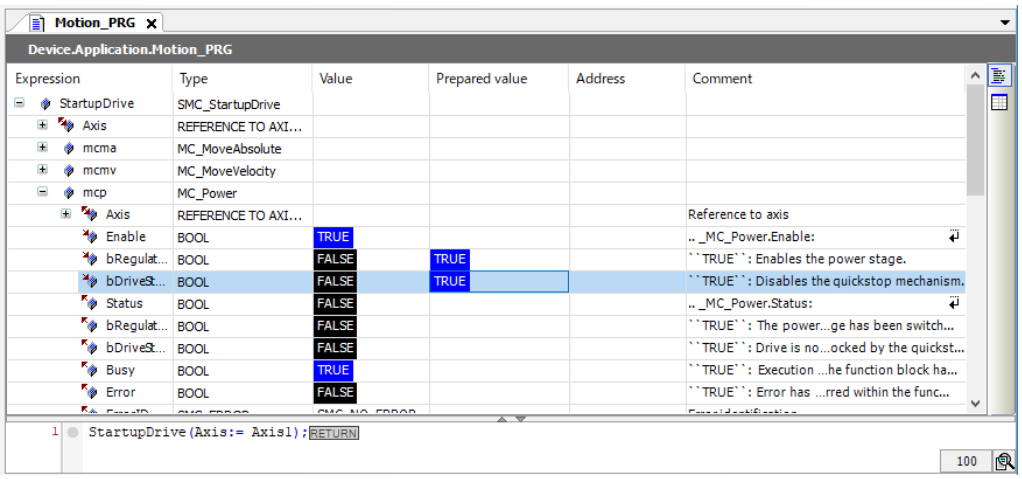


Fig.7.25 Enter value

9. When servo is turned on, the screen will look like the following. MC_Power.Status = TRUE when servo is on.

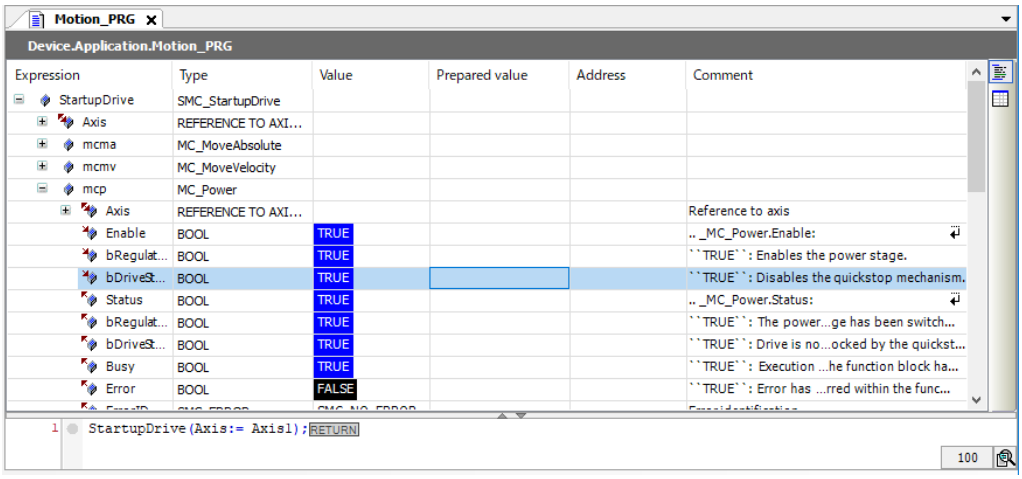


Fig.7.26 Enter value

10. Set the speed and acceleration / deceleration of the jog operation. Please expand MC_Jog. Enter the following values in the "Set value" column of "Velocity", "Acceleration", "Deceleration" and press Ctrl + F7. For MC_Jog please refer to "9.2.1.10 MC_Jog".

Device.Application.Motion_PRG

Expression	Type	Value	Prepared value	Address	Comment
mcj	MC_Jog				
Axis	REFERENCE TO AXI...				Reference to axis
JogForw...	BOOL	FALSE			``TRUE``: Axis is mo...with the specified d...
JogBack...	BOOL	FALSE			``TRUE``: Axis is mo...with the specified d...
Velocity	LREAL	1	10		Velocity in [u/s]
Accelerat...	LREAL	10	100		Acceleration in [u/s²]
Decelera...	LREAL	10	100		Deceleration in [u/s²]
Jerk	LREAL	0			Jerk in [u/s³]
Busy	BOOL	FALSE			| ``TRUE``: Function...k is in operation dur...
Comman...	BOOL	FALSE			| ``TRUE``: Executio... interrupted by anot...
Error	BOOL	FALSE			``TRUE``: Error has ...rred while ``JogFor...
ErrorId	SMC_ERROR	SMC_NO_ERROR			Error identification
fbHalt	MC_Halt				
fbMoveV...	MC_MoveVelocity				

Fig.7.27 Parameter setting of MC_Jog

11. Perform jog operation. When MC_Jog.JogForward = TRUE, forward rotation is performed, and if MC_Jog.JogBackward = TRUE, reverse rotation will occur. Set MC_Jog.JogForward = FALSE and MC_Jog.JogBackward = FALSE to stop.

7.3. Manual drive program by visualization

Visualization to create the suitable user interface for your application. You link the visualization to the application variables and in this way they can animate and display data. When creating a visualization and an application, you use common functions, for example, as library and source code management or find/replace throughout the project. The visualization can be accessed also from general Web browser（Web Visualization）.

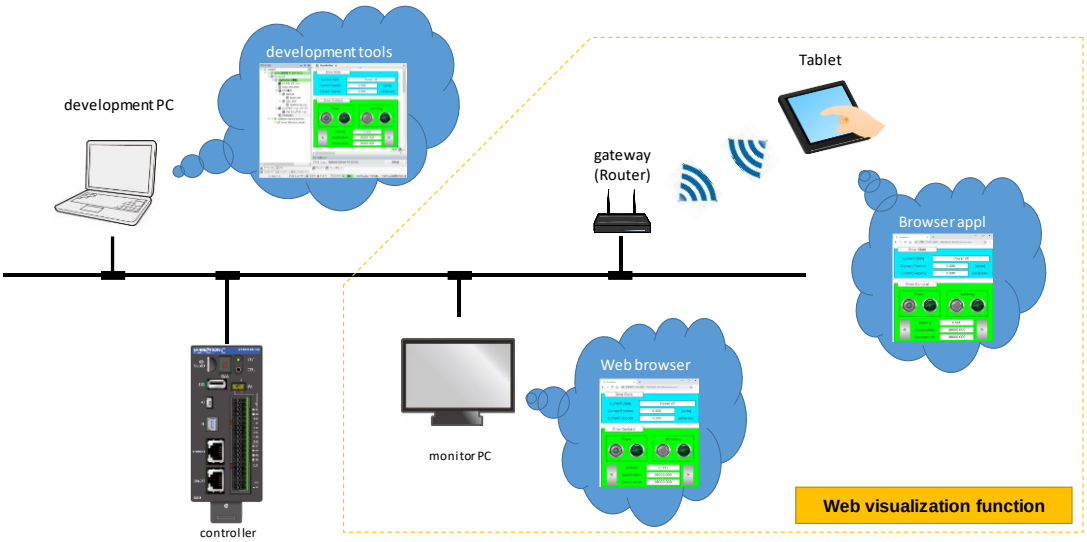


Fig.7.28 Visualization

7.3.1. Sample program summary

The procedure for creating a sample program that controls the single axis JOG operation project from the visualization is described below. Add the following information to "Motion_PRG".

The visualization configuration of the sample program is as follows.

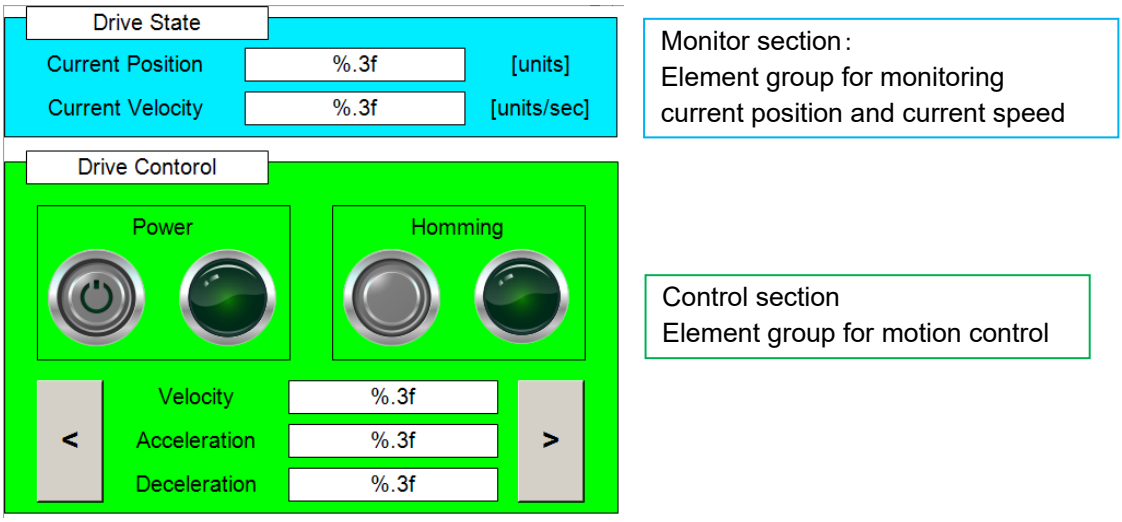


Fig 7.29 Sample visualization configuration

7.3.2. Configuration

Right-click on "SoftMotion General Axis Pool" and select "Add Device" → "virtual drives" → "SM_Drive_Virtual" to add axis objects. Since you can set the object name when adding, please set it as "Drive 1".

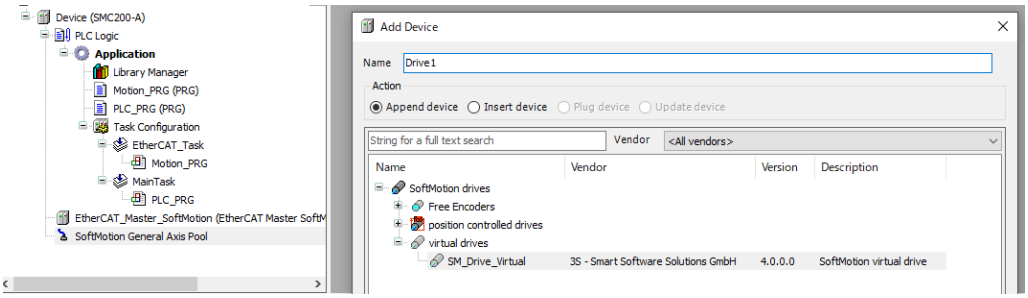


Fig.7.30 Add virtual axis

7.3.3. Sample program

Please add the following to "PLC_PRG". The program described here is the motion control execution unit, and control is done from the visualization.

【Declaration section】

```
PROGRAM PLC_PRG
VAR
        Axis1_SonFlag     :     BOOL;
        Axis1_Power       :     MC_Power;
        Axis1_Home        :     IoSanyoDevice.SanHome;
        Axis1_Jog         :     MC_Jog;
END_VAR
```

【Implementation section】

```
Axis1_Power(Axis:= Drive1, Enable:= TRUE,
                bRegulatorOn:= Axis1_SonFlag, bDriveStart:= Axis1_SonFlag);
Axis1_Home(Axis:= Drive1);
Axis1_Jog(Axis:= Drive1);
```

7.3.4. Creation of visualization screen

7.3.4.1. Add a visualization

The usage of visualization is described below.

1. Create a visualization object

   Right-click on "Application" and select "Add Object" → "Visualization" to add visualization object. You can set the object name when adding, so please set any name.

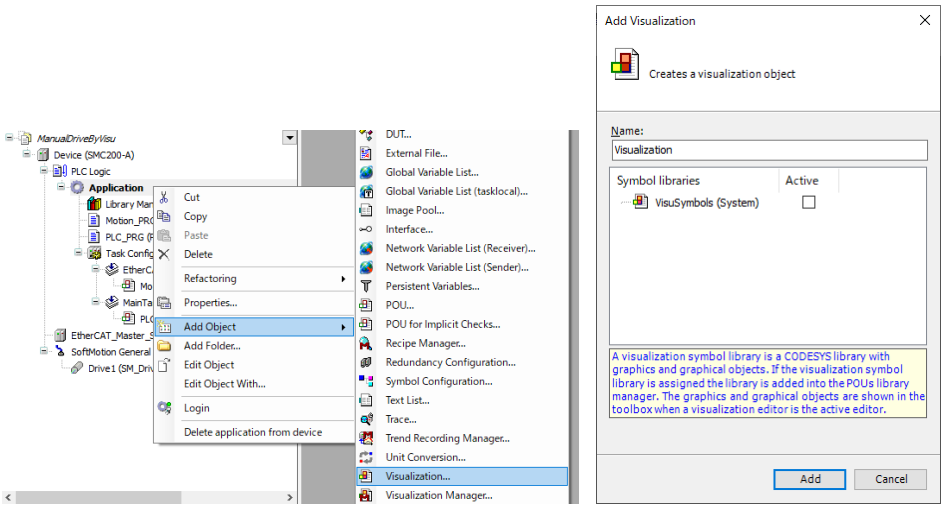


Fig.7.31 Visualization object

2. Double-click on the created visualization object, the following visualization editor opens.

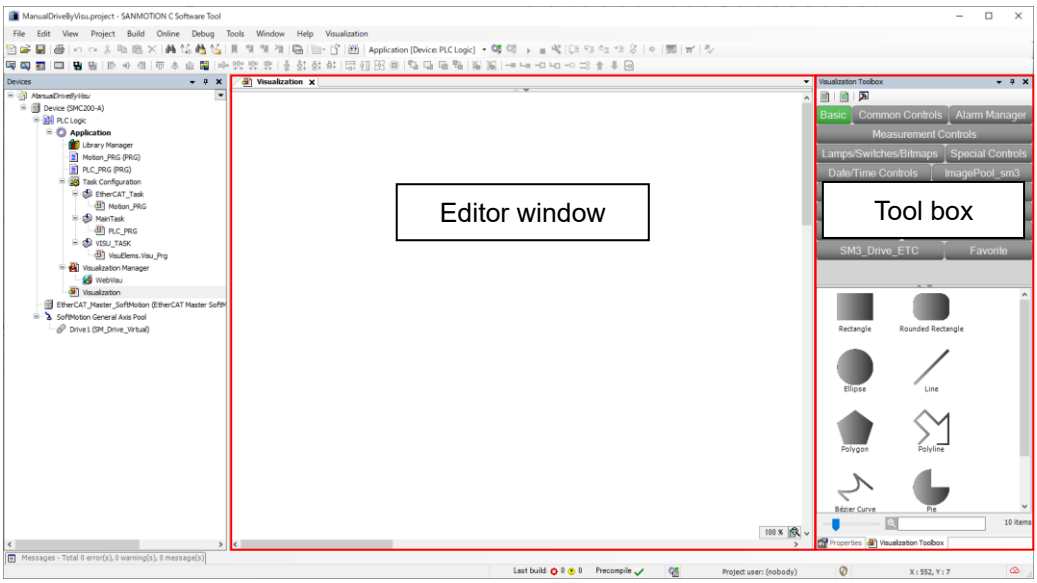


Fig.7.32 Visualization editor

3. Create a visualization screen by dragging Element (figure, button, etc.) in the tool box to the editor window.

   When you add an Element, the Element Properties window will be displayed in the Toolbox. In the Properties window, you can set the color and size of Element, the variables to assign, and so on.

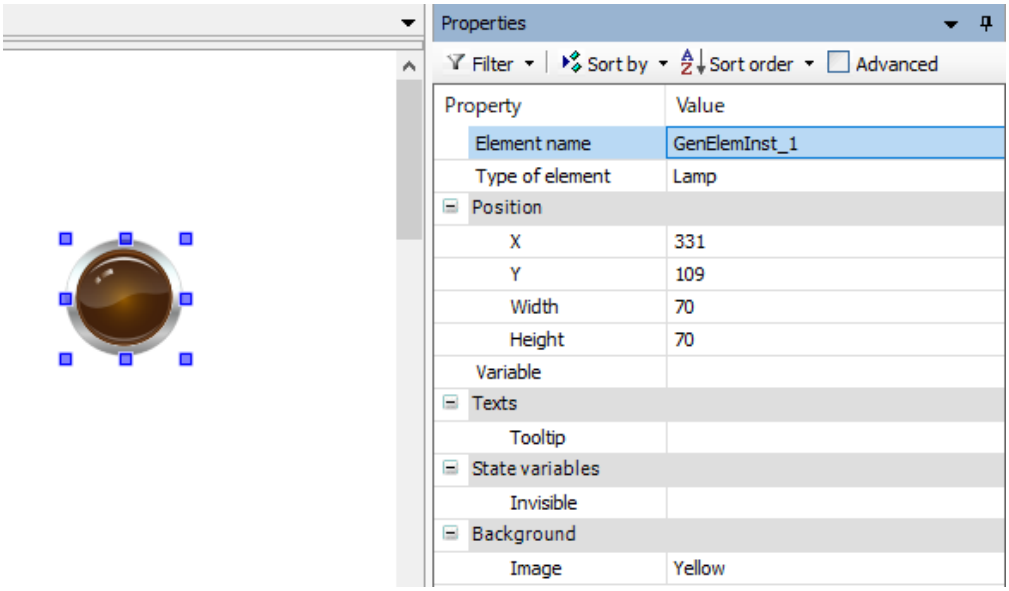


Fig.7.33 Property window

7.3.4.2. Creation of monitor section of visualization

The monitor consists of eight Elements.

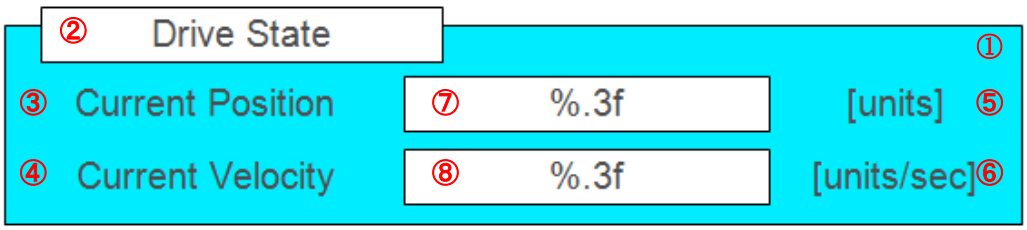


Fig.7.34 Element structure of monitor unit

No.	Detail
①	Background element
②	Title element
③	Position display label
④	Velocty display label
⑤	Position unit display label
⑥	Velocty unit display label
⑦	Current position monitor element
⑧	Current Velocty monitor element

The contents of each setting are described below.

【①Background Element】

Item	Detail
Element	Rectangle（Tag: Basic）
Colors—Normal state—Fill color	LightBlue

【②Title element】

Item	Detail
Element	Rectangle（Tag: Basic）
Texts—Text	Drive State

【③～⑥Each label】

Item	Detail
Element	Label（Tag: Common controls）
Texts—Text	See Figure 5.11
Text properties—Font	Size：15

【⑦Current position monitor element】

Item	Detail
Element	Rectangle（Tag: Basic）
Texts—Text	%.3f
Text properties—Font	Size：15
Text variables—Text variable	Drive1.fActPosition

【⑧Current Velocty monitor element】

Item	Detail
Element	Rectangle（Tag: Basic）
Texts—Text	%.3f
Text properties—Font	Size：15
Text variables—Text variable	Drive1.fActVelocity

A character string that is output in the visualization can include the placeholder % for a variable. At runtime, the placeholder is replaced by the current value of the variable in the defined format.

Printing a variable as a decimal number %d

Printing a variable as an unsigned decimal number %u

Printing a variable as an unsigned hexadecimal number %x

Printing a character string %s

Printing a Real number %f

How to display after the decimal point in real number display.

%. < Specify the number of digits >f

Example：If you want to display the third decimal place, please write "%. 3f".

7.3.4.3. Creation of control section of visualization

For details on how to use Label Element, refer to How to create monitor section.

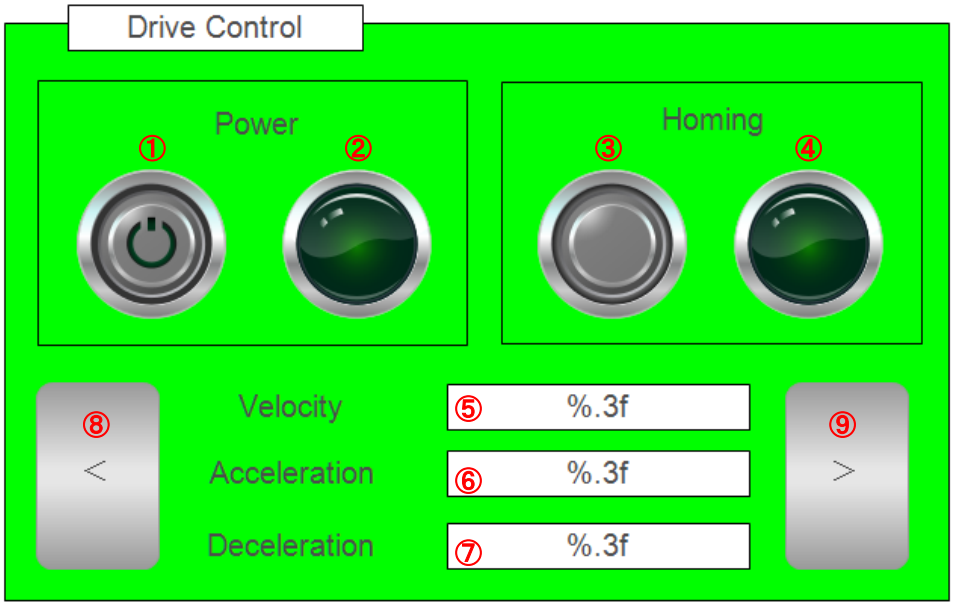


Fig.7.35 Element structure of control unit

No.	Detail
①	Servo on / off button
②	Servo on status lamp
③	Homing button
④	Homing status lamp
⑤	Element for JOG velocity setting
⑥	Element for JOG acceleration setting
⑦	Element for JOG deceleration setting
⑧	JOG forward button
⑨	JOG backward button

The contents of each setting are described below.

【①Servo on / off button】

Item	Detail
Element	Power switc（Tag: Lamps/Switches/images）
Variable	PLC_PRG.Axis1_SonFlag
Background—Image	Green

【②Servo on status lamp】

Item	Detail
Element	Lamp（Tag: Lamps/Switches/images）
Variable	PLC_PRG.Axis1_Power.Status
Background—Image	Green

【③Homing button】

Item	Detail
Element	Push switch（Tag: Lamps/Switches/images）
Variable	PLC_PRG.Axis1_Home.Execute
Background—Image	Gray

【④Homing status lamp】

Item	Detail
Element	Lamp（Tag: Lamps/Switches/images）
Variable	PLC_PRG.Axis1_Home.Done
Background—Image	Green

【⑤Element for JOG velocity setting】

Item	Detail
Element	Rectangle（Tag: Basic）
Texts—Text	%.3f
Text properties—Font	Size：15
Text variables—Text variable	PLC_PRG.Axis1_Jog.Velocity
Inputconfiguration—OnMouseClick—Write a Variable	
Input type	Text input with limits
Min	0
Max	3600

【⑥Element for JOG acceleration setting】

Item	Detail
Element	Rectangle（Tag: Basic）
Texts—Text	%.3f
Text properties—Font	Size：15
Text variables—Text variable	PLC_PRG.Axis1_Jog.Acceleration
Inputconfiguration—OnMouseClick—Write a Variable	
Input type	Text input with limits
Min	0
Max	3600

【⑦Element for JOG deceleration setting】

Item	Detail
Element	Rectangle（Tag: Basic）
Texts—Text	%.3f
Text properties—Font	Size：15
Text variables—Text variable	PLC_PRG.Axis1_Jog.Deceleration
Inputconfiguration—OnMouseClick—Write a Variable	
Input type	Text input with limits
Min	0
Max	3600

【⑧JOG forward button】

Item	Detail
Element	Button（Tag: Common controls）
Clolors—Alarm color	Yellow
Texts—Text	＜
Text properties—Font	Size：15
Color variables—Toggle color	PLC_PRG.Axis1_Jog.JogBackward
Inputconfiguration—Toggle—Variable	PLC_PRG.Axis1_Jog.JogBackward

【⑨JOG backward button】

Item	Detail
Element	Button（Tag: Common controls）
Clolors—Alarm color	Yellow
Texts—Text	＞
Text properties—Font	Size：15
Color variables—Toggle color	PLC_PRG.Axis1_Jog.JogForward
Inputconfiguration—Toggle—Variable	PLC_PRG.Axis1_Jog.JogForward

This completes the creation of the visualization screen. By downloading the project, you will be able to control motion from the visualization.

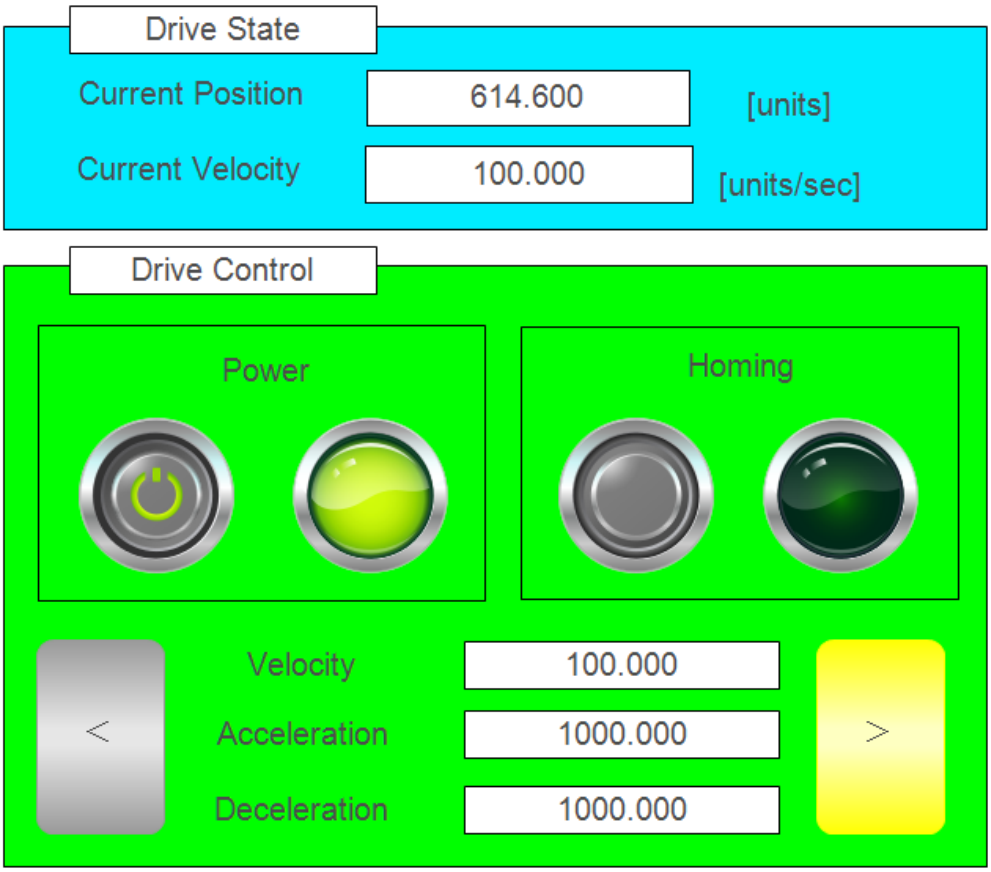


Fig.7.36 Visualization control screen

7.3.5. Web Visualization

In the Web Visualization, you can access the visualization screen from a general-purpose Web browser such as PC or tablet.

Please use HTML 5 compatible web browser.

If the Web visualization function is enabled, the following output is made in the log.

**********	CmpWebServer
Connection type : HTTP, HTTPS	CmpWebServer
HTTPS port : 8443	CmpWebServer
HTTP port : 8080	CmpWebServer
Host : SMC200	CmpWebServer
Root directory : $PlcLogic$/$visu$	CmpWebServer
Web Server	CmpWebServer
**********	CmpWebServer

Fig.7.37 Web visualization log

To access from the web browser please set the following URL.

http: // [IP address]: [8080] / [Name of .htm file]

If logs are not output, please check if there is a Web visualization object in the device tree. If you do not do it, right click on "Visualization Manager" and select "Add Object" → "Web Visualization" in order to add Web Visualization Object.

With the Web Visualization Object you can configure the Web Visualization function.

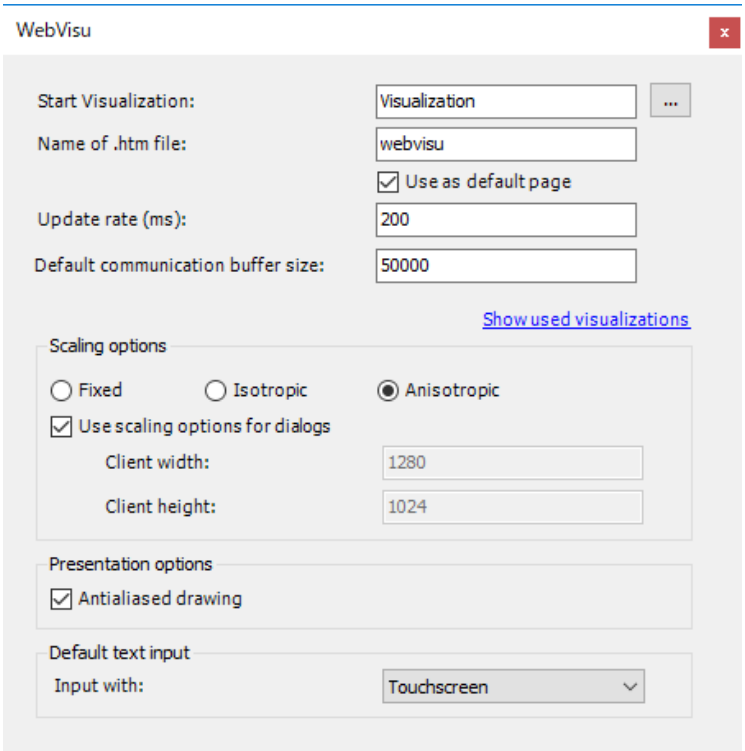


Fig.7.38 Web visualization object

Item	Detail
Start visualization	Do not change from the default "Visualization".
Name of .htm file	Base URL of the web page. The URL is also specified as the address in the web browser. Example: http://localhost:8080/webvisu.htm
Update rate (ms)	Refresh rate (in milliseconds) in the web browser
Default communication buffer size	Default size for communication buffer (in bytes). Defines the maximum available memory for data transfer between the web client and the web server.
Scaling options	**Fixed** : Fixed size of the visualization. The values used are Client height and Client width. **Isotropic** : The size of the visualization is adapted to the dimensions of the web browser, retaining the proportions of the visualization. **Anisotropic** : The size of the visualization is adapted to the web browser.
Antialiased drawing	Antialiasing is used when drawing the visualization in the web browser.
Default text input	**Touchscreen** : Text input on the WebVisu with touchscreen **Keyboard** : Text input on the WebVisu with keyboard

7.4. Single axis control program

We prepare a program assuming the following devices as a sample program of single axis control. Use "Motion Standard project" as a template. For the FB used in the program, refer to "9.2.1 Function block for single axis control" or later.

7.4.1. Sample program summary

We will assume a single axis robot transport system. When the object to be transported is placed on the table, the sensor reacts and moves to the target position (500 mm). When the goods are removed from the table, return it to its original position.

Transport device shaft is attached to a ball screw of 10 mm lead, use it as a finite axis. The operating range is from -10 mm to 600 mm.

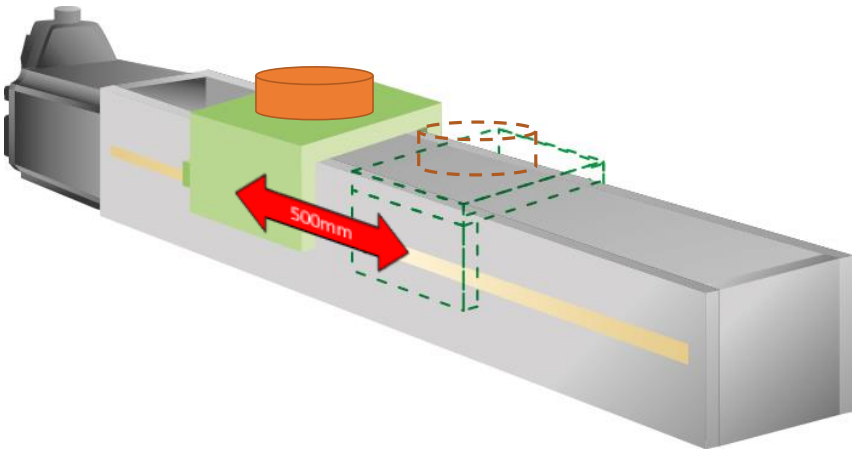


Fig 7.39 Schematic diagram of the device

7.4.2. Configuration

7.4.2.1. I/O setting

In the sample program, we use two sensors (placement and removal of goods). Connect each signal to the digital input of the S200. Also, give a variable name to the digital input.Double-click "Device (SMC200-A)" and select "Device I/O Mapping". After that, set the variable name as shown in the figure below.

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI0~7	%IB38	BYTE		
xSet		Bit0	%IX38.0	BOOL		
xGet		Bit1	%IX38.1	BOOL		
		Bit2	%IX38.2	BOOL		

Fig.7.40 I/O mapping setting

7.4.2.2. Axis setting

Add the slave and axis in the same procedure as “7.2.2.1 Add slave“ and “7.2.2.2 Add axis“. Change the name to "Drive 1".

Set the axis settings as shown below.

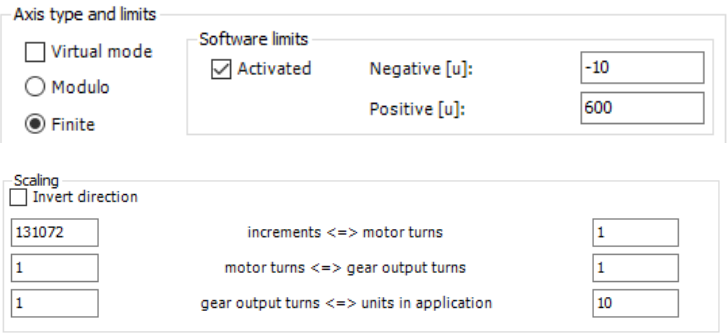


Fig.7.41 Transport device axis setting

7.4.3. Sample program

A list of variables used in the sample program is shown below. Add the following information to "Motion_PRG".

【Declaration section】

```
Drive1_Power      : MC_Power;          // For servo on/off contro
Drive1_Home       : IoSanyoDevice.SanHome ;    // For homing
Drive1_Move       : MC_MoveAbsolute;// For moving
MainStep          : INT;               //Main operation step management variable
Trigger           : R_TRIG;            // Execution trigger detection FB
```

In this program, separate FB execution and flag control are described. Write the execution part of FB at the top of the program.

【Implementation section】

```
Drive1_Power(Axis := Drive1, Enable := TRUE);
Drive1_Home(Axis := Drive1);
Drive1_Move(
        Axis:= Drive1,
        Velocity:= 100,
        Acceleration:= 10000,
        Deceleration:= 10000);
```

Describe the flag control part below the execution part.

【Implementation section】

```
CASE MainStep OF

 0 : (* S-ON *)
    Drive1_Power.bDriveStart := TRUE;
    Drive1_Power.bRegulatorOn := TRUE;
    IF Drive1_Power.Status THEN
      MainStep := 1;
    END_IF

 1 : (* Homing *)
    Drive1_Home.Execute := TRUE;
    IF Drive1_Home.Done THEN
      MainStep := 2;
    END_IF

 2 : (* Set Wait *)
    Trigger(CLK:= xSet);

    (* Move Start *)
    IF Trigger.Q THEN
      Drive1_Move.Execute    := TRUE;
      Drive1_Move.Position := 500;
      MainStep := 3;
    END_IF

 3 : (*Move Done *)
    IF Drive1_Move.Done THEN
      Drive1_Move.Execute    := FALSE;
      MainStep := 4;
    END_IF

 4 : (*Get Wait *)
    Trigger(CLK:= xGet);

    (* Move to original position *)
    IF Trigger.Q THEN
      Drive1_Move.Execute    := TRUE;
      Drive1_Move.Position := 0;
      MainStep := 5;
    END_IF

 5 : (* Arrive at original position *)
    IF Drive1_Move.Done THEN
      Drive1_Move.Execute    := FALSE;
      MainStep := 2;
    END_IF

END_CASE
```

7.4.4. Operation check by trace

Test this sample program by trace.

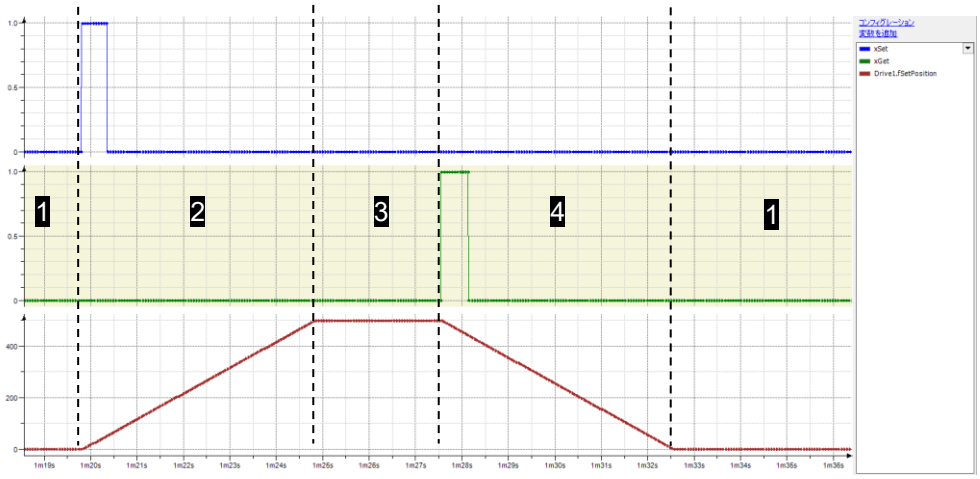


Fig.7.42 Tracing the sample program

No.	Detail
1	Waiting for placement of goods
2	Move to target position
3	Waiting for taking out goods
4	Move to original position

7.5. PTP control program

This section gives an overview of PTP control and creates a sample program. Refer to "9.2.2 PTP control function block" or later for FB used for PTP control. Use "Motion Standard project" as a template.

7.5.1. Sample program summary

Assume a transfer system that handles three types of workpieces with a single-axis robot. It is necessary to prepare multiple target positions for each work. Assign and control the target position for each switch.

After the work is placed on the table, press the work-specific switch to move the table to the target position, stop for 2 seconds, and return to the start position.

The transfer device axis is mounted on a ball screw with a lead of 10 mm and used as a finite axis. The work is called work A, work B and work C respectively. Each target position is 100 mm for Work A, 200 mm for Work B, and 300 mm for Work C.

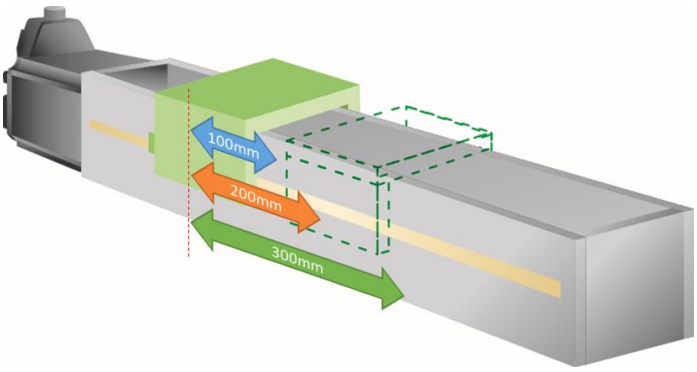


Fig 7.43 Device schematic

7.5.2. Configuration

7.5.2.1. I / O setting

The I/O settings used in the sample program are shown below. These are switches for work A, work B and work C respectively. Double-click "Device (SMC200-A)" and select "Device I/O Mapping". After that, set the variable name as shown in the figure below.

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI0~7	%IB38	BYTE		
bWorkA		Bit0	%IX38.0	BOOL		
bWorkB		Bit1	%IX38.1	BOOL		
bWorkC		Bit2	%IX38.2	BOOL		

Fig 7.44 I / O mapping setting

7.5.2.2. Add PTP control axis

Add an axis for PTP control and set it.

1. Add a slave in the same procedure as “7.2.2.1Add slave“.
2. Right-click on the added slave and click "Add SoftMotionLight CiA402 axis".

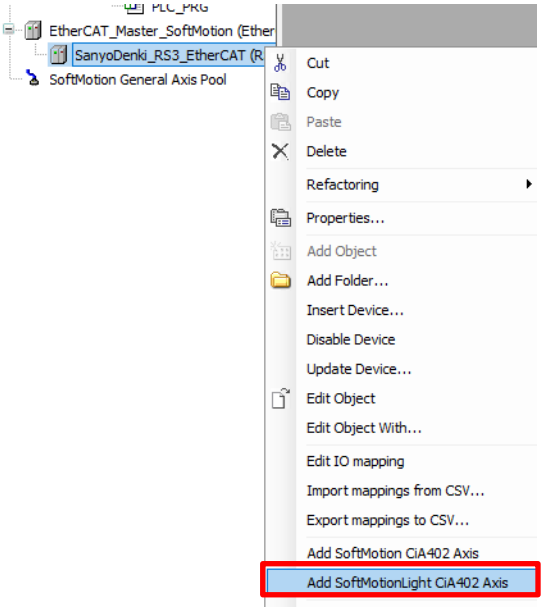


Fig 7.45 Add axis for PTP control

3. As the axis is added, change the name to "Drive 1".

7.5.2.3. Axis setting for PTP control

The following items can be set for axes for PTP control.

