深圳市威科达科技有限公司

SHENZHEN VECTOR SCIENCE AND TECHNOLOGYCO.,LTD.

VE 型运动控制器

VE Motion Controllers

VE 控制器编程应用手册

VE Controller Programming Application Manual

Contact phone/fax：0769-22235716

Address: Building 12, CIMC Zhigu, No. 1, Nanshan Road, Songshan Lake High-tech Industrial Development Zone, Dongguan City, Guangdong Province

Foreword

Thank you for purchasing a VE motion controller! The VE Motion Controller is a high-performance EtherCAT bus-based controller developed by our company. This manual describes the VE Motion Controller software and the quick application of the motion control functions. For more detailed function descriptions users can go to the Viktor website athttp://www.szvector.com/.

Thank you for purchasing VE series motion controller! VE series controller is a high-performance EtherCAT bus type motion controller developed by our company. This programming manual describes the VE motion controller programming software and the use of motion control functions. The user should read this manual carefully before using the controller and software, and operate correctly under the premise of full attention to safety.

User-oriented

This manual is provided to the following readers: persons with electrical professional knowledge (qualified electrical engineers or persons with equivalent knowledge).

In addition, the readers of the programming language are those who understand the content of the international standard IEC 61131-3.

Target product

VE series: VEC-VE-MU

Change time	version number	Change Description
June 2020	Version 001	First edition released

Version update record

1 PLCopen Introduction to the Code

PLCopen is an independent global organisation that delivers industrial automation efficiency according to user needs. It was founded in 1992 and is based in the Netherlands with support offices in the USA, Japan, China and Korea. PLCopen follows the requirements of market demand and its main focus is to improve automation efficiency by defining common standards. PLCopen and its members focus on technical specifications around the IEC 61131-3 standard to reduce the cost of industrial engineering.

The syntax of the IEC 61131-3 specification presents a set of mechanisms for implementing programmable controllers across different target platforms. Through modular planning and design, the specification divides control actions into two parts: logical operations and hardware actions. The logical part unifies the syntax defined in IEC 61131-3 in a common description format and implements it, while the hardware actions are designed with a proprietary firmware library for each hardware, allowing the control logic to use hardware resources on each target platform. This design allows different control chips to execute control actions designed in the IEC 61131-3 syntax, and designers only need to learn the IEC 61131-3 syntax to use the supported control chips for programmable controller design. The IEC 61131-3 standard defines six standard programming languages.

指令表（Instruction List，IL）

梯形图（Ladder Diagram，LD）

功能块图（Function Block Diagram，FBD）

结构化文字（Structured Text Language，STL）

顺序功能流程图（Sequential Function Chart，SFC）

连续功能图（Continuous Function Chart，CFC）

The VE motion controller uses the CODESYS programming platform, which fully supports the PLCopen specification and allows the user to refer to many standard function libraries; the high-level language programming method makes it easy for PLC manufacturers and users to develop their own proprietary function blocks and instruction libraries, and to borrow similar control programs to form industry-specific "process packages", which can significantly improve the user's programming efficiency.

2 VEC-VE AND CODESYS

2.1 VEC-VE controller

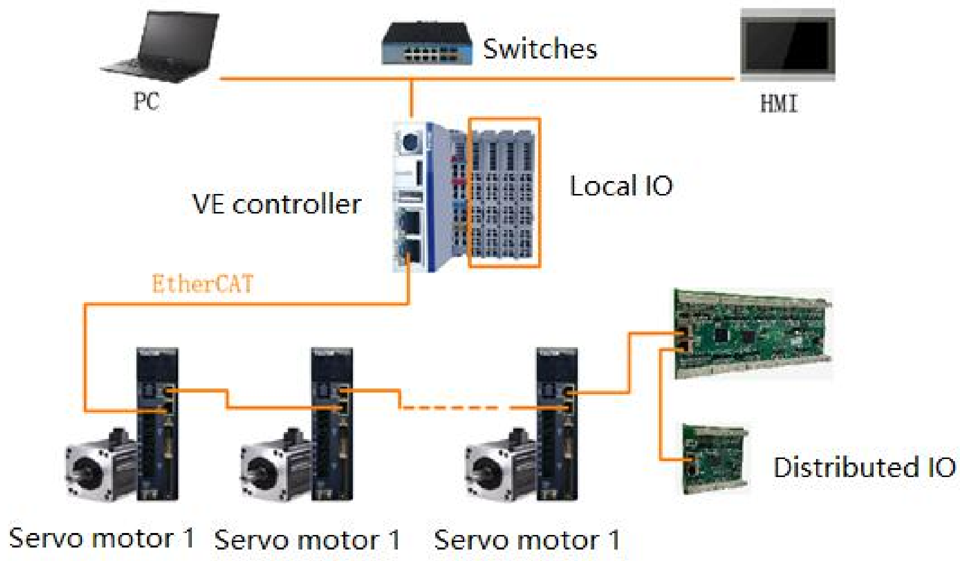
2.1.1 Product overview

The VEC-VE series controller (hereinafter referred to as VE controller) is a programmable logic controller designed with a modular structure to provide users with intelligent automation solutions.The VE controller adopts the IEC61131-3 programming language system and supports the PLCopen standard 6 programming languages. The system uses a rack layout and each rack supports local expansion modules and remote expansion of the rack via the EtherCAT bus. local expansion modules of the VE controller allow IO expansion via internal bus protocols and support a wide range of functional modules such as digital input/output modules, analogue input/output modules and temperature modules. High performance motion control functions can be realised via EtherCAT bus; single axis acceleration/deceleration control functions, electronic gear functions, electronic cam functions, CNC functions and Robotic functions etc.; communication functions such as RS485, Ethernet and USB are also supported to meet the diverse application requirements of users. The VE controller has the following functional advantages:

- Multiple motion control functions
- Support for a larger number of I/O points;
- Larger program capacity and data storage areas;
- Faster instruction execution;
- support for more EtherCAT buses, Modbus communication;
- easier-to-use software for different user application requirements;
- support for online editing mode

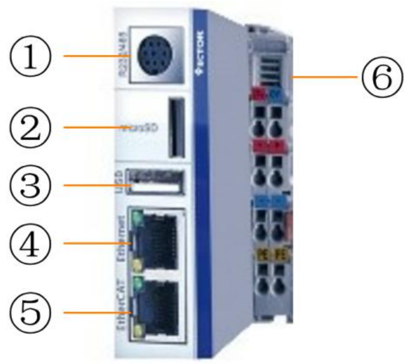
2.1.2 Product configuration and module description

A schematic diagram of the VE controller architecture integration is shown below:



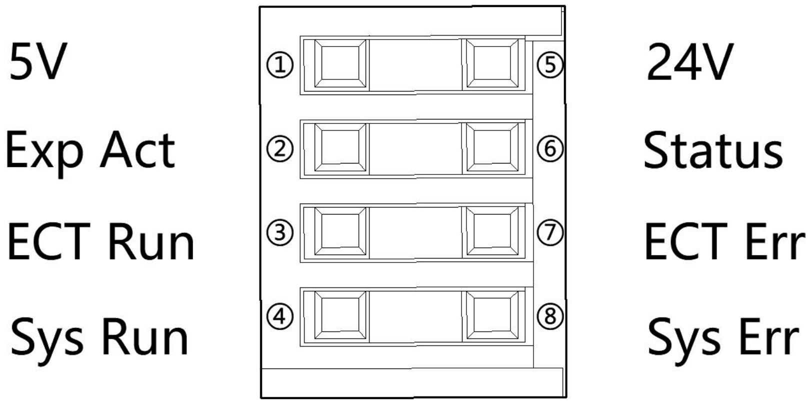
VE controller hardware port description:

(1) Host Interface



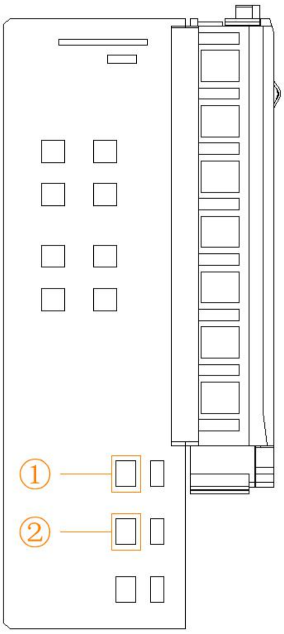
①	RS232/485	②	MicroSD card
③	USB	④	Ethernet
⑤	EtherCAT	⑥	Power supply modules

(2) Power Module Indicator



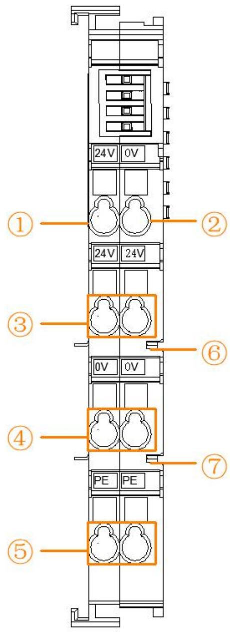
Number	Always bright	Always extinguished	Blinking
① 5V power supply	The main unit is powered properly	The host does not have power	
② Extend the mesh light		Local extensions are not accessed	Extended access locally
② EtherCAT RUN	EtherCAT RUN	EtherCAT STOP	
③ Run&Stop	Run	Stop	
⑤ 24V power supply	IO power supply access 24V	IO power supply not connected to 24V	
⑥CODESYS	In progress/dead	In progress/dead	CODESYS software in operation
⑦EtherCAT Error	EtherCAT Error	EtherCAT Not Error	
⑧Error	System failure	No system failure	

(3) Reset and IP setting buttons



Number	Name	Description
①	Reset button	After powering up and running, press and hold the button for 3 seconds, then power up again, which will clear the user program as well as restore the controller's default IP (192.168.1.123)
②	IP setting button	After powering up and running, press and hold the button for 3 seconds, then power up again, the last digit of the controller IP address will be minus one, for example, the default IP minus one will be: 192.168.1.122

(4) Power Module Wiring



Number	Interface name	Interface role

①	Mainframe 24V input	Mainframe power supply 24V
②	Mainframe 0V input	Mainframe power supply 0V
③	IO power supply 24V	Local IO supply 24V
④	IO power supply 0V	Local IO supply 0V
⑤	PE	Ground line
⑥	Extended power supply 24V	Local IO supply 24V, connected to ③
⑦	Extended power supply 0V	Local IO supply 0V, connected to ④

2.2 CODESYS Software overview

2.2.1 CODESYS Introduction to the software

CODESYS software is standard software for the development and application of VE programmable motion controller products. CODESYS software platform provides VE controllers with a complete configuration, programming, debugging, monitoring environment, flexible and free to handle the powerful IEC language.

The codesYS software enables the management of engineering and equipment, providing the following configurations for VE controller products:

- CPU configuration;
- I/O module configuration;
- EtherCAT bus;
- ModbusRTU/ModbusTCP bus;
- Standardized programming (IEC61131-3 compliant).
- Supports all six programming languages: Structured Text (ST), Function Block Chart (FBD), Instruction List (IL), Keystone (LD), Sequential Function Map (SFC), and IEC61131-3 Extended Programming Language Continuous Function Map (CFC)
- Flexible and comprehensive feature block libraries and support user-defined libraries
- Offline simulation function, do not need to connect PLC hardware, complete the program simulation debugging
- Intelligent debugging error check function
- Compile errors, quickly locate programming errors, and diagnose logs
- Sample tracking
- The time series chart of the process variable is established

2.2.2 Software access and installation requirements

(1) Software acquisition

VE programmable motion controller user programming software CODESYS for free software, installation packages as well

Information on VE controller-related products is available to users through:

- On the official website of Wykoda (www.wikoda.com) szvector.com"page of the "Services and Support" and "Material Downloads" page for download
- Available for download onCODESYS's official website codesys.cn .4000

(2) Software installation environment requirements

- Windows XP/Windows 7/Windows 8 or Windows 10 operating system
- CPU master frequency: 2GHz or more (recommended)
- Memory: 2GB or more

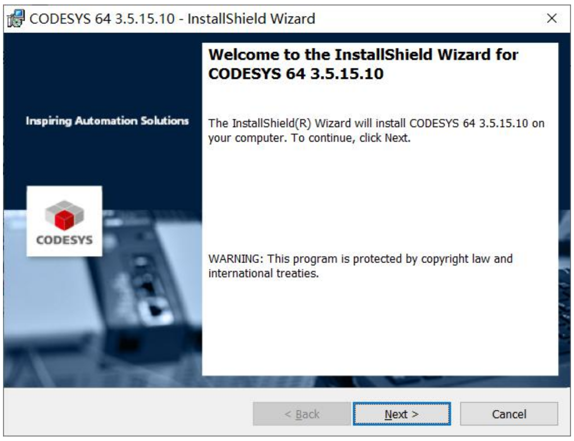
- Space: More than 5GB of available hard drive memory
- Other: There is an idle LAN port in the local network (with LAN network cable connection controller)

2.2.3 The software installation procedure

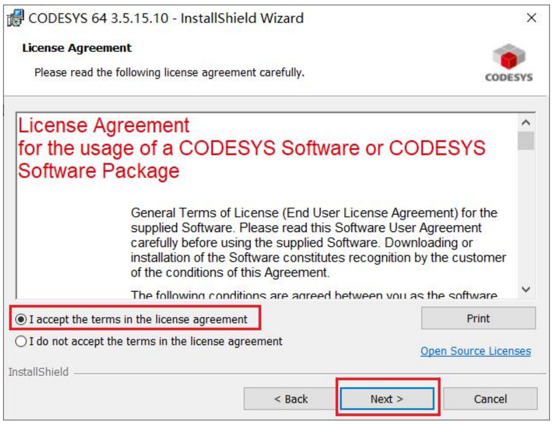
CODESYS Development System is an IEC 61131-3 programming tool for industrial control and automation technology, with 32-bit and 64-bit versions where users choose according to the number of system bits on their computer and then start installing (64 bits are shown here).

1) Double-click to open the installation software icon and start the installation, as shown in the following image

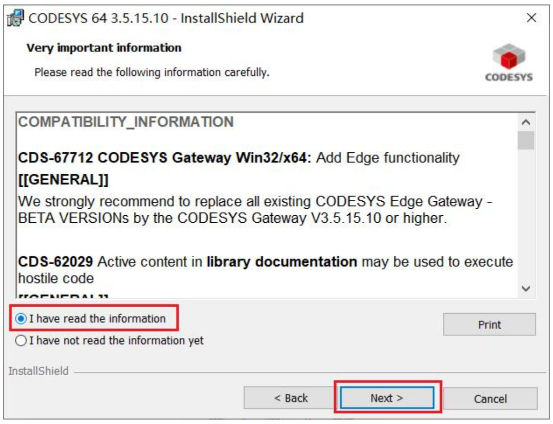
2) When the prompt screen appears, click "Next"as shown in the following image



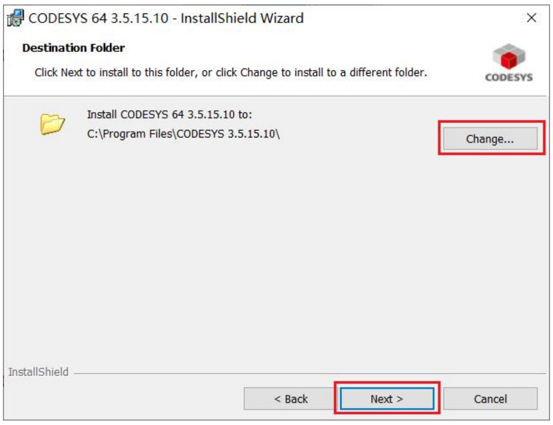
3) Select "I accept... ", click next; .



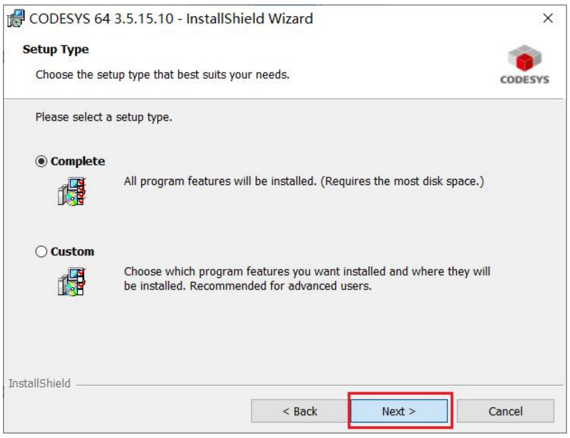
4) Select "I have..." " and then click "Next"



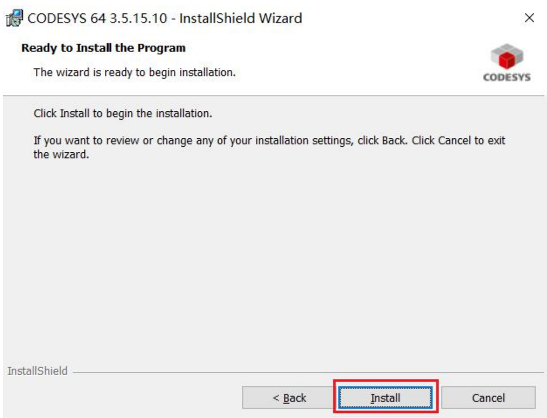
5) Select the installation location, select the default bit here, click "Next"



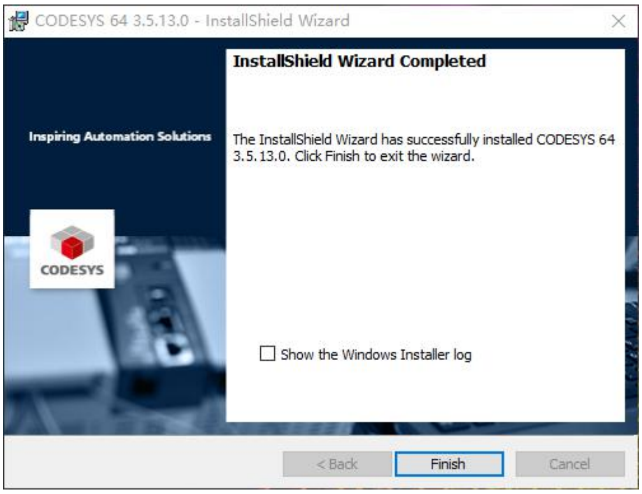
6) Installation type Select "Complete", install all, click "Next"



7) Click "Install" and the software will start installing



8） After waiting for the installation to be completed, click "Finish" to complete the installation, click on the desktop icon "CODESYS V3.5" to enter the CODESYS programming environment

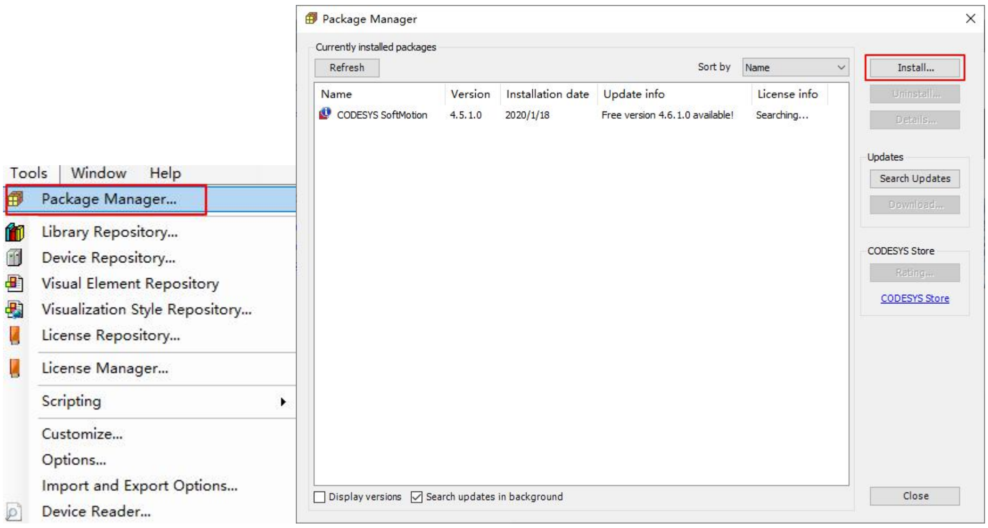


9） The software interface operating language defaults to Chinese Simplified, and if → you need to switch to a different → language,click: Tool Options Language Settings(Select: Tools Options→International Settings),click on the language→drop-downmenu, and select the language you want

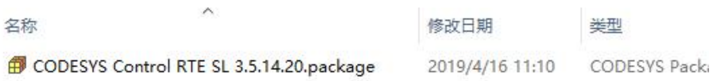
2.2.4 Install Package

Before the software can connect to the controller, You need to install Package, and you don't need to install it again after the installation is complete.

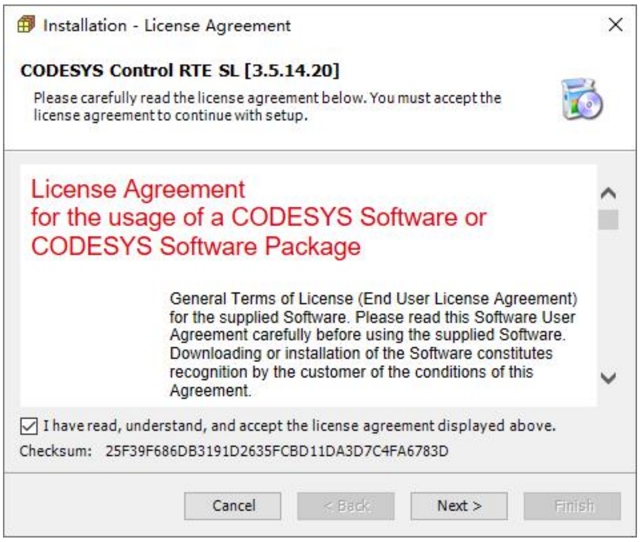
1） Open：CODESYS →Tools→Package Manager →Install，Open file browsing



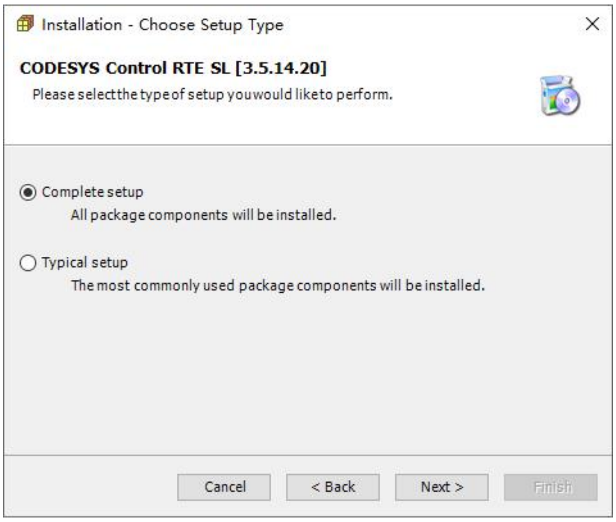
2）Find the downloaded CODESYS Control RTE SL.package andopen the installation



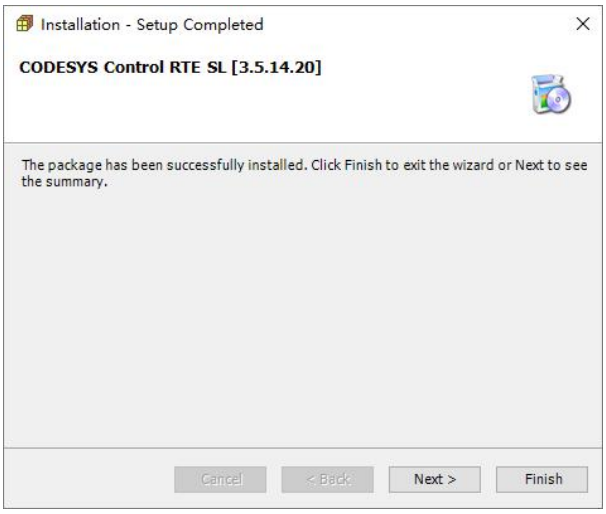
3）Check "I have read... " ", click "Next" as shown in the following image



4）Select Complete setup for the full installation, click Next, and wait for the installation to complete, as shown in the following image



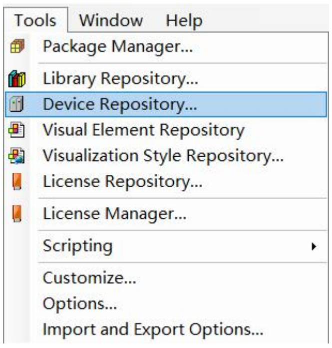
5）Once the installation is complete, click "Finish" as shown in the following image

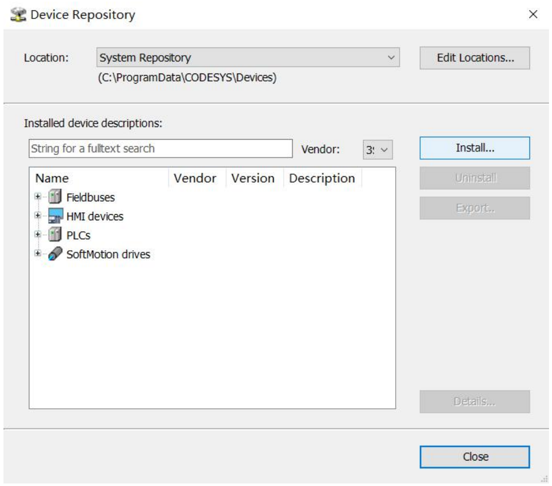


2.2.5 Install the device description file

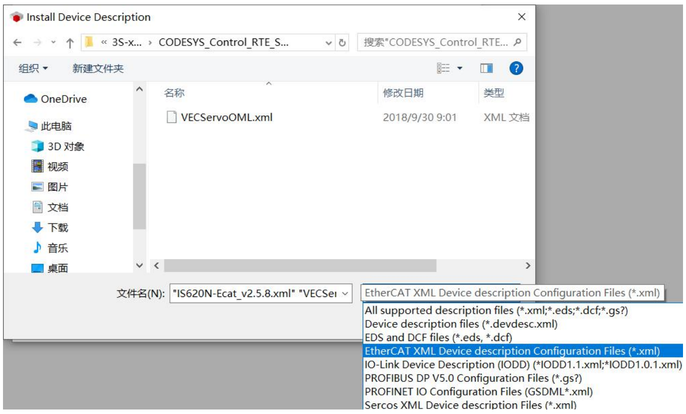
Before using a VE controller or VC servo drive, you need to install the motion controller and the servo drive XML description file, file acquisition please log on to the Wykoda Technology website： www.szvector.comdownload， Once the installation is complete, the controller or servo does not need to be installed repeatedly.

Here is a demonstration of how to install the XML description file： Open CODESYS →Tools→Device repository→Install， Find theE therCAT bus servo description file and click Open





Note the settings add file type: EtherCAT XML Device Description Description Configuration Files (*.xml), the device description file is requested from the servo vendor, and the Wykoda EtherCAT bus servo description file is"VECServoOML.xml".



When the installation is complete, the following image shows

2.2.6 Uninstall CODESYS

Uninstall CODESYS using the standard Windows system uninstall software method, as follows:

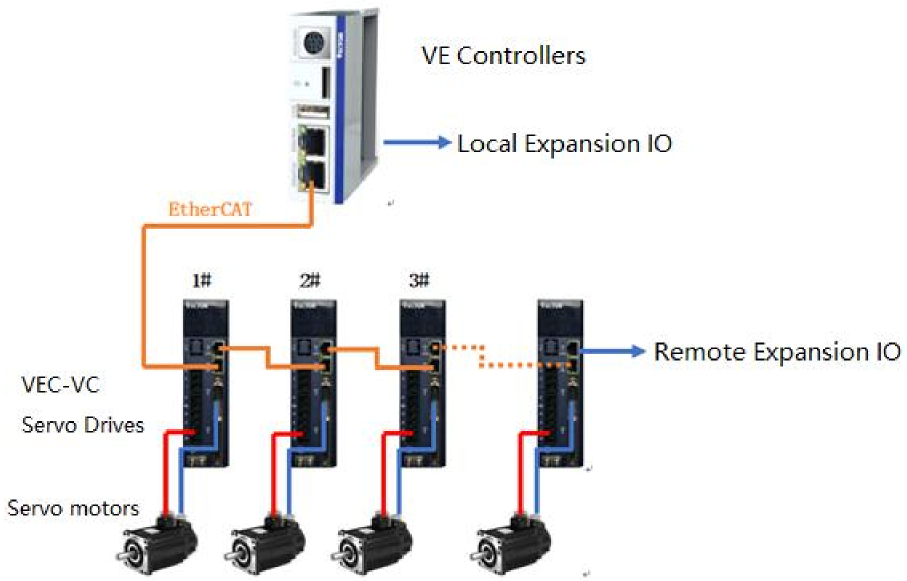
- Exit the CODESYS software to confirm that Gateway is turned off and, if the operating system taskbar has a CoDeSys icon, right-click the mouse on that icon and select Exit to turn off Gateway
- Choose“Start→Settings→Control Panel”
- Click "Add/Delete Program"
- Select the software you need to uninstall and find CODESYS
- Right-click to "uninstall" and confirm

3 The motion control system is composed of procedures

3.1 The motion control system of the VE controller consists of

The VE controller is a universal programmable controller with SoftMotion motion control (CAM/CNC/ROBOT) that controls multiple motion axes via the EtherCAT bus, as shown below for a typical control bus network. The servo uses a VEC-VC bus-controlled servo, and the IO expansion rack is also connected to the CPU module of the VE controller via the EtherCAT bus.

As shown in the typical motion control network, where the VE controller is the control master, servo axis, remote IO, etc. for the access station. The EtherCAT bus is a real-time bus, and its first station clock will serve as a reference synchronization clock for the entire network, so the servo should be installed at the front end of the EtherCAT bus network, i.e. the 1-way reference of the network must be servo, while the EtherCAT remote module does not have a clock unit inside, and is typically installed in the middle or back end of the network in a network that requires motion control.



Motion Control is characterized by the controller through software calculations, digital commands through the EtherCAT real-time bus to control servo operation, the Use of EtherCAT bus high-speed (100Mbps), high frequency (up to 1ms communication once) to interact, compared to traditional pulse control methods, motion control can be more timely and accurate. Some of the resulting programming methods are also different from the previous keystone logic controls, requiring the use of "function blocks" with more underlying functionality.

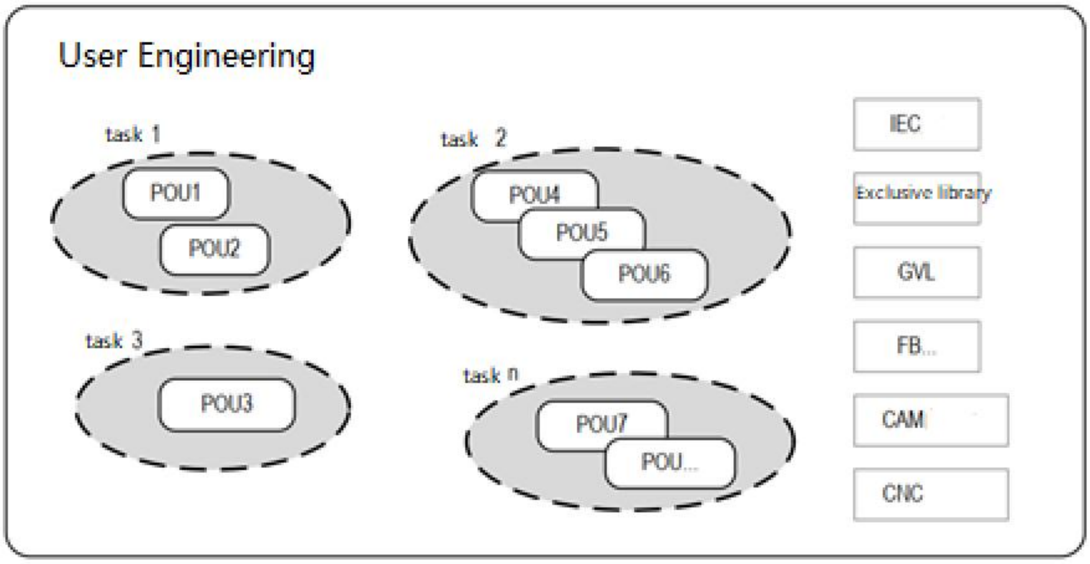
3.2 The VE controller motion control program consists of

VE controller is a controller developed based on multi-tasking operating system, the system runs multiple functional modules in a multi-tasking manner, for user programs, can also be divided into multiple tasks, according to the user-set task priority, respectively.

When writing a user program for a VE controller, the user can divide into several program organization units according to the different types of tasks and urgency processed in the application system, and can specify different execution trigger conditions for each task, or the corresponding execution interval (also known as execution cycle), so that the control response of the application system can be optimal.

3.2.1 The user program of the VE controller is composed of

Ve controllers can use a multitaste execution pattern, in which several tasks can be performed "at the same time", each of which can have several user program organizational units (CALLEDUs),typicalof which are shown below:



User engineering consists of several OUUs, which are divided into task groups according to POU execution characteristics, configuring their execution characteristics, and POU that is not included in the task configuration will not be executed.

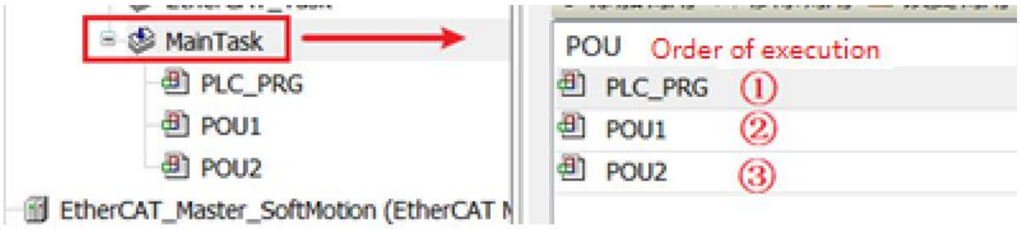
In addition, there are some objects supporting the user program, such as library functions, global variable gVL, function block FB, cam definition CAM curve, multi-axis interpolation track definition CNC curve, etc., as part of the user program.

3.2.2 The type of task in the VE controller

Task configuration is to divide the user program into several task groups according to execution requirements, each of which can set different execution trigger conditions, execution intervals, priorities, and so on. VE controllers Common tasks are: EtherCAT tasks, main loop tasks, etc., where the motion control-related user program body is scheduled to be performed under the EtherCAT task.

The EtherCAT task is one of the most important tasks in the VE controller, and the real-time processing of motion control functions is done in this task, it is a clock-interrupted task with a short interval and the highest priority, and once the time conditions are met, it can unconditionally interrupt other tasks and start performing the EtherCAT task until all the POU is configured for that task.

In each task, you can specify that multiple user program units (i.e. PUS) are executed in sequence, in order of task configuration, as shown in the following illustration:



In the figure, the three POUes are executed in the order of PLC_PRG, POU1,POU2, and when there are global variable update operationsand judgments, care needs to be taken to arrange the appropriate order.

There is also a EtherCAT_Task task in the figure, which isadded automatically when the EtherCAT_ Master_SoftMotion device is added, which, depending on the priority, can be understood as default processing of bus communication tasks performed by the system when entering the EtherCAT task, including the primary station sending and receiving from all the PDO of the host station, the update processing of the data structure of each servo axis, and so on.

3.2.3 The benefits of a user program consisting of multiple PUS

Handlers for different execution cycles should be written in different PUS. For example, a POU executed by EtherCAT cycle, an external interrupter POU, and a program POU processed at 20ms time must be written into separate PUS.

In order to improve the readability of the program, according to different control process segments, different operating objects, different physical structure parts, etc., each POU is handled with different POU, each POU is named easy to understand the name;

When multiple people work together on programming, each programmer writes and debugs the POU of the process segment under his or her responsibility, which eventually becomes a user program project; CODESYS programming software supports 6 programming languages, depending on the type of processing logic required, a language may be more convenient, while in general, each POU can only be written in one programming language, and if multiple programming languages need to be used simultaneously, it is also a good countermeasure to write multiple PUs.

3.2.4 How the user program can do both logical control and motion control

Application system synchronization control, track control, often have high real-time requirements, and the timeliness requirements of logical control is relatively low, in the VE controller user program, the motion control (MC) POU can be executed in the EtherCAT task cycle, and logical control POU can be executed in the ordinary task cycle. If a specific program variable is declared as a global variable, the coordinated action with logical control can be implemented in motion control.

For single-axis MC control, which is mainly controlled by servo driver and motor, servo enablement, origin regression, positioning control, speed control, torque control, stop and reset, and for applications of multi-axis synchronous MC control, such as cam control, track interpolation control, etc., the controller provides corresponding MC function blocks to complete these operations. Therefore, function blocks are commonly used control commands in motion control programming, just as prefabribored parts are used in buildings instead of gravel cement to improve construction efficiency.

The user program can according to the control logic of the application system, control the function block's operation trigger, terminate execution, etc., at the same time can judge the function block's execution state, whether there is an error, etc., in the PLCopen specification, also introduces the axis state data structure, the controller system has established a corresponding data structure for each servo axis that the user has configured, and automatically updates its state in time in each EtherCAT cycle, and the user program can access the variables of the data structure. The operating state of the servo axis can be monitored and the state variables can be used as the basis for logical control, which makes logical control and motion control easily implemented in a user program.

3.3 Typical steps to write a user program

A complete user program, writing generally takes 5 steps, users need to pay attention to.

1) According to the PLC module hardware connection mode of VE controller application system, the hardware system configuration is carried out.

2) According to the control process of the application system, the user program is written. Programming user program data storage width, use range, from the defined variables, can be independent of the hardware configuration;

3) associate the input port variable (I) and output port variable (Q) corresponding to each hardware port in the system composition with the variable in the user program;

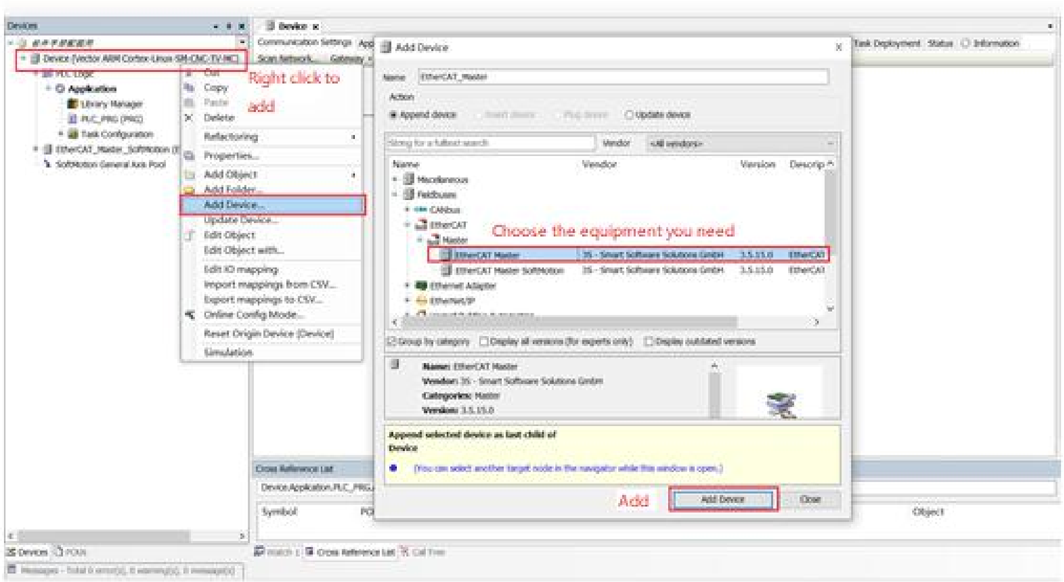
4) Configure the synchronization cycle of network communication (e.g. EtherCAT bus), according to the real-time requirements of each task, configure the execution cycle of the user program unit;

5) In the CODESYS programming environment, log on to the VE controller, download the user program, simulation and debugging modifications, until accurate operation.

3.3.1 The configuration of the user system

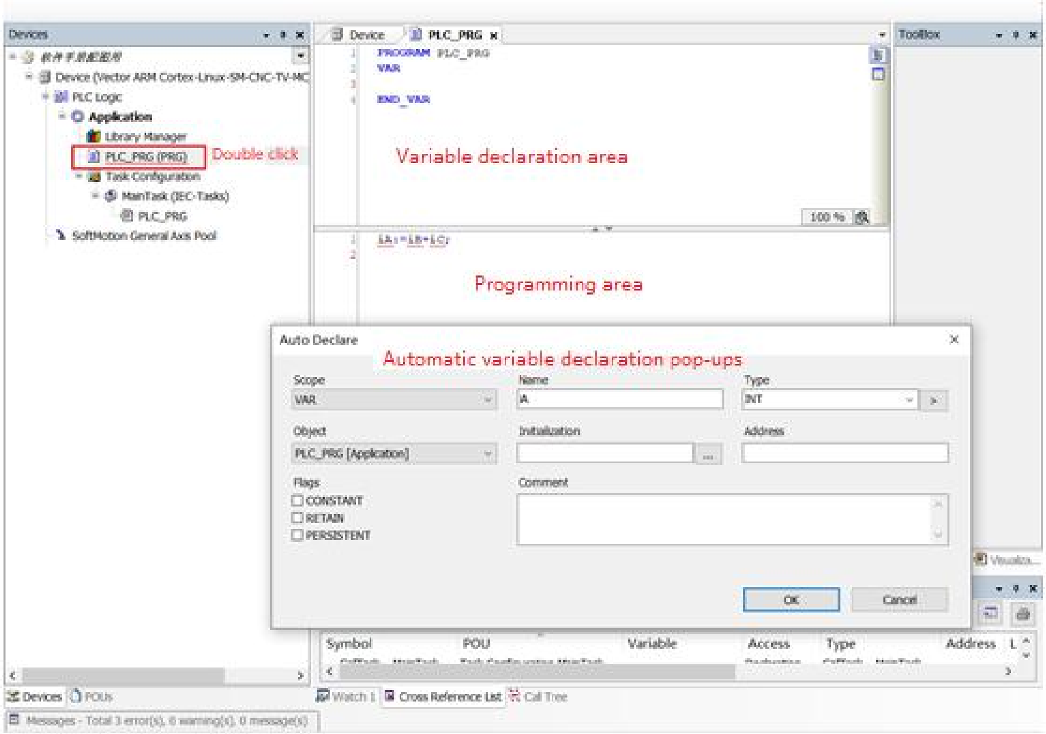
In the main screen of CODESYS software, through the right-click left tree window"Device" item,that is, to enter thedevice add interface, according to the actual application system used module model, installation order, in turn from the interface of the device library, double-click selected, or drag placed under "Device", to delete a module, after selecting the module, press Del key can be deleted.

Placement screen:



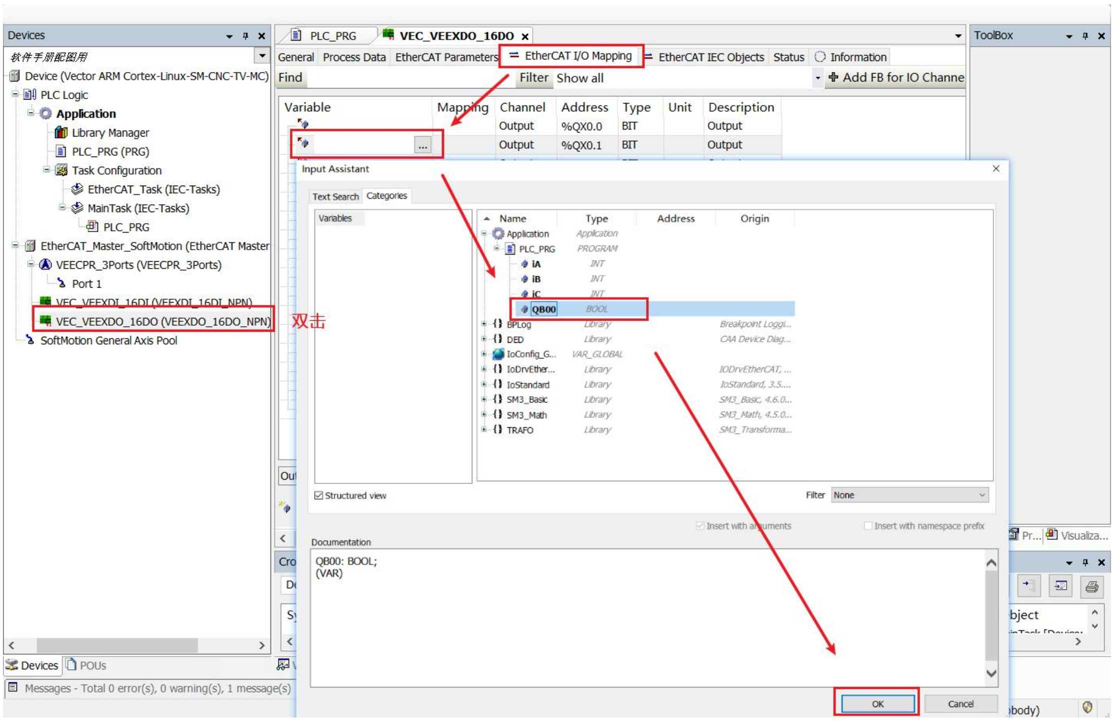
3.3.2 The writing of the user program

Double-click on the"PLC_PRG (PRG)" item in the left tree window to open the user programming interface, which is ST (selected when creating a new project), as shown in the following image. Similar to C language programming, each variable needs to be declared before it can be used, if you write the program statement directly, when entering, the programming environment will automatically pop up the declaration box, let the user fill in, once click "OK", the variable declaration window will automatically increase the variable declaration statement, simplifying programming.



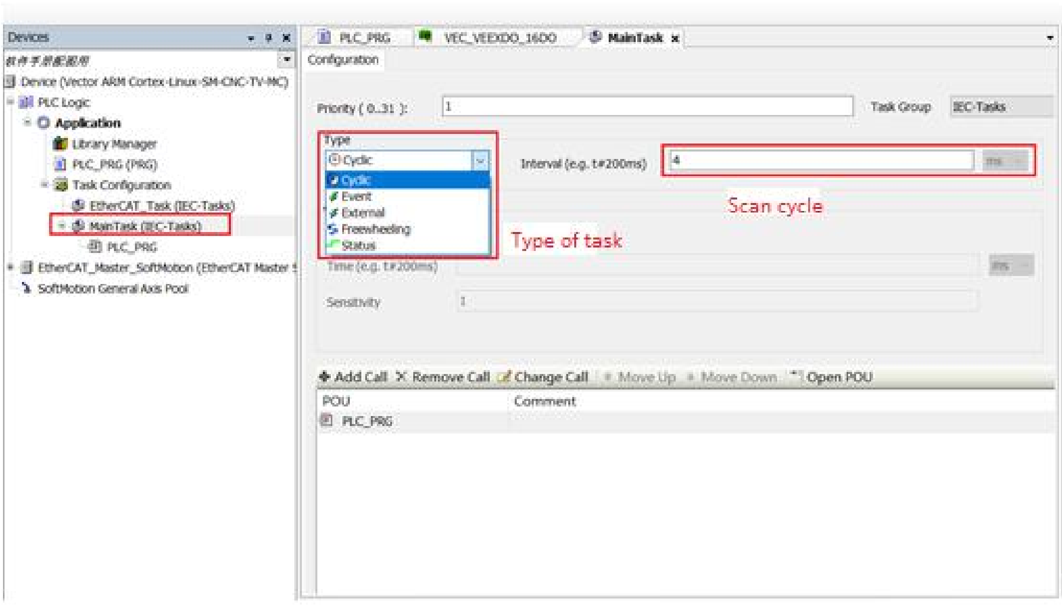
3.3.3 The user program variable is associated with the port

On the local bus configuration page, the required hardware ports are associated with variables in the user program, and the variable valueof "QB00" is shown below, output on the output port of the first DO module, as follows:



3.3.4 How the user program is executed and how it is configured to run

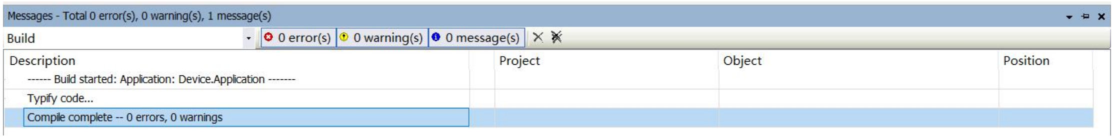
When the program is complete, you need to add the program to the task and configure the task, which defaults to 4 ms once, and if you want to change to other execution methods, such as repeatedexecution, scheduled execution, execution cycles, and so on, you can set it separately, as shown in the following image:



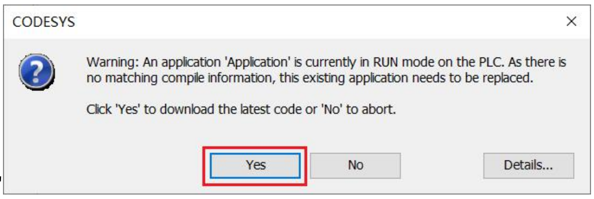
3.3.5 Program compilation and login download

Once the program is written, click Compile .Generate the user application, check if there are errors and if so, click on the error message line to locate the error reporting point of the user application for easy modification until all errors are eliminated.

The relevant compilation information will be displayed in the following compilation information
box:

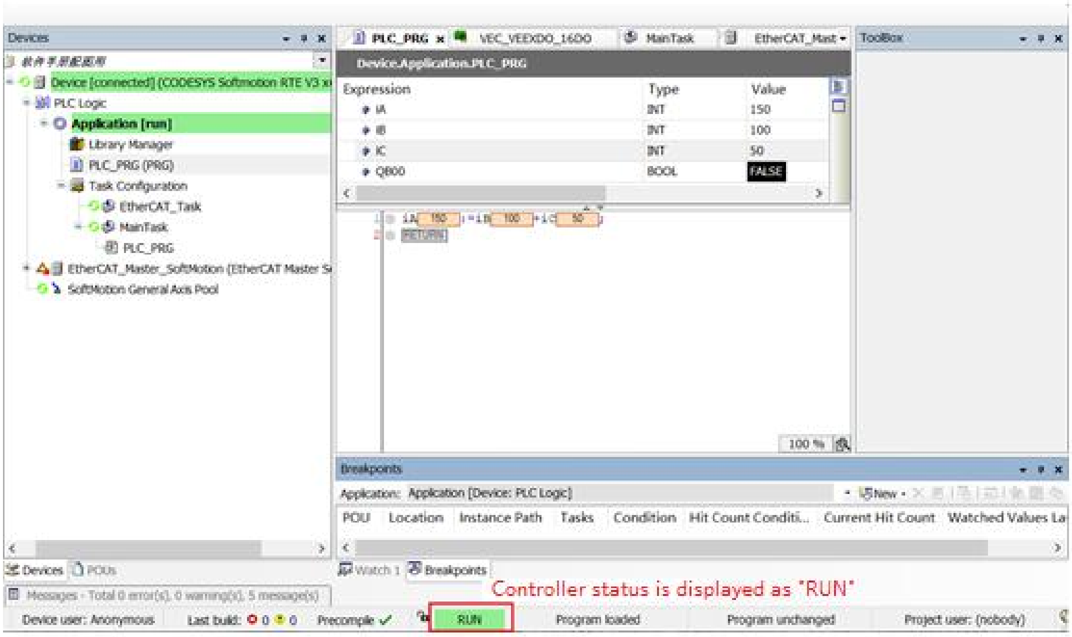


Once compiled, click on "Online "-"Log in to ", The following dialog box pops up, select "Yes" "No" "...... Do you wish to create and continue the download?" , select "Yes"



Once the download is complete, click on "Run ", Run and debug the program.

The following image shows the monitoring screen of a running user program.



4 A simple user program

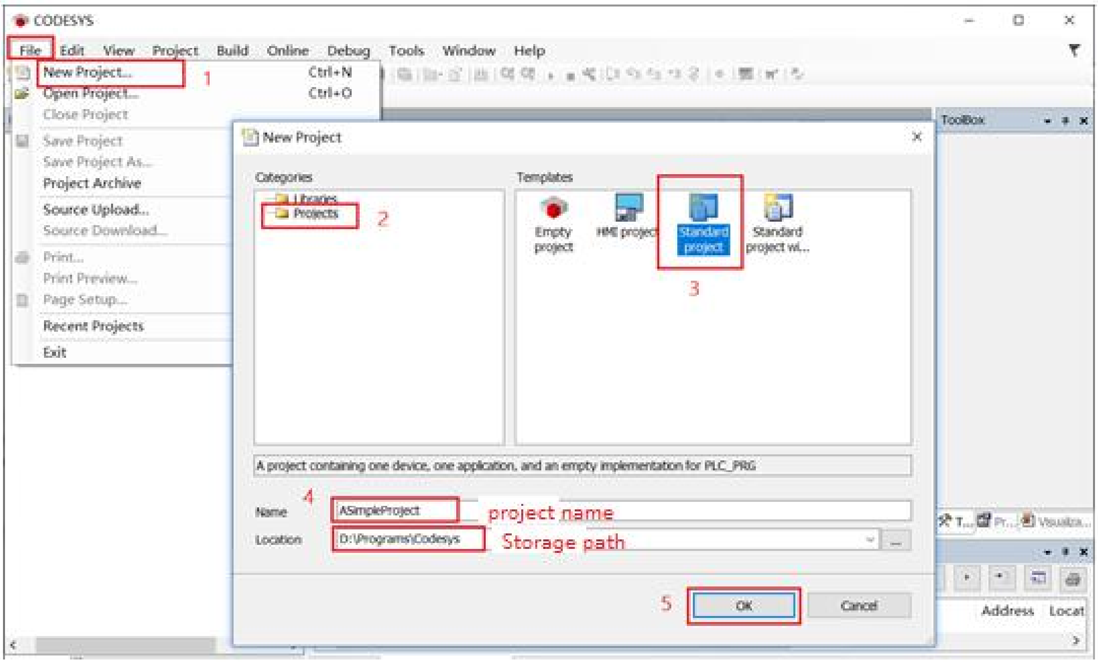
In order to make users more familiar with the software and hands-on programming, this section will demonstrate how to useCODE SYS to build a simple EtherCAT bus project, using a VE motion controller, through the EtherCAT bus, control VECServo (Wykoda bus servo), to complete the enable, position mode operation and stop and other actions.

Note: Engineering is not a standard template and the content is for informational purposes only.

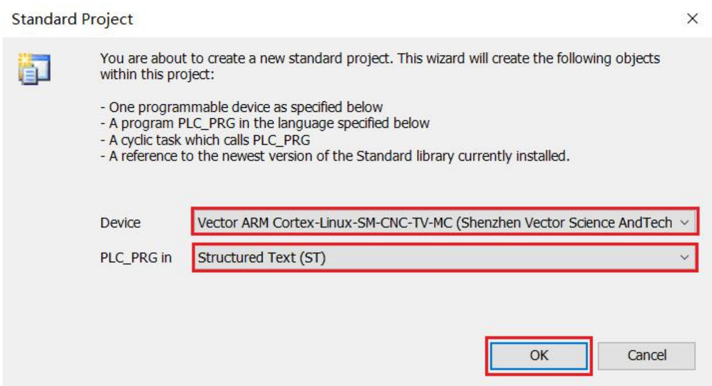
4.1 Create a project and download debugging

4.1.1 Create a new standard project

1) Once the software is open, click on "File"→"New Project", A dialog box pops up and select "Projects", Select "Standard Project" as the project type, choose your own project path, name the project "ASimpleProject", and click "OK".



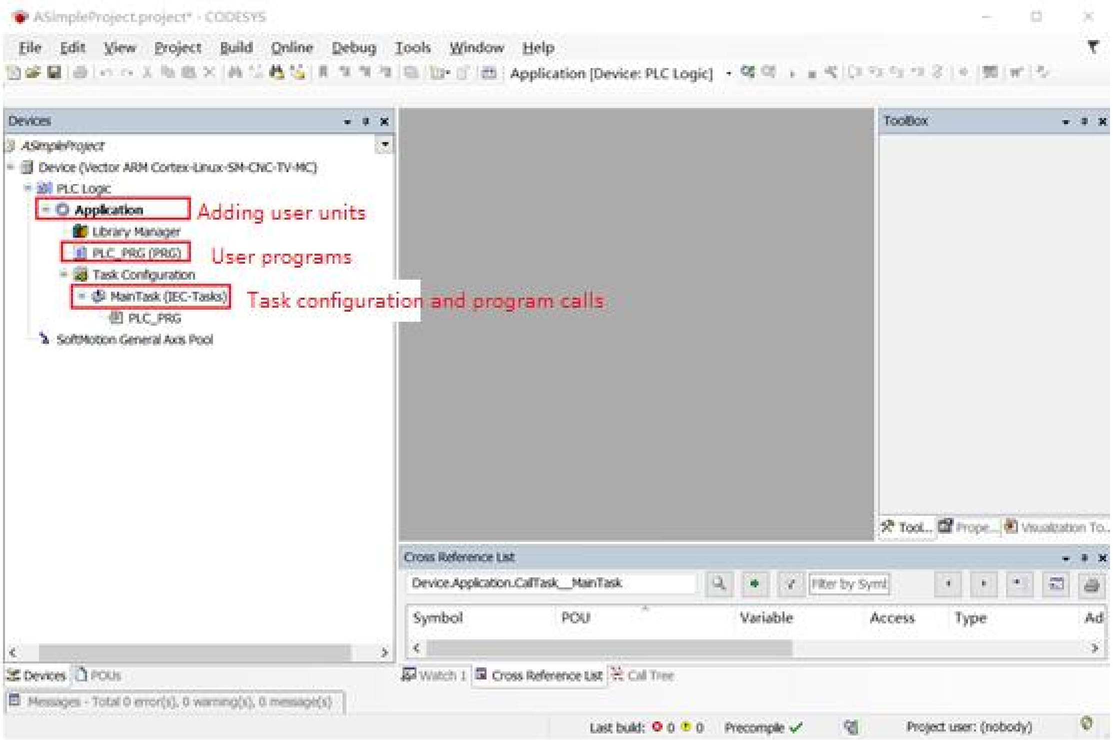
2) Entering the standard project screen, the user can select the device type and programming language for that project. The image below:



Device: Select the model of the main module, select Vector ARM Cortex-Linux-CNC-TV-MC (embedded platform controller, need to install the XML description file first: Vector ARM Cortex-Linux-SM-CNC-TV-MC.xml, please refer to the installation method Install the device description file) , or CODESYS Softmotion RTE V3 x64(Soft platform controller for real-time control).

Programming language: ST, other programming languages are also available to the user. The selection can still be modified after entering the project.