In this case, set the increments to 131072 and the unit in application to 10.

Do not change any other settings from the default settings.

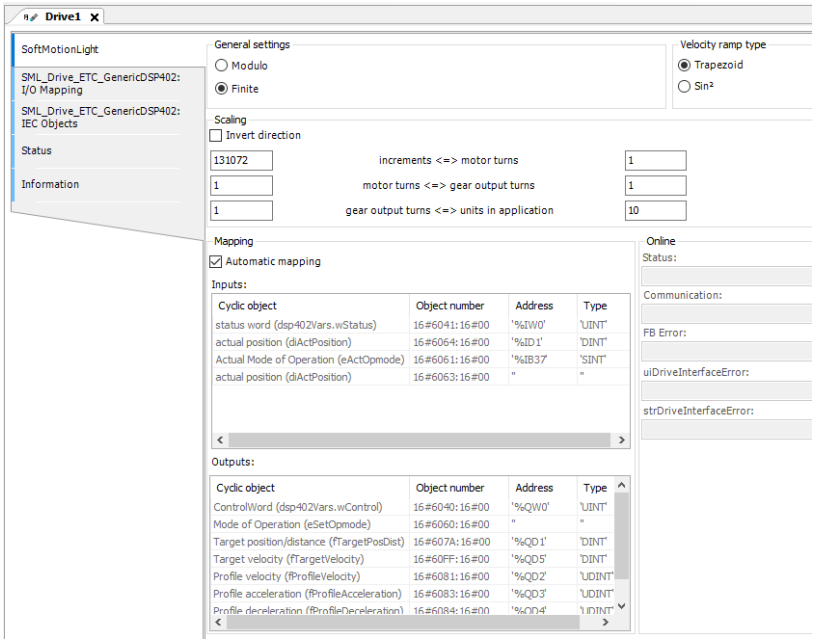


Fig 7.46 Axis setting screen for PTP control

Item	Detail
Modulo	The drive turns endlessly without limiting the traversing range
Finite	The drive has a fixed work area
Velocity ramp type	Defines the velocity profile for motion-generating single-axis and master/slave modules. Trapezoid, Sin^2
Invert direction	The direction of rotation is reversed. The motor receives the specified values with inversed signs. (Do not enable the "Invert direction" parameter of the SML axis. If it is enabled, unexpected behavior may occur. If you want to set the reversal of the rotation direction, set it on the EtherCAT slave side)
increments<=> motor turns	Number of increments that correspond to a given number of motor rotations. If the number of pulses per revolution is 131072 (17 bits), set increment to 131072 and motor rotation to 1.
motor turns <=> gear output turns	Number of motor rotations that correspond to a given number of gear output rotations. If the reduction ratio is 1/10, please set the motor rotation to 10 and the gear output rotation to 1.
gear output turns <=> units in application	Number of gear output rotations that correspond to a unit in the application. For one gear revolution at 360 degrees, set the gear output rotation to 1 and the unit within the application to 360.
Automatic mapping	IEC parameters that affect the drive are automatically mapped to the corresponding inputs and outputs of the device.

7.5.3. Sample program

The following is a list of variables used in the sample program.

【Declaration section】

```
Drive1_Power     :    MC_Power_SML;     //For servo on/off contro
Drive1_Home      :    MC_Home_SML;      //For homing
Drive1_Move      :    MC_MoveAbsolute_SML;      // For moving
MainStep         :    INT;              //Main operation step management variable
Trigger          :    R_TRIG;           // Execution trigger detection FB
Delay, Timer     :    TON;              //Timer
SetOpmode        :    SML_SetOpmode;    //Change of operation mode
```

In this program, FB execution and flag control are described separately. Write the execution part of the FB at the beginning of the program.

【Implementation section】

```
Timer(PT:= T#2S);
Delay(PT:= T#0.5S);
SetOpmode(Axis:= Drive1);
Drive1_Power(Axis := Drive1, Enable := TRUE);
Drive1_Home(Axis := Drive1);
Drive1_Move(Axis:= Drive1,
            Velocity:= 100,
            Acceleration:= 10000,
            Deceleration:= 10000);
```

Describe the flag control part below the execution part.

【Implementation section】

```
CASE MainStep OF

 0 : (*S-ON*)
    Drive1_Power.bDriveStart := TRUE;
    Drive1_Power.bRegulatorOn := TRUE;
    IF Drive1_Power.Status THEN
      MainStep := 1;
    END_IF

 1 : (*Delay*)
    Delay.IN := TRUE;
    IF Delay.Q THEN
        MainStep := 2;
    END_IF

 2 : (*Change of operation mode *)
    SetOpmode.eOpmode := SML_OPMODE.SML_OP_HOMING;
    SetOpmode.bExecute := TRUE;
    IF Drive1.eActOpmode = SML_OPMODE.SML_OP_HOMING THEN
        SetOpmode.bExecute := FALSE;
        MainStep := 3;
    END_IF

 3 : (*Homing*)
    Drive1_Home.Execute := TRUE;
    IF Drive1_Home.Done THEN
        MainStep := 4;
    END_IF

 4 : (*Change of operation mode *)
    SetOpmode.eOpmode := SML_OPMODE.SML_OP_POSITION;
    SetOpmode.bExecute := TRUE;
    IF Drive1.eActOpmode = SML_OPMODE.SML_OP_POSITION THEN
        SetOpmode.bExecute := FALSE;
        MainStep := 5;
    END_IF
```

```
 5 : (*Waiting for transfer*)
    Trigger(CLK:= bWorkA OR bWorkB OR bWorkC);

    (*Transfer start*)
    IF Trigger.Q THEN
        IF bWorkA THEN
            Drive1_Move.Position := 100;
        ELSIF bWorkB THEN
            Drive1_Move.Position := 200;
        ELSIF bWorkC THEN
            Drive1_Move.Position := 300;
        END_IF
        Drive1_Move.Execute    := TRUE;
        MainStep := 6;
    END_IF

 6 : (*Transport complete*)
    IF Drive1_Move.Done THEN
      Drive1_Move.Execute    := FALSE;
      MainStep := 7;
    END_IF

 7 : (*Stop for 2 seconds*)
    Timer.IN := TRUE;

    (*Return to the start position*)
    IF Timer.Q THEN
        Timer.IN := FALSE;
        Drive1_Move.Execute    := TRUE;
        Drive1_Move.Position := 0;
        MainStep := 8;
    END_IF

 8 : (*Arrive at start position*)
    IF Drive1_Move.Done THEN
      Drive1_Move.Execute    := FALSE;
      MainStep := 5;
    END_IF

END_CASE
```

In PTP control, when executing an FB of an operation mode different from the current operation mode, it is necessary to change the operation mode in advance. For example, you need to change to the homing mode before execute homing. If you want to perform position control after homing in homing mode, you need to change to profile position mode in advance.

7.5.4. Operation check by trace

Test this sample program by trace.

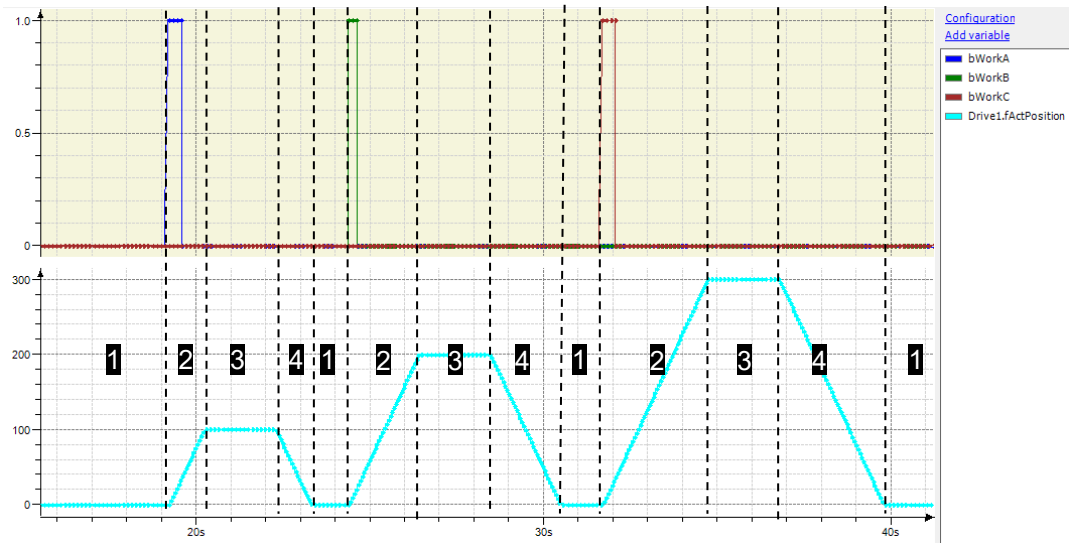


Fig 7.47 Tracing the sample program

No.	Details
1	Waiting for input from switch
2	Move to target position
3	Stop for 2 seconds at the target position
4	Move to start position

7.6. Infinite rotation axis control program

Depending on the system, such as the rotation axis of a belt conveyor, the servo motor is controlled as an infinite rotation axis (modulo axis).

This section explains the points to note when controlling the infinite rotation axis, and creates a sample program. Use "Motion Standard Project" for the template.

7.6.1. Precautions for infinite rotation axis control

When controlling the infinite rotation axis with the SANMOTION C Software Tool 2.0.0, it is necessary to pay attention to the setting of the modulo value (the value at which the infinite rotation axis is reset).

As an example, assume the following two systems.

(1) Mechanism: Gear ratio between motor and machine: [128: 1]

Controller: Modulo function enabled, user coordinates (angle): 360 [°]

Encoder: Multi-rotation backup absolute encoder (23 bits)

Modulo value: 1,073,741,696 [pulse] ([2 ^ 23-1] x 128)

(2) Mechanism: Gear ratio between motor and machine: [100: 1]

Controller: Modulo function enabled, user coordinates (angle): 360 [°]

Encoder: Multi-rotation backup absolute encoder (23 bits)

Modulo value: 838,860,700 [pulse] ([2 ^ 23-1] x 100)

Since the two mechanisms have different gear ratios, the modulo value (the value at which the infinite rotation axis is reset) also has a different value.

Inside the servo amplifier, the current position is managed by 32-bit data. When the axis rotates forward and reaches the maximum value of 32 bits, the value is rounded and added again from the minimum value.

When the modulo function is enabled, the controller calculates the actual position of the infinite rotation axis based on the reference position and the 32-bit data received from the amplifier. The position data for systems ① and ② are shown below.

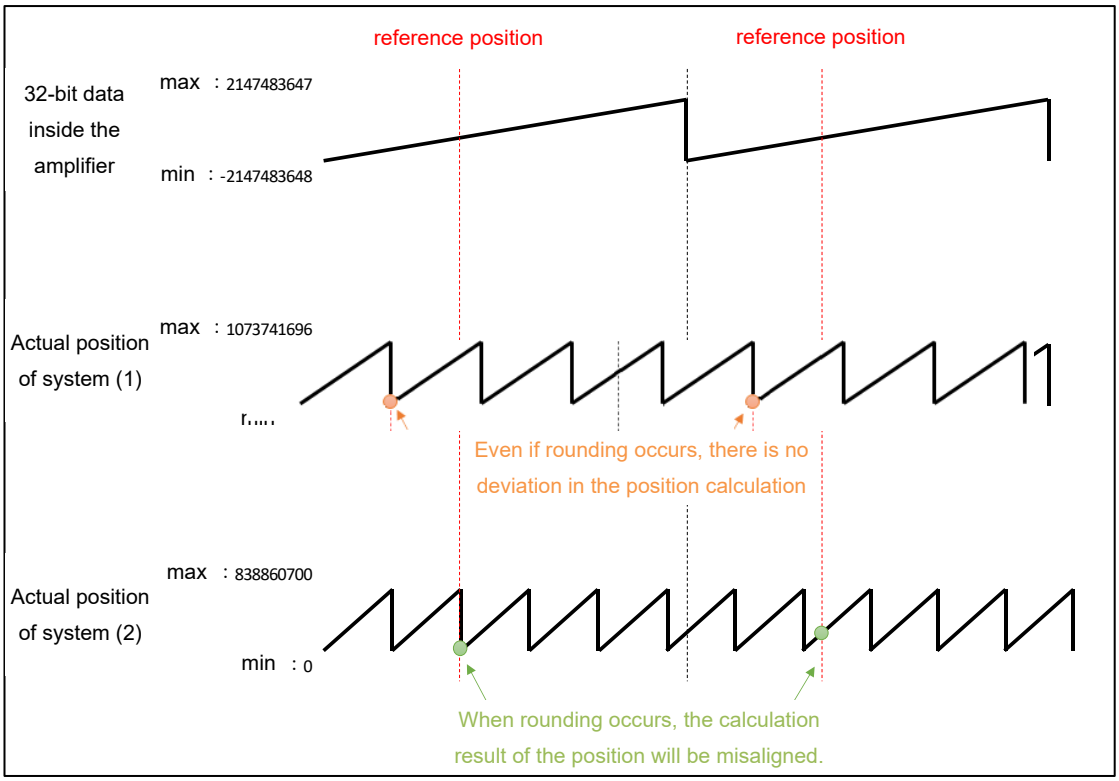


Fig 7.48 Correspondence between reference position and actual position on infinite rotation axis

In the case of system (1), the modulo value is a power of 2, and 32 bits can be divided equally. Even if 32-bit data is rounded, the correspondence between the reference position and the actual position does not change, and the calculation can be performed normally.

In the case of system (2), the modulo value is not a power of 2, so if 32-bit data rounding occurs, the correspondence between the reference position and the actual position will shift. There is no problem during continuous operation, but if the power of the controller is turned on again in this state, an incorrect value will be set at the current position of the axis.

For the above reasons, in a system where the modulo value is not a power of 2, it is necessary to perform the process of updating the reference position by the program shown in the next section.

7.6.2. Sample program summary

Assume the following infinite rotation axis.

Mechanism: Gear ratio between motor and machine: [10: 1]

Controller: Modulo function enabled, user coordinates (angle): 360 [°]

Encoder: Multi-rotation backup absolute encoder (23 bits)

Modulo value: 838,860,70 [pulse] ([2 ^ 23-1] x 10)

In this system, the modulo value is not a power of 2, so 32-bit rounding causes a shift in the reference position. You need to update the reference position that the controller uses to calculate the actual position.

Use the following FB "SMC3_PersistPosition".

This FB can hold the 32-bit position data of the servo amplifier inside the controller when the axis passes the user coordinate "0" of the controller. By using this value as a new reference position, even if 32-bit data is rounded, the actual position can be calculated without any deviation.

Enter a persistent variable in "Persistent Data" because normal variables are initialized when the controller is turned on again.

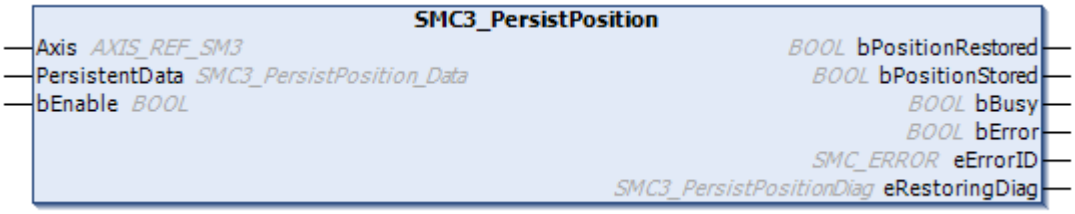


Fig 7.49 SMC3_PersistPosition

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
PersistentData	SMC3_PersistPosition_Data	Structure to store the data to be retained
VAR_INPUT		
Enable	BOOL	TRUE: Enable FB
VAR_OUTPUT		
bPositionRestored	BOOL	TRUE: The position has been restored during the last start-up of the axis
bPositionStored	BOOL	TRUE: The position has been stored during the last call
bBusy	BOOL	TRUE: FB is not idle
bError	BOOL	TRUE: Error has occured within the function block
eErrorID	SMC_ERROR	Error identification
eRestoringDiag	SMC3_PersistPositionDiag	Diagnostic information about restoring

7.6.3. Configuration

7.6.3.1. I/O setting

In the sample program, the axis is rotated and stopped by the digital input of S200. Double-click "Device (SMC200-A)" and select "Device I / O Mapping". After that, set the variable name as shown in the figure below.

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI0~7	%IB38	BYTE		
xStart		Bit0	%IX38.0	BOOL		
xStop		Bit1	%IX38.1	BOOL		
		Bit2	%IX38.2	BOOL		
		Bit3	%IX38.3	BOOL		

Fig 7.50 I/O setting

7.6.3.2. Axis setting

Configure the infinite rotation axis in the project. Follow the same procedure as in "7.2.2.1. Add slave" and "7.2.2.2. Add axis" to add a slave and an axis. Change the name of the axis to "Axis1".

The infinite rotation axis is connected to the reducer, and the gear ratio is 10: 1. The modulo value is "360" for one rotation of the gear output. This sets the internal modulo value to 838,860,70 [pulse].

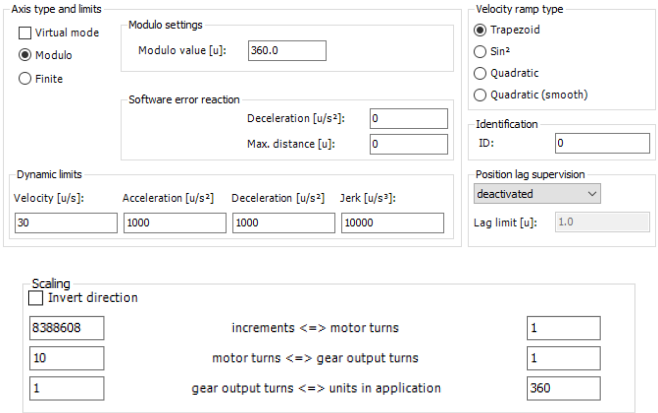


Fig 7.51 Infinite rotation axis configuration

7.6.3.3. Persistent variables setting

Create a persistent variable to store the reference position used for position calculation. Persistent variables retain their values even when the controller is powered on again.

Right-click on “Application” in the device tree and select “Add Object” → “Persistent Variables” to add a persistent variable. Edit the declaration part of the added persistent variable as follows.

```
VAR_GLOBAL PERSISTENT RETAIN
        PersistentData: SMC3_PersistPosition_Data;
END_VAR
```

7.6.4. Sample program

The variables used in the sample program are described below. Add the following to "Motion_PRG".

【Declaration section】

```
PersistPosition     :     SMC3_PersistPosition; // For retaining the reference position
Axis1_Power         :     MC_Power; //For servo on / off control
Axis1_Home          :     IoSanyoDevice.SanHome; // For homing
Axis1_Move          :     MC_MoveVelocity; // For velocity movement
Axis1_Stop          :     MC_Stop; // For stopping
MainStep            :     INT; // Operation step management variables
```

In this program, FB execution and flag control are described separately. Write the execution part of the FB at the beginning of the program.

【Implementation section】

```
PersistPosition(Axis:=Axis1,PersistentData:=PersistentVars.PersistentData, bEnable:= TRUE);
Axis1_Power(Axis := Axis1, Enable := TRUE);
Axis1_Home(Axis := Axis1);
Axis1_Move(
        Axis:= Axis1,
        Velocity:= 90,
        Acceleration:= 900,
        Deceleration:= 900);
Axis1_Stop(Axis:= Axis1);
```

Describe the flag control part below the execution part.

```
CASE MainStep OF
        0 : (* Waiting for the start signal *)
                IF xStart THEN
                        MainStep := 1;
                END_IF

        1 : (* Servo on *)
                Axis1_Power.bDriveStart := TRUE;
                Axis1_Power.bRegulatorOn := TRUE;
                IF Axis1_Power.Status THEN
                        MainStep := 2;
                END_IF

        2 : (* Homing, start of operation *)
                Axis1_Home.Execute := TRUE;
                IF Axis1_Home.Done THEN
                        Axis1_Move.Execute := TRUE;
                        MainStep := 3;
                END_IF

        3 : (* Stopping operation *)
                IF xStop THEN
                        Axis1_Stop.Execute := TRUE;
                END_IF
 END_CASE
```

7.6.5. Operation check by trace

Check the operation of the sample program with a trace.

Set the variables to be traced as follows.

Diagram	Variable name	Detail
1	Axis1.diSetPosition	Command position to amplifier (32bit)
	Axis1.diActPosition	Current position received from the amplifier (32bit)
2	Axis1.fSetPosition	Command position to amplifier (user unit)
	Axis1.fActPosition	Current position received from the amplifier (user unit)
3	PersistentData.dwPosOffsetForResiduals	Retained reference position (32bit)

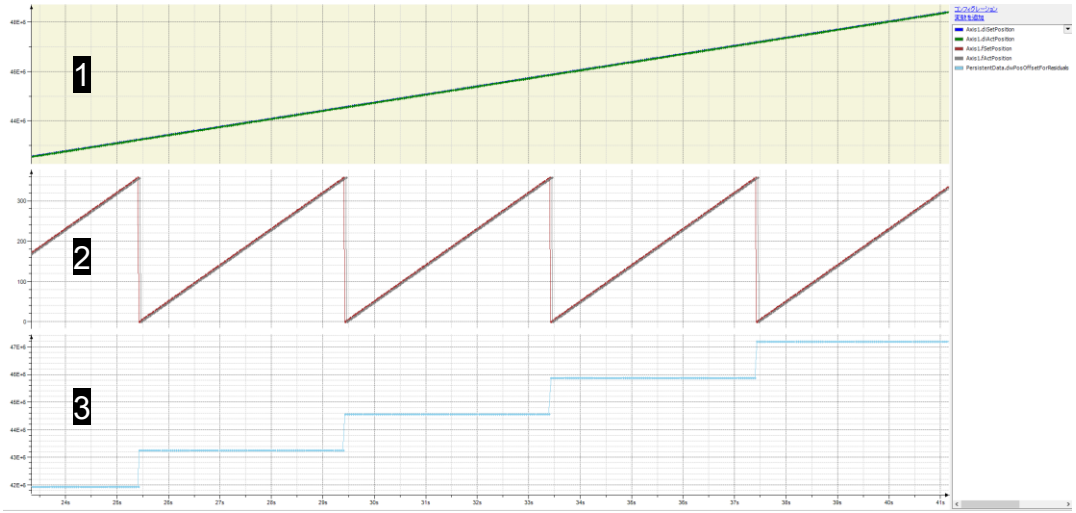


Fig 7.52 Operation check by trace

You can see that the value of "PersistentData.dwPosOffsetForResiduals" is updated every time the axis passes the user coordinate "0".

After "Axis1.diActPosition" exceeds the maximum value and becomes a negative value, set xStop to TRUE to stop the axis. Record the value of "Axis1.fActPosition" at that time.

Turn on the power of the controller again, and check that the value of "Axis1.fActPosition" is correct.

In this sample program, it takes time for "Axis1.diActPosition" to exceed the maximum value. Adjust the velocity of the axis appropriately.

Since the deviation between the command position and the current position is small, the lines in diagrams 1 and 2 appear to overlap.

7.7. Synchronous Motion Control

Synchronous Motion Control means that the motion of an axis (Slave Axis) is derived from the motion of another axis (Master Axis) by a defined relation. Such a relation might be an electronic gear (MC_GearIn and MC_GearInPos) or an electronic cam (MC_CamIn). With electronic gearing a linear relationship between master axis and slave axis exists, which is specified in form of a gear ratio. With electronic camming any relation between the position of the master axis and the position of the slave axis can be achieved by means of cam tables.

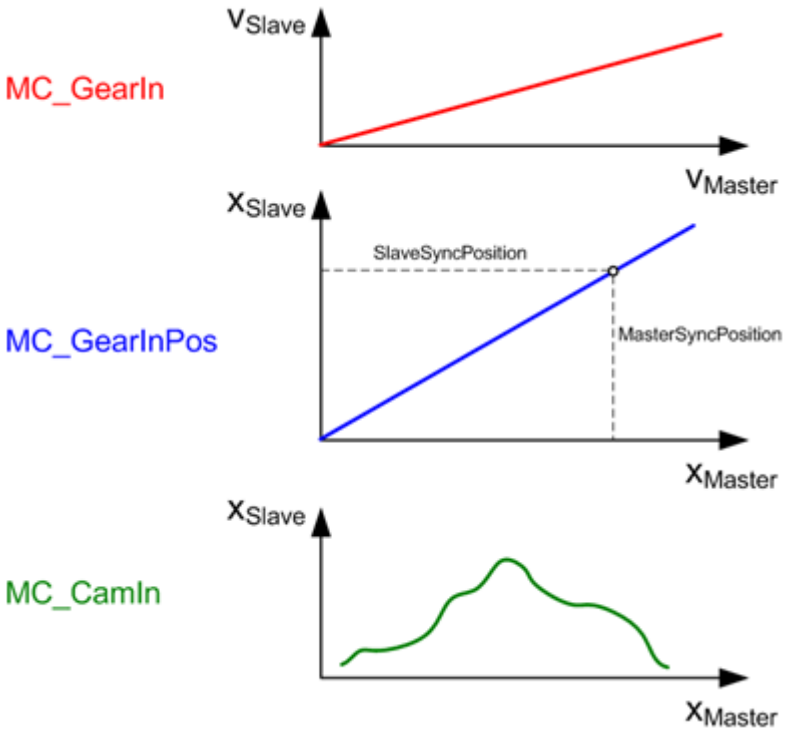


Fig.7.53 Illustration of relationship between Slave Axis and Master Axis for MC_GearIn, MC_GearInPos and MC_CamIn

By means of the FB MC_Phasing an additional phase shift between slave axis and master axis can be established. This phase shift acts as a position offset to the position of the master axis.

For details of each FB, refers to”9.2.3Function block for multi-axis control”.

7.7.1. Electronic gear

Create a sample program of electronic gear control. The template uses "Motion Standard project".

7.7.1.1. Sample program summary

Endless material (e.g. paper) should be cut into a given length. The cutting of the material must be done without interrupting the machine. Primer goal is to maximize the amount of cuts per time.

The material is transported by a belt conveyor. The belt conveyor is driven by a servo drive and operates at 100 mm / s. The cutting device is mounted on a linear axis driven by a servo shaft. The maximum velocity of the cutting device is 200 mm / s.

The length of the cut length is 1000 mm. The servo drive of the cutter is a linear drive. This axis moves synchronously with the servo drive mounted on the belt conveyor during the cutting operation.

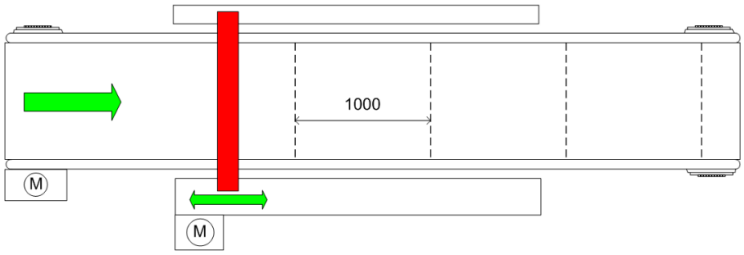


Fig 7.54 Top view of the machine

7.7.1.2. Sequence

During initialization, the servo axis is switched on and if necessary, both axes are referenced. Also move the belt conveyor at a constant speed. Then the cyclic part of the program starts.

If the encoder axis crosses a determined position, the gear in process starts. The slave axis (cutting machine) starts to synchronize to the master axis (belt conveyor). When the axis is synchronous to the belt conveyor, the cutting process is started. After the cutting has finished, the linear axis is moved back to its starting position to start a new cycle.

In this sample, it is assumed that the cutting process requires about 4 seconds and the waiting time at the start position takes about 1 second.

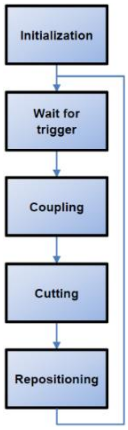


Fig.7.55 Execution diagram of the PLC program

7.7.1.3. Configuration

An axis for the belt conveyor axis with the name “Conveyor “ are configured and a linear axis with the name “Cutter” are configured in the project. Follow the same procedure as for "6.2.2.1 Add slave" and "6.2.2.2Add axis" for " 6.6.1.3. Configuration" to add slaves and axes.The belt conveyor is driven by a gear, which has a gear ratio of 20:1. After the gear box a pulley with a reduction of 500 is added. The belt conveyor axis is used as modulo axis. To reach the requested cutting length of 100 modulo overrun is configured at 1000.

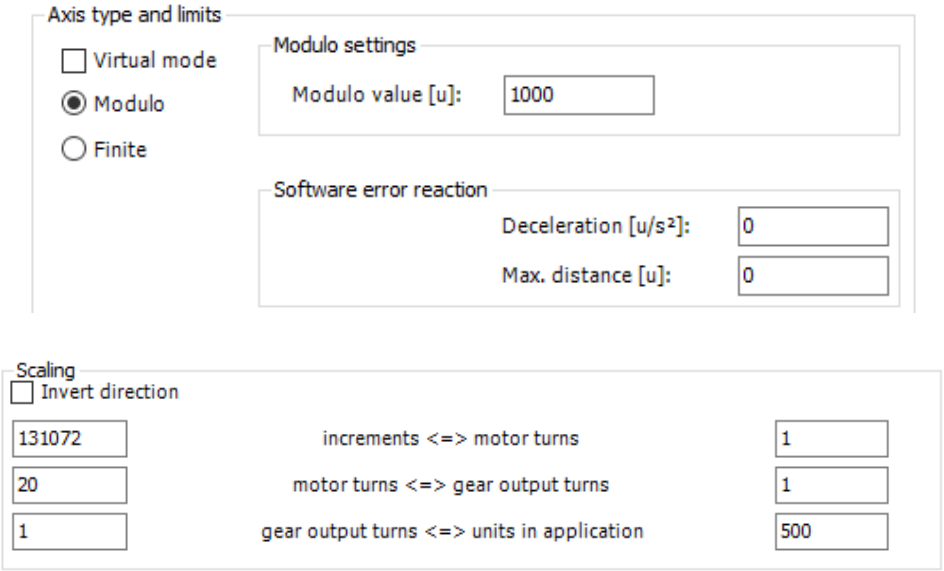


Fig.7.56 Conveyor configuration

Cutter is attached to ball screw of 20 mm lead. this axis is used as a finite axis. The operating range from -20 mm to 1000 mm.

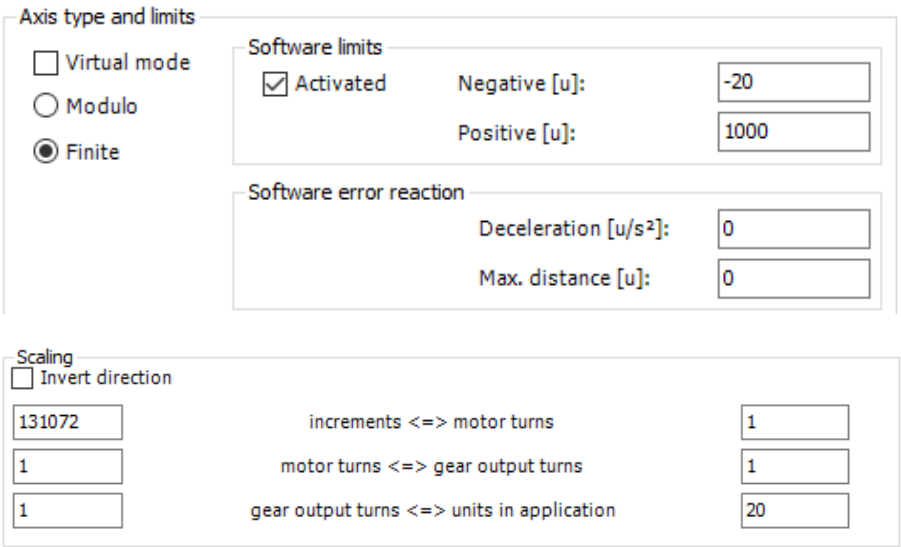


Fig.7.57 Cutter configuration

7.7.1.4. Sample program

A list of variables used in the sample program is shown below.

【Declaration section】

```
Cutter_Power      :    MC_Power;            //For servo on/off control of cutter
Conveyor_Power    :    MC_Power;            //For servo on/off control of Conveyor
Cutter_Home       :    IoSanyoDevice.SanHome;     //For homing of cutter
Conveyor_Home     :    IoSanyoDevice.SanHome;     //For homing of Conveyor
Cutter_Move       :    MC_MoveAbsolute;     //For moving the waiting position of cutter
Conveyor_Move     :    MC_MoveVelocity;     //For endless motion at a specified
                                              velocity of Conveyor
GearInPos         :    MC_GearInPos;        //Velocity synchronization FB
MainStep          :    INT;                 //Main operation step management
                                              variable
SyncStep          :    INT;                 //Synchronization step management
                                              variable
TimeToCut         :    TON;                 //Cutting timer
```

In this program, separate FB execution and flag control are described. Write the execution part of FB at the top of the program.

【Implementation section】

```
Cutter_Power(Axis:= Cutter, Enable:= TRUE);
Conveyor_Power(Axis:= Conveyor, Enable:= TRUE);
Cutter_Home(Axis:= Cutter);
Conveyor_Home(Axis:= Conveyor);
Cutter_Move(Axis:= Cutter, Position:= 0, Velocity:= 200, Acceleration:= 3000, Deceleration:= 3000);
Conveyor_Move(Axis:= Conveyor, Velocity:= 100, Acceleration:= 1000, Deceleration:= 1000);
GearInPos(
        Master:= Conveyor,
        Slave:= Cutter,
        Execute:= ,
        RatioNumerator:= 1,
        RatioDenominator:= 1,
        MasterSyncPosition:= 200,
        SlaveSyncPosition:= 200,
        MasterStartDistance:= 200);
```

Describe the flag control part below the execution part. The relationship between "7.7.1.2Sequence" and program steps is described below.

Step	Detail
Initialization	MainStep : 0～2
Wait for trigger	SyncStep : 0
Coupling	SyncStep : 0
Cutting	SyncStep : 1
Repositioning	SyncStep : 2

【Implementation section】

```
CASE MainStep OF
 0 : (*Initialization*)
    Cutter_Power.bDriveStart := TRUE;
    Cutter_Power.bRegulatorOn := TRUE;
    Conveyor_Power.bDriveStart := TRUE;
    Conveyor_Power.bRegulatorOn := TRUE;
    IF Cutter_Power.Status AND Conveyor_Power.Status THEN
      MainStep := 1;
    END_IF

 1 : (*Homing*)
    Cutter_Home.Execute   := TRUE;
    Conveyor_Home.Execute := TRUE;
    IF Cutter_Home.Done AND Conveyor_Home.Done THEN
      MainStep := 2;
    END_IF

 2 : (*Conveyor start*)
    Conveyor_Move.Execute := TRUE;
    IF Conveyor_Move.InVelocity THEN
      MainStep := 3;
    END_IF

 3 : (*Wait coupling*)
    IF SyncStep = 0 THEN
      GearInPos.Execute := TRUE;
      Cutter_Move.Execute := FALSE;
      (*Start synchronization*)
      IF GearInPos.StartSync THEN
        SyncStep := 1;
      END_IF
    END_IF
    (*Cutting*)
    IF SyncStep = 1 THEN
      TimeToCut(IN:= GearInPos.InSync, PT:= T#4S);
      (*Disconnection processing completed*)
      IF TimeToCut.Q THEN
        TimeToCut(IN:= FALSE);
        SyncStep := 2;
      END_IF
    END_IF

    (*Return the cutting device to the waiting position*)
    IF SyncStep = 2 THEN
      Cutter_Move.Execute := TRUE;
      GearInPos.Execute := FALSE;
      IF Cutter_Move.Done THEN
        SyncStep := 0;
      END_IF
    END_IF
END_CASE
```

7.7.1.5. Operation check by trace

Test this sample program by trace.

Blue line: Motion_PRG.GearInPos.StartSync

Green line: Motion_PRG.GearInPos.InSync

Brown line: Motion_PRG.Cutter_Move.Busy

Gray line: Conveyor.fSetPosition

Light blue line: Cutter.fSetPosition

Orange line: Conveyor.fSetVelocity

Yellow line: Cutter.fSetVelocity

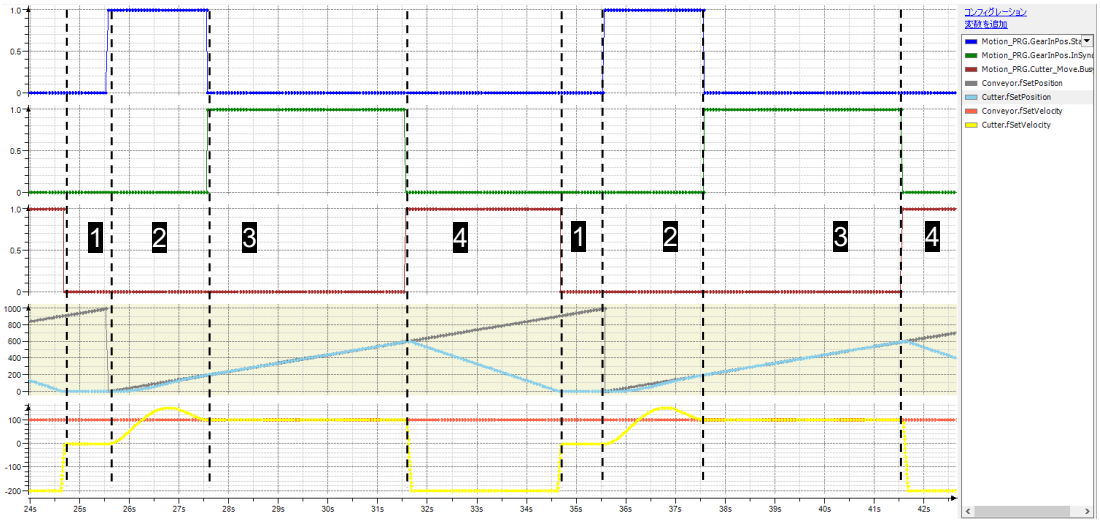


Fig 7.58 Tracing the sample program

No.	Detail
1	Trigger wait
2	Start of synchronization of belt conveyor and cutter
3	Synchronous operation of belt conveyor and axis “Cutter” During cutting process
4	Move “Cutter” to start position

7.7.2. Electronic cam

Create a program that assumes the following device as a sample program for electronic cam control.

Use "Motion Standard project" as a template.

7.7.2.1. Sample program summary

Endless material (e.g. toilet paper) should be perforated each 11 cm. In order not to damage the material, it is important to move the perforation knife synchronous to the material. The synchronization must be kept as long as the knife is in contact with the material.

A belt conveyor transports the material towards the perforation knife. The belt conveyor is driven by a servo axis.

The perforation machine is a cylindrical roll on which 2 knives are mounted. The cylinder has a perimeter of 400mm. That means the distance between each knife is 200mm

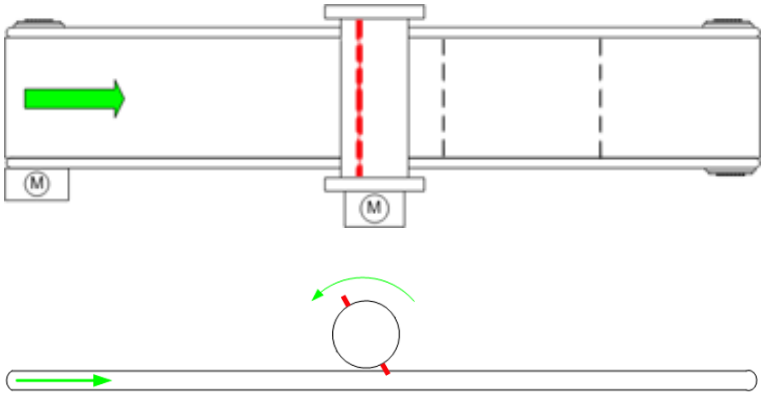


Fig.7.59 Top view of the machine

The belt conveyor is configured as modulo axis with the requested period length of 11cm.

The servo axis of the perforation axis is configured as a linear modulo axis.

Linear therefore, because in the following one can directly refer to the perimeter of the axis. This perimeter of the rotary knife must be moved synchronous to the belt conveyor in order not to damage the material. If this axis would not be configured as a rotary axis, the conversion form [mm] of the perforation axis to [mm] of the belt conveyor would have to be done together with the coupling process. So, the conversion already is done within the axis itself. For gearing a gear ratio of 1:1 can be used: 1mm of the rotary knife corresponds 1mm of the belt conveyor.

As there are 2 knives per revolution and the perimeter of the cylinder is 400mm, the axis has its modulo overrun at 200mm. In order not to damage the material during the perforation process, the cylinder must be synchronous in a specific range. The following sketch demonstrates the range in which the axes must be synchronous.

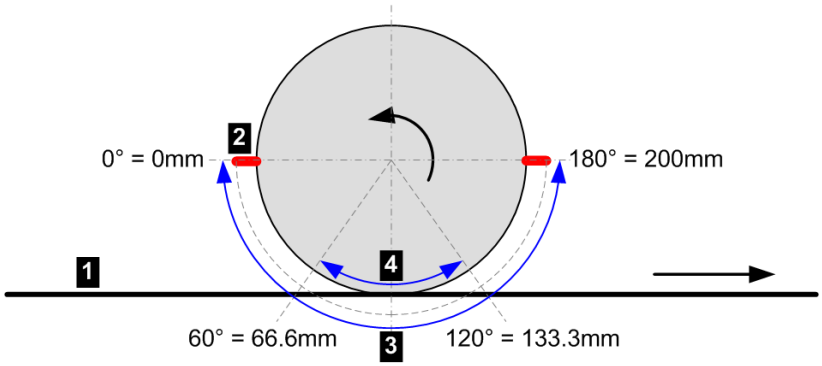


Fig.7.60 Schematic sketch of the perforation cylinder

1 ··· material being perforated	2 ··· Perforating knife
3 ··· both perforation knives are mounted exactly with a shift of 180°. The distance on the perimeter between the knives is 200mm.	4 ··· While the knife is in contact with the material, the cylinder must move synchronous to the belt conveyor. Here a distance of +/- 30° is determined.

The drawing shows the cylinder in zero position. Both knifes are in vertical position. The axis should be synchronous from 60° to 120°. Converted to the perimeter of the axis, the axis must be synchronous from 66.6mm to 133.3mm.

7.7.2.2. Sequence

During initialization, the axis are switched on and if necessary the axis are homed. Further the movement of the belt conveyor is started with constant velocity.

The program flow can be reduced to the following diagram.

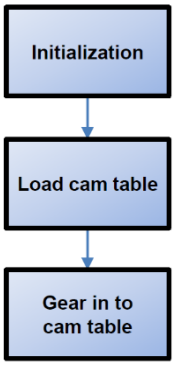


Fig.7.61 Execution diagram of the PLC program

7.7.2.3. Configuration

In the configuration two servo axes with the name “Conveyor” and “RotaryKnife” are configured. Follow the same procedure as for "6.2.2.1 Add slave" and "6.2.2.2Add axis" for " 6.6.2.2. Configuration" to add slaves and axes. Both are linear axis. The belt conveyor is driven by a gear, which has a gear ratio of 20:1. After the gear box a pulley with a reduction of 500 is added. The belt conveyor axis is used as modulo axis. The belt conveyor has modulo overrun of 110mm, the requested perforation distance of 11cm.

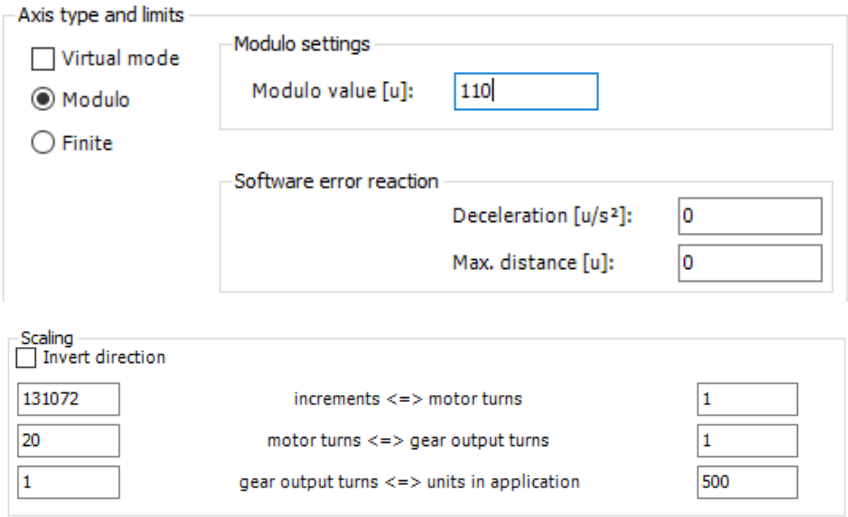


Fig.7.62 Belt conveyor configuration

The rotary knife is also a linear axis and has modulo overrun of 200mm, the distance of the two knifes on the perimeter of the cylinder. The conversion of degrees (°deg) of the cylinder to mm on the perimeter of the cylinder is done in the axis itself.

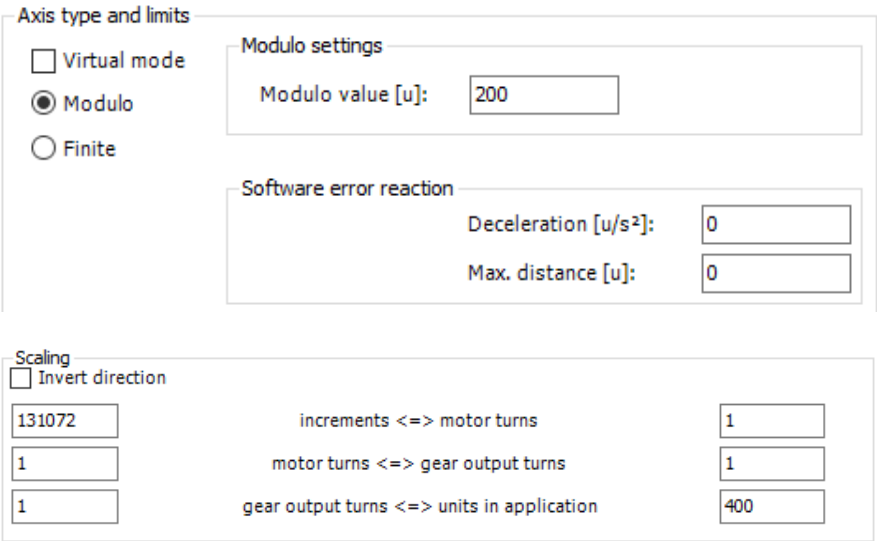


Fig.7.63 RotaryKnife configuration

7.7.2.4. Create a cam table

Create a cam table for electronic cam control.

1. Right click “Application” and select “Cam Table ...” from “Add Object”. You can set the object name when adding.

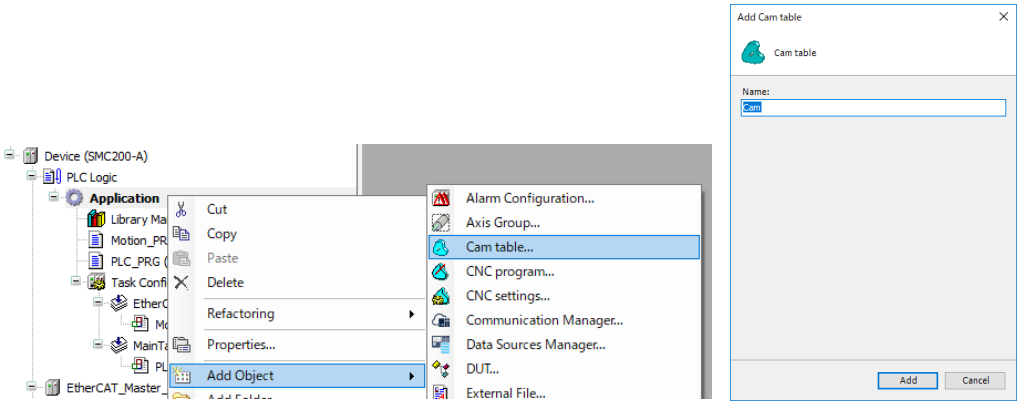


Fig.7.64 Create cam table

2. Set the properties of the cam table.

   Right click on “Cam Table” and select “Properties” → “Cam”, the following window will be displayed. Here you can edit the graph display of the cam table. By setting as follows, the position (0 to 200 mm) of the slave axis is drawn with respect to the master axis position (0 to 110 mm).

   If the option “periodic” is set, the end point of the cam table can be set to 200 mm. This also guarantees, that First (velocity) and second (acceleration) derivation in the starting point and the end point of the cam table match. So, a continuous movement of the slave axis is reached.

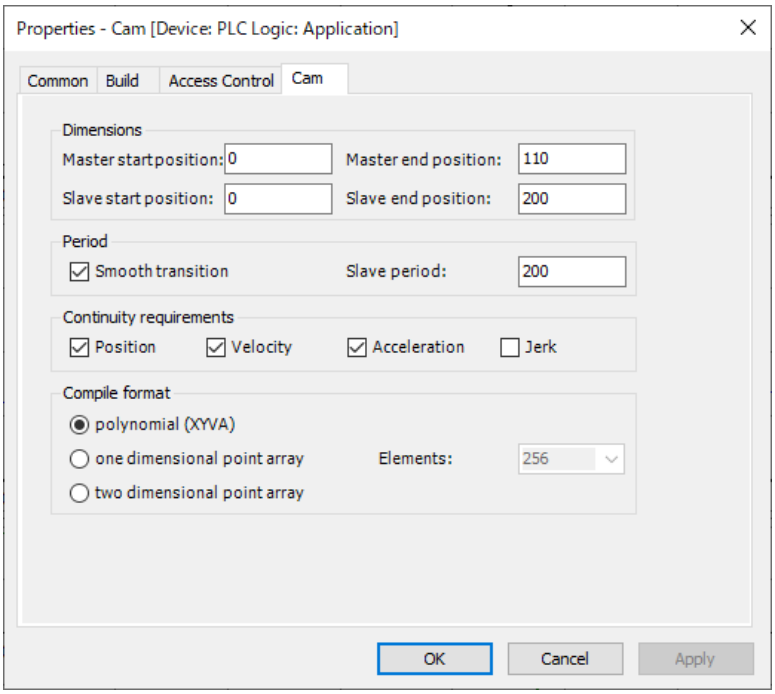


Fig.7.65 Poperties of the cam table

3. Edit the cam table.

The figure below shows the relationship between the master axis and the slave axis on a straight line. The relationship between this master axis and slave axis is defined in the cam table.

In the sample program, the slave axis must be cut to the correct position according to the master axis. Therefore, you need to add a straight line to the cam table.

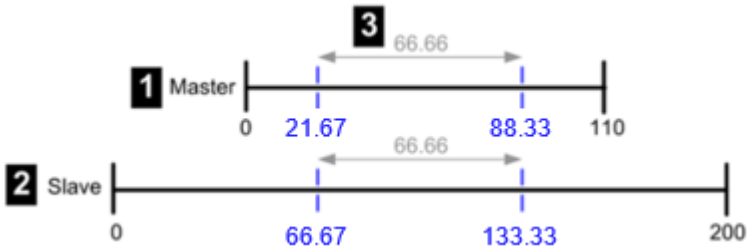


Fig 7.66 Master axis and slave axis positions represented on a straight line

1 ··· Position of the master axis	2 ··· Position of the slave axis
3 ··· Range in which both axes must be synchronous	

The horizontal axis is the master axis, and the vertical axis is the slave axis. The graph shows position, speed, acceleration, jerk in order from the top. In this “cam” tab, you can edit points by dragging. In this “Cam Table” tab, you can edit points by inputting a numerical value.

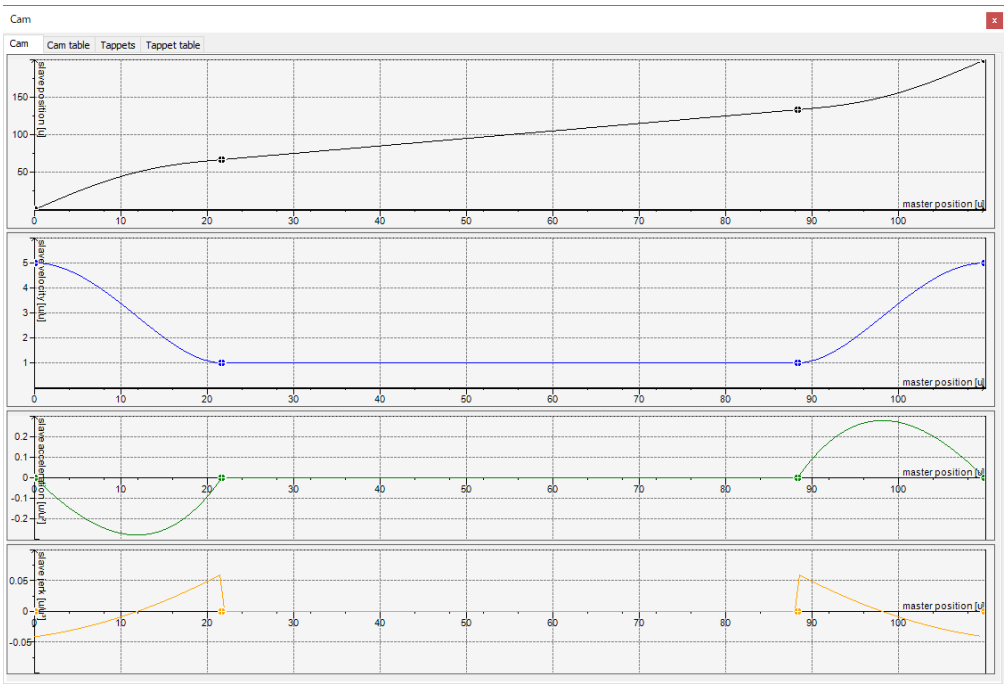


Fig.7.67 Cam Table Waveform

To add a pointer, click ⊕ in the “Cam Table” tab. To delete a pointer, click 🗑 in the “Cam Table” tab.

Select ⊕ twice on the "Cam table" tab, and set as follows.

Cam | Cam table | Tappets | Tappet table

	X	Y	V	A	J	Segment Type	min(Position)	max(Position)	max(|Velocity|)	max(|Acceleration|)
	0	0	5	0	0					
⊕						Poly5	0	66.665999999999954	5	0.279450616549047
🗑	21.666	66.666	1	0	0					
⊕						Line	66.666	133.333	1	0
🗑	88.333	133.333	1	0	0					
⊕						Poly5	133.333	200	5.0000000000000053	0.27943155902290451
	110	200	5	0	0					

Fig 7.68 Cam table settings

A maladjusted velocity in position 0.0 would cause the slave axis to accelerate and decelerate between two periods.

7.7.2.5. Sample program

The list of variables used in the sample program is shown below.

【Declaration section】

```
RotatoryKnife_Power:        MC_Power;            //For servo on/off control of rotaryKnife
Conveyor_Power     :        MC_Power;            // For servo on/off control of Conveyor
RotatoryKnife_Home:         IoSanyoDevice.SanHome;      // /For homing of rotaryKnife
Conveyor_Home      :        IoSanyoDevice.SanHome;      // For homing of Conveyor
Conveyor_Move      :        MC_MoveVelocity;    // For endless motion at a specified
                                                 velocity of Conveyor
CamTableSelect     :        MC_CamTableSelect; //Cam table selection FB
CamIn              :        MC_CamIn;           //Cam synchronous FB
MainStep                    :          INT;      //Main operation step management
                                                 variable
```

In this program, separate FB execution and flag control are described. Write the execution part of FB at the top of the program.

【Implementation section】

```
RotatoryKnife_Power(Axis:= RotatoryKnife, Enable:= TRUE);
Conveyor_Power(Axis:= Conveyor, Enable:= TRUE);
RotatoryKnife_Home(Axis:= RotatoryKnife);
Conveyor_Home(Axis:= Conveyor);
Conveyor_Move(Axis:= Conveyor, Velocity:= 100, Acceleration:= 1000, Deceleration:= 1000);
CamTableSelect(Master:= Conveyor, Slave:= RotatoryKnife, Periodic := TRUE, CamTable:= Cam);
CamIn(
        Master:= Conveyor,
        Slave:= RotatoryKnife,
        MasterOffset:= 0,
        SlaveOffset:= 0,
        MasterScaling:= 1,
        SlaveScaling:= 1,
        CamTableID:= CamTableSelect.CamTableID,
        VelocityDiff:= 100,
        Acceleration:= 500,
        Deceleration:= 500);
```

Describe the flag control part below the execution part.

【Implementation section】

```
CASE MainStep OF
 0 : (* Initialization *)
    RotatoryKnife _Power.bDriveStart := TRUE;
    RotatoryKnife _Power.bRegulatorOn := TRUE;
    Conveyor_Power.bDriveStart := TRUE;
    Conveyor_Power.bRegulatorOn := TRUE;
    IF RotatoryKnife _Power.Status AND Conveyor_Power.Status THEN
      MainStep := 1;
    END_IF

 1 : (* Homing *)
    RotatoryKnife _Home.Execute    := TRUE;
    Conveyor_Home.Execute := TRUE;
    IF RotatoryKnife _Home.Done AND Conveyor_Home.Done THEN
      MainStep := 2;
    END_IF

 2 : (*Conveyor start *)
    Conveyor_Move.Execute := TRUE;
    IF Conveyor_Move.InVelocity THEN
      MainStep := 3;
    END_IF

 3 : (* Synchronous control *)
    IF NOT CamTableSelect.Done THEN
      CamTableSelect.Execute := TRUE;
    ELSE
      CamIn.Execute := TRUE;
    END_IF

END_CASE
```

7.7.2.6. Operation check by trace

Test this sample program by trace.

Blue line: Conveyor.fSetPosition

Green line: RotatoryKnife.fSetPosition

Brown line: Conveyor.fSetVelocity

Gray line: RotatoryKnife.fSetVelocity

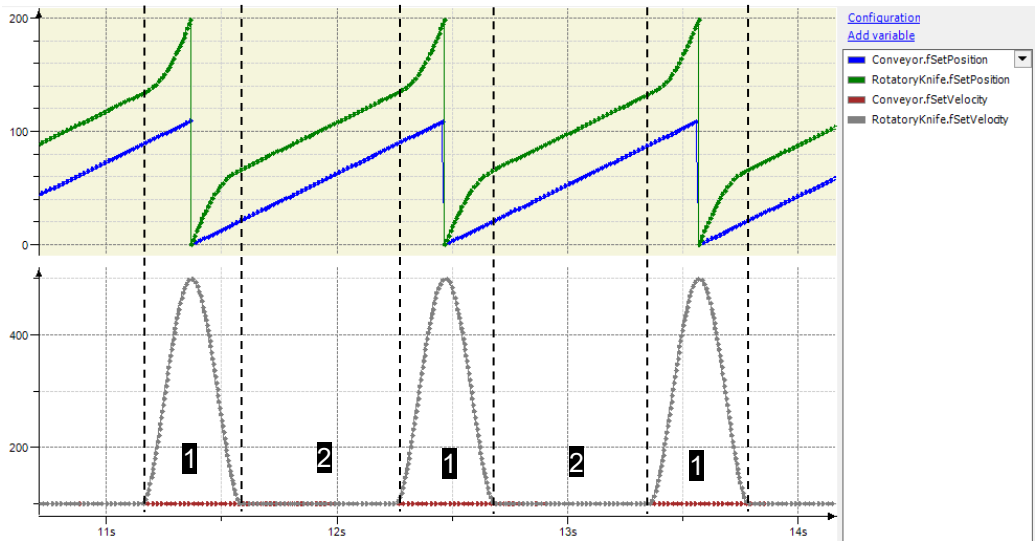


Fig.7.69 Tracing the sample program

No.	Detail
1	The rotary knife moves to the next starting position of the perforating process. In this phase the perforation cylinder runs faster than the conveyor belt.
2	The rotary knife is synchronous to the conveyor belt. In this phase the perforation cylinder must run with the same velocity than the conveyor belt in order not to damage the material.

7.8. CNC control program

Describes how to program CNC control. Use "Motion Standard project" as a template.

This sample program uses the automatic variable declaration function.

For details of the function, please refer to "4.8.4 Input Assistant function".

7.8.1. Sample program summary

The apparatus is the XY table for painting. If the start switch is pushed, the apparatus paints while moving by linear interpolation and circular interpolation.

The movable range is between -100 mm and 1000 mm for both X axis and Y axis.

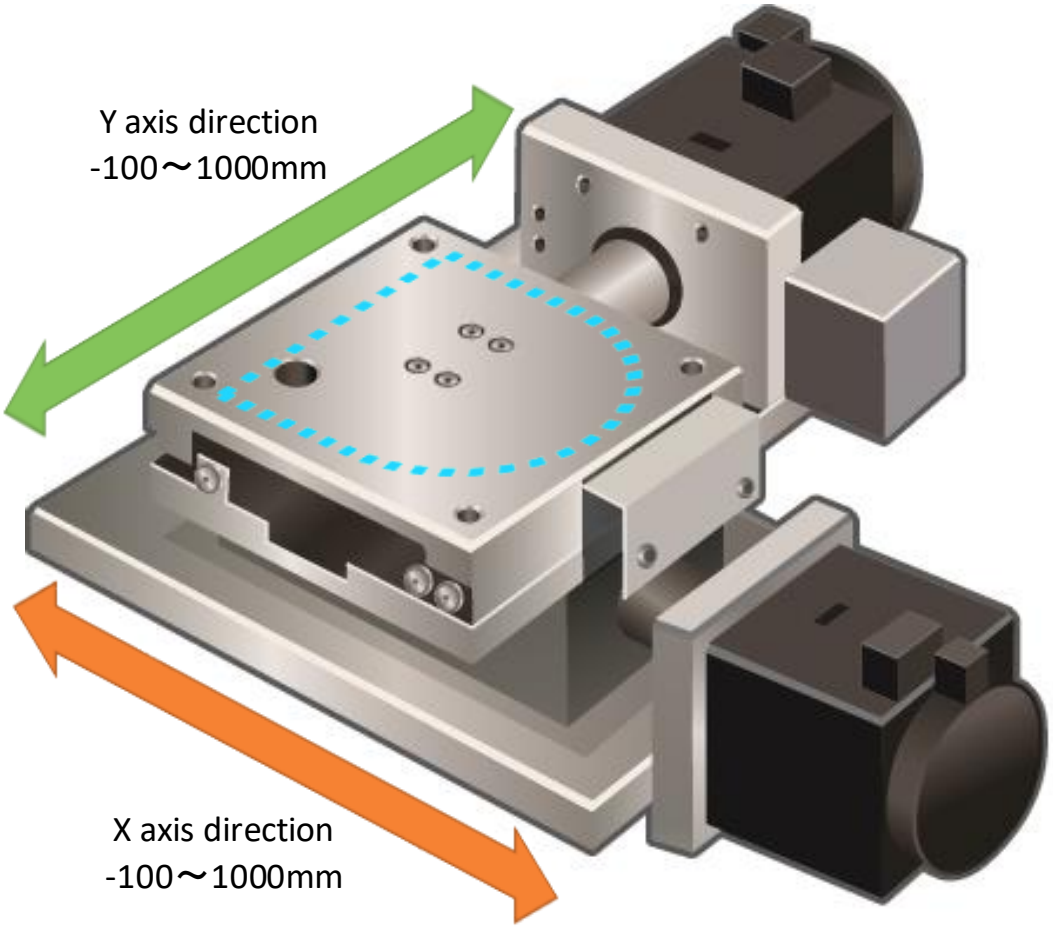


Fig.7.70 Apparatus Overview

7.8.2. CNC Editor

In the CNC editor, you implement complex multidimensional motion in the table editor or text editor according to the CNC language of DIN 66025. There are two ways to edit the CNC program that is manually and import from DXF file.

7.8.2.1. Add and edit CNC program (Manually)

Add and edit the CNC program manually. Follow these steps for add and edit.

1. 1. Add CNC program. Right-click “Application” and select “CNC program ...” from “Add object”.

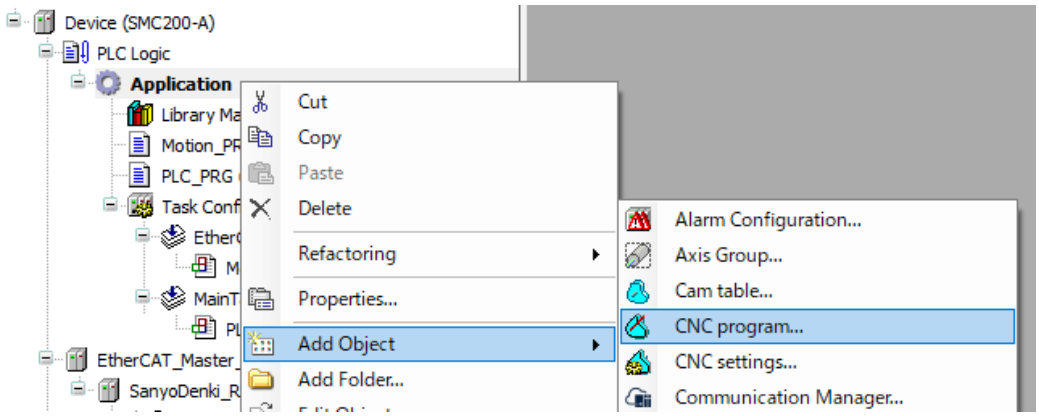


Fig.7.71 Add CNC program

2. The following window will be displayed. Enter an arbitrary name and click “Add”.

   In this sample, name it "CNC".

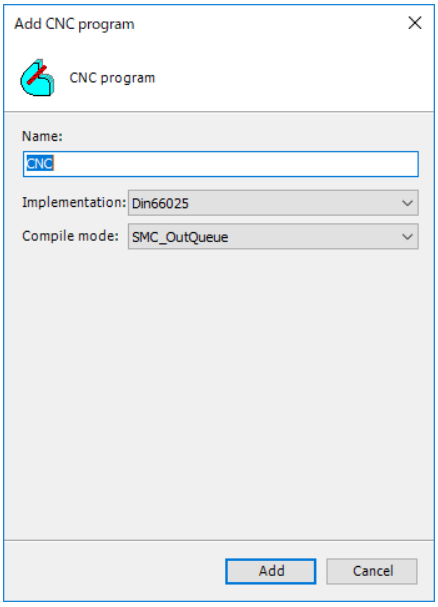


Fig.7.72 Window of adding CNC program

3. The following CNC editor screen is displayed.This screen consists of a main editor, a graphical editor and a tool box.The main editor is the screen for entering the G code. When inputting the G code into the main editor, it outputs the trajectory to the graphical editor.

   The graphical editor is the screen that display the trajectory of movement. You can also draw straight lines and curves in the graphic editor using the toolbox. It is reflected in the G code.

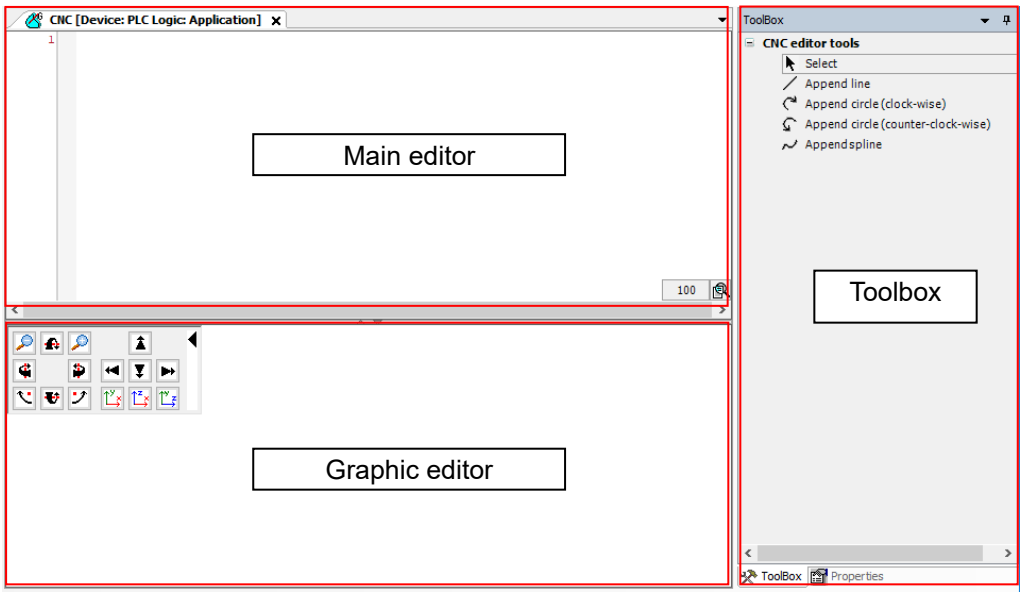


Fig.7.73 CNC editor screen

4. The procedure for creating the program is described below.

   If you use smooth path (G50, G51), you need to add an active FB instance as shown in the next step.

```
N000 F80 E10 E-10      Setting movement parameters (Velocity:80, Acceleration:10, Deceleration 10)
N010 G01 X800 Y0
N020 G51 D100
N030 G01 X800 Y800
N040 G01 X0 Y800
N050 G01 X0 Y0
N060 G50
```

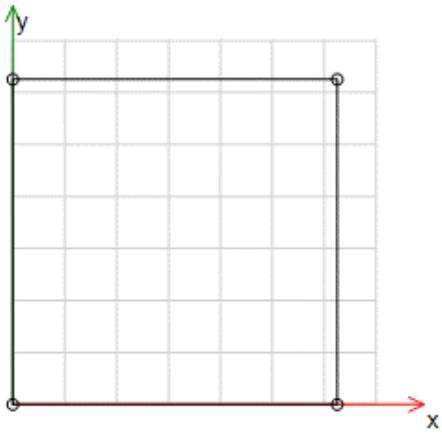


Fig 7.74 CNC example program

In addition, please refer to the help for explanation of each G code.

The explanation of the G code is in the "Travel commands and corresponding path elements" in the principle of Add-ons → CODESYS SoftMotion → CNC → CNC language DIN 66025 → DIN 66025 Fundamentals in the help.

In case of using G code such as smooth path (G50, G51), it is necessary to add corresponding active FB instance.

5. Add active FB instance. Double click on CNC setting. Please select "SMC_SmoothPath" and click ">".

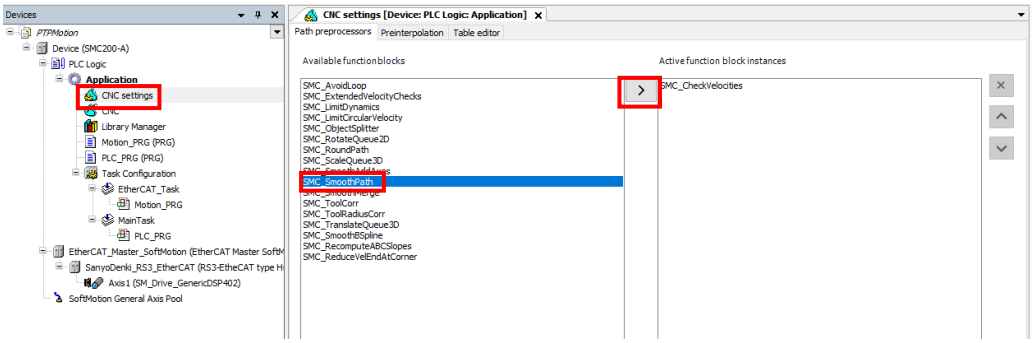
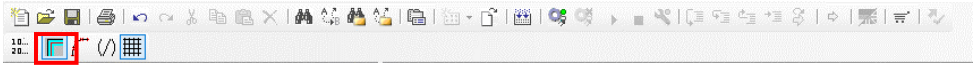


Fig.7.75 CNC settings

6. Please return to the screen of the CNC editor and click  on the upper left of the screen. If the display is active, then the path is displayed with preprocessing and the original path is displayed in light gray in the background.



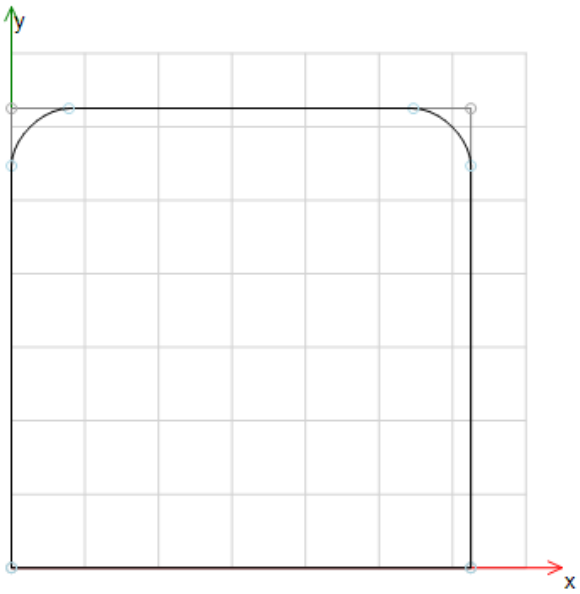
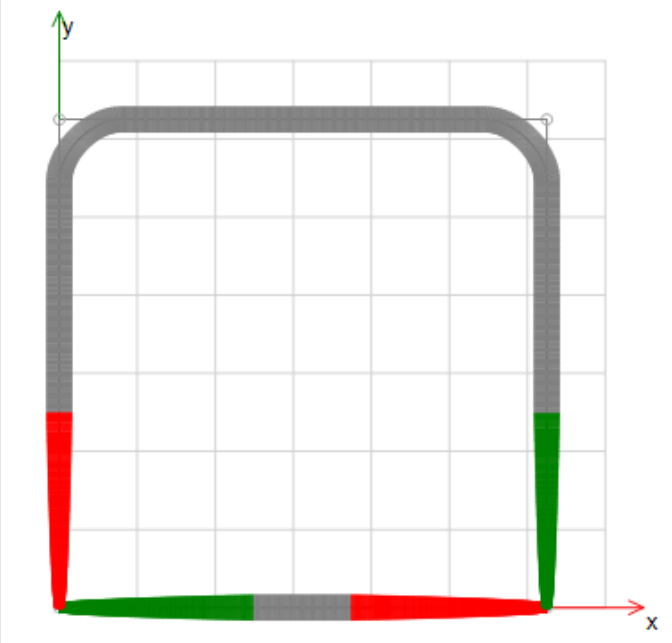


Fig.7.76 CNC editor screen after smooth activation

Please click the icon of  in the upper left of the screen.

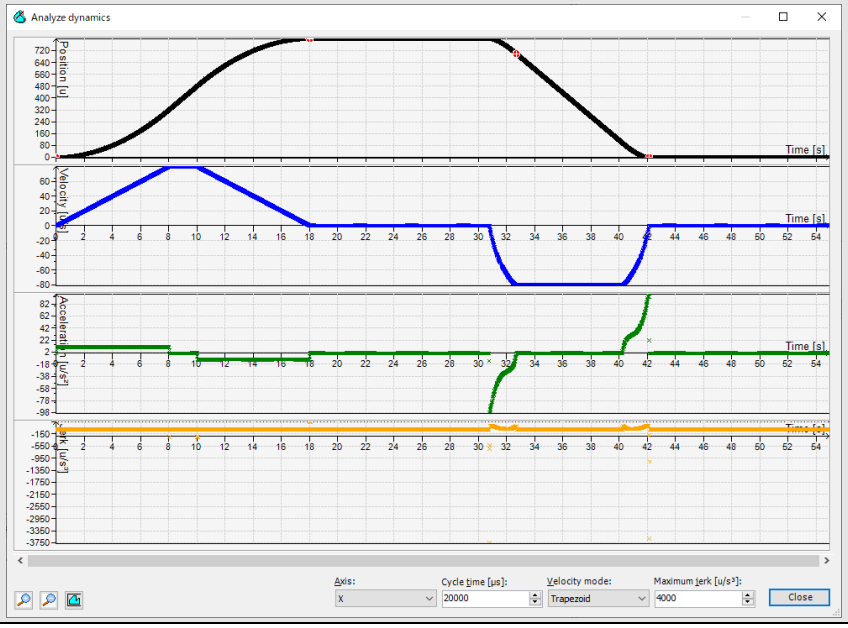
The constant speed section, the acceleration section, and the deceleration section are displayed in the graphical editor.

Red	:	*Interpolator is decelerated.*
Green	:	*Interpolator is accelerated.*
Gray	:	*Interpolator has constant velocity.*



Please click "Analyze dynamics" in "CNC" of the menu bar.

Speed, acceleration/deceleration and jerk can be graphed.



7.8.2.2. Edit CNC program (Import from DXF file)

The CNC program is edited with “Import from DXF file”.DXF Please follow the procedure below.

1. The additional procedure refer to steps 1 and 2 in “7.8.2.1 Add and edit CNC program (Manually)”. This time, name it "CNC_1".

2. Click "CNC" on the menu bar and click "Import from DXF file".

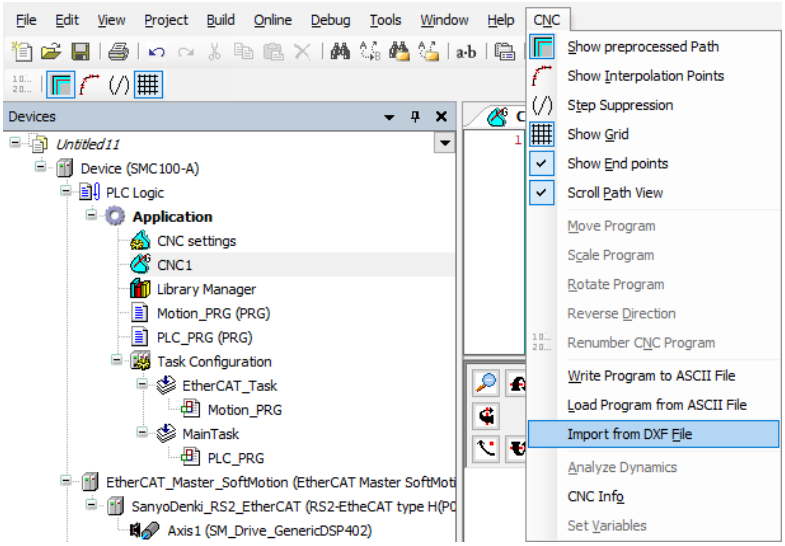


Fig.7.77 Import from DXF file

3. A window for selecting the DXF file is displayed. Please select the DXF file to import.

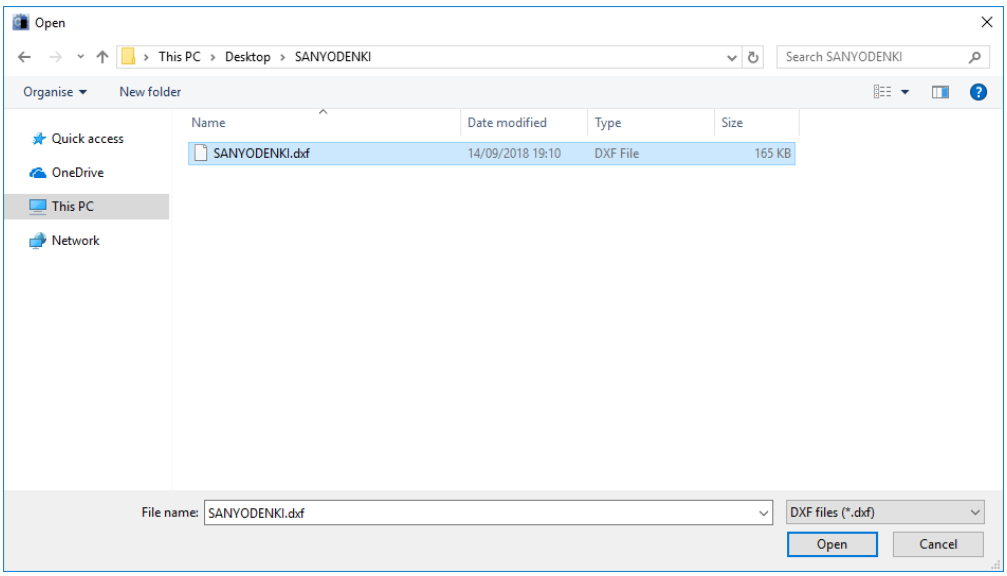


Fig.7.78 Window of File selection

4. After selection, the data is imported . Please click “Import”.

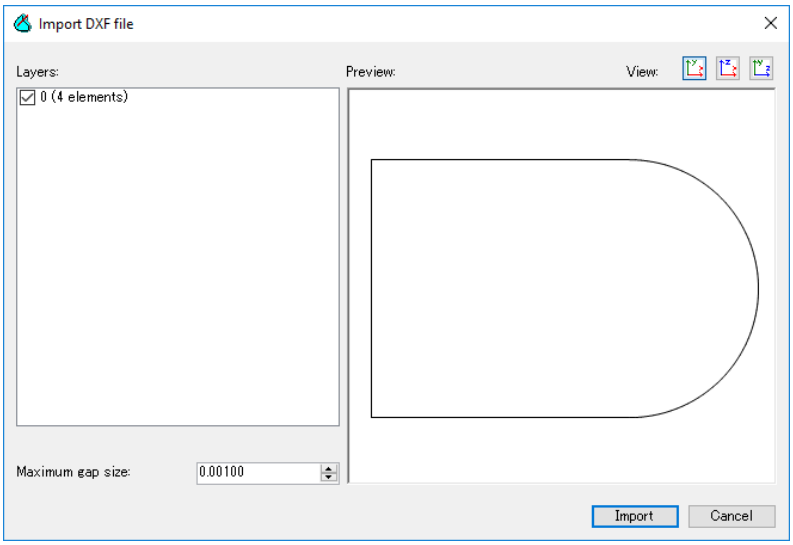


Fig.7.79 Import DXF file

5. When the import is completed, the G code is automatically generated.

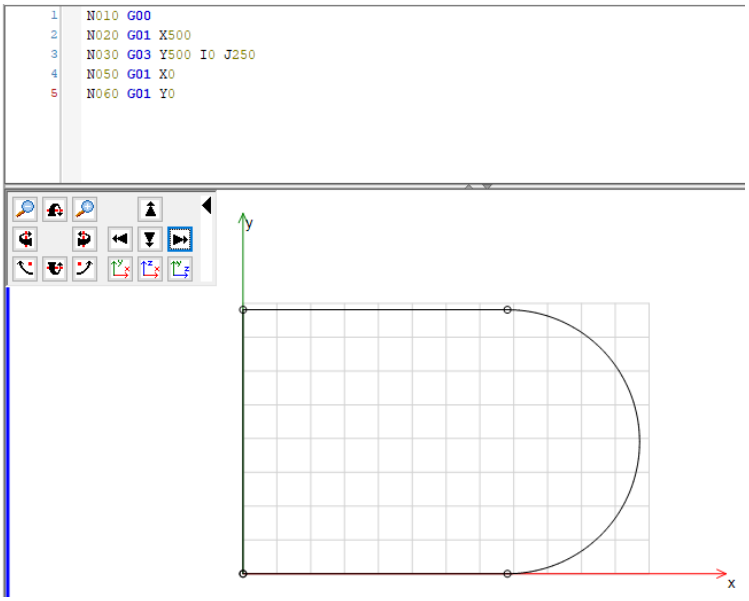


Fig.7.80 Screen after DXF file import

6. When the import is completed, the G code is automatically generated.

```
N000 F80 E800 E-800
N010 G00
N020 G01 X500
N030 G03 Y500 I0 J250
N040 G01 X0
N050 G01 Y0
```

The block number is automatically assigned By clicking in the upper left corner of the screen.