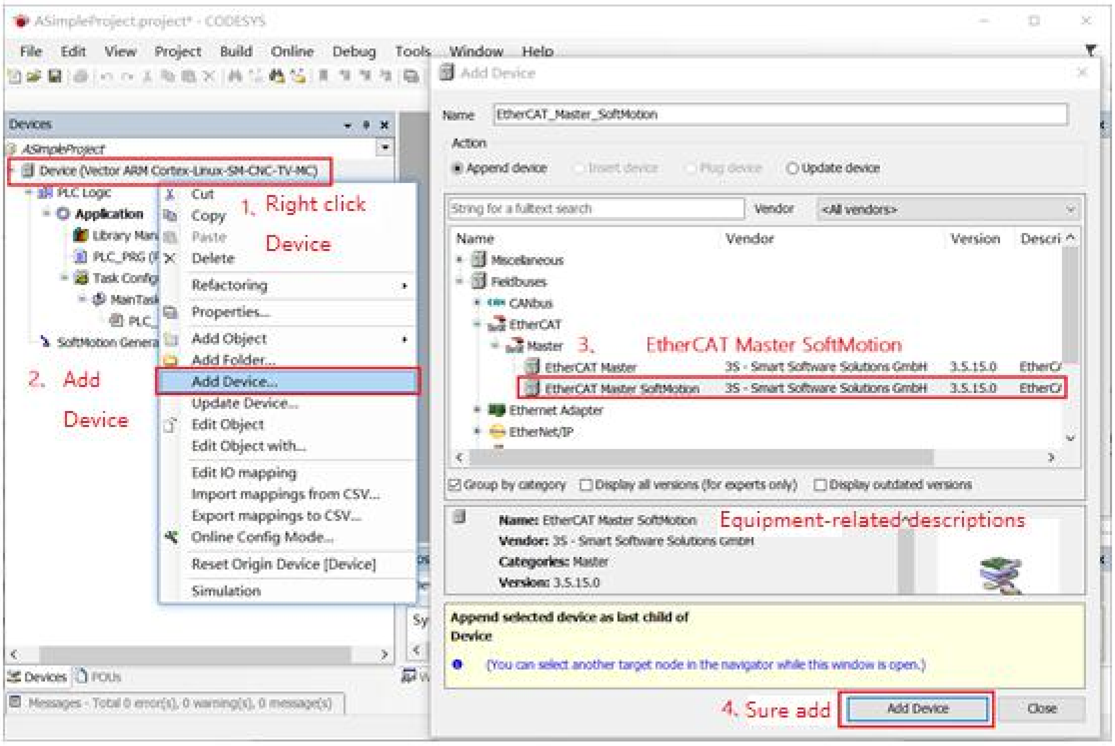
Click on "OK", and when it is finished, it will look like this



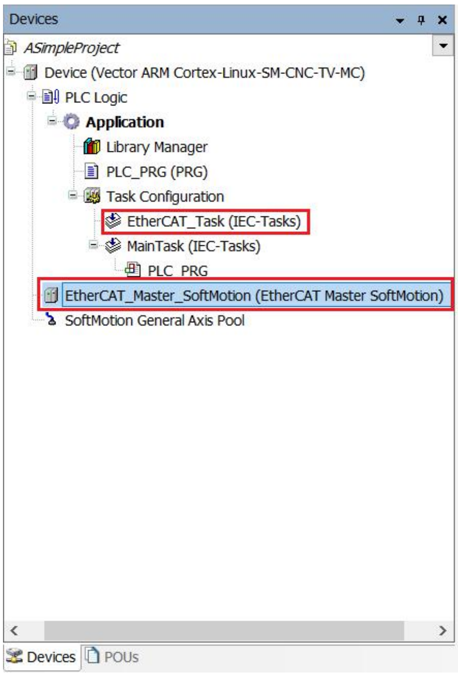
4.1.2 System configuration and parameter setting

Add EtherCAT_Master_Softmotion

The EtherCAT Master Softmotion is an EtherCAT master module with real-time motion control. How to add it: Right click "Device→Add Device→EtherCAT Master Softmotion→OK", Adding an EtherCAT master



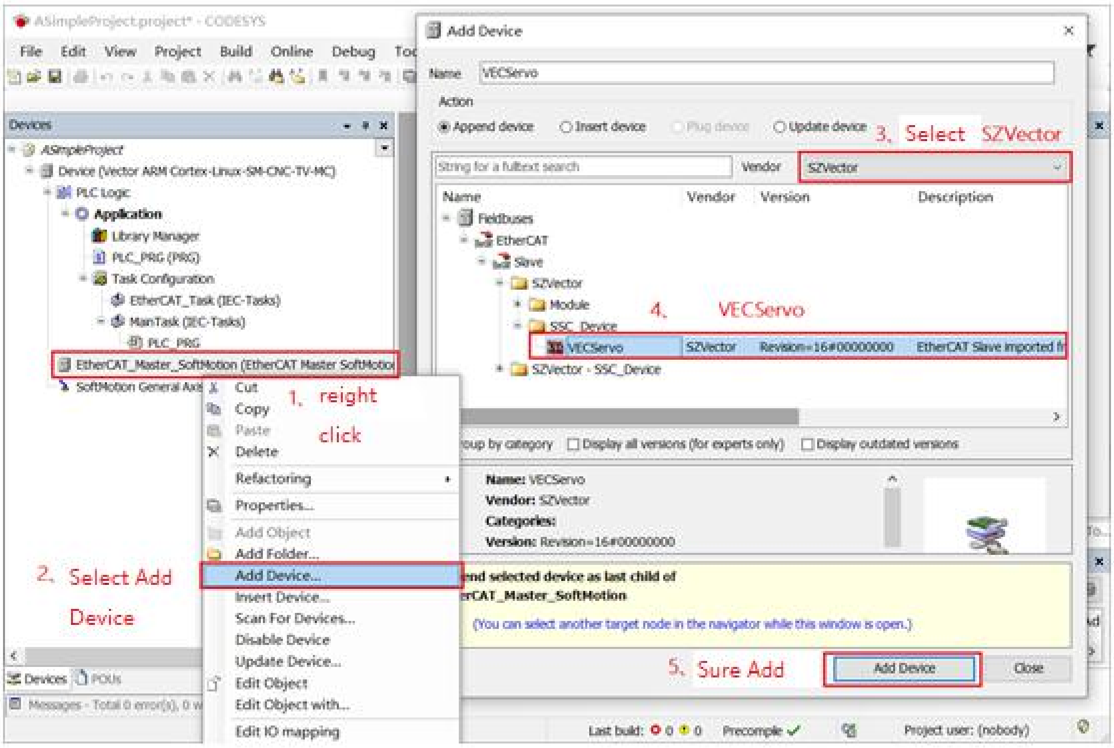
Once added, as shown in the figure, an EtherCAT_Task is assigned at the same time and the EtherCAT task-related parameters can be configured.



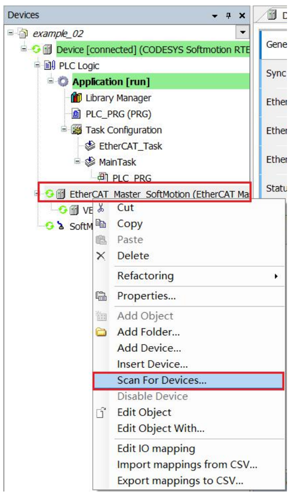
Add VECServo

After adding the master device, add the slave device below the master, in this case "VECServo (Viktor EtherCAT Servo)". The servo device description file (.XML) must be installed before adding the device. Installation process reference :2.2.5 Install the device description file。This is added as follows:

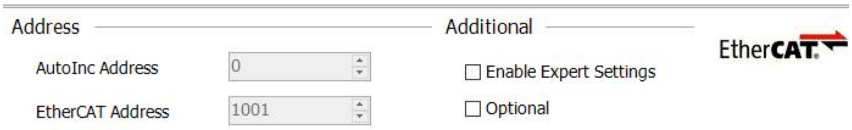
1） If you are offline without a connection to the master, you can access it by right-clicking "EtherCAT_Master_Softmotion → Add Device", Find the corresponding manufacturer and device model in the device pop-up window : "SZVector→SSC_Device→VECServo", Click on Confirm to add the device.



If you are online while logged into the main site, you can also right-click "EtherCAT_Master_Softmotion→Scan For Devices", Add by scan.

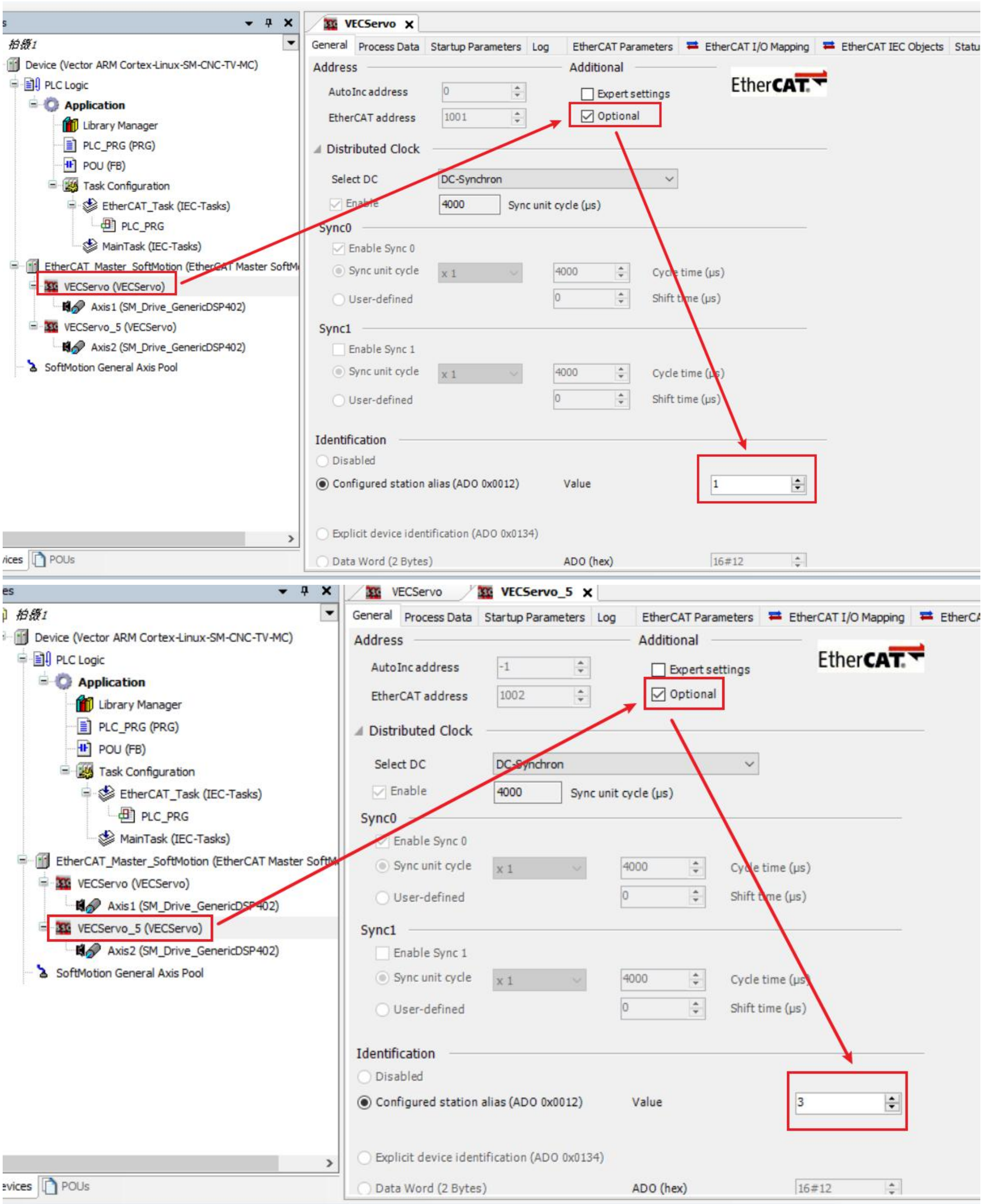


Note: The node address of the device, by default, is automatically assigned, i.e. the node address is assigned from near to far from the host, this example is not modified and is set by default.



If you need to assign node addresses manually, you can refer to the following method, using the VC bus servo as an example:

a） Assign the address as shown below: check "optional" in the additional field for the slave station, then fill in the station number (1-65535) in the configured station alias



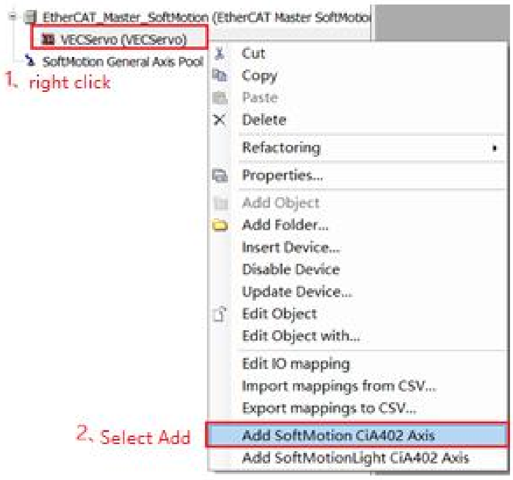
b) Set the parameters P08.41 (servo station number) of 1 and 3, respectively, to reset the servo or power up again.

c) Then log in to the device Download the program, and if the first connection is not successful, reset and run again.

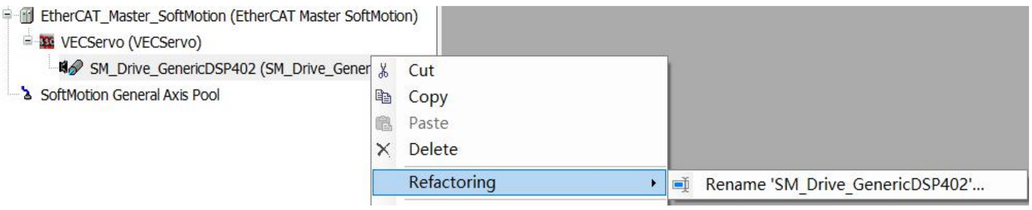
d) As long as the alias from the station is the same as the alias configured for the backgroundproject, it will work regardless of the order.

Add CiA402 Axis

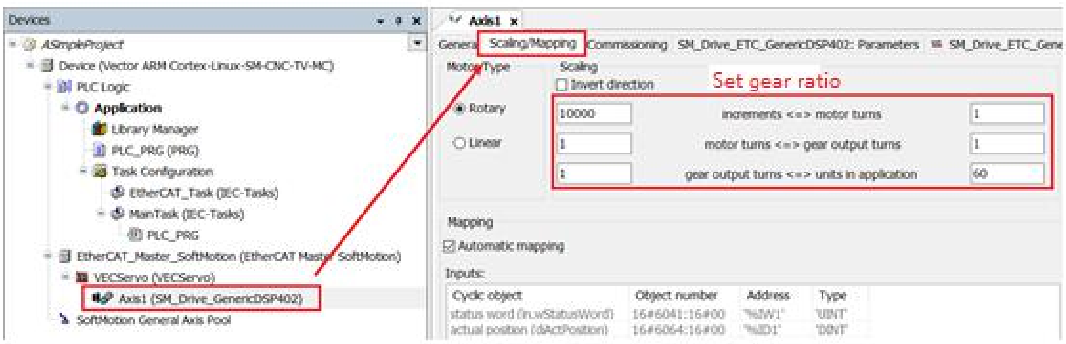
1) The device runs in association with the axis, adding method: right-click "VECServo→Add SoftMotion CiA402 Axis"to add the motion control axis, as shown below.



After adding the following image, rename the axis "Axis1" for programming convenience



2) Set the control related parameters, double-click Axis1, open the parameter configuration page, set the gear ratio

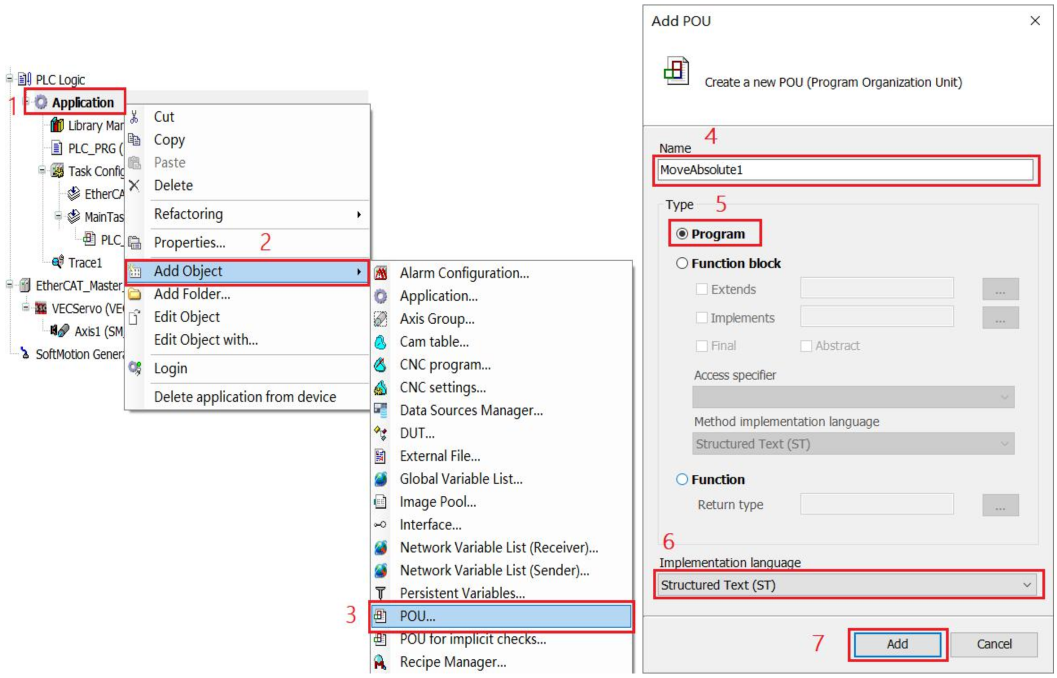


4.1.3 The user controls the program writing

Write a program here that enables the controller to control the servo motor to perform absolute position commands and make round-trip movements.

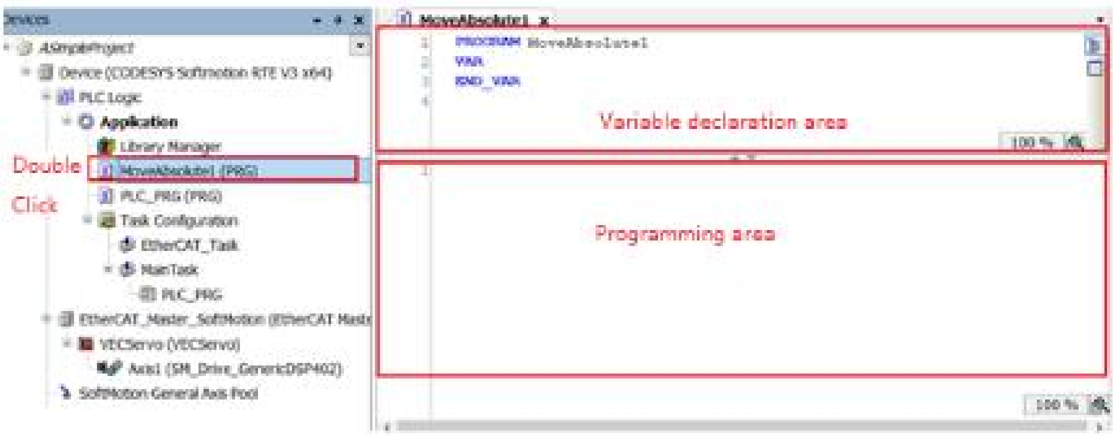
Create an object

As shown below, right-click the mouse "Application→Add Project→POU",name the newPOU "MoveAbsolute"in the spring window, type select "Program", programming language select "Structured Text (ST), click "OK" to complete the addition.



Open the programming environment

Double-click to open MoveAbsolute, as shown below, and the programming interface includes the variable declaration area and the programming area.

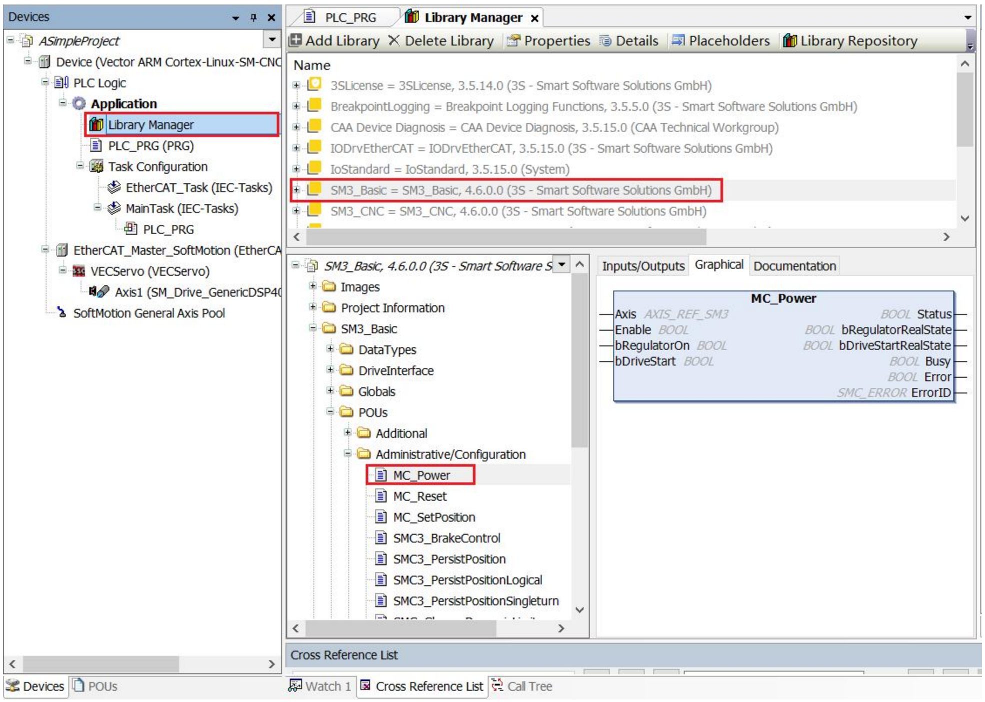


Define variables

Add variables in the variable declaration area, and the variable declaration code is as follows.

```
PROGRAM MoveAbsolute1
VAR
    iStatus:INT;
    Power:MC_Power;            //Enable module
    MoveAbsolute:MC_MoveAbsolute;   //Absolute displacement module
    p:REAL:=180;  // Displacement value
    ActPos:LREAL;   //The actual location value
END_VAR
```

It's important to note here that engineering Library Manager Whether the library "SM3_Basic"is added to the library is generally added by default, if not, you need tomanually right-click "Library Manager→Add Library",find the library "SM3_ Basic" and then choose toadd,or you can add morelibraries in this way.



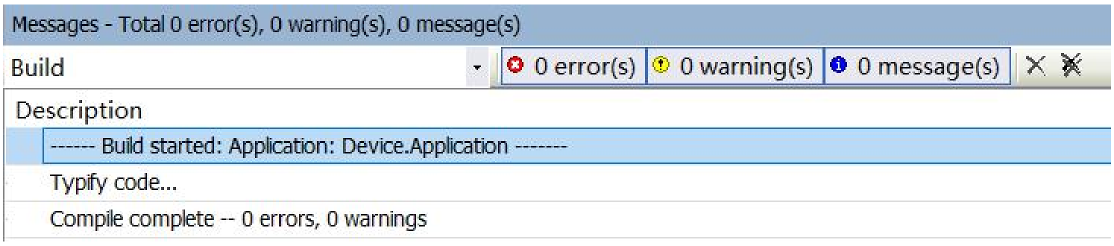
Program writing

The program is added in the programming area as follows. (Program function: when the program is executed, immediately enable servo, and so on servo enable success, control motor between position P and starting point 0 to do round-trip movement.))

```
CASE iStatus OF
0:                              // Power-on automatic enable servo
    Power(Axis:=Axis1, Enable:=TRUE , bRegulatorOn:=TRUE, bDriveStart:=TRUE );
    IF Power.Status THEN
```

```
            iStatus:=iStatus+1;
      END_IF
1:                                        // Walk absolute displacement and run to P
      MoveAbsolute(Axis:=Axis1,  Execute:=TRUE,  Position:=  p,  Velocity:=100  ,  Acceleration:=  100,
Deceleration:=100    );
      IF MoveAbsolute.Done THEN
            MoveAbsolute(Axis:=Axis1, Execute:= FALSE);
            iStatus:=iStatus+1;
      END_IF
2:                                        // Walk absolute displacement and run back to 0
      MoveAbsolute(Axis:=Axis1,  Execute:=TRUE,  Position:=  0,  Velocity:=100  ,  Acceleration:=  100,
Deceleration:=100    );
      IF MoveAbsolute.Done THEN
            MoveAbsolute(Axis:=Axis1, Execute:= FALSE);
            iStatus:=1;
      END_IF
END_CASE

ActPos:= Axis1.fActPosition;              // Read the actual location value
```

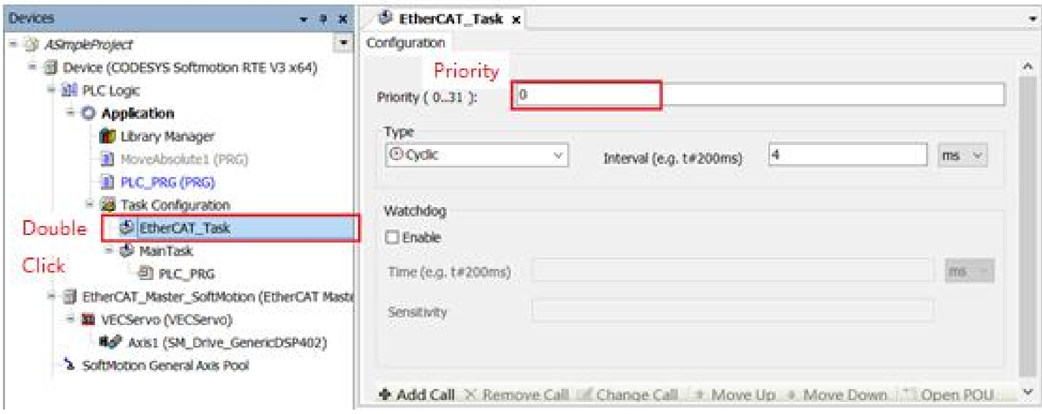
Once the program is written, click Compile , Make sure it's written correctly.



4.1.4 Bus and task cycle

Bus task cycle

When you add EtherCAT Master SoftMotion,the project automatically adds bus tasks EtherCAT_Task,setting bus execution and cycle times, and taskpriority (0 to 31,0 isthe most advanced), where EtherCAT _Task priority isset to 0, and other tasks, such as Main_Taskpriority, are set to 1 to 31.

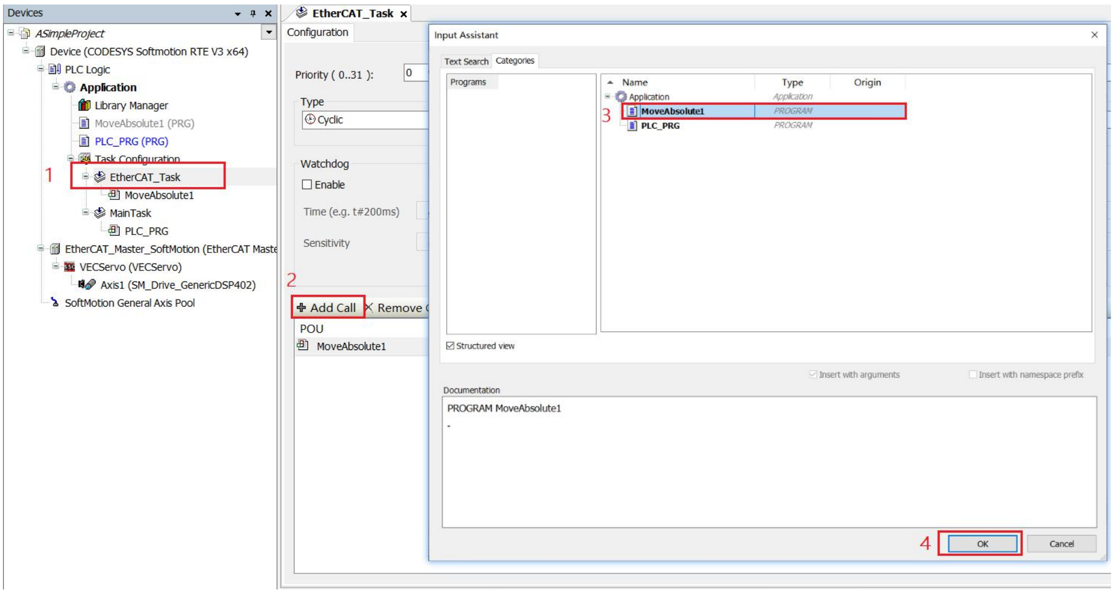


The program task cycle

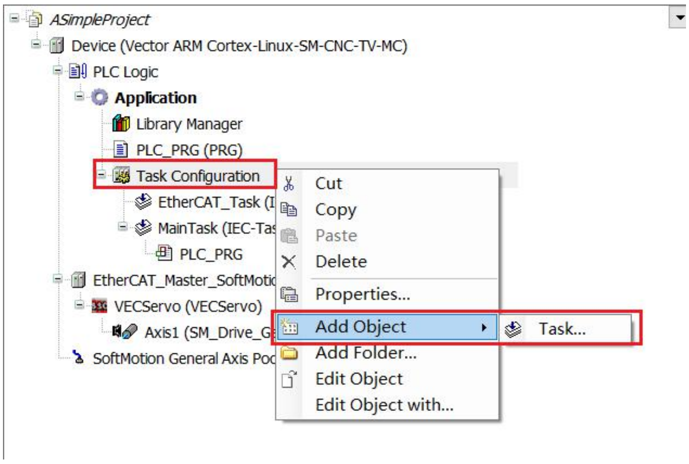
Once the program is written, you need to add the program to the task and configure it.

Motion-related POU recommendations are added toEtherCAT _Task, and logic or computational-related POU recommendations are added to other Tasks (e.g. Main_Task, a program "PLC PRG" and a task "Main Task" have been established by default when the project is started, and "PLC PRG" has been added to "Task Task".)。

The new POU object "MoveAbsolute" needs to be manually added to theEtherCAT _Task task bydouble-clicking "EtherCAT_Task Add → Call", selecting "MoveAbsolute1"andclicking "OK".



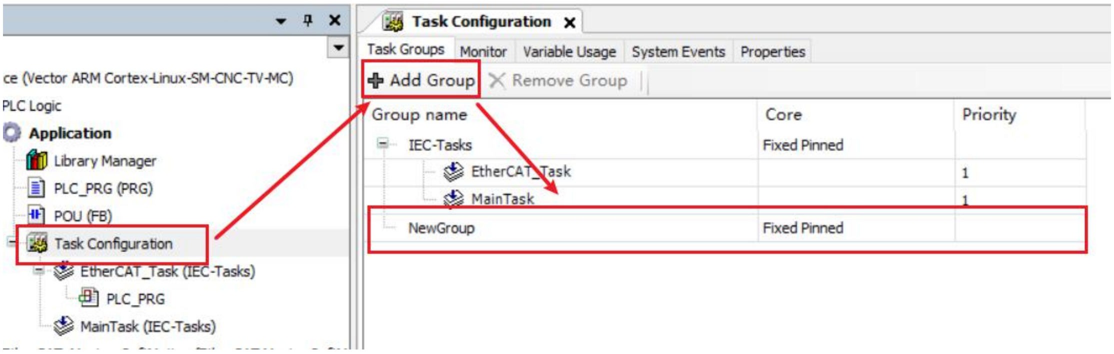
In addition to the default tasks, you can add new tasks yourself, as follows: right-click Task Configuration, select Add Project→Task,you can add newtasks,and double-click tasks to configure tasks.



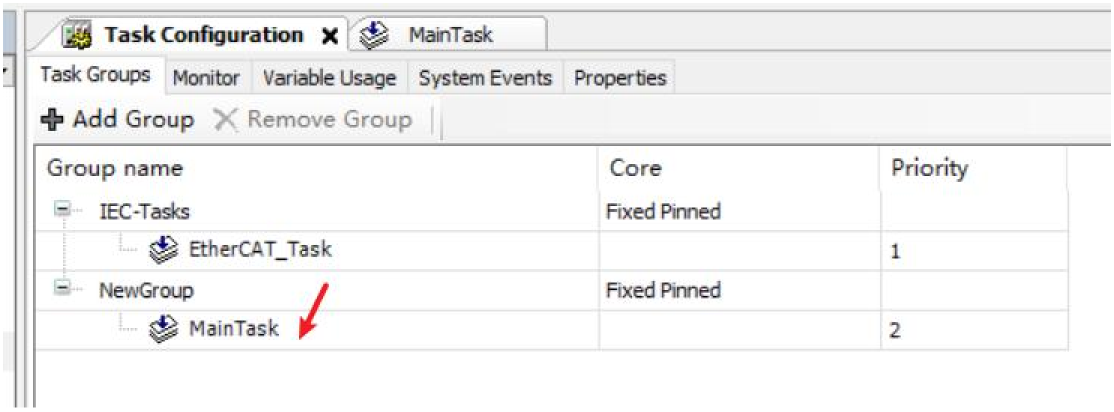
4.1.5 Mission sub-core

The VE motion controller is designed with a four-core core, which allows the bus to be sub-coreed for smoother operation. The steps are as follows

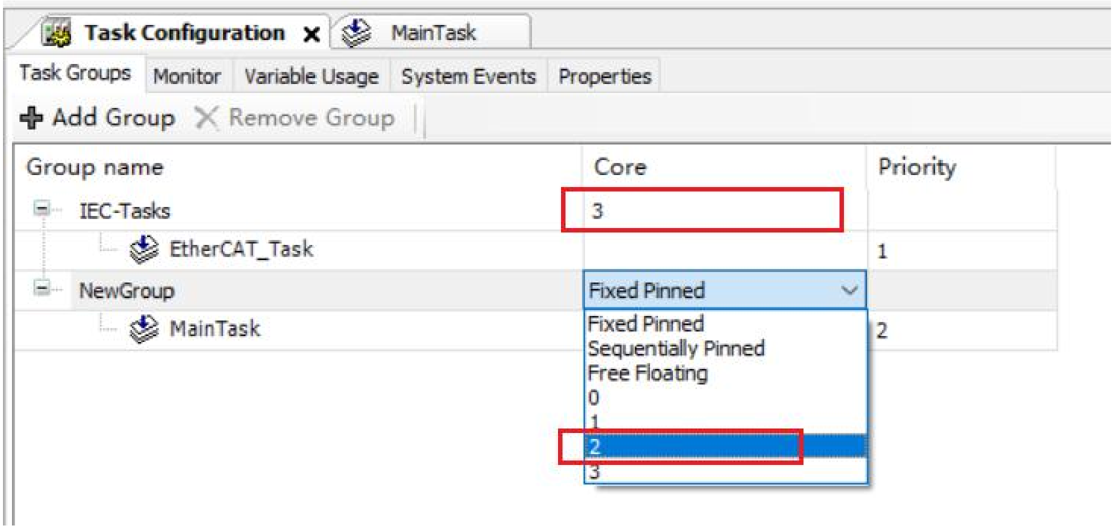
(1) Open Task Configuration, click Add Group, Add NewGroup



(2) Add other tasks to a new group, and EtherCAT_Task a separate group



(3) EtherCAT_Taskassigned to the 3rd core and other tasks to the 2nd core to ensure the stable operation of the EtherCAT mission



Attention:

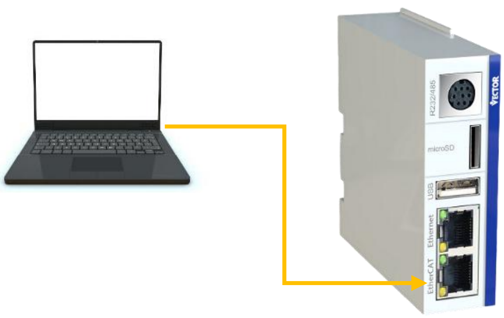
When the Modbus device sets up tasks, it cannot be addedto the _Task and needs to be added to other tasks.

4.1.6 Sign in to the device

Connect the controller

The environment in which CODESYS is run on a PC, communication with the VE controller, user-ordered downloads, start-stop and monitor the operation of user programs, parameter viewing or modification, and so on.

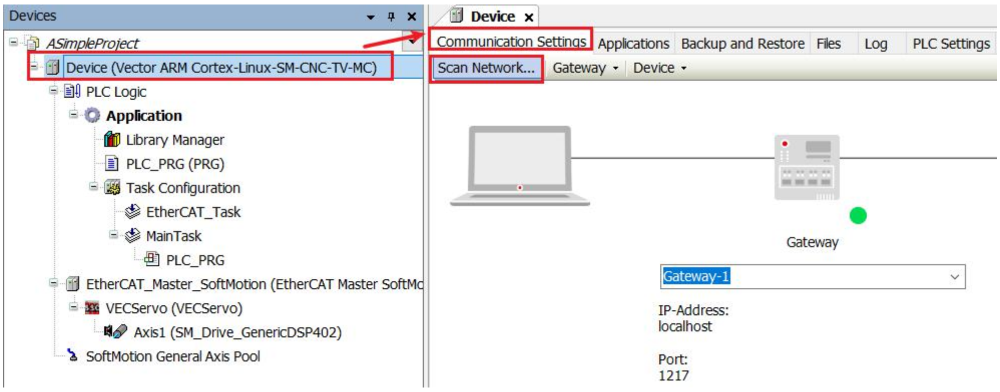
The VE controller can currently be logged in via the LAN LAN network, a 1-to-1 direct connection between the PC computer and the VE controller can be made over a network cable, or online via a router or hub, in which case one PC can be connected to multiple VE controllers or multiple PCs can access the same VE controller.



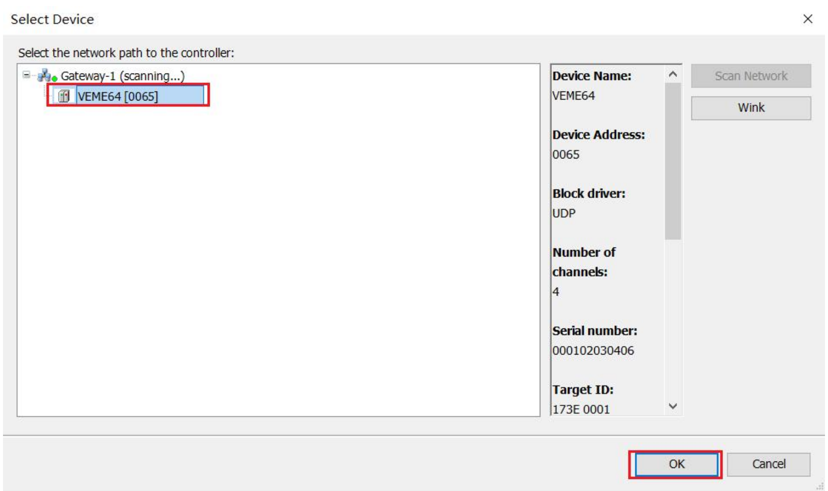
The IP address of both the PC computer and the VE controller must be the same network segment by default to log on to the VE controller, otherwise the VE controller will not be scanned in CODESYS. The factory default IP address for VE controllers is 192.168 1. 。 123,if the IP address of the PC is 192.168. 1.xxx, (here xxx represents the range of 1 to 254, but not the same as the END address of the VE controller IP), then CODESYS can scan to the VE controller, and can interact with the data, download the user program, run monitoring, etc. If the IP of the VE controller has been man-made, its address is not in the IP address segment where the PC is located, the PC cannot be accessed, the VE controller's IP address can be restored to the factory default IP address:192.168 1. 123,and then change the address of the PC machine to 192.168. 1.xxx, with which you set up a 1-to-1 online, you can modify the address of the VE controller to the desired IP segment address.

Scan the network

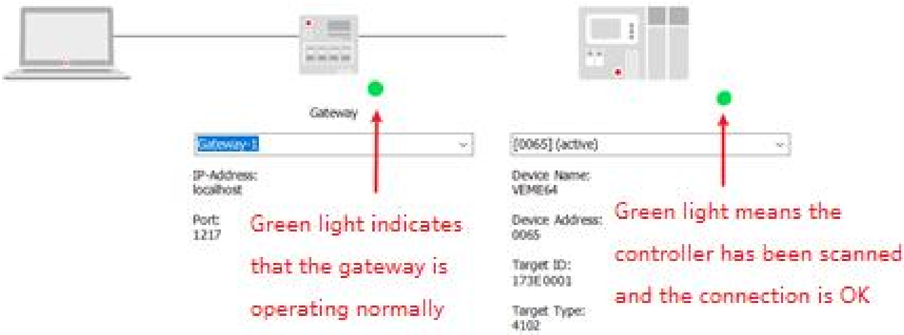
Double-click "Device" in the engineering tree and pop up the following interface



On this screen, the mouseclicks on the "Communication Set Scan→network" tab, pops up the following interface, scans to the VE controller, clicks on its name on the left side of the window, can see its introduction information on the right side of the window, click OK, can connect the device:

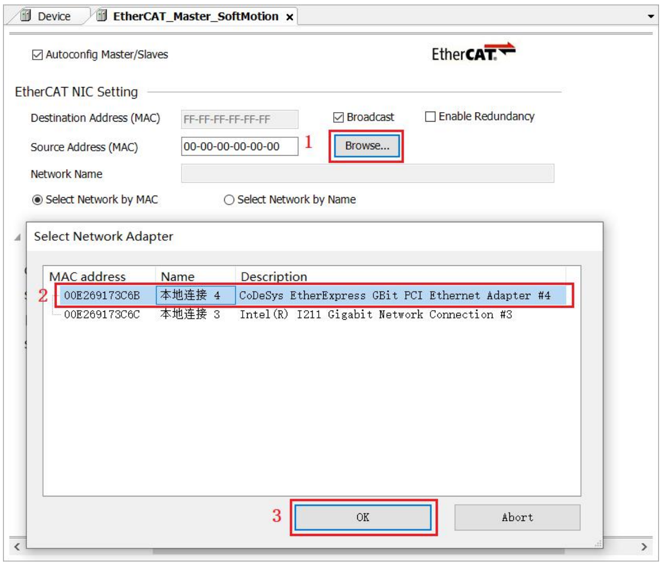


After logging in, you can modify the device name according to your own needs, change to a device name that is easy to identify, can be easily identified, in the application of multiple controllers, very helpful.



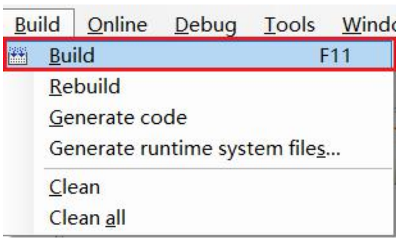
Set the bus control gate

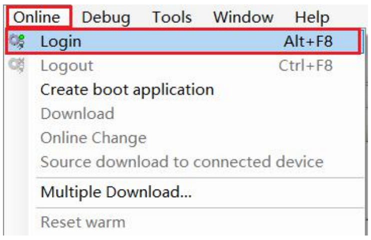
Double-click "EtherCAT_Master_SoftMotion"to set up the EtherCAT network card, as shown, click on"Browse",select the Name of the EtherCAT network card in the spring window (connect to the servo endnetwork port), click "OK"



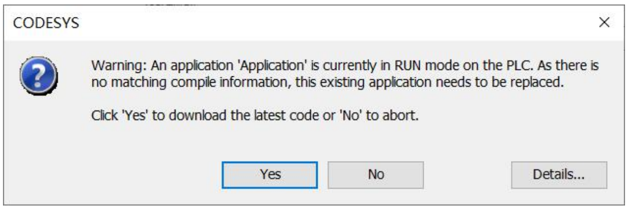
Sign in to download

Click on "Build." ", compile the error-corrected, and then click "Login." "



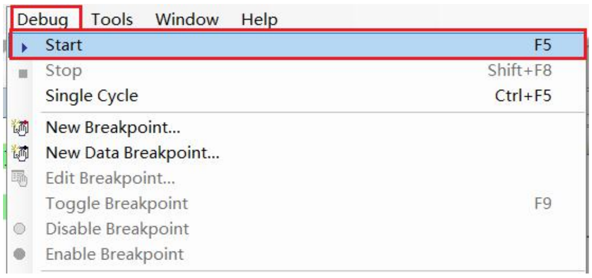


Pop up the dialog box, select Yes, and download the program to the controller.

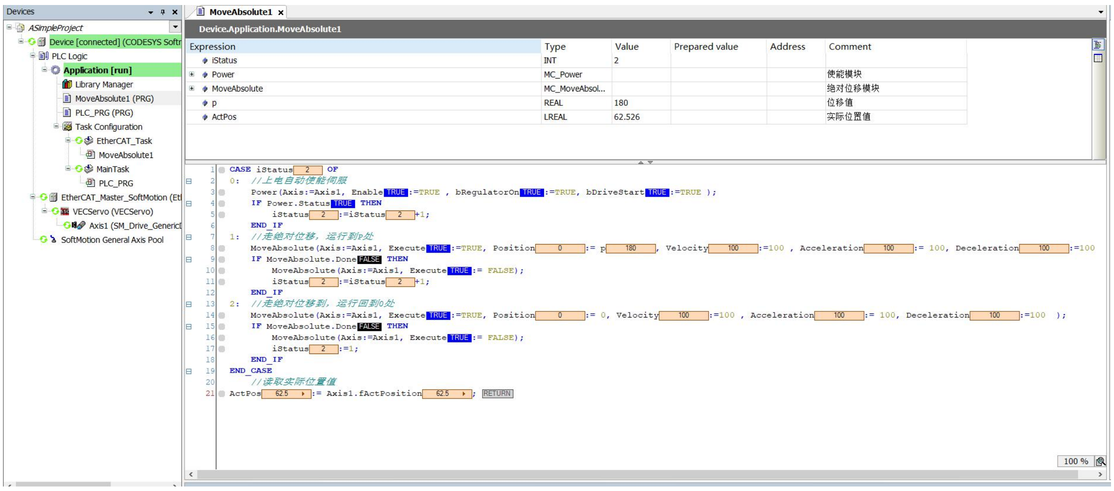


4.1.7 Start debugging

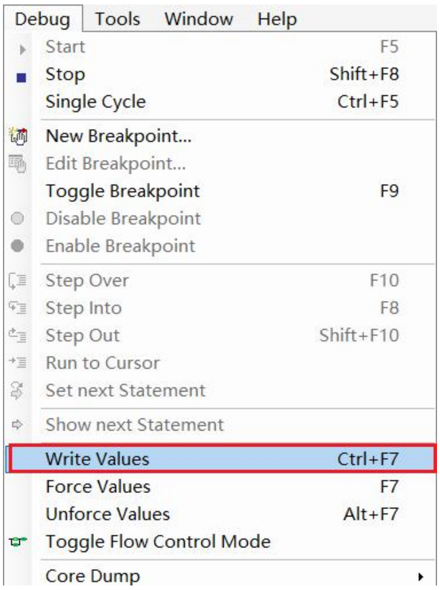
After the login is successful, select"Debug→Start ", The controller is up and running.



Open MoveAbsolute, the program runs as shown in the following image, after the program performs servo enabling, the motor between position P and starting position 0 to do round-trip movement.



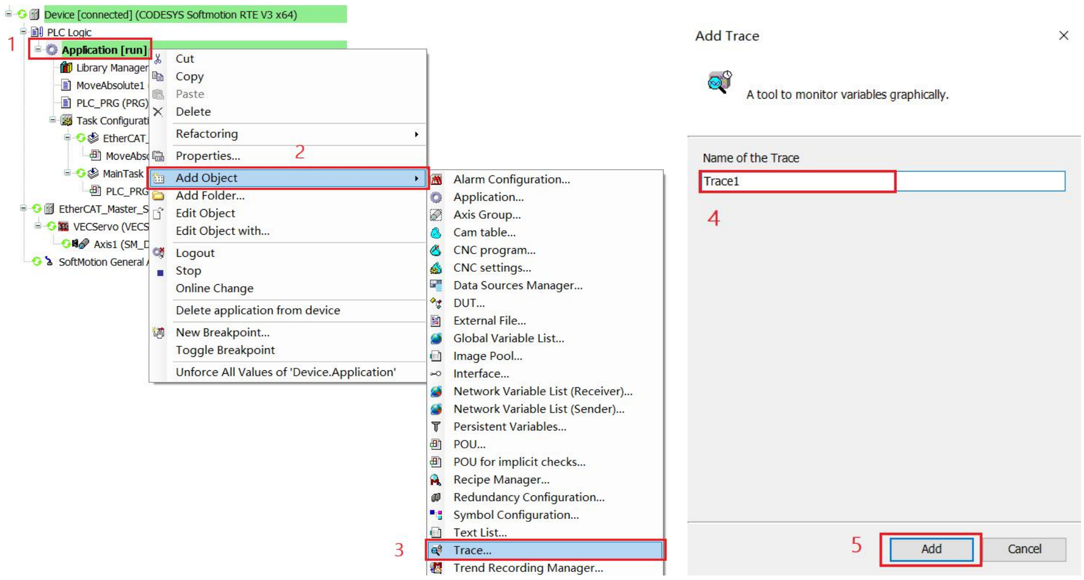
Modify the value of location P online: Click the preset value of "Prepared Value" for the variable "P" to enter thevalue "360" and then select"DebugWrite Values" or the shortcut "Ctrl-F7" to write the value to "Value" to modify the value of the variable→ "P" online.



4.1.8 Add a Trance trace

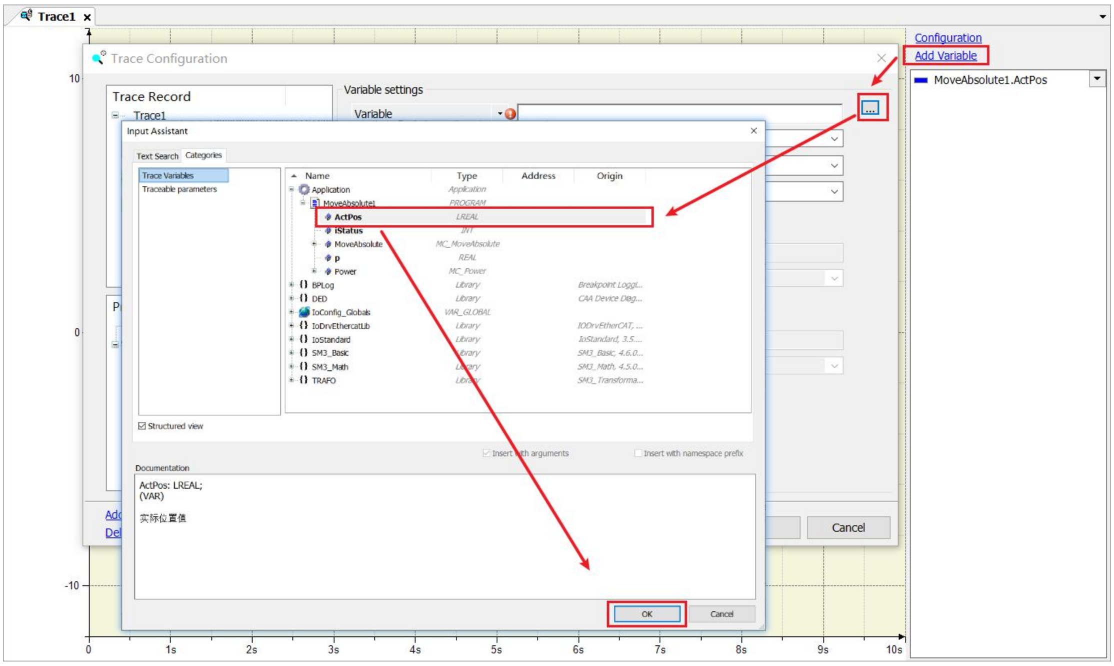
Add Trance

To more intuitively observe changes in the position of the servo axis, a logic analyzer is added to record the motion curve. Right-click "Application", select "Add object → Trance",pop up the dialog box, name it and then click"Add" to add Trance(tracking), as shown below.

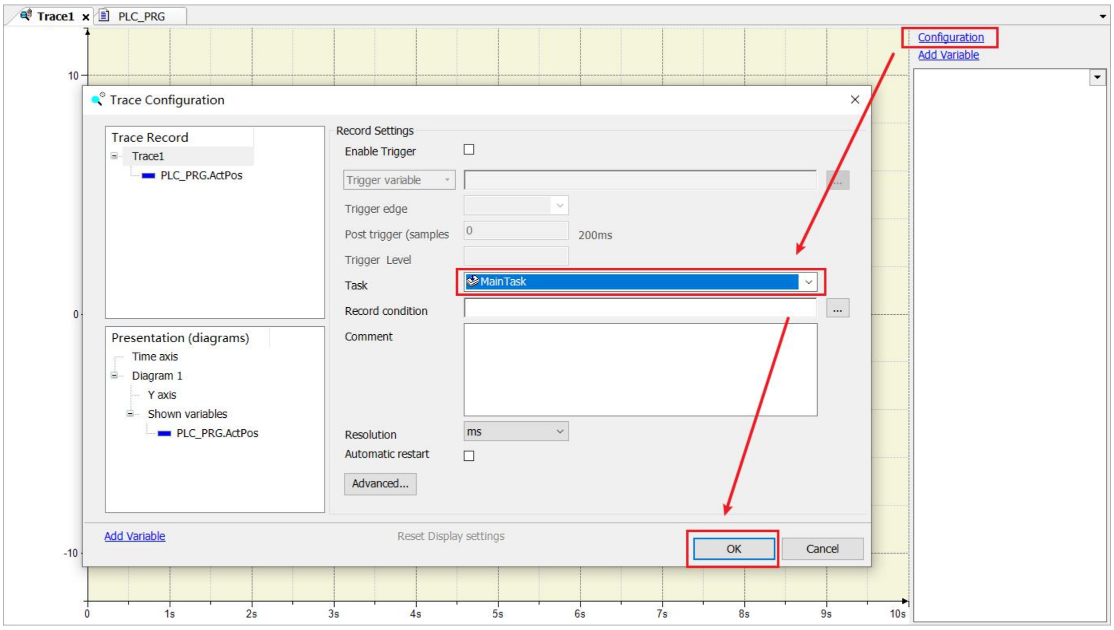


Configure Trance

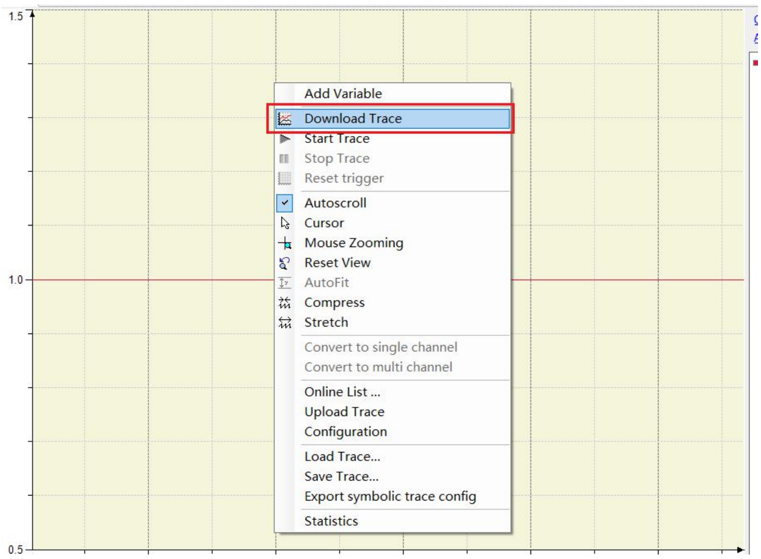
1) Click on add_Variable(add variables) and select the button ..., Find the variable "ActPos" in the variable pop-up window, click "OK" and add it to the tracker.



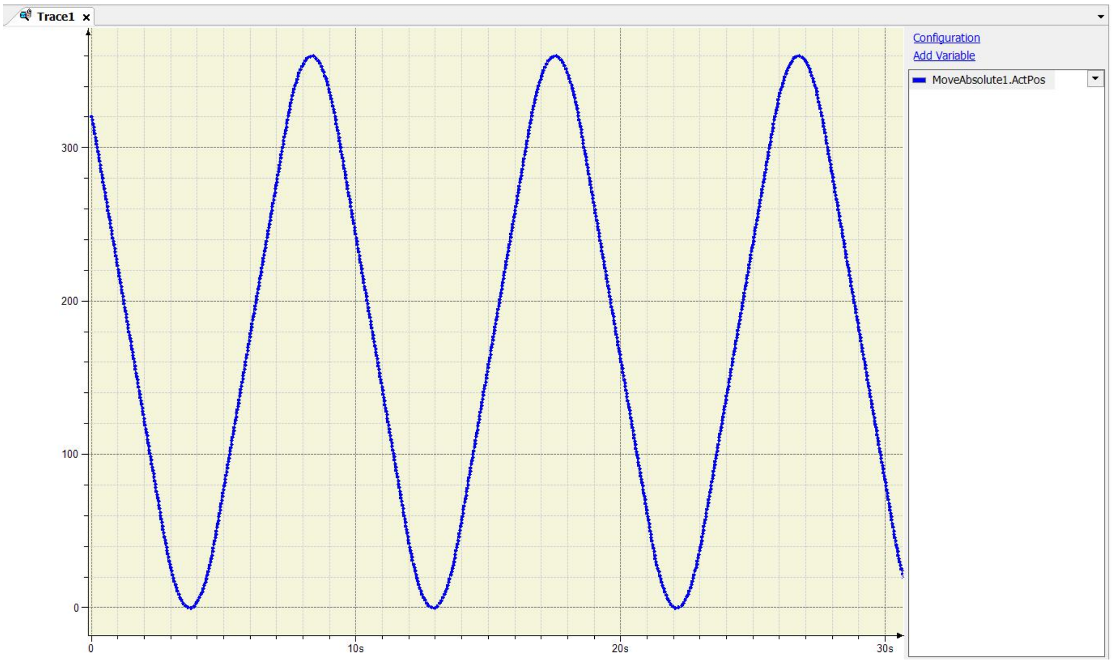
2) Click "Configuration" and select"Main Task" inthe Taskoption, click "OK";



3) Right-click on the oscillostor blank interface, select "Download Trace", download tracking;

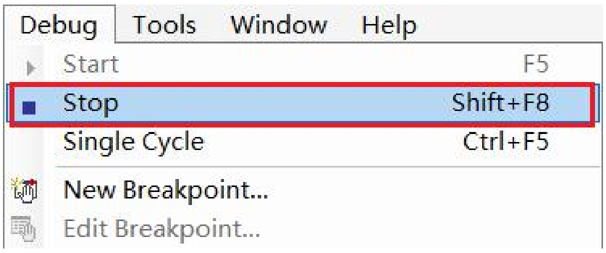


4) Axis1's actual position curve is printed below.



4.1.9 Stop debugging

Once the commissioning is complete, click "Debug→Stop." " ", Stop executing the program



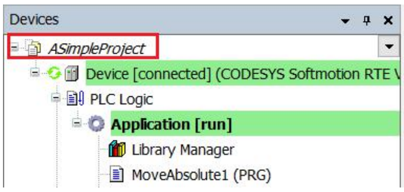
4.2 Common configuration instructions for devices

4.2.1 Device tree and device editor

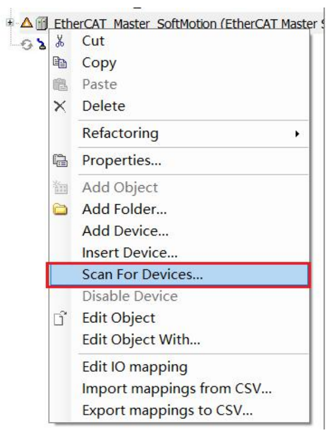
The device tree

In device views, also known as device trees, applications can be organized based on the target device. In this view, you can view PLC hardware and field bus systems, configure hardware communication, and assign applications.