7.8.3. Configuration

7.8.3.1. I/O Mapping

Each digital input is used to start the CNC operation and select the operation pattern. Double-click "Device (SMC200-A)" and select "Device I/O Mapping". After that, set the variable name as shown in the figure below.

Variable	Mapping	Channel	Address	Type	Unit	Description
		DI0~7	%IB38	BYTE		
bStart		Bit0	%IX38.0	BOOL		
bSelect		Bit1	%IX38.1	BOOL		

Fig.7.81 I/O Mapping

7.8.3.2. EtherCAT master setting

In the sample program, the communication cycle of EtherCAT communication is set to 4msec.Double-click "EtherCAT_Master_SoftMotion" and select "General". Then set the Cycle time to 4000 as shown in the figure below.

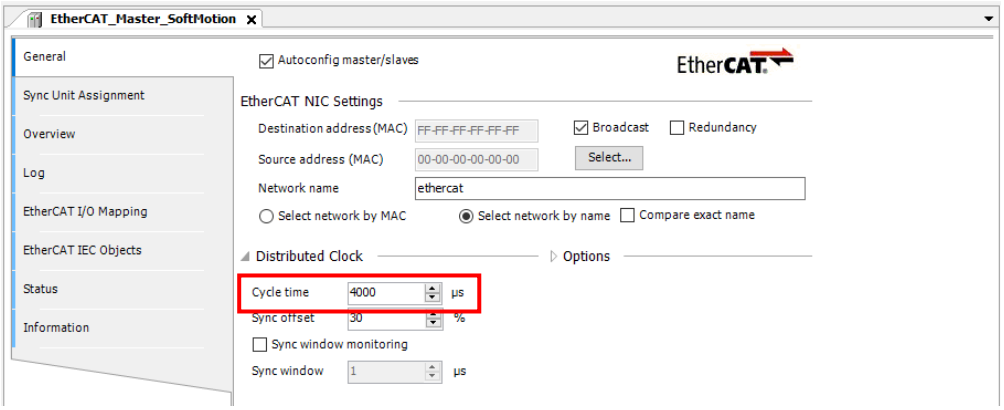


Fig.7.82 EtherCAT master setting

7.8.3.3. Axis setting

The project consists of “AxisX” for moving to the X axis and “Axis Y” for moving to the Y axis. Each axis is attached to a 6 mm lead ball screw. And it is finite axis. The movable range is between -100 mm and 1000 mm for both X axis and Y axis. The configurations of AxisX and AxisY are identical.

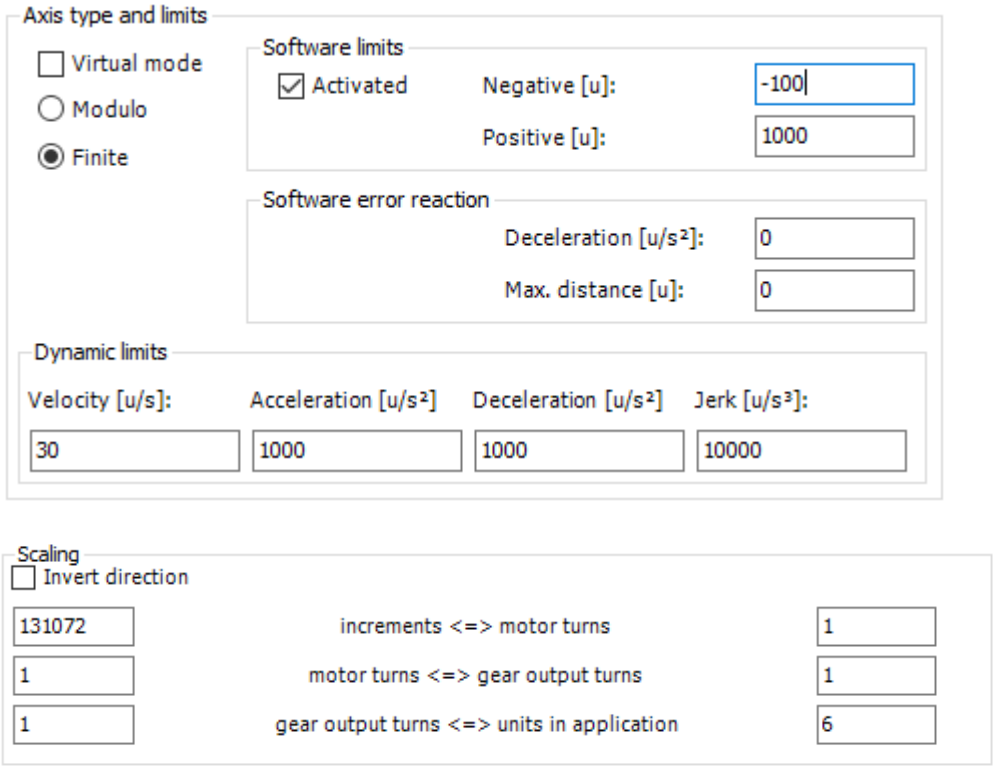


Fig.7.83 Setting of linear axis

7.8.4. Sample program

Create a sample program. First, I will explain the CFC basic operation. Then we show the whole sample program.

For details of FB in the program, refers to “9.2.4 Function block for CNC control “.

1. Please add POU (“CNCTest”). And the description language should be CFC. The basic screen of CFC is as follows.

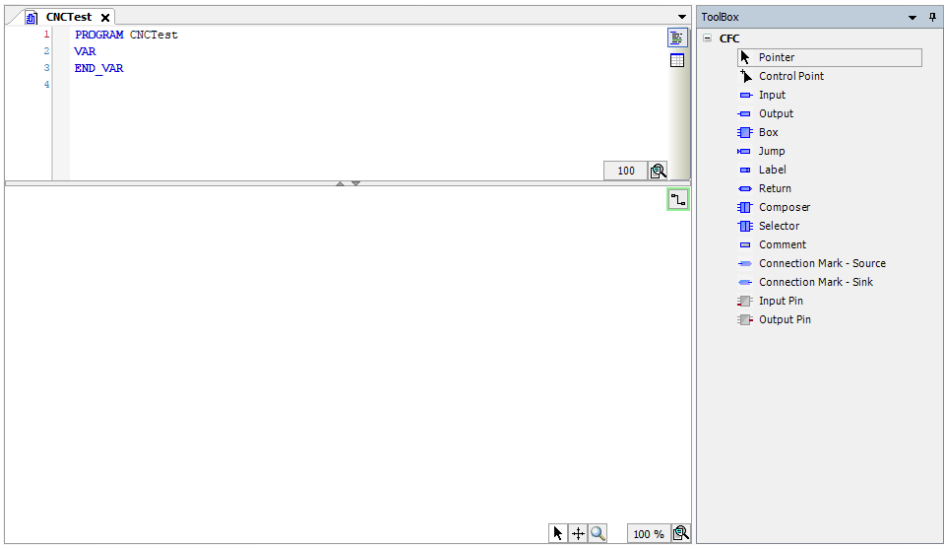


Fig.7.84 CFC basic screen

2. Please select the box from the tool box on the right and click on the mounting section.

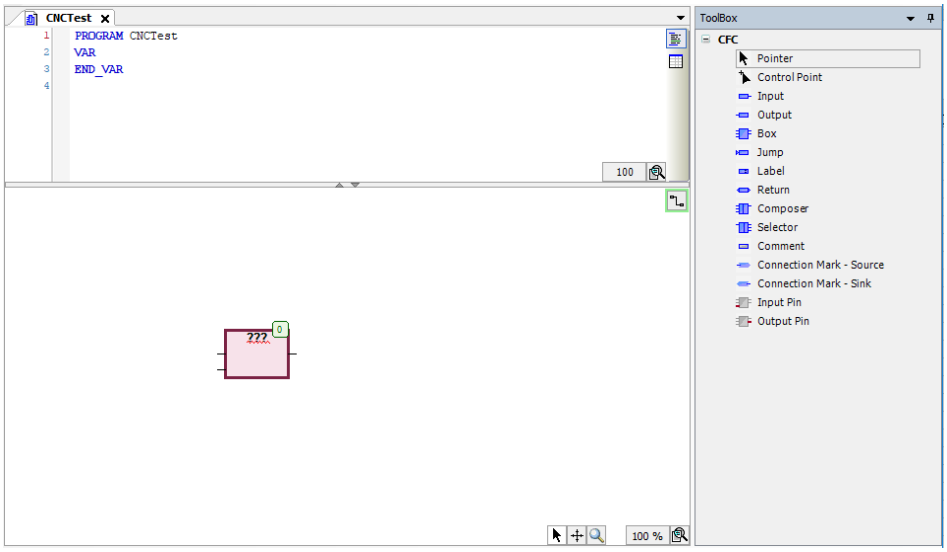


Fig.7.85 Add box

3. Please press F2 key. The input assistant window will be displayed. Select “MC_Power” and click “OK”.

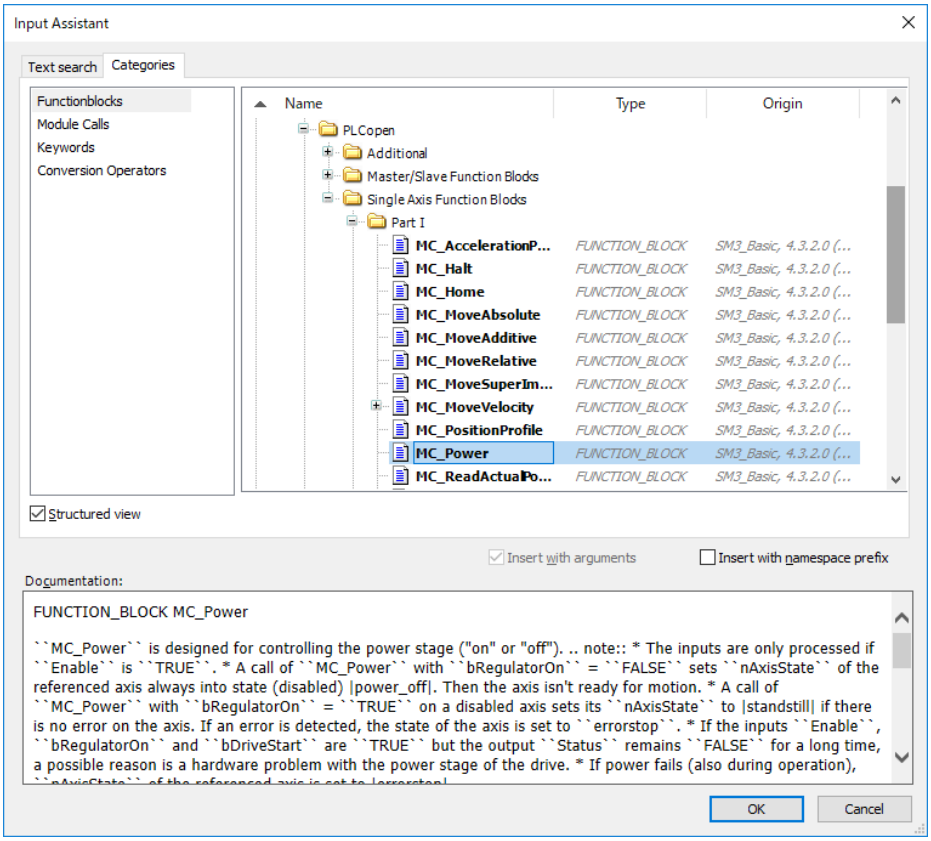


Fig.7.86 Input Assistant

4. The display changes to the following display. Please enter an arbitrary name and press the enter key.

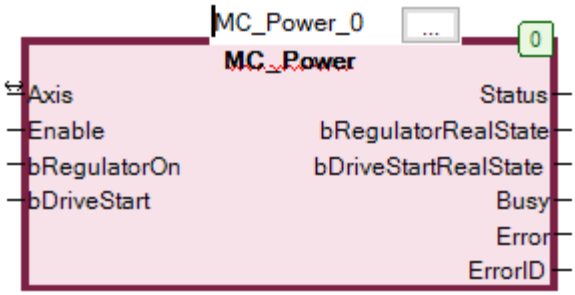


Fig.7.87 Entering name into FB

5. The automatic declare window will be displayed, please click “OK”.

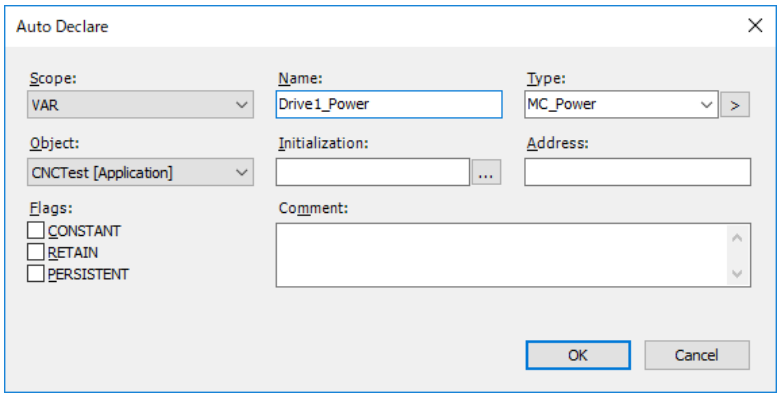


Fig.7.88 The automatic declare window

6. Please select “input” and click on the mounting part.

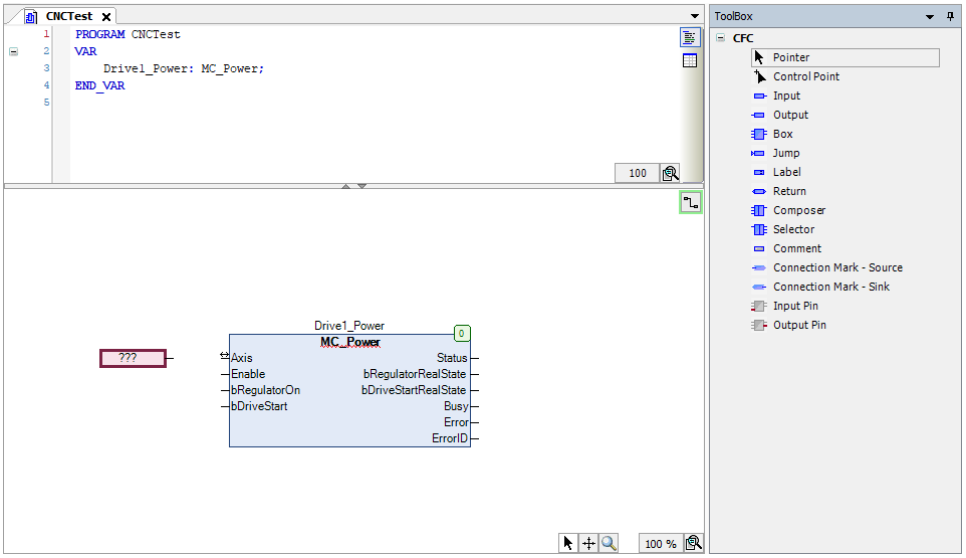


Fig.7.89 Add input

7. After clicking “???”, set the name of the axis (here, AxisX).

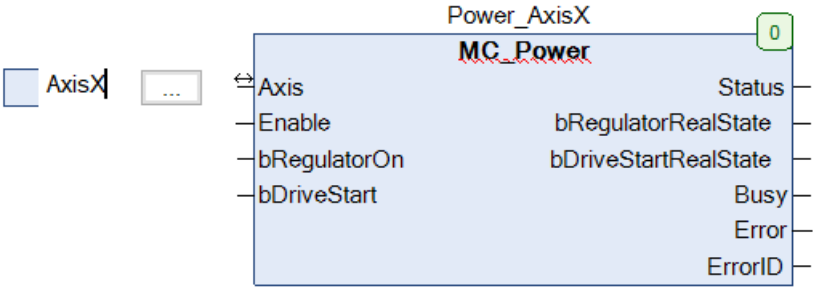


Fig.7.90 Assignment of axes

8. Drag a connecting line from the output of the Input element to the input of the Box element.

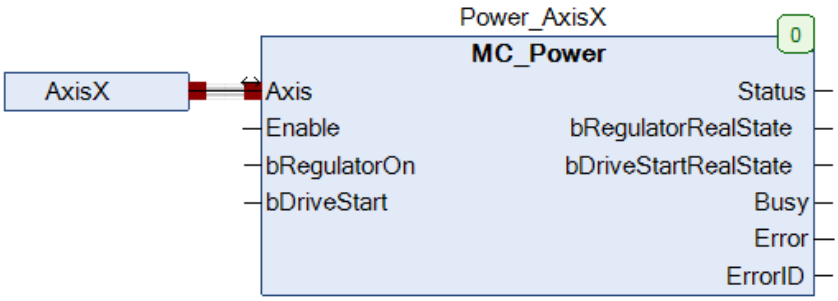


Fig.7.91 Connecting between input and box

The basic operation of CFC is over

9. The sample program for CNC control is shown below. Please create a program.

The list of variables used in the sample program is shown below.

【Declaration section】

```
Power_AxisX, Power_AxisY   : MC_Power;            //For servo on/off control of axisX and AxisY
Home_AxisX, Home_AxisY   : IoSanyoDevice.SanHome;     //For homing of axisX and AxisY
Interpolator     : SMC_Interpolator;        // convert a continuous path into discrete path position points taking
TRAFOGantry2     : SMC_TRAFO_Gantry2;        // Reverse transformation FB
// Forward transformation FB，Use for visualization purpose
TRAFOFGantry2     : SMC_TRAFOF_Gantry2;
// This FB writes the set position for AxisX and AxisY
ControlAxisByPos_AxisX, ControlAxisByPos_AxisY : SMC_ControlAxisByPos;
```

The execution part is described below.

【Implementation section】

CNC operation is started by turning on "bStart".

You can select the CNC table with "bSelect".
TRUE：CNC_1
FALSE：CNC

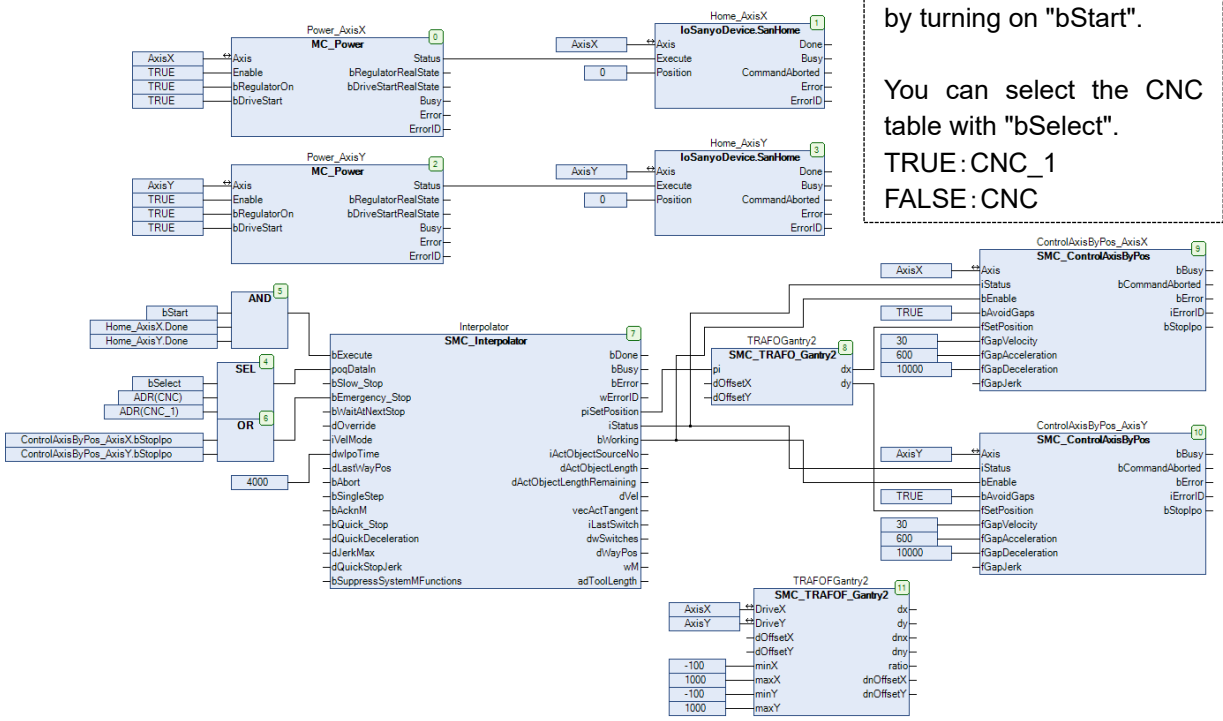


Fig.7.92 CNC example program

10. Assign the sample program for CNC control to the task.Double-click "EtherCAT_Task" on the device tree, and select "CNCTest" created from "Add Call".

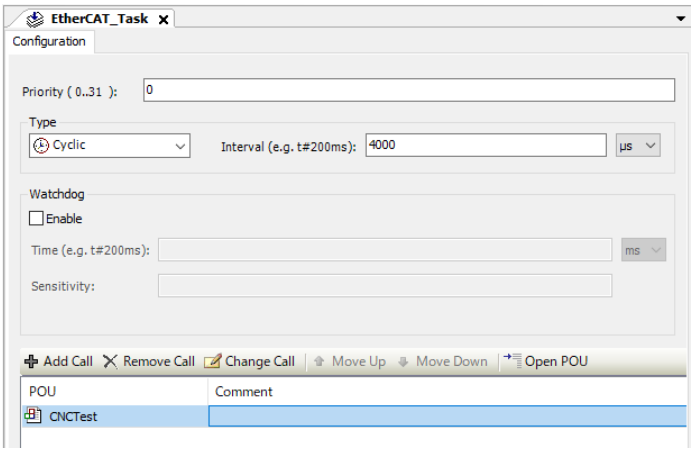


Fig.7.93 Assigning Tasks

7.8.5. Operation check by visualization

Check the operation of the sample program with visualization and trace.

The screen configuration of visualization is described below. Please set according to the following procedure.

Fig 7.94 Element of visualization

1. The setting of Element is described below. Please make settings.

Item	Detail
Element	SMC_VISU_Gantry2（Tag: SM3_CNC）
References —SM3_CNC.SMC_Visu_Gantry2 —m_Input_SMC_TrafoF_Gantry	CNCTest.TRAFOFGantry2

2. After setting, log in and check the operation.

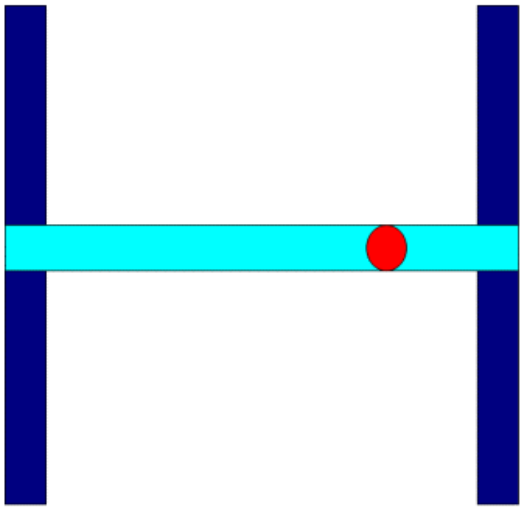


Fig 7.95 Visualization screen when online

7.8.6. Operation check by trace

In case of “CNC” created manually, the trace during program execution is as follows.

Blue line: bStart

Green line: AxisX.fSetPosition

Brown line: AxisY.fSetPosition

Gray line: AxisX.fSetVelocity

Light blue line: AxisY.fSetVelocity

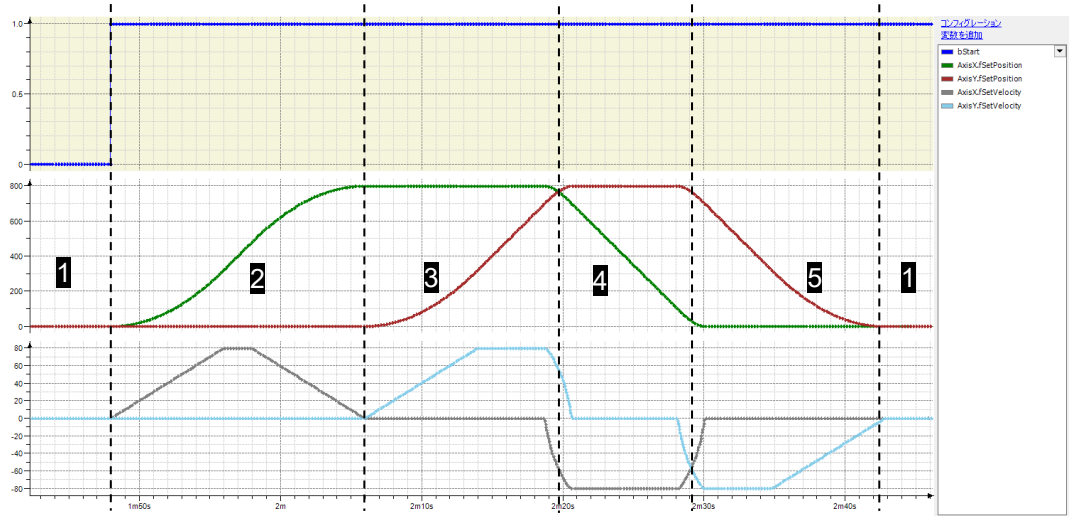


Fig.7.96 Tracing the sample program (CNC)

No.	Detail
1	Trigger(bStart) wait
2	Movement by “N010 G01 X800 Y0”
3	Movement by “N030 G01 X800 Y800”
4	Movement by “N040 G01 X0 Y800”
5	Movement by “N050 G01 X0 Y0”

In case of “CNC_1” imported from DXF file, the trace during program execution is as follows.

Blue line: bStart
Green line: AxisX.fSetPosition
Brown line: AxisY.fSetPosition
Gray line: AxisX.fSetVelocity
Light blue line: AxisY.fSetVelocity

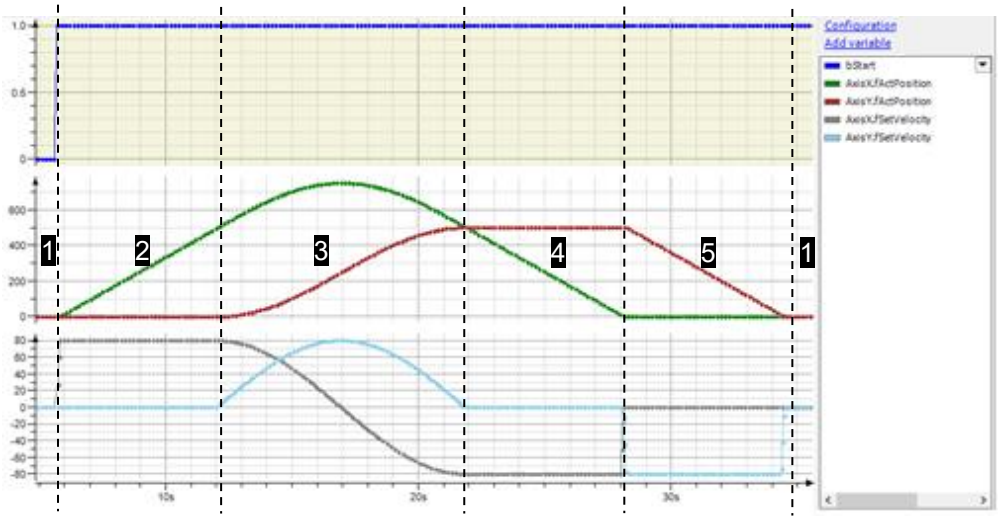


Fig.7.97 Tracing the sample program(CNC_1)

No.	Detail
1	Trigger(bStart) wait
2	Movement by “N010 G01 X800 Y0”
3	Movement by “N030 G01 X800 Y800”
4	Movement by “N040 G01 X0 Y800”
5	Movement by “N050 G01 X0 Y0”

7.9. File control program

With this product, it is possible to read and write files to part of the user area (Directory under /sancontrol) and USB memory and microSD memory (for details on the user area, refer to "6.4.2 Directory structure of user area"

7.9.1. Access path

The path when accessing the user area from the program is described below.

Item	Description in program	Access directory
Default path	File name or './'	/sancontrol
Media path ※1	'$MEDIA$/[Device Name]'	/tmp/media/[Device Name]
Data area path	'$DATA$'	/data
Status report storage path	'$REPORT$'	/report
Other than those above	Not accessible	

※1 The device name of USB memory is output to the log.

[Device Name] is USB: "usbxp1", MicroSD: "microsd0p1".

⇒x is a numeric value. Example: usb0p1

⇒The "p1" part is the partition number.

Unexpected behavior may occur if memory is removed while being accessed from the S200. Therefore, when removing the memory, make sure that the program is not performing file control.

7.9.2. String literal

A string literal is a constant that indicates a string written in a program. A single-byte character string is expressed by sandwiching single quotations (') in SANMOTION C.In single-byte strings, the following specifications exist for the combination of characters following the dollar sign ($).

String	Details
'$$'	Dollar sign ($)
'$' '	Single quotation (')
'$R'	Carriage Return (CR)
'$L' ,' $N'	Line feed (LF)
'$T'	Tab

The path to the media contains a dollar sign. Therefore, when setting the path on the program, it is described as '$$MEDIA$$/...'.

Since the status report storage path starts with 'R' and is recognized as CR, it must be described as follows.

strSample := UTF8#'$$REPORT$$/....'

7.9.3. Sample program summary

Create a function to output log to USB memory and microSD memory as a sample program. The log output function is a function that outputs the current time and the variable value at that point to USB in the format of [DT # YYYY-MM-DD-hh: mm: ss: variable value].

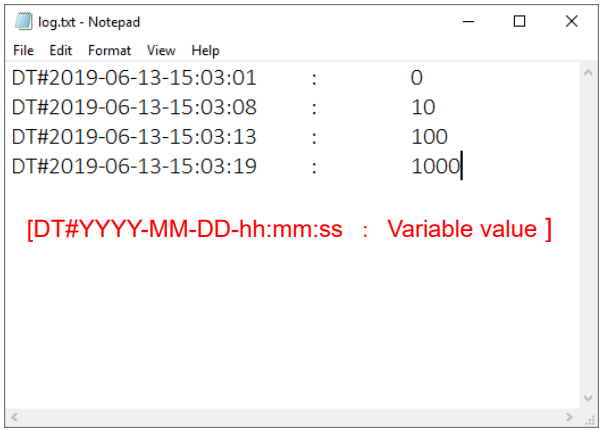


Fig 7.98 Log output result

7.9.4. Sample program

7.9.4.1. Create log output function

Create a function. Please add the following library.

【Used library】

Library Name	Purpose
SysFile	To open, close and write files
SysTime23	To get the current time in local time
CAA DTUtil Extern	To combine acquired time to DT type

【Declaration section】

```
FUNCTION TestLog : BOOL
VAR_INPUT
        FileName        : STRING;//Fail name
        Data            : STRING;// Variable value
END_VAR
VAR
        SystemTime      : SysTime64;
        LocalTime       : SystemTimeDate;
        dtLocalTime     : DT;
        GetLocalTime    : CurTimeEx;
        FileHandle      : sysFile.RTS_IEC_HANDLE;
        WriteData       : STRING;
END_VAR
```

【Implementation section】

```
(* Get current time in local time *)
GetLocalTime(SystemTime:= SystemTime, TimeDate:= LocalTime);
dtLocalTime := DTU.DTConcat(uiYear:= LocalTime.Year, uiMonth:= LocalTime.Month, uiDay:= LocalTime.Day, uiHour:=
LocalTime.Hour, uiMinute:= LocalTime.Minute, uiSecond:= LocalTime.Second, peError:= null);

(*Open the file in append mode *)
FileHandle := SysFileOpen(szFile:= FileName, am:= sysFile.AM_APPEND, pResult:= null);

IF FileHandle <> sysFile.RTS_INVALID_HANDLE THEN

        (* Create write data *)
        WriteData := CONCAT(CONCAT(CONCAT(DT_TO_STRING(dtLocalTime), '$T:$T'), Data), '$R$N');

        (* Write to file *)
        SysFileWrite(hFile:= FileHandle, pbyBuffer:= ADR(WriteData), ulSize:= LEN(WriteData), pResult:= null);

        (* Close file *)
        SysFileClose(hFile:= FileHandle);

        (* Return write complete *)
        TestLog := TRUE;
END_IF
```

7.9.4.2. Log output function usage example

【How to use】

By setting the following in any PRG, log output can be performed when xWriteLog is set to TRUE.

```
IF xWriteLog THEN
  // for USB
  TestLog(FileName:= '$$MEDIA$$/usb0p1/log.txt', Data:= INT_TO_STRING(Parameter));
  // for microSD
  TestLog(FileName:= '$$MEDIA$$/microsd0p1/log.txt', Data:= INT_TO_STRING(Parameter));
  xWriteLog := FALSE;
END_IF
```

7.10. Serial control program

Serial communication is a communication method that sends communication data one bit at a time. Serial communication has the advantages of low cost and resistance to noise.

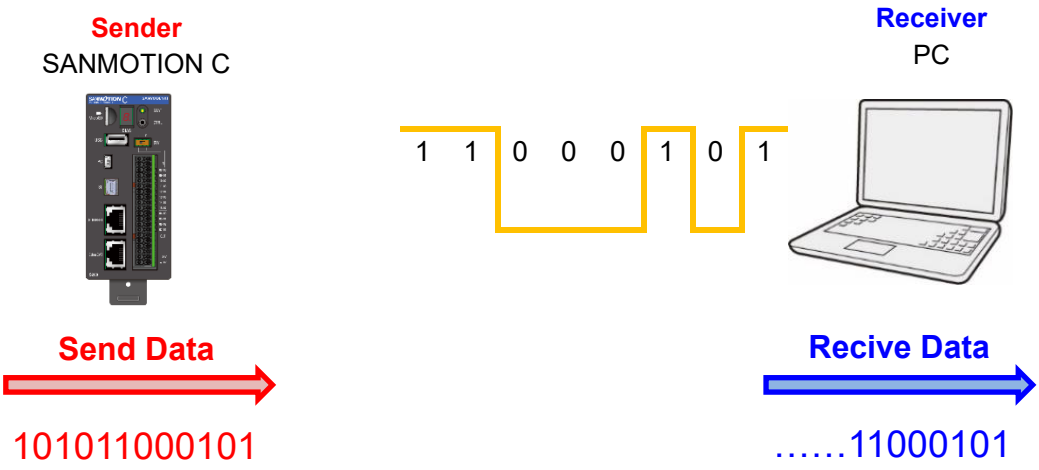
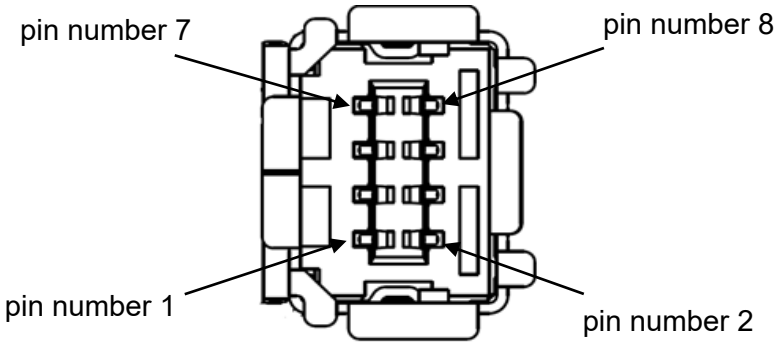


Fig 7.99 Outline of serial communication

The specifications of serial communication of SMC200 are described below.

Item	Detail
Interface name	SI
Connector	Made by TE Connectivity Industrial Mini I/O
Communication standard	RS-485
Baud rate [bps]	4800～115200
Pin assignment	

Signal name	Signal content	Pin number
DATA－	RS485 Send and receive data (－)	6
DATA＋	RS485 Send and receive data (＋)	3
DGND	Ground	4, 8

*Bus Termination:

Bus termination should be done at the ends of the bus (first and last device on the bus). The S200 has a built-in 120Ω termination resistor between pin number 3 and pin 6. When connecting three or more devices to the RS485 interface, wire so that the S200 is at the end of the bus.

7.10.1. Sample program summary

The specifications of the sample program are shown below. Use "PLC Standard project" as a template.

【Interface specification】

Item	Details
Baud rate	115200bps
Bit length	8bit
Parity	None
Stop bit	1bit
Flow control	None

【Serial communication specification】

Item	Details
Environment	Client : Development PC Server : SMC200
Server specification	Echo server
Maximum number of communication data	100byte

7.10.2. Sample program

The following library is used in this sample program. Please add a library.

【Used library】

Library Name	Purpose
Serial Communication	To open, read, and write serial ports

Write the following in PLC_PRG.

【Declaration section】

```
VAR
        ComOpen : COM.Open;
        ComRead: COM.Read;
        ComWrite: COM.Write;
        SrialParameter : ARRAY [1..7] OF COM.PARAMETER := [
        (udiParameterId := COM.CAA_Parameter_Constants.udiPort,          udiValue := 1),
        (udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate,      udiValue := 115200),
        (udiParameterId := COM.CAA_Parameter_Constants.udiParity,        udiValue := COM.PARITY.NONE),
        (udiParameterId := COM.CAA_Parameter_Constants.udiStopBits,      udiValue := COM.STOPBIT.ONESTOPBIT),
        (udiParameterId := COM.CAA_Parameter_Constants.udiTimeout,       udiValue := 0),
        (udiParameterId := COM.CAA_Parameter_Constants.udiByteSize,      udiValue := 8),
        (udiParameterId := COM.CAA_Parameter_Constants.udiBinary,        udiValue := 1)
        ];
        xStart: BOOL;
        iStep: INT;
        ReadData: ARRAY [0..99] OF BYTE;
END_VAR
```

【Implementation section】

```
ComOpen(usiListLength := SIZEOF(SrialParameter)/SIZEOF(COM.PARAMETER), pParameterList := ADR(SrialParameter));
ComWrite(udiTimeOut:= 1000, hCom:= ComOpen.hCom, pBuffer:= ADR(ReadData), szSize:= ComRead.szSize);
ComRead(udiTimeOut:= 1000, hCom:= ComOpen.hCom, pBuffer:= ADR(ReadData), szBuffer:= SIZEOF(ReadData));

CASE iStep OF

 0 : (* Waiting for start *)
    IF   xStart THEN
      iStep := 1;
    END_IF

 1 : (* Serial port open *)
    ComOpen.xExecute := TRUE;
    IF ComOpen.xDone THEN
      iStep := 2;
    ELSIF ComOpen.xError THEN
      iStep := -1;
    END_IF

 2 : (* Read and write data *)
    ComWrite.xExecute := FALSE;
    ComRead.xExecute := TRUE;
    IF ComRead.xDone THEN
      ComRead.xExecute := FALSE;
      IF ComRead.szSize > 0 THEN
        ComWrite.xExecute := TRUE;
      END_IF
    ELSIF ComRead.xError THEN
      iStep := -2;
    END_IF

END_CASE
```

7.11. Socket control program

Socket communication means sending and receiving data between processes. Each process has a number, which is called a port number. By specifying the IP address of the address on the network and the port number that is the process address in the computer, data can be exchanged from the outside to the specified process.

In socket communication immediately after startup, the PLC application may be executed before the network status is set. It is recommended to install a timer or implement a retry function when performing socket communication immediately after startup.

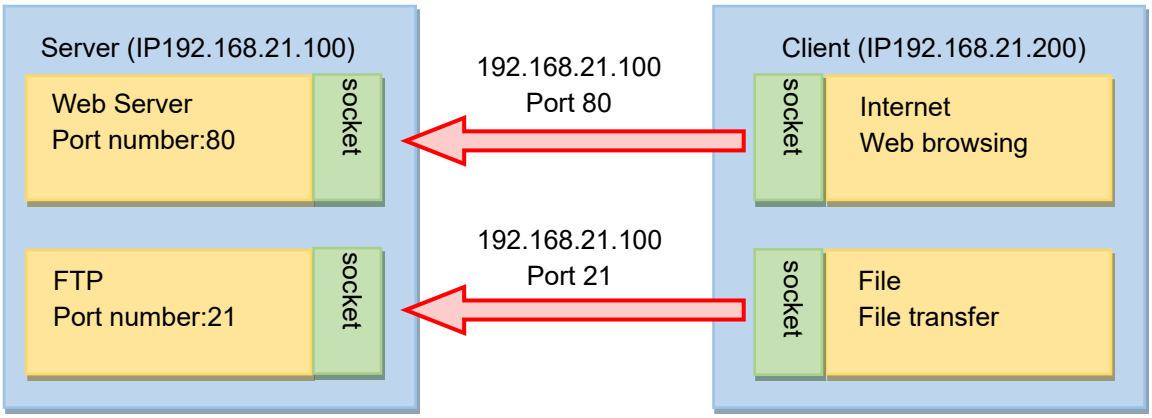


Fig 7.100 Outline of socket communication

7.11.1. Socket type

There are several types of sockets depending on the communication method. The socket corresponds to the session layer when the OSI reference model is associated. TCP and UDP exist in the transport layer one lower than the socket, and these two can be accessed from the socket.

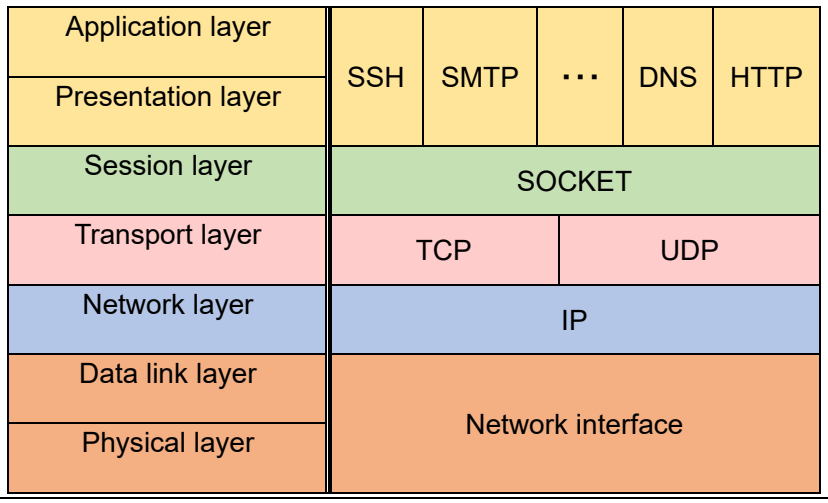


Fig 7.101 OSI reference model

7.11.2. TCP communication

TCP communication provides reliable two-way communication between two systems (one-to-one) on the network. Highly reliable data transfer can be performed, such as acknowledgments and packet sequence checks to retransmit lost packets. However, it is slower than UDP communication because the protocol overhead (usually 20 bytes) is large. The flow from connection to disconnection by TCP communication is described below.

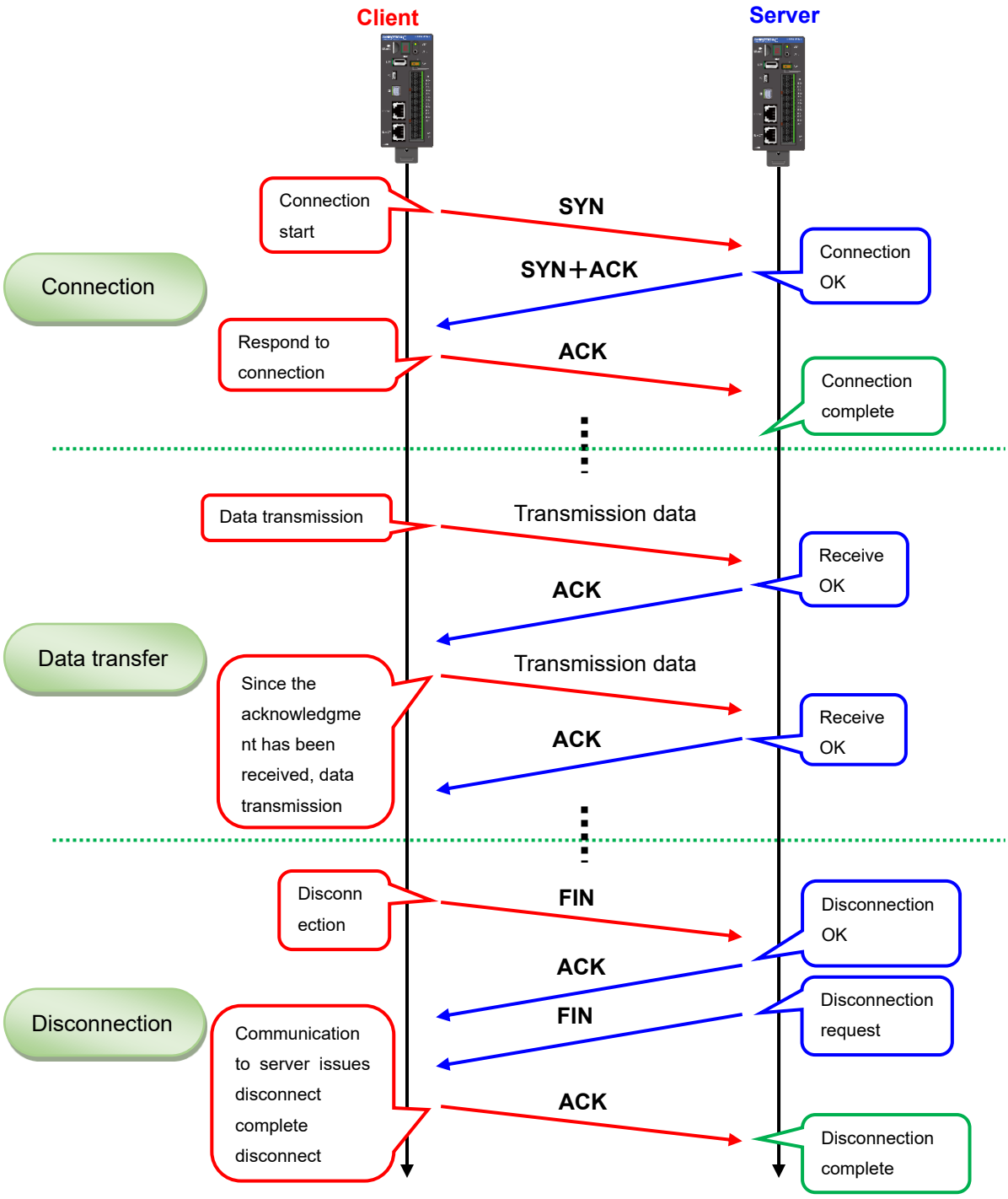


Fig 7.102 TCP communication flow

7.11.3. UDP communication

UDP communication can perform high-speed transfer although it is not reliable compared to TCP communication. Also, since the overhead (8 bytes) is small, it is possible to send and receive a lot of application data. However, since there is no guarantee that a packet will arrive, in the case of packet loss, etc., the application must perform retransmission processing to establish communication, or an application that can tolerate packet loss is required. UDP communication is used by the following applications.

◆ **Broadcast**

TCP communicates on a one-to-one basis, but UDP can communicate with multiple parties.

◆ **Communication requiring real-time capability**

Streaming applications need to send and receive data in real time at high speed.

◆ **Communication that does not require reliability**

Because UDP does not establish a connection, it is not reliable for TCP. However, by not establishing a connection, high-speed data transfer can be performed if communication is frequently performed with a small amount of communication data. TCP needs to exchange 3 packets to establish a connection, which reduces the data transfer rate.

You can also improve reliability with UDP by adding processing that resends even if data is lost.

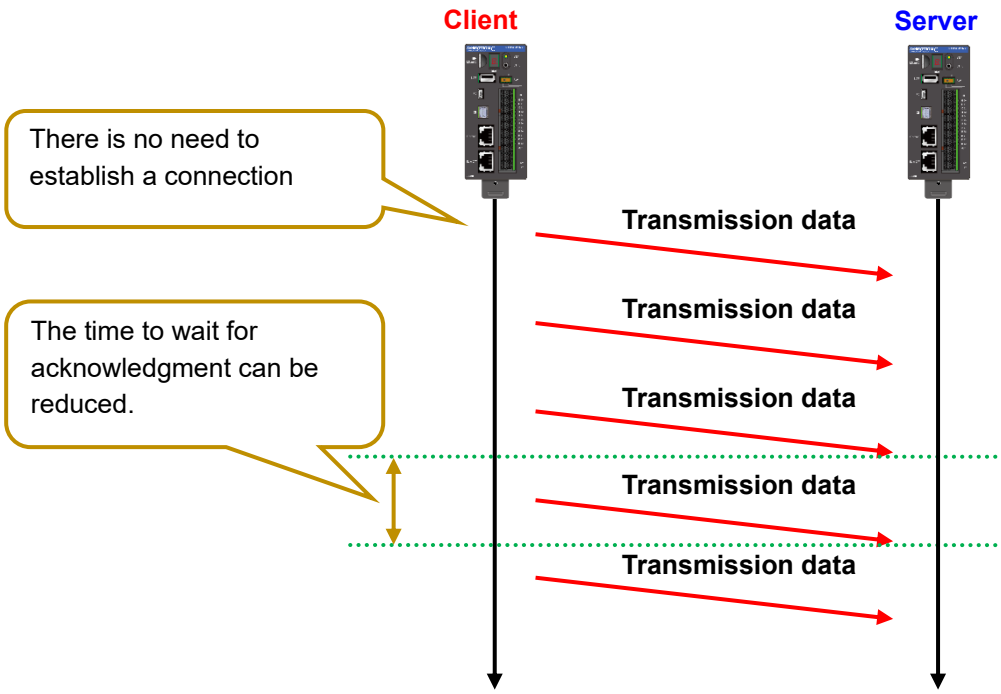


Fig 7.103 UDP communication flow

7.11.4. Sample program summary

The specifications of the sample program are described below. Use "PLC standard project" as a template.

【Communication specification】

項目	詳細
Environment	Client : Development PC Server : SMC200
Communication protocol	TCP/IP （Non-procedure）
Connection form	1:1
Server specification	Echo server
Interface	ethernet (IP : 192.168.21.101)
Port number	60000
Maximum number of communication data	100byte

7.11.5. Sample program

The following library is used in this sample program. Please add a library.

【Used library】

Library Name	Purpose
Network	To open, read, and write Ethernet port

Write the following in PLC_PRG.

【Declaration section】

```
VAR
        IPAddress           : NBS.IP_ADDR := (sAddr:='192.168.21.101');
        TCP_Server          : NBS.TCP_Server :=(ipAddr:=IPAddress, uiPort:= 60000);
        TCP_Connection      : NBS.TCP_Connection;
        TCP_Write           : NBS.TCP_Write;
        TCP_Read            : NBS.TCP_Read;
        RecvData            : ARRAY [0..99] OF BYTE;
        RecvNum             : __UXINT;
END_VAR
```

【Implementation section】

```
TCP_Server(xEnable:= TRUE);
TCP_Connection(xEnable:= TCP_Server.xBusy, hServer:= TCP_Server.hServer);
TCP_Read(xEnable:= TCP_Connection.xActive, hConnection:= TCP_Connection.hConnection, szSize:= SIZEOF(RecvData), pData:= ADR(RecvData));
TCP_Write(xExecute:= TCP_Read.xReady, udiTimeOut:= 1000, hConnection:= TCP_Connection.hConnection, szSize:= TCP_Read.szCount, pData:= ADR(RecvData));
```

7.12. Camera control program

The camera control function saves still images and delivers videos in real time.

CAUTION!

- Always use the function blocks included in this library with a "task" priority of 16 or lower. Unexpected behavior may occur if executed in a "task" with a high priority.
- When using the camera function, set the shortest task cycle to 4 ms or more.
- For an application that constantly monitors stream screens, set the shortest task cycle to 8 ms or more.

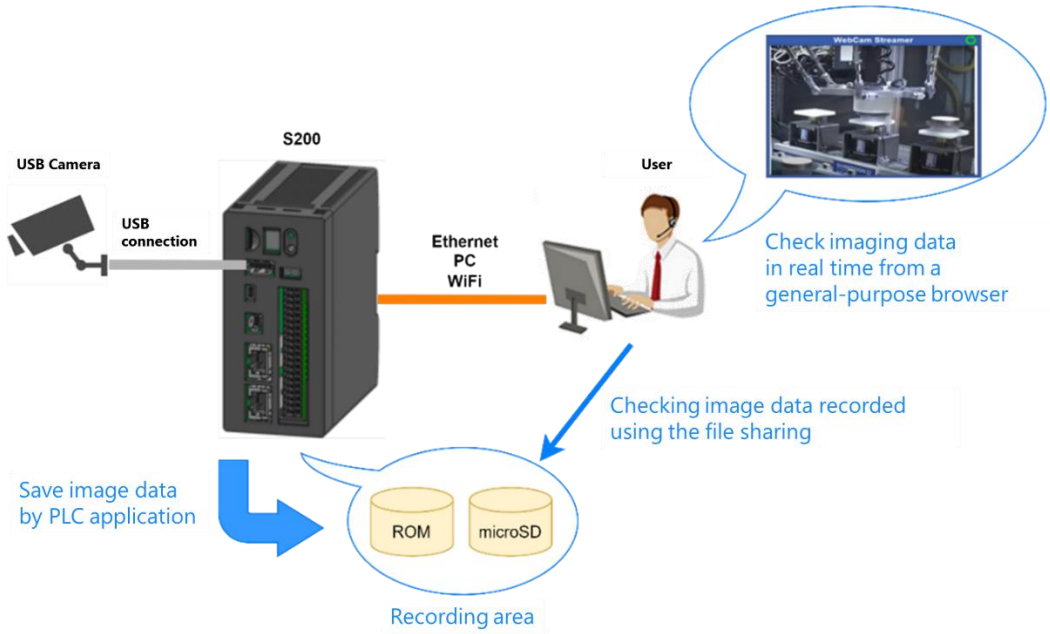


Fig 7.104 Overview of camera control function

7.12.1. Specification

Item	Detail	Note
Supported camera	USB camera (UVC supported)	Depending on the camera, power supply shortage may occur. In that case, please use a USB cable supplied from a separate power supply.
Interface	USB2.0	
Maximum connection number	1	
Effective timing	When a supported camera is inserted	
Streaming port	10443	Only one device can access this port. A 599 error is returned for the second and subsequent machines.
Still image data format	JPEG	
Still image interval when batch saving	100ms	ImagesSave***(FB) saves still images in the specified time range all at once. It means the shortest interval between. still images at that time.
Resolution	640x480	
Frame rate	30fps	

7.12.2. Function block

7.12.2.1. ImageSave

This function block saves still images captured by the connected camera. Saves the still image captured at the next capture event that occurs after the function block is executed to the specified path.

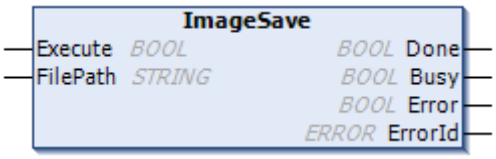


Fig. 7.105 ImageSave

VAR_INPUT		
Execute	BOOL	Save streamed still image on rising edge
FilePath	STRING	Image save destination (If not set, the latest file will be stored in the $DATA$/image folder)
VAR_OUTPUT		
Done	BOOL	Execution completed state
Busy	BOOL	Running state
Error	BOOL	Error condition
ErrorId	ERROR	Error detail

7.12.2.2. ImagesSaveGoingBackInTime

This function block saves all still images from the time rewound by the specified amount of time, based on the time stamp of the still image acquired at the next imaging event that occurs after execution. Maximum rewind time is 10 seconds.

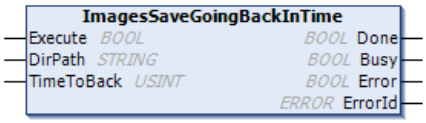


Fig. 7.106 ImagesSaveGoingBackInTime

VAR_INPUT		
Execute	BOOL	Save streamed still image on rising edge
DirPath	STRING	Image save destination (If not set, store in $DATA$/image folder)
TimeToBack	USINT	Rewind time (unit: seconds) Input range: 1 to 10 seconds
VAR_OUTPUT		
Done	BOOL	Execution completed state
Busy	BOOL	Running state
Error	BOOL	Error condition
ErrorId	ERROR	Error detail

7.12.2.3. ImagesSaveTriggerPrePost

This function block saves the still images before and after the specified time based on the time stamp of the still image acquired at the next imaging event that occurs after execution.

Fig. 7.107 ImagesSaveTriggerPrePost

VAR_INPUT		
Execute	BOOL	Save streamed still image on rising edge
DirPath	STRING	Image save destination (If not set, store in $DATA$/image folder)
PrePostTime	USINT	Time before and after the trigger (unit: seconds) Input range: 1 to 5 seconds
VAR_OUTPUT		
Done	BOOL	Execution completed state
Busy	BOOL	Running state
Error	BOOL	Error condition
ErrorId	ERROR	Error detail

7.12.2.4. Error list

Below is a list of errors that occur in this function block.

Error ID	Error name	Detail
0	NO_ERROR	No error occurred
1	ERROR_FILE_PATH	Invalid save destination path
2	ERROR_CAM_NOT_READY	The camera is not ready
3	ERROR_GOBACK_TIME_INVALID	Invalid rewind time
4	ERROR_IMAGE_COPY	Failed to save still image
90	ERROR_INTERNAL	Internal error
99	ERROR_TIMEOUT	Timeout error

7.12.3. Visualization Objects

7.12.3.1. VisuStreamer

VisuStreamer is an object for displaying stream distribution on Webvisu. By setting the URL of the stream server using Webvisu's "Web browser" object, monitoring of stream delivery is realized. The URL that is set is "https://<host name>:10443" and the host name is used in the URL, so if you access from an environment where the host name cannot be resolved (external network, etc.), use this object It can not be used.

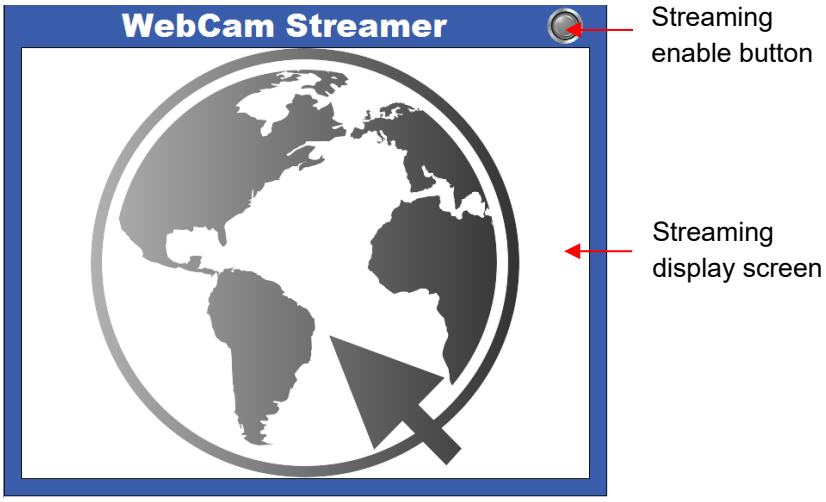


Fig. 7.108 VisuStreamer

The following FBs are also provided as input variables for each object.

- VisuStreamerCtrl (FB)

  This function block provides the URL of the stream server.

VAR_INPUT		
Enable	BOOL	Enable streamed object
VAR_OUTPUT		
none	-	-

7.12.3.2. VisuDispImage

VisuDispImage is an object that updates a still image after a specified amount of time. It is an object that allows you to check the image data even in an environment where host name resolution is not possible, and the CPU load is not as large as VisuStreamer.

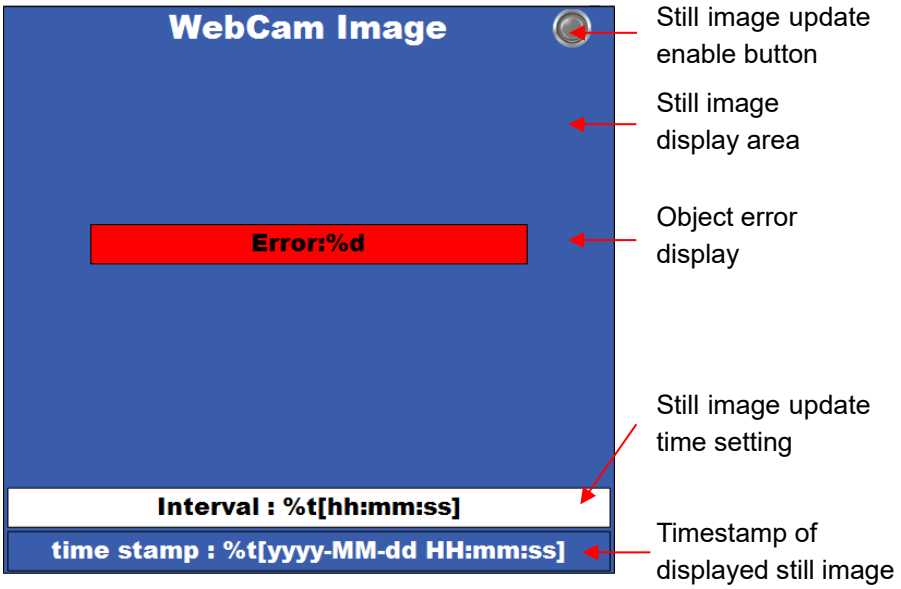


Fig. 7.109 VisuDispImage

The following FBs are also provided as input variables for each object.

- VisuDispImageCtrl (FB)

  This function block enables still image acquisition and timer processing performed by VisuDispImage.

VAR_INPUT		
Enable	BOOL	Enable still image update object
VAR_OUTPUT		
none	-	-

7.12.4. Sample program summary

Below is a sample program that saves a still image to microSD 10 seconds before an error occurred. Use "PLC Standard project" as a template.

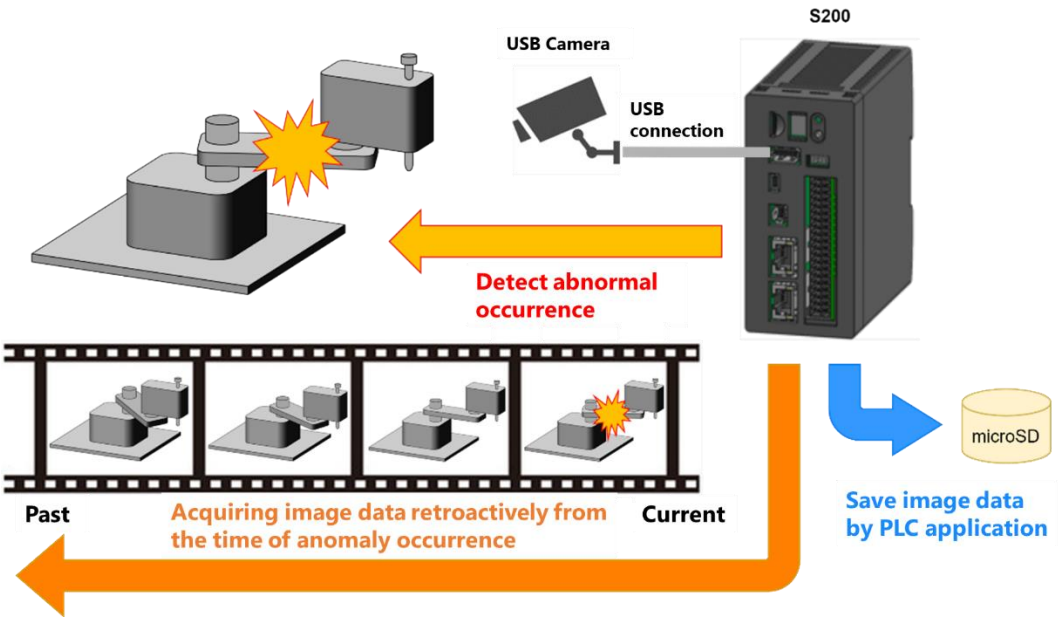


Fig. 7.110 Camera control sample program

7.12.5. Sample program

The sample program uses the following libraries. Please add the library.

【Used library】

Library Name	Purpose
SanCamera	To use the function block that saves still images

Write the following in PLC_PRG.

【Declaration section】

```
VAR
  ImagesSaveGoingBackInTime: SanCamera.ImagesSaveGoingBackInTime;
  AlarmFlg: BOOL;
END_VAR
```

【Implementation section】

```
ImagesSaveGoingBackInTime(Execute:= AlarmFlg, DirPath:= '$$MEDIA$$/microsd0p1', TimeToBack:= 10);
```

Since it is not recommended to run the camera control function block in a real-time task, change the priority of MainTask in the template project to 16 or higher.

7.12.6. Operation check

In the sample program, the stopwatch is imaged so that the rewind time can be grasped quantitatively, and AlarmFlg is set to TRUE when 15 seconds have elapsed.

If the image is captured successfully, a still image from 5 seconds to 15 seconds will be saved as shown below.

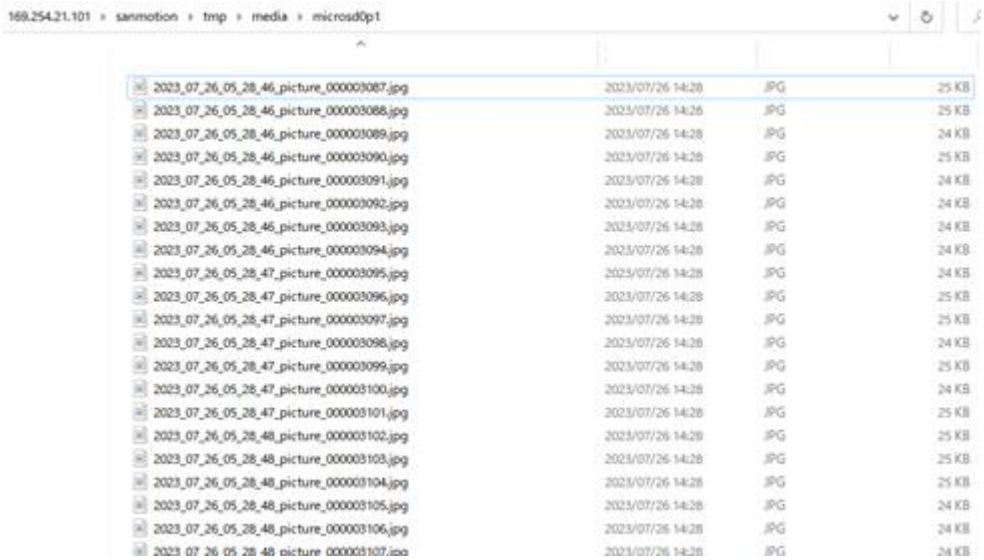


Fig. 7.111 Top directory of microSD after function block execution

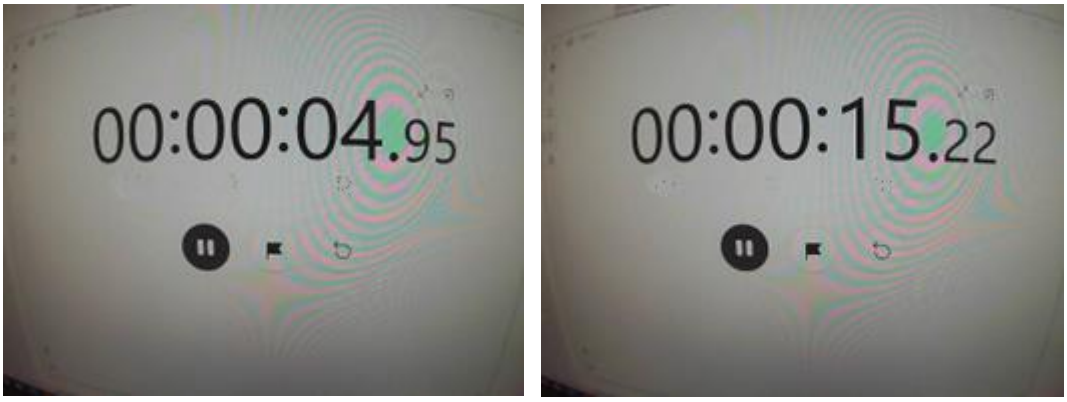


Fig. 7.112 Saved data (left: oldest still image, right: newest still image)

7.13. Mail sending program

The S200 is equipped with a function to send emails via the SMTP server prepared by Sanyo Denki, and emails can be sent using web applications or function blocks. E-mail can also be sent using the SMTP server provided by the customer.

For the SMTP server operated by Sanyo Denki, please refer to "M0021001 SMTP Server Terms of Use" before using.

7.13.1. Email settings via web app

With "SMTP" in the "Communication function" tab of the web application, you can set the parameters of the mail and send a test.

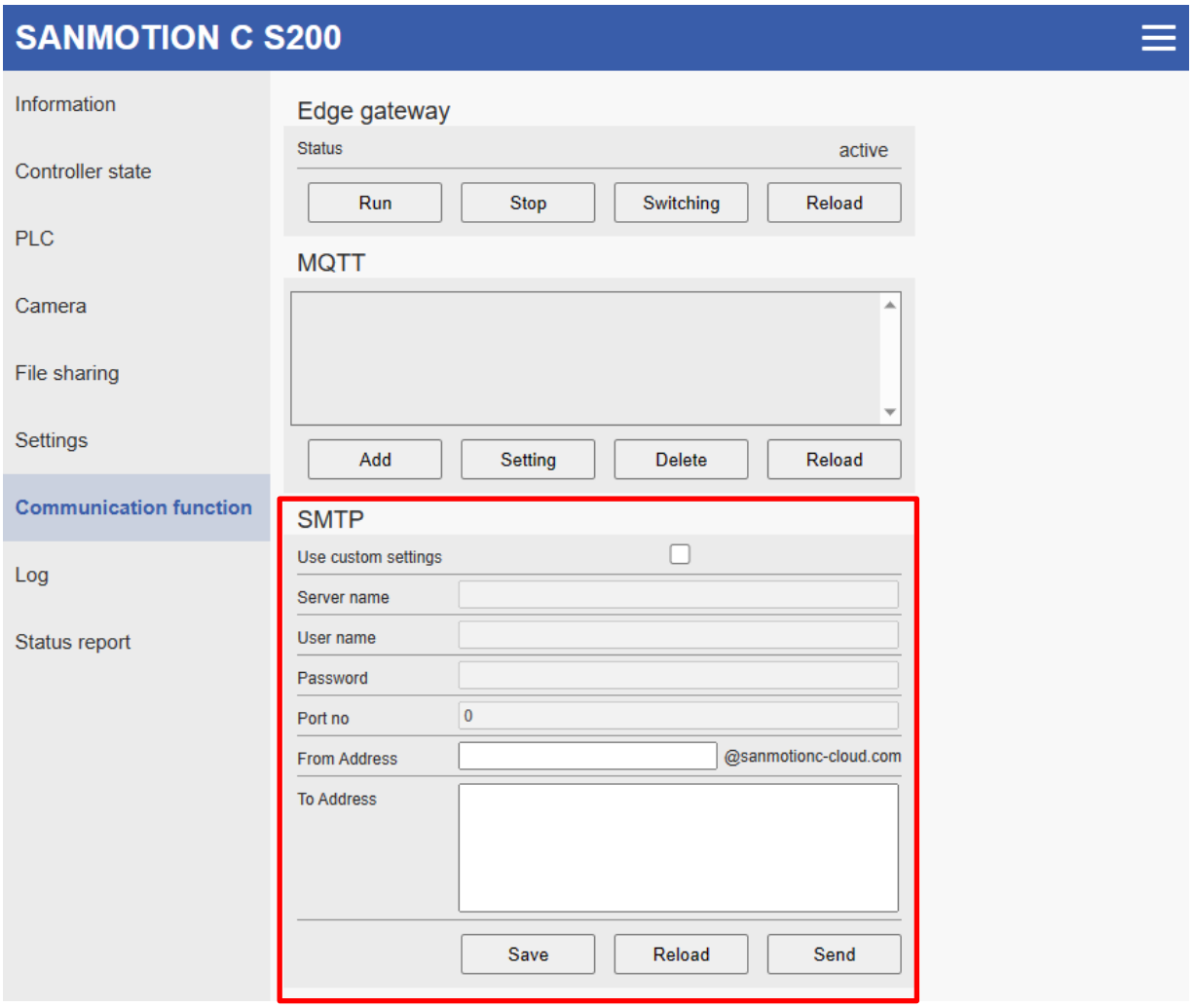


Fig.7.113 Email settings via web app

【SMTP setting items】

Item	Detail
Use custom settings	Select an SMTP server. **Enabled**: SMTP server prepared by the customer **Disabled**: SANYO DENKI SMTP server (initial value)
Server name※1	Set the connection destination of the SMTP server.
User name※1	Set the user ID when connecting to the SMTP server.
Password※1	Set the password for connecting to the SMTP server.
Port no.※1	Set the port number when connecting to the SMTP server.
From Address	Set the email sender address. As this is a send-only address, you cannot reply to this address. When using SANYO DENKI's SMTP server, specify the sender's address below. The domain name is fixed. <Account name(arbitrary setting)>@sanmotionc-cloud.com
To Address	Set the e-mail destination address. When setting multiple addresses, separate the addresses with a ";". Setting Example: a@xxx.com;b@xxx.com;

*1: Specify only when "Use custom settings" is enabled

【Button operation】

Item	Detail
Save	Saves the values of "SMTP setting items" that have been entered in the controller.
Reload	Reads the values of "SMTP setting items" saved in the controller.
Send	The email will be sent according to the contents of the displayed "SMTP setting items". Contents of the test email: Subject :From SANMOTION C Main text :This message is a test mail from SANMOTION C.

7.13.2. Function block

Below is a list of function blocks for notifying emails from the controller.

CAUTION!

- Always use the function blocks included in this library with a "task" priority of 16 or lower. Unexpected behavior may occur if executed in a "task" with a high priority.

7.13.2.1. Send_Mail

Execute the email notification using the email notification information set in the Web application.

Fig. 7.114 Send_Mail

VAR_INPUT		
Execute	BOOL	FB execution
Subject	WSTRING(255)	Subject
Content	WSTRING(2000)	Main text
VAR_OUTPUT		
Busy	BOOL	Sending mail
Done	BOOL	Mail transmission completed
BlackListExist	BOOL	Notification of presence of blacklisted mails
Error	BOOL	Error condition
ErrorId	UINT	Error detail

Sanyo Denki's SMTP server is used, and when an e-mail does not reach the specified destination, the address may be registered in the unsendable list. BlackListExist is set to TRUE if you attempt to send mail to a destination registered in the unsendable list. You can use the test sending function of the web application to check which email addresses are registered in the unsendable list among the email addresses specified as recipients.

In FB, emails are sent to email addresses other than those registered in the unsendable list, and the recipients list includes the addresses registered in the unsendable list.

Please contact us if you would like to remove your email address from the unsendable list.

7.13.2.2. SM_Alarm_SendMail

An email will be sent when an amplifier alarm occurs on the SM axis.

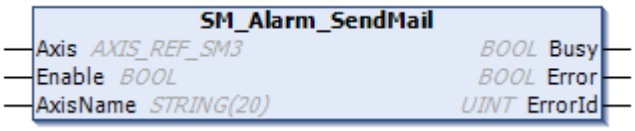


Fig. 7.115 SM_Alarm_SendMail

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Axis reference with alarm management
VAR_INPUT		
Enable	BOOL	FB execution
AxisName	STRING(20)	Name of the axis used for notification (If not set, the device tree axis name is used.)
VAR_OUTPUT		
Busy	BOOL	FB running
Error	BOOL	Error condition
ErrorId	UINT	Error detail

The subject and main text of the email sent by this function block are as follows.

Item	Detail
Subject	[Emergency] Drive Alarm Notification
Main text	[Axis Information] Axis Name : < Axis name (e.g. Axis1)> Driver : < Sanyo Denki driver series name (e.g. RS3)> [EtherCAT Information] ESM : <ESM state when an alarm occurs (e.g. OP)> ALStatus : <AL status when an alarm occurs> StatusWord : <Status word when an alarm occurs> ControlWord : < Control word when an alarm occurs > ActOperationMode : < Operation mode when an alarm occurs > SetOperationMode : < Set Operation mode when an alarm occurs > [Alarm Information] Alarm Code : <Alarm code (e.g. value of 0x2101)>

7.13.2.3. SML_Alarm_SendMail

An email will be sent when an amplifier alarm occurs on the SML axis.

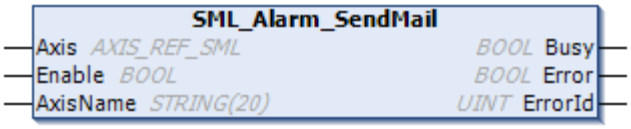


Fig. 7.116 SML_Alarm_SendMail

VAR_IN_OUT		
Axis	AXIS_REF_SML	Axis reference with alarm management
VAR_INPUT		
Enable	BOOL	FB execution
AxisName	STRING(20)	Name of the axis used for notification (If not set, the device tree axis name is used.)
VAR_OUTPUT		
Busy	BOOL	FB running
Error	BOOL	Error condition
ErrorId	UINT	Error detail

The subject and main text of the email sent by this function block are as follows.

Item	Detail
Subject	[Emergency] Drive Alarm Notification
Main text	[Axis Information] Axis Name : < Axis name (e.g. Axis1)> Driver : < Sanyo Denki driver series name (e.g. RS3)> [EtherCAT Information] ESM : <ESM state when an alarm occurs (e.g. OP)> ALStatus : <AL status when an alarm occurs> StatusWord : <Status word when an alarm occurs> ControlWord : < Control word when an alarm occurs > ActOperationMode : < Operation mode when an alarm occurs > SetOperationMode : < Set Operation mode when an alarm occurs > [Alarm Information] Alarm Code : <Alarm code (e.g. value of 0x2101)>

7.13.2.4. Error list

Below is a list of errors that occur in this function block.

Error ID	Error name	Detail
0	ERR_NOERR	No error occurred
1	ERR_RECV_DATA_FORMAT	Receive data format error
2	ERR_PARAMETER	Email parameter setting error
6	ERR_NOT_FOUND	Specified server not found
7	ERR_CONN_LOST	Connection refused
28	ERR_TIMEOUT	Timeout occurred
75	ERR_CONV_CHAR	Failed to convert to character code
99	ERR_INTERNAL	Internal error
500	ERR_UNSUPPORTED_DRIVE	Unsupported driver specified
501	ERR_GET_ALARMCODE	Failed to get alarm code
1000~	ERR_SMTP_STATUS	Error response on SMTP server
2000~	ERR_GET_TOKEN_STATUS	Error response when obtaining authentication token to Sanyo server
3000~	ERR_GET_BLACKLIST_STATUS	Error response when acquiring the Sanyo server's blacklist

7.13.3. Sample program summary

Below is a sample program that sends an email with the following contents. Use "PLC Standard project" as a template.

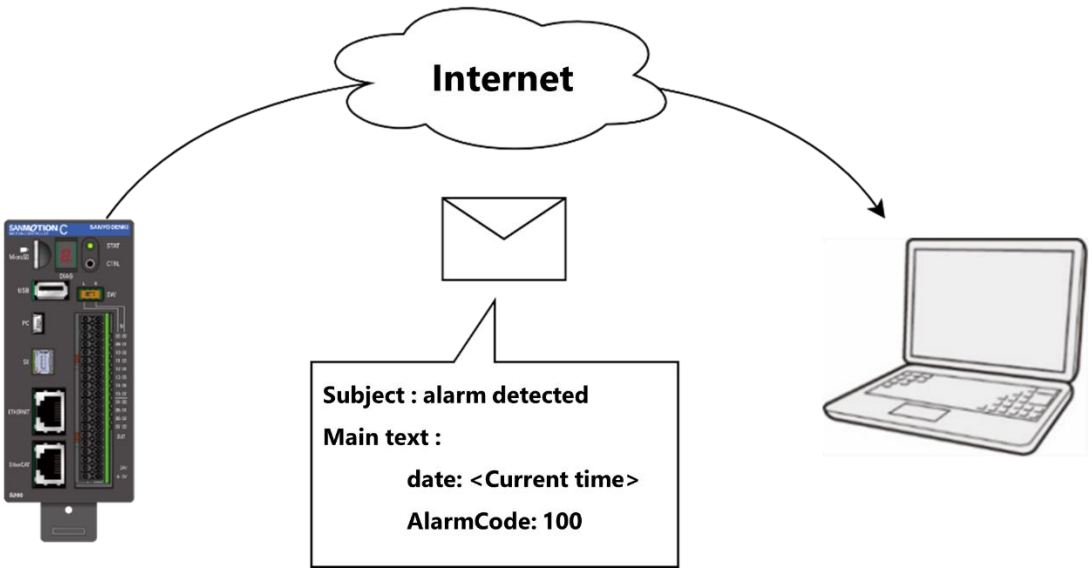


Fig. 7.117 Email sending sample program overview

7.13.4. Sample program

The sample program uses the following libraries. Please add the library.

【Used library】

Library Name	Purpose
SanMail	To use a function block that sends mail
SysTimeRtc	To use a function that obtains the current time

Write the following in PLC_PRG.