The root node of the device tree is a symbol nodeentry: Here's what it is

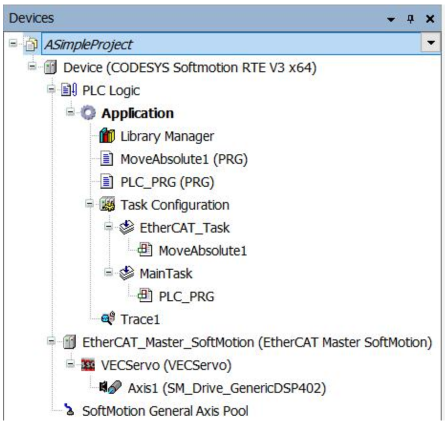


Insert a device object, also known as a target system, under this node of one or more PLCs. Each device object represents a specific hardware component, such as a controller, field bus, bus coupler, driver, I/O module, or monitor. If you are already connected to a controller network, you can scan the hardware to find available devices and save them to the currently configured device tree, as shown.



Each device is defined by the device description file and must be installed on the local system to be plugged into the device tree. Device description files define device properties for configurability, programmability, and possible connections to other devices.

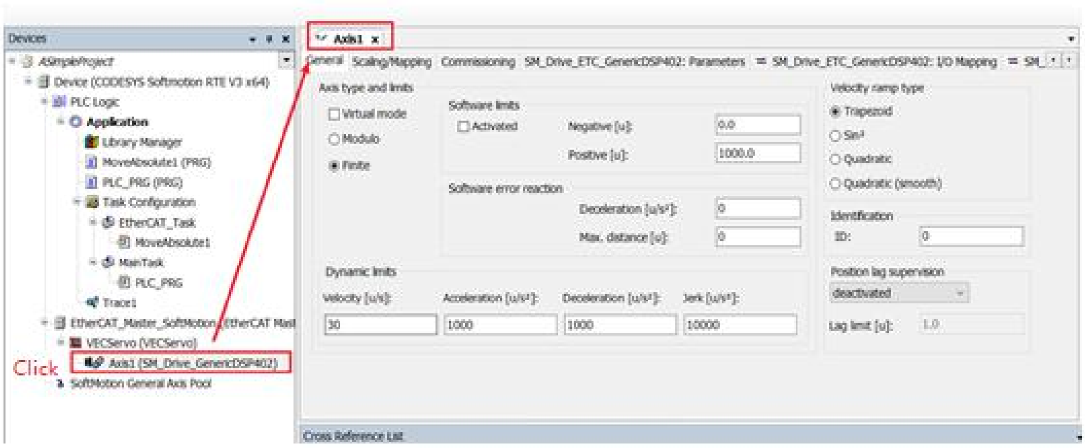
Example of a device tree:



Device entries in the device tree consist of device symbols, device names, and device types, such as:

。

Device communication, number of participants, and IO mapping can be configured in the device editor dialog box. Double-click the device object to open the editor.



The device tree in online mode

When CODESYS is in online mode, the current symbol of the device bar indicates the device state:

:The PLC is connected, the application is running, the device is running, and the data is being exchanged.

:The PLC is connected and in the STOP state.

:The PLC is connected and the application is running. Diagnostic information is available.

: The device is in pre-operation mode and is not yet running. Diagnostic information is available.

: The device did not exchange data, the bus was incorrect, and it was unable to enter configuration or simulation mode.

: The device runs in demo mode for 30 minutes. After this time, the demo mode will terminate and the field bus will end the data exchange.

: The device is configured, but not fully operational. There is no data exchange.

: Redundancy mode is active. The field bus master does not send any data because the other master is active.

: The device description could not be found in the device repository.

: The device itself is running, but the child device is not running. The child device is not visible because the device tree is collapsed.

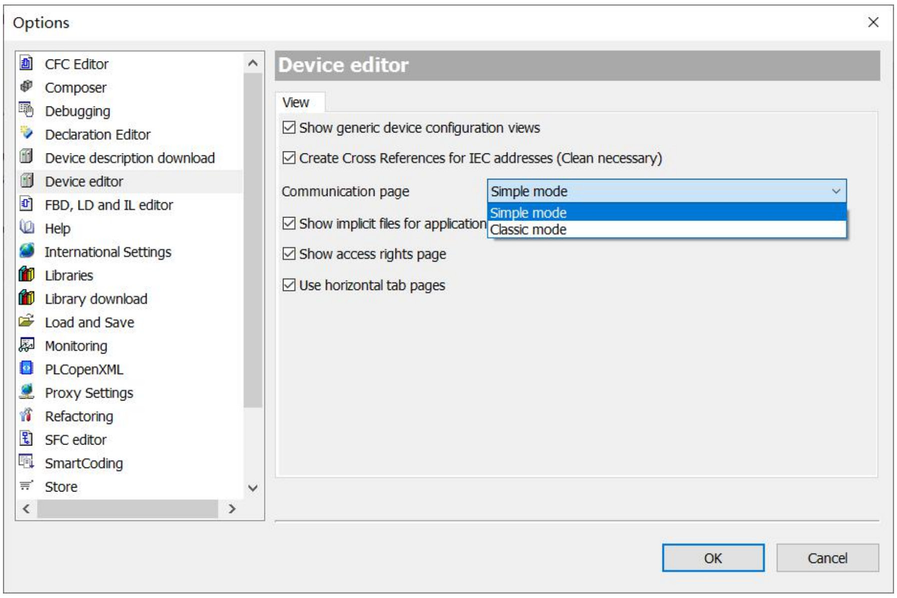
The names of all connected devices and applications are highlighted in green

The name of the device running in analog mode is displayed in italics: 

Additional diagnostic information is located on the Status status tab of the device editor.

The device editor

Double-click the device object in the device tree to open the editor. The editor includes regular labels and specific labels, and its title contains the device name. Click Tools→Options Device→Editorin the toolbar to open the device editor option to set the style or content of thedevice editor.



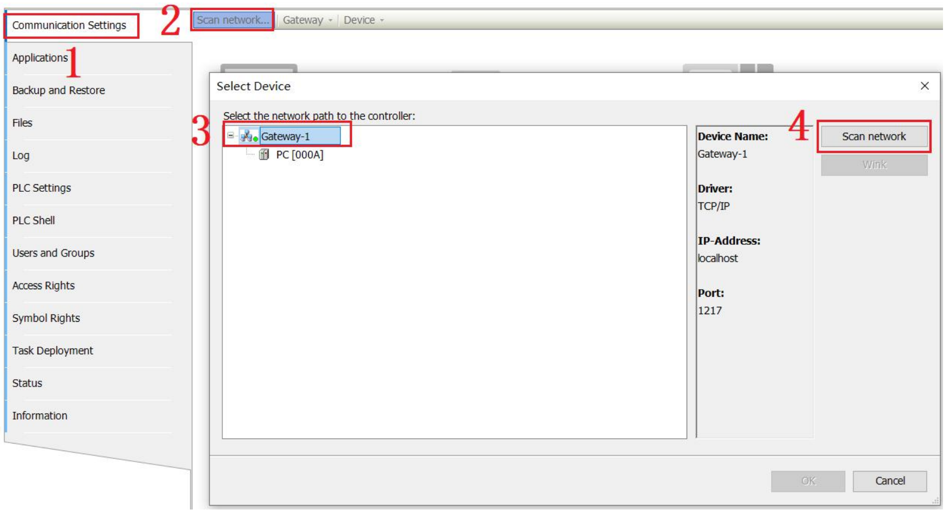
4.2.2 Device device

CommunicationSetting communication settings

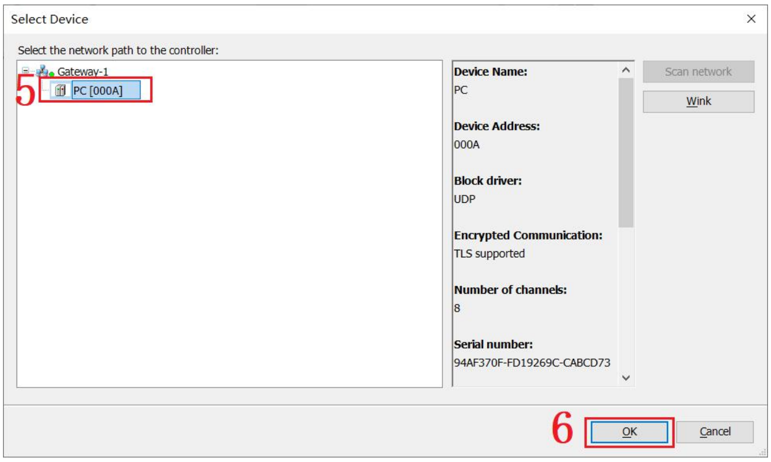
In this tab of the Universal Device Editor, you define the connection between CODESYS and the device in which the application should run.

Scan network: Scan the network

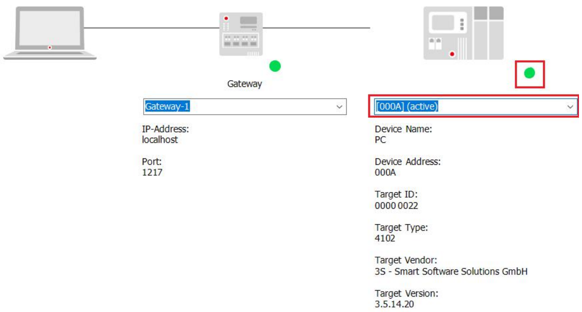
The scan network steps are as follows, click "Gateway-1"(gateway) and click "Scan network" to scan the network



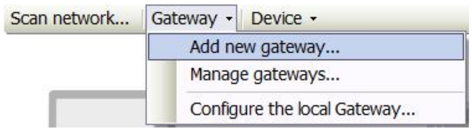
Click on the scanned device name, such as"PC . . .000A"and then click OK



When the device status light is green and the status is described as Active, the PC is connected to the device

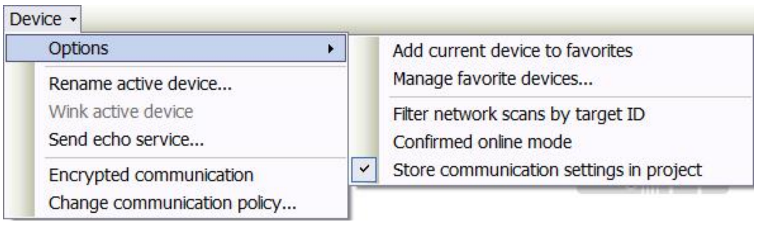


Gateway Gateway:



You can add, manage, or configure a local gateway

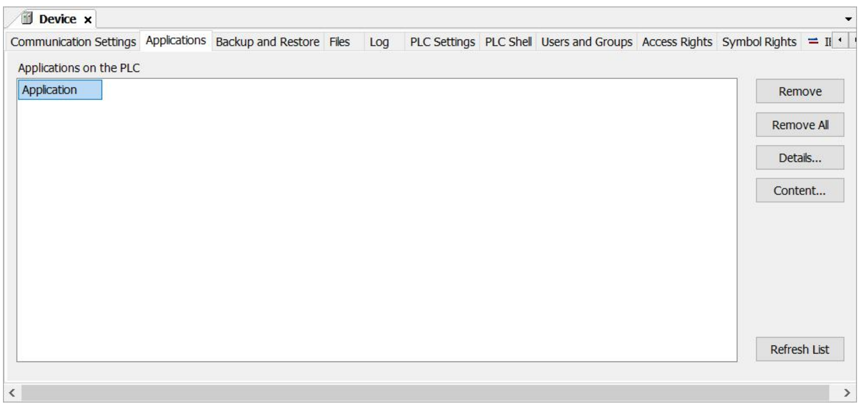
Device:



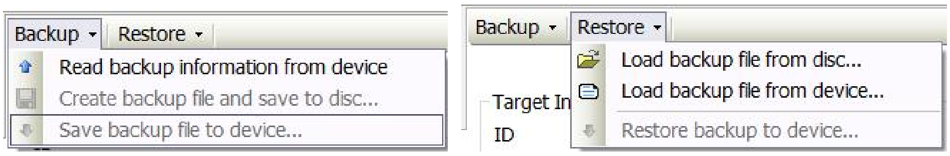
Filter network scans by target ID: Filter network scans via device ID (unchemed).
Store communation settings in project:Save communication settings to project (check).

Applications app

On this tab of the Universal Device Editor, you can see which applications exist on the device. Depending on the system, you can remove the application from the device or retrieve details about the application.



Backup and Restore backup restore



Read Backup Information fromDevice: Read backup information from the device The command searches for application-specificfiles from the PLC's PlcLogicdirectory and lists them in a table at the bottom of the tabbed page.

Create Backup File and Save to Disk: Read backup information commands from your device to determine which files are relevant to the backup. This command compresses the files and meta.info information files in the table set to Active into backup zip files. The file extension is tbf ("target backup file").

Save Backup File to Device: Save backup files to disk. This command saves the backup file to the TBF directory of the PLC.

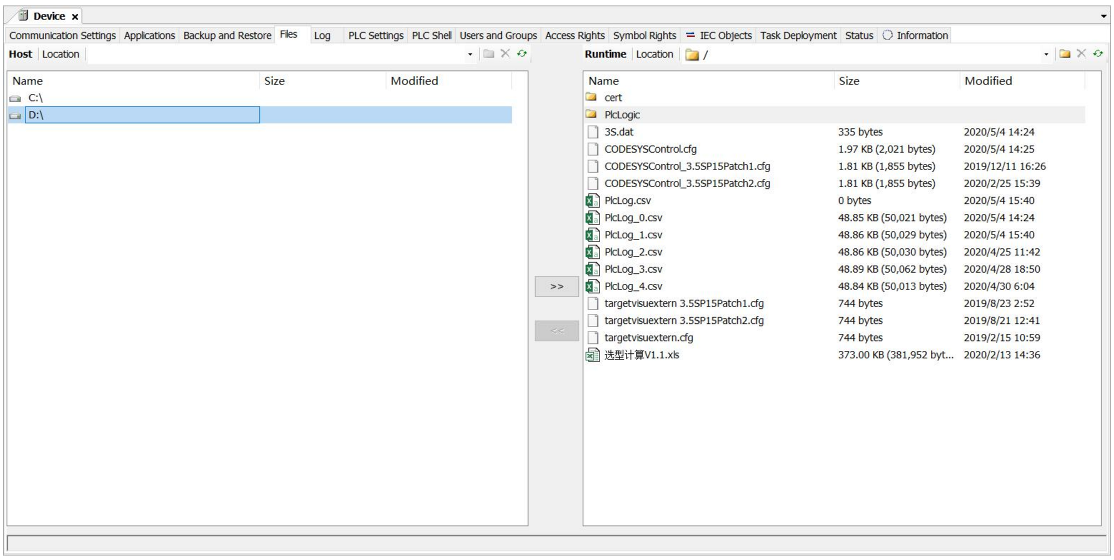
Load Backup File from Disk: This command opens a dialog box to navigate through the file system for saved backup files.

Load Backup File from Device: This command generates a list of all backup files found on the PLC. Select one of these files to view its contents in a table on a tabbed page.

Restore backup to Device: If at least one component of the backup file currently loaded on the tabbed page is set to active, this command is available and prompts to restore the state of the application on the device.

Files file

In this tab of the Device device editor, files can be transferred between CODESYS(PC host) and PLC. If the communication is set up correctly and the PLC is online, CODESYS automatically establishes a connection to the PLC during file transfer.



Log log

View the PLC log. It lists the events logged on the target system. This involves:

- Events during system start-up and shutdown (loaded components with version number)
- Download and load the launch application
- Custom entries
- Logs from I/O drivers
- Logs from the data source

PLC Settings PLC settings

Basic settings for PLC widding, such as the processing of inputs and outputs and bus cycle tasks.

Update IO while in stop: When checked, CODESYS refreshes the values of the input and output channels even if the PLC is stopped. If the gate keeper detects a fault, the output is set to a predefined default. When not checked, CODESYS does not refresh the values of the input and output channels when the PLC is stopped.

Behavior of the outputs at stop: The processing of the output channel when the controller enters a stop state

- Retain values: Keep the value, keep the current value.
- All outputs to default value: All outputs are default, and the default values are assigned based on I/O mapping.
- Execute program: Executes the program, controls the processing of output valuesthrough the program contained in theproject, and CODESYS executes the program at STOP. Enter the name of the program in the field on the right.

Always updatevariables: Define whether CODESYS updates the I/O variables in the bus cycle task. This setting is valid for the I/O variables of the from the station and module only if it is defined as Disabled in the update settings for the station and module.

- Deactivated (update only if used in a task): Deactivated (updated only when used in tasks),CODESYS is updated only when the I/O variable is used in tasks.
- Activates 1 (use bus cycle task if not used in another task): Activate 1 (use bus loop tasks if they are not used in other tasks) and codeSYS update the I/O variables in bus loop tasks if they are not used in other tasks.
- Activate 2 (always in bus cycle task): Activate 2 (always in the bus loop task): CODESYS updates all variables in each loop of the bus loop task, whether or not they are used and mapped to the input or output channels.

Bus cycletask: The task of controlling bus cycles. By default, enter tasks defined by the device description.

By default, the bus cycle settings for the parent bus device (the cycle usage settings for the parent bus) are applied, i.e. the device tree is scanned up to find a valid bus cycle task definition.

Users and Groups users and groups

On this tab of the Universal Device Editor, you can edit the controller's device user management. Depending on how the device is supported, you can define user accounts and groups of users. Combined with the configuration on the Access tab, you can control access to control objects and files at runtime.

Sync:Turns synchronization between editor and user management on the device on and off, and if the button is not pressed, the editor is blank. If you press this button, CODESYS continuously synchronizes the display in the editor with the current user management on the connected device.

Import fromdisk: Used to select and import user-managed configurations from the hard disk.

Users

Add:Open the Add User dialog box to create a new user account

Import:Open the dialog box to import the user.

Groups

Add:Open the dialog box to add groups. Define a new group name, and select the users that belong to that group from the list of defined users.

Import:Open the dialog box to import the user.

Access Rights access

On this tab of the device editor, define device user access to the device for objects on the controller. As with project user management, users must be a member of at least one user group and can only grant certain access rights to user groups.

Symbol Rights symbol permissions

In this tab of the Universal Device Editor, different user groups (clients) are defined for access to the individual symbol sets available on the controller.

Requirements: User management must be set up on the PLC. An application has been downloaded to a controller that defines a set of symbols for it in the CODESYS project. They have access data to log on to the controller.

In the Symbol Set view, all symbol sets are listed under the Application node, the definition of which is downloaded to the controller with the application. In the Permissions view, the user groups defined in the controller's user management are listed in the list of tables. When you select a symbol set, you'll see the user group's access to that symbol set。 : Grant access; : No access rights have been granted. Access rights can be changed by double-clicking on the symbol.

Click button to save the current access configuration to an XML file. The file type is device symbol management file (* .dsm).Click button to read such a file from the hard drive.

Task deployment task deployment

The device editor's sub-dialog box displays the input and output tables and their assignments to defined tasks.

This information becomes visible only after the code is generated for the application. It is used for troubleshooting because it shows where inputs or outputs are used in multiple tasks with different priorities.

Status status

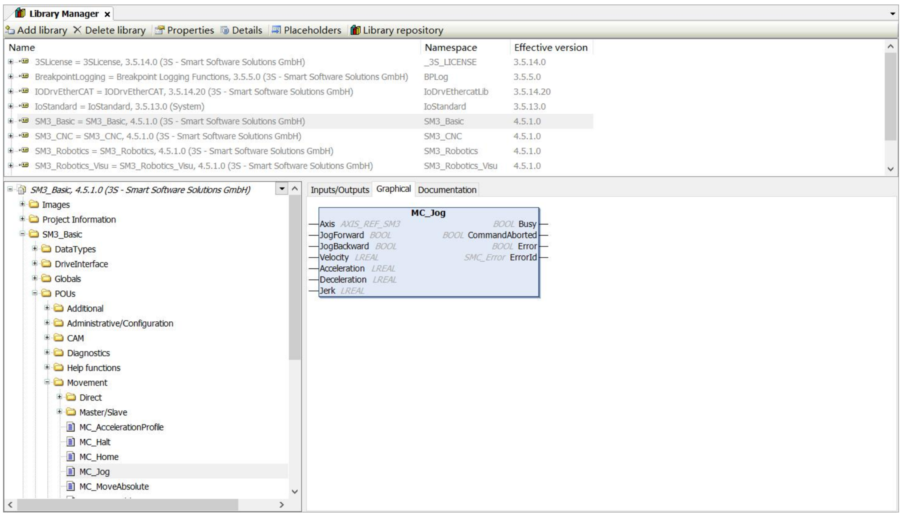
This tab for the Universal Device Editor displays status information, such as Running or Stopped, as well as specific diagnostic messages from individual devices, as well as information about the internal bus system.

Information

This tab for the Universal Device Editor displays general information from the device description file: name, vendor, category, version, order number, description, and so on.

4.2.3 Library Manager Library Manager

The library manager lists all the libraries integrated in the project to create applications. It provides information about library types, properties, and content, and can expand or collapse a list of integration libraries.



A list of all libraries integrated in the project. If one library depends on another, the referenced libraries are automatically integrated. The library manager contains three views:

Top view: Integrated library list

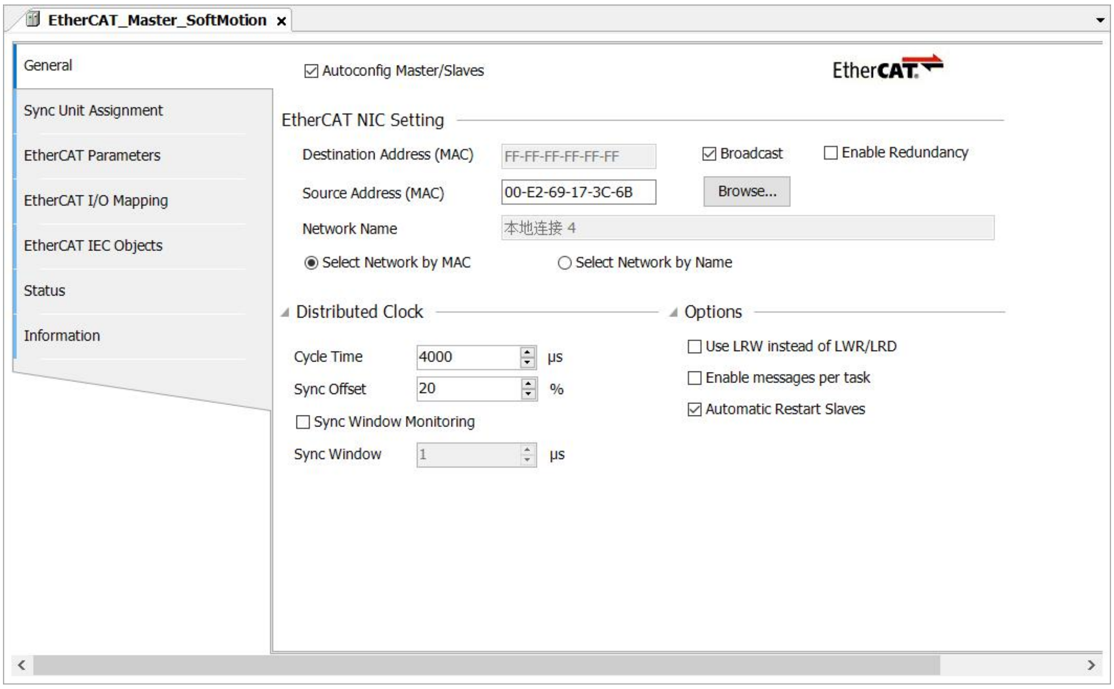
Bottom left view: Tree structure, all modules of the library are selected in the view above

Bottom right view: Documentation for the module selected in the tree

4.3 EtherCAT busses are commonly used

4.3.1 EtherCAT_Master main station

General(General).



Autoconfig Master/Slaves: Auto-configuration mode (Autoconfig Master/Slaves option) is active by default and is available for standard applications. If this mode is not activated, all configuration settings for the host and the machine must be done manually, which requires expertise. When checked, most master-from configurations are automated, depending on the device description file and implicit calculations. Check by default.

EtherCAT NIC setting(EtherCAT NIC settings).

Destinationaddress: TheMAC address of the device in the EtherCAT network to receive the telegram.

Broadcast: Broadcast without specifying a destination address(MAC). Check by default.

Enable redundancy: Activate the feature if the bus is constructed as a ring topology and redundancy is to be supported. With this feature, the EtherCAT network works even when the cable is disconnected. If this feature is activated, parameters must be defined in the Redundant EtherCAT NIC Settings area. The default does not tick.

Sourceaddress: The MAC address or network card name (i.e. PLC) of the source address controller (target system). Click onB rowse to select.

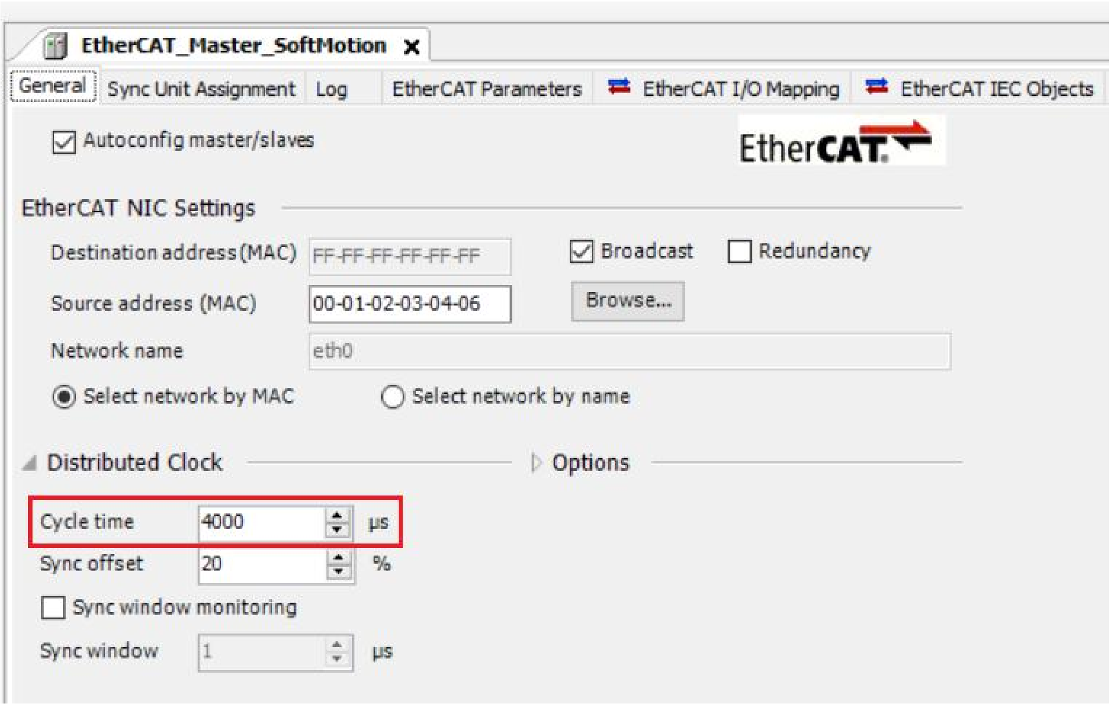
Network name: The name of the network, depending on which of the following options is activated, depending on the Source address.

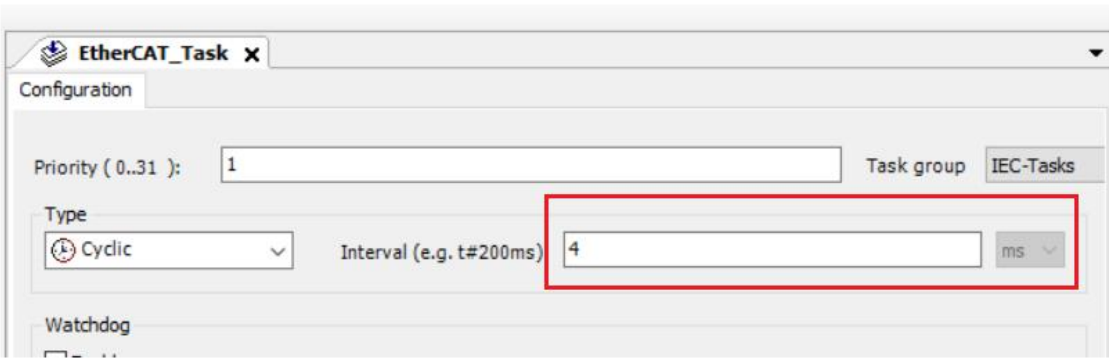
Browse: Scans the network for the MAC-ID or name of the target device that is currently available.

Distributed Clocks(Distributed Clock).

Cycle time: The interval at whichnew data messages are assigned on the bus. If the distributed clock function is activated in the from the station, the master cycle time specified here is transferred to the master clock. In this way, accurate synchronization of data exchange can be achieved, which is especially important when the process of spatial distribution requires simultaneous action. For example, simultaneous action is an application in which multiple axes must perform coordinated motion at the same time. In this way, a very precise full-network time base can be achieved, with jitters of less than 1 microsecond.

Note: Distributed clock time settings are consistent by default with EtherCAT_Task time settings, such as modifying distributed clock time or EtherCAT_Task task time.





Syncoffset: Allows the time delay of the sync interrupt from the EtherCAT station to be adjusted to the cycle time of the PLC. Typically, the PLC cycle starts 20% later than the synchronization interruption from the station. This means that the PLC loop may delay the cycle time by 80% without losing any messages.

Sync windowmonitoring: You can monitor synchronization from the slave.

Syncwindow: The time that the sync window monitors. If all synchronizations from the station are within this time window, the variable xSyncInWindow (IoDrvEthercat) is set to TRUE, otherwise it is set to FALSE.

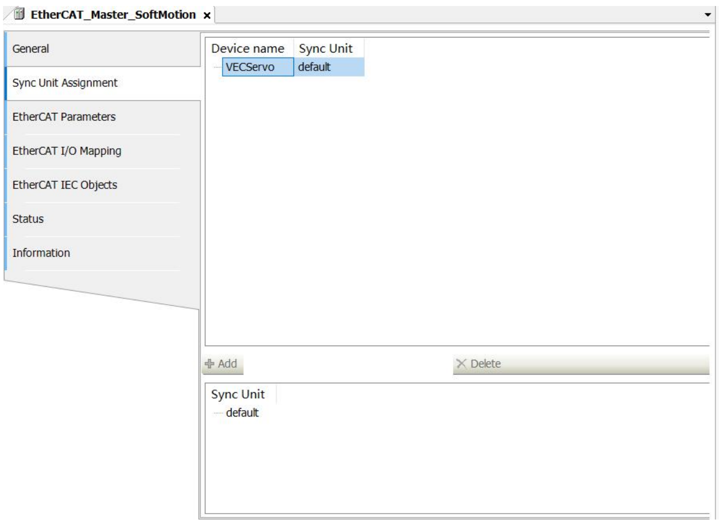
Options(option).

Use LRW instead of LWR/LRD: Direct communication from the station to the source is possible. Use a combination of read/write commands (LRW)instead ofseparate read(LRD) and write commands (LWRs).

Send/Receive per task: Read and write commands, that is, the processing of input and output messages, can be controlled by a variety of tasks.

Automatically restart slaves: If communication is interrupted, the primary station immediately attempts to restart the slave.

Sync Unit Assignment

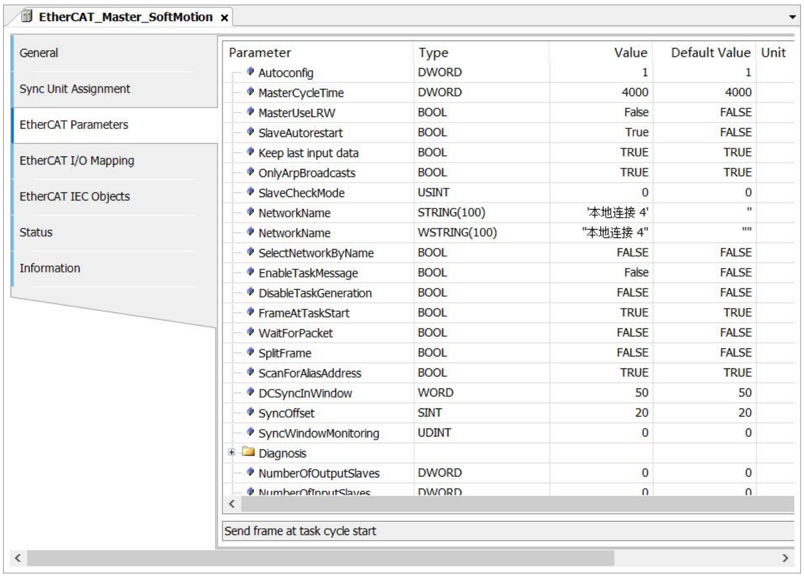


This tab displays all the stations inserted below a particular primary station and assigns a synchronization unit.

With EtherCAT synchronization units, multiple stations can be configured as groups and then subdivided into smaller units. For each group, you can monitor the work counters to improve and more accurate error detection. Once one of the stations is missing from the synchronization unit group, the other stations in the group also appear to be missing. Because the work counter is continuously checked, it is detected immediately in the next bus cycle. Device diagnostics allow you to correct missing groups as quickly as possible.

Unaffected groups continue to function without any interference.

Parameters



This tab contains the main parameters defined in the device description file.

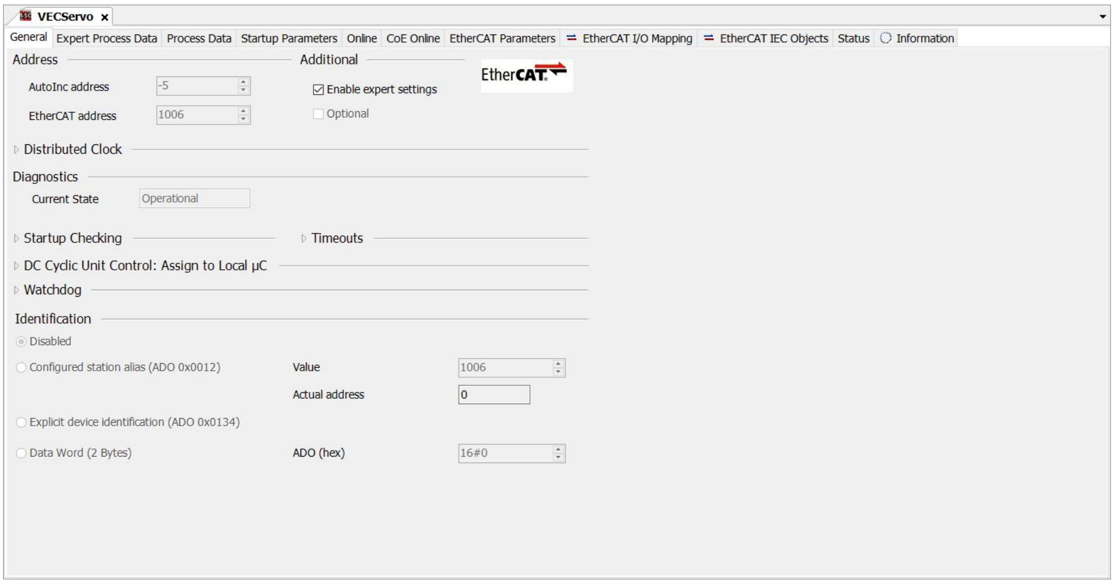
If automatic configuration mode is activated in the Main dialog box, parameters are automatically set here based on the device description file and the specifications in the network topology. Nothing should be changed in the Universal Editor because invalid configurations can be set here.

4.3.2 EtherCAT_ slaveslave from the station

Object: EtherCAT from the station

The basic settings for the EtherCAT from the station are configured in this option. The device description file is preset to basic settings.

General(General).



Address(address).

Fields can only be edited if the automatic configuration mode of the EtherCAT master is disabled.

AutoInc address: The self-added address (16 bits) is generated by the location of the from the station in the network. Addresses are used during system startup only when the primary station assigns the EtherCAT address to its base station. When the first message passes through the station for this purpose, the AutoInc address for each station adds 1.

EtherCAT address: The final address assigned to the master in the startup, the address is independent of the location from the stand in the network.

Additional

Enable Expert Settings: Expert settings. Additional settings are available when starting checks and time monitoring (see below). When checked, the Expert Process Data tab is available in the device editor, however, expert settings are not required for standard applications.

Optional: Optional. The from the station is defined as optional and does not generate an error message when a device is missing from the bus system. Note: If the from the station is defined as Optional, the from the station must have a unique identity. You can change this by changing three possible settings in the Identification section. This feature is only available if the master/master automatic configuration option is activated in the EtherCAT master and the EtherCAT from the master supports the feature.

Distributed Clocks(Distributed Clock).

Select distributed clocks:A down-to-back list of all settings for distributed clocks in the device description file.

Activate: Displayed in the synchronization unit cycle (s), the cycle time used for data exchange is determined by the cycle time of the primary station, so that the master time can synchronize the data exchange in the network.

The Sync0 and Sync1 settings described below are dependent:

Sync0

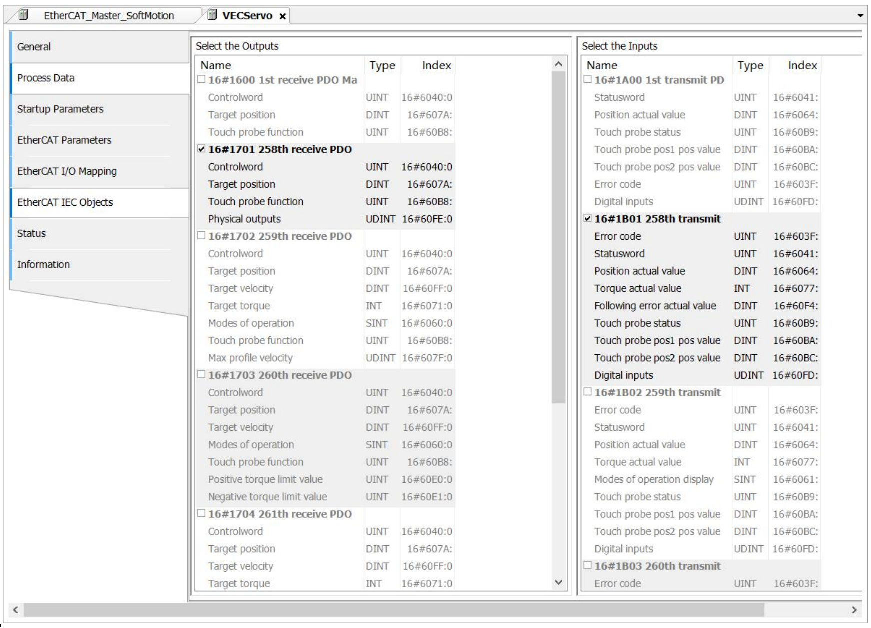
Activate Sync0	☑: Use the synchronisation unit Sync0. The synchronisation unit describes a string of synchronously exchanged process data.
Synchronised unit cycles	☑ : The master cycle time (multiplied by the factor selected from the drop-down list) is used as the slave's synchronous cycle time and the cycle time (µs) shows the currently set cycle time.
User defined	☑: User defined cycle times (in milliseconds) can be specified in the cycle time (µs) field.

Sync1

Activate Sync1	☑: Use the synchronisation unit Sync1. The synchronisation unit describes a string of synchronously exchanged process data.
Synchronised unit cycles	☑ : The master cycle time (multiplied by the factor selected from the drop-down list) is used as the slave's synchronous cycle time and the cycle time (µs) shows the currently set cycle time.
User defined	☑: User defined cycle times (in milliseconds) can be specified in the cycle time (µs) field.

Process Data（过程数据）

This EtherCAT configurator option displays the process data for the slave inputs and outputs, which are derived from the device description



file.

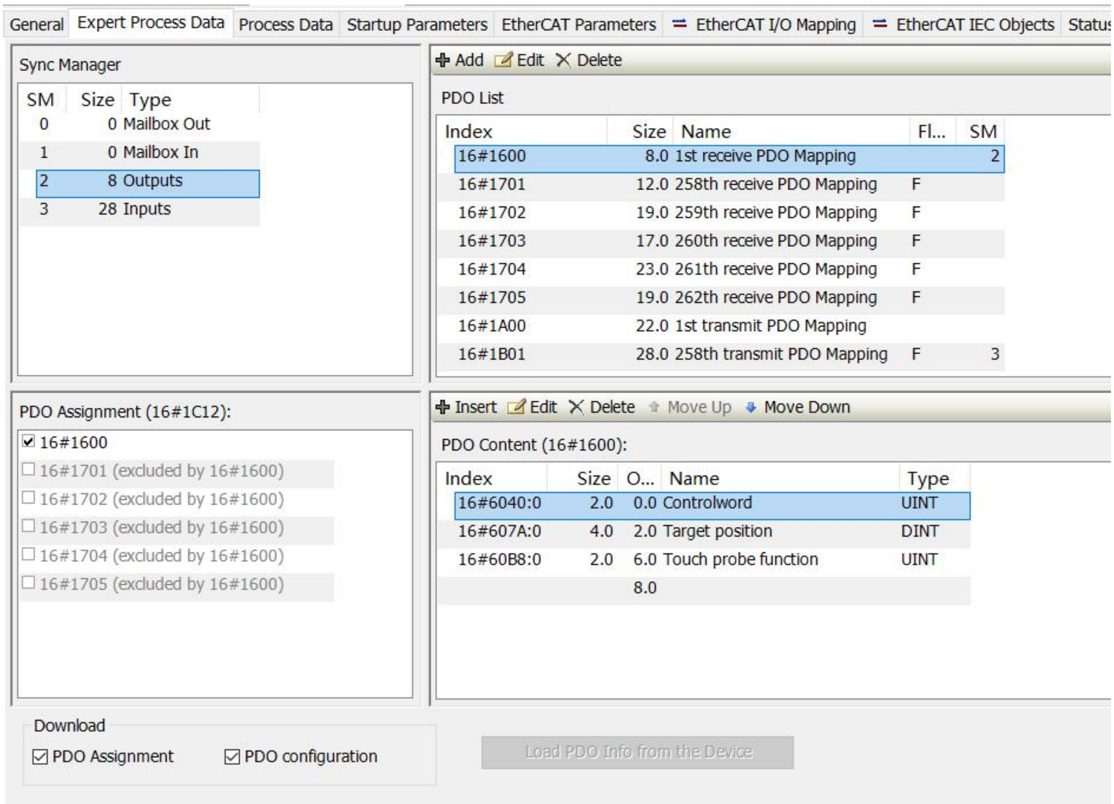
Selectoutputs: The table shows the output name, type, and index address from the station. If the device output here (for writing) is activated, these outputs can be assigned to the list of items in the EtherCAT I/O mapping dialog box.

Selectinputs: The table shows the inbound name, type,and index address from the station. If the device inputs here (for reading) are activated, these inputs can be assigned to the list of items in the EtherCAT I/O mapping dialog box.

Expert Process Data（专家过程数据）

To set this option, you need to check the box set by the expert from the station first 

, When checked, a new tab appears, as shown below, which provides a different and more detailed view of the process data.



SyncManager: A list of sync managers with data size and PDO type

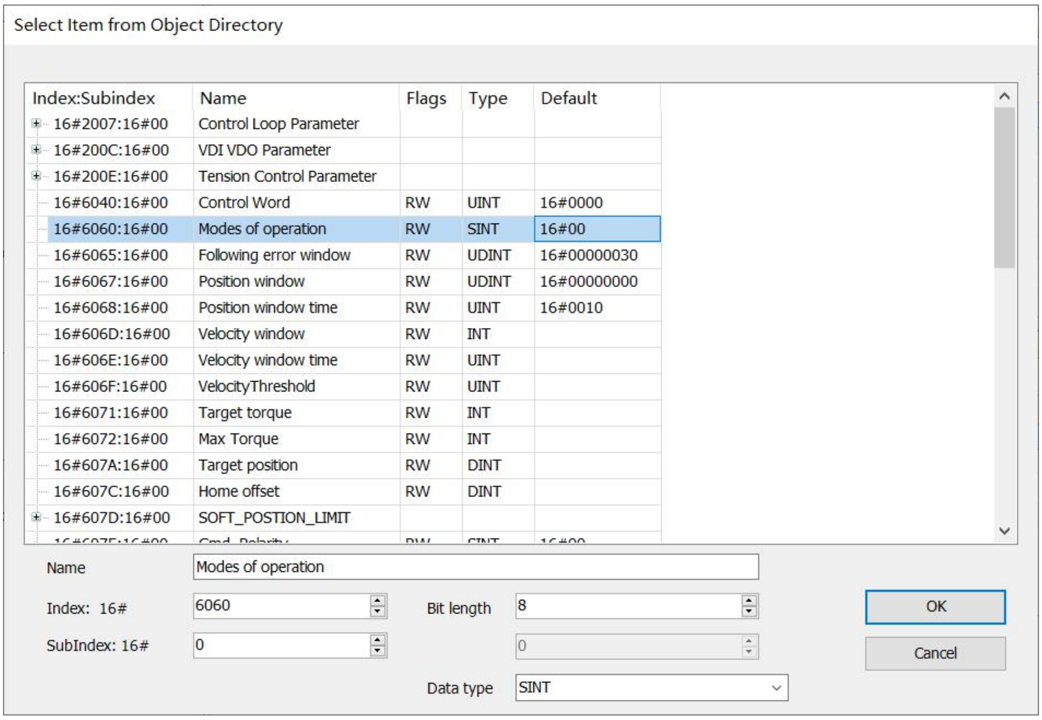
PDO Assignment: A list of PDOs assigned to the Selected Sync Manager, and if check box isselected, activate the PDO and create an I/O channel.

PDO List: The list of PDOs assigned to the Selected Sync Manager can add newPDOs or edit and delete existing PDOs by executing different commands in the command bar or shortcut menu (add, delete, edit).

PDOContent: Displays the selected PDOs content in the PDO list. You can add new entries or edit and delete existing ones by executing different commands in the command bar or shortcut menu (Insert Add, Delete, Edit Edit Edit). You can change the PDO order by clicking Move Up and Move Down To move.

Attention:

When the project requires a custom PDO, the Wykoda Bus Servo offers two sets of PDOs that can be customized by the user: 16 s 1600 and s16 s1A00. How to add: Select 16-1600 or 16-1A00 on PDO List, then click Insert To add, pop up the selection dialog box, which contains all the objects of the servo, the user can choose according to the needs of the project, and then click OK to insert, customize the addition of PDO.



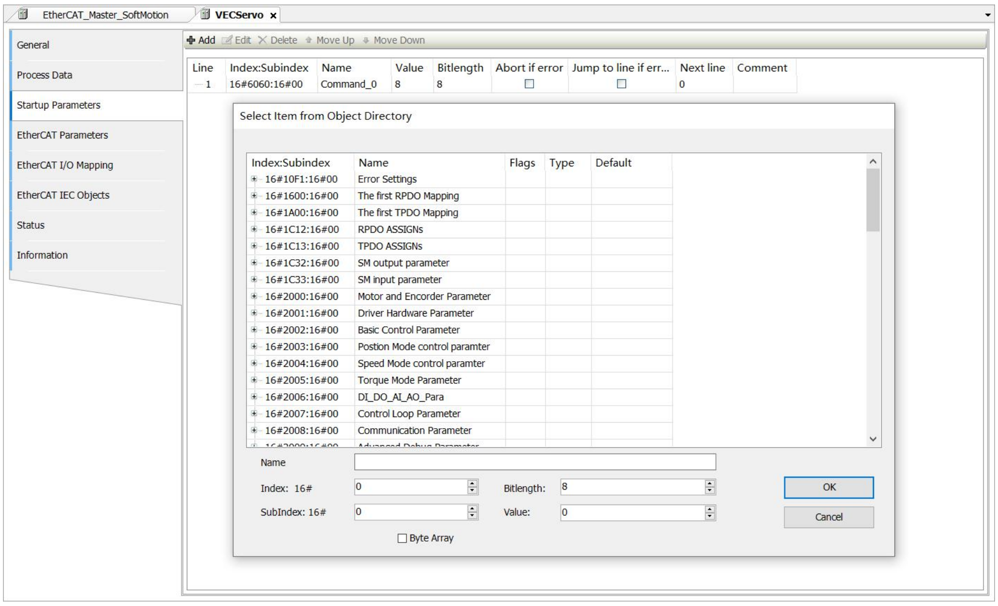
Download:

PDO Assignment	☑: Generate specific CoE commands for initialising the 0x1cxx object and write them to the slave.
PDO Configuration	☑: Generate a CoE command for 0x16xx or 0x1axx to load the PDO mapping to the slave. As a rule, the default value is taken from the ESI file and the device must support this function.
Loading PDO information from a device	Read the current PDO configuration from the slave and enter the configuration. Then delete the list in the top and bottom right corner and fill it with the read data. This is particularly effective when the ESI file is incomplete and the configuration is only available on the slave.

Startup Parameters（启动参数）

Define in this option for the current slave the transfer of the specified parameters to the SDO (Service Data Object) of the device at system start-up or as described in the EDS file referenced in the XML file.

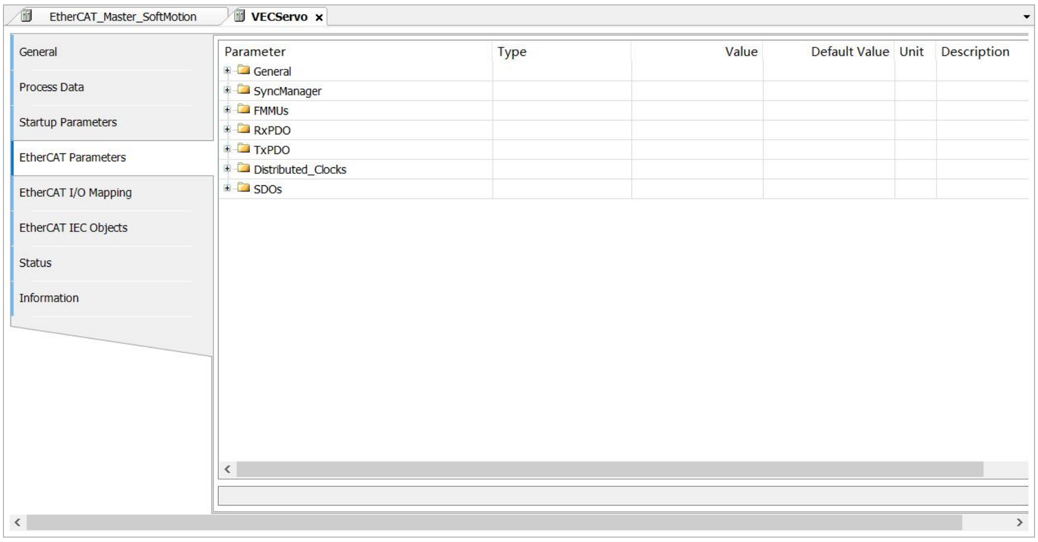
Requirements: Device supports CAN over EtherCAT or Servodrive over EtherCAT



Note: Some of the modules inserted under the slave have their own start-up parameters. These parameters are then displayed in this list, but cannot be modified. These parameters can be changed in the editor of the relevant module.

The order in the SDO table (from top to bottom) specifies the order in which the SDOs are transferred to the module.	
line	Line number
Index: Subindex	Index number and sub-index number of SDO
Bit Length	Bit length of SDO
Abort on error	☑: In the event of an error condition, the transmission is interrupted.
Jump to Line on Error	☑: To prevent errors, restore the SDO pass on the specified line.
Next Line	☑: Resume the transfer using SDO in the next line.
Comment	Input fields for comments
Move Up	Move the selected row up one line
Move Down	Move the selected row down one line
Add	Opens a dialog box to select an entry from the object catalogue where the SDO parameters can be changed before the SDO is added to the configuration. By specifying a new index/sub-index entry, new objects can be added to the SDO that are not already described in the EDS file.
Delete	Removes the selected entry.
Change	Opens a dialog box to select an entry from the object catalogue for the selected SDOs parameter in the table.

EtherCAT Parameters（EtherCAT 参数）

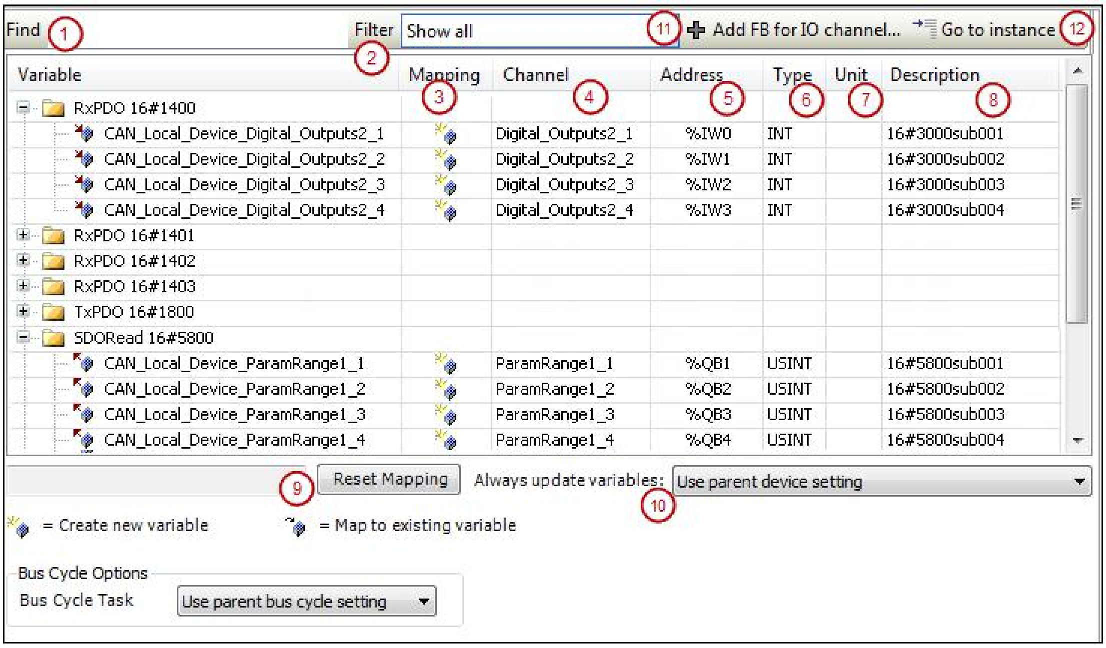


This option contains the frombound parameters defined in the device description file.

If the automatic configuration mode of the primary station is activated, these parameters are automatically set here according to the specifications of the device description file and the network topology. Standard applications generally do not need to be modified.

EtherCAT I/O Mapping(EtherCAT Input and Output Map).

The inputs and outputs from the station selected in the Process Data option are listed here, which shows the available channels and allows the controller's input, output, and memory addresses to be mapped to the variables of the application or the entire function block. In this way, you can create so-called "I/O mappings."



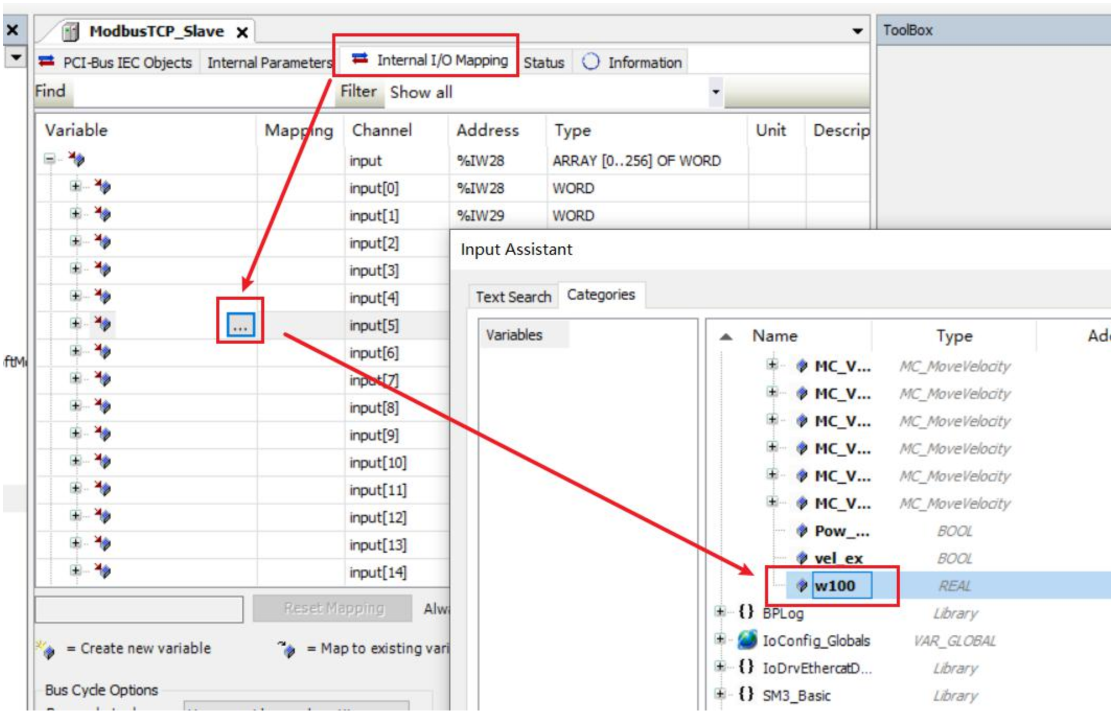
(1) Find	The mapping table searches for strings in the input fields.
(2) Finter	Drop-down list for filtering I / O mappings listed in the mapping table. Show all Show outputs only Show inputs only Show only unmapped variables Show only mapped variables Show only mapped to existing variables Show only mapped to new variables
(11) Add FB for I/O Mapping	Depending on the device, a channel entry is available if it is selected in the mapping table. Open the "Select Function Block" dialog to select the function block that should be linked directly to the channel.
(12) go to Instance	Available if the entry is selected in the mapping table. Jump to the corresponding entry in the <Device Name> IEC Objects tab.
Variable	Depending on the device, the inputs and outputs of the device are displayed as nodes with indented associated channels below the nodes or, depending on the device, only implicitly created instances of the device are displayed. The symbols indicate the type of channel. : input : output Double-clicking on a cell will open an input field. Option 1: The variable already exists; specify the full path: <application name>. <module name>. <variable name>; e.g.: app1.plc_prg.ivar; by typing help 。 Option 2: The variable does not yet exist; enter a simple name; it is automatically created internally as a global variable. Depending on the device, the input or output can be linked directly to a function block. In this case, the activation of the " Add FB" button for the I / O channel.
(3) Mapping	Map type: : The variable already exists : The new variable : Map to a feature block instance
(4) channel	The symbolic name of the channel.
(5)	The channel address, e.g. %IW0.