【Declaration section】

```
VAR
    Send_Mail: SanMail.Send_Mail;
    Alarmflg: BOOL;
    AlarmCode: UDINT := 100;
    CurDT: DT;
    wstrTMP: WSTRING;
    result: SysTimeRtc.RTS_IEC_RESULT;
END_VAR
```

【Implementation section】

```
IF Alarmflg THEN
    Alarmflg := FALSE;
    Send_Mail.Content := "date: ";
    wstrTMP := DT_TO_WSTRING(DWORD_TO_DT(SysTimeRtcGet(pResult:= result)));
    SanMail.StrConcatW(pstFrom := ADR(wstrTMP), pstTo:= ADR(Send_Mail.Content), iBufferSize:= TO_INT(SIZEOF(Send_Mail.Content)));
    wstrTMP := "$R$N";
    SanMail.StrConcatW(pstFrom := ADR(wstrTMP), pstTo:= ADR(Send_Mail.Content), iBufferSize:= TO_INT(SIZEOF(Send_Mail.Content)));
    wstrTMP := "AlarmCode: ";
    SanMail.StrConcatW(pstFrom := ADR(wstrTMP), pstTo:= ADR(Send_Mail.Content), iBufferSize:= TO_INT(SIZEOF(Send_Mail.Content)));
    wstrTMP := UDINT_TO_WSTRING(AlarmCode);
    SanMail.StrConcatW(pstFrom := ADR(wstrTMP), pstTo:= ADR(Send_Mail.Content), iBufferSize:= TO_INT(SIZEOF(Send_Mail.Content)));
    Send_Mail.Execute := TRUE;
ELSE
    Send_Mail.Execute := FALSE;
END_IF
Send_Mail(Subject:= "alarm detected");
```

Since it is not recommended to run the camera control function block in a real-time task, change the priority of MainTask in the template project to 16 or higher.

7.13.5. Operation check

On the web application screen described in "7.13.1 Email settings via web app", set the sender address and destination address, and set TRUE to Alarmflg of the sample program.

If the transmission is successful, you can check the following email at the specified destination.

alarm detected

sample_program@sanmotionc-cloud.com

date: DT#2023-07-26-04:20:20
AlarmCode: 100

Fig. 7.118 Sample program sent email contents

7.14. 1-Wire communication program

1-Wire is a serial interface standard that transfers data using a ground line and a single signal line. One master and multiple slaves can be connected to the bus, and the master executes communication with any slave.

As a master, the S200 can acquire measurement data from the corresponding 1-Wire sensor.

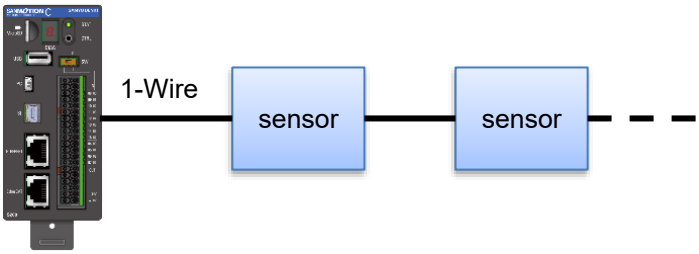
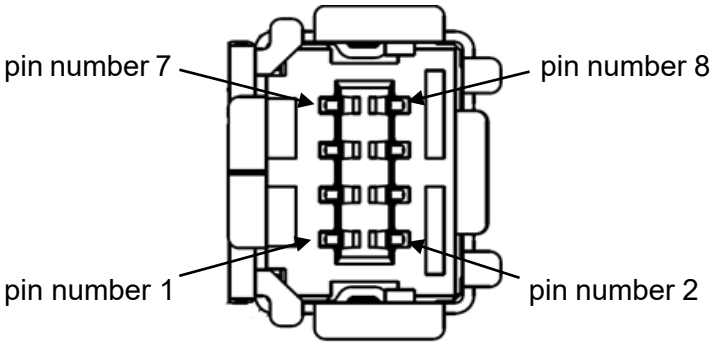


Fig. 7.119 1-Wire Communication Overview

7.14.1. Specification

Item	Detail
Supported device Automatic communication	9CT1-T（Temperature and humidity sensor） 9CT1-P（Barometric pressure sensor）
Manual communication	General purpose device
Interface	SI（Industrial Mini I/O）
Pin assignment	 Signal name / Signal content / Pin number 1-Wire / 1-Wire signal / 5 5V / Power supply for 1-Wire sensor (DC5V) / 7 DGND / Ground / 4, 8
デバイス最大接続台数※	7 台
配線長	最大 200m
通信速度	15400 [bps]
最大通信データ長	255 [Bytes] （手動通信時）
通信開始タイミング	起動直後
通信周期	デバイス未接続時のデバイス検出周期:60 秒 デバイス接続時のデバイス検出周期:30 秒 デバイスの値取得周期:100ms/台
検出デバイスのリセット方法※	再起動

* Detected devices are retained in the detection list even after disconnection. Therefore, the maximum number of connected devices means the total number of devices detected after power-on. For example, after detecting 7 devices, if one device is disconnected and a new device is connected, an error in the number of connected devices will occur.

7.14.2. Function block

Below is a list of function blocks that control 1-wire communication.

CAUTION!

- Always use the function blocks included in this library with a "task" priority of 16 or lower. Unexpected behavior may occur if executed in a "task" with a high priority.

7.14.2.1. GetList

Gets information about all devices connected to the 1-Wire bus and returns an array of structure DeviceList that stores data according to the 64-bit ID and device type.

Fig. 7.120 GetList

VAR_INPUT		
Execute	BOOL	Start getting list on rising edge
VAR_OUTPUT		
Done	BOOL	TRUE: Acquisition of device list completed
Busy	BOOL	TRUE: Getting device list
Error	BOOL	TRUE: Failed to get device list
ErrorID	ERROR	Error identifier
DeviceList	ARRAY[1..7] OF DeviceList	Detected device list
DetectedDeviceNum	USINT	Number of devices detected

Due to the specifications of the 1-Wire communication protocol, the list output to DeviceList may differ from the actual connection order. Please check the detected 64-bit ID carefully before using it.

7.14.2.2. GeneralCom

Using the obtained DeviceID, perform 1-Wire communication with a general-purpose device. General-purpose communication is possible by entering the command to be sent and the data size to be received.

Fig. 7.121 GeneralCom

VAR_INPUT		
Execute	BOOL	Send command on rising edge
ComDeviceID	DeviceID	DeviceID of the communicating device
WriteData	POINTER TO BYTE	A pointer to the GeneralCommandData array to send
uiWriteArrayNum	UINT	Number of elements in the GeneralCommandData array to send
ReadData	POINTER TO BYTE	A pointer to the GeneralCommandData array containing the received data
VAR_OUTPUT		
Done	BOOL	TRUE: Data transmission/reception completed
Busy	BOOL	TRUE: In communication
Error	BOOL	TRUE: Failed to send and receive data
ErrorID	ERROR	Error identifier
uiReceiveNum	UINT	Number of packets received

7.14.2.3. Error list

Below is a list of errors that occur in this function block.

Error ID	Error name	Detail
0	NO_ERROR	No error occurred
1	COMMUNICATION_BUSY	1-Wire bus busy
2	TIMEOUT	Timeout error
3	NO_DEVICE_DETECTED	1-Wire device not detected
4	TOO_MANY_DEVICE_DETECTED	More than maximum number of devices detected
5	COMMAND_TOO_LONG	General purpose communication command exceeds maximum length
6	INCORRECT_ARRAY_NUM	The number of general communication commands is 0 or exceeds the maximum number
7	INCORRECT_DEVICE_ID	The DeviceID entered does not exist in the list
8	INCORRECT_DEVICE_TYPE	Invalid DeviceType entered
9	READ_WRITE_FAILED	Failed to write/read to 1-Wire bus
10	INTERNAL_ERROR	Internal error

7.14.3. List information structure

The structures used in function blocks are described below.

7.14.3.1. DeviceList (STRUCT)

The structure used for GetList output.

Variable name	Data type	Detail
DeviceID	DeviceID	Stores device ID and device type
CommonData	CommonData	Stores the device common data part
UniqueData	UniqueData	Stores device-specific data

7.14.3.2. DeviceID (STRUCT)

A structure containing the 1-Wire device ID and device type.

Variable name	Data type	Detail
DeviceID	LWORD	64-bit ID of the device
DeviceType	DeviceType	Device type

7.14.3.3. DeviceType (ENUM)

Enumerated type that defines SANYODENKI devices.

Variable name	Data type	Detail
General	0	General purpose device
San_9CT1_T	1	San Ace temperature and humidity sensor
San_9CT1_P	2	San Ace barometric pressure sensor

7.14.3.4. CommonData (STRUCT)

The structure used for GetList output.

Variable name	Data type	Detail
DeviceStatus	DeviceStatus	Device communication status
ComCount	UDINT	Number of communication executions with the device
ErrorCount	UDINT	Number of communication failures with the device

7.14.3.5. DeviceStatus (ENUM)

Enumerated data that defines the communication state of the device.

Variable name	Data type	Detail
NOT_COM	0	Communication not executed (transition only at first detection)
OK	1	Communicating normally
ERROR	2	Communication error

7.14.3.6. UniqueData (UNION)

The union used for the output of GetList.

Variable name	Data type	Detail
General	U_General	Data part structure for general-purpose devices
San_9CT1_T	U_9CT1_T	Data part structure for temperature and humidity sensor
San_9CT1_P	U_9CT1_P	Data part structure for barometric pressure sensor

7.14.3.7. U_General (STRUCT)

The structure used for GetList output.

Variable name	Data type	Detail
Com_Busy	BOOL	General purpose communication busy flag **TRUE**: Busy **FALSE**: Available
LastError	San1WireCom_ERROR	ID of the last error that occurred
TimeStamp	TimeStamp	Time of last general purpose communication

7.14.3.8. U_9CT1_T (STRUCT)

The structure used for GetList output.

Variable name	Data type	Detail
Temperature	REAL	Sensor temperature reading [°C]
Humidity	REAL	Sensor humidity reading [%]
TimeStamp	TimeStamp	Time when the measured value was acquired

7.14.3.9. U_9CT1_P (STRUCT)

The structure used for GetList output.

Variable name	Data type	Detail
Pressure	REAL	Pressure measurement value of the sensor [Pa]
TimeStamp	TimeStamp	Time when the measured value was acquired

7.14.3.10. TimeStamp (STRUCT)

A structure for representing the time of communication with a 1-Wire device.

The time is the RTC value.

Variable name	Data type	Detail
TsDate	DATE	Year, month, day of timestamp
TsTimeOfDay	TIME_OF_DAY	Timestamp hour, minute, second (in ms)
RawData	LWORD	Raw timestamp data

7.14.3.11. GeneralCommandData (STRUCT)

Packet data structure used for general-purpose 1-Wire communication. Used for input/output of GeneralCom (FB).

Variable name	Data type	Detail
Data	ARRAY [0..255] OF BYTE	An array that stores data for one command
Length	USINT	Data length

7.14.4. Sample program summary

In this sample program, measurement data is acquired from the temperature/humidity sensor 9CT1-T and barometric pressure sensor 9CT1-P connected to the S200. Use "PLC Standard project" as a template.

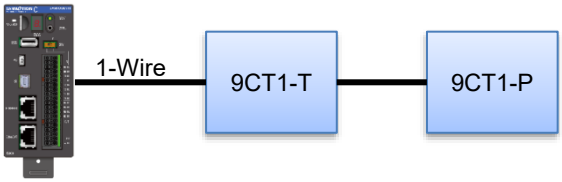


Fig. 7.122 1-Wire communication sample program overview

7.14.5. Sample program

The sample program uses the following libraries. Please add the library.

【Used library】

Library Name	Purpose
San1WireCom	To use the function block for 1-Wire communication

Write the following in PLC_PRG.

【Declaration section】

```
VAR
        GetDeviceList   :San1WC.GetList;        // FB to get device list
        Timer           :TON;                   // Timer
        MainStep        :INT;                   // Main operation step control variable
END_VAR
```

【Implementation section】

```
GetDeviceList();
Timer(PT:= T#10S);
CASE MainStep OF
        0 : (* Get list *)
                GetDeviceList.Execute:= TRUE;
                IF GetDeviceList.Done THEN
                        MainStep:= MainStep + 1;
                ELSIF GetDeviceList.Error THEN
                        GetDeviceList.Execute:= FALSE;
                END_IF
        1 : (* Timer *)
                Timer.IN := TRUE;
                IF Timer.Q THEN
                        GetDeviceList.Execute:= FALSE;
                        Timer.IN := FALSE;
                        MainStep:= MainStep - 1;
                END_IF
END_CASE
```

Since the function block for 1-Wire communication is not recommended to be executed as a real-time task, change the priority of MainTask in the template project to 16 or higher.

7.14.6. Operation check

Acquisition of the device list starts automatically after the application starts. After obtaining the list, it is updated every 10 seconds using a timer.

From the DeviceList output, confirm that data such as the device ID, temperature/humidity, and atmospheric pressure have been acquired.

GetDeviceList	San1WC.GetList	
Execute	BOOL	TRUE
Done	BOOL	TRUE
Busy	BOOL	FALSE
Error	BOOL	FALSE
ErrorID	SAN1WIRECOM_ER...	NO_ERROR
DeviceList	ARRAY [1..San1Wir...	
DeviceList[1]	DeviceList	
DeviceID	DeviceID	
ID	LWORD	16#19A25E040000003C
DeviceType	SAN1WIRECOM_DE...	San_9CT1_T
CommonData	CommonData	
DeviceStatus	SAN1WIRECOM_DE...	OK
ComCount	UDINT	651
ErrorCount	UDINT	0
UniqueData	UniqueData	
General	U_General	
San_9CT1_T	U_9CT1_T	
Temperature	REAL	26.2054443
Humidity	REAL	57.69043
TimeStamp	TimeStamp	
San_9CT1_P	U_9CT1_P	
DeviceList[2]	DeviceList	
DeviceID	DeviceID	
ID	LWORD	16#199F6E0300000099
DeviceType	SAN1WIRECOM_DE...	San_9CT1_P
CommonData	CommonData	
UniqueData	UniqueData	
General	U_General	
San_9CT1_T	U_9CT1_T	
San_9CT1_P	U_9CT1_P	
Pressure	REAL	96070.46
TimeStamp	TimeStamp	

Fig. 7.123 1-Wire communication sample execution screen

7.15. MQTT communication program

MQTT (Message Queuing Telemetry Transport) is a publish-subscribe model communication protocol that is specialized for IoT.

In MQTT communication, the message sender is the publisher, the message receiver is the subscriber, and the broker, which is the message relay server, manages the messages and distributes them appropriately to the subscribers.

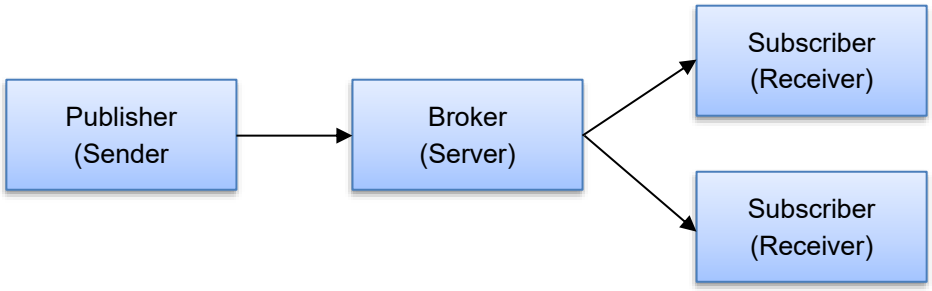


Fig. 7.124 MQTT model

7.15.1. Specification

Item	Detail	Note
Supported MQTT version	3.1, 3.1.1, 5.0	
Supported commands	・Distribution of messages (PUBLISH) ・Subscription registration (SUBSCRIBE) ・Unsubscribe (UNSUBSCRIBE)	
Command transmission method	Manual (FB input variable operation)	Simultaneous transmission times: 10
Supported QoS level	0/1/2	0: Maximum of 1 time. No arrival guarantee 1: Minimum of 1 time. Arrival guaranteed (Possibility of duplicate arrival) 2: Exactly once. Arrival guaranteed
Will function	Supported	Sends the Topic and Payload specified in this Will when the server cannot communicate with the client. The Subscriber side can determine that the Publisher side has been disconnected.
TLS version	tlsv1.3, tlsv1.2, tlsv1.1	Default:tlsv1.2
Certificate designation method	Specify the server name registered in the web application	Supported format: PEM format Configurable file: ・Certificate authority (CA) certificate ・Client certificate ・Client private key
Server certificate verification	Define validation requirements imposed on the server SSL_VERIFY_NONE:don't validate SSL_VERIFY_PEER:verify. Connection is aborted if validation fails	Valid only when a server certificate is set
Multibyte characters	Not supported	
Maximum topic size	255byte	
Maximum payload size	1000byte	

7.15.2. Certificate registration

Various certificates are required for TLS communication with the broker. Certificates can be registered from the web app.

For details, refer to "M0020986 Web Application Instruction Manual".

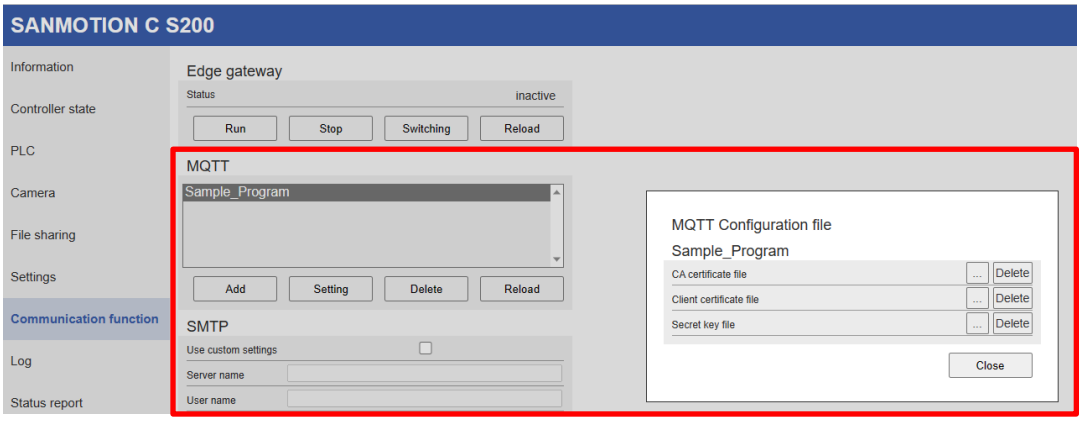


Fig. 7.125 Certificate registration screen

7.15.3. Function block

Below is a list of function blocks that control MQTT communication.

CAUTION!

- When performing TLS communication using the function blocks included in this library, be sure to set the "task" priority to 16 or higher.
  Unexpected behavior may occur if executed in a "task" with a high priority.

7.15.3.1. CONNECT

Performs MQTT connection/disconnection processing.

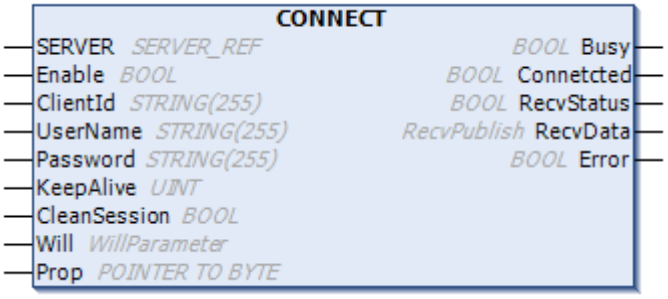


Fig. 7.126 CONNECT

VAR_INOUT		
SERVER	SERVER_REF	Reference variable
VAR_INPUT		
Enable	BOOL	Perform connection/disconnection control
ClientId	STRING(255)	Client ID
UserName	STRING(255)	User name
Password	STRING(255)	Password
KeepAlive	UINT	Keep alive interval
CleanSession	BOOL	Clean session flag
Will	WillParameter	Will command information
Prop	POINTER TO BYTE	Property information
VAR_OUTPUT		
Busy	BOOL	Command transmission status
Connetcted	BOOL	Command transmission completion status
RecvStatus	BOOL	Reception status
RecvData	RecvPublish	Store received data
Error	BOOL	Error condition

7.15.3.2. PUBLISH

Transfer MQTT PUBLISH command.

Fig. 7.127 PUBLISH

VAR_INOUT		
SERVER	SERVER_REF	Reference variable
VAR_INPUT		
Execute	BOOL	Execute FB
Topic	STRING(255)	Topic
ptPayload	POINTER TO BYTE	Payload
Payloadlen	UINT	Payload size
QoS	BYTE	QoS level
xRetain	BOOL	Hold flag
Prop	POINTER TO BYTE	Property information
VAR_OUTPUT		
Busy	BOOL	Command transmission status
Done	BOOL	Command transmission completion status
Error	BOOL	Error condition
ErrorId	UDINT	Error detail number in FB

7.15.3.3. SUBSCRIBE

Transfer MQTT SUBSCRIBE command.

Fig. 7.128 SUBSCRIBE

VAR_INOUT		
SERVER	SERVER_REF	Reference variable
VAR_INPUT		
Execute	BOOL	Execute FB
Topic	STRING(255)	Topic
QoS	BYTE	QoS level
Options	BYTE	Options to use with MQTTv5 subscriptions
Prop	POINTER TO BYTE	Property information
VAR_OUTPUT		
Busy	BOOL	Command transmission status
Done	BOOL	Command transmission completion status
Error	BOOL	Error condition
ErrorId	UDINT	Error detail number in FB

7.15.3.4. UNSUBSCRIBE

Transfer the MQTT UNSUBSCRIBE command.

Fig. 7.129 UNSUBSCRIBE

VAR_INOUT		
SERVER	SERVER_REF	Reference variable
VAR_INPUT		
Execute	BOOL	Execute FB
Topic	STRING(255)	Topic
Prop	POINTER TO BYTE	Property information
VAR_OUTPUT		
Busy	BOOL	Command transmission status
Done	BOOL	Command transmission completion status
Error	BOOL	Error condition
ErrorId	UDINT	Error detail number in FB

7.15.3.5. SERVER_REF

The structure contents of the reference variable are described below.

Variable name		Data type	Detail
HostName		STRING(255)	Host name of the connection destination
PortNo		UINT	Port number of the connection destination
TimeoutTime		TIME	Time until timeout
ProtoVer		Protocol_Version	Protocol version mqtt_v3_1：3 mqtt_v3_1_1：4（Default） mqtt_v5：5
TLS	Enable	BOOL	TLS enable/disable
	Version	TlsVersion	TLS version setting tlsv1_1：1 tlsv1_2：2（Default） tlsv1_3：3
	ServerName	STRING(100)	The server name set when the certificate was registered with the web application
	KeyPassPhrase	STRING(255)	Private key passphrase for client authentication
	ServerCertCheck	BOOL	Enable/disable server certificate check
ErrorId		UINT	Error identifier

7.15.3.6. Error list

Below is a list of errors that occur in this function block.

Error ID	Error name	Detail
0	ERR_SUCCESS	No error occurred
1	ERR_NOMEM	Out of memory error
2	ERR_PROTOCOL	A protocol error occurred while communicating with the broker
3	ERR_INVAL	Input parameter error
4	ERR_NO_CONN	Client not connected to broker
5	ERR_CONN_REFUSED	Connection refused error
6	ERR_NOT_FOUND	Server not found
7	ERR_CONN_LOST	Disconnect from server
8	ERR_TLS	TLS parameter error
9	ERR_PAYLOAD_SIZE	Payload size error
10	ERR_NOT_SUPPORTED	Unsupported property
11	ERR_AUTH	User authentication error
12	ERR_ACL_DENIED	Access was not granted
13	ERR_UNKNOWN	Application specific error
14	ERR_INTERNAL	Internal error
15	ERR_NETWORK	Network setting error
19	ERR_KEEPALIVE	No response within the time set by keep-alive
20	ERR_LOOKUP	Broker not found
22	ERR_DUPLICATE_PROPERTY	Duplicate property error
23	ERR_TLS_HANDSHAKE	TLS handshake failed
24	ERR_QOS_NOT_SUPPORTED	Used a higher QoS than what the broker supports
25	ERR_OVERSIZE_PACKET	The specified packet exceeds the size supported by the broker
27	ERR_TIMEOUT	Timeout error
28	ERR_RETAIN_NOT_SUPPORTED	Broker does not support retention
29	ERR_TOPIC_ALIAS_INVALID	Topic is null or longer than 255 characters
30	ERR_ADMINISTRATIVE_ACTION	Abnormal administrator privileges
31	ERR_ALREADY_EXISTS	Already exists
200	ERR_NOT_READY	Not ready to transfer commands
202	ERR_PAYLOAD_IS_NULL	Payload pointer not set
203	ERR_MAX_PROSECC_OVER	Exceeded number of concurrent executions
1001	ERR_CONNECTION_PROTOCOL	Connection return code (protocol level error)
1002	ERR_CONNECTION_CLIENT_ID	Connection return code (client identifier error)
1003	ERR_CONNECTION_MQTT_SERVICE	Connection return code (MQTT service unavailable)
1004	ERR_CONNECTION_USER_PASS	Connection return code (username, password error)
1005	ERR_CONNECTION_UNAUTH_CLIENT	Connection return code (bad client)
1200	ERR_SUBACK_FAILURE	SUBACK reception failure flag

7.15.4. Sample program summary

An example of creating a communication program with the test broker, test.mosquitto.org, is shown below. By subscribing in advance to the topic to be published, the same processing as the echo server is performed to confirm that publishing and subscribing are performed normally. Use "PLC Standard project" as a template.

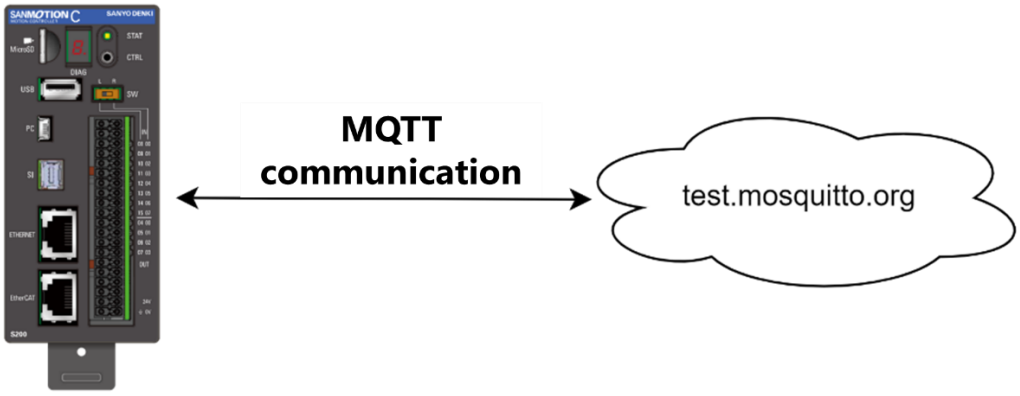


Fig. 7.130 Overview of MQTT communication sample program

7.15.5. Sample program

The sample program uses the following libraries. Please add the library.

【Used library】

Library Name	Purpose
SanMQTT	To use the function block that performs MQTT communication
SysMem	To copy received data to local variables by memory control

Write the following in PLC_PRG.

【Declaration section】

```
VAR
SampleServer : SanMQTT.SERVER_REF := (HostName := 'test.mosquitto.org', PortNo := 1883,
ProtoVer := SanMQTT.Protocol_Version.mqtt_v5);
  CONNECT : SanMQTT.CONNECT;
  PUBLISH : SanMQTT.PUBLISH;
  SUBSCRIBE : SanMQTT.SUBSCRIBE;
  SendData : STRING := '{"massage": "hello"}';
  RecvData : STRING;
  PropHandle : SanMQTT.RTS_IEC_HANDLE;
  CurrPropHandle : SanMQTT.RTS_IEC_HANDLE;
  CurrPropId : SanMQTT.Properties;
  xSendPublish : BOOL;
  SendPropValue : STRING := 'test';
  stpRecvPropName : STRING;
  stpRecvPropValue : STRING;
  byRecvPropValue : BYTE;
  stRecvPropValue : STRING;
END_VAR
```

【Implementation section】

```
CONNECT(SERVER:= SampleServer, Enable := TRUE);
SUBSCRIBE(SERVER:= SampleServer, Execute := CONNECT.Connetcted, Topic:= 'myTest', QoS:= 2);
PUBLISH(SERVER:= SampleServer, Execute := xSendPublish, Topic:= 'myTest', ptPayload:= ADR(SendData), Payloadlen:= INT_TO_UINT(LEN(SendData)), QoS:= 2, Prop:= PropHandle);

IF xSendPublish THEN
    xSendPublish := FALSE;
    PropHandle := SanMQTT.prop_free(Handle:= ADR(PropHandle));
    SanMQTT.prop_add_byte(Handle:= ADR(PropHandle), PropertyId:= SanMQTT.Properties.PayloadFormatIndicator, Value:= 0);
    SanMQTT.prop_add_string(Handle:= ADR(PropHandle), PropertyId:= SanMQTT.Properties.ContentType, Value:= 'application/json');
    SanMQTT.prop_add_string_pair(Handle:= ADR(PropHandle), PropertyId:= SanMQTT.Properties.UserProperty, Name:= 'UserProperty', Value:= 'test');
END_IF

IF CONNECT.RecvStatus THEN
    SysMemCpy(pDest:= ADR(RecvData), pSrc:= CONNECT.RecvData.ptPayload, udiCount:= CONNECT.RecvData.PayloadNum);
    SysMemSet(pDest:= ADR(RecvData)+CONNECT.RecvData.PayloadNum, udiValue:= 0, udiCount:= 1);
    CurrPropHandle := CONNECT.RecvData.Prop;
    REPEAT
        CurrPropId := SanMQTT.prop_get_identifier(PropAddress:= CurrPropHandle);
        CASE CurrPropId OF
            SanMQTT.Properties.PayloadFormatIndicator : SanMQTT.prop_read_byte(Handle:= CurrPropHandle, PropertyId:= CurrPropId, Value:= ADR(byRecvPropValue));
            SanMQTT.Properties.ContentType : SanMQTT.prop_read_string(Handle:= CurrPropHandle, PropertyId:= CurrPropId, Value:= ADR(stRecvPropValue));
            SanMQTT.Properties.UserProperty : SanMQTT.prop_read_string_pair(Handle:= CurrPropHandle, PropertyId:= CurrPropId, Name:= ADR(stpRecvPropName), Value:= ADR(stpRecvPropValue));
        END_CASE
        CurrPropHandle := SanMQTT.prop_get_next_address(PropAddress:= CurrPropHandle);
    UNTIL
        CurrPropHandle = 0
    END_REPEAT
END_IF
```

7.15.6. Operation chek

The sample program starts connecting to test.mosquitto.org as soon as the PLC application starts, and subscribes to the topic 'myTest' when the connection is complete.After completing the above process, set xSendPublish to TRUE at any time.If the communication is successful, the published data (payload and properties) is returned as-is.

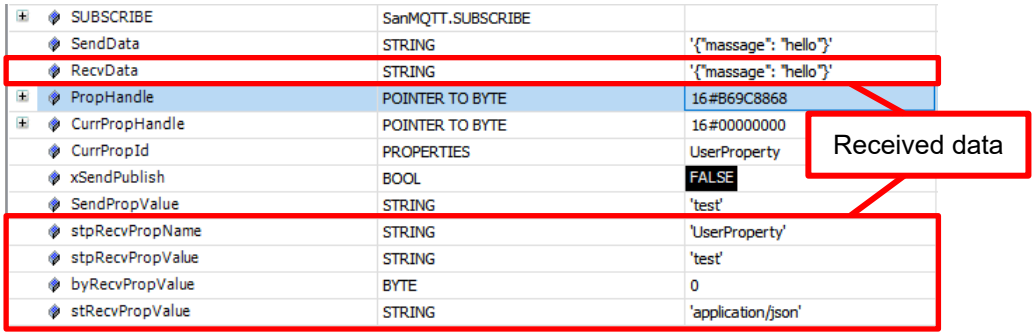


Fig. 7.131 MQTT communication received data

8. Limitations

8.1. For RTC Setting

This S200 manages RTC with UTC. When RTC (real time clock) is set with the following FB or function by setting time zones other than UTC, RTC will be set in local time. Even when using the PLC Shell command "rtc - set", the above phenomenon occurs.

To set the RTC, set the UTC time by setting the time zone to UTC.

Library	POU
CAA Real Time Clock Extern Library CAA DTUtil Extern Library	SetDateAndTime
SysTimeRtc	SysTimeRtcSet
SysRTC23	SysRtcSetTime

8.2. Regarding homing

There are restrictions on homing to the following SANYO DENKI Servo Amplifier.

8.2.1. RS2 series (Model Number：RS2*****K**)

Homing method 35 (homing on current position) can not be performed when the firmware amplifier revision is before "H". It can be executed with "J" or later firmware amplifier revision. If you are using a firmware amplifier revision before “H', please contact us.

Also, if you are using an absolute system, please write "0x65766173" in Sub-Idx01(All parameters storage) of OD:0x1010 (parameter storage) after homing. It takes about 10 seconds to write.For details, please refer to the instruction manual of type K of RS2 (M0008888G).

8.2.2. Homing of SANMOTION EtherCAT slave

When performing homing with SANMOTION EtherCAT slave, use SanHome(FB) instead of MC_Home(FB). If MC_Home(FB) is used, it may not work properly.

For details of SanHome(FB), refers to”9.2.1.11 SanHome”

8.2.3. Cancellation of MC_Home_SML

If the homing operation is canceled by MC_Stop_SML(FB), quick stop, or servo off, homing cannot be performed again. By writing FALSE to "bStartHoming" of the axis reference, the homing can be executed again.

If the homing operation by MC_Home_SML(FB) is canceled by turning the servo off, it is necessary to set a waiting time of at least 1 second before turning the servo on again. If the servo-on timing is too early, homing cannot be performed.

For details of MC_Home_SML(FB), refers to "9.2.2.3 MC_Home_SML".

8.3. Regarding visualization

8.3.1. Antialiasing settings

There is a setting of "Antialiased drawing" in the setting item of Web visualization. There is no clear difference in drawing due to setting changes, but it is recommended to always enable it.

8.3.2. Regarding ActiveX elements

There is ActiveX as a visualization element, but it cannot be used because it does not support this element.

8.4. Regarding retain variables

The S200 uses the nonvolatile memory FeRAM as the storage destination for the retain variables to eliminate the trouble of losing the retain variable value due to the low battery level and the need for battery replacement work.

When there is space in the execution process, the retain variable value is reflected in FeRAM. If the power is turned off immediately after changing the retain variable value, the power may be turned off before it is reflected in FeRAM. In that case, it will start with the value before the change.

Therefore, if you change the retain variable, wait a few seconds before shutting off the power (if the retain variable is used a lot, the time required for reflection will be longer).

8.5. Invert direction parameter of the SML axis

Do not enable the "Reverse direction" parameter of SML. If it is enabled, unexpected behavior may occur. If you want to set the direction of rotation to be reversed, set it on the EtherCAT slave side.

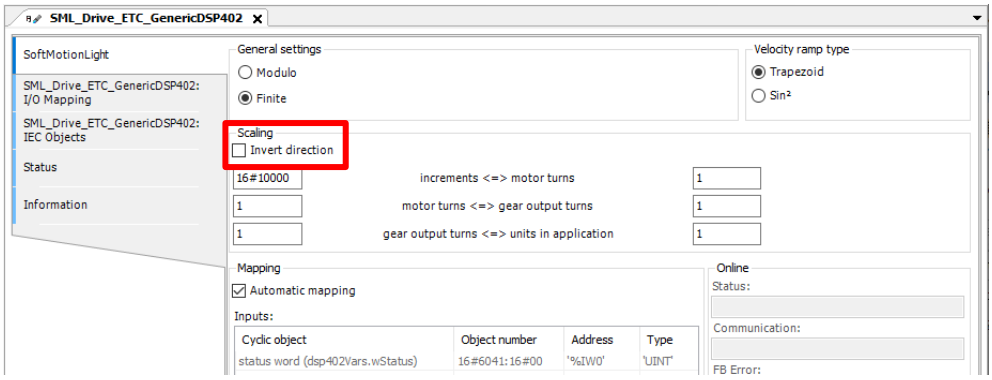


Fig 8.1 Unavailable parameters in SML axis

8.6. Ethernet communication after startup

The Ethernet communication that is executed immediately after startup may cause communication errors due to the limited processing capacity. For this reason, please wait a few seconds after startup before executing Ethernet communication.

9. Appendix

9.1. Time zone list

The list of time zones that can be set with this S200 is shown below. The default value is "Asia/Tokyo" highlighted in yellow.

Africa/Abidjan	Africa/Niamey	America/Chihuahua
Africa/Accra	Africa/Nouakchott	America/Costa_Rica
Africa/Addis_Ababa	Africa/Ouagadougou	America/Creston
Africa/Algiers	Africa/Porto-Novo	America/Cuiaba
Africa/Asmara	Africa/Sao_Tome	America/Curacao
Africa/Bamako	Africa/Tripoli	America/Danmarkshavn
Africa/Bangui	Africa/Tunis	America/Dawson
Africa/Banjul	Africa/Windhoek	America/Dawson_Creek
Africa/Bissau	America/Adak	America/Denver
Africa/Blantyre	America/Anchorage	America/Detroit
Africa/Brazzaville	America/Anguilla	America/Dominica
Africa/Bujumbura	America/Antigua	America/Edmonton
Africa/Cairo	America/Araguaina	America/Eirunepe
Africa/Casablanca	America/Argentina/Buenos_Aires	America/El_Salvador
Africa/Ceuta	America/Argentina/Catamarca	America/Fort_Nelson
Africa/Conakry	America/Argentina/Cordoba	America/Fortaleza
Africa/Dakar	America/Argentina/Jujuy	America/Glace_Bay
Africa/Dar_es_Salaam	America/Argentina/La_Rioja	America/Godthab
Africa/Djibouti	America/Argentina/Mendoza	America/Goose_Bay
Africa/Douala	America/Argentina/Rio_Gallegos	America/Grand_Turk
Africa/El_Aaiun	America/Argentina/Salta	America/Grenada
Africa/Freetown	America/Argentina/San_Juan	America/Guadeloupe
Africa/Gaborone	America/Argentina/San_Luis	America/Guatemala
Africa/Harare	America/Argentina/Tucuman	America/Guayaquil
Africa/Johannesburg	America/Argentina/Ushuaia	America/Guyana
Africa/Juba	America/Aruba	America/Halifax
Africa/Kampala	America/Asuncion	America/Havana
Africa/Khartoum	America/Atikokan	America/Hermosillo
Africa/Kigali	America/Bahia	America/Indiana/Indianapolis
Africa/Kinshasa	America/Bahia_Banderas	America/Indiana/Knox
Africa/Lagos	America/Barbados	America/Indiana/Marengo
Africa/Libreville	America/Belem	America/Indiana/Petersburg
Africa/Lome	America/Belize	America/Indiana/Tell_City
Africa/Luanda	America/Blanc-Sablon	America/Indiana/Vevay
Africa/Lubumbashi	America/Boa_Vista	America/Indiana/Vincennes
Africa/Lusaka	America/Bogota	America/Indiana/Winamac
Africa/Malabo	America/Boise	America/Inuvik
Africa/Maputo	America/Cambridge_Bay	America/Iqaluit
Africa/Maseru	America/Campo_Grande	America/Jamaica
Africa/Mbabane	America/Cancun	America/Juneau
Africa/Mogadishu	America/Caracas	America/Kentucky/Louisville
Africa/Monrovia	America/Cayenne	America/Kentucky/Monticello
Africa/Nairobi	America/Cayman	America/Kralendijk

Africa/Ndjamena	America/Chicago	America/La_Paz
America/Lima	America/St_Barthelemy	Asia/Damascus
America/Los_Angeles	America/St_Johns	Asia/Dhaka
America/Lower_Princes	America/St_Kitts	Asia/Dili
America/Maceio	America/St_Lucia	Asia/Dubai
America/Managua	America/St_Thomas	Asia/Dushanbe
America/Manaus	America/St_Vincent	Asia/Famagusta
America/Marigot	America/Swift_Current	Asia/Gaza
America/Martinique	America/Tegucigalpa	Asia/Hebron
America/Matamoros	America/Thule	Asia/Ho_Chi_Minh
America/Mazatlan	America/Thunder_Bay	Asia/Hong_Kong
America/Menominee	America/Tijuana	Asia/Hovd
America/Merida	America/Toronto	Asia/Irkutsk
America/Metlakatla	America/Tortola	Asia/Jakarta
America/Mexico_City	America/Vancouver	Asia/Jayapura
America/Miquelon	America/Whitehorse	Asia/Jerusalem
America/Moncton	America/Winnipeg	Asia/Kabul
America/Monterrey	America/Yakutat	Asia/Kamchatka
America/Montevideo	America/Yellowknife	Asia/Karachi
America/Montserrat	Antarctica/Casey	Asia/Kathmandu
America/Nassau	Antarctica/Davis	Asia/Khandyga
America/New_York	Antarctica/DumontDUrville	Asia/Kolkata
America/Nipigon	Antarctica/Macquarie	Asia/Krasnoyarsk
America/Nome	Antarctica/Mawson	Asia/Kuala_Lumpur
America/Noronha	Antarctica/McMurdo	Asia/Kuching
America/North_Dakota/Beulah	Antarctica/Palmer	Asia/Kuwait
America/North_Dakota/Center	Antarctica/Rothera	Asia/Macau
America/North_Dakota/ New_Salem	Antarctica/Syowa	Asia/Magadan
America/Ojinaga	Antarctica/Troll	Asia/Makassar
America/Panama	Antarctica/Vostok	Asia/Manila
America/Pangnirtung	Arctic/Longyearbyen	Asia/Muscat
America/Paramaribo	Asia/Aden	Asia/Nicosia
America/Phoenix	Asia/Almaty	Asia/Novokuznetsk
America/Port-au-Prince	Asia/Amman	Asia/Novosibirsk
America/Port_of_Spain	Asia/Anadyr	Asia/Omsk
America/Porto_Velho	Asia/Aqtau	Asia/Oral
America/Puerto_Rico	Asia/Aqtobe	Asia/Phnom_Penh
America/Punta_Arenas	Asia/Ashgabat	Asia/Pontianak
America/Rainy_River	Asia/Atyrau	Asia/Pyongyang
America/Rankin_Inlet	Asia/Baghdad	Asia/Qatar
America/Recife	Asia/Bahrain	Asia/Qyzylorda
America/Regina	Asia/Baku	Asia/Riyadh
America/Resolute	Asia/Bangkok	Asia/Sakhalin
America/Rio_Branco	Asia/Barnaul	Asia/Samarkand
America/Santarem	Asia/Beirut	Asia/Seoul
America/Santiago	Asia/Bishkek	Asia/Shanghai
America/Santo_Domingo	Asia/Brunei	Asia/Singapore
America/Sao_Paulo	Asia/Chita	Asia/Srednekolymsk
America/Scoresbysund	Asia/Choibalsan	Asia/Taipei

America/Sitka	Asia/Colombo	Asia/Tashkent
Asia/Tbilisi	Europe/Dublin	Indian/Chagos
Asia/Tehran	Europe/Gibraltar	Indian/Christmas
Asia/Thimphu	Europe/Guernsey	Indian/Cocos
Asia/Tokyo	Europe/Helsinki	Indian/Comoro
Asia/Tomsk	Europe/Isle_of_Man	Indian/Kerguelen
Asia/Ulaanbaatar	Europe/Istanbul	Indian/Mahe
Asia/Urumqi	Europe/Jersey	Indian/Maldives
Asia/Ust-Nera	Europe/Kaliningrad	Indian/Mauritius
Asia/Vientiane	Europe/Kiev	Indian/Mayotte
Asia/Vladivostok	Europe/Kirov	Indian/Reunion
Asia/Yakutsk	Europe/Lisbon	Pacific/Apia
Asia/Yangon	Europe/Ljubljana	Pacific/Auckland
Asia/Yekaterinburg	Europe/London	Pacific/Bougainville
Asia/Yerevan	Europe/Luxembourg	Pacific/Chatham
Atlantic/Azores	Europe/Madrid	Pacific/Chuuk
Atlantic/Bermuda	Europe/Malta	Pacific/Easter
Atlantic/Canary	Europe/Mariehamn	Pacific/Efate
Atlantic/Cape_Verde	Europe/Minsk	Pacific/Enderbury
Atlantic/Faroe	Europe/Monaco	Pacific/Fakaofo
Atlantic/Madeira	Europe/Moscow	Pacific/Fiji
Atlantic/Reykjavik	Europe/Oslo	Pacific/Funafuti
Atlantic/South_Georgia	Europe/Paris	Pacific/Galapagos
Atlantic/St_Helena	Europe/Podgorica	Pacific/Gambier
Atlantic/Stanley	Europe/Prague	Pacific/Guadalcanal
Australia/Adelaide	Europe/Riga	Pacific/Guam
Australia/Brisbane	Europe/Rome	Pacific/Honolulu
Australia/Broken_Hill	Europe/Samara	Pacific/Kiritimati
Australia/Currie	Europe/San_Marino	Pacific/Kosrae
Australia/Darwin	Europe/Sarajevo	Pacific/Kwajalein
Australia/Eucla	Europe/Saratov	Pacific/Majuro
Australia/Hobart	Europe/Simferopol	Pacific/Marquesas
Australia/Lindeman	Europe/Skopje	Pacific/Midway
Australia/Lord_Howe	Europe/Sofia	Pacific/Nauru
Australia/Melbourne	Europe/Stockholm	Pacific/Niue
Australia/Perth	Europe/Tallinn	Pacific/Norfolk
Australia/Sydney	Europe/Tirane	Pacific/Noumea
Europe/Amsterdam	Europe/Ulyanovsk	Pacific/Pago_Pago
Europe/Andorra	Europe/Uzhgorod	Pacific/Palau
Europe/Astrakhan	Europe/Vaduz	Pacific/Pitcairn
Europe/Athens	Europe/Vatican	Pacific/Pohnpei
Europe/Belgrade	Europe/Vienna	Pacific/Port_Moresby
Europe/Berlin	Europe/Vilnius	Pacific/Rarotonga
Europe/Bratislava	Europe/Volgograd	Pacific/Saipan
Europe/Brussels	Europe/Warsaw	Pacific/Tahiti
Europe/Bucharest	Europe/Warsaw	Pacific/Tarawa
Europe/Budapest	Europe/Zagreb	Pacific/Tongatapu
Europe/Busingen	Europe/Zaporozhye	Pacific/Wake
Europe/Chisinau	Europe/Zurich	Pacific/Wallis
Europe/Copenhagen	Indian/Antananarivo	UTC

9.2. Library for motion Control

This library contains function blocks corresponding to the PLCopen standard “Function Blocks for Motion Control” definated function blocks for motion control.

DANGER!

- The function blocks included in this library must be used with "EtherCAT Task".
  If the function block is executed by other than "EtherCAT Task", unexpected behavior may occur.

9.2.1. Function block for single axis control

9.2.1.1. MC_Power

MC_Power is designed for controlling the power stage (“on” or “off”).

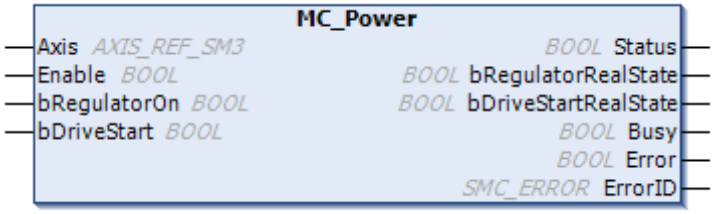


Fig.9.1 MC_Power

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Enable	BOOL	TRUE : Enables the execution of the FB.
bRegulatorOn	BOOL	TRUE : Enables the power stage.
bDriveStart	BOOL	TRUE : Disables the quickstop mechanism. Note: Both "MC_Power.bRegulatorON" and "MC_Power.bDriveStart" must be TRUE for servo on. After the axis has stopped due to a quick stop, set both bRegulatorOn and bDriveStart to FALSE before restarting operation.
VAR_OUTPUT		
Status	BOOL	TRUE : Axis is ready to move.
bRegulatorOnRealState	BOOL	TRUE : The power stage has been switched on.
bDriveStartRealState	BOOL	TRUE : Drive is not blocked by the quickstop mechanism.
Busy	BOOL	TRUE : Execution of the function block has not been finished yet.
Error	BOOL	TRUE : Error has occurred within the function block during execution.
ErrorID	SMC_ERROR	Error identification

9.2.1.2. MC_Reset

This function block designed for the transition from state errorstop to standstill by resetting all internal axis-related errors.

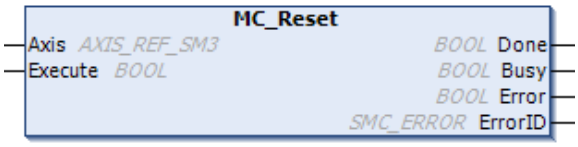


Fig.9.2 MC_Reset

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge：Starts the execution of the FB.
VAR_OUTPUT		
Done	BOOL	TRUE：Reset has been executed.
Busy	BOOL	TRUE：Execution of the function block has not been finished.
Error	BOOL	TRUE：Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.1.3. MC_Home

This function block triggers the “search home” sequence of an axis. Upon successful termination of the homing sequence the axis is in state “StandStill” .This is a precondition that absolute movements (e.g. wie MC_MoveAbsolute_J, MC_GearInPos) can be applied on an axis.

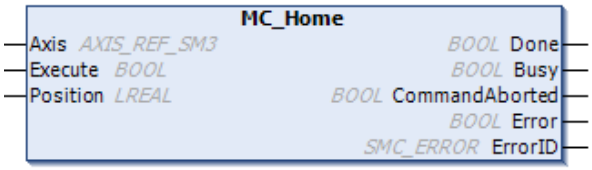


Fig.9.3 MC_Home

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Position	LREAL	Absolute position when the reference signal is detected [u].
VAR_OUTPUT		
Done	BOOL	TRUE：standstill has been achieved.
Busy	BOOL	TRUE：Execution of function block has not been finished.
CommandAborted	BOOL	TRUE：Command has been aborted by another command.
Error	BOOL	TRUE：Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.1.4. MC_Stop

MC_Stop places the axis in the stopping state. As a result, currently running motions of function block instances are aborted.（Please refer to 7.2.2.4The state diagram）。

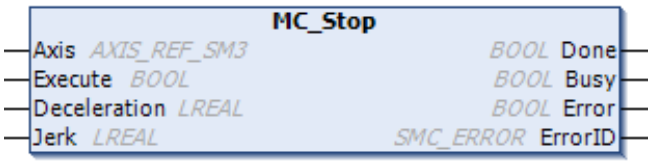


Fig.9.4 MC_Stop

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Deceleration	LREAL	Deceleration [User Unit/s^2]
Jerk	LREAL	Jerk always positive in [User Unit/s^2]
VAR_OUTPUT		
Done	BOOL	TRUE：Axis has reached the velocity 0.
Busy	BOOL	TRUE： Function block is in operation.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.1.5. MC_Halt

This function block stops the referenced axis in a controlled manner. If actions of other function blocks are running at this time, the actions are aborted.

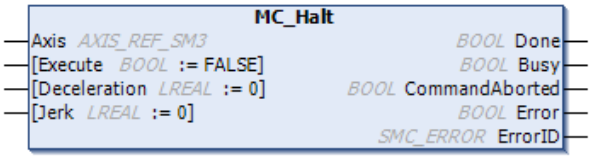


Fig.9.5 MC_Halt

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Deceleration	LREAL	Modulo value of the deceleration in [User Unit/s^2]
Jerk	LREAL	Jerk in [User Unit/s^3]
VAR_OUTPUT		
Done	BOOL	TRUE： Velocity 0 has been achieved
Busy	BOOL	TRUE： Function block is in operation.
CommandAborted	BOOL	TRUE： Execution has been interrupted by another function block instance operating on the axis.
Error	BOOL	TRUE:： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.1.6. MC_MoveAbsolute

This function block causes the axis to be moved to an absolute position and uses the values for Velocity, Deceleration, Acceleration and Jerk.

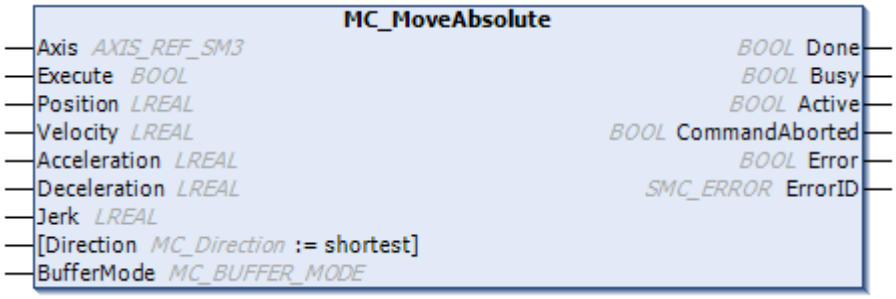


Fig.9.6 MC_MoveAbsolute

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Position	LREAL	Target position of the motion [User Unit]
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
Jerk	LREAL	Jerk in [User Unit/s^3]
Direction	MC_Direction	Direction of movement **fastest**: Automatically select the one that reaches the target position faster (modulo axis only) **current**: Current movement direction (modulo axes only) **positive**: positive rotation direction **shortest**: Shortest direction (modulo axis only) **negative**: Negative rotation direction
BufferMode	MC_BUFFER_MODE	Define the time series sequence of FB to the previous block. BufferMode=Aborting is only allowed if the FB is busy.
VAR_OUTPUT		
Done	BOOL	TRUE: End position has been achieved.
Busy	BOOL	TRUE: Function block is in operation.
Active	BOOL	State in which the function block controls the axis
CommandAborted	BOOL	TRUE: The execution is interrupted by an other function block.
Error	BOOL	TRUE: Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.1.7. MC_MoveRelative

This function block commands a controlled motion of a specified distance relative to the set position at the time of the execution. The motion ends with velocity is 0.

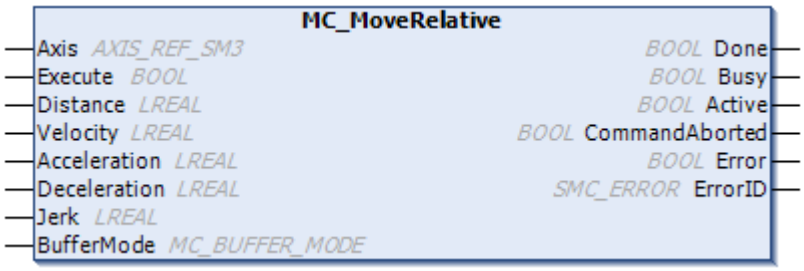


Fig.9.7 MC_MoveRelative

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Distance	LREAL	Relative distance for the motion in technical unit [User Unit]
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
Jerk	LREAL	Jerk always positive in [User Unit/s^3]
BufferMode	MC_BUFFER_MODE	Define the time series sequence of FB to the previous block. BufferMode=Aborting is only allowed if the FB is busy.
VAR_OUTPUT		
Done	BOOL	TRUE： End position has been achieved.
Busy	BOOL	TRUE： Function block is in operation.
Active	BOOL	State in which the function block controls the axis
CommandAborted	BOOL	TRUE： Execution has been interrupted by another function block instance operating on the axis.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.1.8. MC_MoveAdditive

This function block causes a controlled motion that adds the specified distance to the last specified target position. The axis is thereby in the discrete_motion mode. The current target position can result from a preceding motion of MC_MoveAdditive that was aborted. If the function block runs in the continuous_motion mode, the specified distance is added to the current position during the processing time.

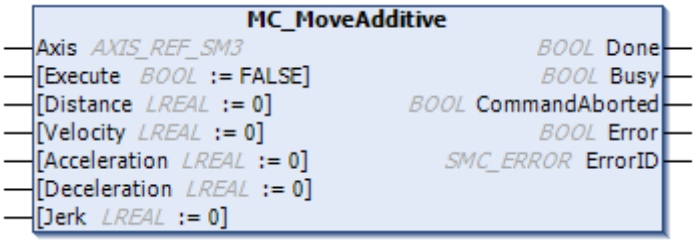


Fig.9.8 MC_MoveAdditive

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Distance	LREAL	Relative distance for the motion in technical unit [User Unit]
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
Jerk	LREAL	Is always positive in [User Unit/s^3]
VAR_OUTPUT		
Done	BOOL	TRUE： Distance has been achieved.
Busy	BOOL	TRUE： Function block is in operation.
CommandAborted	BOOL	TRUE： Execution has been interrupted by another function block instance operating on the axis.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.1.9. MC_MoveVelocity

This function block causes an endless motion at a specified velocity.

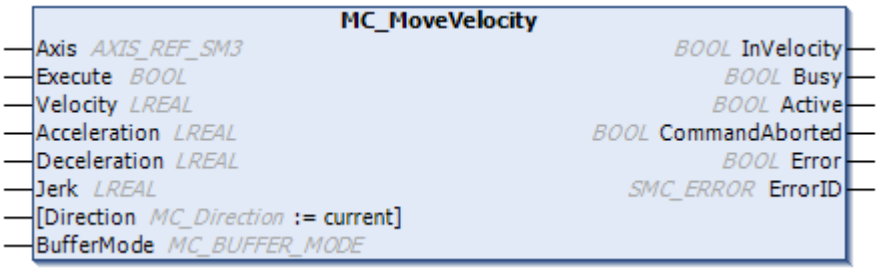


Fig.9.9 MC_MoveVelocity

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
Jerk	LREAL	Jerk [User Unit/s^3]
Direction	MC_Direction	Permitted values for MC_DIRECTION ・positive ・negative ・current ・shortest: Sets to a track that describes the ・shortest path. The choice of direction is based on the position at the time of the command call.
BufferMode	MC_BUFFER_MODE	Define the time series sequence of FB to the previous block. BufferMode=Aborting is only allowed if the FB is busy.
VAR_OUTPUT		
InVelocity	BOOL	TRUE： The set velocity has been reached for the first time.
Busy	BOOL	TRUE： Function block is in operation.
Active	BOOL	State in which the function block controls the axis
CommandAborted	BOOL	TRUE： Execution has been interrupted by another function block instance operating on Axis.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.1.10. MC_Jog

MC_Jog causes a continuous motion on the axis.

Fig. 9.1 MC_Jog

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
JogForward	BOOL	TRUE： Axis is moved with the specified dynamic values Velocity, Acceleration, Deceleration and Jerk in a positive direction.
JogBackward	BOOL	TRUE： Axis is moved with the specified dynamic values Velocity, Acceleration, Deceleration and Jerk in a negative direction. No motion is executed if JogForward is TRUE at the same time.
Velocity	LREAL	Velocity in [User Unit/s]
Acceleration	LREAL	Acceleration in [User Unit/s^2]
Deceleration	LREAL	Deceleration in [User Unit/s^2]
Jerk	LREAL	Jerk in [User Unit/s^3]
VAR_OUTPUT		
Busy	BOOL	TRUE： Function block is in operation during an active motion after JogForward or JogBackward has been set. FALSE： Axis has been decelerated to velocity value zero after JogForward or JogBackward has been set to FALSE.
CommandAborted	BOOL	TRUE： Execution is interrupted by another function block instance operating on Axis. CommandAborted remains set as long as JogForward or JogBackward has been set but for at least one cycle`.
Error	BOOL	TRUE: Error has occurred while JogForward or JogBackward has been set for at least one cycle
ErrorID	SMC_ERROR	Error identification

9.2.1.11. SanHome

This function block is used to perform homing with SAN MOTION EtherCAT slave. The basic operation is the same as MC_Home (FB). It is included in IoSanyoDevice.lib.

Fig.9.10 MC_Home

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Position	LREAL	Absolute position when the reference signal is detected [u].
VAR_OUTPUT		
Done	BOOL	TRUE: standstill has been achieved.
Busy	BOOL	TRUE: Execution of function block has not been finished.
CommandAborted	BOOL	TRUE: Command has been aborted by another command.
Error	BOOL	TRUE: Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.2. PTP control function block

9.2.2.1. MC_Power_SML

MC_Power is designed for controlling the power stage (“on” or “off”).

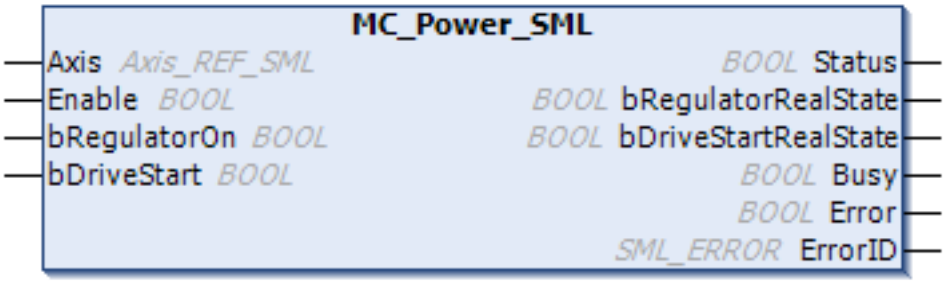


Fig 9.11 MC_Power_SML

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Enable	BOOL	TRUE : Enables the execution of the FB.
bRegulatorOn	BOOL	TRUE : Enables the power stage.
bDriveStart	BOOL	TRUE : Disables the quickstop mechanism. Note: Both "MC_Power.bRegulatorON" and "MC_Power.bDriveStart" must be TRUE for servo on. After the axis has stopped due to a quick stop, set both bRegulatorOn and bDriveStart to FALSE before restarting operation.
VAR_OUTPUT		
Status	BOOL	TRUE : Axis is ready to move.
bRegulatorOnRealState	BOOL	TRUE : The power stage has been switched on.
bDriveStartRealState	BOOL	TRUE : Drive is not blocked by the quickstop mechanism.
Busy	BOOL	TRUE : Execution of the function block has not been finished yet.
Error	BOOL	TRUE : Error has occurred within the function block during execution.
ErrorID	SMC_ERROR	Error identification

9.2.2.2. MC_Reset_SML

This function block designed for the transition from state errorstop to standstill by resetting all internal axis-related errors.

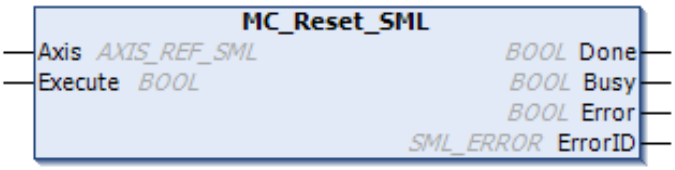


Fig 9.12 MC_Reset_SML

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge：Starts the execution of the FB.
VAR_OUTPUT		
Done	BOOL	TRUE： Reset has been executed.
Busy	BOOL	TRUE： Execution of the function block has not been finished.
Error	BOOL	TRUE： Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.2.3. MC_Home_SML

This function block triggers the “search home” sequence of an axis. Upon successful termination of the homing sequence the axis is in state “StandStill” .This is a precondition that absolute movements (e.g. wie MC_MoveAbsolute_J, MC_GearInPos) can be applied on an axis.

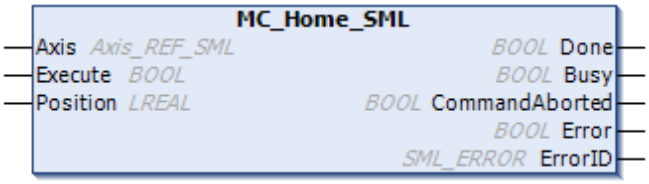


Fig 9.13 MC_Home_SML

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB..
Position	LREAL	Absolute position when the reference signal is detected [u].
VAR_OUTPUT		
Done	BOOL	TRUE: standstill has been achieved.
Busy	BOOL	TRUE: Execution of function block has not been finished.
CommandAborted	BOOL	TRUE: Command has been aborted by another command.
Error	BOOL	TRUE: Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

Refer to "8.2.3 Cancellation of MC_Home_SML" for restrictions on canceling the homing operation.

9.2.2.4. MC_Stop_SML

MC_Stop places the axis in the stopping state. As a result, currently running motions of function block instances are aborted.（Please refer to 7.2.2.4The state diagram）

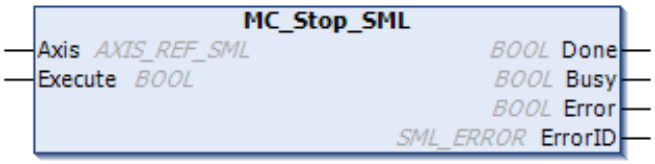


Fig.9.14 MC_Stop_SML

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
VAR_OUTPUT		
Done	BOOL	TRUE：Axis has reached the velocity 0.
Busy	BOOL	TRUE： Function block is in operation.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.2.5. MC_Halt_SML

This function block stops the referenced axis in a controlled manner. If actions of other function blocks are running at this time, the actions are aborted.

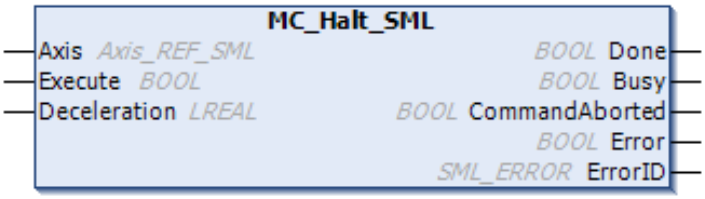


Fig.9.15 MC_Halt

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Deceleration	LREAL	Modulo value of the deceleration in [User Unit/s^2]
VAR_OUTPUT		
Done	BOOL	TRUE： Velocity 0 has been achieved
Busy	BOOL	TRUE： Function block is in operation.
CommandAborted	BOOL	TRUE： Execution has been interrupted by another function block instance operating on the axis.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

Only available when operating in profile velocity mode. Use MC_Stop_SML to stop profile position mode and homing mode operation.

9.2.2.6. MC_MoveAbsolute_SML

This function block causes the axis to be moved to an absolute position and uses the values for Velocity, Deceleration, Acceleration and Jerk.

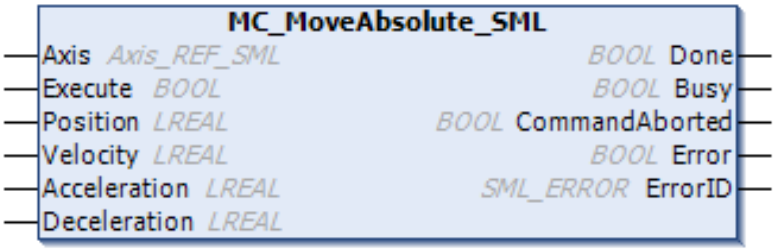


Fig.9.16 MC_MoveAbsolute

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Position	LREAL	Target position of the motion [User Unit]
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
VAR_OUTPUT		
Done	BOOL	TRUE: End position has been achieved.
Busy	BOOL	TRUE: Function block is in operation.
CommandAborted	BOOL	TRUE: The execution is interrupted by an other function block.
Error	BOOL	TRUE: Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.2.7. MC_MoveRelative_SML

This function block commands a controlled motion of a specified distance relative to the set position at the time of the execution. The motion ends with velocity is 0.

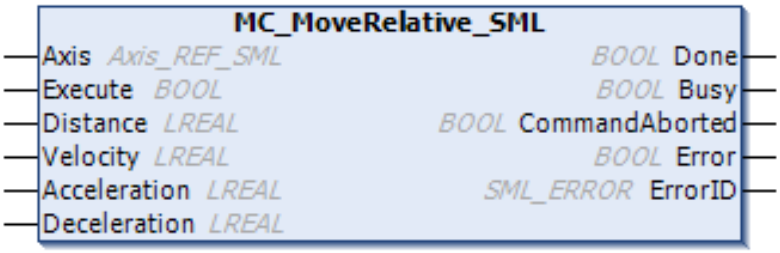


Fig 9.17 MC_MoveRelative

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Distance	LREAL	Relative distance for the motion in technical unit [User Unit]
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
VAR_OUTPUT		
Done	BOOL	TRUE: End position has been achieved.
Busy	BOOL	TRUE : Function block is in operation.
CommandAborted	BOOL	TRUE : Execution has been interrupted by another function block instance operating on the axis.
Error	BOOL	TRUE : Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.2.8. MC_MoveVelocity_SML

This function block causes an endless motion at a specified velocity.

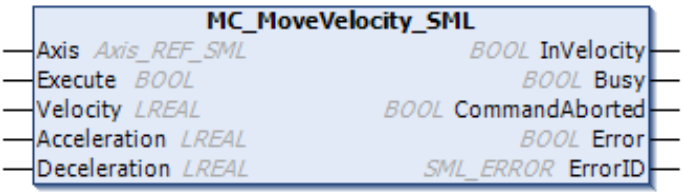


Fig 9.18 MC_MoveVelocity

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Velocity	LREAL	Maximum velocity [User Unit/s]
Acceleration	LREAL	Acceleration [User Unit/s^2]
Deceleration	LREAL	Deceleration [User Unit/s^2]
VAR_OUTPUT		
InVelocity	BOOL	TRUE: The set velocity has been reached for the first time.
Busy	BOOL	TRUE : Function block is in operation.
CommandAborted	BOOL	TRUE : Execution has been interrupted by another function block instance operating on Axis.
Error	BOOL	TRUE : Error has occurred.
ErrorID	SMC_ERROR	Error identification

After stopping the operation in the profile velocity mode by turning the servo off, when the servo is turned on again, the operation resumes at the velocity before stopping.

If you do not want to restart the operation when the servo is turned on, set the target velocity to 0 using one of the methods below.

1. Execute MC_Stop_SML during stop. (MC_Stop_SML will output an error)

2. Write 0 to "rpTargetVelocity.fVal" of the axis reference.

9.2.2.9. SML_SetOpmode

Sets the mode of operation to a new value, if necessary.

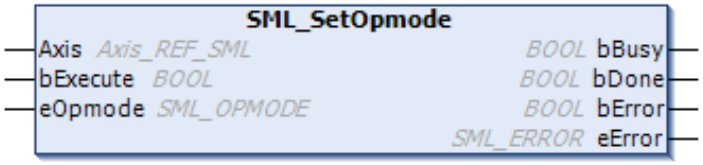


Fig 9.19 SML_SetOpmode

VAR_IN_OUT		
Axis	AXIS_REF_SML	Reference to the axis
VAR_INPUT		
bExecute	BOOL	Operates on the rising edge
eOpmode	SML_OPMODE	The desired mode of operation
VAR_OUTPUT		
bBusy	BOOL	The FB is not finished and new output values are to be expected
bDone	BOOL	The mode of operation has been successfully set
Error	BOOL	An error has occured
ErrorID	SMC_ERROR	Error number

In PTP control, when executing an FB of an operation mode different from the current operation mode, it is necessary to change the operation mode in advance. For example, you need to change to the homing mode before execute homing. If you want to perform position control after homing in homing mode, you need to change to profile position mode in advance.

9.2.3. Function block for multi-axis control

9.2.3.1. MC_GearIn

The function block couples the slave axis to the master axis specifying a certain velocity transmission ratio and applies a certain velocity ratio between master and slave velocity.

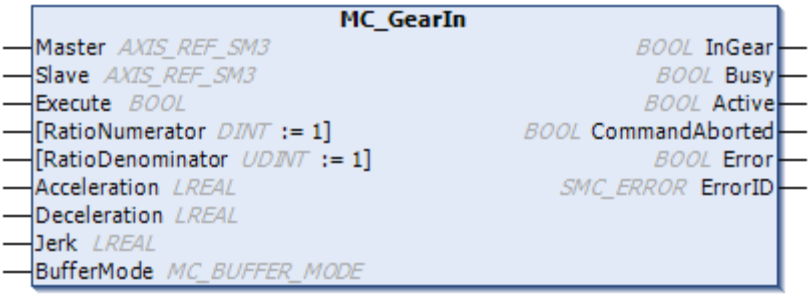


Fig.9.20 MC_GearIn

VAR_IN_OUT		
Master	AXIS_REF_SM3	Reference to master axis. Master needs not to be stationary.
Slave	AXIS_REF_SM3	Reference to slave axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
RatioNumerator	DINT	Numerator of the quotient for the desired transmission ratio
RatioDenominator	UDINT	Numerator of the quotient for the desired transmission ratio
Acceleration	LREAL	Target acceleration when coupling [User Unit/s^2] （>0）
Deceleration	LREAL	Target deceleration when coupling [User Unit/s^2] （>0）
Jerk	LREAL	Jerk in [User Unit/s^3]
BufferMode	MC_BUFFER_MODE	Define the time series sequence of FB to the previous block. BufferMode=Aborting is only allowed if the FB is busy.
VAR_OUTPUT		
InGear	BOOL	TRUE： Coupling has taken place.
Active	BOOL	State in which the function block controls the axis
Busy	BOOL	TRUE： Function block is in operation.
CommandAborted	BOOL	TRUE： Execution has been interrupted by another function block instance operating on the axis.
Error	BOOL	TRUE： Error has occurred.
ErrorID	SMC_ERROR	Error identification

9.2.3.2. MC_GearInPos

MC_GearInPos couples the slave axis to the master axis taking into account a specific positional relationship.

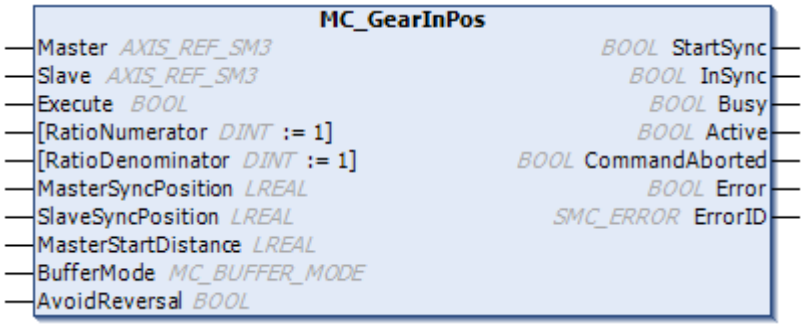


Fig.9.21 MC_GearInPos

VAR_IN_OUT		
Master	AXIS_REF_SM3	Reference to master axis
Slave	AXIS_REF_SM3	Reference to slave axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
RatioNumerator	DINT	Gear ratio numerator
RatioDenominator	DINT	Gear ratio denominator
MasterSyncPosition	LREAL	Master position where the axes run in sync. [User Unit]
SlaveSyncPosition	LREAL	Slave position where the axes run in sync. [User Unit]
MasterStartDistance	LREAL	Master distance for the gear in procedure (where the slave axis will be started for getting into synchronization). [User Unit]
BufferMode	MC_BUFFER_MODE	Define the time series sequence of FB to the previous block. BufferMode=Aborting is only allowed if the FB is busy.
AvoidReversal	BOOL	FALSE: Signals that the reversal of the slave is physically possible and acceptable. TRUE: Signals that the reversal of the module slave is physically impossible or might lead to damage.
VAR_OUTPUT		
StartSync	BOOL	TRUE : Commanded gearing has been started.
InSync	BOOL	TRUE : Cmmanded gearing has been completed.
Busy	BOOL	TRUE : Execution of the function block has not been finished.
Active	BOOL	State in which the function block controls the axis
CommandAborted	BOOL	TRUE : Command has been aborted by another command.
Error	BOOL	TRUE : Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.3.3. MC_GearOut

This function block disengages the slave axis from the master axis.

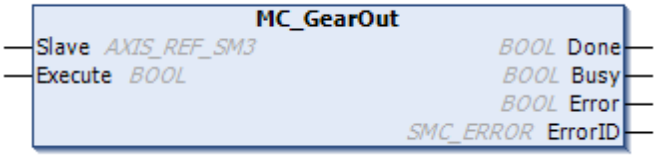


Fig.9.22 MC_GearOut

VAR_IN_OUT		
Slave	AXIS_REF_SM3	Reference to slave axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
VAR_OUTPUT		
Done	BOOL	TRUE： Cam has been disengaged.
Busy	BOOL	TRUE： Execution of the function block has not been finished.
Error	BOOL	TRUE： Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.3.4. MC_CamTableSelect

This function block is designed for selecting the cam tables by setting connections to relevant tables.

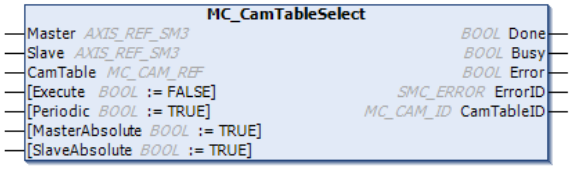


Fig.9.23 MC_CamTableSelect

VAR_IN_OUT		
Master	AXIS_REF_SM3	Reference to the master axis
Slave	AXIS_REF_SM3	Reference to the slave axis
CamTable	MC_CAM_REF	Reference to the cam description
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
Periodic	BOOL	TRUE: Periodic FALSE: Non periodic
MasterAbsolute	BOOL	TRUE: Absolute FALSE: Relative coordinates
SlaveAbsolute	BOOL	TRUE: Absolute FALSE: Relative coordinates
VAR_OUTPUT		
Done	BOOL	TRUE： Preselection has been done.
Busy	BOOL	TRUE： Execution of function block has not been finished.
Error	BOOL	TRUE: Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification
CamTableID	MC_CAM_ID	Identifier of the cam table be used for the function block.

9.2.3.5. MC_CamIn

This function block sets the cam table and implements synchronous operation.

Synchronize the slave axis with the master axis and control the slave axis with the set cam table.

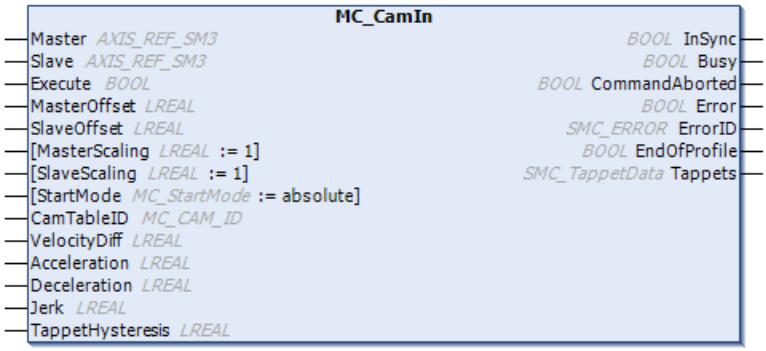


Fig.9.24 MC_CamIn

VAR_IN_OUT		
Master	AXIS_REF_SM3	Reference to master axis. Master need not be stationary.
Slave	AXIS_REF_SM3	Reference to slave axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
MasterOffset	LREAL	Offset on master table
SlaveOffset	LREAL	Offset on slave table
MasterScaling	LREAL	Scaling factor for master profile
SlaveScaling	LREAL	Scaling factor for slave profile
StartMode	MC_StartMode	Start mode
CamTableID	MC_CAM_ID	Identification of the cam plate. The input is connected with the output of the instance of MC_CamTableSelect.
VelocityDiff	LREAL	Maximum velocity difference for ramp_in mode in [User Unit/s]
Acceleration	LREAL	Acceleration for ramp_in mode in [User Unit/s^2]
Deceleration	LREAL	Deceleration for ramp_in mode in [User Unit/s^2]
Jerk	LREAL	Jerk for ramp_in mode in [User Unit/s^3]
TappetHysteresis	LREAL	Size of the hysteresis for tappets in [u].
VAR_OUTPUT		
InSync	BOOL	Cam has been engaged for the first time.
Busy	BOOL	TRUE： Execution of function block has not been finished.
CommandAborted	BOOL	Command has been aborted by another command
Error	BOOL	TRUE: Error has occured within the function block.
ErrorID	SMC_ERROR	Error identification
EndOfProfile	BOOL	Pulsed output: Cyclic end of the cam profile.
Tappets	SMC_TappetData	Tappets: Has to be evaluated by SMC_GetTappetValue function blocks.

9.2.3.6. MC_CamOut

This function block disengages the slave axis from the master axis immediately.

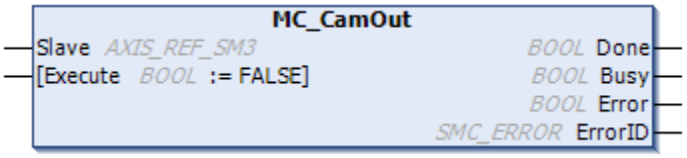


Fig.9.25 MC_CamOut

VAR_IN_OUT		
Slave	AXIS_REF_SM3	Reference to slave axis
VAR_INPUT		
Execute	BOOL	Rising edge: Starts the execution of the FB.
VAR_OUTPUT		
Done	BOOL	TRUE : Cam has been disengaged.
Busy	BOOL	TRUE : Execution of function block has not been finished.
Error	BOOL	TRUE: Error has occurred within the function block.
ErrorID	SMC_ERROR	Error identification

9.2.4. Function block for CNC control

9.2.4.1. SMC_Interpolator

This function block is used to convert a continuous path described by SMC_GEOINFO objects into discrete path position points taking into account a defined velocity profile and time pattern. Afterwards, these position points will typically be transformed by the IEC-program (e.g. to drive-axis-positions) and sent to the drives.

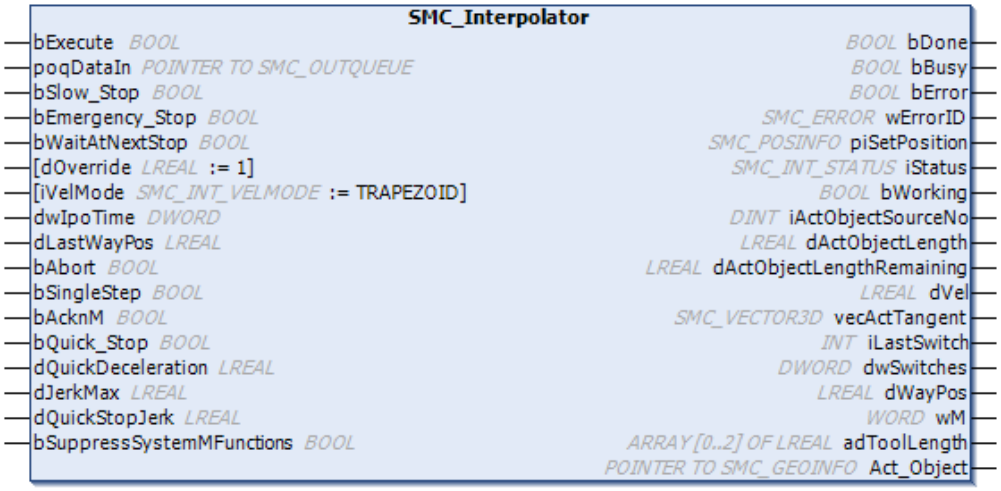


Fig.9.26 SMC_Interpolator

VAR_INPUT		
bExecute	BOOL	Rising edge: Starts the execution of the FB.
poqDataIn	POINTER TO SMC_OUTQUEUE	This variable points to the SMC_OUTQUEUE structure object, which contains the SMC_GEOINFO objects of the path; typically it points to the output poqDataOut of SMC_CheckVelocities .
bSlow_Stop	BOOL	If this variable is set to FALSE, the path will be passed non-stop. Otherwise, the SMC_Interpolator will be caused to reduce the velocity to 0 according to the defined velocity profile (byVelMode), and the maximum delay of the current SMC_GEOINFO object (dDecel, see below) and to wait until bSlow_Stop will be reset to FALSE.
bEmergency_Stop	BOOL	As soon as this input gets TRUE, the SMC_Interpolator will cause an immediate stop, this means that the position will be retained. Hence, the velocity will be set to 0 immediately.
bWaitAtNextStop	BOOL	As long as this variable is FALSE (default), the path is passed non-stop. Otherwise, the SMC_Interpolator will be caused to retain the position at the next regular stop this means at position points where the velocity is 0, typically at path angles and to pause until bWaitAtNextStop will be reset to FALSE.