Address	Address strikethrough: indicates that you should not assign any other variables to this address. Reason: Although the variables specified here (as already existing variables) are managed in different storage locations, ambiguity may arise during the writing of values, especially for output. M： Indicates that this address has been edited and repaired. CODESYS does not automatically adjust this address if the arrangement of device objects in the device tree is changed.
(6) Type	The data type of the channel, e.g. BOOL. Structures or bit fields defined in the device description are only displayed if they belong to the IEC standard and are identified in the device description as an IEC data type. Otherwise, the table cells remain empty. When mapping structure variables, the editor prevents the simultaneous entry of structure variables (e.g.) %QB0 and individual structure elements (e.g. %QB0.1 and QB0.2). Therefore, if a master entry with a subtree of bit channel entries exists in the mapping table, the following condition applies: a variable can be entered in the row of the master entry or in the row of the child element (bit channel) , but not both.
Default Value	Default values for the parameters applicable to the channels: only displayed if the option "Set all output to defalt" is activated in "PLC Setting" for the output behaviour at stop.
(7) Unit	The unit of the parameter value, e.g. ms milliseconds.
(8) Description	A brief description of the parameters.
Current Value	The actual value of the parameter applied to the channel; displayed in online mode only.

(9) Reset Mapping	CODESYS resets the mapping settings to the default values defined in the device description file.
(10) Update Variables	Definition of the device object regarding the update of the I / O variable. The default value is defined in the device description. Use parent device setting Use parent device setting: Updates according to the settings of the parent device. Enable 1 (use it if not in any task): CODESYS updates the I / O variables in the bus cycle tasks if they are not used in any other task. Enable 2 (always in the bus cycle task): CODESYS updates all variables in each cycle of the bus cycle task, regardless of whether they are used and whether they are mapped to an input or output channel.

There are two ways to associate I/O mappings to a program.

(1) Selecting variables in the mapping

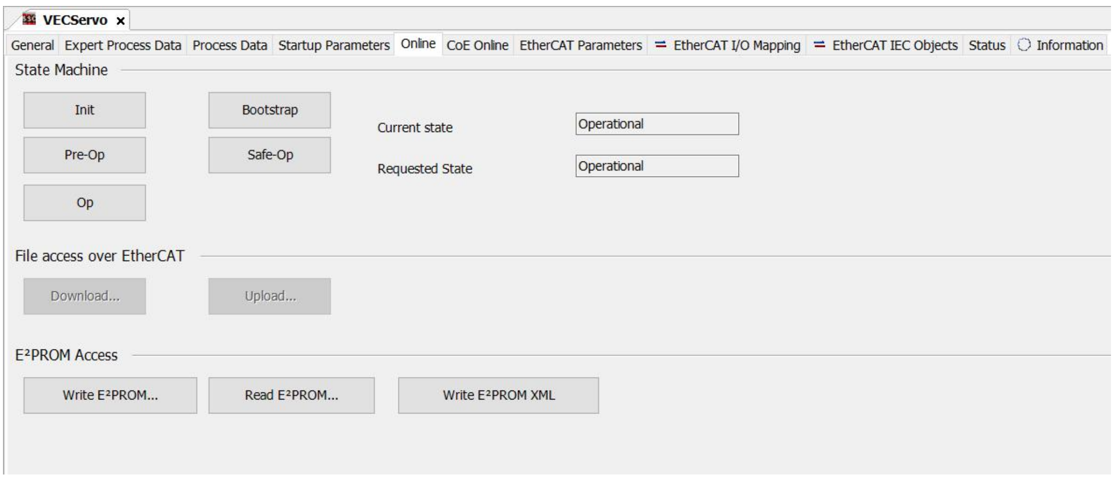


(2) Address assignment in the program

```
PROGRAM PLC_PRG
VAR
    w100 AT %IW5: REAL;
END_VAR
```

Online（在线）

Once you have logged in to the device online, the Online tab appears. With EtherCAT, you can use the slave status information and the functions for transferring files to the slave.



State Machine：状态机制

Init	Initialisation for debugging purposes
Boot Strap	The slave switches to Bootstrap mode. Required if firmware files are to be transferred with the slave device
Pre-Op	Pre-operation mode for commissioning purposes
Safe-Op	Safe boot mode for debugging purposes
Op	for debugging purposes
Current status	The current state
Requested status	Request Status

File access over EtherCAT： File access via EtherCAT

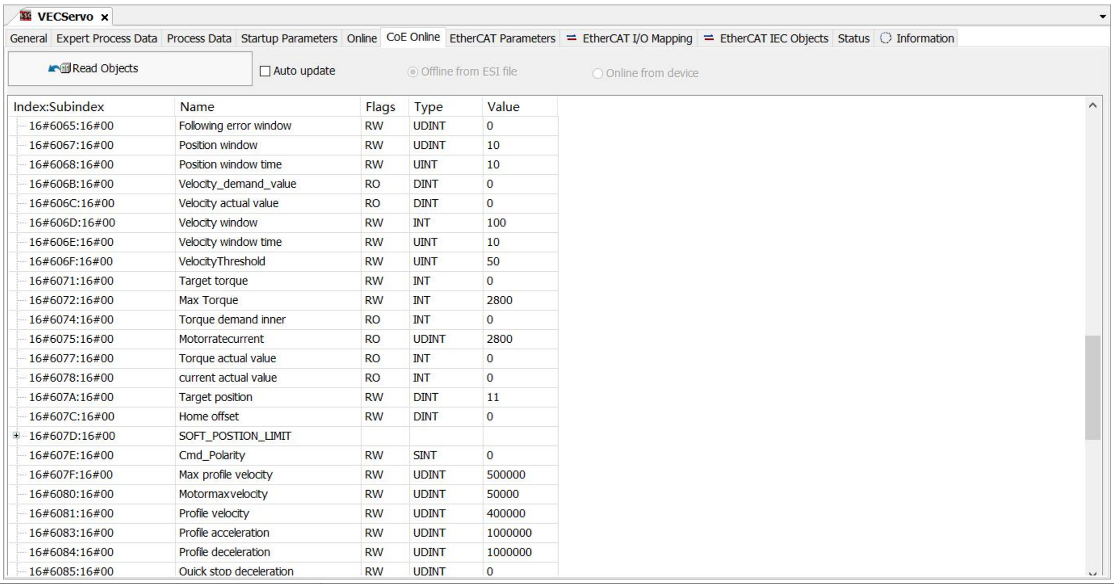
Download	Downloading the firmware file A dialog box appears for storing the firmware file. In this dialog box, a string and a password must be entered in order to perform the file transfer. This information will be taken from the data sheet of the slave station.
Upload	Uploading a firmware file A dialog box appears for opening the firmware file. In this dialog box, a string and a password must be entered in order to perform the file transfer. This information will be taken from the data sheet of the slave station.

E²PROM

Write E²PROM	Write the slave's configuration to the E²PROM.
Read E²PROM	Reads the configuration of the slave from the E²PROM. Uploading the firmware file
Write E²PROM XML	Writes the slave configuration directly from the XML file to the device. Only executed if configuration data (<ConfigData> section) is present in the XML file.

CoE Online（CoE 在线）

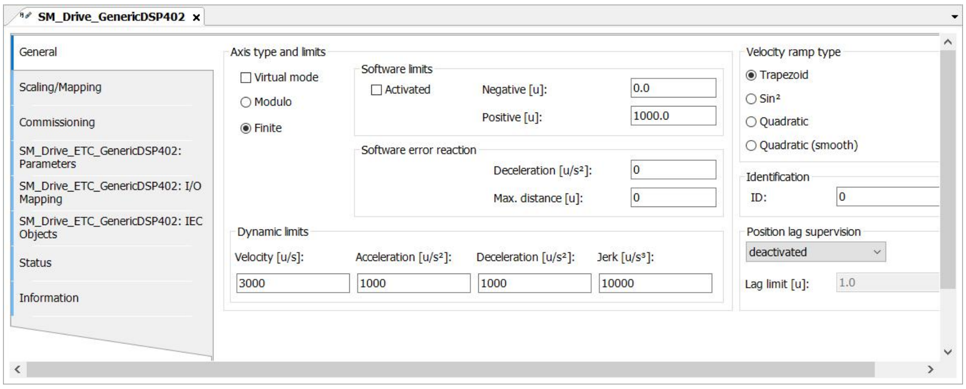
To set this option, you need the slave to support CoE Online mode, first check the checkbox for expert settings in the slave ☑ Enable expert settings， And after logging into the device online, a new tab CoE Online will appear, as shown below, this option displays the object index of the ESI or slave.



Read Objects	The object index is read once
Auto Update	Objects are read periodically and updated automatically
Offline from ESI File	This dialog box displays the contents of the object index in the device description
Online from Device	SDOInfo, which displays the object index in the device, must be enabled in the ESI file
Flags	RO:value is write-protected RW:value can be changed
Type	Data type of the parameter
Value	Values can be edited by double clicking into the text area

4.3.3 SM_Drive_GenericDSP402 Shaft configurations

General（通用）



Axis type and limits(axis type and limit).

Virtualmode: Virtual mode. The drive is replaced by a simulation similar to a virtual drive unit. If there are coupled drives, there is no effect on the field bus devices, which operate as usual without sending messages to or receiving messages from the physical device.

Note: You can also SMC3_ReinitDrive the virtual mode of the drive with IEC code by using the function block.

Modulo: Module. The drive rotates indefinitely without limiting the range of operation, such as a belt drive.

The value of the module: the value of a period (mold period). The value is saved in the fPositionPeriod AXIS_REF_SM3 the function block and the function block.

Note: If you select the Modelo drive type, the product must be an integer.

fPositionPeriod * dwRatioTechUnitsDenom

Finite: Limited. The drive has a fixed working area, such as a linear drive.

ActivatedThe software limit switch is activated : Position values are limited by lower negative values and upper-limit positive values.

Negative Reverse: The input field for the negative limit value

Positive Positive: The input field for the positive limit value

Software error reaction(software error response).

Deceleration (u/s2): Thedeceleration value when the limit switch is reached.

Max Distance: Optional,the drive must reach a stop state within this distance after an error hasoccurred.

Dynamic limits(dynamic limits).

Velocity (u/s):The limit of speed

Acceleration (u/s2):The limit of acceleration

Deceleration (u/s2) : The limitof thedeceleration

Jerk (u/s3):The limit value of the acceleration change rate

Velocity ramp type(speed ramp type).

Trapezoid: Keystone velocity curve (with constant acceleration in each segment).

Sin2: A velocity curve defined by the sin2 function (with a constant acceleration curve).

Quadratic: A trapezoidal acceleration curve with acceleration limits

Quadratic (smooth): Similar to Quadratic, but the resulting beating curve does not jump

Identification(ID card).

ID: Integer identifier. Each drive should be unique. For example, this identifier is used in the PLC log to identify the drive in the event of an error.

Lag supervision(lagging regulation).

Deactivated: Deactivated: Deactivated. No response or traction error monitoring is disabled.

DisableDrive: Disable the drive. The bRegulatorOn bit is forced to be set to FALSE (MC_Power input), which first forces the drive to slow down and then deactivation the drive (depending on the drive implementation).

Do quickstop: Stop quickly. The bDriveStart bit is forced to be set to FALSE (compared to MC_Power input), which forces the drive to perform a quick stop.

Stayenabled: Remain enabled. The drive remains open, but all running actions suddenly stop.

Laglimit: Lag limit. Drag the error monitoring in the controller.

Scaling/Mapping（缩放/映射）

Motor Type

Rotary: Rotating. The settings in the scale apply to rotating motors.

Linear: Linear. The settings in the scale are suitable for linear motors. (simplified configuration without gears and motor turns).

Scaling

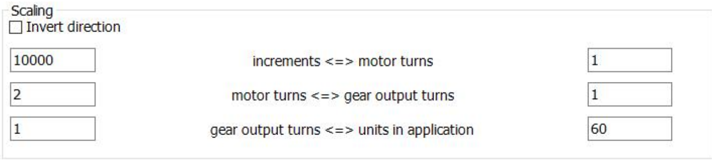
Invertdirection: Reverse the direction of the motor. The motor obtains a specified value with the opposite symbol.

increments and motor turns: pulseincrements and number of motor turns.

Motor turns and gear output turns: number ofmotor turns and number of given gear output laps.

gear outputs and It; units in application: numberof gear output turns and applied units.

Cases:



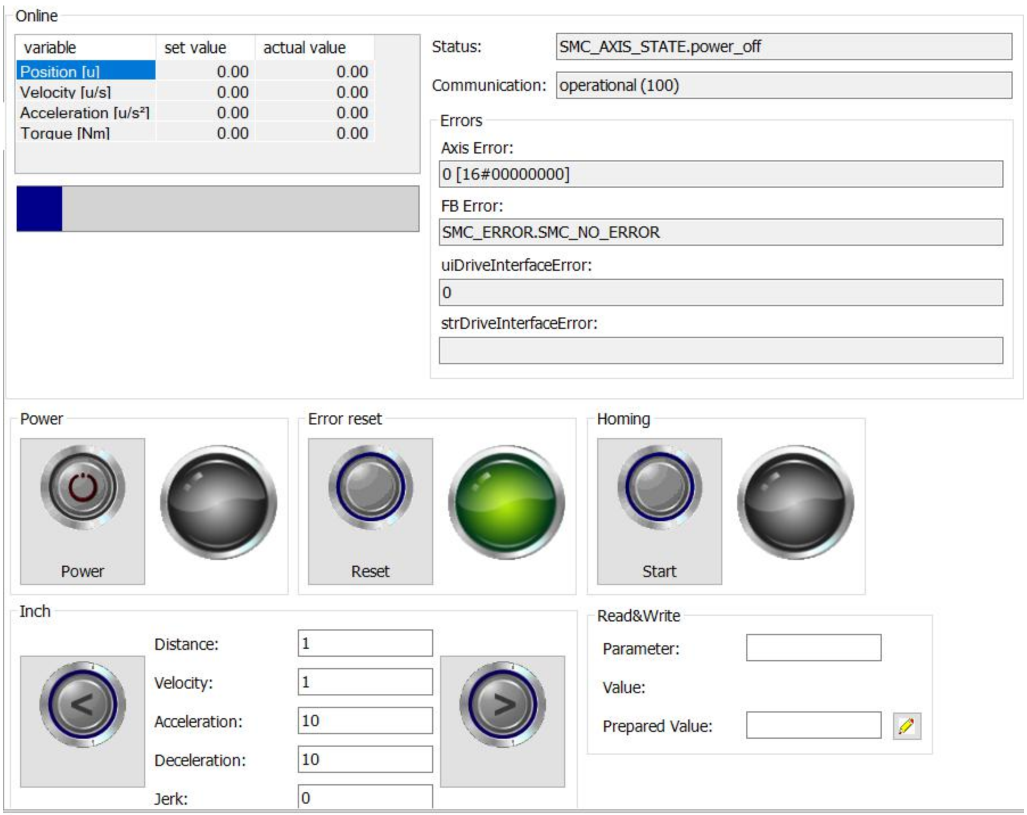
As shown in the figure, the pulse increment corresponding to the motor turn 1 turn is 10000, the decelerationratio is 2:1,and the gear output is 1 revolution correspondingto the terminal traveling 60 units.

Mapping

Automatic Mapping: Automatic mapping. Checked by default, the IEC parameters that affect the drive are automatically mapped to the appropriate inputs and outputs of the device. After you deactivate this option, you can edit the map manually.

Commissioning（调试）

This tab is used for testing purposes when commissioning physical drives. It is only available when the "online configuration mode  " is activated. In this mode, the development system is connected to the device. However, the application does not have to be downloaded.



Online

Requirement: PLC in online mode	
Variable table	List of drive variables with variable name, set value and current value
Status	Shows the current status of the SoftMotion drive
Communication	Display of the current communication status
Error	Axis error: FB error: uiDriveInterfaceError: strDriveInterfaceError:

Operating elements

Power: Driver enable (compare MC_Power)

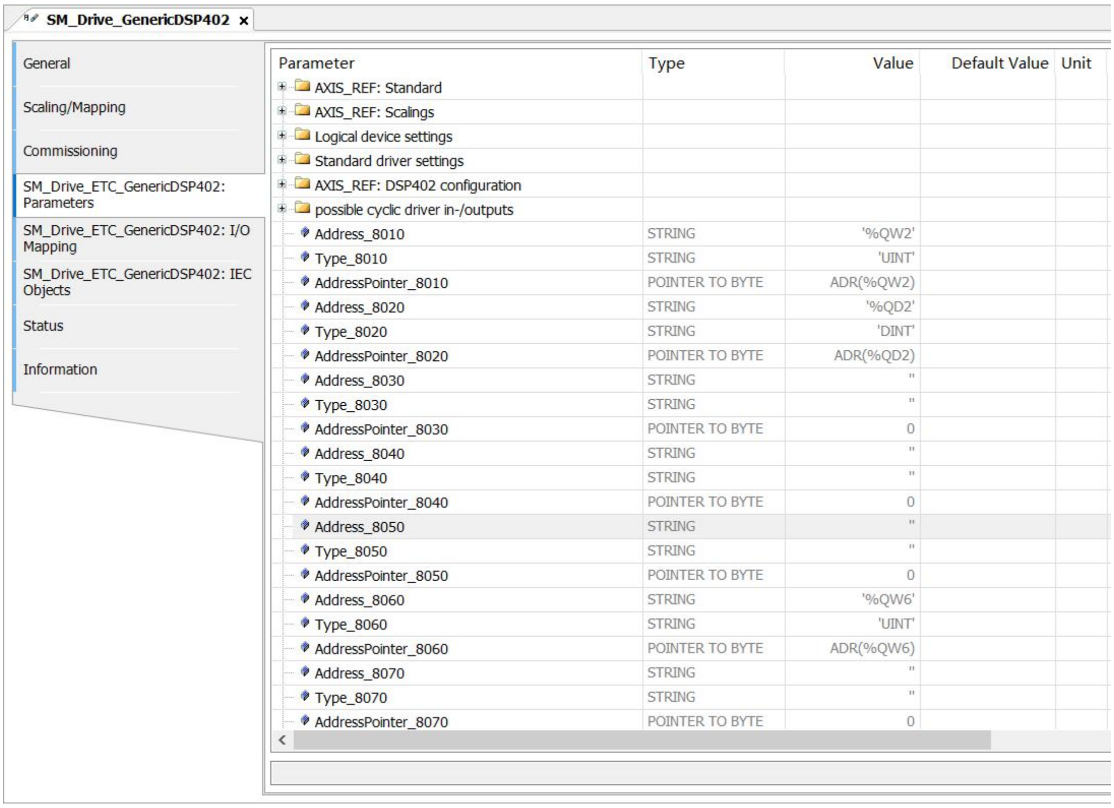
Error reset: Reset the drive after an error (compare MC_Reset)

Starthoming: Use the settings parameters in the drive to perform origin regression (MC_Home).

Joggingmode: The driver can move forward and back according to the specified values of Distance, Velocity, Speed, Deceleration, and Jerk (Compare MC_Inch).

ReadWrite: For the specified drive parameters, the current value is read and displayed from the PLC. In Prepare Values, you can specify a new value and write parameters to the drive MC_ReadParameteraMC_WriteParameter button (to the ratio ofMC_WriteParameter, the ratio).

Parameters（参数）



I/O Mapping（I/O 映射）

Bus Cycle Task: The definition of the device object that updates the bus. The default value is defined in the device description:

Use Parent bus cycle setting (using parent device settings): Update based on the settings of the parent device.

EtherCAT Task: Update the device object of the bus using EtherCAT Task.

Main Task: Update the device object of the bus with Main Task.

IEC Objects（IEC 对象）

In this tab of the Universal Device Editor, Objects are listed that allow access to devices from IEC applications. In online mode, it is used as a monitoring view.

Status（状态）

In online mode, the axis status is displayed.

Information（信息）

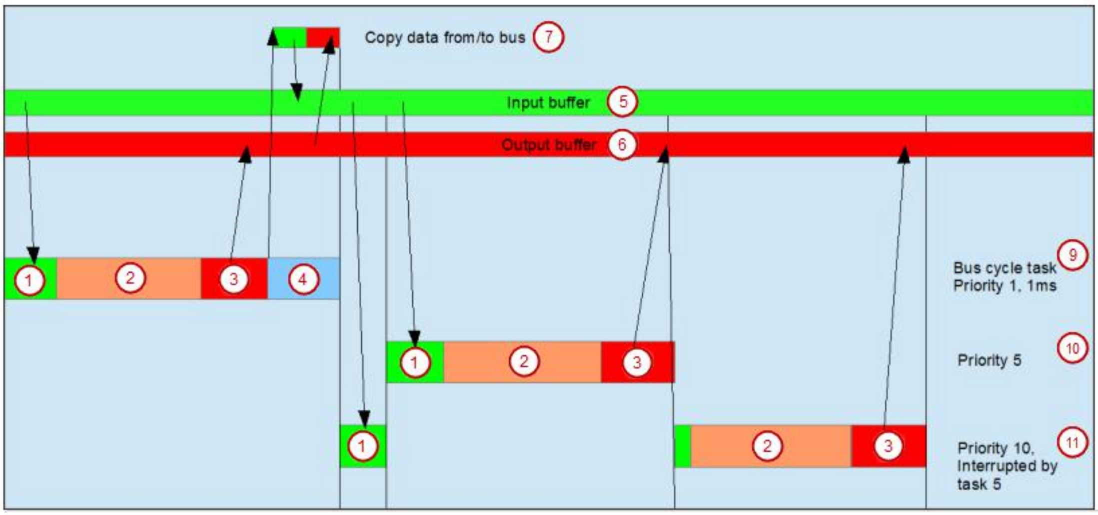
Displays axis information.

4.3.4 EtherCAT bus cycle behavior

Typically, for each IEC task, the input data used is read at the beginning of each task (1) and the output data is transferred to the I/O drive (3) at the end of the task. The implementation in the I/O driver is decisive for the further transfer of I/O data.

The PLC's bus cycle task can be defined for all field busses in the PLC settings. However, for some field busses, you can change this setting independently of the global settings. A task with the shortest cycle time is used as a bus loop task (not specified in the PLC settings). In this task, messages are usually transmitted on the bus.

Other tasks replicate only I/O data in the internal buffer, which is exchanged only with the physical hardware in the bus loop task.



(1) Read input (2) IEC tasks from the input cache

(3) Write output to the output cache (4) bus cycle

(5) Enter the cache (6) output cache

(7) Copy data to/from bus (9) bus cycle tasks, priority 1, 1 ms

(10) Bus cycle task, priority 5

(11) Bus cycle task, priority 10, interrupted by task 5

4.3.5 Ether CAT specific variables

If the primary device is plugged into the device tree, EtherCAT_Master task is inserted into the task configuration that is currently applied. As usual, POUcalls can be added to the task configuration. EtherCAT specific Boolean variables can be set in the POU to affect the EtherCAT configuration behavior in the context of the application:

1) Optional devices are supported

The loss of the EtherCAT device in the application causes an error in starting the bus to prevent the stack from loading, and the variable is set at the beginning of the first PLC cycle

```
<instance name of EtherCAT master>. StartConfigWithLessDevice := TRUE;
```

the lost device is treated as an optional device that does not affect the normal stack startup process.

2) Suppress additional message scheduling

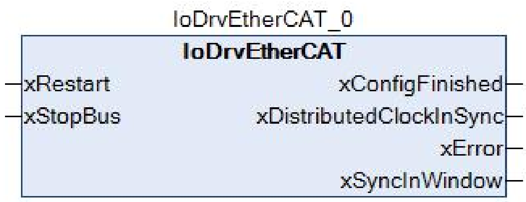
To refresh the output as quickly aspossible, the EtherCAT master sends its own messages to each individual task. However, if the satellite driver is synchronized with the real-time output data, the bus loop task should be the only one that allows the output to be set. Additional messages disrupt synchronization. To suppress additional task messages

```
<instance name of EtherCAT master>. EnableTaskOutputMessage := FALSE;
```

Must be set once in the first PLC cycle.

4.3.6 EtherCAT Library

The primary instance



Create an instance of type IoDrvEtherCAT for each EtherCAT master plugged into the device tree. The name of the instance corresponds to the name of the primary server in the device tree. The availability of the instance appears in the IEC Objects tab of the device editor.

Input:

Name	Data type	Description
xRestart	BOOL	Restart the main server along the rise and reload all configuration parameters.

xStopBus	BOOL	TRUE: Communication is stopped. No further EtherCAT telegrams will be sent. Afterwards, most devices need to be restarted as they have switched to the error state.

Output

Name	Data type	Description
xConfigFinished	BOOL	TRUE: Transfer of all configuration parameters is completed without errors, communication is running on the bus.
xDistributed ClockInSync	BOOL	If a distributed clock is used, the PLC is synchronised with the first EtherCAT slave that activates the DC option. As soon as the synchronisation has been successfully completed, the output changes to TRUE. this signal can then be used, for example, to activate the SoftMotion function block only when the PLC is in synchronisation mode, otherwise a position jump may occur. When starting the PLC the output is FALSE and changes to TRUE after a few seconds. if the synchronisation is lost due to some error the output is reset to FALSE.
xError	BOOL	The output becomes TRUE if ● An error occurred during the start of the EtherCAT stack. • Communication with the from the station is interrupted because no other messages can be received (for example, due to a brokencable).
xSyncInWindow	BOOL	If the Sync Window Monitoring option is activated and all synchronization from the station is within Sync Window, the output becomes TRUE. If the Sync Window Monitoring option is activated and all synchronization from the station is within Sync Window, the output becomes TRUE.

For example.

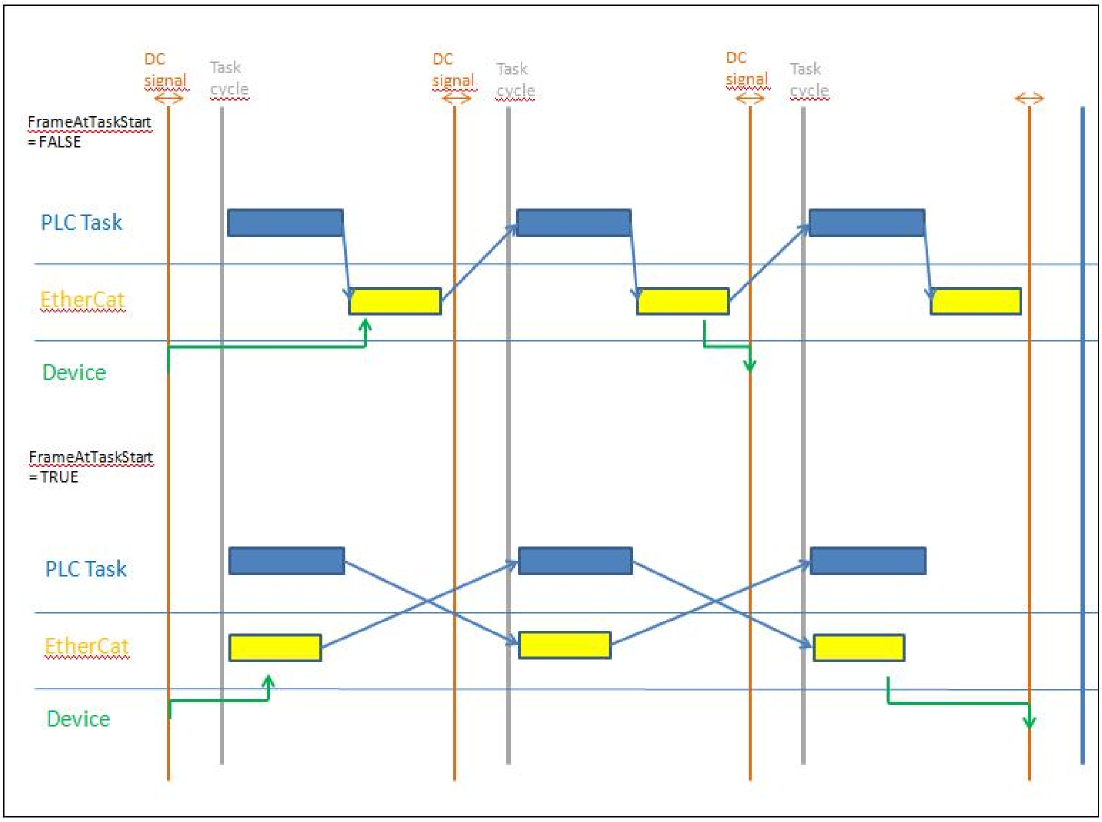
- Start the restart of the primary server with the xRestart variable:
```
EtherCAT_Master();
EtherCAT_Master.xRestart := xRestart;
```
- Stop communication on the bus via the xStop variable:
```
EtherCAT_Master.xStopBus := xStop;
```
- Call the main station for information about the success of downloading configuration parameters:
```
EtherCAT_Master();
xFinish := EtherCAT_Master.xConfigFinished;
```

Properties of the main site:

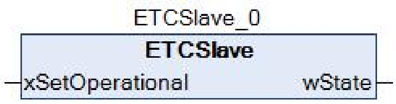
AutoSetOperational	TRUE: If communication is interrupted, the master will try to restart the slave immediately

	Default: FALSE
ConfigRead	TRUE: The reading configuration is complete and the user can edit the settings, for example in order to add a custom SDO.
DCInSyncWindow	The time window for XDistributedClockInSync. The jitter must be within the window in order for the XDistributedClockInSync output signal to remain TRUE at all times. Default:50 ms (200 microseconds when using CODESYS SoftMotion)
DCClockReferenceTime	Use the "distributed clock" to return the current time of the first slave. This time is the reference time for all other slaves and the PLC itself.
DCIntegralDivider	Integration Factor for Distributed Clock Control Loops Default:20
DCPropFactor	Scale factor for distributed clock control loops Default:25
DCSyncToMaster	Distributed clock synchronisation on the master. If set to TRUE, all slaves are synchronised to the master rather than to the first slave of the PLC. Default:FALSE
DCSyncToMasterWith SysTime	The distributed clock on the master is synchronised. TRUE:All slaves are synchronized with the system time of the master and the time read from SysTimeRtcHighResGet is used to distribute the system time of the PLC to all EtherCAT slaves. Default:FALSE
EnableTaskOutput Message	EtherCAT signals are normally scheduled by the bus cycle task and additionally from each task that uses the slave outputs. In the bus cycle tasks, all outputs are written and all inputs are read. In other tasks, the outputs are transferred again so that they can be written to the corresponding slave immediately. In this way an attempt is made to keep the stagnation time until the shortest possible write. This together with distributed clocks can cause problems in some devices, for example if the servo controller is not synchronised with the Sync interrupt but uses the write time for internal synchronisation. In this case, multiple write accesses may occur in one cycle. If EnableTaskOutputMessage is set to FALSE, only bus cycle tasks are used. Further tasks will not result in further messages. Default: TRUE
FirstSlave	Pointer to the first slave below the master
FrameAtTaskStart	TRUE: The frame of the slave is transmitted at the beginning of the task (before the IEC task), which ensures the minimum jitter. This command is used to realize the non-impact movement of

	the servo drive. If this flag is set to TRUE, the frame of the output buffer will be written in the next cycle (see the chart below). Default: FALSE (TRUE when using CODESYS SoftMotion)
LastInstance	Pointer to the list of connected master stations -> previous master station.
LastMessage	This property returns a string that contains the last message from the EtherCAT stack. If the startup completes successfully, return to All slavesdone. Use the same string as the diagnostic message displayed in the EtherCAT main device editor window in online mode.
NextInstance	Pointer to the list of connected master stations -> next master station.
NumberActiveSlaves	This attribute returns the number of slaves actually connected. StartConfigWithLessDevice :=TRUE: The number of the physical device detected.
OpenTimeout	Turning on the network adapter timed out. The default value is 4 seconds.
SplitFrame	Used to divide the frame into two parts. The first part contains process data, and the second part contains asynchronous mailbox communication and status flags. Due to the separation, the process data is received earlier, so the jitter of the PLC has less impact on the slave device.
StartConfigWithLess Device	Used to affect the startup behavior of the stack. For example, if five servo controllers are configured in the project but only three are connected, the EtherCAT stack will usually stop. However, if in the first cycle, StartConfigWithLessDevice := TRUE, then the stack will still try to start. In this way, for example, a general configuration of 10 servo controllers can be implemented, but the number of actual connections can be kept variable. Please note that the vendor ID and product ID of each slave must be checked anyway. If a difference is found, the stack stops.



The from the station instance



An instance of the ETCSlave data type is generated for each EtherCAT slave inserted into the device tree. The name of the instance corresponds to the name of the subordinate in the device tree, and the availability of the instance is displayed on the IEC object tab in the device editor.

The slave instance is used in the application to query or change the state of the slave at runtime.

Input

Name	Data type	Description
xSetOperational	BOOL	Rising edge: Try to switch to ETC_SLAVE_ OPERATIONAL mode.

Output

Name	Data type	Description
wState	ETC_SLAVE_STATE	The current status of the slave: • 0:ETC_SLAVE_BOOT • 1:ETC_SLAVE_Init

		• 2:ETC_SLAVE_PREOPERATIONAL • 4:ETC_SLAVE_SAVEOPERATIONAL • 8:ETC_SLAVE_OPERATIONAL The configuration has been successfully completed. If an error occurs during configuration, the slave can fall back to an earlier state.

Properties of the slave:

VendorID	After the EtherCAT stack is started, this attribute will return the vendor ID read from the device
ConfigVendorID	Read the vendor ID from the configuration
ProductID	After the EtherCAT stack is started, this attribute will return the product ID read from the device
ConfigProductID	Read product ID from configuration
SerialID	After the EtherCAT stack is started, this attribute contains the serial number of the device.
LastEmergency	If a message is received, this information is stored in the slave server and can be queried from the application using this property, and a log message is also added.

Note: If you activate the vendor or product ID check in the expert settings, as long as there is a difference between VendorID and ConfigVendorID or ProductID and ConfigProductID, stop the stack startup.

Check the chained list of all slaves

In order to monitor each slave station in the program, the instance will be called and the state wState will be determined through. For the sake of simplicity, all masters and slaves can be determined through the linked list, and all slaves can be checked through a simple WHILE loop. The properties NextInstance and LastInstance exist for both the master and the slave. These attributes point to the next or previous subordinate. For the master server, there is an additional attribute FirstSlave, which provides a pointer to the first slave server. According to the following example, all slaves can be checked.

Example:

```
statement:
pSlave:POINTER TO ETCSlave;

program:
pSlave := EtherCAT_Master.FirstSlave;
WHILE pSlave <> 0 DO
pSlave^();
IF pSlave^.wState = ETC_SLAVE_STATE.ETC_SLAVE_OPERATIONAL THEN
;
END_IF
```

```
pSlave := pSlave^.NextInstance;
END_WHILE
```

At the beginning, the first slave is extracted to the master through EtherCAT_Master.FirstSlave. In the WHILE cycle, each master station is called individually, and wState is also specified, and then the status can be checked. The pointer to the next slave station is extracted by pSlave^.NextInstance. If the list is complete, the pointer is zero and the loop ends.

4.3.7 IODrvEtherCAT

If EtherCAT configuration is supported and executed, the library is automatically integrated into the project. It contains function blocks for reading and writing device parameters. It is therefore possible to check and even change individual parameters at runtime. Several functional blocks can be active at the same time. Each request in the loop is managed internally and processed continuously.

ETC_CO_SdoRead

Library: IODrvEtherCAT

This function block is used to read EtherCAT slave parameters. Unlike ETC_CO_SdoRead4, it also supports parameters longer than 4 bytes. The parameters to be read are specified by Index and Subindex, as used in the object catalog.

Input

Name	Data type	Description
xExecute	BOOL	Rising edge: Start to read the slave parameters. In order to release the internal channel again later, the instance must be called at least once by xExecute: = FALSE.
xAbort	BOOL	TRUE: The current reading process is aborted.
usiCom	USINT	EtherCAT master station number: If only one EtherCAT master station is used, usiCom is always 1. If multiple master stations are used, 1 specifies the first one, 2 specifies the second, and so on.
uiDevice	UInt	The physical address of the slave. If the automatic configuration mode is deactivated in the master station, the slave station can be provided with its own address. This address must be specified here. If the automatic configuration mode is activated, the address of the first slave is 1001. The current slave address can be checked in the Slave dialog box in the EtherCAT address area device editor.
usiChannel	USINT	Reserved for future expansion
wIndex	WORD	The index of the parameter in the object directory.
bySubindex	BYTE	The sub index of the parameter in the object catalog.

Input

Name	Data type	Description
udiTimeOut	UDINT	The definition of monitoring watchdog time, in milliseconds. If the reading of the parameters has not been completed when this time expires, an error message is output.
pBuffer	CAA_PVOID	Pointer to the data buffer, the data buffer stores data after successful transmission of parameters
szSize	CAA_SIZE	The size of the data buffer (pBuffer) in bytes.

输出

Name	Data type	Description
xDone	BOOL	TRUE: Complete parameter reading without error.
xBusy	BOOL	TRUE: The reading has not been completed yet.
xError	BOOL	TRUE: An error occurred during reading.
eError	ETC_CO_ERROR	Information about the cause of the error displayed by xError, such as ETC_CO_TIMEOUT when timed out
udiSdoAbort	UDINT	If an error occurs in the device, this output will provide more information about it.
szDataRead	CAA_SIZE	Number of bytes read; maximum szSize (input).

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	The cause of the error is stored in the output udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Can't find the main station
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited and size> 4
ETC_CO_TIME_OUT	5753	Beyond time limit
ETC_CO_FIRST_MF	5770	Unused
ETC_CO_LAST_ERROR	5799	Unused

ETC_CO_SdoRead4

Library: IODrvEtherCAT

This function block is used to read EtherCAT slave parameters. Unlike ETC_CO_SdoRead, it only supports parameters no longer than 4 bytes. The parameters to be read are specified using Index and Subindex, as used in the object catalog.

输入

Name	Data type	Description
xExecute	BOOL	Rising edge: Start to read the slave parameters. In order to release the internal channel again later, the instance must be called at least once by xExecute: = FALSE.

输入

Name	Data type	Description
xAbort	BOOL	TRUE: The current reading process is aborted
usiCom	USINT	EtherCAT master station number: If an EtherCAT master station is used, usiCom is always 1. If multiple master stations are used, 1 specifies the first one, 2 specifies the second, and so on.
uiDevice	UInt	The physical address of the slave. If the automatic configuration mode is deactivated in the master station, the slave station can be provided with its own address. This address must be specified here. If the automatic configuration mode is activated, the address of the first slave is 1001. The current slave address can be checked in the slave dialog box in the EtherCAT address area device editor.
usiChannel	USINT	Reserved for future expansion
wIndex	WORD	The index of the parameter in the object directory.
bySubindex	BYTE	The sub index of the parameter in the object catalog.
udiTimeOut	UDINT	The definition of monitoring time, in milliseconds. If the reading of the parameters has not been completed when this time expires, an error message is output.

输出

Name	Data type	Description
xDone	BOOL	TRUE: Complete parameter reading without error.
xBusy	BOOL	TRUE: The reading has not been completed yet.
xError	BOOL	TRUE: An error occurred during reading.
eError	ETC_CO_ERROR	Information about the cause of the error displayed by xError, such as ETC_CO_TIMEOUT when timed out
udiSdoAbort	UDINT	If the device has an error, this output will provide more information about it
abyData	ARRAY [1..4] OFBYTE	Read the 4-byte array to which the parameter data is copied
usiDataLength	USINT	The number of bytes read (1, 2, 4).

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	No error
ETC_CO_FIRST_ERROR	5750	The cause of the error is stored in the output udiSdoAbort
ETC_CO_OTHER_ERROR	5751	Master station not found
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited and size> 4
ETC_CO_TIME_OUT	5753	Beyond time limit
ETC_CO_FIRST_MF	5770	Unused

ETC_CO_LAST_ERROR	5799	Unused

ETC_CO_SdoRreadDWord

Library: IODrvEtherCAT

Similar ETC_CO_SdoRead4, this function block is used to read TheerCAT from the station parameters. However, the data to be read is transmitted in DWWORD (dwData) instead of an array. If byte switching is required, it is performed automatically, so the read data can be reused directly.

ETC_CO_SdoWrite

Library: IODrvEtherCAT

This function block is used to write EtherCAT dependent parameters. Unlike ETC_CO_SdoWrite4, parameters that are not longer than 4 bytes can be supported. The parameters to write are specified by the index and sub-index, as used in the object directory.

Input

Name	**Data type**	**Description**
xExecute	BOOL	Rising edge: start reading from the parameter. In order to release the internal channel again later, the instance must be called at least once by xExecute: = FALSE.
xAbort	BOOL	TRUE: End of the current write process.
USICOM	USINT	EtherCAT Master Number: If only one EtherCAT master station is used, usiCom is always 1. If you use more than one primary station, 1 specifies the first, 2 specifies the second, and so on.
UIDevice	UINT	The physical address from the station. If you deactivate automatic configuration mode in the primary station, you can provide your own address for the from the station. This address must be specified here. If the automatic configuration mode is activated, the address of the first from the station is 1001. The address of the current from the station is always located in the EtherCAT address area and the tab of the from the station.
usiChannel	USINT	Reserved for future expansion
wIndex	WORD	The index of the parameter in the object directory.
bySubindex	BYTE	The sub index of the parameter in the object catalog.
udiTimeOut	UDINT	The definition of monitoring time, in milliseconds.

Input

Name	Data type	Description
		If the parameter writing has not been completed when this time expires, an error message is output.
pBuffer	CAA_PVOID	Pointer to the data buffer containing the data to be written.
szSize	CAA_SIZE	The size of the data buffer (pBuffer) in bytes
eMode	ETC_CO_MODE	The number of bytes to be written. Possible inputs: • ETC_CO_AUTO • ETC_CO_EXPEDITED • ETC_CO_SEGMENTED AUTO mode is usually set, so the mode suitable for the length is automatically used.

Output

Name	Data type	Description
xDone	BOOL	TRUE: The parameter writing is completed without error.
xBusy	BOOL	TRUE: Writing has not been completed yet.
xError	BOOL	TRUE: An error occurred during writing.
eError	ETC_CO_ERROR	Information about the cause of the error displayed by xError, e.g. ETC_CO_TIMEOUT on timeout
udiSdoAbort	UDINT	If an error occurs in the device, this output will provide more information about it
szDataWritten	CAA_SIZE	Number of bytes written; max szSize (input).

ENUM ETC_CO_MODE

AUTO	0	Automatic mode selection by the client
EXPEDITED	1	Client uses acceleration protocol
SEGMENTED	2	Client uses segmentation protocol

ETC_CO_SdoWrite4

Library:IODrvEtherCAT

This function block is used to write EtherCAT slave parameters. Unlike ETC_CO_SdoWrite, only parameters not longer than 4 bytes can be supported. The parameters to be written are specified by Index and Subindex and are used in the object directory.

Input

Name	Data type	Description
xExecute	BOOL	Rising edge:Start reading slave parameters.
xAbort	BOOL	TRUE:The current writing process is aborted.

Input

Name	Data type	Description
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 on the Slave tab 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 writing of the parameters is not yet complete on expiry of this time, an error message is output.
abyData	ARRAY [1..4] BYBYTE	Contains the data to be written. The data must be saved in the Intel byte order.
usiDataLength	USINT	Number of bytes (1,2,4) to be written.

Output

Name	Data type	Description
xDone	BOOL	TRUE: Writing of the parameter was completed without error.
xBusy	BOOL	TRUE: Writing is not yet completed.
xError	BOOL	TRUE: An error occurred during writing.
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

ENUM ETC_CO_MODE

AUTO	0	The client automatically selects the mode
EXPEDITED	1	The client uses the expedited protocol
SEGMENTED	2	The client uses the segmented protocol

ETC_CO_SdoWriteDWord

Library:IODrvEtherCAT

Just like ETC_CO_SdoWrite4, this function block is used to write EtherCAT slave parameters. However, the data to be written is not transferred as an array but is passed to DWORD(dwData). If byte swapping is required, this is performed automatically. The value to be written can therefore be specified directly.

ReadMemory

Library: IODrvEtherCAT

This function block is for reading the memory of EtherCAT Slaves.

Input

Name	**Data type**	**Description**
xExecute	BOOL	Rising edge: Starts the reading. Falling edge: Resets outputs. If a falling edge occurs before the function block has completed the command, the outputs continue working normally. They are reset only if the command has either been fully executed or aborted (xAbort) or if an error occurs. In this case the corresponding output values (xDone, xError, iError) are present at the output for precisely one cycle.
xAbort	BOOL	TRUE: Command is immediately aborted and all outputs are set to their initial values.
USICOM	USINT	Index number of the EtherCAT master (1 for the first master…)
wSlaveAddress	WORD	Automatically increased address or physical address of the device.
xAutoIncAdr	BOOL	Flag for interpretation of the address
xBroadcast	BOOL	Flag indicating whether broadcast reading is to be used. TRUE: wSlaveAddress and bAutoIncAdr are not used
uiMemOffset	UINT	Offset of the memory in the EtherCAT slave memory image
iSize	INT	Number of bytes to be read.
pDest	POINTER OF BYTE	Buffer for the storage of the data
udiTimeOut	IDINT	Watchdog time for the command in ms

Output

Name	Data type	Description
xDone	BOOL	TRUE: Reading was completed without error.
xBusy	BOOL	TRUE: Reading is not yet completed.
xError	BOOL	An error occurred during reading; the function block aborts the command.
xAborted	BOOL	Command was aborted by the user.

Example: reading the register 0x130 (current status)

```
PROGRAM  PLC_PRG
VAR
      etcreadmemory  :ReadMemory;
      wStatus  :WORD;
      xRead  :BOOL;
END_VAR

etcreadmemory(xExecute  :=  xRead,  usiCom:=1,  wSlaveAddress  :=  1002,
            xAutoIncAdr  :=  FALSE,  xBroadcast  :=  FALSE,  uiMemOffset  :=  16#130,
            iSize  :=  2,  pDest  :=  ADR(wStatus),  udiTimeout  :=  500);
```

WriteMemory

Library: IODrvEtherCAT

This function block is for writing the memory of EtherCAT slaves.

Input

Name	Data type	Description
xExecute	BOOL	Rising edge: Starts the writing Falling edge: Resets outputs. If a falling edge occurs before the function block has completed the command, the outputs continue working normally. They are reset only if the command has either been fully executed or aborted (xAbort) or if an error occurs. In this case the corresponding output values (xDone, xError, iError) are present at the output for precisely one cycle.

Input

Name	Data type	Description
xAbort	BOOL	TRUE: Command is immediately aborted and all outputs are set to their initial values.
usiCom	USINT	Index number of the EtherCAT master (1 for the first master…)
wSlaveAddress	WORD	Automatically increased address or physical address of the device.
xAutoIncAdr	BOOL	Flag for interpretation of the address
xBroadcast	BOOL	Flag indicating whether broadcast reading is to be used. TRUE: wSlaveAddress and bAutoIncAdr are not used
uiMemOffset	UINT	Offset of the memory in the EtherCAT slave memory image
iSize	INT	Number of bytes to be written
pDest	POINTER OF BYTE	Buffer for the storage of the data
udiTimeOut	IDINT	Watchdog time for the command in ms

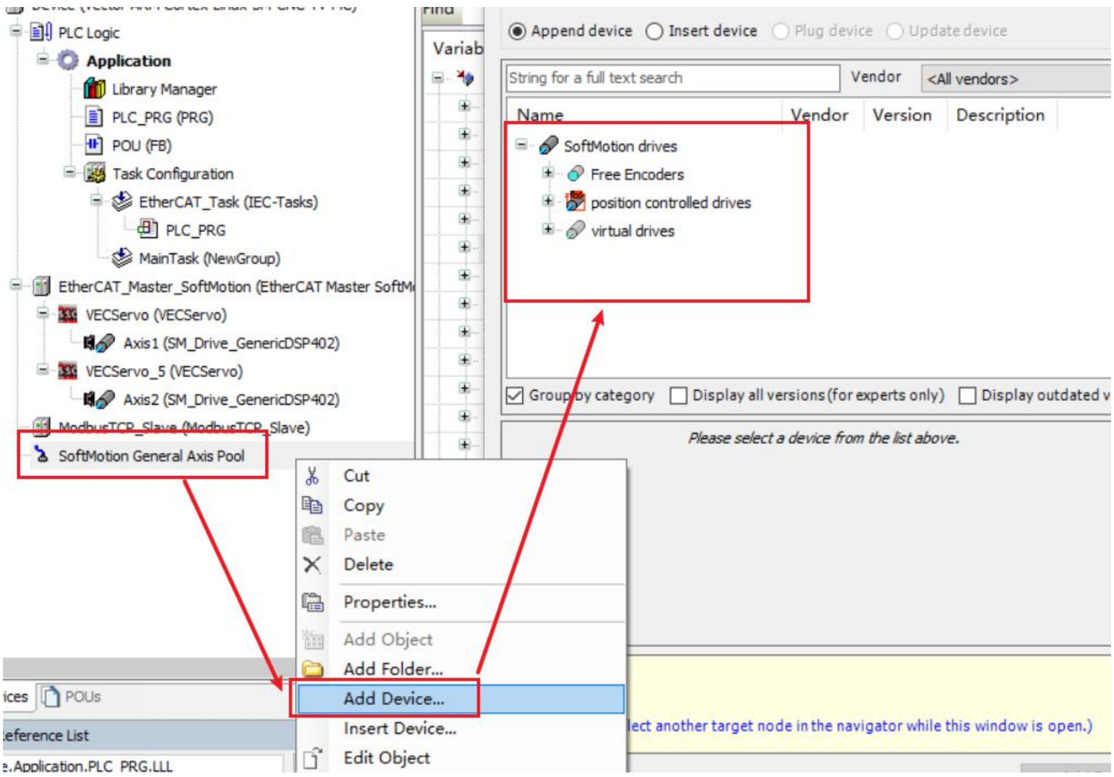
Output

Name	Data type	Description
xDone	BOOL	TRUE: Writing was completed without error.
xBusy	BOOL	TRUE: Reading is not yet completed.
xError	BOOL	TRUE: An error occurred during reading; the function block aborts the command.
xAborted	BOOL	TRUE: Command was aborted by the user.

4.3.8 SoftMotion General Axis Pool

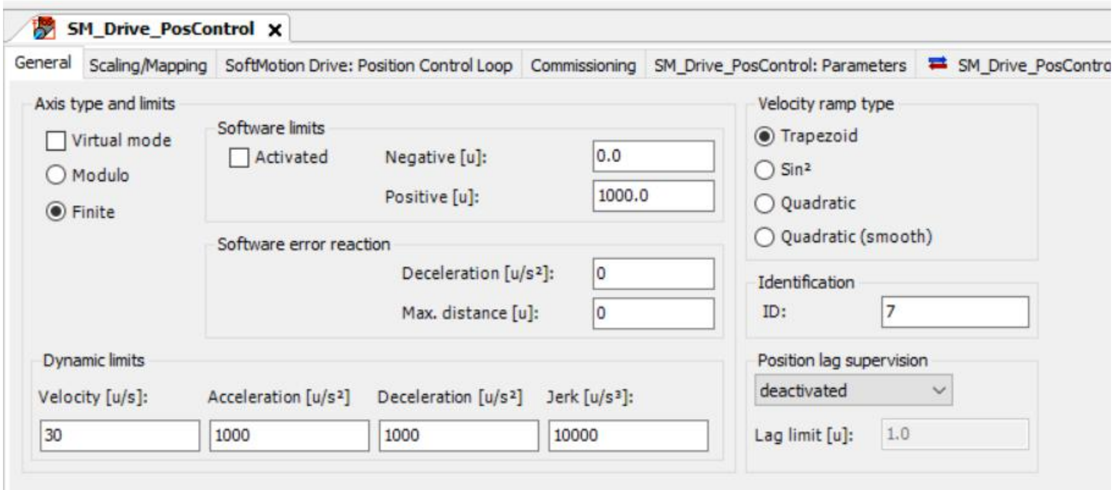
If a SoftMotion PLC is used (e.g. CODESYS SoftMotion Win V3), the base libraries are automatically linked in the Library Manager. A SoftMotion General Axis Pool is available for these types of controllers. SoftMotion free drive units can be inserted here.
The SoftMotion Drive Interface is a standard interface for linking, configuring and addressing drive hardware in the IEC program. By mapping different hardware to one interface, drives can be easily exchanged and IEC programs can be reused. The interface couples the drive to the I / O mapping and is responsible for updating the required motion data and transferring it to the drive control.

The method for adding a SoftMotion free drive is shown in the following diagram.



Position control drives

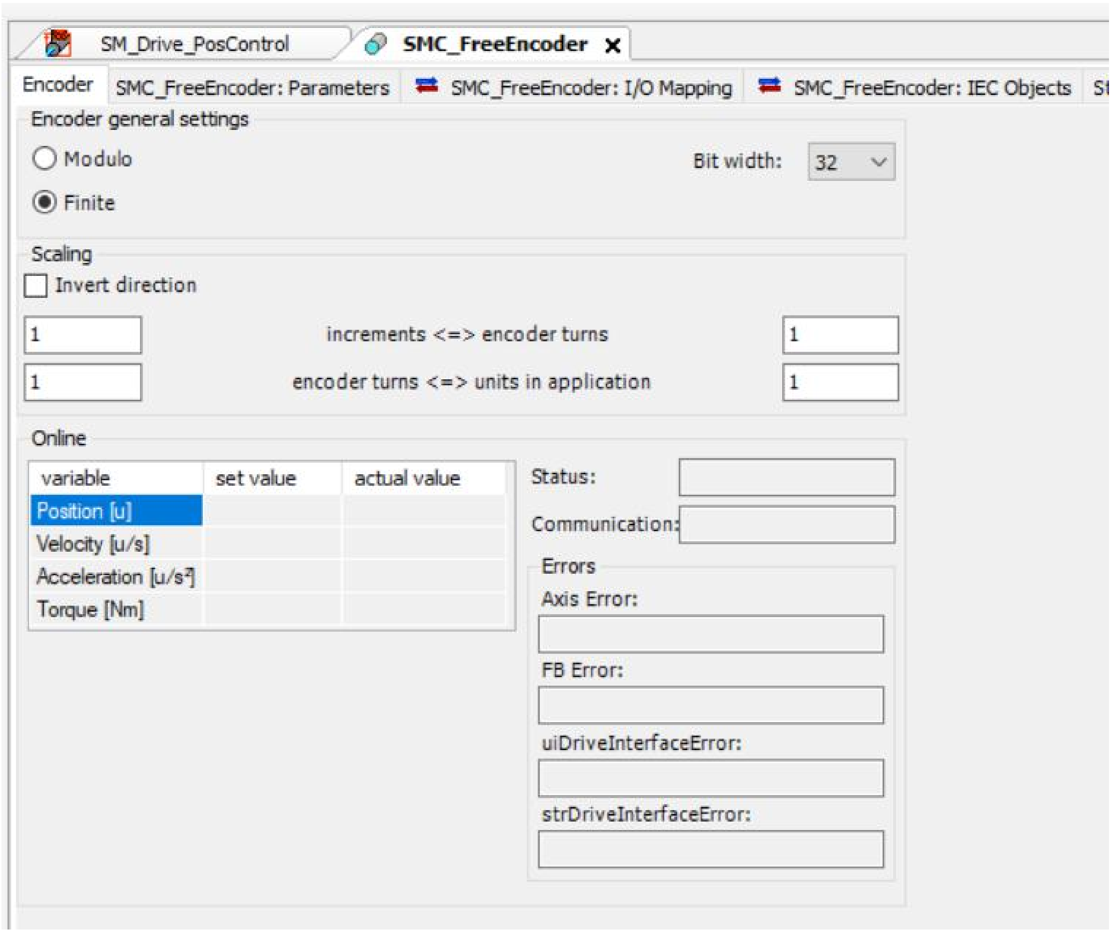
Position control of the CODESYS axes can be run using the SM_Drive_PosControl drive control. The requirement is for a device that is controlled by the set speed and returns its current position. It can be, for example, a speed control device (frequency converter) with position feedback.



Free encoder

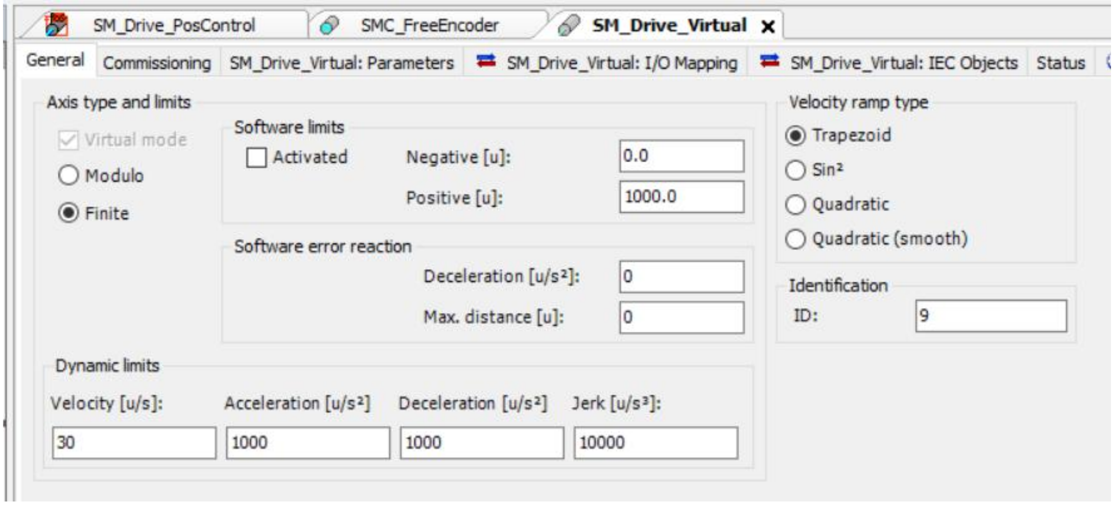
Use SMC_FreeEncoder to integrate encoders that are not permanently coupled to I/O or hardware.

Assign the input value of the encoder to the variable <FREE_ENCODER_AXIS>.diEncoderPosition. this can be done as IEC code or by mapping the memory of the input data.



Virtual drives

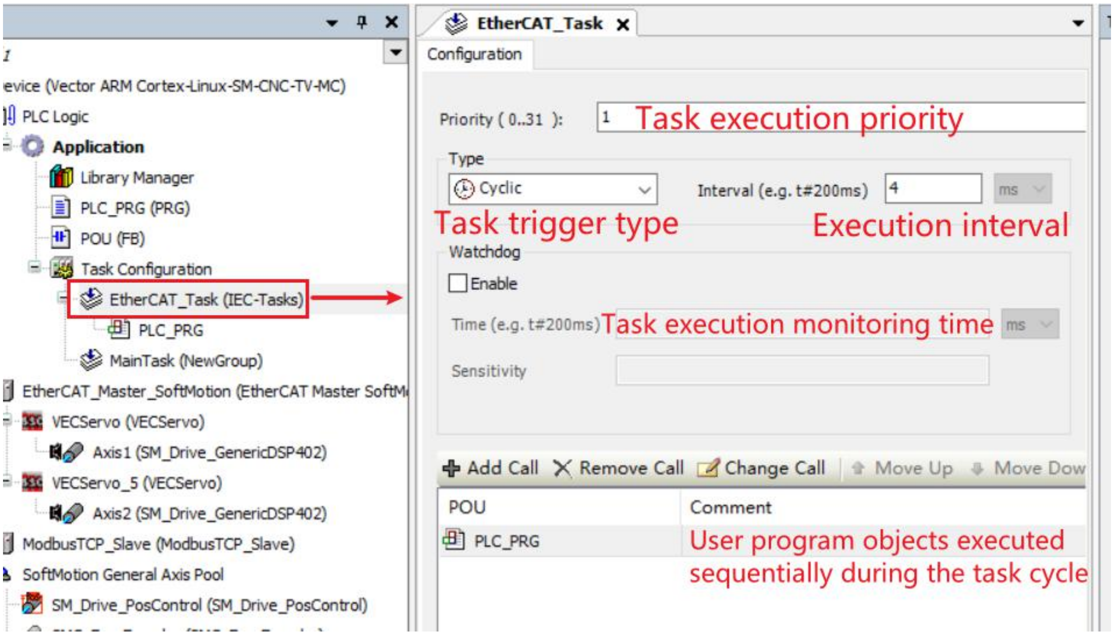
The virtual drive SM_Drive_Virtual is a simulated drive in software. It is possible to test programs or implement extended functions without connecting hardware. These types of functions include, for example, the control of axis movements.