VAR_INPUT		
dOverride	LREAL	This variable can be used to handle the override.
iVelMode	SMC_INT_VELMODE	This input defines the velocity profile as defined in SMC_INT_VELMODE.
dwIpoTime	DWORD	This variable has to be set for each call. It represents the cycle time in µsec.
dLastWayPos	LREAL	This input allows the user to measure the stretch of the path that is racked out by the interpolator. Output dWayPos is the sum of dLastWayPos and the distance covered within the current cycle.
bAbort	BOOL	This input set to TRUE will abort the function block.
bSingleStep	BOOL	This input effects that the interpolator will stop at the transition between two path objects (also at transitions with identical tangent) for the duration of one cycle.
bAcknM	BOOL	This input can be used to acknowledge an M-function. If the input is TRUE, the output wM will be cleared and the path processing will be continued.
bQuick_Stop	BOOL	If this input is TRUE, the interpolator will reduce the velocity to zero, until bQuick_Stop is reset to FALSE.
dQuickDeceleration	LREAL	Deceleration value used for bQuick_Stop [User Unit/s^2]
dJerkMax	LREAL	Magnitude of the maximum allowed jerk: It's only used for the quadratic velocity modes.
dQuickStopJerk	LREAL	The magnitude of the jerk is used by a quick stop for ramping down the acceleration if one of the quadratic velocity modes is selected.
bSuppressSystem MFunctions	BOOL	If this option is set, then the output wM will not be set for internal M-functions created by G75 or G4 commands.

VAR_OUTPUT		
bDone	BOOL	This variable will be set to TRUE as soon as the input data (poqDataIn) has been processed completely.
bBusy	BOOL	TRUE while execution of function block is not finished
bError	BOOL	Signals that an error has occurred within the function block
wErrorID	SMC_ERROR	Error identification
piSetPosition	SMC_POSINFO	It reflects the calculated set position and contains the cartesian coordinates of the next position as well as the state of the additional axis. SMC_POSINFO
iStatus	SMC_INT_STATUS	This enumeration variable reflects the current status of the function block defined in SMC_INT_STATUS .
bWorking	BOOL	This output is intended to be connected to input bEnable of SMC_ControlAxisByPos .
iActObjectSourceNo	DINT	Value of member iSourceLine_No of active SMC_GEOINFO object of poqDataIn-queue. (bWorking = FALSE), the value is set to “-1”.
dActObjectLength	LREAL	The length of the current object; valid if bWorking = TRUE.
dActObjectLength Remaining	LREAL	The remaning length of the current object; valid if bWorking = TRUE.
dVel	LREAL	This variable contains the current path velocity.
vecActTangent	SMC_VECTOR3D	This structure contains the path tangent, a unit vector.
iLastSwitch	INT	This output contains the number of the last switch passed.
dwSwitches	DWORD	This DWORD describes the current switch status of all switches 1 32.
dWayPos	LREAL	See input dLastWAyPos.
wM	WORD	If the interpolator passes an M-function, this output will be set to the value associated to the M-function.
adToolLength	ARRAY [0..2] OF LREAL	Parameters for tool length compensation
Act_Object	POINTER TO SMC_GEOINFO	A pointer to the currently interpolated path element

9.2.4.2. SMC_TRAFO_XXXXX

This function block solves Reverse transformations of the robot(TCP⇒each axis position). “XXXXX” contains the name of the kinematics to be Reverse transformatio.（e.g.：SMC_TRAFO_Gantry2）.

Fig.9.27 SMC_TRAFO_Gantry2

VAR_INPUT		
pi	SMC_PosInfo	Target vector position (x,y), output of interpolator.
dOffsetX	LREAL	Additional offset for x-axis
dOffsetY	LREAL	Additional offset for y-axis
VAR_OUTPUT		
dx	LREAL	Resulting position for x-axis
dy	LREAL	Resulting position for y-axis

9.2.4.3. SMC_TRAFOF_XXXXX

This function block solves Forward transformation of the robot(each axis position⇒TCP). “XXXXX” contains the name of the kinematics to be Reverse transformatio.（e.g.：SMC_TRAFOF_Gantry2）.

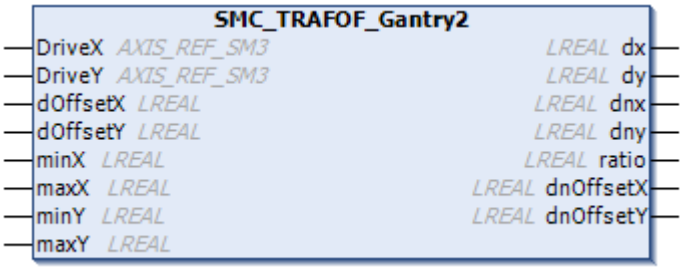


Fig.9.28 SMC_TRAFOF_Gantry2

VAR_IN_OUT		
DriveX	AXIS_REF_SM3	xReference to axis
DriveY	AXIS_REF_SM3	yReference to axis
VAR_INPUT		
dOffsetX	LREAL	Offset x-position. Equivalent to SMC_TRAFO_Gantry2
dOffsetY	LREAL	Offset y-position. Equivalent to SMC_TRAFO_Gantry2
minX	LREAL	Lower bound of move range in x-direction (for visualization purpose)
maxX	LREAL	Upper bound of move range in x-direction (for visualization purpose)
minY	LREAL	Lower bound of move range in y-direction (for visualization purpose)
maxY	LREAL	Upper bound of move range in y-direction (for visualization purpose)
VAR_OUTPUT		
dx	LREAL	X-position
dy	LREAL	Y-position
dnx	LREAL	Normed x-position (with value in [0,1])
dny	LREAL	Normed y-position (with value in [0,1])
ratio	LREAL	Ratio x-interval / y-interval
dnOffsetX	LREAL	X-offset for visualization
dnOffsetY	LREAL	Y-offset for visualization

9.2.4.4. SMC_ControlAxisByPos

The function block writes the set position fSetPosition to the drive structure Axis and monitors Axis for jumps. SMC_ControlAxisByPos is mostly used with CNC and an instance of the SMC_Interpolator.

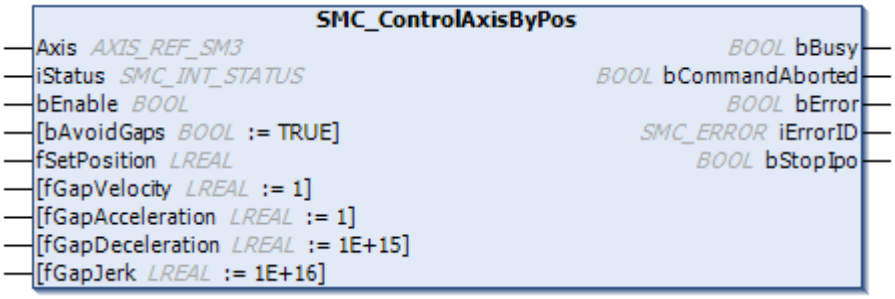


Fig.9.29 SMC_ControlAxisByPos

VAR_IN_OUT		
Axis	AXIS_REF_SM3	Reference to axis
VAR_INPUT		
iStatus	SMC_INT_STATUS	Status of the instance of SMC_Interpolator
bEnable	BOOL	TRUE : Starts execution
bAvoidGaps	BOOL	TRUE : Starts the monitoring of the position
fSetPosition	LREAL	Set position of the axis in [u]. Typically connected to the output of the transformation block
fGapVelocity	LREAL	Velocity for the bypassing of the jump in [User Unit/s]
fGapAcceleration	LREAL	Acceleration for the bypassing of the jump in [User Unit/s^2]
fGapDeceleration	LREAL	Deceleration for the bypassing of the jump in [User Unit/s^2]
fGapJerk	LREAL	Jerk for the bypassing of the jump in [User Unit/s^3]
VAR_OUTPUT		
bBusy	BOOL	TRUE : Function block operating
bCommandAborted	BOOL	TRUE : Execution was interrupted by another function block instance operating on axis.
bError	BOOL	TRUE : Error has occurred
iErrorID	SMC_ERROR	Error identification
bStopIpo	BOOL	TRUE : Jump in velocity or position occurred and adaptation to new position is running.

9.3. G code list

The following list shows the G codes that can be used with CNC.

Travel command	Description	Path element
G0	Direct movement without tool operation; linear motion	Positioning
G1	Linear movement with tool operation	Linear Motion
G2	Circular segment or circle, clockwise	Arc
G3	Circle segment or circle, counterclockwise	Arc
G4	Dwell time	Dwell Time
G5	Point of a 2D cardinal spline	Spline
G6	Parabola	Parabola
G8	Ellipse segment or ellipse, clockwise	Ellipse
G9	Ellipse segment or ellipse, counterclockwise	Ellipse
G10	Point of a 3D cardinal spline	Spline
G15	Switch to 2D	3D mode
G16	Switch to 3D by activating 3D mode with normal vector I/J/K to the plane	3D mode
G17	Switch to 3D by activating 3D mode in X/Y plane	3D mode
G18	Switch to 3D by activating 3D mode in Z/X plane	3D mode
G19	Switch to 3D by activating 3D mode in Y/Z plane	3D mode
G20	Conditional jump to L, if K <> 0	Jump
G36	Write value D to variable O	Changing Variable Values
G37	Increment variable O by value D	Jump
G40	End of tool radius compensation	Preprocessing
G41	Start of tool radius compensation, left of travel direction	Preprocessing
G42	Start of tool radius compensation, right of travel direction	Preprocessing
G43	Starts tool length compensation.	Preprocessing
G50	End of angle rounding/smoothing	Preprocessing
G51	Start of angle rounding	Preprocessing
G52	Start of angle smoothing	Preprocessing
G53	End the coordinate transformation and resets the decoder coordinate system to the original position (= machine coordinate system).	Shifting, Rotating, and Scaling the Coordinate System
G54	Absolute transformation of the coordinates.	Shifting, Rotating, and Scaling the Coordinate System
G55	Relative transformation of the coordinates.	Shifting, Rotating, and Scaling the Coordinate System
G56	Sets the current orientation, position, and scaling of the DCS is set as a reference point.	Shifting, Rotating, and Scaling the Coordinate System
G60	End of loop suppression	Preprocessing
G61	Start of loop suppression	Preprocessing
G70	End of smoothing additional axes.	Preprocessing
G71	Start of smoothing additional axes.	Preprocessing
G75	Timing synchronization with interpolator	Timing Synchronization with Interpolator

Travel command	Description	Path element
G90	The coordinates (X/Y/Z/A/B/C/P/Q/U/V/W) are interpreted as absolute values. (This is the default setting.)	Modes
G91	The coordinates (X/Y/Z/A/B/C/P/Q/U/V/W) are interpreted as values relative to the current position.	Modes
G92	Positioning by jump	Positioning
G98	The axis midpoints (I/J/K) are interpreted as absolute values.	Modes
G99	The axis midpoints (I/J/K) are interpreted as values relative to the start position. (This is the default setting.)	Modes

9.4. Instruction

The instructions used in the ST language are as follows.

9.4.1. IF

The IF instruction is used to check a condition and, depending on this condition, to execute instructions.

Syntax:

```
IF <boolean expression_1>    THEN
 <IF-instructions>
 {ELSIF <boolean expression_2> THEN
 <ELSIF-instruction_1>

 ELSIF <boolean expression_n> THEN
 <ELSIF_instruction_n-1>
 ELSE
 <ELSE_instructions>}
 END_IF;
```

The section inside the curly parentheses {} is optional.

If <boolean expression_1> returns TRUE, Controller executes only the <IF_instructions> and none of the other instructions.

Otherwise Controller checks the boolean expressions in succession, starting with <boolean expression_2, until an expression returns TRUE. Subsequently, Controller evaluates all instructions located between this expression and the next ELSE or ELSIF instruction and executes them accordingly.

If none of the boolean expressions returns TRUE, Controller evaluates only the <ELSE instructions>.

9.4.2.CASE

Use this dialog box for pooling several conditional instructions containing the same condition variable into a construct.

Syntax:

CASE <Var1> OF

<value1>:<instruction1>

<value2>:<instruction2>

<value3, value4, value5>:<instruction3>

<value6 ... value10>:<instruction4>

...

<value n>:<instruction n>

{ELSE <ELSE-instruction>}

END_CASE;

The section within the curly brackets {} is optional.

.Processing scheme of a CASE instruction.

- If the value of the variable <Var1> is <value i>, then the instruction <instruction i> is executed.
- If the variable <Var1> has non of the given values, then the <ELSE-instruction> is executed.
- If the same instruction is executed for several values of the variable, then you can write the values in sequence, seperated by commas.

9.4.3.FOR

The FOR loop is used to execute instructions with a certain number of repetitions.

Syntax:

FOR <counter> := <start value> TO <end value> {BY <increment> } DO

<instructions>

END_FOR;

The section inside the curly parentheses {} is optional.

Controller executes the <instructions> as long as the <counter> is not greater, or - in case of negative increment - is not smaller than the <end value>. This is checked before the execution of the <instructions>.

Every time the instructions <instructions> have been executed, the counter <counter> is automatically increased by the increment <increment>. The increment <increment> can have any integral value. If you do not specify an increment, the standard increment is 1.

The end value <end value> may not attain the same value as the upper limit of the data type of the counter.

For example, an endless loop results in the above example if counter is of the data type SINT and the <end value> equals 127, since the data type SINT has the upper limit 127.

9.4.4.WHILE

The WHILE loop is used like the FOR loop in order to execute instructions several times until the abort condition occurs. The abort condition of a WHILE loop is a boolean expression.

Syntax:

WHILE <boolean expression> DO

<instructions>

END_WHILE;

Controller repeatedly executes the <instructions> for as long as the <boolean expression> returns TRUE. If the boolean expression is already FALSE at the first evaluation, then Controller never executes the instructions. If the boolean expression never adopts the value FALSE, then the instructions are repeated endlessly, as a result of which a runtime error results.

9.4.5. REPEAT

The REPEAT loop is used like the WHILE loop, but with the difference that Controller only checks the abort condition after the execution of the loop. The consequence of this behavior is that the REPEAT loop is executed at least once, regardless of the abort condition.

Syntax:

```
REPAEAT
<instructions>
 UNTIL <boolean expression>
 END_REPEAT;
```

Controller executes the <instructions> until the <boolean expression> returns TRUE.

If the boolean expression already returns TRUE at the first evaluation, Controller executes the instructions precisely once. If the boolean expression never adopts the value TRUE, then the instructions are repeated endlessly, as a result of which a runtime error results.

9.4.6. EXIT

The EXIT instruction is used in a FOR, WHILE or REPEAT loop in order to end the loop regardless of other abort conditions.

Syntax:

```
WHILE TURE DO
   IF bBreak THEN
      EXIT;
   END_IF
END_WHILE
```

In this example, when “bBreak” becomes TRUE, it exits the WHILE loop.

9.4.7.RETURN

Use the RETURN instruction in order to exit from a function block. You can make this dependent on a condition, for example.

Syntax:

```
POU1

    POU2

    WHILE TURE DO
      IF bReturn THEN
        RETURN;
      END_IF
    END_WHILE

    (*This is an unexecuted process *)
    IF boolean expression THEN
      < Process >
    END_IF

    ●

    POU3
```

In this example, when “bReturn” becomes TRUE in POU 2, it goes out of POU 2 and moves to ●. Since RETURN exits POU when it is executed, processes after RETURN are not executed.

9.5. Cast

Information can be lost when converting from larger data types to smaller data types. At SANMOTION C Software Tool 2.0.0（e.g. INT type to BYTE type, DINT type to WORD type, etc）. If you want to convert, a function for type conversion is necessary.

The name of the type conversion function is configured as follows.

“Data type before conversion”_TO_”Data type after conversion”

Open the Input Assistant dialog box by clicking “Edit” ⇒ “Input Assistant” Click “Conversion Operators” to display a list of function for type conversion.

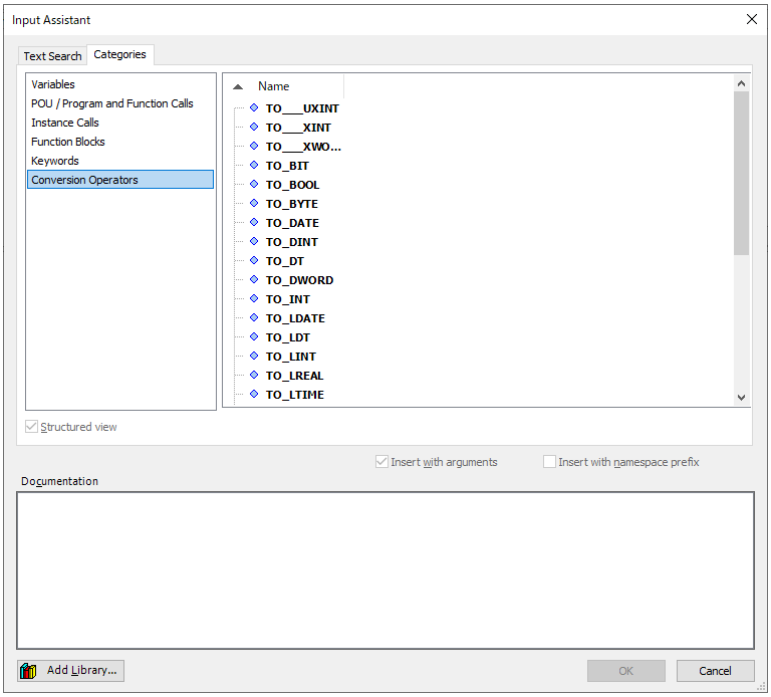


Fig.9.30 Input Assistant

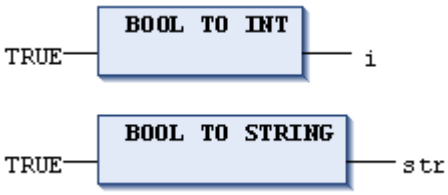
＜Example＞

In case of ST：

i := BOOL_TO_INT(TRUE); (The result is 1 *)*

str := BOOL_TO_STRING(TRUE); (The result is 'TRUE' *)*

In case of FBD：



(* The result is 1 *)

(* The result is 'TRUE' *)

9.6. Operators

9.6.1. List

Operators	Detail
'	String delimiter (e.g. 'string1')
[..]	Specify array range (e.g. ARRAY [0..3] OF)
:	Operands and types in declarative part (e.g. var1 : INT;)
;	Command end symbol (e.g. var1 : INT;)
^	Indirect reference symbol of pointer (e.g. pointer1^)
AND	This IEC operator is used for the bitwise AND of bit operands.
OR	This IEC operator is used for the bitwise OR of bit operands.
XOR	This IEC operator is used for the bitwise XOR of bit operands.
NOT	This IEC operator is used for the bitwise NOT of a bit operand.
+, ADD	This IEC operator is used for adding variables.
-, SUB	This IEC operator is used for subtracting variables.
*, MUL	This IEC operator is used for multiplying variables.
/, DIV	This IEC operator is used for dividing variables.
>, GT	This IEC operator is used for the “greater than” function.
>=, GE	This IEC operator is used for the “greater than or equal to” function.
=, EQ	This IEC operator is used for the “equals” function.
<>, NE	This IEC operator is used for the “does not equal” function.
<=, LE	This IEC operator is used for the “less than or equal to” function.
<, LT	This IEC operator is used for the “less than” function.
MOD(in)	This IEC operator is used for modulo division.
INDEXOF(in)	This operator is an extension of the IEC 61131-3 standard.
SIZEOF(in)	This operator is an extension of the IEC 61131-3 standard.
SHL(K,in)	This IEC operator is used for bitwise shift of an operand to the left.
SHR(K,in)	This IEC operator is used for bitwise shift of an operand to the right.
ROL(K,in)	This IEC operator is used for bitwise rotation of an operand to the left.
ROR(K,in)	This IEC operator is used for bitwise rotation of an operand to the right.
MAX(in0,in1)	This IEC operator is used for the maximum function. It yields the largest value of two values.
MIN(in0,in1)	This IEC operator is used for the minimum function. It yields the smallest value of two values.
LIMIT(MIN,in,Max)	This IEC selection operator is used for limiting.
MUX(K,in0,...in_n)	This IEC operator is used as a multiplexer.
ADR(in)	ADR yields the address of its argument in a DWORD.
ADRINST()	Output address of instance of function block
BITADR(in)	BITADR yields the bit offset within a segment in a DWORD.
ABS(in)	This IEC operator yields the absolute value of a number.
SQRT(in)	This IEC of course yields the square root of a number.
LN(in)	This IEC operator yields the natural logarithm of a number.
LOG(in)	This IEC operator yields the base-10 logarithm of a number.
EXP(in)	This IEC operator yields the exponential function.
SIN(in)	This IEC operator yields the sine value of a number.
COS(in)	This IEC operator yields the cosine value of a number.
EXP(in)	This IEC operator yields the exponential function.
SIN(in)	This IEC operator yields the sine value of a number.

Operators	Detail
COS(in)	This IEC operator yields the cosine value of a number.
TAN(in)	This IEC operator yields the tangent value of a number.
ASIN(in)	This IEC operator yields the arcsine value of a number.
ACOS(in)	This IEC operator yields the arccosine value of a number. The value is computed in radians.
ATAN(in)	This IEC operator yields the arctangent value of a number. The value is computed in radians.
EXPT(in,expt)	This IEC operator raises a number to a higher power and returns the power of the base raised to the exponent: power = in ^{exponent}.
LEN(in)	Returns the number of characters of a string
LEFT(str,size)	Returns a specific number of characters of a string, starting from left
RIGHT(str,size)	Returns a specific number of characters of a string, starting from right
MID(str,size,pos)	Returns a specific number of characters of a string, starting from a specific position
CONCAT('str1','str2')	CONCAT(STR1,STR2) means: Connect STR1 and STR2 to a single string STR1STR2.
INSERT('str1','str2',pos)	Inserts a string into another string at a specific position
DELETE('str1',len,pos)	This function block deletes a file.
REPLACE('str1','str2',len, pos)	Replaces a specific number of characters of a string by another string
FIND('str1','str2')	Searches for the position of a partial string within a string.
SR	FB: Realizes a bistable set-dominat latch
RS	FB: Realizes a bistable reset-dominat latch
SEMA	FB: Semaphore
R_TRIG	FB: Detects a rising edge of a boolean signal
F_TRIG	FB: Detects a falling edge of a boolean signal
CTU	FB: Increments a given value
CTD	FB: Decrements a given value
CTUD	FB: Increments and decrements a given value
TP	FB: Implements a pulse timer
TON	FB: Implements a timer with a turn-on delay
TOF	FB: Implements a timer with a turn-off delay
RTC	FB: Calculates the elapsed time since a given start time

9.6.2.Priority

In the calculation of expressions, operators are processed according to operator precedence.After the highest priority operation is processed, the next highest priority join operation is performed.If there is an operation with the same priority, it is processed from left to right.Below is a list in descending order of operator precedence.

Calculation	Symbol	Priority
In parentheses	(Formula)	High
Function call Function name	Function name (Parameter)	↑
Exponentiation operation	EXPT	
Negative number	-	
Not	NOT	
Multiplication	*	
Division	/	
Remainder	MOD	
Addition	+	
Subtraction	-	
Comparison	<, >, <=, >=	
Equal sign	=	
Inequality sign	<>	
AND	AND	
XOR	XOR	↓
OR	OR	Low

9.7. Pointer

Pointers store the addresses of variables, programs, function blocks, methods and functions while an application program is running. By using pointers, you can execute processes efficiently.

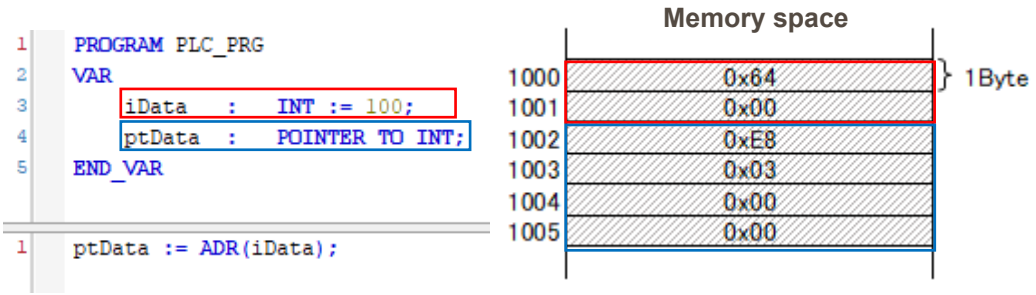


Fig.9.31 Using pointers

For example, when writing to POU as shown in the upper left figure, memory is reserved as shown in the upper right figure. In this example, the values of each variable are as follows.

Variable	Detail	Value
iData	Value of iData	100
ADR(iData)	Address of iData	1000
ptData	Value of ptData	1000
ADR(ptData)	Address of ptData	1002
ptData^	The value stored in the address of set in ptData	100

By using the pointer in the following cases, the process can be executed efficiently.

1) Data transfer

As shown above, pointers pass and receive addresses. For example, declare large data such as "ARRAY [0..9999] OF LREAL" as input variables of FB, and data transfer. In this case, CPU resources and memory are used inefficiently.

Therefore, in case of passing large data, pass only the address using the pointer. As a result, only 4 bytes of the address are used for memory, without passing all the data of the array. Accelerate processing and saving memory can be realized.

2) Passing by reference to FUN

Only one output data of FUN can be set. Therefore, the address of the storage destination is passed to FUN by the pointer. And the data is written to that pointer variable. As a result, multiple data can be output even in FUN.

3) Converting Data Types

By using a pointer, the data stored in the input address can be referred to with the declared data type.As a result, you can refer to the data you passed the address with a different data type.For example, if you pass STRING(10) := '0123456789' address' to POINTER TO ARRAY [1..10] OF BYTE, the data is referenced like "30h,31h,..38h,39h"

9.8. Confirm CPU utilizationCPU

CPU utilization is the load on the CPU. If the CPU utilization is high, the process can not be completely processed, possibly not conforming to the program. Therefore, it is necessary to grasp the CPU load at execution of the created application.

There are two ways to check CPU utilization as shown below.

1) Check from the PLC shell

This tab of the generic device editor includes a text-based control monitor for querying specific information from the S200. You can specify device-dependent commands for this and receive the response from the controller in a result window.

When you select the PLC shell in the "Device" tab, the following window will be displayed.

Fig.9.32 PLC shell window

Click "..." at the bottom of the window to display the command list. Please select "plcload". After entering the command, move the cursor to the command input field and press "Enter" to send the command. The CPU utilization of each task is displayed in the result window.

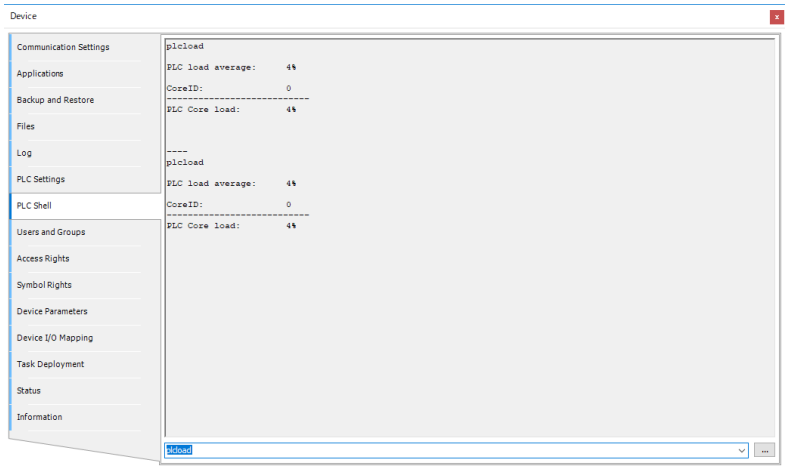
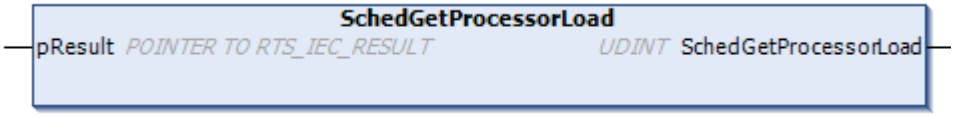


Fig.9.33 Confirm CPU utilization by PLC shell window

"PLC Core load" shows the total CPU utilization of each task.If "PLC Core load" exceeds 70%, you need to review the program.

2) Check from SchedGetProcessorLoad (FUN)

One of the functions contained in CmpSchedule.lib is SchedGetProcessorLoad. SchedGetProcessorLoad is a UDINT type function that returns the current CPU utilization.



VAR_INPUT		
pResult	POINTER TO RTS_IEC_RESULT	Returns the execution result of the function.
VAR_OUTPUT		
SchedGetProcessorLoad	UDINT	Returns the current CPU usage.

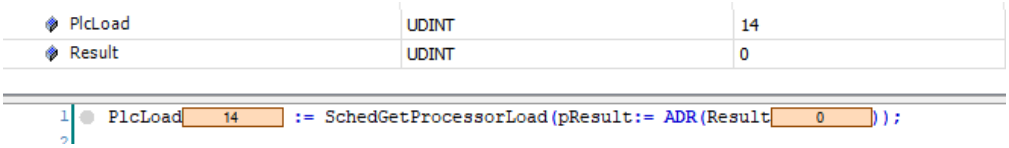


Fig.9.34 Example of using SchedGetProcessorLoad

The value returned by SchedGetProcessorLoad will be the total CPU usage of each task. Therefore, if the return value exceeds 70%, it is necessary to review the program.

If the CPU load is heavy, the connection with the development PC and the control buttons may not work. In that case, you can prevent the application from starting by holding down the control button during startup.

Since the application will not start, the CPU load will be reduced and you will be able to log in. Please use the debug function to correct the cause of the CPU load.

9.9. Language selection

The display language of SANMOTION C Software Tool 2.0.0 can be changed from "Tools" ⇒ "Options" ⇒ "International settings" on the menu bar.

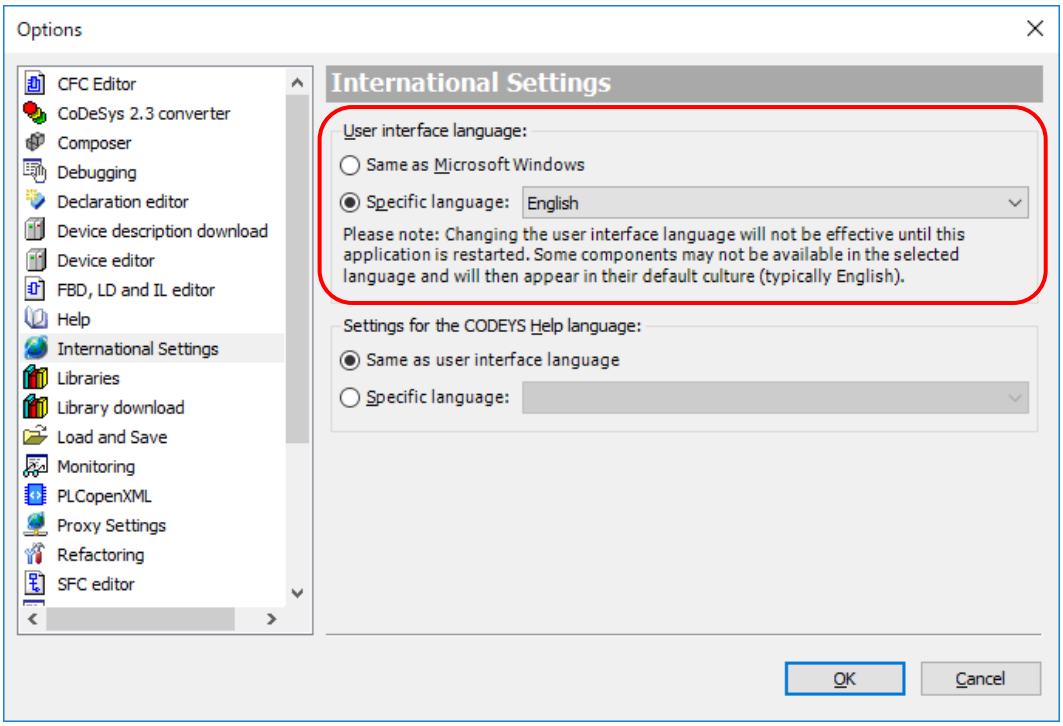


Fig.9.35 Options window

9.10. Rules for identifier designation

9.10.1. Characters that can be used

The rules for characters that can be used for variable names and POU names are described below.

- Single-byte alphanumeric characters can be used, but there is no distinction between uppercase and lowercase letters. A number cannot be used as the first character. As an example, VAR1 and var1 refer to the same variable.
- Identifiers cannot contain spaces or special characters. Only underscores (_) are allowed. Consecutive underscores are not allowed.
- Reserved words cannot be used. (Example: IF, AND, etc.)

Character type	Usable characters	Note
Alphabet	a ~ z, A ~ Z	Not case sensitive
Numbers	0 ~ 9	Do not use the first letter of the name
Special characters	_	Prohibition of continuous use

9.10.2. Recommendations on how to assign identifiers

Use Hungarian Notation for naming conventions for variables in your applications and libraries whenever possible. The base name should be a meaningful short description for each variable. The first letter of each word in the base name must be uppercase, the rest must be lowercase. You can create translation files for other languages as needed. Prefix the base name with a lowercase prefix that corresponds to the data type of the variable.

Data type	Prefix	Comment
BOOL	x, b	xExecute, bExecute
BYTE	by	byData
WORD	w	wBuff
DWORD	dw	dwAddress
SINT	si	siCnt
USINT	usi	usiNum
INT	i	iParam
UINT	ui	uiLen
DINT	di	diPlus
UDINT	udi	udiCounter
REAL	r	rOvl
LREAL	lr	lrPos
STRING	s	sFileName
TIME	tim	timDelay
ENUM	e	eColor
POINTER	p	pSendData
ARRAY	a	abyTelegramData
STRUCT	Library prefix Example: CAN	CAN_SDOTelegram A short description of the structure is recommended for the base name
POU	Library prefix Example: CAN	CAN_SendTelegram It is recommended that the base name consist of a short description of the POU with verbs and nouns

10. Technical data

10.1. Functional specifications

Item		S200 series	
		SMC200-A	SMC200-B
Memory	Program size	8MB	
	Retain variable size	500KB	
	RAM	1GB	
Task	Minimum calculation cycle	2ms~	
	Minimum motion control cycle	2ms~	
	Minimum robot control cycle	8ms~	
	Maximum number of tasks	32	
Programming	Programming language	IEC61131-3 standard 5 languages + CFC LD : Ladder Diagram IL : Instruction List FBD : Function Block Diagram CFC : Continuous Function Chart ST : Structured Text SFC : Sequential Function Chart	
	G-Code	○（DIN 66025）	-
EtherCAT	Communication cycle	2ms, 4ms, 8ms, 16ms	
	Maximum number of connected slaves	8 slaves	
	CoE	○	
	FoE	○	
	Hot connect	○	
Motion control	Maximum number of control axes	2ms/1~4 axes，8ms/5~8 axes	
	Control mode	Position control Velocity control Torque control	Position control（PTP）
	Synchronous control	Electronic cam･gear, CNC	-
	Control unit	Any（pulse, mm, inch, degree）	
Robot control	Gantry robot	○	-
	Scara robot	○	-
	Delta robot	○	-
	Maximum control number	1	-
	Maximum number of constituent axes	4 axes	-
Remote control		○	○
Mail notification		○	○
Corresponding communication	EtherCAT master	○	○
	EtherNet/IP	○	-
	OPC UA server	○	○
	Web Visualization	○	○
	Samba	○	○
	Modbus-TCP	○	○
	Modbus-RTU	○	○
	MC Protocol	○	○
	FINS	○	○

10.2. Factory default setting

The factory settings are shown below. The factory reset function changes the settings as follows.

Item	Value
Host name	SMC200
Ethernet port	DHCP : Inactive IP address : 192.168.21.101 Subnet mask : 255.255.255.0
USB port	IP address : 169.254.21.101 Subnet mask : 255.255.0.0
Wireless LAN	DHCP : Inactive IP address : 192.168.100.101 Subnet mask : 255.255.255.0 Mode : AP SSID : SMC200-AP Security : Personal Password : 123456789 Country code : US
Date and time	2001-01-01 00:00:00
Time zone	UTC
Web app password	User : Administrator Password : sanyodenki
Samba password	User : sanmotion Password : sanmotion
FTP password	User : ftp Password : ftp
Auto start	Active : plc, samba Inactive : ntp, ftp

Release
Revision A Aug. 2023

ECO PRODUCTS

■ECO PRODUCTS
Sanyo Denki's ECO PRODUCTS are designed with the concept of lessening impact on the environment in the process from product development to waste. The product units and packaging materials are designed for reduced environmental impact.
We have established our own assessment criteria on the environmental impacts applicable to all processes, ranging from design to manufacture.

■Precautions For Adoption

Failure to follow the precautions on the right may cause moderate injury and property damage, or in some circumstances, could lead to a serious accident.
Always follow all listed precautions.

⚠ Cautions

- Read the accompanying Instruction Manual carefully prior to using the product.
- If applying to medical devices and other equipment affecting people's lives please contact us beforehand and take appropriate safety measures.
- If applying to equipment that can have significant effects on society and the general public, please contact us beforehand.
- Do not use this product in an environment where vibration is present, such as in a moving vehicle or shipping vessel.
- Do not perform any retrofitting, re-engineering, or modification to this equipment.
- The Products presented in this Instruction Manual are meant to be used for general industrial applications. If using for special applications related to aviation and space, nuclear power, electric power, submarine repeaters, etc., please contact us beforehand.

* For any question or inquiry regarding the above, contact our Sales Department.

https://www.sanyodenki.com

SANYO DENKI CO., LTD. TEL: +81 3 5927 1020
3-33-1, Minami-Otsuka, Toshima-ku, Tokyo, 170-8451, Japan
Singapore Branch TEL: + 65 6223 1071
988 Toa Payoh North, #04-08, Singapore 319002
Jakarta Representative Office TEL: + 62 21 252 3202
Summitmas II 4th Floor, Jl. Jend. Sudirman Kav.61-62, Jakarta 12190, Indonesia

SANYO DENKI EUROPE SA. TEL: +33 1 48 63 26 61
P.A. PARIS NORD II, 48 Allée des Erables-VILLEPINTE, BP.57286, F-95958 ROISSY CDG CEDEX, France
Poland Branch TEL: +48 12 427 30 73
ul. Wodociagowa 56 30-205 Kraków, Polska

SANYO DENKI AMERICA, INC. TEL: +1 310 783 5400
468 Amapola Avenue Torrance, CA 90501, U.S.A.

SANYO DENKI SHANGHAI CO., LTD. TEL: +86 21 6235 1107
Room 2106-2110, Bldg A, Far East International Plaza, No.319, Xianxia Road, Shanghai, 200051, China

SANYO DENKI (H.K.) CO., LIMITED TEL: +852 2312 6250
Room 2305, 23/F, South Tower, Concordia Plaza, 1 Science Museum Road, TST East, Kowloon, Hong Kong

SANYO DENKI TAIWAN CO., LTD. TEL: +886 2 2511 3938
N-711, 7F, Chia Hsin 2nd Bldg., No.96, Sec.2, Zhongshan N. Rd., Taipei 10449, Taiwan

SANYO DENKI GERMANY GmbH TEL: +49 6196 76113 0
Frankfurter Strasse 80-82, 65760 Eschborn, Germany

SANYO DENKI KOREA CO., LTD. TEL: +82 2 773 5623
15F, KDB Building, 372, Hangang-daero, Yongsan-gu, Seoul, 04323, Korea
Busan Branch TEL: +82 51 796 5151
8F, CJ Korea Express Building, 119, Daegyo-ro, Jung-gu, Busan, 48943, Korea

SANYO DENKI (Shenzhen) CO., LTD. TEL: +86 755 3337 3868
04B-07, 11/F, AVIC Center, No.1018 Huafu Road, Futian District, Shenzhen, 518031, Guangdong, China
Chengdu Branch TEL: +86 28 8661 6901
Room2105B, Block A, Times Plaza, 2 Zongfu Road, Jinjiang District, Chengdu, 610016 China

SANYO DENKI (THAILAND) CO., LTD. TEL: +66 2261 8670
388 Exchange Tower, 25th Floor, Unit 2501-1, Sukhumvit Road, Klongtoey, Klongtoey, Bangkok 10110 Thailand

SANYO DENKI INDIA PRIVATE LIMITED TEL: +91 44 420 384 72
#14 (Old No.6/3), Avenue Road, Nungambakkam, Chennai - 600034, Tamil Nadu, India

SANYO DENKI (Tianjin) CO., LTD. TEL: +86 22 2320 1186
Room AB 16th Floor TEDA Building, No. 256 Jie Fang Nan Road, Hexi District, Tianjin 300042 China
Beijing Branch TEL: +86 10 5861 1508
Room1807, Gaohe Lanfeng Buliding, No.98 East Third Ring South Road, Chaoyang District, Beijing 100122 China

Translated version of the original instructions