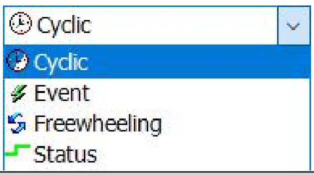
5 VE controller program execution mechanism

5.1 User engineering tasks and configuration

As shown in the diagram, each task group can have its own execution trigger conditions, execution period, execution priority, etc.



The types of tasks supported by the VE controller are as follows:



Type of task execution	Type description	Example
Cycle	Execute the corresponding POU once at each set time interval	EtherCAT Bus Tasks General Task Loop
Event	In the set Bool type variable state 0→1 Trigger execution once	Soft interrupt handling POU
Freewheeling	Once execution has started, the cycle is repeated without interruption	General task cycle
Status	If the state of the set Bool variable is 1, the loop is repeated	Conditional execution task POU

5.1.1 Key points of task configuration

When set to the "Cycle" type, the "task cycle" refers to the time interval to perform the task. For general logic control, where the state of common IO port variables changes slowly, the task cycle can be set to a larger period, e.g. 20ms; for tasks that need to be processed in time, the task cycle can be set to a smaller period.

The task configuration for EtherCAT bus communication is a special "cyclic" task with the highest priority. The set value for the task cycle is also the EtherCAT bus communication cycle, usually set to 1ms-4ms; the smaller the set value, the higher the accuracy of the motion control; the larger the number of axes to be controlled, the larger the set cycle, otherwise the CPU will be overloaded with calculations.

A task configuration can only be set to one execution type, time interval and priority, and to obtain different execution characteristics, multiple task configurations can be added. A task configuration can contain multiple POUs, all of which will be executed at the same time interval and in the order in which the POUs are added to the task.

5.1.2 Prioritisation of tasks

For tasks with different object types, it is recommended that different priorities are assigned to ensure that important tasks such as motion control are prioritised, allowing the controller's performance to be used wisely in some applications where high performance motion control (MC) is required. The order of task priority is as follows.

Priority	Type of task	Description
0	EtherCAT Bus tasks	Highest priority, only one EtherCAT task allowed
1	ModbusTCP	
2	ModbusRTU	
3	MainPOU	Lowest priority

When the controller performs a task, there is a time alignment point unobserved by the user at which it starts, at the highest priority → Second highest priority →···Execution starts in the order of the lowest priority; a lower priority task may be interrupted by a higher priority task while it is being executed, and when the execution of the higher priority task is complete, the interrupted task is returned and execution of that lower priority task continues.

The EtherCAT task is the highest priority task and is entered in the EtherCAT cycle and all POUs within the task are executed before returning to the lower priority task.

5.1.3 Execution cycle setting in task configuration

The CODESYS software uses a multitasking approach to execute the user program's "tasks", each of which is assigned a different execution period. Some global variables may have to be accessed and modified between different POUs, so global variables need to be synchronised interactively, also at the "time alignment point" of the task, in integer multiples when setting the period of a cyclic type task. Do not set the EtherCAT period to 3ms, 6ms, 7ms, 9ms etc. as this may result in a non-integer multiple relationship.

5.2 Data flow analysis in EtherCAT bus networks

5.2.1 Network overview of the EtherCAT bus

The EtherCAT bus is commonly connected using RJ45 plugs, multi-core Ethernet cables and recommended Super 5 cables for improved interference immunity. Similar to a common Ethernet network, the network communicates at a rate of 100 Mbps, with link cable lengths of up to 100 m per adjacent slave, etc.

The EtherCAT network differs significantly from a normal Ethernet network in that there is only one EtherCAT master in the network and the network-specific ESC (EtherCAT Slave Controller) inside the master can receive the communication data sent to this station and insert the reply data from this station into the frame in real time. The communication data frames in the EtherCAT bus follow the Ethernet data UDP/IP frame structure, type 0x88A4, except that the intermediate data fields have to be prepared and analysed according to the EtherCAT communication protocol.

The EtherCAT segments can be further defined and parsed by some protocol, and data communication can be achieved as long as both the primary and the host stations comply with this protocol. Protocols typically used include CANopen Over EtherCAT (CoE) and Sercos Over EtherCAT (SoE), just as Modbus Protocol Frame Data (ModbusTCP) is transmitted on TCP/IP networks.

The VE controller uses the CoE protocol, the DS402 regulation (also known as CiA402) for the CANopen protocol, which is a dedicated protocol for servo motion control classes, the most important features of which are:

(1) In order to improve communication efficiency, the master-from station is not accessed by means of a question-and-answer approach, but during the initialization phase of the bus network, the master gives the host station a list of data items to be sent in advance to the master, such as a "process data PDO", informing it that the host station will send the data items and sequence (TPDO), requiring thedata items and sequence (RPDO) sent from the station, so that the receiving from the station To the main station data frame know how to parse, you can also prepare the required answer data in advance, when the main station dataframe arrived, each from the network control chip (ESC) can take the data segment sent to the station, for the station's processor according to the configuration table for analysis, and in the appropriate stage of The EtherCAT communication frame timely insert the answering data block of the station, returned to the main station;

② The data to be communicated by the user is divided into "process data PDO" and "service data SDO" according to the real-time requirements, with the former PDO arranged for high-frequency cyclic sending and receiving, and the latter SDO communicating only when needed.

③ The control command parameters, operation status parameters and function code setting parameters of the servo drive, the most number of which reaches hundreds, are named differently for each brand of servo parameters, and in order to ensure that the master and slave stations of different brands are interchangeable, an "object dictionary OD" has been developed in the CiA402 protocol to list all the functions used in the servo drive. All the function codes, operation commands and their set value meanings, operation status parameters and the scale to be used in the drive are defined specifically, forming a professional technical specification, and equipment suppliers of different brands can operate with the VE controller as long as the products developed in accordance with this CiA402 protocol specification can ensure universality and interchangeability.

④ The configuration of communication objects between master and slave stations is a condition to ensure the successful execution of the functions of the operation and control function block. When executing the MC function block in the user program, the controller needs to use specific "communication data objects" to send commands to the servo slave and read the slave's axis status.

⑤ The slave device may not support all the items defined in the "Object Dictionary OD", but the device manufacturer has defined the "Device Description File EDS" for the device, so the programming user needs to import the device description file EDS of the slave device in CODESYS before configuring the device, and can see the contents of the supported objects.

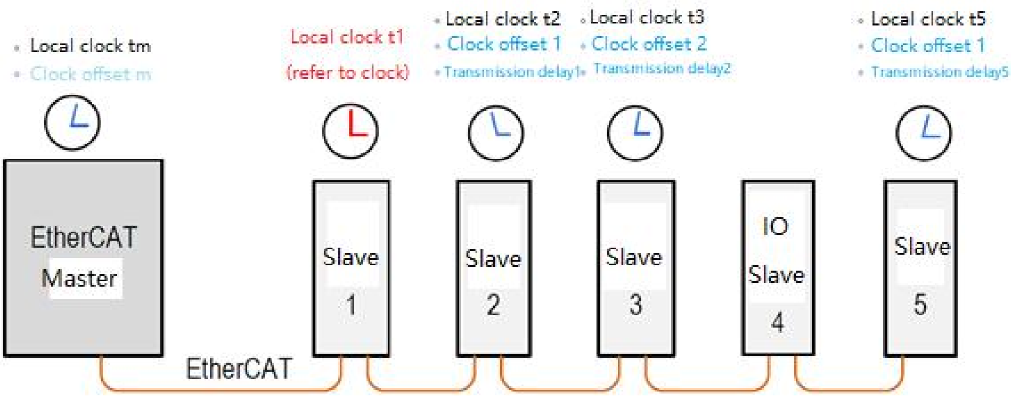
(6) When writing the user project, the user selects and configures the TPDO and RPDO data object tables according to the control needs, which will be automatically forwarded by the master to the corresponding slave during operation by means of communication; try to select only the required configuration items and reduce the configuration items of irrelevant data objects, which will reduce the load of EtherCAT communication and help to improve the communication efficiency.

(7) The SDO configuration item is generally used to initialise the function code of the slave device at the beginning of the system, and can also be used to access the parameters via function blocks such as MC_SDOread during operation, which has a lower communication timeliness and takes up additional EtherCAT communication overhead, and can cause synchronisation timeout failures in applications with a high bus load rate.

5.2.2 Synchronous clocking of the EtherCAT bus

As a multi-axis motion control network, it is often necessary to have multiple slave stations start or stop motion at the same time. The EtherCAT network has a Distributed Clock (DC) mechanism, which allows each intelligent slave station (e.g. servo drive, intelligent

high-speed expansion module) to have a consistent clock, and each slave station outputs the data written by the master station to the execution unit according to a set synchronisation trigger period to achieve simultaneous
operation.



During the initialisation phase of the EtherCAT bus, the master reads the current time of each slave and uses the local time of the first slave as the "reference clock" for the network, so that the "clock offset Toffset" of each slave relative to the reference clock can be calculated, and the clock offset of each slave is written to the corresponding slave so that it can correct its clock and eliminate static errors.

In addition, during the transmission of the communication data frames, there will be a transmission delay time due to the hardware network, the master will send a specific broadcast frame, let each slave station record the data arrival time, the master will then read the time value recorded by each slave station, while measuring the total delay of the data return data frame, can accurately calculate the "transmission delay Tdelay" of each slave station "The master then writes the transmission delay time of each slave into the memory of each slave. With these clock corrections, the slave gets the same clock as the reference clock t1 by calculating TLocal-Toffset-Tdelay.

In EtherCAT networks, IO slaves that are not sensitive to the DC clock can be set up without DC handling and the EtherCAT master ignores their clock calibration during the DC calibration. Once the synchronisation unit has been activated, a SYNC synchronisation signal is generated at regular intervals to validate the currently received data and, in the case of servo drives, to start execution with the received position command as the target point.

The initialisation and calibration of the DC clocks of the EtherCAT slaves as described above is done automatically by the EtherCAT master without user intervention and is completed when the EtherCAT bus is ready. It is important to note that slaves with internal clock functions are placed at the front of the network as far as possible.

5.3 Communication flow between VE controller and servo slaves

In EtherCAT communication, the CoE is used at the application level. When the controller executes the Motion Control (MC) user program, the communication data between the controller system software and the servo is processed through a multi-level functional unit.

5.3.1 Step-by-step description of the control information process

Step 1: Execute the MC function block of the user program and process the command data to be sent

When the controller executes the user program, it executes an MC motion control function block instance, e.g. MC_MoveRelative (Axis_1), and the controller, based on the slave (Axis_1) state machine and data structure in memory.

checking the current state of the slave axis, reporting an MC execution error if the slave axis is not enabled, or is running in torque mode, or is running in synchronous mode, or is Homing running, or is alarming, etc.; if the slave axis is stopped, or is running in non-synchronous mode in position mode, sending the make slave axis run command ControlWord.

analyses the current running speed fActPosition of the slave axis, the running speed fActVelocity, and constraints such as target position, maximum allowed speed, acceleration, deceleration, etc. to calculate the required motion position command TargetPosition for the next running cycle.

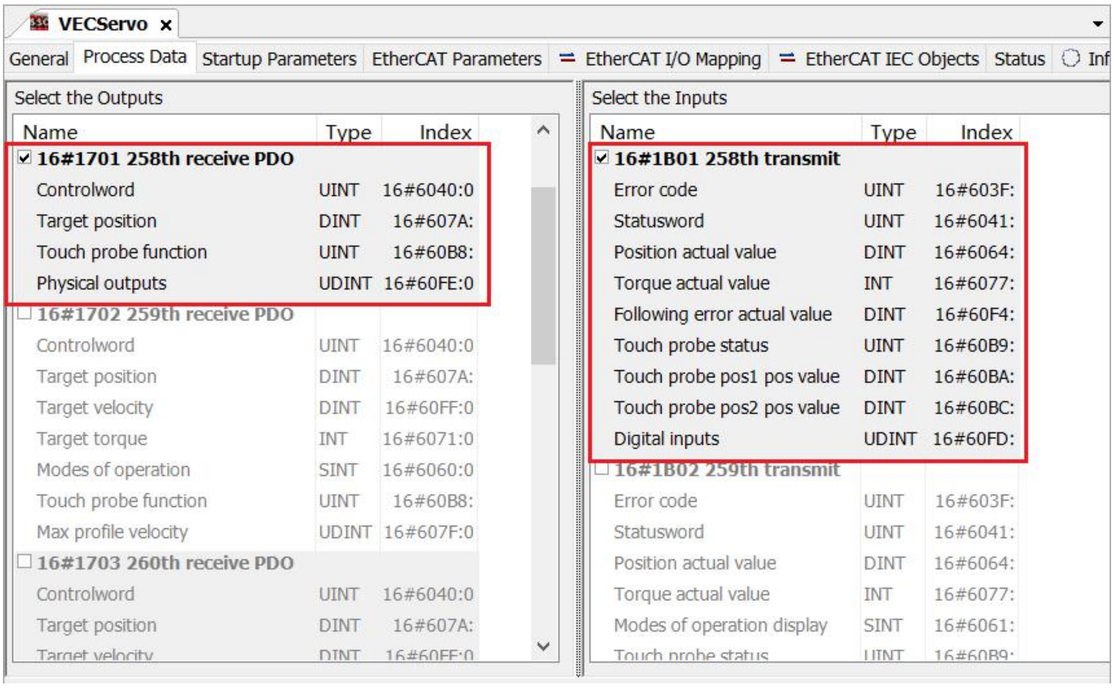
The controller also needs to wait for the data returned from the slave station in the next communication cycle to analyse and judge the execution of this MC function block instruction, so that the user can know whether the execution is Busy, Done, Error, or Aborted by other MC instructions, etc.

Step 2: Place the control command data to be sent into the EtherCAT Send Buffer Unit

The command data ControlWord and TargetPosition that need to be sent to the slave Axis_1 are stored in the PDO send buffer unit, a prerequisite for this operation is that these two parameters (called "objects" in the CiA402) are already present in the PDO configuration table.) option; the "PDO configuration table" holds the "index number" (main index number: sub-index number) of the control parameters (objects) that need to be sent and read by the master, divided into TPDO and RPDO.

	Description of use
TPDO layout table	A list table of objects and attributes that need to be configured by the user when programming, based on what needs to be sent cyclically for the control of the slave. This table is automatically sent by the controller to the slave ESC during the network initialisation phase. (b) The controller master will arrange the size of the transmit buffer according to this table and at runtime will store the command data to be sent into the transmit buffer. the slave stations at runtime parse the received data frames according to this table. each slave can have a different TPDO configuration table.
RPDO layout table	A list table of objects and attributes that need to be configured by the user when programming, based on the content of the objects that need to be automatically answered by the slave. This table is automatically sent to the slave ESC during the network initialisation phase. The runtime slave prepares the data according to this table and returns it to the master in time for insertion into the time slot of the EtherCAT data frame when the master accesses this slave. At runtime the master parses the returned data frames according to this table and returns the slave answer data. Each slave can have a different RPDO configuration table.

The following diagram shows the PDO configuration table, in which the index number and data type of each control parameter are specified by the CiA402 protocol, and the "index number" can be used to find out what the parameter is and the width type of the parameter in the "Object Dictionary
OD".



During the initialisation phase of the network, the master sends the "PDO Configuration Table", which contains the TPDO, RPDO, data type and width of each object, to the slave as the basis for parsing the data frames.

The order of the objects in the table will be the basis for the system to place the data to be sent by the MC command into the transmit buffer unit. As shown above, the ControlWord is placed in the first transmit unit, the TargetPosition is placed in the second unit, and so on.

The slave station, according to the RPDO configuration table (9 "objects" in the above diagram), will put the index number and order of each object, in turn, the servo's operation status data into the answer cache unit, when the master station communication frame accesses this slave station, the ESC will automatically insert the data of this cache unit into the appropriate time slot of the data frame and return it to the master station.

The RPDO table will also be the basis for the master to parse the slave's answer data.

Step 3: The master control chip sends the data from the transmit cache unit to the slave ESC at regular intervals and the slave sends the reply data at the same time

The controller, as master, generates EtherCAT interrupts at regular intervals according to the EtherCAT clock cycle set by the user. After entering an EtherCAT interrupt, it initiates EtherCAT communication and sends the data of the PDO transmit cache unit to several slaves in one or several frames, and incidentally retrieves the answer data of each slave in the same communication frame.

In chronological order, the data in the controller's transmit cache is the command data from the previous EtherCAT interrupt POU execution; the reply data from the slaves is not the answer to the master's query, but the current value of the cyclic reply "object" as required by the RPDO configuration.

Step 4: The slave receives and parses the data sent to this site from the master

After entering the normal operation state of the network, the ESC from the station will receive the communication data frame sent by the main station on a timely date and automatically store the data sent to the station in the communication frame to the local cache.

After receiving a string of PDO data, the processor of the station intercepts the received data string according to the specified object data type (width) according to the TPDO table, and stores the parameter properties represented by the "object index" number to the corresponding control command unit for the operation control of the servo

The processor of the station will, according to the object properties and order required by the configuration table of the RPDO, brush the current running state and parameters of this servo axis, cycle through the answering cache unit in the new local ESC, and insert the cached data into the EtherCAT communication frame with high-speed hardware operation to the main station in the appropriate communication frame time slot.

Step 5: The main station receives and parses the data from the station answer, updates the axis state parameters, and determines whether the execution is complete

The controller, as the etherCAT network master station, sends data frames at the same time, it receives the communication frames sent back by the closed loop of the network from the station at the same time, and can extract the data strings that the receiving station answers from from , and can also judge the communication status of the network and analyze the success of the communication operation.

Based on the data answered from the station, such as Error Code, Status Word, Position Actual Value... The controller system can determine whether it has reached the operating position required by the MC function block instance and refresh the output variable state of the MC function block instance

The above is the VE controller's EtherCAT packet sending, receiving and parsing principle process description, easy for users to understand its internal mechanism, many of the links are automatic system completion, do not need user intervention, users only need to understand the CiA402 object concept, master the servo axis commonly used "like" type, TPDO, RPDO configuration table object selection on it.

5.3.2 CiA402 Data Object Dictionary and Servo Common Objects

The EtherCAT bus communication layer of the VE Motion Controller uses the CANopen DS402 protocol, also known as CiA402, which is part of the CANopen protocol "Servo and Motion Control" protocol family. EtherCAT bus networks. Controllers and servo drives (slave devices) developed by different device manufacturers according to this protocol can be used in conjunction with or instead of each other, giving the user more choice and meeting the aims of the PLCopen specification.

CiA402 object types are grouped into index number segments by attribute, as follows:

Master index number range	Meaning	Description
0x0000~0x1FFF	Protocol type descriptions, manufacturer information, line type descriptions, configuration table description information, etc.	There is information initialisation by the device manufacturer and the configuration information is done automatically by the system software without the intervention of the controller user.
0x2000~0x5FFF	Manufacturer-defined objects, functional properties of the objects defined by the equipment vendor	The device vendor can design the master index number as a function code for the servo drive and use it to set the function code parameters for the static parameters
0x6000~0x9FFF	Line definition data objects for the control and monitoring of equipment	Data for communication interaction between the controller and the servo for control
0xA000~0xFFFF	Reserved	

As can be seen from the table above, the objects required for motion control are mainly in the(0x6000-0x9FFF) index segment section, if you want SDO configuration to modify the servo function code, you need to pay attention to the(0x2000-0x5FFF) index segment section. Detailed index number instructions may be referred to in the EtherCAT bus-type servo instructions of the Wykoda servo instruction manual, which will not be repeated here.

The VE controller controls the operation of the servo and generally has several command types:

1) control the operating state of the servo, such as enabler, origin regression, start-stop operation, alarm reset, etc.

2) set the operating mode of the servo, such as position mode, speed mode, torque mode;

3) Set the target position, running speed and output torque of servo operation;

4) Read the operating status of the servo system, such as operating state, operating mode, position, current speed, output torque, etc.

5) Set or modify the function code parameters of the servo system, run the constraint parameters, etc.

To complete these control operations, there are several commonly used data objects, which must be programmed in the PDO or SDO configuration table, and some data objects, which are added as appropriate based on the functionality used in the user program.

This section describes the value of the data object, used to explain the functional definition of the object, the actual runtime, the controller will automatically send the corresponding value according to the required control operation: for the PDO configuration table, the user only needs to add the controller to use the data object, do not need to fill in the specific parameter value or variable name,

When codeSYS software is compiled, the variables in the MC function block are automatically associated with the object of the PDO, and for the SDO configuration table, the operation (write) that is typically used for the controller to initialize the servo function code is a definite constant, since the constant to be filled in must conform to the DS402 specification definition, and some are defined as a constant by the functional code unique to the servo drive.

5.3.3 Configuration of servo shaft motor parameters

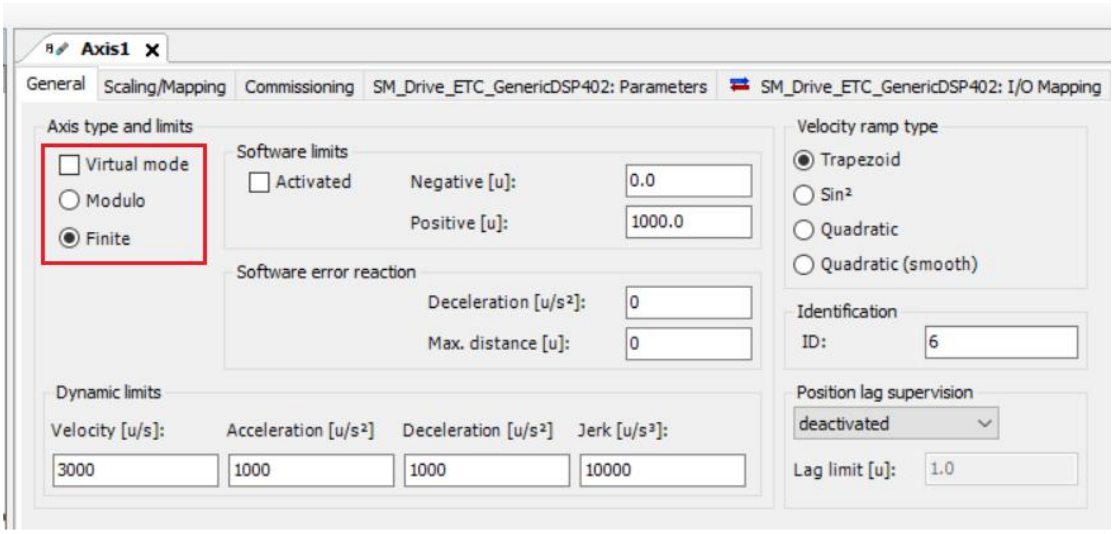
Motion control of the action, the most through the operation of the servo motor to achieve, to servo motor in accordance with our hope, the controller needs to know the servo motor parameters, the application system mechanical transmission mechanism characteristics parameters, as well as the user's desired operating characteristics, in order to send the appropriate operating position instructions, which need to be programmed to set these characteristic parameters on the controller.

The method of setting servo motor parameters is shown in the following diagram, double-click SM_Drive_GenericDSP402,and themotor-related parameters can be set in the window on the right.

(1) Under the basic parameter label, the main axis position counter module value is set. If the servo motor is the characteristics of round-trip operation, such as the re-operation of the wire rod, you can choose "limited", (also known as multi-turn mode, limited long mode), convenient in the case of servo motor rotation multi-turn, can carry out absolute position mode positioning;

(2) If the servo motor is running in one direction indefinitely, e.g. the operation of the fly shear roll, you can select "module", its position counter in each operating cycle, counting from 0, will not produce a position counter overflow;

(3) If you are running without an actual access servo, you can check Virtual Mode, which can be used for simulation operations

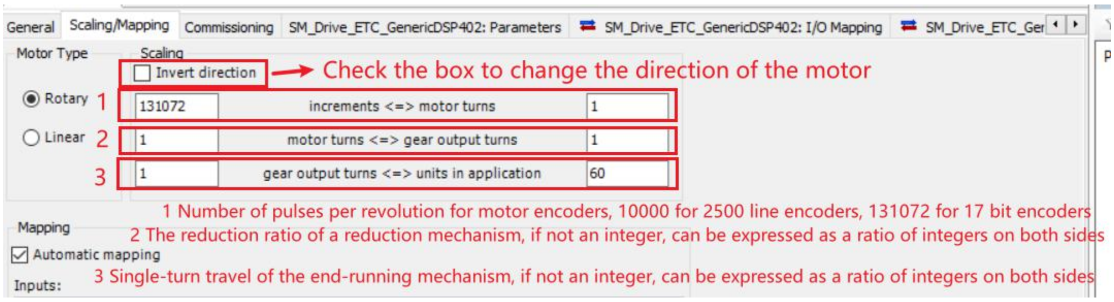


Attention:

The above setting rules, both applicable to the incremental encoder servo motor, but also suitable for absolute encoder servo motor, the above settings are not given to the servo driver, and the current position of the motor, by the VE controller according to the position signal of the motor feedback, as well as the above-mentioned setting parameters, automatically carry out the cumulative and quantum calculation of the position, therefore, if the servo position has the power-off hold characteristics, the user program needs to back up the current position of the shaft to the power-down hold variable, and then restore the relevant parameters.

The "software restriction" in the figure above refers to the software of CODESYS, which protects the travel limit of the servo motor so that the VE controller does not issue the over-limit positioning instructions, which is very useful in the application system of the MC instructions of the absolute position encoder and the absolute positioning instructions. There is also a selection of additional deceleration-special curves, which can be selected during commissioning to make the mechanical system run more smoothly.

Because the controller always takes the number of pulses that make the servo run as the operating position command, the controller must know the pulse value of the encoder of the servo motor per turn, and also know the mechanical parameters such as the deceleration ratio of the operating mechanism, the wire rod guide, the wheel doughnut peritution, which can be entered under the "zoom/map" label of the motor parameters, as shown below:

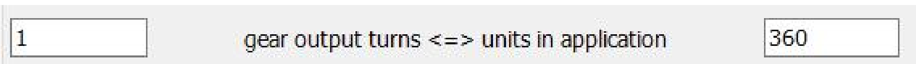


Item 1 in the figure above, which is used to set the number of pulses per lap;

Item 2 in the figure above, which is used to set the deceleration ratio of the gearbox, shows that the servo motor shaft rotates 1 turn for every 5 turns, and if the gearbox is not used, the deceleration ratio is 1:1;

Item 3 in the figure above sets the physical distance of the work piece for each 1 turn of the output axis. For example,

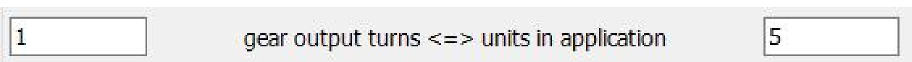
◆If the use of flying shear roll, we only pay attention to its rotation angle, can be filled in this way:



The MC_MoveRelative of the instruction is that the mechanism rotates by 1 degree;

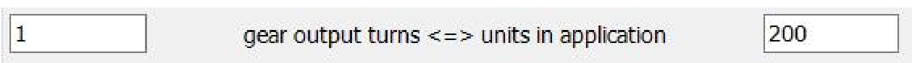
The MC_MoveRelative of the instruction is that the mechanism rotates by 360 degrees;

◆ If you are using a wire rod with a guide of 5mm, i.e. the slider on the wire rod moves 5mm at 1 turn, fill in this:



The MC_MoveRelative of the instruction is that the slider mechanism travels 1mm;

◆If you are using a synchronous gear with a diameter of 63.7mm, the moving distance of the synchronous belt is 63.7mm×3.14 x 200mm for each turn, which can be filled in as this:



The MC_MoveRelative of the instruction is that the belt mechanism travels 100mm;

It can be learned that through the accurate setting of 1/2/3 items, the application system physical units and MC operating instruction units can be achieved consistent, so that the instructions of the user program intuitive, convenient variable settings, not easy to errors.

Note: Setting the motor parameters is used for the VE controller to perform the conversion of the electronic gear ratio when sending the final (number of pulses) position command, and is not downloaded to the servo driver, while the electronic gear ratio set by the function code in the servo also attenuated the operating instruction, so in the following figure, the actual impact on the servo motor is Rc*Rd:

Therefore, to ensure that the user program performs the same in each application device, it is necessary to initialize the electronic gear ratio function code of the servo to the specified parameter value through SDO operation, otherwise, because of the different servo function code settings, it will result in a difference in the operating response.

5.3.4 EtherCAT network state initialization and management

(1) Initialization and status of the EtherCAT network

After the controller is powered on, it will start itself and complete the loading of the operating system and user programs.

If the user program does not use the EtherCAT bus, the controller will start the user program execution after the user program initializes the operation of the other bus, and if the user program uses the EtherCATcommunication network, the VE controller as theEtherCAT main station will initialize the EtherCAT network bus, including:

(1) According to the user's EtherCAT configuration, it takes about 3 seconds to configure the main station;

(2) Send the initial command of the network, let all from the station ESC control chip to start the initial operation, read the EtherCAT network information in turn, and compare with the EtherCAT network configuration in the user program, if there is a difference in the number and order of the station, it will report errors;

(3) If the network configuration is normal, SDO, PDO will be sent to each station ESC chip in turn;

(4) Let the network first enter Pro-OP, Safe-OP, and then OP operation;

The above operation process, is the controller automatically completed, does not need user intervention, from the number of stations, the network initialization time increased accordingly, the user program to determine whether the application system's network state started normally, the simplest and most reliable way is to detect each servo axis of the MC_Power.status state is true, because the state can indicate that the network is normal, servo normal, with the conditions to start running.

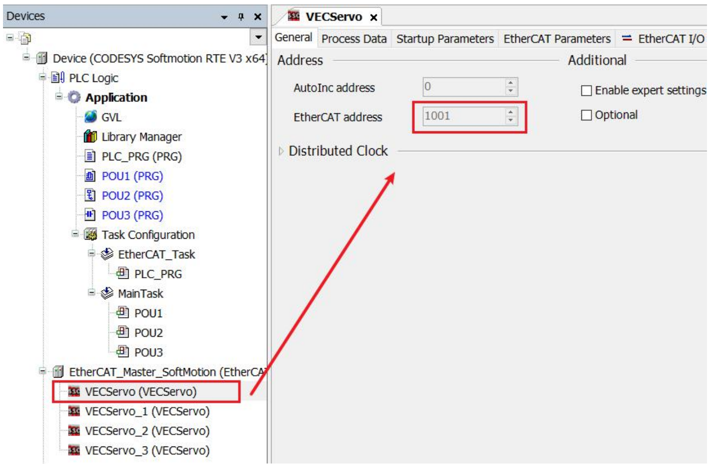
(2) Communication drop-off and communication recovery

EtherCAT from the station can communicate with the main station normal premise, from the station ESC after the main station configuration, has entered the network Pro-OP, Safe-OP, and then op operating state, ESC internal typical configuration content contains PDO configuration table, this information is only when the main station to the network configuration, from Station ESC can be obtained, and once the main station network into the OP operation state, can no longer occur configuration information, therefore, when the EtherCAT network main station into the operating state, from the station to power up, or from the station after the power down, will not be able to enter the network operation state.

EtherCAT currently resumes network operation after powering down from the station, only for the primary station to restart and start running, such as powering up again, and for the primary station to start running again, but this affects the operation of other stations.

(3) Address and setting of the address from the station

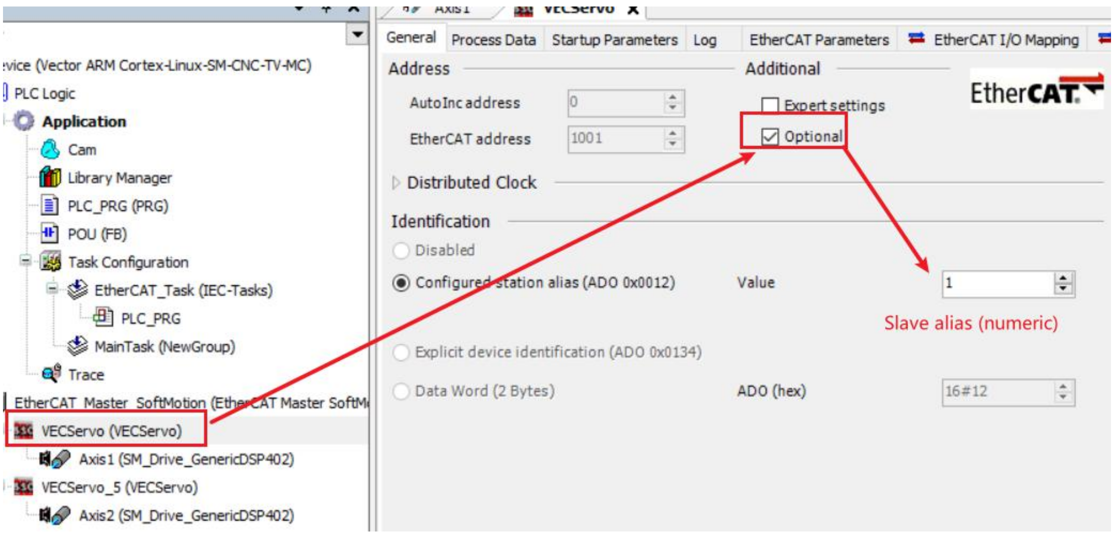
In writing the user program is, by default, the VE controller is in accordance with the EtherCAT from the network cable link order, automatically addressing and addressing, the advantage of this addressing method is that the user does not need to worry about the naming and renaming of the device, just according to the user program bus network configuration, easy for the main controller to check the network configuration, find hardware connection errors. The automatic naming rules that the controller adds to the user program are as follows:



Where the station serial number starts from 1001, according to the order of addition, in turn, the runtime, according to which the servo network cable link order, will be directly connected with the VE controller servo named 1001, in turn to the servo name, the user program, the control function of an axis, give the corresponding serial number of the servo. The point of this search is that the order inwhich EtherCAT network cables are linked must be in the order in which the network configuration isconfigured in the user program.

However, in some applications, some of the functions of the axes have been clearly defined, and there is a fixed name, requiring the VE controller's user program can be addressed according to this fixed name, which requires the user to program, the network from the station addressing method is set to "from the station alias" addressing, and in addition to the servo set the corresponding "from the station alias."

The method from which the station will be set to address by "from the station alias" is as follows:



Set the alias from the station in the servo station.

For example, for the Wykoda bus servo, we can put its "from station number" function code P08 41 The function code is set to 6.

After the user program is configured in this way, regardless of the order of access locationof the alias "6" servo, the servo can be found and given the operating function characteristics of the servo axis in the user program.

Attention:

If some servo axes in the application system are automatically named, the system will first determine the "alias" of the station, the rest of the stations are still automatically named.

5.3.5 Detect the EherCAT communication status

EtherCAT main station status flag bit

1, EtherCAT main station communication status flag bit:

The main station can use the following parameters to determine whether the network is healthy.

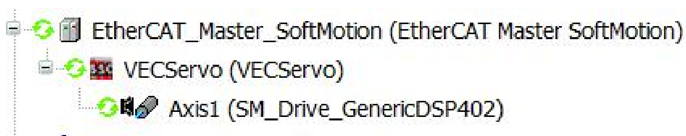
1) xConfigFinished: If this parameter is TRUE, the transfer of all configuration parameters has been completed correctly. The communication is running.

2) xDistributedClockInSync: If a distribution clock is used, the PLC will synchronize with the first EtherCAT from the station that activates the distribution clock setting. As long as synchronization completes successfully, the output is TRUE. xDistributedClockInSync for ON does not guarantee that communication must be completely normal and needs to be judged by xError and from the station state together.

This signal can be used in synchronous mode to activate the SoftMotion function block before the PLC starts, as positional jumps may occur otherwise. When the PLC starts, the output is FALSE, and after a few seconds it becomes TRUE. If synchronization is lost due to any failure, the output is reset to FALSE.

3) xError: Useful for all drop-off stations or communication errors (xError-TRUE). If an error is detected at the start of the EtherCAT stack, or if communication with the from the station is interrupted during operation, the output is TRUE because no messages (e.g. due to a wire outage) can be received. The cause of the error can be understood through a list of errors or error messages.

Example: VE Controller and VECServo



A) Standard bit status when communication is normal:

xConfigFinished= TRUE;

xDistributedClockInSync = TRUE;

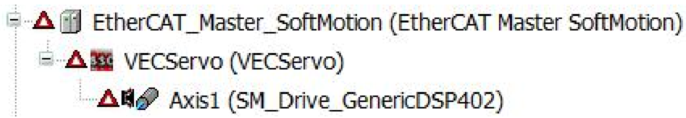
xError= False。

B) There are no from or from the network:

xConfigFinished= False;

xDistributedClockInSync = False;

xError=TRUE。




```
mt1 TRUE :=EtherCAT_Master_SoftMotion.xConfigFinished TRUE ;
mt2 FALSE :=EtherCAT_Master_SoftMotion.xDistributedClockInSync FALSE ;
mt3 FALSE :=EtherCAT_Master_SoftMotion.xError FALSE ;
```

xError does not output True

C) Disconnect the network cable between the primary station and the first from the station when communication is normal, i.e. interrupt all the from the station data

xConfigFinished = TRUE;

xDistributedClockInSync= False;

xError=False。

D) Disconnect the network cable between the first and second from the station when communication is normal, i.e. disconnect all dc-enabled access stations

xConfigFinished = TRUE;

xDistributedClockInSync= False;

xError=False。

E) Disconnect the network cable between the second and last from the station when communication is normal.

xConfigFinished = TRUE;

xDistributedClockInSync= TRUE;

xError=False。

EtherCAT from the station detection

The current state returned from the station, the program should detect the state of the station in real time, motion control is generally considered to be ETC_SLAVE_OPERATIONAL before the commonly used PLCopen instructions can be used to control the axis. The current state of the station is divided into:

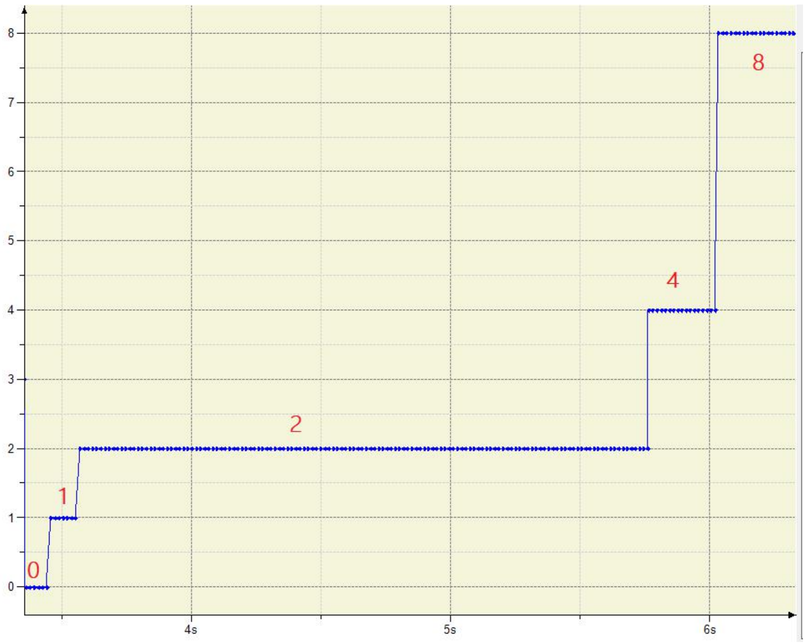
0: ETC_SLAVE_BOOT

1: ETC_SLAVE_INIT

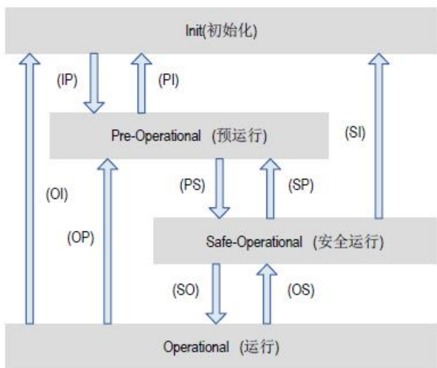
2:ETC_SLAVE_PREOPERATIONAL

4: ETC_SLAVE_SAVEOPERATIONAL

8: ETC_SLAVE_OPERATIONAL



Normal communication automatically switches to the operating state, and the VE controller is initialized after it is stopped. When you convert from an initialization state to a running state, you must convert in the order of Initialization, Pre-Run, Run Safe, and you must not go over the level. You can go through the conversion when you return from the running state. The conversion operation and initialization process of the state



As with the EtherCAT master, each from the station can be considered a function block, and the name of the from the station is an instance of the ETCslave function block, which only needs to be used in the program.

The basic direct judgment is whether the stand is ETC_SLAVE_OPERATIONAL state

Detect whether the from the station is OP mode

```
IF _VECServo.wState<>8 THEN
    bnoOP:=TRUE;
END_IF
```

The above method, if there are dozens from the station, each from the station is judged to need dozens of IF statements, more troublesome. The EtherCAT master provides pointers and lists to the first station, all of which can be found on the chain, so programming can be simplified with the while loop.

Defined:

```
Was
    pSlave: POINTER TO ETCSlave;
    i:  INT;
    iErrorSlave: INT;
END_VAR
```

Programming:

pSlave := EtherCAT_Master_SoftMotion.FirstSlave; // First find the EtherCAT_Master from the main station by using the 19th station.FirstSlave.

WHILE pSlave 0 DO // Call each instance in the 'WHILE' loop to determine the wState and then check the status.

```
    pSlave^();
IF pSlave^.wState = ETC_SLAVE_STATE. ETC_SLAVE_OPERATIONAL THEN
    i:=i+1;
else
    exit;
```

END_IF // by pSlave. NextInstance finds a pointer to the next from the station. At the end of the list the pointer is empty and the loop ends.

```
    pSlave := pSlave^. NextInstance;
END_WHILE
iErrorSlave:=i+1;   // Get the first few station number faults
i:=0;
```

5.4 The MC motion controls the timing of the transmission of the data

The VE controller periodically enters the EtherCAT interrupt according to the user-set EtherCAT cycle, performs a full EtherCAT task, first performs the communication operation between the master and each EtherCAT from the station, and then performs all the POU configured by the user under the task, in the order of the POU in the task configuration table.

The contents of the master's communication operation with each EtherCAT from the station:

(1) Start the EtherCAT bus send operation, send buffer data to the TPDO prepared by the last EtherCAT cycle system, send the data sent to each station in turn, and the communication frame, according to the RPDO configuration, reserves a number of byte gaps that the answer data needs to occupy from the station, in order to get the data back from each station;

The TPDO sends the data in the buffer in the order in which it follows from the station, and the data sent contains normal I/O data and control data of the MC motion control axis

When the number of stations is relatively large, more than the allowed data length of a communication frame, a number of communication frames are used to carry out, if the user program performs SDO read and write operations, the SDO send request is finally sent;

(2) The master resolves the communication return frame, takes out the answer data of each station, analyzes the answer data of the MC from the station axis, updates the data structure such as axis state and position, speed, torque, etc., and determines and updates the execution status indication of the MC function block for the user program to read.

Each time entering EtherCAT interrupts, the axis parameters read by the user program are the data that the system has automatically processed and updated in this link.

5.5 The processing mechanism of the MC function block

5.5.1 Cycle synchronization position mode

Wykoda servo default to the cycle synchronization position mode for control, the so-called "cycle synchronization position mode", that is, the VE controller according to the user wants to arrive from the station axis, allowed operating speed, acceleration, EtherCAT bus cycle and other conditions, in each EtherCAT task execution, by the relevant MC function block to calculate the next cycle point required to reach the location (TargetPosition), sent to the servo drive, and the servo will be based on this distance / time command, the movement to reach the next target point. In this mode of operation, the controller is responsible for planning the location and speed of each point in time for the servo, which only knows the target point and speed to be reached at the next EtherCAT time.

5.5.2 The data structure of the servo axis

In the VE controller, the servo station is managed as a special "axis" and the axis is an important object.

In CODESYS, for each servo axis configured by the user, the system automatically declares a data structure corresponding to the axis at the same time, and automatically updates maintenance in real time when each EtherCAT interrupts the operation;

The following illustration is an example of a monitoring window for Axis, an axis in a user program, whose information is the data structure from that axis.

监视 1

表达式	应用	类型	值	准备值	执行点	地址	注释
⊟ IoConfig_Globals.Axis1	Device.Application	SM3_Drive_...			循环监测		
wAxisStructID		WORD	65042		循环监测		
nAxisState		SMC_AXIS_...	continuous_moti...		循环监测		State of the axis according to the ``PLCopen`` sta...
bRegulatorOn		BOOL	TRUE		循环监测		Parameter number: 1010
bDriveStart		BOOL	TRUE		循环监测		Parameter number: 1011
bCommunication		BOOL	TRUE		循环监测		| ``TRUE``: Communication OK
wCommunicationState		WORD	100		循环监测		Parameter number: 1013
uiDriveInterfaceError		UINT	0		循环监测		| Drive interface error number
bRegulatorRealState		BOOL	TRUE		循环监测		Parameter number: 1015
bDriveStartRealState		BOOL	TRUE		循环监测		Parameter number: 1016
wDriveId		WORD	0		循环监测		Parameter number: 1021
iOwner		INT	1542		循环监测		Parameter number: 1022
iNoOwner		INT	1542		循环监测		Parameter number: 1023
fCycleTimeSpent		LREAL	0.001		循环监测		Parameter number: 1024
fTaskCycle		LREAL	0.001		循环监测		Parameter number: 1025
bError		BOOL	FALSE		循环监测		Parameter number: 1030
dwErrorID		DWORD	0		循环监测		Parameter number: 1031
bErrorAckn		BOOL	FALSE		循环监测		Parameter number: 1032
bDisableErrorLogging		BOOL	FALSE		循环监测		Parameter number: 1036
⊞ fbeFBError		ARRAY [0..g...			循环监测		
dwRatioTechUnitsDenom		DWORD	2097152		循环监测		Parameter number 1051
iRatioTechUnitsNum		DINT	15		循环监测		Parameter number 1052
nDirection		MC_DIRECT...	negative		循环监测		Parameter number 1053
fScalefactor		LREAL	139810.133333...		循环监测		Parameter number 1054
fFactorVel		LREAL	139810.133333...		循环监测		Parameter number 1055
fFactorAcc		LREAL	139810.133333...		循环监测		Parameter number: 1056
fFactorTor		LREAL	787.401574803...		循环监测		Parameter number: 1057
fFactorJerk		LREAL	139810.133333...		循环监测		| Factor jerk
fFactorCur		LREAL	1		循环监测		Parameter number: 1059

监视 1 | 断点

With regard to axis data structure, users need to understand and pay attention to the following characteristics:

◆When the user applies the servo axis configuration of the application network, the system automatically declares the data structure at the same time, the name of the data structure is the same as the name of the axis, and the variable name and data type in the data structure are defined by the system.

◆When there are multiple servo axes in the user project, each axis has its corresponding data structure.

◆If a virtual axis is used in the user program, including the encoder axis, the system will also declare and maintain a corresponding axis data structure for it, but whether some of the structural variables change.

◆Axis data structure variables are global variable types, i.e. they can be accessed directly in all OUes of the user project.

◆As long as the controller computing power meets the requirements of the application system, the number of axes allowed by the system has no clear limit, there is a corresponding number of axis data structure.

◆Once the controller starts to run, during each EtherCAT task run phase, the controller automatically updates the servo's backfly value to the data structure after it picks up the answer data from the station, and the variables of the data structure can be accessed when the user POU is executed.

◆ Axis data variables are specified by: "Data structure name. Structure variable name", as shown in the data structure, we often use the following parameters:

Axis.nAxisState: the current operating state of the axis, servo feedback to the status parameters of the controller;

Axis.fSetPostion: axis setting position, parameters sent by the controller to the servo axis;

Axis.fActPostion: the current actual position of the axis, servo feedback to the status parameters of the controller, the outline and the user program set the same command unit;

Axis.fActVelocity: the current actual speed of the axis, servo feedback to the controller of the status parameters of the same as the user program set the command unit;

In the user program, these variables can be used as the basis for motion control calculation and judgment, some variables in the axis structure are command data sent by the controller to the servo axis, and in the user program, the servo axis can also be controlled by assigning these variables directly. For example, the following ST statement:

Axis.fSetPostion:=500; The units of this parameter are the same as the instruction units

5.5.3 Servo axis status machine and transfer conditions

In motion control systems, the operational state of the axis is divided into several logical states, and the direct transfer of each logical state requires specific conditions, or specified MC operation commands. The advantage of this division processing is that it is easy for the axis to be controlled by motion mode classification, the axis can only be in a logical state at a time, and the transfer of the logical state needs to be carried out according to rules, not due to the wrong trigger of different MC caused by the chaos of operation.

The axis data structure variable (Axis.nAxisState), which indicates the current running state of the axis, axis.nAxisState, is an enumeration variable, common as

The next 8 possible states:

0:Power_off: The shaft is not powered up or enabled, and the command is MC_Power executed

1：Errorstop; ------------------- execute the MC_Reset/MC_Power directive first

2：Stopping; -------------------- waits for the shutdown operation to complete

3：Standstill; ------------------- axis has stopped running and is out of sync

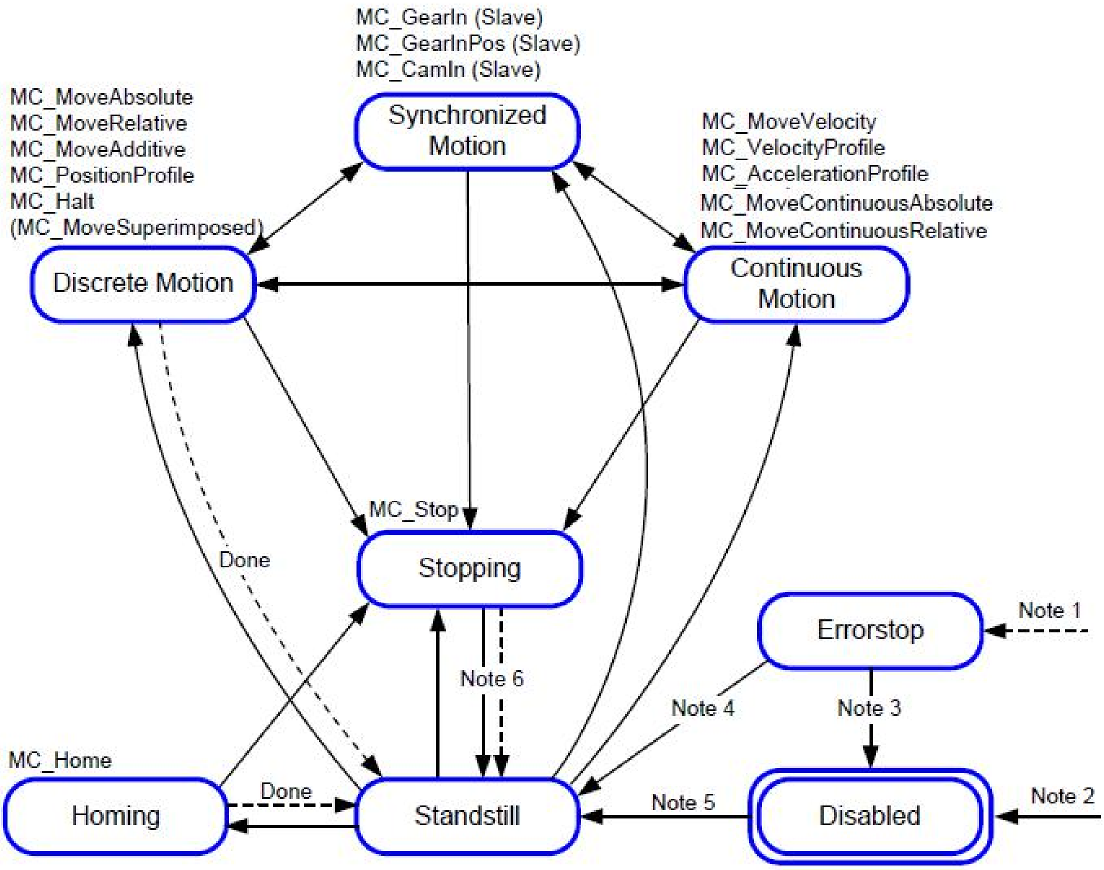
4：Discrete_Motion; --------- axis is discretely running

5：Continuous_Motion; ----- axis is running continuously

6：Synchronized_Motion; --- axis is running in sync

7： Homing; ---------------------- axis is running back to zero, waiting for the zeroing operation to complete

The axis state transfer diagram is as follows, moving from one state to another requires running the corresponding conditions, such as running mc instructions, or an external failure, the user cannot enforce its state, programming must follow the logical requirements, run the relevant instructions:



Note 1: From any state, an error occurred with the axis. .

Note 2: From any state, MC_Power.Enable s FALSE,the axis did not have an error.

Note 3: MC_Reset and MC_Power.Status = FALSE

Note 4: MC_Reset and MC_Power.Status = TRUE and MC_Power.Enable = TRUE

Note 5: MC_Power.Enable = TRUE and MC_Power.Status = TRUE

Note 6: MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

The MC function block in the figure transfers the axis state to the specified state, as can be seen in the figure:

In the axis stop state (Standstill, i.e. Axis.nAxisState 3) can be transferred to various operating states;

Can be transferred from a variety of states to a stop state (Standstill, i.e. Axis.nAxisState, 3),

If an alarm appears on the servo axis (Errorstop, i.e. Axis.nAxisState-1), it is necessary to run the MC_Reset command, the MC_Power command to enter the Standstill state before the shaft can run again;

If you do not use the MC command axis motion according to the above transfer diagram, the shaft will not respond, but get the error warning information of the MC function block;

In user programs, it is sometimes necessary to initiate subsequent control logic based on the state of the axis, which is more accurate and reliable than the one signal judgment of the MC function block, based on Axis.nAxisState.

Be familiar with the transfer conditions of the axonal state diagram above, and pay attention to the logic and order of mc instructions when programming, in order to write a stable and reliable application program.

5.5.4 The execution logic of the MC function block:

Axis control commands related to servo stations appear in the form of MC function blocks (also known as instructions), because MC function blocks require short intervals of sustained execution, need to monitor the servo's operational response in a timely manner, so the MC function blocks related to axis motion can only be called to execute in the EtherCAT task.

The system is handled as follows:

(1) When performing an MC function block, the MC block is effectively triggered before execution, and for multiple instances of the same MC function block (for the same axis object), the principle of priority is given to the trigger;

(2) For the MC control command of the servo axis, check the legality of the operation according to the axis state transfer specification, and then process it, including the axis state transfer, the target parameter update of the axis, and finally prepare the control command data of the shaft;

(3) The system software of the EtherCAT bus control part will send cache data into PDO according to the TPDO configuration table and object dictionary of each user-set station axis

(4) The system software of the EtherCAT bus control section will, according to the user-set RPDO configuration table and object dictionary of each server-obeying station axis, reserve the axis state parameters that the main station needs to read, reserve a number of byte gaps that need to be occupied by the receiving segment of the EtherCAT frame, and finally "group" to the EtherCAT frame to send the cache area, waiting for the next EtherCAT cycle to start, sending to the station;

(5) With regard to the operation results of EtherCAT Remote IO, stored in the cache area in the order of connection from the station rack and sent with the servo station. However, the state of the data for the send cache is updated only after the normal task cycle (task priority is 15 or lower, such as 20ms);

(7) In the user operation control program, if the servo system is in operation, it is necessary to have an MC function block that is triggered to execute in supervising the servo axis, to avoid a running servo axis due to the logic jump of the program, showing a state without MC monitoring, with MC_Stop to stop it, is also a kind of monitoring.

Thus, theoperational commands involved in axis control in the EtherCAT task are not sent out during the current POU execution cycle, but have a delay of one EtherCAT cycle, which in some applications that require precise position and length control, such as the trigger of the MC_CamIn of multi-axis synchronous control, etc., the error caused by this delay needs to be considered.

For the delay error caused by the above reasons, the programming should be handled as follows:

a, 1 EtherCAT cycle in advance to trigger the operation control instructions;

b. The operation control start-up required by the control process is not necessarily exactly at the beginning of the EtherCAT cycle, but at a certain moment in the middle of the cycle, the elimination of this discrete deviation should be taken into account in programming, using the Offset parameters provided by the MC motion control function block to compensate;

(8) Eliminate discrete deviation and estimate the error caused by this communication mechanism based on parameters such as the current object's operating position, speed, acceleration, etc. Reducing the setting of the EtherCAT bus cycle is beneficial to reduce uncontrollable errors.

5.5.5 Data interactions between different priority tasks POU

To have variable access interactions between multiple PVUs, you need to use global variables, which are declared in the GVL global variable list, but if the POU is in a different priority task, the data does not interact in real time, and the update results of the data are related to the task priority and the setting of the task cycle, variable access type, and so on. We need to pay attention to the following mechanisms:

When the user program executes, for tasks of different priorities and cycles, the system adopts the rule of starting time alignment, that is, there is a common start time calculation point of the task cycle, and if the periods of the two tasks are multiplied by integers, then they will have a point in time (alignment point), which is often used as the GVL data interaction point;

Only after the task is completed will the POU modification of the variable be written to the GVL cache, and the modification of the GVL parameters by the low priority task will only take effect at the end of its task cycle

High-priority POU, the rewriting operation of GVL, will take effect immediately;

The GVL value is copied from the GVL cache once before the first task starts at the alignment point, as the basis used during the execution of the POU of this task, and the GVL cache variable is no longer read during the execution of this task;

The servo axis data structure is a global variable automatically defined by the system, and each time the ECT task is performed, the system automatically refreshes the data structure, and if the main task POU reads the variable of the data structure, its reading is also the first ECT task after each "task cycle alignment point" The updated data, the same principle, if the main task POU to modify the data structure of the variable, is the next "task cycle alignment point" after the first ECT task sent to the servo axis, there will be about one Main task cycle lag;

Attention:

Thus, in setting the user program's EtherCAT task and the normal main loop task cycle, should maintain the relationship between the two integers, (e.g. EtherCAT task is 2ms or 4ms, the main loop task is 20ms), so as to avoid the GVL parameter interaction abnormal situation;

In different priority tasks, if there are modifications to the same GVL variable, there may be cases of overlaying each other, programming, it is recommended that a global variable only have one POU override or position operation, and the other POU simply read and reference or reset the operation to avoid unexpected errors.

6 Programming Languages and References

6.1 Data types

CODESYS supports all IEC 61131-3 data types, types that extend IEC 61131-3 and user-defined data types.

IEC 61131-3 data types	Type of extension to IEC 61131-3	User-defined data types
'BOOL' Integer 'REAL' / 'LREAL' 'STRING' 'WSTRING' Time 'LTIME'	'UNION' 'BIT' '__UXINT' and '__XWORD' Reference Pointers	'ARRAY' Structure Enumerations Reference Pointers Subrange Types Identifiers

6.1.1 BOOL Boolean types

Data Type	**Values**	**Memory**
BOOL	TRUE (1), FALSE (0)	8 bit

6.1.2 Integer

CODESYS offers the following integer data types.

Data Type	**Lower Limit**	**Upper Limit**	**Memory**
BYTE	0	255	8 bit
WORD	0	65535	16 bit
DWORD	0	4294967295	32 bit
LWORD	0	$2^{64}-1$	64 bit
SINT	-128	127	8 bit
USINT	0	255	8 bit
INT	-32768	32767	16 bit
UINT	0	65535	16 bit
DINT	-2147483648	2147483647	32 bit
UDINT	0	4294967295	32 bit
LINT	-2^{63}	$2^{63}-1$	64 bit

Data Type	Lower Limit	Upper Limit	Memory
ULINT	0	2^{64}-1	64 bit

6.1.3 REAL/LREAL Floating point type

Data Type	Lower Limit	Upper Limit	Memory
REAL	-3.402823e+38	3.402823E+38	32 bit
LREAL	-1.7976931348623157E+308	1.7976931348623157E+308	64 bit

6.1.4 STRING String type

Variables of the STRING data type can contain any string. The amount of memory allocated during the declaration relates to characters and is shown in brackets or square brackets. If the size is not defined, CODESYS allocates 80 characters by default. Normally, CODESYS does not limit the length of strings. However, string functions can only handle lengths between 1 and 255. If a variable is initialised with a string that is too long for the data type, CODESYS truncates the string accordingly from the right hand side. the memory required for a STRING variable is always one byte per character plus one additional byte (e.g. 81 bytes for a STRING [80] declaration).

Example of a 35-character string declaration.:

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

6.1.5 WSTRING

In contrast to string (ASCII) data types, WSTRING data types are interpreted in Unicode format. As a result of this encoding, the number of WSTRING display characters depends on the characters. A length of 10 means that the length of WSTRING can contain up to 10 words. However, for some characters in Unicode, encoding a character requires more than one WORD, so that the number of characters does not have to correspond to the length of WSTRING (in this case 10). Data types require 1 WORD memory per character, plus 1 WORD extra memory. Only 1 byte per STRING is required. Data type WSTRING to terminate 0.

Cases:

wstr : WSTRING := "This is a WString";

6.1.6 TIME time type

The time data type is internally considered DWORD. TIME and TIME_OF_DAY are resolved in milliseconds, DATE_AND_TIME in milliseconds, and in seconds.

Data type	Lower limit	Upper limit	Storage space	Resolution
TIME	T#0d0h0m0s0ms	T#49d17h2m47s295ms	32 bit	ms
TIME_OF_DAY TOD	TOD#0:0:0.000 00:00:00.000	TOD#23:59:59.999 23:59:59.999	32 bit	ms
DATE	D#1970-1-1 01/01/70	DATE#2106-2-7 February 07, 2106	32 bit	Day
DATE_AND_TIME DT	DT#1979-1-1-00:00:00 01/01/1970 00:00:00	DT#2106-2-7-6:28:15 February 07, 2106 6:28:15	32 bit	Seconds

6.1.7 LTIME

LTIME is used as the time base for the High Resolution Timer. The resolution of the high-resolution timer is measured in nanoseconds.

Data type	Lower limit	Upper limit	Storage space
LTIME	0	213503d23h34m33s709ms551us615ns	64 bit

```
LTIME#<time declaration>
```

The time declaration may include units of time applicable to TIME constants, and

- “us”: microseconds
- “ns”: nanoseconds

example:

```
LTIME1 := LTIME#1000d15h23m12s34ms2us44ns
```

6.1.8 UNION Joint Statement

UNION is a data structure that usually contains different data types.

In a union, all components have the same offset and therefore the same amount of memory. In the joint statement example below, the assignment of name.a also affects the .b.

```
TYPE name:
UNION
    a : LREAL;
    b : LINT;
END_UNION
END_TYPE
```

6.1.9 BIT bit

Only BIT data types can be used for individual variables in a structure or function block. Possible values are TRUE(1) and FALSE(0). A BIT element requires 1 bit of memory. Therefore, you can refer to a single bit of the structure by name. BIT continuously declared elements are bundled together in bytes. In this way, you can optimize how memory is used instead of using the BOOL type, which retains 8 bits per type. Bit access, on the other hand, is significantly more time-consuming. Therefore,BIT should only use data types if data needs to be defined in a predefined format.

6.1.10 __UXIN and __XWORD are pseudo-data types

CODESYS supports systems with 32-bit and 64-bit-wide address registers. To make the IEC code as independent as possible from the target system, use pseudo-__UXINT and __XWORD. The compiler checks which target system types are up-to-date, and then converts these data types to the appropriate standard data types.

__UXINT converted to ULINT on a 64-bit platform and UDINT on a 32-bit platform.

__XWORD converted to LWORD on a 64-bit platform and DWORD on a 32-bit platform.

6.1.11 POINTERS pointer

The syntax declaration of the pointer

Pointers store the addresses of variables, programs, function blocks, methods, and functions while the application is running. The pointer points to one of the objects mentioned or a variable of any data type. The syntax of the pointer declaration:

<identifier>: POINTER TO <data type | function block | program | method | function>;

When you de-reference a pointer, the value of the address to which the pointer points is determined. In order to de-reference the pointer, attach the content operator to the pointer identifier (see the example below pt.

Using the address operator ADR, youcan assign the address of a variable to a pointer.

```
Was
    pt：POINT TO INT; (Declaration of pointer pt)
    var_int1：INT：= 5; (Declarations of var_int1 and var_int2 variables)
    var_int2：INT;
END_VAR
```

pt：= ADR（var_int1）; (The address that the pointer ptisassigned tovar s int1).
var_int2：= pt ^; (The value 5 of the value of the var_int1 is assigned to the variable by canceling the reference var_int2 pt)

Attention:

If a pointer to the device input is used, the access (e.g.,"pTest: . ... invalid assignment target") is considered a write access. This causes the compiler to warn when the code is generated. If this construct is required, the input value (input) must first be copied to a variable with write access. In online mode, you can jump from the pointer to the declared position of the reference variable by clicking the Go to Reference command.

A function pointer to an external function

CODESYS supports function pointers that replace the INDEXOF operator and can be passed to an external library. However, CODESYS does not provide any way to call function pointers from within the application in the development system. The runtime system function used to register callback functions (system library functions) requires a function pointer. Depending on which callback is registered, the runtime system implicitly calls related functions (for example, in the case of STOP). In order for such a system call (runtime system) to be possible, the appropriate object properties must be set in the Build tab.

You can use the ADR operator for functions, programs, function blocks, and methods. CODESYS outputs the address of the pointer to the function, not the address of the function, because the value of the function can be changed after it is changed online. This address is valid as long as the feature exists on the target system.

The index access pointer

At CODESYS, index access to the " input POINTER,STRING, and WSTRING variables is allowed.

Pint s i returns the basic data type

- Index access to pointers is done arithmetically: if index access is used for pointER TO variables, CODESYS calculates the offset pint s (pint s i s sizeOF (base type)). Index access also causes the pointer to be implicitly de-referenced. The resulting data type is the basic data type of the pointer. Note that pint s7 s
- When index access is used for variables of type type, STRING gets characters at the offset of the index expression. The result is a BYTE type. The first character of the string is returnedbystr (i) in SINT (ASCII).
- When index access is used for variables of type type, WSTRING gets characters at the offset of the index expression. The result is a WORD type. wstr (i) returns the firstcharacter of the string inINT (Unicode).

Attention:

1. DWORD When a pointer is a 64-bit pointer, even on a 64-bit platform, the difference between the two pointers results in a type.

2. You can use a reference to a value that is directly controlled compared to a pointer.

3. Memory access to pointers can be monitored at runtime through the implicit monitoring feature CheckPointer.

6.1.12 REFERENCE Reference

REFERENCE is also a pointer, but it has some advantages over POINTER:

- Easy to use: You don't have to explicitly de-reference a reference (using s) to access the contents of the reference object.
- Better syntax for passing values: If the input is REFERENCE TO, a (refInput: s value) does not have to write ADR explicitly.
- Type safety: Unlike pointers, for references, the compiler checks that the base type is the same when assigning two references

References can be declared in the following syntax:

```
<identifier> : REFERENCE TO <data type>
```

```
A : REFERENCE TO DUT;
B : DUT;
C: DUT;

A REF= B; // 对应于 A := ADR(B);
A: C; 对应于 A-
```

To check for a valid reference, you can use the operator __ISVALIDREF to check that the reference points to a valid value, that is, a value that is not equal to 0. Language:

```
<Boolean variable> := __ISVALIDREF(<with REFERENCE TO <data type> declared identifier);
```

When the reference points to a valid value, the .lt; Boolean variable is TRUE; otherwise ITSE. Example:

```
ivar : INT;
ref_int : REFERENCE TO INT;
ref_int0 : REFERENCE TO INT;
testref : BOOL := FALSE;
```

How to do it:

```
ivar：= ivar + 1;
ref_int REF = ivar;
ref_int0 REF = 0;
testref：= __ISVALIDREF（ref_int）; (s true becauseref_int point to ivars that are not zero).
testref：= __ISVALIDREF（ref_int0）; (. . . falsebecausethe ref_int0 set to 0 ..
```

6.1.13 ARRAY array

An array is a collection of data elements of the same data type. CODESYS supports one-dimensional and multi-dimensional arrays of fixed or variable lengths. Array types are: fixed-length arrays, array arrays, and variable-length arrays, which can be defined in the declaration section of the POU or in the list of global variables.

An array of fixed lengths

The syntax of a one-dimensional array declaration:

```
<variable name> :   ARRAY[ <dimension> ] OF <data type> ( := <initialization> )? ;
<dimension> :   <lower index bound>..<upper index bound>
<data type> : elementary data types | user defined data types | function block types
// (...)? : Optional
```

The syntax of the two-dimensional array declaration:

```
<variable name> :   ARRAY[ <1st dimension> ( , <next dimension> )+ ] OF <data type> ( :=
 <initialization> )? ;

<1st dimension> :   <1st lower index bound>..<1st upper index bound>
<next dimension> : <next lower index bound>..<next upper index bound>
<data type> : elementary data types | user defined data types | function block types
// (...)+ : One or more further dimensions
// (...)? : Optional
```

The index limit is an integer;
Data access syntax:

```
<variable name>[ <index of 1st dimension> ( , <index of next dimension> )* ]
// (...)* : 0, one or more further dimensions
```

Example One:
One-dimensional array of 10 integer elements defined:

```
Was
aiCounter:ARRAY[0..9] OF INT; //Index Lower Limit:0 , Index Upper Limit:9
END_VAR
```

Program:

```
aiCounter：ARRAY[0..9]：= [0，10，20，30，40，50，60，70，80，90];  //初始化：
iLocalVariable：= aiCounter [2];     // Data access
```

The value 20 in the array is assigned to the local variable iLocalVariable.

Example 2:
Two-dimensional array Definition:

```
Was
aiCardGame：ARRAY [1..2，3..4] OF INT;    //1D: 1 to 2,   2D: 3 to 4
END_VAR
```

Initialization program:

```
aiCardGame：ARRAY [1..2，3..4] OF INT：= [2（10），2（20）]; //A short history of [10,10,20, 20]
```

	1	2
3	10	10
4	20	20

Data Access Program:

```
iLocal_1：= aiCardGame [1，3];     // Assign 10
iLocal_2：= aiCardGame [2，4];     // Allocate 20
```

Example three:

3D array definition:

```
Was
    aiCardGame：ARRAY [1..2，3..4，5..6] OF INT;
END_VAR
```

1st dimensional: 1 to 2

2nd dimensional: 3 to 4

3D: 5 to 6

A totalof: 2 x 2 x 2 x 8 array elements

Initialize 1:

```
aiCardGame：ARRAY [1..2，3..4，5..6] OF INT：= [10，20，30，40，50，60，70，80];
```

Data Access 1:

```
iLocal_1：= aiCardGame [1，3，5];     // Assign 10
iLocal_2：= aiCardGame [2，3，5];     // Allocate 20
iLocal_3：= aiCardGame [1，4，5];     // Allocate 30
iLocal_4：= aiCardGame [2，4，5];     // Allocate 40
iLocal_5：= aiCardGame [1，3，6];     // Allocate 50
iLocal_6：= aiCardGame [2，3，6];     // Assign 60
iLocal_7：= aiCardGame [1，4，6];     // Assign 70
iLocal_8：= aiCardGame [2，4，6];     // Allocate 80
```

Initialize 2:

```
aiCardGame：ARRAY [1..2，3..4，5..6] OF INT：= [2（10），2（20），2（30），2（40）];
//Short for [ 10, 10,20,20,30,30,40 ]
```

Data Access 2:

```
iLocal_1：= aiCardGame [1，3，5]; // Assign 10
iLocal_2：= aiCardGame [2，3，5]; // Assign 10
iLocal_3：= aiCardGame [1，4，5]; // Allocate 20
iLocal_4：= aiCardGame [2，4，5]; // Allocate 20
iLocal_5：= aiCardGame [1，3，6]; // Allocate 30
iLocal_6：= aiCardGame [2，3，6]; // Allocate 30
iLocal_7：= aiCardGame [1，4，6]; // Allocate 40
iLocal_8：= aiCardGame [2，4，6]; // Allocate 40
```

Example Four：

3-dimensional arrays of user-defined structures：

```
TYPE DATA_A
```

```
STRUCT
 iA_1 : INT;
 iA_2 : INT;
 dwA_3 : DWORD;
END_STRUCT
END_TYPE

PROGRAM PLC_PRG
VAR
    aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A;
END_VAR
```

The array aData_A consists of a total of 3 * 3 * 10 = 90 array elements of the data type DATA_A.

Partial initialisation:

```
aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A := [(iA_1 := 1, iA_2 := 10, dwA_3 := 16#00FF),(iA_1 := 2,
iA_2 := 20, dwA_3 := 16#FF00),(iA_1 := 3, iA_2 := 30, dwA_3 := 16#FFFF)];
```

In this example, only the first three elements are explicitly initialised. Elements that are not explicitly assigned an initialisation value are initialised internally using the default value of the base data type. This will begin with the element aData_A[2, 1, 1] as the initialised 0 structure component.

Data access

```
iLocal_1: = aData_A [1,1,1] .iA_1;     // Allocation 1
dwLocal_2: = aData_A [3,1,1] .dwA_3;     // Allocation 16＃FFFF
```

An array of arrays

Declaring"array of arrays" is another syntax for multi-dimensional arrays. The collection of nested elements rather than the dimensions of the dimensions of the dimensions of the dimensions. The depth of nesting is infinite.

Array array declaration syntax:

```
<variable name> :   ARRAY[<first>] ( OF ARRAY[<next>] )+ OF <data type> ( := <initialization> )? ;

<first> :   <first lower index bound>..<first upper index bound>
<next> : <lower index bound>..<upper index bound> // one or more arrays
<data type> : elementary data types | user defined data types | function block types
// (...)+ : One or more further arrays
// (...)? : Optional
```

Data access syntax:

```
<variable name>[<index of first array>] ( [<index of next array>] )+ ;
// (...)* : 0, one or more further arrays
```

Example One:

```
PROGRAM PLC_PRG
```

```
VAR
    aiPoints : ARRAY[1..2,1..3] OF INT := [1,2,3,4,5,6];
    ai2Boxes : ARRAY[1..2] OF ARRAY[1..3] OF INT := [ [1, 2, 3], [ 4, 5, 6]];
    ai3Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF INT := [ [ [1, 2, 3, 4], [5, 6, 7, 8 ], [9, 10,
11, 12] ], [ [13, 14, 15, 16], [ 17, 18, 19, 20], [21, 22, 23, 24] ] ];
    ai4Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF ARRAY[1..5] OF INT;
END_VAR

aiPoints[1, 2] := 1200;
ai2Boxes[1][2] := 1200;
```

The variables aiPoints and ai2Boxes collect the same data elements, but the syntax of the declaration differs from that of the data access.

aiPoints	ARRAY [1..2, 1..3] OF INT	
aiPoints[1, 1]	INT	1
aiPoints[1, 2]	INT	1200
aiPoints[1, 3]	INT	3
aiPoints[2, 1]	INT	4
aiPoints[2, 2]	INT	5
aiPoints[2, 3]	INT	6

ai2Boxes	ARRAY [1..2] OF ARRAY [1..3] OF INT	
ai2Boxes[1]	ARRAY [1..3] OF INT	
ai2Boxes[1][1]	INT	1
ai2Boxes[1][2]	INT	1200
ai2Boxes[1][3]	INT	3
ai2Boxes[2]	ARRAY [1..3] OF INT	
ai2Boxes[2][1]	INT	4
ai2Boxes[2][2]	INT	5
ai2Boxes[2][3]	INT	6

An array of variable lengths

In a function block, function, or method, you can declare an array VAR_IN_OUT variable length in the declaration section of the file. the LOWER_BOUND and UPPER_BOUND operators provide an index range to determine which arrays are actually used at runtime.

Variable length The syntax declared by a one-dimensional array:

```
<variable name> :   ARRAY[*] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function block types
// (...)? : Optional
```

A variable-length multi-dimensional array declares the syntax of

```
<variable name> :   ARRAY[* ( , * )+ ] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function block types
// (...)+ : One or more further dimensions
```

```
// (...)? : Optional
```

Syntax of operators for calculating limit indices

```
LOWER_BOUND( <variable name> , <dimension number> )
UPPER_BOUND( <variable name> , <dimension number> )
```

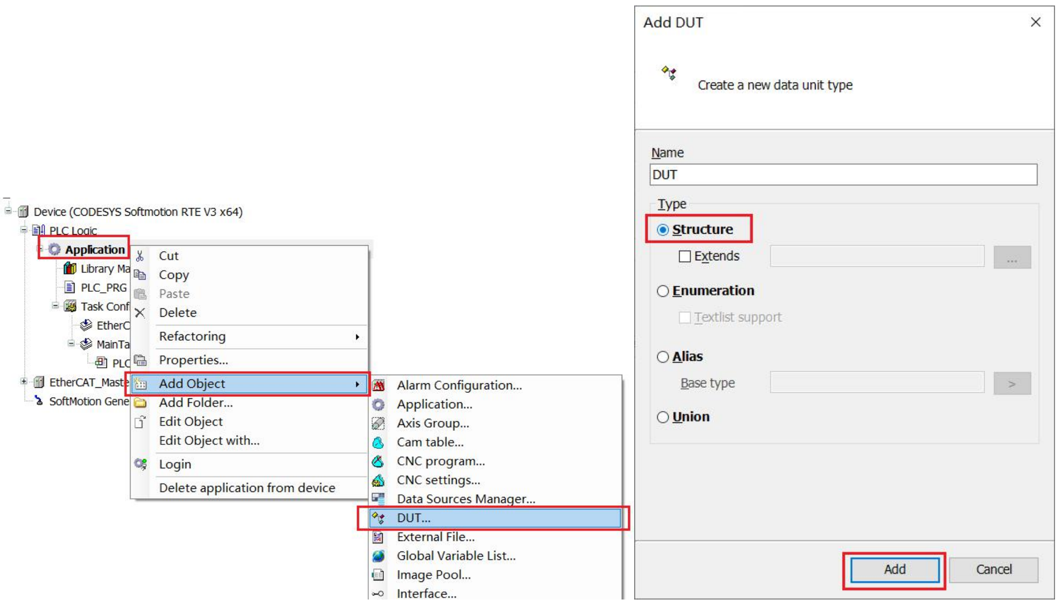
Example One:

This SUM function adds up the integer values of the array elements and returns the calculated sum as the result. The sum is calculated over all the array elements available at runtime. As the actual number of array elements is only known at runtime, the local variables are declared as variable length one-dimensional arrays.

```
FUNCTION SUM: INT;
VAR_IN_OUT
    aiData : ARRAY[*] OF INT;
END_VAR
VAR
  diCounter, diResult : DINT;
END_VAR

diResult := 0;
FOR diCounter := LOWER_BOUND(aiData, 1) TO UPPER_BOUND(aiData, 1) DO // Calculates the length of the current array
  diResult := diResult + A[i];
END_FOR;
SUM := diResult;
```

6.1.14 Structure structure

Create a structure in a project that has aDUTobject by clicking Add Objects.



The structure declares the keywords TYPE and STRUCT at the beginning, and is associated with both END_STRUCT and END_TYPE.

The syntax of the structure declaration:

```
TYPE <structure name>:
STRUCT
    <variable declaration 1>
    ...
    <variable declaration n>
END_STRUCT
END_TYPE
```

<structure name> is a type that CODESYS can recognise as a whole item and can be used as a standard data type. Nested structures can also be used. The only restriction is that you are not allowed to assign addresses to variables (as AT declarations are not allowed).

Example One:

```
TYPE polygonline:
STRUCT
    start:ARRAY [1..2] OF INT;
    point1:ARRAY [1..2] OF INT;
    point2:ARRAY [1..2] OF INT;
    point3:ARRAY [1..2] OF INT;
    point4:ARRAY [1..2] OF INT;
    end:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Initializing the structure

Example:

```
pPoly_1 : polygonline := ( start:=[3,3], point1:=[5,2], point2 := [7,3], point3 := [8,5], point4 := [5,7], end :=
[3,5]);
```

Initialisation with variables is not allowed.

Accessing structure members

Structure members can be accessed using the following syntax:

`<structure name>.<component name>`

Thus, the start component polygonline of the structure can be accessed using poly_1.start in the above example.

Bit access in structures

Bit is a special data type defined only in structures. It reserves one memory bit and allows the use of names to address individual bits of the structure.

```
TYPE <structure name>:
STRUCT
    <bit name bit1> : BIT;
    <bit name bit2> : BIT;
    <bit name bit3> : BIT;
    ...
    <bit name bitn> : BIT;
END_STRUCT
END_TYPE
```

BIT structure members can be accessed using the following syntax:

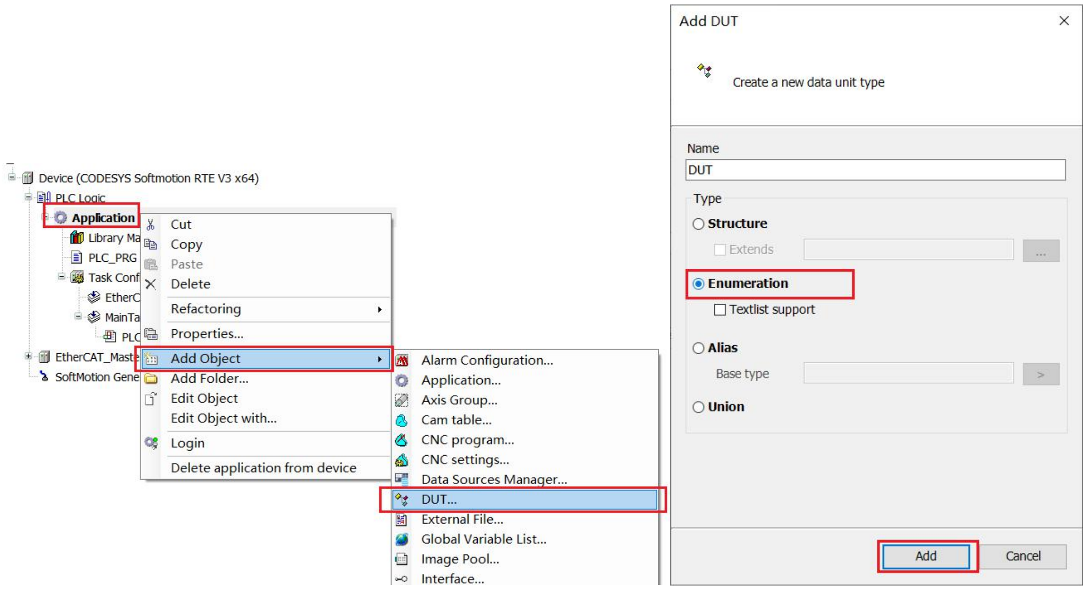
```
<structure name>.<bit name>
```

6.1.15 Enumerations

Enumeration is a user-defined data type consisting of a series of comma-separated components (enumeration values) that declare user-defined variables. In addition, you can use enumered components, such as constants, whose identifiers are globally recognized in the project.

```
<enumeration name>.<component name>
```

You can declare an enumerated in a DUT object and create itinyour project by clicking AddObject.



Declaration syntax:

```
( {attribute 'strict'} )? // Pragma optional but recommended
TYPE <enumeration name> :
```

```
(
    <first component declaration>,
    ( <component declaration> ,)+
    <last component declaration>
)( <basic data type> )? ( := <default variable initialization> )? ;
END_TYPE

( ... )? : Optional
<component declaration> : <component name> ( := <component initialization> )?
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT | ULINT | BYTE | WORD | DWORD |
LWORD
<variable initialization> : <one of the component names>
```

In enumerumered declarations, at least two components are typically declared. However, you can declare as many as you want. Each component can be assigned its own initialization. Enumeration automatically has a base data type INT,but you can specify additional base data types. In addition, you can specify a component in a declaration and then use that component to initialize enumerated variables.

The use of the term 'attribute 'strict') causes the rigorous type test described below.

Example 1:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE COLOR_BASIC :
(
    yellow,
    green,
    blue,
    black
)
;    // The base data type is INT,and COLOR_BASIC the default initialization of all variables is yellow
END_TYPE
```

6.1.16 Subrange Types

A sub-range type is a data type whose value range is a subset of the base type.

The syntax of the declaration:

`<name> : <int type> (<lower limit>..<upper limit>);`

<name>	Valid IEC identifiers
<int type>	Sub-range data types （SINT，USINT，INT，UINT，DINT，UDINT，BYTE，WORD，DWORD，LINT，ULINT，LWORD）。

<lower limit>	Lower limit of the range: constants that must be compatible with the base data type. The lower limit is also included in the scope.
<upper limit>	Upper limit of the range: constants that must be compatible with the base data type. The upper limit is also included in the range.

Example One：

```
VAR
 i：INT（-4095..4095）；
 ui：UINT（0..10000）；
END_VAR
```

CODESYS issues an error message if the value assigned to a subrange type in the declaration or implementation section is not in the range (e.g. i: = 5000). In runtime mode, the implicit monitoring function CheckRangeSigned and the range restriction CheckRangeUnsigned, which monitors subrange types, can be used.

6.2 Variable

6.2.1 Local variable -VAR

Local variables are declared in the declaration section END_VAR between the VARs and the programming object. You can use instance paths for read-only access to local variables, or you can extend local variables using the attribute keyword.

Example:

```
Was
    iVar1：INT；
END_VAR
```

6.2.2 Enter the variable - VAR_INPUT

The input variable is used for the input of the function block.

VAR_INPUT variable is declared in VAR_INPUT declaration END_VAR the programming object between the keyword and the target. You can use the attribute keyword extension to enter variables.

Example:

```
VAR_INPUT
iIn1：INT;   (*the first input variable*)
END_VAR
```

6.2.3 Output variable - VAR_OUTPUT

The output variable is used for the output of the function block.

VAR_OUTPUT variable is declared in VAR_OUTPUT declaration END_VAR the programming object between the keyword and the target. CODESYS returns the value of this variable to the calling POU. You can retrieve values there and continue to use them, or you can extend the output variables using the attribute keyword.

Example：

```
VAR_OUPUT
    iOut1：INT;   (* First output variable *)
END_VAR
```

Output variables in functions and methods.

Functions and methods have additional outputs according to the IEC 61131-3 standard. These additional outputs must be assigned when the function is called, as shown below.

Example：

```
fun（iIn1：= 1，iIn2：= 2，iOut1 => iLoc1，iOut2 => iLoc2）；
```

6.2.4 Input and output variables -VAR_IN_OUT

VAR_IN_OUT variable is an input/output variable that is part of the POU interface and passes the parameter as a formal reference. Input and output variables FUNCTION_BLOCK declared in the VAR_IN_OUT of PRG, method, METHOD, or FUNCTION.

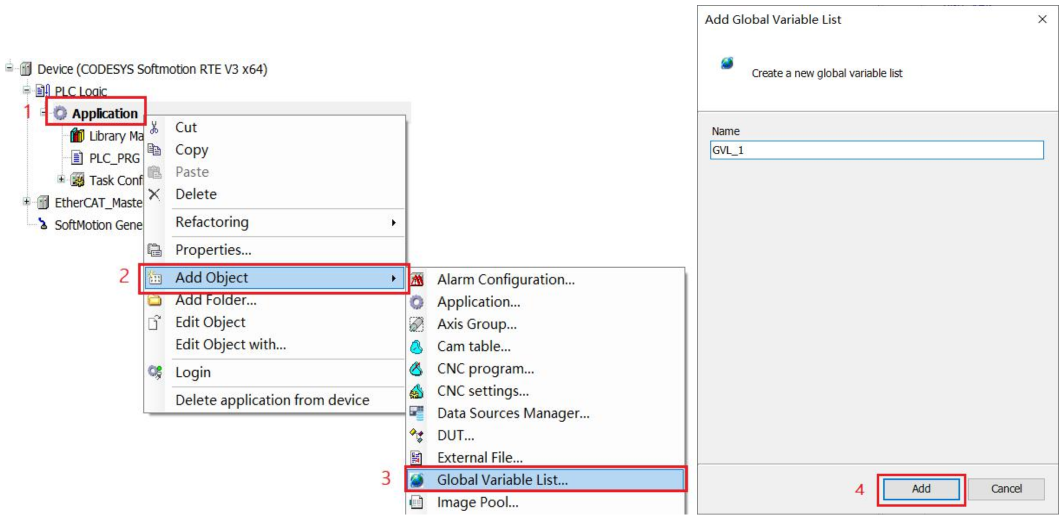
Declaration syntax:

```
<keyword> <POU name>
VAR_IN_OUT
    <variable name> : <data type> ( := <initialization value> )? ;
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG
```

6.2.5 Global variable - VAR_GLOBAL

Global variables are ordinary, constant, external, or remaining variables that can be identified throughout the project.

Global variables can be declared in the list of global VAR_GLOBAL or in the declaration END_VAR of a programming object between the keywords and and .



When you add a point before a variable name, such as .iGlobVar1,a global variable is recognized.

Attention:

If the local variable declared in the block has the same name as the global variable, it has priority in the block. CODESYS always initializes global variables before local POU variables

Example One:

```
VAR_GLOBAL
    iVarGlob1: INT;
END_VAR
```

6.2.6 Temporary variable - VAR_TEMP

The Temporary Variables feature is an extension of the IEC 61131-3 standard.

You can declare VAR_TEMP locally END_VAR between the keywords and the relevant keywords. VAR_TEMP declaration is only available in program blocks and function blocks. CodeSYS initializes the temporary variable each time the block is called. Applications can only access temporary variables in the implementation part of a block or function block.

```
VAR_TEMP
iVarTmp1:INT;(the first temporary variable
END_VAR
```

6.2.7 Static variable - VAR_STAT

This feature is an extension of the IEC 61131-3 standard.

You can declare VAR_STAT locally END_VAR between the keywords and and the key. CodeSYS initializes static variables the first time each block is called. Static variables can only be accessed from the namespace where the variable is declared. However, when the application leaves the block, the static variable retains its value. For example, you can use static variables as counters for function calls. Static variables can be extended using the attribute keyword.

Example 1:

```
VAR_STAT
    iVarStat1: INT;
END_VAR
```

6.2.8 External variable - VAR_EXTERNAL

An external variable is a global variable that is imported into a block.

You can declare VAR_EXTERNAL variables between the keywords and END_VAR. If the global variable does not exist, an error message is displayed.

Attention:

CODESYS does not require you to declare a global variable as an external variable to use it in the POU. This keyword is used only to maintain compliance with IEC 61131-3.

Language:

```
<POU keyword> <POU name>
VAR_EXTERNAL
    <variable name> : <data type>;
END_VAR
```

This variable does not allow initialization.

Example:

```
FUNCTION_BLOCK FB_DoSomething
VAR_EXTERNAL
    iVarExt1: INT; (*First variable*)
END_VAR
```

6.2.9 Instance variable - VAR_INST

CODESYS does VAR_INST method variables in the method stack, but in the stack of feature block instances. This means VAR_INST functions like other variables in the function block instance and is not reinitialized each time the method is called.

VAR_INST only allow variables in methods, and they can only be accessed within the method. Monitor the variable values of instance variables in the declaration section of the method.

You can extend instance variables using the attribute keyword.

Example One:

```
Method meth_last: INT
VAR_INPUT
  iVar: INT;
END_VAR
VAR_INST
  iLast: INT: = 0;
END_VAR
meth_last: = iLast;
iLast: = iVar;
```

6.2.10 Configuration variable - VAR_CONFIG

Use configuration variables to assign full addresses to variables with incomplete addresses declared in the function block, which are mapped to device I/O. Declare VAR_CONFIG in END_VAR list of global variables between and . The list of global variables is called a "variable configuration" in which you can type configuration variables with the full instance path and the correct address.

Example 1:

Declare the variable %I,which is not complete in the function block, is :

```
FUNCTION_BLOCK locio
Was
    xLocIn AT%I *：BOOL：= TRUE;
END_VAR
```

The locio function block is used PLC_PRG program:

```
PROGRAM   PLC_PRG
Was
    locioVar1：locio;
END_VAR
```

The correct variable configuration in the list of global variables is as follows:

```
VAR_CONFIG
  PLC_PRG.locioVar1.xLocIn AT%IX1.0：BOOL;
END_VAR
```

6.2.11 Constant variable - VAR CONSTANT

Constant variables are declared in the list of global variables or in the declaration section of a programming object. In an implementation, constant variables can be accessed as read-only through the instance path.

Language:

```
<scope> CONSTANT
    <identifier> : <data type> := <initialization> ;
END_VAR

<scope> : VAR | VAR_INPUT | VAR_STAT | VAR_GLOBAL
```

When you declare a constant variable, you always assign an initialization value. So you can't write constants anymore.

Example 1:

Statement

```
VAR CONSTANT
    c_rTAXFACTOR : REAL := 1.19;
```

```
END_VAR
```

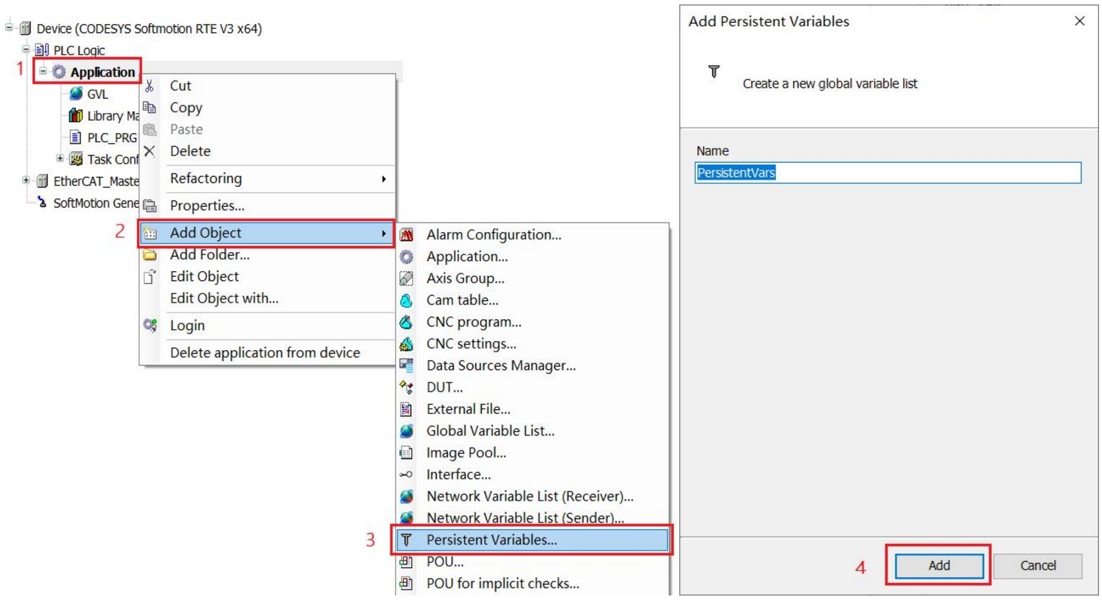
Call

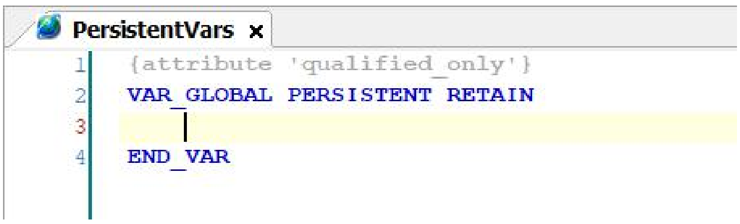
```
rPrice := rValue * c_rTAXFACTOR;
```

In an implementation, you can access constant variables only in a read-only manner. The constant variable is to the right of the assignment operator.

6.2.12 Persistence variable -PERSISTENT

Persistent variables are declared in the VAR_GLOBAL RETAIN PERSISTENT declaration section of the persistent global VAR_GLOBAL list. For variables marked with keywords outside the persistence editor, the instance path is added to it.





A variable declares PERSISTENT RETAIN withthe same effect as RETAIN PERSISTENT or PERSISTENT.

The syntax persistentVars declared in the list of global persistent variables:

```
VAR_GLOBAL PERSISTENT RETAIN
    <identifier>: <data type> (:= <initialization>)?;
 <instance path to POU variable>
END_VAR
```

The syntax declared in the POU:

```
<scope> PERSISTENT RETAIN
    <identifier>: <data type> ( := <initialization> )?;        // ( ... )? : Optional
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```

Attention:

1. Never use pointer TO data types in a persistent variable list. If you download the application again, its address may change.

2. The AT keyword is not allowed to be used to assign input, output, or memory addresses.

3. If you frequently change the name or data type of the remaining variables, it is best for RETAIN to declare them as reserved variables only using keywords.

4. There are two ways to declare: a, declare variables directly in the list of persistent variables, and avoid inserting instance paths. b, declare locally in the program or function block, and add the instance path in the list of persistent variables (here's how). Both methods, which use twice as much memory, also increase cycle time.

Declare directly in the list of permanent global variables	This variable is persistent and is located in a protected memory area.
Local declarations in programs with instance paths in the list of persistent variables Local declarations in function blocks with instance paths in the list of persistent variables	The variable is persistent and is located in a protected memory area and in memory (double allocation).
Local to the program only Local in function blocks only	This variable is not persistent. A warning is displayed in the message window. Tip: "Click on Declaration ‣ Add all instance paths" to import the variable into the list of persistent variables.
Local functions	This declaration has no effect. This variable is not persistent.

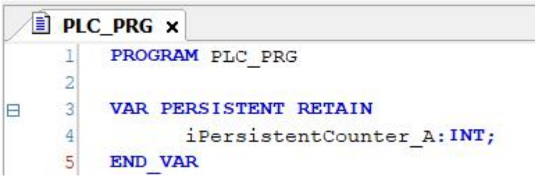
Method a:

Declaration of PersistentVars in the list of persistent variables:

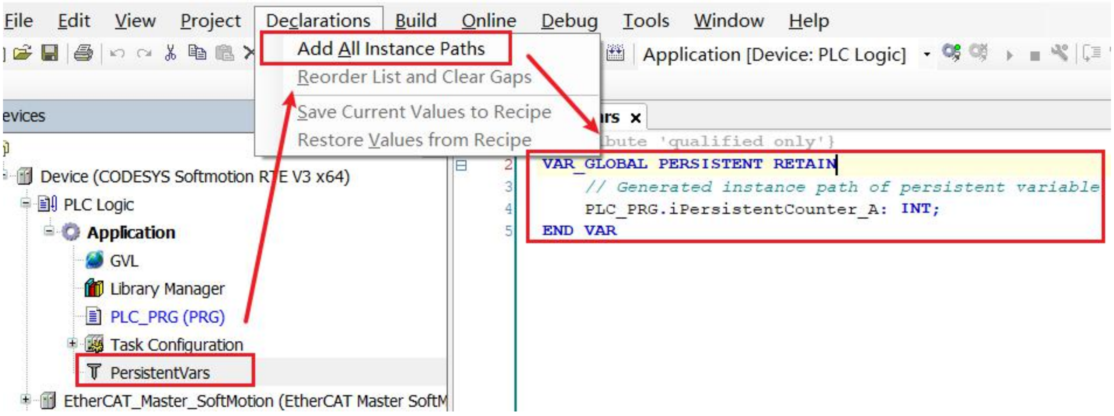
```
{attribute'qualified_only'}
    VAR_GLOBAL
PERSISTENT RETAIN g_iCounter: INT;
    // Generated persistent variables
    PLC_PRG.fb_A.iPersistentCounter_A: INT;
END_VAR
```

Method b:

Declare locally or in a function block:



Add an instance path to the list of persistent variables



Avoid, as far as possible, the variable PERSISTENT declared in the function blockby atag. This is because the function block instance is stored entirely in the remaining memory, not just the tagged variable.

6.2.13 Reserved variable - RETAIN

The declaration of a reserved variable is in the keyword PRESERVE,declaring thescope:VAR,VAR_INPUT,VAR_OUTPUT,VAR_IN_OUT,VAR_STAT, or VAR_GLOBAL.

Declaration syntax:

```
<scope> RETAIN
    <identifier>: <data type> ( := <initialization> )? // ( ... )? : Optional
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```

Attention:

The use of AT key words to assign input, output, or memory addresses is not allowed.

The declared area：

In program local	Only the variables are located in the reserved storage area.
Global in the list of global variables	Only the variables are located in the reserved storage area.
Local in function block	The entire instance of a function block and all its data are located in the reserved memory. Only the declared reserved variables are protected.
Local functions	Even the variable is not located in the reserved storage area. This

	statement has no effect.
Local and persistent operation	Even if the variable is not located in a reserved storage area. The statement has no effect.

Avoid using variables from the RETAIN marker function block where possible.

6.2.14 Special variables -SUPER

SUPER is a special variable for use in object-oriented programming.

SUPER is a pointer to a function block, which points to the basic function block instance from which the function block is generated. The pointer therefore also allows access to the implementation of the methods of the basic function block (basic class).The SUPER pointer is automatically available for each function block.SUPER can only be used within the methods and the associated function block implementation.

Dereferencing of pointers: SUPER^

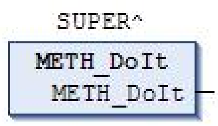
Use of the SUPER pointer: With the keyword SUPER it is possible to call methods that are valid in the base class or in an instance of the parent class.

Example1:

ST:

```
SUPER^.METH_DoIt();
```

FBD / CFC / LD:



6.3 Operators

CODESYS supports all IEC-61131-3 operators. These operators are implicitly recognised throughout the project. In addition to these IEC operators, CODESYS also supports certain non-IEC 61131-3 standard operators.

Operators are used in blocks such as function blocks. They include arithmetic operators, bit string operators, bit shift operators, selection operators, comparison operators, address operators, call operators, type conversion operators, numerical operators, namespace operators, multicore operators, other operators, etc.

Arithmetic Operators

" ADD"
" SUB"
" MUL"

“ DIV”
'MOD'
“ MOVE”
“ INDEXOF”
“ SIZEOF”

Bit String Operators

“ AND”
“ OR”
“ XOR”
“ NOT”
“ AND_THEN”
“ OR_ELSE”

Shift Operators

“ SHL”
“ SHR”
“ ROL”
“ ROR”

Selection Operators

“ SEL”
“ MAX”
“ MIN”
“ LIMIT”
“ MUX”

Comparison Operators

'GT'
“ LT”
“ LE”
'GE'
“ EQ”
'NE'

Address Operators

“ ADR”

Content Operators

“ BITADR”

Call Operators

“ CAL”

Type conversion operators

Implicit conversion from a larger type to a smaller type is not possible (for example, from INT to BYTE or from DINT to WORD). A special type conversion must be used to convert a larger type to a smaller type. Normally, you can convert any basic type to any other basic type.

Type conversion: `<elementary type1>_TO_<elementary type2>`

Overflow conversion: `TO_<elementary type2>`

Overflow conversion of numeric arithmetic symbols

“ ABS”
“ SQRT”
“ LN”
“ LOG”
“ EXP”
“ EXPT”
“ SIN”
“ ASIN”
'COS'
“ TAN”
'ACOS'
“ ATAN”

6.3.1 Arithmetic operator

Add "ADD" to the operation

The IEC operator is used to add variables.

Allowed data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, TIME_OF_DAY (TOD)DATEDATE_AND_TIME(DT)

Possible combinations of TIME data types:TIME-TIME-TIME, TOD-TIME-TOD, DT-TIME,DT-TIME, DT

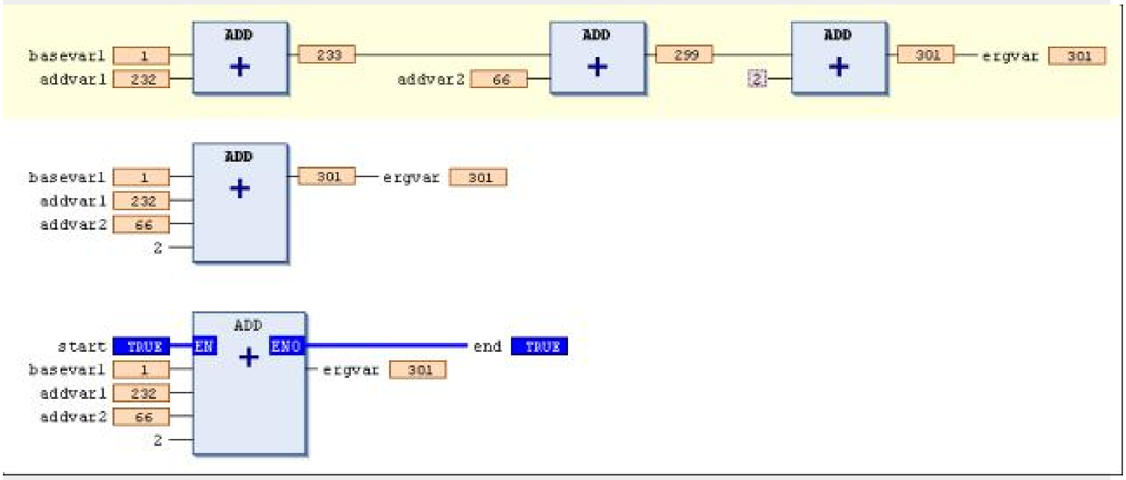
Features in the FBD/LD editor: ADD can be extended to function block inputs. The number of additional function block inputs is limited.

Example:

ST:

```
var1: = 7 + 2 + 4 + 7;
```

FBD:



"MUL" Multiplication operations

This IEC operator is used to multiply variables together.

Allowed data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME

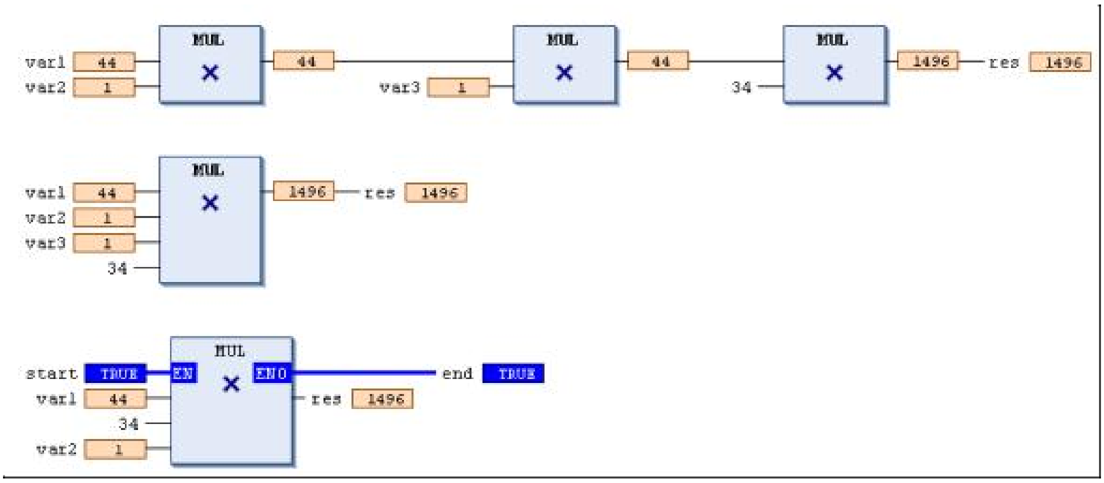
Function in the FBD / LD editor: MUL operators can be extended to additional function block inputs. The number of additional function block inputs is limited.

Example:

ST:

```
var1: = 7 * 2 * 4 * 7;
```

FBD:



“SUB” Subtraction operations

This IEC operator is used to subtract the variable.

Allowed data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, , , TIME_OF_DAY (TOD)DATEDATE_AND_TIME (DT)

Possible combinations of TIME data types: TIME-TIME= TIME, DATE-DATE= TIME, TOD-TIME= TOD, TOD-TOD= TIME, DT-TIME= DT, DT-DT=TIME

Note

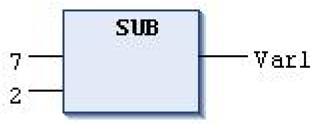
Negative TIME values are not defined.

Example:

ST:

```
var1: = 7-2;
```

FBD:



“DIV” Division operations

This IEC operator is used to divide variables.

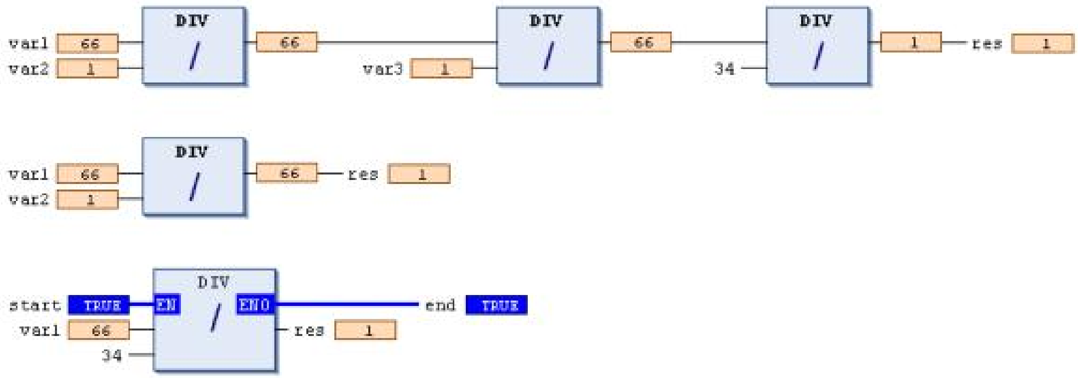
Allowed data types: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, TIME, The result of dividing by zero may vary depending on the target system.

Example:

ST:

```
var1: = 8/2;
```

FBD:

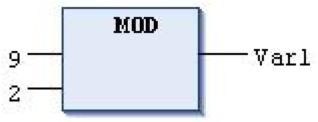


"MOD"take-out operation

This IEC operator is used for die-out.
The result of the function is an integer remaining part of the divide.
Allowed data types:BYTE,WORD,DWORD,LWORD,SINT,USINT,INT,UINT,DINT,UDINT,LINT,ULINT.
Results divided by zero may vary depending on the target system.
Example
ST:

```
var1: = 9 MOD 2;    //result: 1
```

FBD:



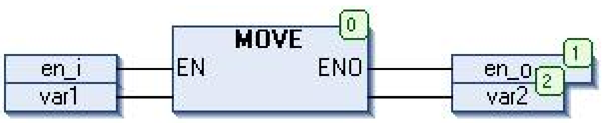
"MOVE" Assignment operations

This IEC operator is used to assign a variable to another variable of the corresponding type.

As this MOVE block is available in the CFC, FBD and LD editors, it is also possible to use the EN / ENO function for variable assignment.

CFC with EN / ENO function:

CODESYS assigns the value of var1 to var2 only if "en_i" is TRUE.



ST:

```
ivar2 := MOVE(ivar1);
```

Equivalent to:

```
ivar2 := ivar1;
```

“SIZEOF“Byte operations

This operator is an extension of the IEC 61131-3 standard. This operator is used to define the number of bytes x required for a variable.

The SIZEOF operator always produces an unsigned value. The type of the returned variable is adapted to the detected variable size x.

Return value of SIZEOF(x)	Constant data types (CODESYS uses implicit size detection)
0 <= size of x < 256	USINT
256 <= size of x < 65536	UINT
65536 <= size of x < 4294967296	UDINT
4294967296 <= size of x	ULINT

Example:

ST:

```
arr1 : ARRAY[0..4] OF INT;
Var1 : INT;
var1 := SIZEOF(arr1);          (* var1 := USINT#10; *)
```

6.3.2 Bit-Serial Operators

“AND”

This IEC operator is used to operate on numbers by AND bits.

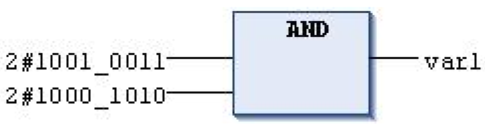
When the input bits are all 1, the output bit is 1; otherwise, the output is 0. Allowed data types:BOOL, BYTE, WORD, DWORD, LWORD

Example:

ST:

```
var1: = 2#1001_0011 AND 2#1000_1010;   // The result var1 is 2#1000_0010
```

FBD:



“OR”

This IEC operator is used to operate on numbers by OR bits.

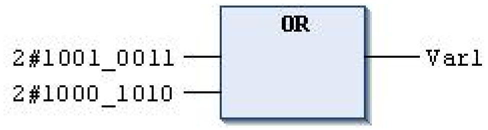
When at least one of the input bits produces a 1, the output bit also produces a 1; otherwise, the output bit is 0. Allowed data types:BOOL, BYTE, WORD, DWORD, LWORD

Example:

ST:

Var1: = 2#1001_0011 OR 2#1000_1010; //The result var1 is 2#1001_1011

FBD:



"NOT"

This IEC operator is used for the by bit of the NOT bit operand.

The output bit produces a 1 when the corresponding input bit produces a 0, and vice versa.

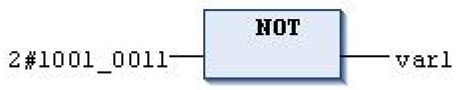
Allowed data types: BOOL, BYTE, WORD, DWORD, LWORD

Example:

ST:

var1: = NOT 2#1001_0011; //The result var1: 2#0110_1100

FBD:



"XOR"

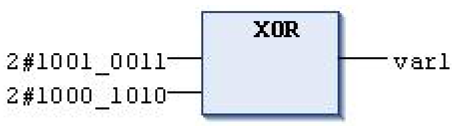
When and only when one of the two input bits is 1, the output bit also produces a 1. When both inputs are 1 or both are 0, the output produces a 0. Allowed data types: BOOL, BYTE, WORD, DWORD, LWORD

Example:

ST:

var1: = 2#1001_0011 XOR 2#1000_1010; //The result var1: 2#0001_1001

FBD:



"AND_THEN"

TRUE when all operands are produced and the result of the operand is produced TRUE; otherwise FALSE.

"OR_ELSE"

When at least one operand is produced TRUE, the result of the operation also produces TRUE; otherwise FALSE.

6.3.3 Shift operators

"SHL" Left shif

This IEC operator is used to shift the operand to the left.

`erg := SHL (in, n)`

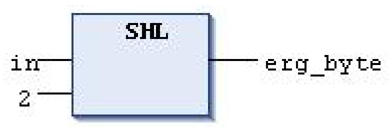
in: the operand shifted to the left, n: the number of bits of in shifted to the left. The number of bits n for this operation is defined by the data type of the input variable in.

Example:

ST:

```
PROGRAM shl_st
VAR
 in_byte : BYTE := 16#45;         (* 2#01000101 )
 in_word : WORD := 16#0045;         (* 2#0000000001000101 )
 erg_byte : BYTE;
 erg_word : WORD;
 n: BYTE := 2;
END_VAR

erg_byte := SHL(in_byte,n);     (* Result is 16#14, 2#00010100 *)
erg_word := SHL(in_word,n);     (* Result is 16#0114, 2#0000000100010100 *)
```

FBD:



"SHR" Right shift

This IEC operator is used to move the operand to the right.

erg := SHR (in, n)

in: the operand shifted to the right, n: the number of bits of in shifted to the right.

Example.:

ST:

```
PROGRAM shr_st
VAR
 in_byte : BYTE:=16#45; (* 2#01000101 )
 in_word : WORD:=16#0045; (* 2#0000000001000101 )
 erg_byte : BYTE;
 erg_word : WORD;
 n: BYTE :=2;
END_VAR

erg_byte := SHR(in_byte,n); (* Result is 16#11, 2#00010001 *)
erg_word := SHR(in_word,n); (* Result is 16#0011, 2#0000000000010001 *)
```

FBD:

“ROL” Cyclic left shift

This IEC operator is used to loop the operand to the left. Allowed data types: BYTE, WORD, DWORD, LWORD

erg := ROL (in, n)

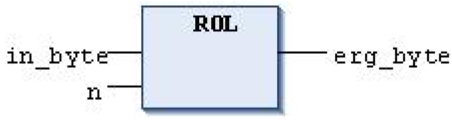
Moves in n bits to the left and then adds that bit from the right to the leftmost position. Define in by the data type of the input variable. if this is a constant, the smallest data type is used. The data type of the output variable still does not affect this operation.

Example:

ST:

```
PROGRAM rol_st

VAR
 in_byte : BYTE := 16#45;
 in_word : WORD := 16#45;
 erg_byte : BYTE;
 erg_word : WORD;
 n: BYTE := 2;
END_VAR

erg_byte := ROL(in_byte,n); (* Result: 16#15 *)

erg_word := ROL(in_word,n); (* Result: 16#0114 *)
```

FBD:



“ROR” Cyclic right shift

This IEC operator is used to loop the operand to the right. Allowed data types: BYTE, WORD, DWORD, LWORD

erg := ROR(in,n)

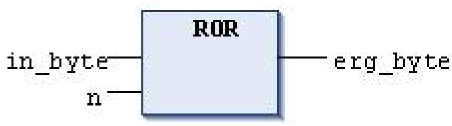
Moves in n bits to the right and then adds that bit from the left to the rightmost position. Define in by the data type of the input variable. if this is a constant, the smallest data type is used. The data type of the output variable still does not affect this operation.

Example:

ST:

```
PROGRAM ror_st

VAR
  in_byte : BYTE := 16#45;
  in_word : WORD := 16#45;
  erg_byte : BYTE;
  erg_word : WORD;
  n: BYTE := 2;
END_VAR

erg_byte := ROR(in_byte,n); (* Result: 16#51 *)

erg_word := ROR(in_word,n); (* Result: 16#4011 *)
```

FBD:



6.3.4 Selection operators

“SEL” Select

The IEC operator is used to select by bit.

OUT := SEL(G, IN0, IN1)

Equivalent to :

OUT := IN0; if G = FALSE

OUT := IN1; if G = TRUE

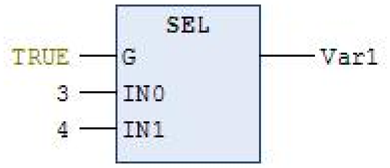
IN0, ... Data types for INn and OUT: any of the same data types， G: BOOL。

Example:

ST:

Var1: = SEL（TRUE，3,4）； (*Result: 4 *)

FBD:



“MAX” Maximum value

This IEC operator is used for the maximum function. It produces the maximum of two values.

OUT := MAX(IN0, IN1)

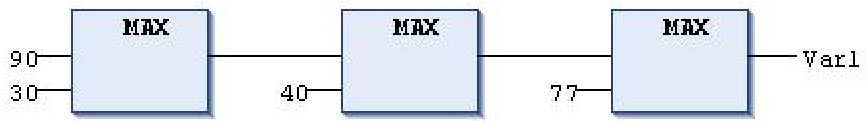
Allowed data types: all

Example

ST:

Var1: = MAX（40，MAX（90,30)）； Result：90

FBD:



“MIN” Minimum value

This IEC operator is used for the minimum function. It yields the smallest value of two values.

OUT := MIN(IN0,IN1)

Permitted data types: all

Example:

ST:

Var1: = MIN（MIN（90, 30）,40）； “ Result，30”

FBD:

"LIMIT" Limit values

This IEC selection operator is used to restrict the.

`OUT := LIMIT(Min, IN, Max)`

Equivalent to: OUT := MIN (MAX (IN, Min), Max), Max is the upper limit of the result, Min is the lower limit of the result. If the value IN is higher than the upper limit of Max, the LIMIT result is Max. If the value IN is lower than the minimum Min lower limit, the result is Min.

Allowed data types IN and OUT: All

Example

ST:

```
Var1: = LIMIT (30,90,80) ;          //The result Var1 is 80
```

"MUX" Multiplexing

This IEC operator is used as a multiplexer.

`OUT := MUX(K, IN0,...,INn)`, Equivalent to: `OUT = IN_K`

MUX selects the Kth value from a set of values. The first value is K = 0. If K is greater than the number of other inputs (n), the last value is passed (INn)

Allowed data types K: BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT, UDINT。

例:

ST:

```
Var1: = MUX (0,30,40,50,60,70,80) ;   // The result Var1 is 30.
```

6.3.5 Comparison operators

"GT" greater than

This IEC operator is used for the "greater than" function. If the first operand is greater than the second operand, the operator produces a TRUE result; otherwise FALSE.

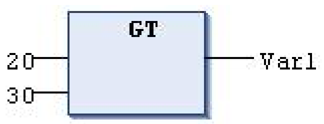
Allowed data types: any basic data type.

Example:

ST:

```
VAR1: = 20> 30; Result: FALSE
```

FBD:



"LT" Less than

This IEC operator is used for the "less than" function. If the first operand is less than the second, the operator produces a TRUE result; otherwise FALSE.

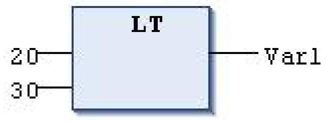
Allowed data types: any basic data type.

Example:

ST:

```
Var1: = 20 <30;     //Result:   TRUE
```

FBD:



"LE" Less than or equal to

This IEC operator is used in the "less than or equal to" function. If the first operand is less than or equal to the second operand, the operator produces a TRUE result; otherwise FALSE.

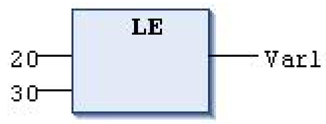
Allowed data types: any basic data type.

Example:

ST:

```
Var1: = 20 <= 30; Result:Var1: TRUE
```

FBD:



"GE" Greater than or equal to

This IEC operator is used in the "greater than or equal to" function. If the first operand is greater than or equal to the second operand, the operator produces a TRUE result; otherwise FALSE.

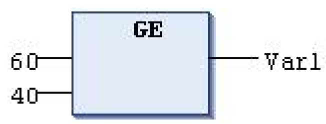
Allowed data types for operands: any basic data type.

Example:

ST:

```
VAR1: = 60> = 40;        result:   TRUE
```

FBD:



“EQ” equals

The IEC operator is used for the "equal" function. If the input numbers are equal, the operator produces a TRUE result, otherwise FALSE.

Allowed data types: any basic data type.

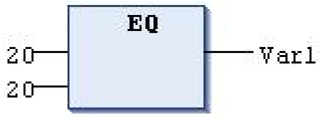
Example:

result：TRUE

ST：

```
VAR1： = 40 = 40；
```

FBD：



“NE”Not equals

This IEC operator is used in the "not equal" function. If the operands are not equal, the operator produces a TRUE result; otherwise FALSE.

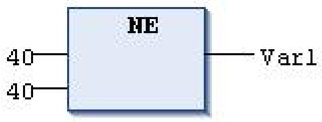
Allowed data types: any basic data type.

Example:

ST：

```
Var1： = 40 <> 40; result Var1 is FALSE
```

FBD：



6.3.6 Address operators

“ADR”

This operator is an extension of the IEC 61131-3 standard. the ADR generates the address DWORD of its parameter in. this address can be passed to the manufacturer function or they can be assigned to a pointer in the project.

Caution.

The ADR operator can be used together with function names, program names, function block names and method names. Thus, ADR replaces the INDEXOF operator.

ST：

```
VAR
    <address name> : DWORD | LWORD | POINTER TO < basis data type>
END_VAR
<address name> := ADR( <variable name> );
```

Example:

```
FUNCTION_BLOCK FB_Address
VAR
        piAddress1: POINTER TO INT;
        iNumber1: INT := 5;
        lwAddress2
        iNumber2: INT := 10;
END_VAR

piAddress1 := ADR(iNumber1); // piNumber is assigned to address of iNumber1
lwAddress2 := ADR(iNumber2); // 64 bit runtime system
```

“Content Operator”

This operator is an extension of the IEC 61131-3 standard. This operator can be used to dereference a pointer by appending ^ to the pointer identifier. When using a pointer to an address, please note that applying online changes may shift the address content.

Example:

ST:

```
pt: POINT TO INT;
var_int1: INT;
var_int2: INT;
pt: = ADR (var_int1) ;
var_int2: = pt ^;
```

“BITADR”

This operator is an extension of the IEC 61131-3 standard. bit offset DWORD in the segment generated by BITADR. the offset depends on whether the byte addressing checkbox is selected in the target system settings. The highest half-byte (4 bits) DWORD defines the storage range:

Flag: 16x40000000

Input: 16x80000000

Output: 16xC0000000

When using a pointer to an address, please note that applying online changes may shift the contents of the address

Example:

ST:

```
WHERE
    Var1 AT%IX2.3：BOOL;
    bitoffset：DWORD;
END_VAR

bitoffset：= BITADR（var1）；
 (* Byte addressing = TRUE 时：16x80000013，Byte addressing = FALSE 时：16x80000023 *)
```

6.3.7 Calling operators

“CAL” Call

This IEC operator is used to call function blocks. Example of a CAL call to a function block in IL

CAL <function block> (<input variable1> := <value>, <input variableN> := <value>)

Example：

`Inst calls the instance of the function block with the input variables Par1, Par2 and the assigned 0 or TRUE`

```
CAL Inst(Par1 := 0, Par2 := TRUE);
```

6.3.8 Numerical operators

“ABS” Absolute values

This IEC operator yields the absolute value of a number.
Allowed data types: any numeric basic data type
Example：
ST：

```
i：= ABS（-2）；    //result I is 2
```

FBD：

“SQRT”

This IEC of course yields the square root of a number.

Permitted data types for input variables: any numeric basic data type

Permitted data types for output variables: REAL or LREAL

Example：

ST：

```
q：= SQRT（16）；          // Result q：4
```

FBD：

"LN" Natural logarithm

This IEC operator yields the natural logarithm of a number.

Allowed data types for input variables: any numeric basic data type.

Permissible data types for output variables: REAL and LREAL.

Example：

ST：

```
Q = LN（45）；//result：  3.80666
```

FBD：

"LOG" Constant logarithm

This IEC operator yields a logarithm with a base of 10.

Allowed data types for input variables: any numeric basic data type.

Permissible data types for output variables: REAL and LREAL.

Example:

ST：

```
q：= LOG（314.5）;//result q：2.49762
```

FBD：

"EXP" Exponent of the natural number e

This IEC operator produces an exponential function.

Allowed data types for input variables: any numeric basic data type

Permissible data types for output variables: REAL and LREAL

Example:

ST:

```
q: = EXP (2) ;      //result q: 7.389056099
```

FBD:

"EXPT"（ Yth power of X)

This IEC operator is used to calculate the power function, power = base ^{exponent}.

Grammar:

```
EXPT(<base>,<exponent>)
```

Input value data type: SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, BYTE, WORD, DWORD, LWORD

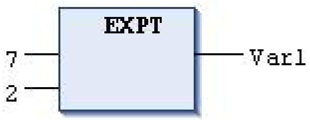
Data type of return value: floating point type（REAL and LREAL）

Example 1:

Power functions with text

```
Var1 := EXPT(7,2);
```

FBD:



result: Var1 = 49

Example 2:

Power functions with variables

```
PROGRAM PLC_PRG
VAR
    lrPow : LREAL;
    iBase : INT := 2;
    iExponent : INT := 7;
END_VAR

lrPow := EXPT(iBase, iExponent);    // result: lrPow = 128
```

“SIN” Sine function

This IEC operator yields the sine of a number.

Input variables for measuring angles in radians, allowed data types: any numeric basic data type

Allowed data types for output variables: REAL 和 LREAL

Example:

ST:

```
q: = SIN (0.5);            //result q: 0.479426
```

FBD:

“COS” The cosine function

This IEC operator yields the cosine value of a number.

Input variables for measuring angles in radians, allowed data types: any numeric basic data type

Allowed data types for output variables: REAL 和 LREAL

Example:

ST:

```
q: = COS (0.5) ;           //result q: 0.877583
```

FBD:

“TAN” tangent function

This IEC operator yields the tangent value of a number.

Input variables for measuring angles in radians, allowed data types: any numeric basic data type

Allowed data types for output variables: REAL 和 LREAL

Example:

ST:

```
q: = TAN (0.5); //result q: 0.546302
```

FBD:

“ASIN” Sine function anyway

This IEC operator yields the inverse sine value of the number.

Permissible data types for input variables: any numeric basic data type

Permissible data types for output variables：REAL 和 LREAL

Example：

ST：

```
q：=ASIN（0,5）;          //result q：  0.523599
```

FBD：

“ACOS” The inverse cosine function

This IEC operator yields the inverse cosine of the number. The value is calculated in radians.

Permissible data types for input variables: any numeric basic data type

Permissible data types for output variables：REAL 和 LREAL

Example：

ST：

```
q：=ACOS（0.5）;          //result : q=1.0472
```

FBD：

“ATAN” Anyway tangent function

This IEC operator yields the arctangent value of the number. The value is calculated in radians.

Input variables for measuring angles in radians, permitted data types: any numeric basic data type

Allowed data types for output variables：REAL 和 LREAL

Example：

ST：

```
q: =与 (0.5);      //result q: 0.463648
```

FBD:

6.3.9 Type conversion operators

“BOOL_TO”

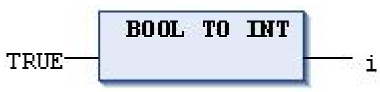
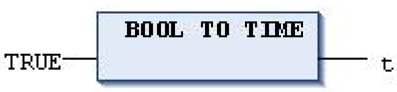
The IEC operator is used to convert a BOOL data type to another data type. Syntax:

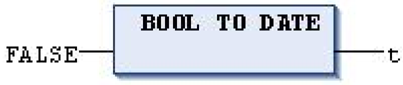
`BOOL_TO_<data type>`

When the data type is NUMBER, the result is 1 when the Boolean value is TRUE and 0 when it is FALSE.

When STRING data type, the result is TRUE or FALSE。

Example:

ST code	Result
i := BOOL_TO_INT(TRUE);	1
str := BOOL_TO_STRING(TRUE);	TRUE
t := BOOL_TO_TIME(TRUE);	T#1ms
tof := BOOL_TO_TOD(TRUE);	TOD#00:00:00.001
dat := BOOL_TO_DATE(FALSE);	D#1970
dandt := BOOL_TO_DT(TRUE);	DT#1970-01-01-00:00:01
FBD code	**Result**
	1
	TRUE
	T#1ms
	TOD#00:00:00.001

FBD code	Result
	D#1970-01-01
	DT#1970-01-01-00:00:01

“TO_BOOL”

The IEC operator is used to convert other variable types to BOOL variables. Syntax:

```
<data type>_TO_BOOL
```

The result is TRUE when the operand is not equal to 0. The result is FALSE when the operand is equal to 0.

For the STRING type, the result is TRUE if the operand is "TRUE"; otherwise it is FALSE.

Example:

ST code	Result
b := BYTE_TO_BOOL(2#11010101);	TRUE
b := INT_TO_BOOL(0);	FALSE
b := TIME_TO_BOOL(T#5ms);	TRUE
b := STRING_TO_BOOL('TRUE');	TRUE

FBD code	Result
	TRUE
	FALSE
	TRUE
	TRUE

“TO_ <xxx>”

The IEC operator is used to convert a variable of one data type to another data type.

```
TO_<data type>
```

Information may be lost when converting from a larger data type to a smaller data type.

If the value to be converted is outside the range limit, CODESYS will ignore the first few bytes of the value. This is the case, for example, when converting from LREAL to a negative DINT input value.

Example:

ST:

```
VAR
    iVar : INT;
    bVar : BOOL;
    sVar : STRING;
    rVar : REAL;
END_VAR

iVar := TO_INT(4.22);           (* Result:   4 *)
bVar := TO_BOOL(1);             (* Result: TRUE *)
sVar := TO_STRING(342);         (* Result: '342' *)
rVar := TO_WORD('123');         (* Result: 123 *)
```

“<INT Type>_TO_<INT Type>”

Converts one integer data type to another integer data type.

`<INT data type>_TO_<INT data type>`

Note that information may be lost when converting from a larger data type to a smaller data type. If the value to be converted is outside the range limit, CODESYS will ignore the first few bytes of the value.

Example：

ST:

```
si := INT_TO_SINT(4223);      //Result: Result in si: 127
```

If the integer 4223 (represented in hexadecimal as 16 #107f) is saved as a separate SINT variable, the value 127 (represented in hexadecimal as 16 #7f) will be assigned to this variable.

FBD:

“REAL_TO- / LREAL_TO”

The IEC operator is used to convert REAL and LREAL data types to another data type.

`REAL_TO_<data type>`

`LREAL_TO_<data type>`

Rounds the real value of an operand up or down to an integer value and then converts it to the appropriate type. (Exceptions are STRING, BOOL, REAL, and LREAL data types).

Examples

ST code	Result
`i := REAL_TO_INT(1.5);`	2
`j := REAL_TO_INT(1.4);`	1
`i := REAL_TO_INT(-1.5);`	-2
`j := REAL_TO_INT(-1.4);`	-1
FBD code	**Result**
	2

"TIME_TO / TIME_OF_DAY_TO"

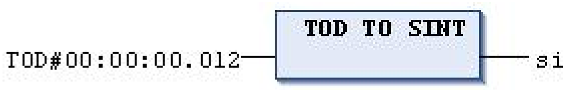
This IEC operator is used for converting the `TIME` and `TIME_OF_DAY` data types into another data type.

`<TIME data type>_TO_<data type>`

Internally, CODESYS saves the time (in milliseconds) to a `DWORD` (for `TIME_OF_DAY` since 00:00). CODESYS converts this value.

For the `STRING` data type, the result is the time constant.

Example:

ST code	Result
`str := TIME_TO_STRING(T#12ms);`	T#12ms
`dw := TIME_TO_DWORD(T#5m);`	300000
`si := TOD_TO_SINT(TOD#00:00:00.012);`	12
FBD code	**Result**
	T#12ms
	30000
	12

"DATE_TO / DT_TO"

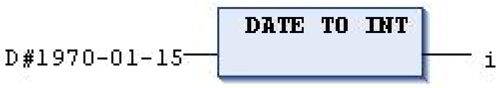
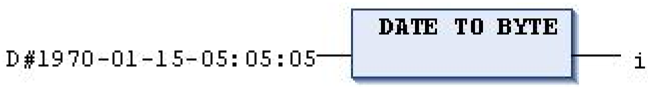
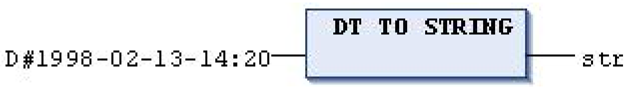
This IEC operator is used for converting the `DATE` and `DATE_AND_TIME` data types into another data type.

`<DATE data type>_TO_<data type>`

Internally, CODESYS saves the date to a `DWORD` (in seconds since 01 January 1970). CODESYS converts this value.

For the `STRING` data type, the result is the date constant.

Example:

ST code	Result
`b := DATE_TO_BOOL(D#1970-01-01);`	FALSE
`i := DATE_TO_INT(D#1970-01-15);`	29952
`i := DT_TO_BYTE(DT#1970-01-15-05:05:05);`	129
`str := DT_TO_STRING(DT#1998-02-13-14:20);`	DT#1998-02-13-14:20
FBD code	**Result**
	FALSE
	29952
	129
	DT#1998-02-13-14:20

"STRING_TO"

This IEC operator is used for converting the `STRING` data type into another data type.

`STRING_TO_<data type>`

You must define the `STRING` operand according to the IEC 61131-3 standard. The value has to be a valid constant (literal) of the target type. This affects any given exponential values, infinite values, prefixes, grouping characters (_), and commas. Additional characters after the digits of a number are permitted (for example, `23xy`). Additional characters before a number are not permitted.

The operand must be a valid value of the target data type.

Example:

ST code	Result
b := STRING_TO_BOOL('TRUE');	TRUE
w := STRING_TO_WORD('abc34');	0
w := STRING_TO_WORD('34abc');	34
t := STRING_TO_TIME('T#127ms');	T#127ms
r := STRING_TO_REAL('1.234');	1.234
bv := STRING_TO_BYTE('500');	244
FBD code	**Result**
	TRUE

"TRUNC"

This IEC operator is used for converting the `REAL` data type into the `DINT` data type. CODESYS takes only the integer part of the number.

Example:

ST:

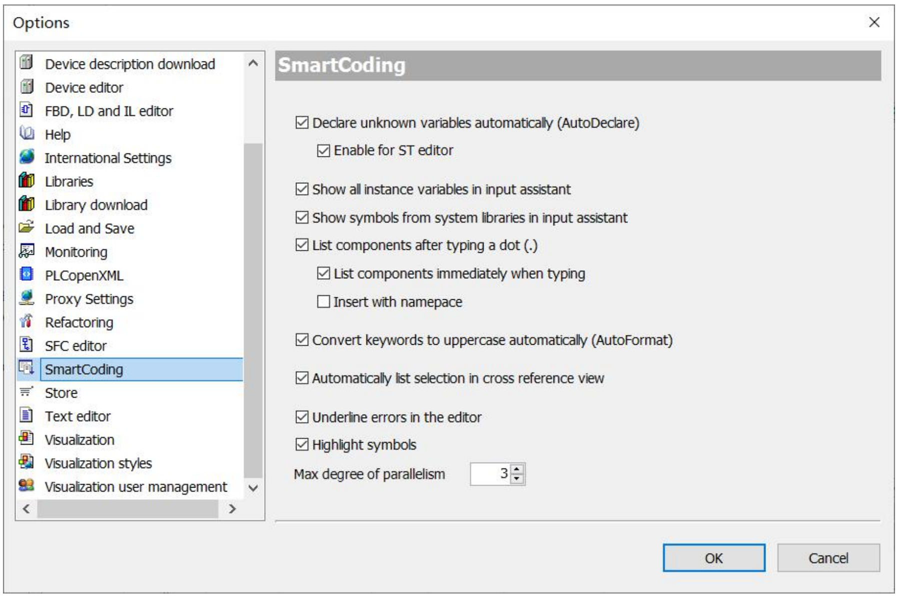
```
diVar := TRUNC(1.9);        (* Result: 1 *)

diVar := TRUNC(-1.4);      (* Result: -1 *)
```

6.4 Structured text(ST).

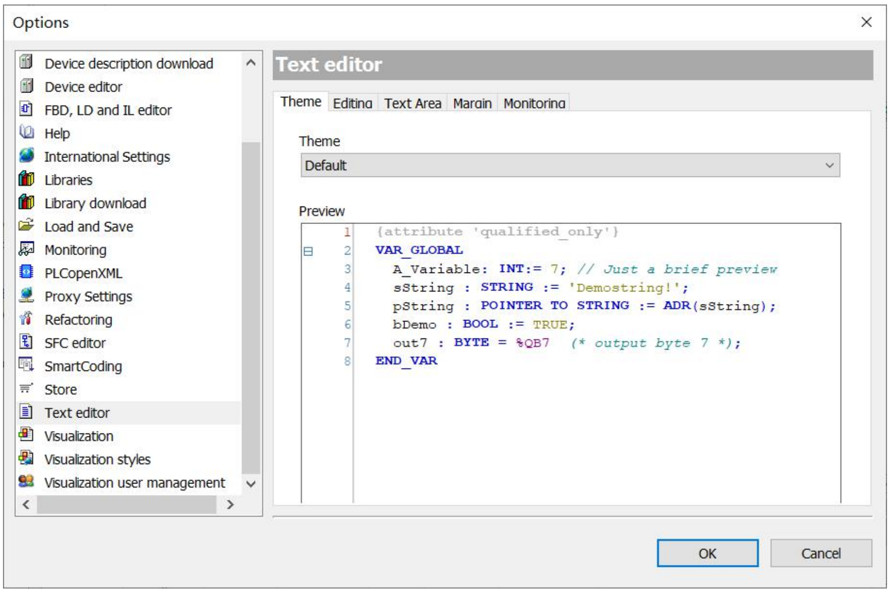
6.4.1 ST Editor

The ST Editor is a text editor for implementing code in structured text (ST) and extended structured text (ExST).

The line number appears on the left side of the editor. The List components feature (activated in the SmartCoding category of the CODESYS option) and input assistant F2 are also helpful when entering programming elements. When the cursor is placed on a variable, CODESYS displays a tool tip that contains information to declare the variable.



The behavior of the editor (for example, parentheses, mouse actions, tabs) and appearances are configured in the Text Editor category of codeSYS options.



6.4.2 The ST expression

An expression is a construct that returns a value after its evaluation.

Expressions consist of operators and operans. In Extended Structured Text (ExST), you can also use assignments as expressions. Operans can be constants, variables, function calls, or other expressions.

Cases：

2014	(* Constant *)
ivar	(* Variable *)
fct(a,b)	(* Function call *)
(x*y)/z	(* Expression *)
real_var2 := int.var;	(* in ExST: Assignment *) *)

You can prioritize expressions by handling operators based on certain connection rules. CODESYS first handles operators with the strongest connections. Operators with the same connection strength are processed from left to right.

Operators	Symbols	Connection strength
Parenthesize	(Expression)	Strongest binding
Function Call	Function name (parameter list) all operators with syntax: <operator> ()	
Exponentiate	EXPT	
Negate Complementation	- NOT	
Multiplication Division	* /	

Operators	Symbols	Connection strength
Modulo	MOD	
Addition Subtraction	+ -	
Comparison	<,>,<=,>=	
Equality Inequality	= <>	
Bool AND	AND AND_THEN	
Bool XOR	XOR	
Bool OR	OR OR_ELSE	Weakest binding

6.4.3 ST assignment method

The assignment expression

语法：<operand> := <expression>

The assignment operator performs the same functions as the MOVE operator.

The ST assignment operator for the output

The assignment operator assigns the output of a function, function block, or method to a variable.

Grammar:

```
<output> => <variable>
```

Example:

```
FBcomp_Output1 => bVar1;
FBcomp_Output2 =>;
```

FBcom_Output1 and FB_Output2 are the values of the output of the function block,FBcom_Output1 assigned to the variable bVar1.

Extended ST assignments "S", "R""

"S" is equivalent to "SET" in the PLC,withthe syntax:

```
<variable name> S= <operand name> ;
```

The data type of the variable and operand is BOOL, and true is assigned to the variable Variable when operand switches FROM TO TRUE. However, the variable remains in true state even if Operaand continues to change its state.

Cases:

```
PROGRAM PLC_PRG
VAR
    xOperand: BOOL := FALSE;
    xSetVariable: BOOL := FALSE;
END_VAR

xSetVariable S= xOperand;
```

"R" is equivalent to "RST" in the PLC, syntax:

`<variable name> R= <operand name> ;`

The data type of the variable and operand is BOOL, and false is assigned to the variable Variable when operand switches from FALSE to TRUE. However, even if the operans continue to change their state, the variable remains in state FALSE.

6.4.4 ST syntax

IF statement

If statements are used to examine a condition and execute subsequent statements based on that condition. Grammar:

```
IF <condition> THEN
    <statements>
ELSIF <condition> THEN
        <statements>
…
ELSE
        <statements>
END_IF;
```

The ELSIF branch and the ELSE branch are optional.

Cases:

```
PROGRAM PLC_PRG
VAR
    iTemp: INT;
    xHeatingOn: BOOL;
    xOpenWindow: BOOL;
END_VAR

IF iTemp < 17 THEN
    xHeatingOn := TRUE;
```

```
ELSIF iTemp > 25 THEN
    xOpenWindow := TRUE;
ELSE xHeatingOn := FALSE;
END_IF;
```

The FOR statement

The FOR loop is used to execute instructions that have a certain number of repetitions. Grammar:

```
FOR <counter> := <start value> TO <end value> {BY <increment> } DO
<instructions>
END_FOR;
```

The parts in parentheses are optional. As long as the value of slt;counter is not greater than that of slt;end value; and not less than slt;start value, then the counter slt;instructions are executed and the counter is automatically increased every time the instructions are executed. Increments can be any integer value. If no increment is specified, the standard increment is 1.

Cases:

```
FOR iCounter := 1 TO 5 BY 1 DO
iVar1 := iVar1*2;
END_FOR;
Very := iVar1;
```

When the initial valueof iVar 1 is 1, the value of iVar is 32at the end of the FORloop.

CASE statement

Use this dialog box to combine multiple conditional instructions that contain the same condition variable into a construct. Grammar:

```
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.

Example:

```
CASE iVar OF
1, 5: bVar1 := TRUE;
        bVar3 := FALSE;

2: bVar2 := FALSE;
    bVar3 := TRUE;

10..20: bVar1 := TRUE;
           bVar3= TRUE;
ELSE
    bVar1 := NOT bVar1;
    bVar2 := bVar1 OR bVar2;
END_CASE;
```

WHILE statement

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;
```

CODESYS 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 CODESYS 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.

example:

```
WHILE iCounter <> 0 DO
Var1 := Var1*2
iCounter := iCounter-1;
END_WHILE;
```

In a certain sense the WHILE and REPEAT loops are more powerful than the FOR loop, since

you don't need to already know the number of executions of the loop before its execution. In some cases it is thus only possible to work with these two kinds of loop. If the number of executions of the loop is clear, however, then a FOR loop is preferable in order to avoid endless loops.
As an extension to the IEC 61131-3 standard you can use the CONTINUE instruction within the WHILE loop.

REPEAT statement

The REPEAT loop is used like the WHILE loop, but with the difference that CODESYS 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:

```
REPAEAT
<instructions>
 UNTIL <boolean expression>
 END_REPEAT;
```

executes the <instructions> until the <boolean expression> returns TRUE.
If the boolean expression already returns TRUE at the first evaluation, CODESYS 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.

Example:

```
REPEAT
Var1 := Var1*2;
iCounter := iCounter-1;
UNTIL
iCounter = 0
END_REPEAT;
```

RETURN

Use the `RETURN` instruction in order to exit from a function block. You can make this dependent on a condition, for example.
Example:

```
IF xIsDone = TRUE THEN
    RETURN;
END_IF;

iCounter := iCounter + 1;
```

If the value of `b` is `TRUE`, the function block is exited immediately and CODESYS does not execute the instruction `a:=a+1;`.

JMP

The `JMP` instruction is used to execute an unconditional jump to a program line that is marked by a jump label.

Syntax:

```
<label>: <instructions>
JMP <label>;
```

The jump label `<label>` is any unique identifier that you place at the beginning of a program line. On reaching the `JMP` instruction, a return to the program line with the `<label>` takes place.

```
iVar1 := 0;
_label1: iVar1 := iVar1+1;
(*instructions*)

IF (iVar1 < 10) THEN
JMP _label1;
END_IF;
```

EXIT

The EXIT instruction is used in a FOR, WHILE or REPEAT loop to immediately end the loop regardless of its stop condition.

CONTINUE

`CONTINUE` is an instruction of the Extended Structured Text (ExST).

The instruction is used inside `FOR`, `WHILE` and `REPEAT` loops in order to jump to the beginning of the next execution of the loop.

Example:

```
FOR Counter:=1 TO 5 BY 1 DO
INT1:=INT1 / 2;
 IF INT1=0 THEN
  CONTINUE; (* to provide a division by zero *)
 END_IF
Var1:=Var1/INT1; (* executed, if INT1 is not 0 *)
END_FOR;
```

```
Erg:=Var1;
```

ST Function Block Call

Syntax of ST function block calls:

```
<FB-instance>(<FB input variable>:=<value or address>|, <other FB input variables>);
```

Example:

```
TMR:TON;

TMR (IN:=%OX5, PT:=T#300ms);
varA:=TMR.Q;
```

The timer function block TON has been instantiated in TMR:TON and is called using the allocated parameters IN and PT. The output is addressed with Q to TMR.Q and assigned to the variable varA.

ST COMMENTS

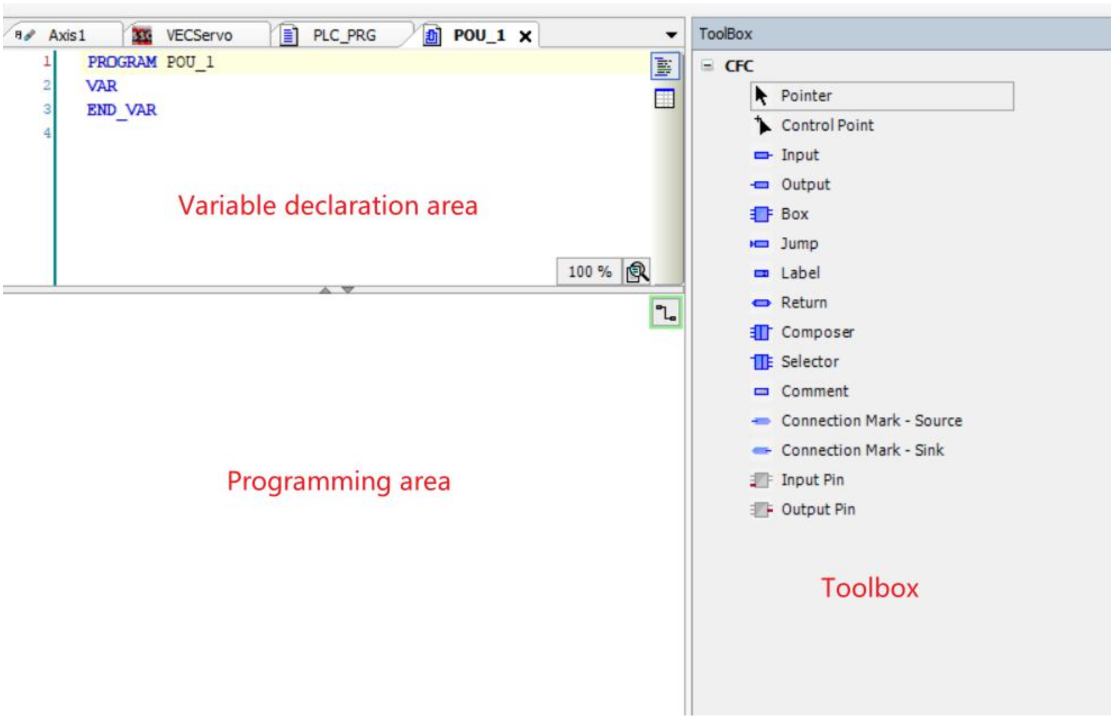
Note	Description	**example**
Single line	// starts and ends at the end of the line	`// This is a comment`
Multi-line	(* opening, *) ending	`(* This is a multi-line comment *)`
Nested	(* opening, *) closing, possibly with comments inside the comment (*.... *)	`(* a:=inst.out; (* 1st comment *) b:=b+1; (* 2nd comment *) *)`

6.5 Continuous function diagrams（CFC）

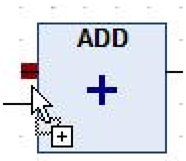
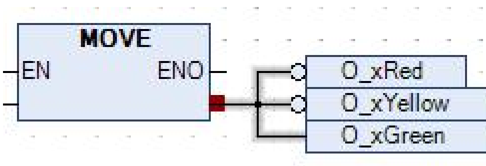
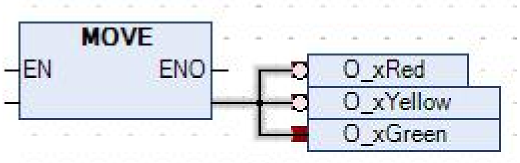
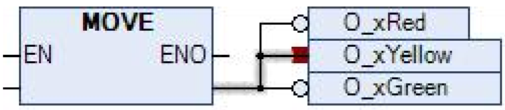
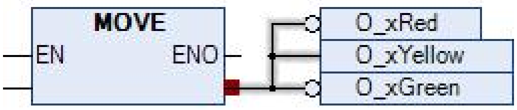
The Continuous Function Chart (CFC) language is a graphical programming language that is an extension of the standard language of IEC 61131-3. Systems can be programmed graphically using the POU in CFC. Elements can be inserted and placed freely, connections inserted and elements connected to a network in order to create well-structured functional block diagrams.

The order of execution of the function block diagram is based on the data flow. Furthermore, the POU can handle multiple data streams. This way the data streams do not have any common data. In the editor, there are no connections between multiple networks.

6.5.1 CFC Editor



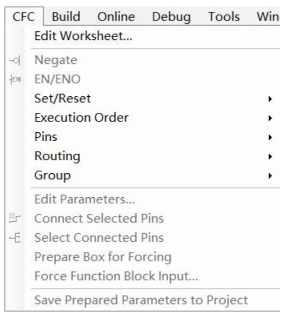
Cursor symbols:	Requirement: Pointer selected in the toolbox view 。 The symbols indicate that you can edit in the editor. Select elements or links to move them or to execute commands.
Cursor symbols: +	Requirement: Any of the elements is selected in the toolbox view. Clicking in the editor will insert the selected element. You can also drag the element into the editor.
Example of dragging a function block from an editor's declaration	Requirement: A line is selected in the declaration of the CFC. Instances will be inserted as POUs with name, type and all pins.

Dragging variables from a declaration to a POU pin in the editor	The variable is inserted as an input or output and is connected to the POU pin where the focus is located. Hint: The cursor indicates when your focus position is valid for the variable 
`Ctrl` + Click on the programming area	Requirement: An element is selected in the toolbox view. Each click in the programming area will create a selected element each time as long as the Ctrl key is held down.
`Ctrl+Right Arrow`	Requirement: In the CFC program, exactly one output pin is selected for an element. Move the selection so that the input pin at the end of the connection line is selected. If there are multiple pins, select them all.  
`Ctrl+Left Arrow`	Requirement: In the CFC program, exactly one element selects an input pin. Move the selection so that the output pin at the beginning of the connection line is selected. If there are multiple pins, select them all.  

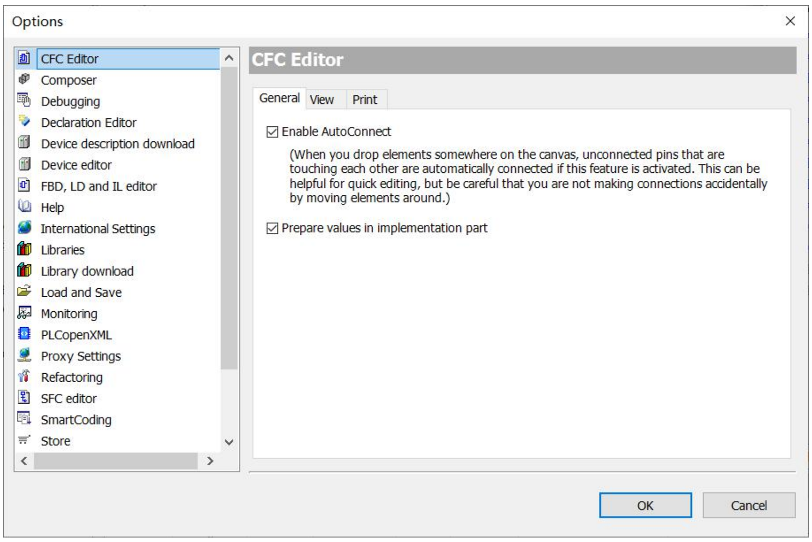
6.5.2 The order in which the CFC data flow is executed

In the CFC editor, elements are freely placed, so the execution order is not unique at first. Therefore, the software determines the order of execution through the data flow and, in the case of multiple networks, the order of execution is determined by the topological position of the element: the element is sorted from top to bottom, from left to right.

After adding the CFC language object POU, the menu bar appears in the POU interface with CFC options, as follows

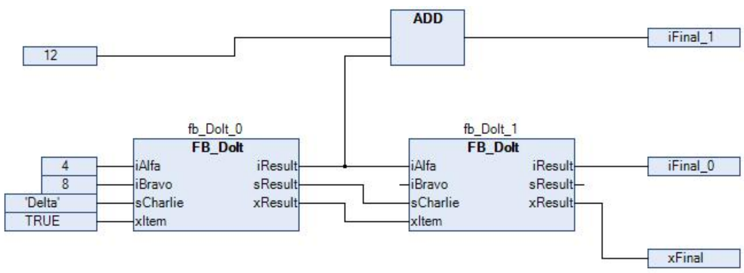


Open Tooloptions → CFC → Editor,openCFC editor settings, and customize the editor content.

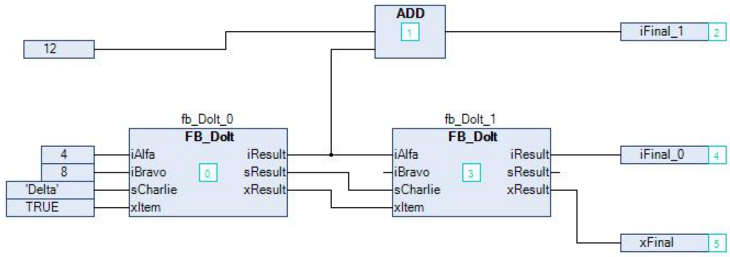


By default, the order in which CFC objects are executed is automatically determined. To do this, check the Display Execution Order property. You can check in the CFC Editor to temporarily display the automatically determined order of execution.

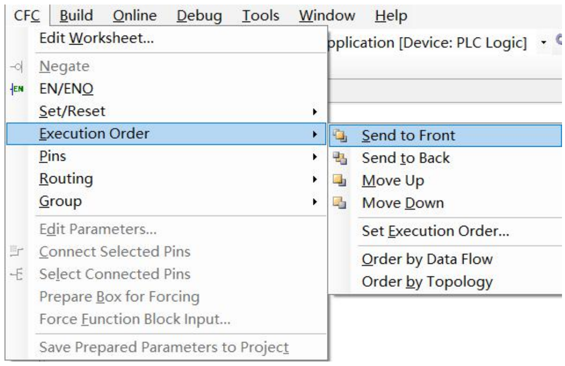
Example: The addition program is as follows



Click"CFC, Execution Order, Display Execution Order" to showthe order in whichobjects are executed. The boxes and inputs are numbered accordingly and reflect the chronological order. When you click again in the CFC editor, the number is hidden.



Right click on the function block, or click on the “CFC ‣ Execution Order”, The order of execution can be changed.



6.5.3 CFC elements

Page page

Symbol:

Element inserts a new page into the editor. Available only in page-oriented CFC editors. Page numbers are automatically assigned based on their location. You can enter the name and description of the page in the orange title. Use the Edit Page Size command to resize the page.

Control Point control point

Symbol:

Before adjusting the route, you can use the Control Point control point to secure the connection point. This way, the element is dragged to the desired location on the connector, and the connector with the control point is no longer automatically clothed.

Input

Symbol:

CODESYS inserts an input element by default with the text "???". You can directly edit this field by clicking on it and entering a constant value or a variable name. Alternatively you can open the input assistant in order to select a variable by clicking on .

Output

Symbol:

CODESYS inserts an output element by default with the text "???". You can directly edit this field by clicking on it and entering a constant value or a variable name. Alternatively you can open the input assistant in order to select a variable by clicking on .

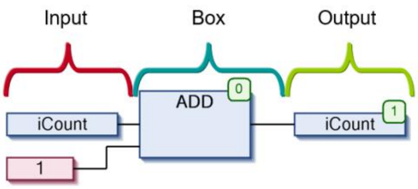
Box 运算块

符号：

Symbol:

You use the element in order to insert an operator, a function, a function block or a program. By default CODESYS inserts the element with the name "*???*". You can directly edit this field by clicking on it and entering a function block name. Alternatively you can open the input assistant and select a function block by clicking on .

In the case of a function block, CODESYS additionally displays an input field (*???*) above the function block symbol. You must replace this name by the name of the function block instance. If you instance a function block with constant input parameters, the function block element displays the 'Parameter...' field in the bottom left corner. You edit the parameters by clicking on this field：



Jump

Symbol:

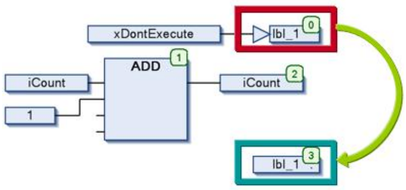
You use the element in order to define a position at which program execution is to continue. You must define this target position by a label. To do this, enter the name of the mark in the input field *???*. If you have already inserted the corresponding label, you can also select it via the input assistant ( ).

Label

Symbol:

A label defines a position to which program execution jumps with the help of a jump element.

Example: Jump and Label usage



Return

Symbol:

Use the elements to exit the function block POU.

Composer

Symbol:

The composer element is for handling structural components. The individual components of a structure are made available to you as an input. For this purpose you must name the composer element like the structure concerned (replace the *???*).

The composer element is the counterpart to the selector element.

Selector

Symbol:

The selector element is for handling structural components. The individual components of a structure are made available to you as an output. For this purpose you must name the selector element like the structure concerned (replace the "*???*")

The selector element is the counterpart to the composer element.

Comment

Symbol:

With this element you input a comment in the CFC editor. Replace the placeholder text in the element by the comment text. A line break can be inserted with the aid of the shortcut `Ctrl` + `Enter`.

Connection Mark - Source/Sink

Symbol: ,

You can use connection marks instead of a connecting line between elements. That helps you to display complex diagrams more clearly.

For a valid connection you must connect an element *Connection Mark - Source* with the output of an element and an element *Connection Mark - Sink* with the input of another element. Both marks must bear the same name. The names are not case-sensitive.

Input Pin

Symbol: 

Depending on the type of function block you can add further inputs to an inserted function block element. To do this you must select the function block element and drag the function block input element onto the body of the function block.

Please note: You can drag an input or output connection to another position on the function block with the `Ctrl` key pressed.

Output Pin

Symbol: 

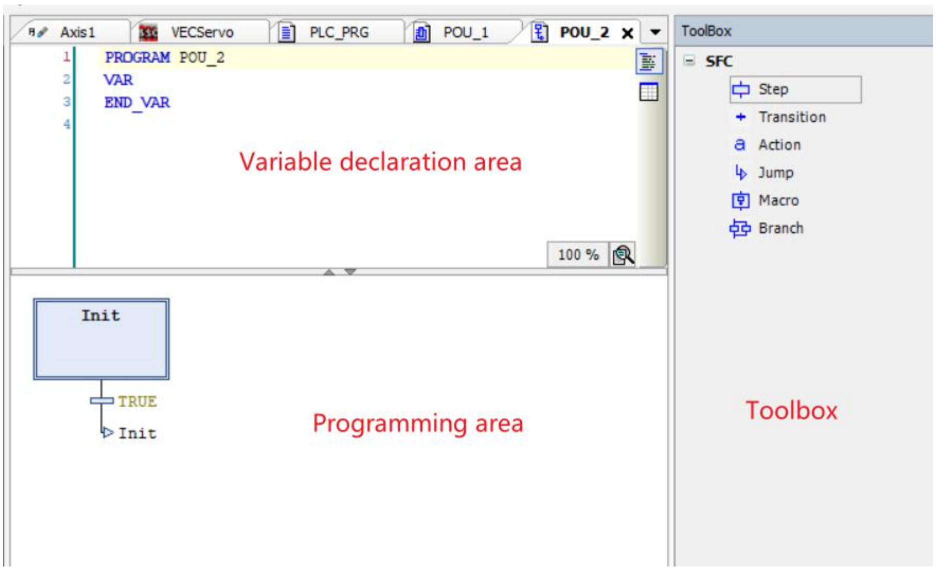
Depending on the type of function block you can add further outputs to an inserted function block element. To do this you must select the function block element and drag the function block output element onto the body of the function block.

Please note: You can drag an input or output connection to another position on the function block with the `Ctrl` key pressed.

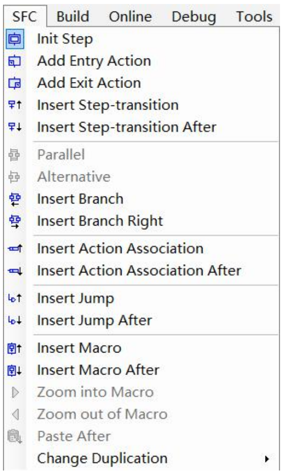
6.6 Sequential functionmap (SFC).

6.6.1 SFC Editor

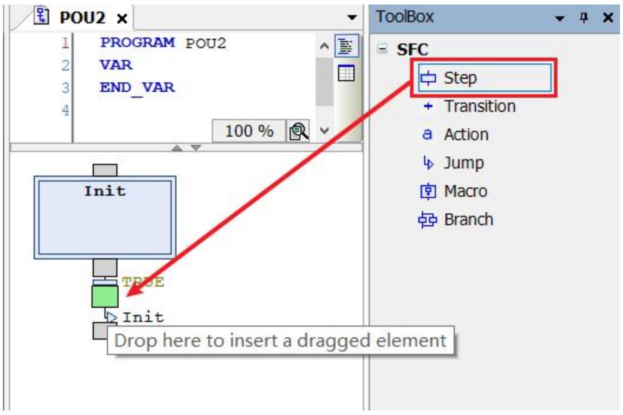
The SFC editor is the graphics editor. The new SFC POU contains an Init step and subsequent transitions.



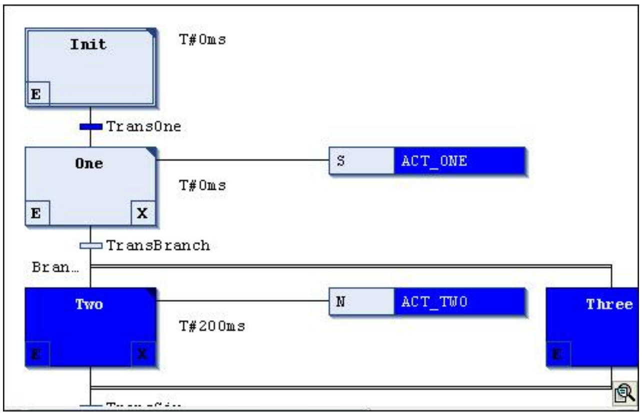
In the SFC editor, a single element can be inserted into the diagram via the SFC menu (which automatically appearswhen the POU in the SFC language isopened).



You can also drag SFC elementsfrom theToolBox Toolbox view to the figure. As you drag elements on the editor, the software marks all possible insertion points with a gray box. If you move the mouse over the gray box, the color of the box turns green. When you release the mouse button, the object is inserted into that location.



In online mode, CODESYS displays the activity steps in blue.



6.6.2 Theorder in which S FCs are processed

1, reset the IEC action

CODESYS resets the internal motion control flags of action qualifiers (N, R, S, L, D, P, SD, DS, SL), which control IEC actions.

2, perform an exit action

The software verifies that all steps meet the criteria for each step to perform an exit action. The validation order follows the layout in the SFC diagram, from top to bottom, left to right. When the step is disabled, the software performs an exit action (after any input and step actions were performed in the previous loop, and the conditions for the subsequent steps are true).

3, perform input actions

The software verifies that all steps meet the criteria for each step to perform an input action. The validation order follows the layout in the SFC diagram, from top to bottom, left to right. If the condition is met, the software performs an input operation. Once the conversion of the next step is complete and produces TRUE,the software immediately performs an input action indicating that the step has been activated.

4, time check /step action

The software performs the following checks on each step in the order of the SFC layout:

● The software copies the passing time of the active step to the corresponding implicit step variable. <step name>.t

● If a timeout occurs, CODESYS sets its own error flags.

● For non-IEC steps: CODESYS performs step actions.

5, the implementation of IEC action

CODESYS performs IEC actions in alphabetical order, twice through the action list. In the first cycle, the software performs IEC actions on each step that was disabled in the last loop. In the second time, perform the IEC action for each active step.

6, transition check/activate the next step

The transition passes as follows: If a step is active in the current loop and subsequent transitionsare generated, TRUE and the minimum time defined by the step has passed, the next steps are activated.

6.6.3 SFC Action conditions

Qualifiers can be assigned to IEC steps. The qualifier describes how the step action is handled.

The qualifiers are handled by the function block "IecSfc.library" in the "SFCActionControl" library. This library is automatically integrated into the project via the SFC plug-in.

N	Non-storage	The action is active as long as the step is active.
R0	Override reset	The action is disabled.
S0	Settings (stored)	When this step is activated, the software will perform this operation immediately. Even if the step is de-activated, the operation will continue until a reset is received.
L	Time limit	The software will perform this operation immediately after the step is activated. The operation will be performed until the step is disabled or the given time interval has elapsed.
D	Time delay	The software will only start executing the operation after a given delay time has elapsed since the step was activated and the step is still active. The operation is performed until the step is disabled.
P	Pulse	The software performs the action exactly twice: once to activate the step and once to activate the step.
SD	Store and time delay	The software will only start executing the action after a given delay time has elapsed since the step was activated. The action will be

		executed until a reset is received.
DS	Delay and storage	The software will only start executing an action after a given delay time has elapsed since the step was activated and the step is still active. The action will be executed until a reset is received.
SL	Limited storage time	The software will perform this operation immediately after this step has been activated. The operation will be performed until the given time has elapsed or until a reset is received.

6.6.4 SFC Implicit variables and flags

SFC Implicit variables

Each SFC object provides implicit variables that monitor the state of the step and IEC action at runtime, and CODESYS declares these implicit variables automatically for each step and each IEC action.

Implicit variables are structural instances of the step type SFCStepType and the action type SFCActionType. Variables have the same name as their elements, e.g. "step1" step name is "step1" variable name. The structure member describes the state of the step or action or the current time elapsed in the active step.

Syntax for implicit variable declarations:

```
<step name>:SFCStepType;
_<action name>:SFCActionType;
```

The following implicit variables are available for step or IEC action states:

Step	
<step name>.x	Shows the active status of the current cycle. When <step name>.x = TRUE , it indicates that the software is processing the steps of the current cycle.
<step name>._x	Displays the activation status of the next cycle. When <step name>. _x = TRUE and <step name>.x = FALSE, the software is processing the step in the next cycle
<step name>.t	The current elapsed time since the step was activated. This only applies to steps, regardless of whether a minimum time is defined in the step properties.
<step name>._t	Internal use only
IEC Action	
`_<action name>.x`	TRUE : The action is being executed.
`_<action name>._x`	TRUE : The action is active.

The above variables can be used to force a specific status value to a step (activation step).

Accessing implicit variables Syntax.

a. Assigning implicit variables directly in the POU:

`<variable name>:=<step name>.<implicit variable>`or

`<variable name>:=_<action name>.<implicit variable>`

Example:

```
status:=step1._x;
```

b. From another POU with the following name:

`<variable name>:=<POU name>.<step name>.<implicit variable>` or

`<variable name>:=<POU name>._<action name>.<implicit variable>`

Example:

```
status:=SFC_prog.step1._x;
```

SFC flags

SFC flags are implicitly generated variables with predefined names that are used to control the processing of SFC diagrams. For example, these flags can be used to display timeouts or reset step chains. In addition, the hint mode can be activated specifically to activate transitions. These variables must be declared and activated in order to access them.

Name	Date type	Description
SFCInit	BOOL	TRUE: The software resets the sequence to the initial step. The other SFC flags are also reset (initialised). When the variable is TRUE, the initial step remains set (active), but does not perform its operation. Only when SFCInit is re-given FALSE does its block processing continue down the line.
SFCReset	BOOL	This function is similar to SFCInit. however, the software continues processing after the initialization of the initial steps.
SFCError	BOOL	TRUE: If a timeout occurs in the SFC diagram. If a second timeout occurs in the program, it will not be set to FALSE unless you have previously reset the variable SFCError. other functions used to control the use of chronological flag variables (SFCErrorStep, SFCErrorPOU, SFCQuitError) require SFCError to be declared first.
SFCEnableLimit	BOOL	Used to activate (TRUE) and de-activate (FALSE) the timeout control SFCError. must be set TRUE to SFCError for it to work. If this is not done, the timeout is ignored.
SFCErrorStep	String	Stores the name of the step that caused the timeout. The prerequisite is that SFCError is declared.
SFCErrorPOU	String	Stores the name of the block in which the timeout occurred. The prerequisite is that SFCError is declared.

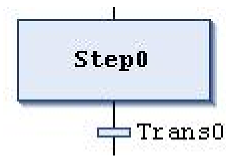
Name	Date type	Description
SFCQuitError	BOOL	TRUE: The software will pause the processing of the SFC graph and any timeout SFCError in this variable will be reset. If the variable is reset to FALSE, all previous times in the active step will be reset. The prerequisite is that SFCError is declared.
SFCPause	BOOL	TRUE: The software suspends the processing of the SFC.
SFCTrans	BOOL	TRUE if the TransitionTransidition is active.
SFCCurrentStep	String	Displays the name of the active step, independent of time monitoring. In parallel branches, the name of the step in the rightmost branch line is always stored.
SFCTip, SFCTipMode	BOOL	Controls the Tip mode of the SFC block. If this flag is enabled SFCTipMode=TRUE, the next step can be activated by setting SFCTip to TRUE. When SFCTipMode is set to when FALSE, the transition is used to continue activation.

6.6.5 SFC Element

Step and Transition

Step symbol ; Transition symbol

As a rule, CODESYS inserts steps and transitions as combinations. Inserting a step without a transition or a transition without a step causes an error when compiling. You can modify this by double-clicking the name.

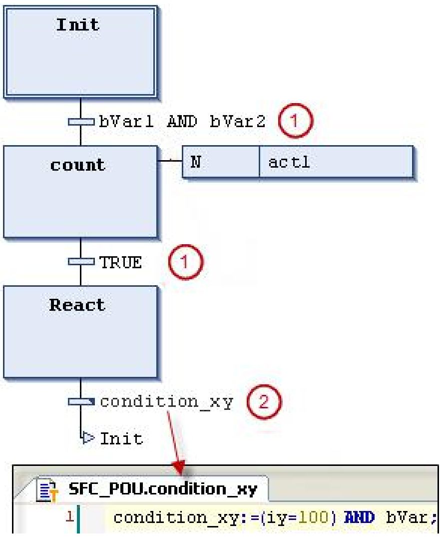


All steps are defined by the step properties, which you can display and edit in the *Properties* view, depending on the set options.

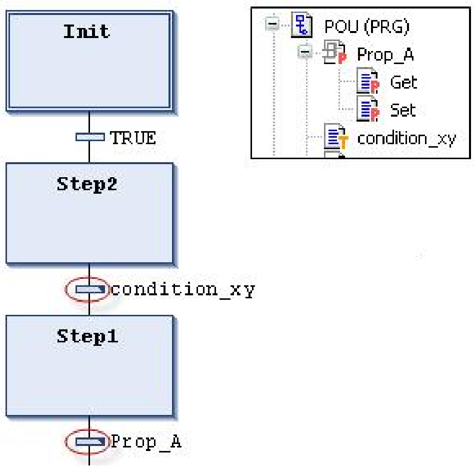
You have to add those actions to the step which are to be executed when the step is active. A distinction is made between IEC actions and step actions. Details for this are found in the chapter about the SFC element “Action”.

A transition must include the condition for the subsequent step to be active as soon as the value of the condition yields `TRUE`. Therefore, a transition condition must yield `TRUE` or `FALSE`. It can be defined in one of two ways:

1. Inline condition (direct): You replace the default transition name with either the name of a Boolean variable, a Boolean address, a Boolean constant, or a statement with a Boolean result, for example `(i<100) AND b`. You cannot specify programs, function blocks, or assignments here.
2. Multi-use condition (separate transition or property object): You replace the default transition name with the name of a transition or property object (,). You create these objects by clicking *Project* ‣ *Add Object* . This allows multiple use of transitions, for example "condition_xy" in the figures below. Like an inline condition, the object can contain a Boolean variable, Boolean address, Boolean constant, or an statement with a Boolean result. In addition, it can also contain multiple statements with any code.



Transitions that reference a transition or property object are marked with a small triangle in the upper right corner of the transition box.



As opposed to CoDeSys V2.3, now CODESYS treats a transition condition like a method call. The entry has the following syntax:

```
<transition name>:=<transition condition>
```

Example:

trans1:= a=100

or only

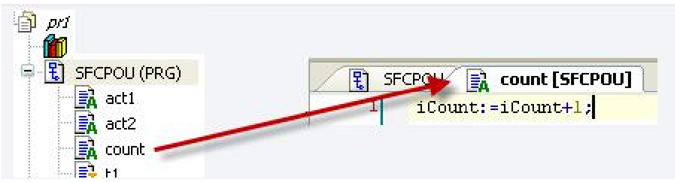
<transition condition>　　　　　//（for example :a=100）

Action

Symbol: 

An action includes a series of statements in one of the valid implementation languages. You can assign an action to a step.

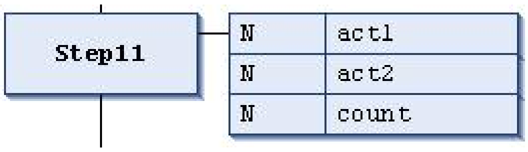
You must create all actions as POUs in the project when they are used in SFC steps.



1. IEC actions

   These actions comply with the IEC1131-3 standard. They are executed according to their qualifiers. Each action is executed two times: first when the step is activated and second when the step is deactivated. If you assign several actions to one step, the action list is processed from top to bottom.

   Each action box includes the qualifier in the first column and the action name in the second column, both of which can be edited directly.



As opposed to step actions, you can use different qualifiers for IEC actions. In addition, each IEC action is provided with a control flag. This directs CODESYS to execute an action only one time at any moment, even if the action is called by another step at the same time. This cannot be guaranteed for step actions.

You assign IEC actions to steps by clicking *SFC* ‣ *Insert action association* .

2. Step actions:

You can use these step actions to extend the IEC standard.

- **Entry action**:

CODESYS executes this action after the step is activated and before the main action is executed.

These reference a new action, or action created in below the SFC object, from a step by means of the *Entry action* step property. (2). You can also add a new action to the step by means of the *Add entry action* command. The entry action is marked with an E in the lower left corner of the step box.

- **Main action**:

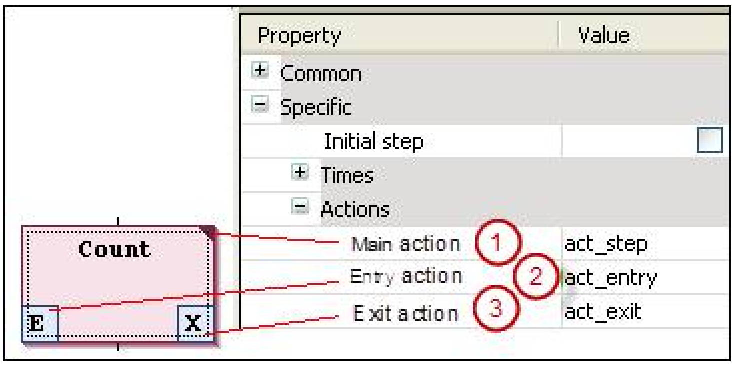
  CODESYS executes this action when the step is active and any entry actions have already been processed. However, as opposed to IEC actions (see above), these step actions a are not executed a second time when the step is deactivated. In addition, you cannot use qualifiers here.

  You add an existing action to a step by means of the *Main action* element property (1). You can create and add a new action by clicking the step element. A main action is marked with a filled triangle in the upper right corner of the step box.

- **Exit action**:

  CODESYS executes this action one time when the step is deactivated. Please note, however, that an exit action is not executed in the same cycle, but at the beginning of the next cycle.

  These reference a new action, or action created in below the SFC object, from a step by means of the *Exit action* step property. (3). You can also add a new action to the step by means of the *EAdd exit action* command. The exit action is marked with an X in the lower right corner of the step box.



3、Difference between IEC actions and step actions:

The basic difference between step actions and IEC actions with a qualifier N is that an IEC action is executed two times: when the step is activated and when the step is deactivated. See the following example.

Branch

Symbol

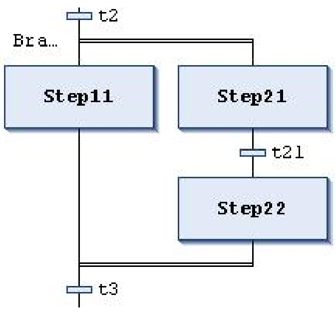
Use branches to program parallel or alternative sequences in the sequential function chart.

For alternative branches, CODESYS processes just one of the branch lines at a time, depending on the preceding transition condition. Parallel branches are processed at the same time.

1、Parallel branch

For parallel branches, the branch lines must begin and end with steps. Parallel branch lines can contain additional branches.

The horizontal lines before and after the branch are double lines.



Processing in online mode: If the preceding transition (t2 in the example) yields `TRUE`, then the first steps in all parallel branch lines are active (Step11 and Step21). CODESYS processes the individual branch lines at the same time and the subsequent transition is passed afterwards (t3).

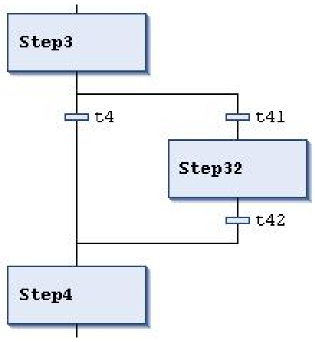
The “Branch<n>” jump marker is added automatically to the horizontal line that indicates the beginning of a branch. You can define this marker as the jump destination.

Please note that you can convert a parallel branch into an alternative branch by clicking *Alternative*.

2、Alternative branch

The horizontal line before and after the branch is a single line.

In an alternative branch, the branch lines must begin and end with transitions. The branch lines can contain additional branches.



If the step before the branch is active, then CODESYS passes the first transition of each alternative branch line from left to right. For the first transition that yields `TRUE`, the associated branch line opens, thus activating the step following the transition.

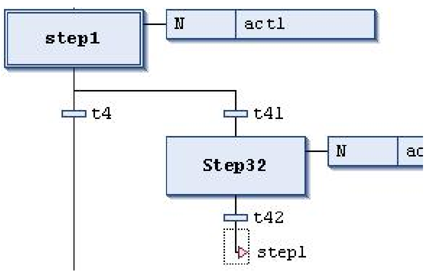
Jump

Symbol 

Use a jump to define which actions in a step should be executed next as soon as the transition preceding the jump is `TRUE`. Jumps may become necessary, as execution paths cannot cross or lead upwards.

Excluding the required jump at the end of a diagram, you can generally insert jumps only at the end of a branch.

The destination of a jump is defined by the added text string, which you can edit directly. The jump destination can be a step name or the marker for a parallel branch.



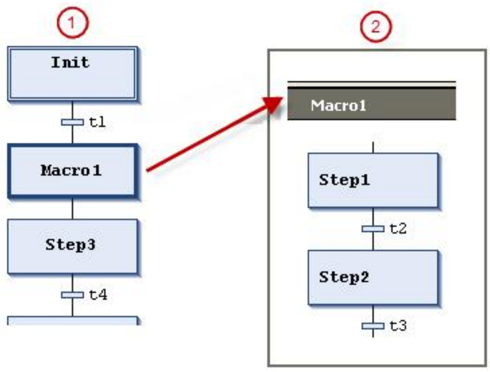
Macro

Symbol 

A macro includes part of the SFC diagram, but it is not displayed in detail in the main view of the editor.

Using macros does not influence the processing flow. Macros are used for hiding specific parts of the diagram, for example to increase overall clarity.

You open the macro editor by double-clicking the macro box or by clicking *SFC ‣ Zoom into macro* . You can program here just like in the main view of the SFC editor. To close the macro editor, click *SFC ‣ Zoom out of macro* .



① Main view in the SFC editor

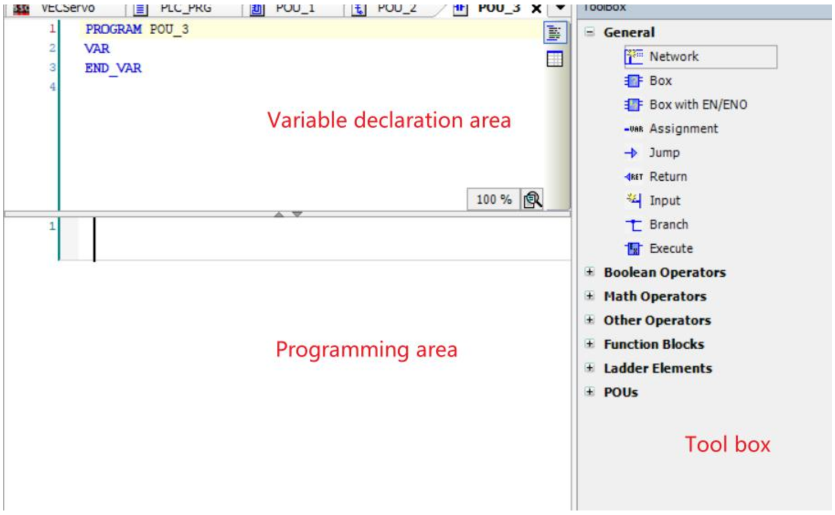
② Macro editor view for Macro1

Macros can also include other macros. The caption of the macro editor always shows the path of the open macro within the diagram, for example:

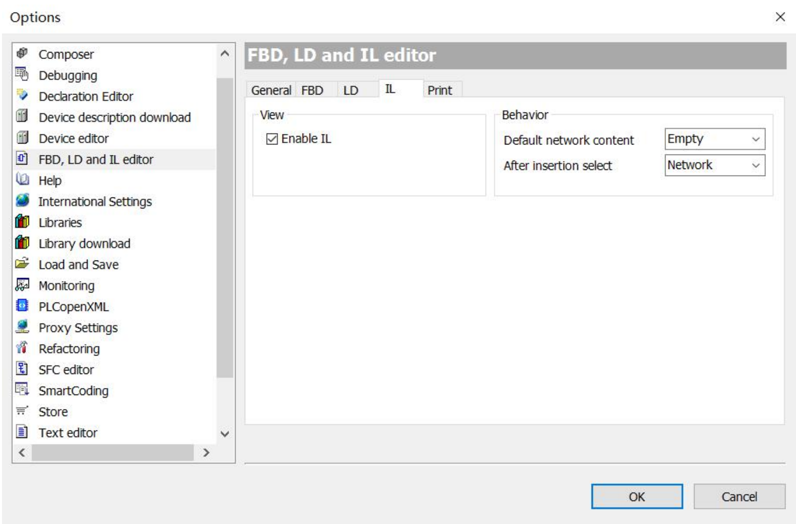
6.7 CFC/LD/IL

6.7.1 FBD / LD / IL Editor

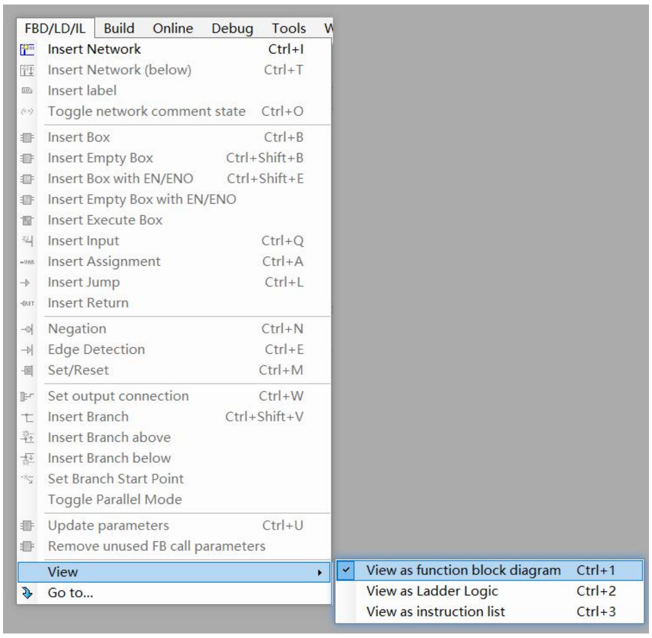
The FBD/LD/IL editor is a combination editor for FBD, LD and IL programming languages.



If necessary, you can activate IL by selecting "Enable IL" in the software option "Tools→Options→FBD, LD, IL Editor".



The three programming languages are automatically converted internally to each other. With the help of the network, the code in the implementation section is constructed in all three languages. The FBD / LD / IL menu provides commands to work in the editor. In offline and online mode, you can switch editors at any time by using menu commands in View.



6.7.2 FBD/LD/IL Element

Network

Symbol

A network is the base unit of an FBD or LD program. In the FBD/LD/IL editor, the networks are arranged in a list. Each network is provided with a sequential network number on the left side and can include: logical and arithmetic expressions, program/function/function block calls, jumps, or return statements.

An IL program consists of at least one network. This network can include all IL statements of the program.

You can provide each network with a title, comment, or label. In the CODESYS options (category *FBD, LD, and IL*, you can define whether network title, comment, and separator between individual networks are displayed in the editor.

Click the first line of the network to enter a network title. Click the second line of the network to enter a network comment.

Box

Symbol:

A box and its call can represent additional functions, for example IEC function blocks, IEC functions, library function blocks, operators.

A box can have any number of inputs and outputs.

If the box also provides an image file, the box icon is displayed inside the box. The requirement is that the option *Show box symbol* is activated in the CODESYS options, category *FBD, LD and IL*.

If you have changed the box interfaces, you can update the box parameters with the command *FBD/LD/IL ‣ Update parameters* without having to re-insert the box.

FBD/LD/IL Element 'Box with EN/ENO'

Symbol:

The element is available only in the FBD and LD editors.

The box generally corresponds to the FBD/LD/IL element *Box*; however, this box additionally contains an EN input and an ENO output. EN and ENO have the data type `BOOL`.

Function of the `EN` input and `ENO` output: if the input `EN` has the value `FALSE` at the time of the calling the box, the operations defined in the box are not executed. Otherwise, i.e. if `EN` has the value `TRUE`, these operations are executed. The ENO output has the same value as the EN input.

Assignment

Symbol:

The FBD editor shows a newly inserted assignment as a line with 3 question marks after it. The LD editor shows a newly inserted assignment as a coil with 3 question marks located above it.

After insertion you can replace the placeholder `???` by the name of the variable to which the signal coming from the left is to be assigned. The input assistant is available to you for this.

Input

Symbol:

The maximum number of inputs depends on the type of box.

A newly added input is first marked with `???`. You can replace the string `???` by a variable or a constant.

Label

Symbol:

The label is an optional identifier for a network in FBD and LD, which you can specify as a destination for a jump.

If you insert a jump label in a network, it will be added as an editable field *Label:* in the network.

Jump

Symbol 

In FBD or LD a jump is inserted either directly before an input, directly after an output or at the end of the network, depending on the current cursor position.

You enter a jump label as the jump destination directly behind the jump element.

Return

Symbol:

This element immediately interrupts the execution of the box if the input of the `RETURN` element goes `TRUE`.

In an FBD or LD network you can place the Return instruction parallel to or after the preceding elements.

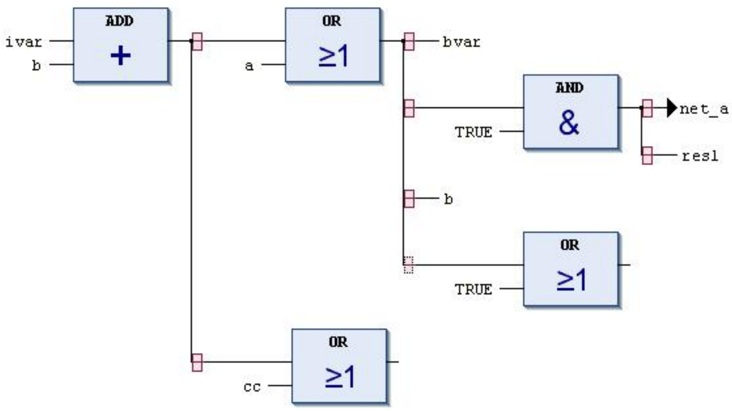
In IL the `RET` instruction is available to you for this purpose.

Branch

Symbol:

The element is available in the LD and FBD editor and represents an open line branch. A line branch splits the processing line from the current cursor position onwards into 2 subnetworks, which are executed in succession from top to bottom. You can branch each subnetwork further, as a result of which multiple branches are created within a network.

Each subnetwork is given a marker symbol (rectangle) at the branch point, which you can select in order to execute further commands.



In order to delete a subnetwork, you must first delete all elements of the network and then the marker symbol of the subnetwork.

Excute

Symbol:

The element is a box that enables you to directly enter ST code in the FBD and LD editors.

You can drag the *Execute* element with the mouse from the *Tools* view into the implementation part of your POU. If you click on *Enter ST code here...*, an input field opens where you can input multiple-line ST code.

Contact

Symbol: , in the editor

The element is available only in the LD editor.

A contact passes on the signal `TRUE` (ON) or `FALSE` (OFF) from left to right until the signal finally reaches a coil in the right-hand part of the network. For this purpose a boolean variable containing the signal is assigned to the contact. To do this, replace the placeholder ??? above the contact with the name of a boolean variable.

You can arrange several contacts both in series and in parallel. In the case of two parallel contacts, only one needs to obtain the value TRUE in order for ON to be passed on to the right. If contacts are connected in series, all of them must obtain the value TRUE in order for ON to be passed on to the right by the last contact in the series. Hence, you can program electrical parallel and series connections with LD.

A negated contact |/| forwards the signal TRUE if the variable value is FALSE. You can negate an inserted contact with the help of the command *FBD/LD/IL → Negation* or insert a negated contact from the *Tools* view.

If you place the mouse pointer on a contact with the left mouse button pressed and with a network selected, the button *Convert to coil* appears in the network. If you now move the mouse pointer onto this button, still with the mouse button pressed, and then release the mouse button over this button, CODESYS converts the contact into a coil.

Coil

Symbol: ( ), in the editor ( )

The element is available only in the LD editor.

A coil adopts the value supplied from the left and saves it in the boolean variable assigned to the coil. Its input can have the value TRUE (ON) or FALSE (OFF).

Several coils in a network can only be arranged in parallel.

In a negated coil (/) the negated value of the incoming signal is stored in the boolean variable that is assigned to the coil.

Set coil, Reset coil

Symbol: (S), (R) , in the editor: (S), (R)

Set coil: If the value TRUE arrives at a set coil, the coil retains the value TRUE. As long as the application is running, the value can no longer be overwritten here.

Reset coil: If the value TRUE arrives at a reset coil, the coil retains the value FALSE. As long as the application is running, the value can no longer be overwritten here.

You can define an inserted coil as a set or reset coil with the help of the command *FBD/LD/IL ‣ Set/Reset* or insert it as an element *Set coil* and *Reset coil* from the *Tools* view.

Branch Start/End 分支开始/结束

Symbol:

The element serves the closed line branch.

7 Motion control instructions

7.1 Motion control programming for single-axis MC instructions

7.1.1 MC instruction programming points

The motion control of the VE controller in conjunction with the servo axis (e.g. VECServo)is based onthe EtherCAT bus network and, unlike the pulse mode of the previous hardware output, is achieved entirely by software, specifically by calculating and releasing a control command in each very short EtherCAT bus cycle to achieve control of the servo. Therefore, the following points need to be noted:

◆ The user MC control program, which is executed in the EtherCAT task cycle, should be configured to execute under the EtherCAT task; most MC function blocks will not function properly if they are placed in the POU of the lower priority Main task;

◆ The execution of MC function blocks requires communication data objects in the communication to be passed on, therefore the necessary configuration items should be present in the PDO configuration table; if a configuration-related data object is omitted, the servo may not function properly and there will be no error alarms;

◆ The controller can initialize the function code of the servo through the configuration of SDO, making the operation mode of the servo (usually CSP mode), servo motor encoder mode, electronic gear ratio, etc., to ensure the correspondence between the control command and the physical operation position; the initialization of the servo can also improve the commissioning efficiency of the equipment, and no error after the replacement of parts;

◆ For the control of servo axes, the rules and logic of axis state transfer are followed, using the appropriate MC function block according to the current state of the axis and the desired motion;

◆ One MC instance can only be used for one servo axis, if it is used for several servo axes at the same time, it will lead to confusion.;

◆ A running servo axis must have an MC function block to monitor its operation, even MC_Stop is a kind of monitoring, to avoid that the system will stop and report an error due to a jump in the program logic without MC function block monitoring, which is not easy to check.;

◆ Pay attention to the safe handling of commissioning. If the servo system uses incremental encoders, a zeroing operation is required before normal operation, the DI signal input port of the servo drive can be connected to the home position signal, and for movements within a limited range (e.g. screw), there should be a limit and safety protection signal before commissioning。

7.1.2 MC function blocks commonly used for single-axis control

Mc Function Block (FB)isalsoknown as the MC instruction, and to be precise, the user program uses an object instance of the MC function block, which is controlled by the servo axis through the MC object instance. Single-axis control, generally used for positioning control, that is, the servo motor drags the external mechanism movement to a specified position, and sometimes also requires the servo to operate at a specified speed or torque, etc., in single-axis control, commonly used to the following MC function blocks:

Control operations	MC commands to be used	Description
Servo enable	MC_Power	Run this command to enable the servo axis before subsequent operation control can take place
Servo pointing operation	MC_Jog	Pointing operation of the servo motor, often used for low speed test runs, to check equipment or to adjust the position of the servo motor
Relative positioning	MC_MoveRelative	Runs a specified distance with the current position as a reference
Relative superimposed positioning	MC_MoveAdditive	The servo is run for a specified distance relative to the current servo run command
Absolute positioning	MC_MoveAbsolute	Command the servo to run to a specified coordinate point
Speed control	MC_MoveVelocity	Command the servo to run at the specified speed
Torque control	MC_MoveTorque	Command the servo to run at a specified torque
Servo pause	MC_Halt	Command the servo to pause, if MC_Movexxx is triggered again, the servo can run again.
Emergency stop	MC_Stop	Command the servo to emergency stop, only after the stop command is reset and MC_Movexxx is triggered, the servo can run again
Alarm reset	MC_Reset	When the servo has an alarm stop, run this command to reset the servo.
Change of operating mode	MC_ControlMode	This command allows the servo to select "position", "speed" or "torque" mode.
Servo home return	MC_Home	The servo is commanded to start a home return operation, with the application system's home signal, both limit signals, etc. connected to the servo's DI port
Controller home	MC_Homing	The control system starts the home return

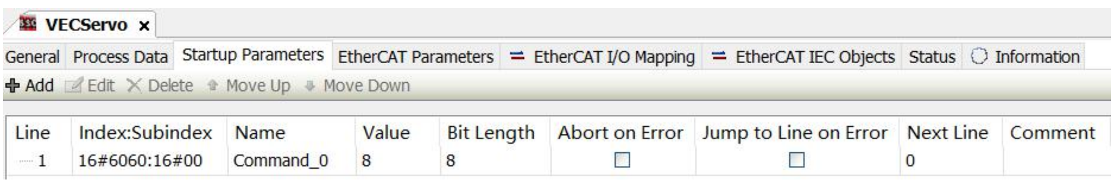
return		operation. The home signal of the application system and the limit signals on both sides are connected to the DI port of the controller.

7.1.3 MC commands and PDO/SDO configuration

When the VE controller executes the servo axis MC control commands from the user program, the information items required to interact with the servo during the execution of the MC commands need to be added to the communication PDO/SDO configuration table in order to perform the required control functions.

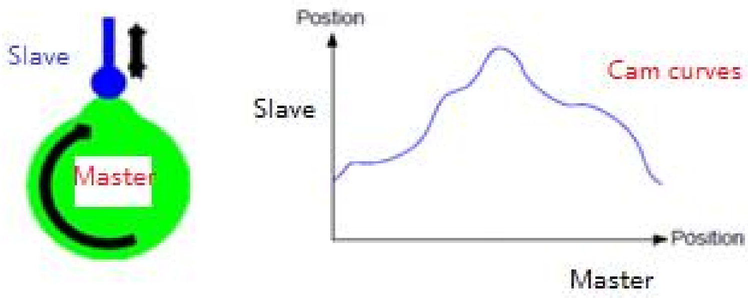
MC Directive	Required TPDO objects	Required RPDO objects
MC_Power MC_Halt MC_Stop MC_Reset MC_Home MC_Homing	ControlWord	StatusWord Errorcode
MC_Jog	TargetPosition	Position actual value Following error actual value
MC_MoveRelative		
MC_MoveAdditive		
MC_MoveAbsolute		
MC_MoveVelocity	Target velocity Max profile velocity	
SMC_SetTorque	Target torque	Torque actual value
SMC_SetControllerMode	Modes of operation	1: Cycle Synchronous Torque Mode CST 2: Cycle Synchronous Velocity Mode CSV 3: Cycle synchronous position CSP

The above mentioned TPDO and RPDO are the basic configuration items required for single axis control. In MC control, the servo is in position mode in most cases, especially in EtherCAT bus based applications, and is in "cyclic synchronous position mode", so the servo is normally initialised to this mode of operation in the SDO configuration during programming.



7.2 Motion control programming for multi-axis CAM cam synchronization

Cam motion is borrowed from the concept of the relative motion characteristics of mechanical cam and lift bar, refers to the controller according to a specific relative position nonlinear relationship, so that the servo follows the spindle, continuous synchronization motion to meet the motion characteristics required by the equipment, in the fixed length cutting, shear control, fly shear control, multi-color overprinting and other synchronous applications, is widely used. The main-axis position of the electronic cam curve is shown below, with the horizontal axis as the primary axis position and the vertical axis as the axle position:



VE controller is a software way, the realization of cam motion control characteristics, that is, the use of software digital "cam meter" instead of mechanical cam, so also known as electronic cam control. Compared to mechanical cams, there are the following features:

- Easy production of cam shapes: cams are described using cam tables, cam curves or arrays;
- Easy and versatile cam shapes: multiple cam table selection, dynamic switching during operation;
- Easy cam shape modification: allows dynamic modification of cam table key points during operation;
- Multiple cam followers: multiple cam followers allowed;
- Cam tappets: multiple cam tappets, multiple setting intervals allowed;
- Cam clutch: in cam operation, user programmable to enter and exit cam operation;
- Electronic cam specific features: virtual spindle support, phase shift, output superimposition;

The VE controller's cam operation is software-only, and if the CAM is running, the next target point from the shaft is calculated once each time the EtherCAT task is entered, so it has better functional flexibility than hardware cam operation.

The control of the electronic cam has three elements:

(1) Spindle: a reference axis used for synchronous control;

(2) From the axis: the servo axis that follows the movement according to the desired

nonlinear characteristics according to the position of the spindle;

(3) Cam table: Describes the spindle - a data sheet or cam curve from the relative position and range of the axis, periodicity, etc.

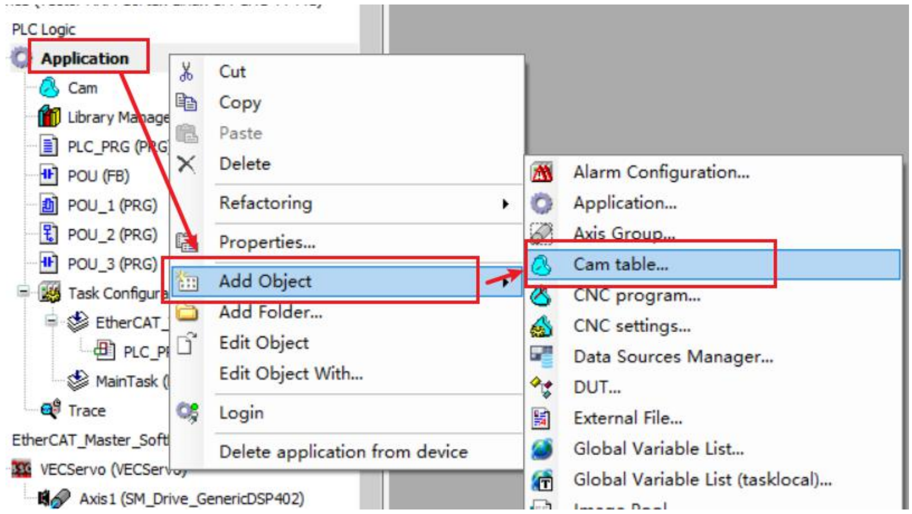
The user writes the program needs to design the cam table, specify the spindle and the axle, trigger the cam run at the right time during operation, so that the cam from the shaft into the cam run.

Basic command function blocks for electronic cam control：

Control operation	MC commands to be used	Description
Cam table selection	MC_CamTableSelect	Run this command to relate the spindle, slave cam table and all three
Entering cam run	MC_CamIn	Putting the slave shaft into cam operation
Exit cam run	MC_CamOut	Getting the slave axis out of the cam run
Correction of cam phase	MC_Phasing	Spindle phase modification

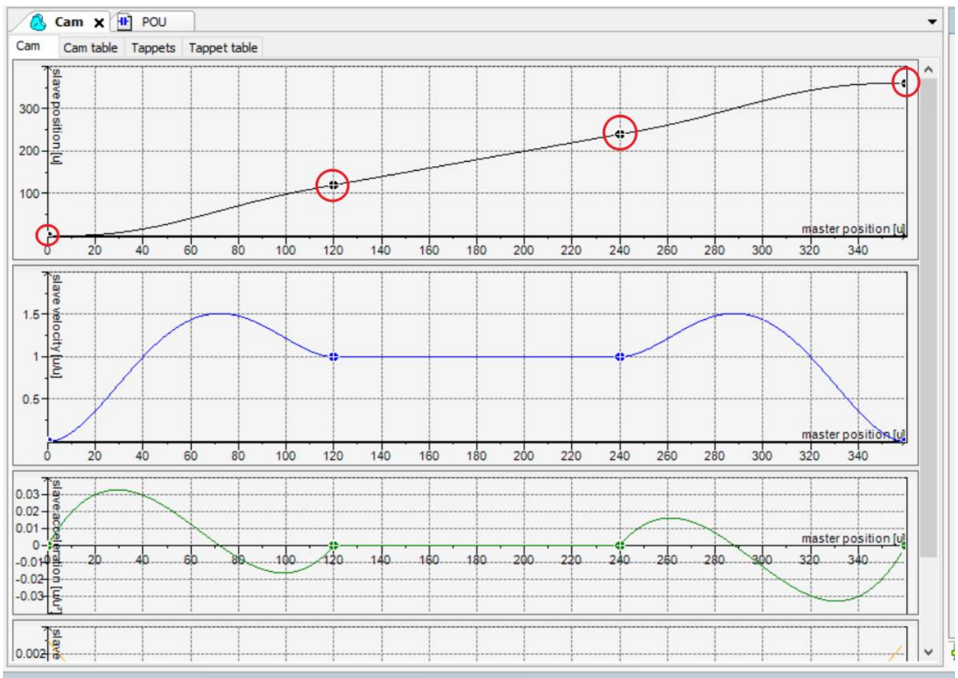
7.2.1 Characteristics of the cam table

The new cam table can be created as follows:



When writing a user program for cam operation, the cam table is one of the items written which determines the characteristics of the cam operation and can be entered in both graphical and tabular form.

The diagram below shows the CAM cam table in graphical form, with the horizontal axis being the spindle position axis and the length of the axis being the travel of the cam run. There are four coordinate curves and the vertical axes are the slave axis position, slave axis speed, slave axis acceleration and slave axis acceleration curves. When programming and commissioning, more attention is often paid to the position curve, the velocity curve and, when commissioning for smoothness, the acceleration curve.



The cam curve has the following characteristics:

◆ In the coordinates of the master-slave position curve, the vertical axis is the range of possible motion of the slave axis; the vertical axes of the other three curves are the ratio of the velocity of the slave to the main axis and the ratio of the acceleration of the slave to the master;

◆ The cam curve is a monotonic curve in the vertical direction, i.e. each coordinate of the main axis can only correspond to a unique coordinate value of the slave axis; when the cam is executed, the main axis coordinates move in a small to large direction;

◆ The cam curve can have a number of key points and the line between two key points can be set as a straight line or a 5-times curve, with the system optimising each 5-times curve to minimise sudden changes in speed and acceleration;

◆ The start and end coordinates of the horizontal axis (spindle) start at 0 and end at 360 by default and can be modified by the user according to the actual physical travel.

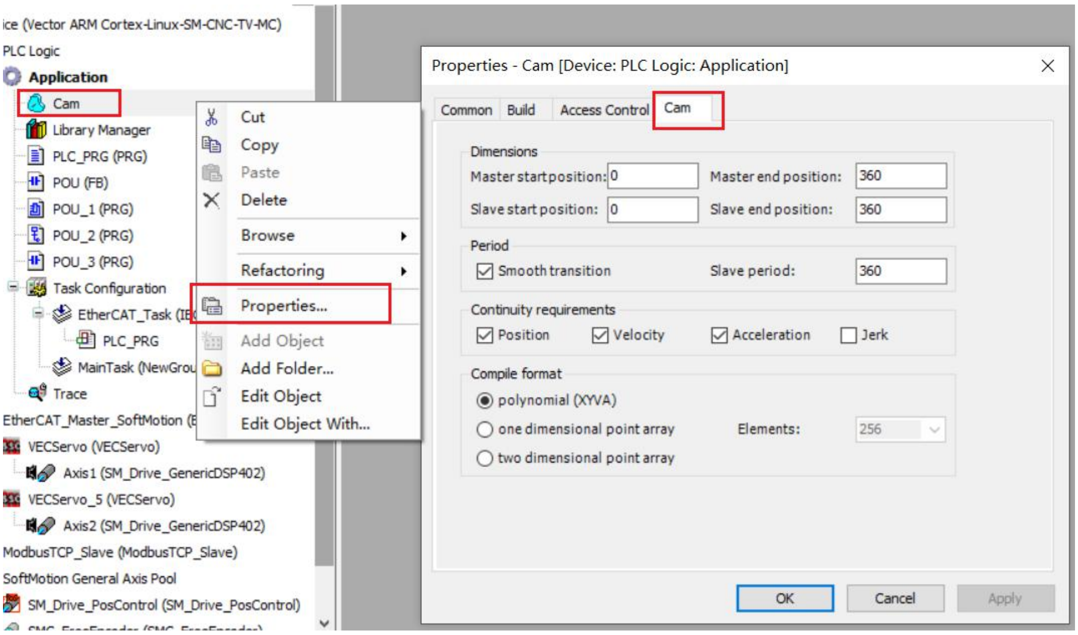
7.2.2 Cam table input

① When a new cam table is created, the system automatically sets up the simplest cam curve, on the basis of which the user can modify it to form the CAM curve table he needs.

② The user can increase or decrease the number of key points of the cam curve and modify the coordinates of the key points.

③ The user can modify the line type between any two adjacent key points, or a 5th curve, or a straight line;

④ The system defaults to a 5-fold curve between key points in the cam curve, which ensures continuity of speed during operation and reduces mechanical shocks.;



Key points in the cam curve, often related to the mechanical movement requirements of the control object, for example

① For chasing shear applications, the coordinate range of the spindle is recommended to correspond to the physical travel of the running interval for easy analysis.

② The starting and ending points of the round trip from the spindle, the starting position point of the synchronous running interval and the point at which it leaves the synchronous position are important key points.

③ The line segments of the cam curve should be straight lines for proportional synchronisation intervals and 5 times curves for other intervals

7.2.3 The internal data structure and array of the CAM cam table

In CODESYS, for each CAM table, there is a data structure that describes the CAM table, describing the characteristic data of the CAM table. The following image describes the data structure of the CAM0 cam table, note the variable names of its structures:

监视 1

表达式	应用	类型	值	准备值	执行点	地址	注释
Cam0	Device.Application	MC_CAM_REF			循环监测		
wCamStructID		WORD	56372		循环监测		By menas of this variable,...ich always has a consta...
byType		BYTE	3		循环监测		Describes the cam type, t... is the way in which the...
byVarType		BYTE	0		循环监测		Only used if byType=1 or byType=2
xStart		LREAL	0		循环监测		Start position of the mast...efining the range of ma...
xEnd		LREAL	360		循环监测		End position of the maste...fining the range of mas...
nElements		INT	5		循环监测		Number of elements that is depending on cam type ...
nTappets		INT	0		循环监测		Number of tappet switch actions.
pce		POINTER TO...	16#0000FFFF8C...		循环监测		Pointer to actual data element which type depends ...
pt		POINTER TO...	16#0000000000...		循环监测		
dwTappetActiveBits		DWORD	0		循环监测		Internal variable
strCAMName		STRING	'Cam0'		循环监测		
byInterpolationQuality		BYTE	1		循环监测		1: Linear interpolation, 3: Cubic interpolation
byCompatibilityMode		BYTE	0		循环监测		Compatibility mode: Bit0:...TRUE``: Periodic execu...
bChangedOnline		BOOL	FALSE		循环监测		Internal variable
xPartofLM		BOOL	TRUE		循环监测		``TRUE``: Generated by programming system -> ...

监视 1 断点

Inside there is a data structure to describe the characteristics of the CAM cam table: if we write a CAM table manually, it is also available, as follows:

Although we don't need to write CAM tables manually, we can modify the required CAM feature data through access to the data structure.

Note: When we declare the CAM0 cam table, the system automatically declares the CAM0 data structure of the global variable type by default, and also declares CAM0_A an array of

For example, in a user program, modify the number of key points or coordinates of the CAM0 cam table：

```
CAM0. nElements:=20; // Change the number of key points to 20
CAM0. xEnd:=500; // Change the end point of the spindle to 500.
// For example, in the user program, modify the coordinates of 2 of the key points：
CAM0_A[3].dx:=30;
CAM0_A[3].dy:=45;
CAM0_A[3].dv:=1;
CAM0_A[3].da:=0;
```

```
CAM0_A[4].dx:=60;
CAM0_A[4].dy:=75;
CAM0_A[4].dv:=1;
CAM0_A[4].da:=0;
```

Online modification of CAM camsheets

On-line modification of CAM curves" refers to the modification of the coordinates of the key points of the CAM curve during the execution of the user written program, according to the control characteristics. The modifications are usually made to the coordinates of the key points, but can also be made to the number of key points, to the distance range of the spindle, etc.

Reminder: Modify the cam table before entering the cam run, not during the run to avoid unanticipated movement results that require modification of the CAM cam table Applications.

① In general, OEM customers use cam tables that have been successfully verified by commissioning.

② If there are several machining objects or modes, multiple cam tables can be considered to be preset and switched automatically according to the needs of the user.

③ Some machines require a wider range of adaptability, e.g. packaging machines, which require a packaging length in the range of 10cm to 25cm and automatic adaptation to changes in operating speed, may require online modification of the CAM cam table.

7.2.4 Reference and dynamic switching of CAM table

C The CAM cam table is stored internally in the controller as an array that can be pointed to by a specific MC_CAM_REF variable type, e.g. by declaring

Cam table p: MC_CAM_REF

This variable can be assigned a value and can also be thought of as pointing to a specific cam table as follows

Cam table p:= Cam0; // point to the desired cam table

```
cam table p: MC_CAM_REF; // cam table pointer.
TableID: uint; // cam table selection command, settable by HMI.
Case TableID of
0: cam table p := cam table A;
1: Cam table p := Cam table B;
2: Cam table p := Cam table C;
End_case
MC_CamTableSelect_0( // Cam relationship
Master:= cam master ,
Slave:= cam slave ,
CamTable:= Cam table p,
Execute:= ReSelect, // Rising edge triggers cam table selection
Periodic:= TRUE,
```

```
MasterAbsolute:= FALSE,
SlaveAbsolute:= FALSE);
```

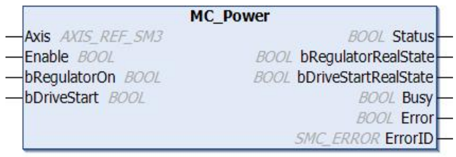
In the above example, the assignment of the MC_CAM_REF variable is used to switch between multiple cam tables.

7.3 Single axis commands

7.3.1 MC_Power

1) Command Format

Instructions	Name	Graphical representation	ST Performance

MC_Power	Axis enable command		`MC_Power(`
			`Axis:= ,`
			`Enable:= ,`
			`bRegulatorOn:= ,`
			`bDriveStart:= ,`
			`Status=> ,`
			`bRegulatorRealState=> ,`
			`bDriveStartRealState=> ,`
			`Busy=> ,`
			`Error=> ,`
			`ErrorID=> );`

2) Related variables

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Axis	Axis	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3
Enable	Input active	BOOL	TRUE,FALSE	FALSE	Set to TRUE to start function block processing
bRegulatorOn	Enabled state	BOOL	TRUE,FALSE	FALSE	Set to TRUE to set the axis to the enable state
bDriveStart	Drive allowed	BOOL	TRUE,FALSE	FALSE	Set to TRUE to disable emergency stop processing of the function block

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Status	Operable state	BOOL	TRUE,FALSE	FALSE	Set to TRUE if the axis is ready to move
bRegulatorRealState	Axis enable signal status	BOOL	TRUE,FALSE	FALSE	TRUE when axis enable is active
bDriveStartRealState	Permissible drive status	BOOL	TRUE,FALSE	FALSE	TRUE if the axis is not interrupted by the fast stop mechanism
Busy	Execution in progress	BOOL	TRUE,FALSE	FALSE	TRUE if the processing of the

					function block has not been completed
Error	Error	BOOL	TRUE,FALSE	FALSE	TRUE if an exception occurs
ErrorID	Error Codes	SMC_ ERROR	Refer to SMC_ ERROR	0	Error code output when an exception occurs

3) Function description

The other inputs are only processed by the function block if the input Enable is TRUE.

If the function block MC_Power has already been called and bRegulatorOn=FALSE, the function block sets the axis state (nAxisState) of the relevant axis to the power_off state, indicating that the drive is not yet ready for motion.

If the function block MC_Power has been called and bRegulatorOn=TRUE, the function block will set the axis state (nAxisState) of the relevant axis to the standstill state if no errors have occurred; if errors have occurred, the corresponding error state will be output.

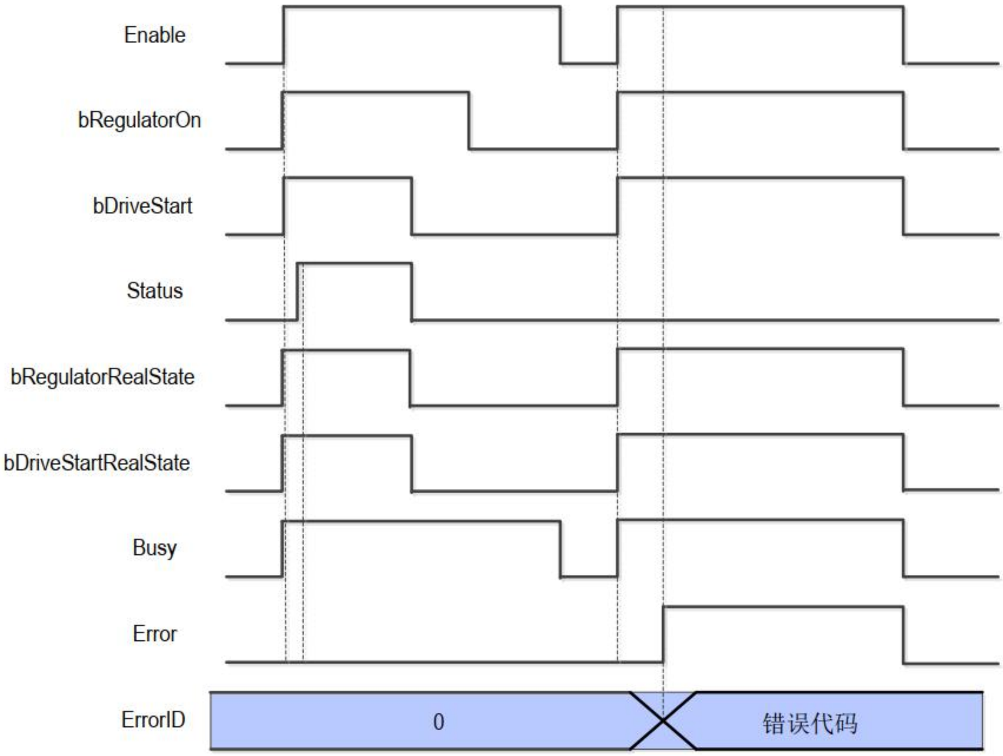
If Enable, bRegulatorOn and bDriveStart are TRUE, but the output Status remains FALSE after a certain time, the output Error will be set. This can happen when a hardware problem is generated in the enable state.

If the enable signal is lost (usually in operating mode), the nAxisState of the relevant axis will be set to the ErrorStop state.

When using this, note the order of operation of Enable and bRegulatorOn. Enable can be held high to control servo enable and disable by controlling bRegulatorOn. Do not turn Enable and bRegulatorOn on and off at the same time. If Enable is disabled, the function block will no longer be executed and changing bRegulatorOn will not take effect, resulting in the "servo is still enabled even though bRegulatorOn has been reset" phenomenon. This will lead to the phenomenon that "the servo is still enabled even though bRegulatorOn has been reset".

◆ Time-series diagram

Set Enable to TRUE, bRegulatorOn to TRUE and bDriveStart to TRUE, indicating that the busy command is being processed becomes TRUE, then the axis enters the Enable ON state and the Status state becomes TRUE.



4) Error description

Do not write a program to start another instance of the MC_Power instruction in the axis where the MC_Power instruction is being executed. In principle, only 1 MC_Power instruction can be set for each axis. If the MC_Power instruction of another instance is started in the axis in which the MC_Power instruction is being executed, the MC_Power instruction that is executed later will be executed first.

Note]: Please read "Appendix C Error Code Descriptions" for the descriptions of the relevant error codes.

7.3.2 MC_Stop

MC_Stop puts the axis in the stop state. The currently running motion of the function block instance is aborted.

1) Command Format

Instructions	Name	Graphical representation	ST Performance

MC_Stop	Axis stop command		`MC_Stop(`
			`Axis:= ,`
			`Execute:= ,`
			`Deceleration:= ,`
			`Jerk:= ,`
			`Done=> ,`
			`Busy=> ,`
			`Error=> ,`
			`ErrorID=> );`

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	Axis	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Deceleration	Deceleration rate	LREAL	"Positive numbers" 、" 0"	0	Deceleration of the function block (u/S^2)
Jerk	Rate of change of speed	LREAL	"Positive numbers" 、" 0"	0	Rate of change of velocity (u /S^3)

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Command execution in	BOOL	TRUE,FALSE	FALSE	TRUE if the current instruction is in progress

	progress				
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE when an exception occurs
ErrorID	Error code	SMC_ ERROR	参阅 SMC_ ERROR	0	Error code is output when an exception occurs

3) Function description

This function block is designed to stop the motion of an axis in normal operation, any command to this axis is not valid when the axis is in the stopping state.

This function block can only be run when the axis is in Motion, but not in any other state. The start command is initiated on the rising edge of Execute. If Busy is active while MC_Stop is active, starting MC_Stop again will cause the command system to change to Errorstop.

◆Timing diagram

the axis must be in the running state (Motion) for the MC_Stop instruction to run.

Execute of the function block must have a rising edge condition.

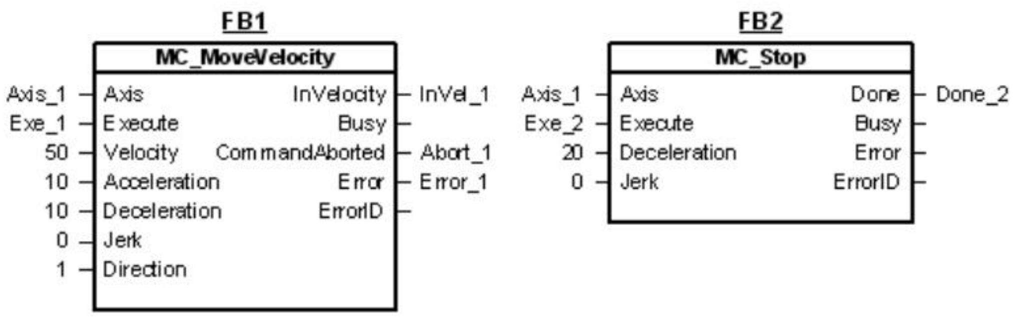
Done of the function block indicates that the instruction has been executed normally.

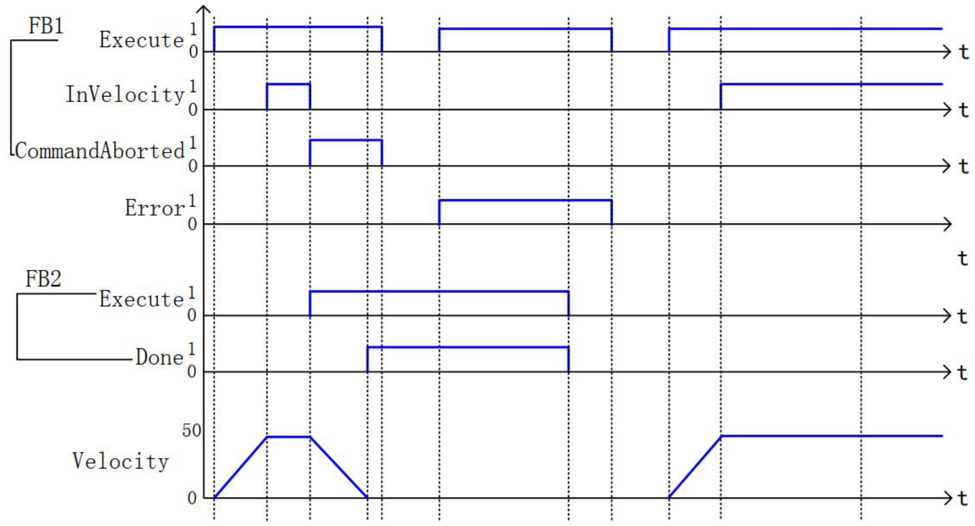
A Busy function block indicates that the function block is currently being executed.

CommandAborted of the function block indicates that the instruction is interrupted by another motion control instruction, and the flag bit is TRUE.

Example: The change of the flag bit during the execution of MC_MoveVelocity instruction and MC_Stop instruction in different timing operations;

The processing of CommandAborted is described in the following timing diagram .





4) Error Description

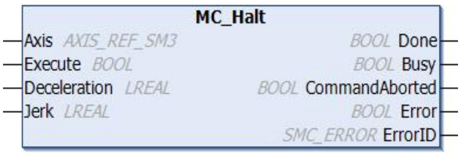
When MC_Stop has a repetitive instruction running, the error flag Error is True,ErrorID is SMC_MS_AXI error ;

Note]: Please read "Appendix C Error Code Descriptions" for the description of the relevant error codes.

7.3.3 MC_Halt

1) Command Format

Instructions	Name	Graphical representation	ST Performance

MC_Halt	Axis normal pause command		`MC_Halt(`
			`Axis:= ,`
			`Execute:= ,`
			`Deceleration:= ,`
			`Jerk:= ,`
			`Done=> ,`
			`Busy=> ,`
			`CommandAborted=> ,`
			`Error=> ,`
			`ErrorID=> );`

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Deceleration	Deceleration	LREAL	“Positive numbers” +” 0”	0	Deceleration of the function block (u/S^2)
Jerk	Leap	LREAL	“Positive numbers” +” 0”	0	Specify the degree of jump [command unit /S^3

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Instruction execution in progress	BOOL	TRUE,FALSE	FALSE	The current instruction is in progress, set to TRUE
Command Aborted	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs

ErrorID	Error code	SMC_ ERROR	Refer to SMC_ ERROR	0	Error code is output when an exception occurs

3) Function description

This function block stops the reference axis in a controlled manner. If the actions of other function blocks are running at this time, these actions are aborted. The axis enters a discrete motion state until the speed reaches 0. If the "Finish" output of MC_Halt is set, the state of the axis will change to stationary. The execution of MC_Halt can be interrupted by issuing a new motion command as long as MC_Halt is active, unlike MC_Stop, which can be interrupted.

This function block can only be run in the running state (Motion), but not in any other state.

The start command is initiated on the rising edge of Execute; the state of the command is Discrete Motion while it is running and Standstill when it is finished.

4) Timing diagram

the axis must be in the running state (Motion) for the instruction to run.

Execute of a function block must have a rising edge condition.

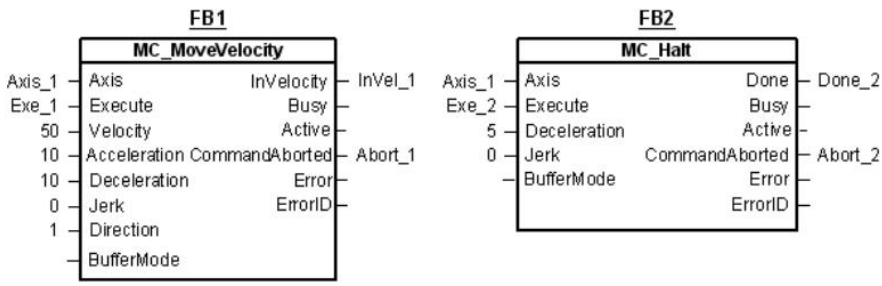
Done for a function block indicates that the instruction is executing normally.

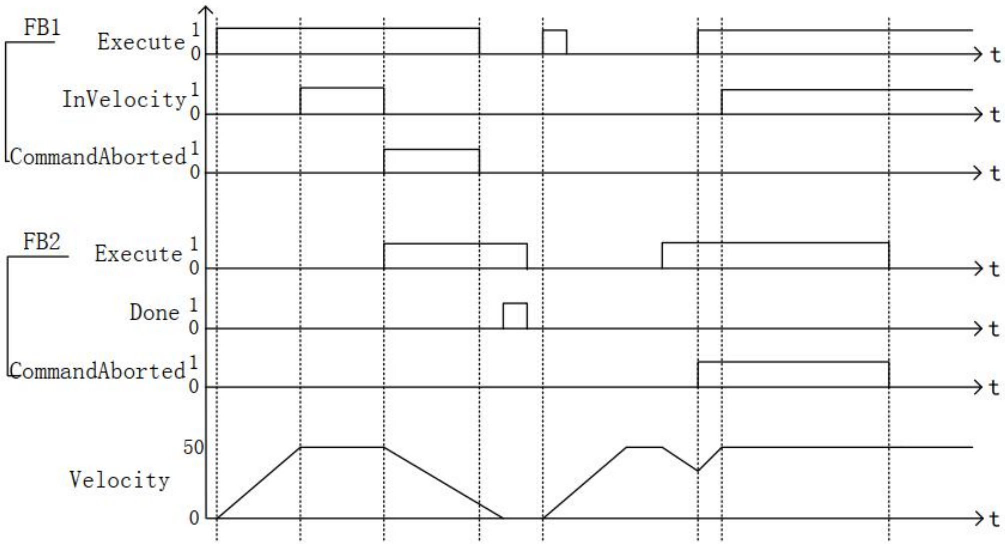
Busy of a function block indicates that the block is currently being executed.

CommandAborted of the function block indicates that the instruction is interrupted by another motion control instruction, and the flag is TRUE.

Example: The change of the flag bit during the execution of MC_MoveVelocity instruction and MC_Halt instruction in different timing operations;

The processing of CommandAborted is described in the following timing diagram。





5) Error description

An error occurs when the axis state is not a parameter error in the start-up instruction or instruction system in Standstill, and the axis error can only be cleared before operation starts.

[Note]: Please read "Appendix C Error Code Descriptions" for a description of the relevant error codes.

7.3.4 MC_Home

Its execution will cause the axis to perform the "search home" sequence. The details of this sequence are manufacturer dependent and can be set by the axis parameters. The Position input is used to set the absolute position when a reference signal is detected. The function block terminates with standstill.

1) Command Format

Instructions	Name	Graphical representation	ST Performance
MC_Home	Axis return to zero command		MC_Home(Axis:= Axis, Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Position	Axis reaches position	LREAL	Data range	0	Represents the zero return position of the axis position

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Instruction execution in progress	BOOL	TRUE,FALSE	FALSE	Execution of current instruction is in progress, set to TRUE
Command Abort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception

					occurs
ErrorID	Error code	SMC_ ERROR	Refer to SMC_ ERROR	0	Error code is output when an exception occurs

3) Function description

This function block is a zero return operation, where the Position data is the zero position of the axis.

The running state of this function block is in Standstill, the state of the instruction is homing when it is running, and no other state can be run.

The start command is the rising edge start command of Execute.

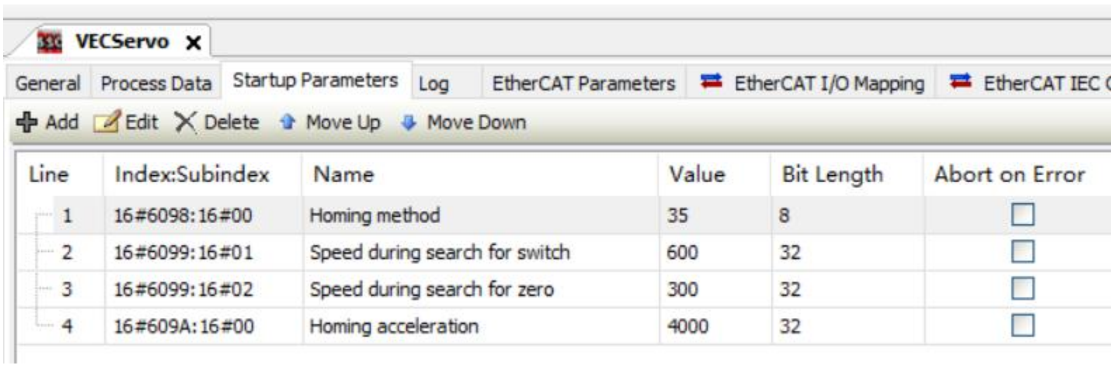
Viktor servo setting instructions :

◆ When using each servo axis to return to the home position, the return mode of the servo parameter must be set; the setting mode can be set manually by setting the function code of the servo;

◆ The corresponding function code can also be configured via the start parameters of the VE slave; the following index and subindex data must be set for the communication method;

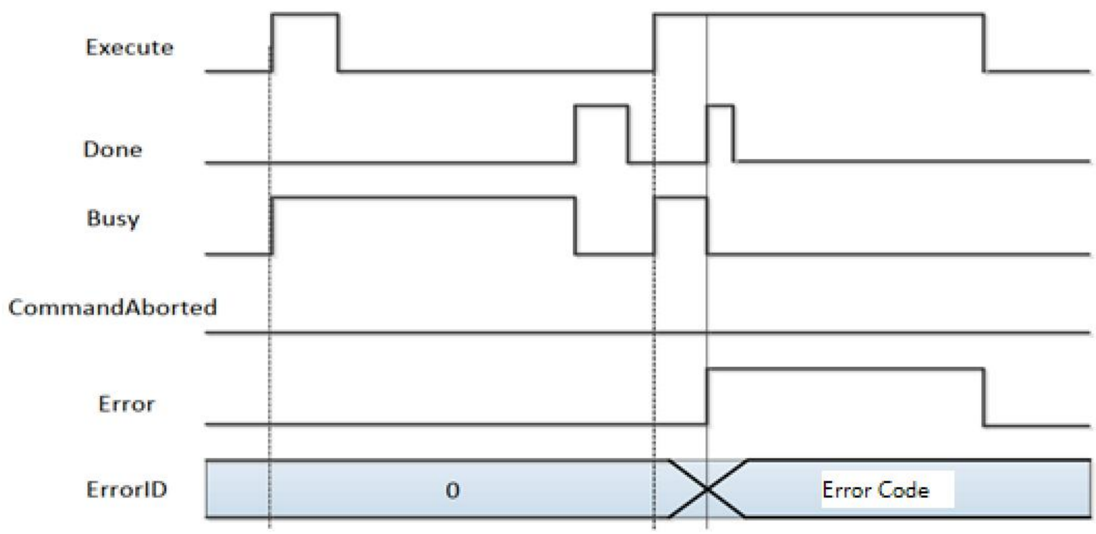
Project	index	Sub-indexes	Description
Zero return method	0x6098		The specific parameters to be set can be selected according to the servo manual
Home velocity	0x6099	0x01	Generally defined speeds are relatively high, with reduced zeroing times in rpm
Find zero velocity	0x6099	0x02	Generally defined speeds are relatively low, in rpm
Home return acceleration/deceleration	0x609A		change in acceleration and deceleration at home return, in u/s^2
Home return timeout	0x200A	0x08	The return time exceeds the set time and the system reports an Er.603 error.

CODESYS screen settings reference :



Line	Index:Subindex	Name	Value	Bit Length	Abort on Error
1	16#6098:16#00	Homing method	35	8	☐
2	16#6099:16#01	Speed during search for switch	600	32	☐
3	16#6099:16#02	Speed during search for zero	300	32	☐
4	16#609A:16#00	Homing acceleration	4000	32	☐

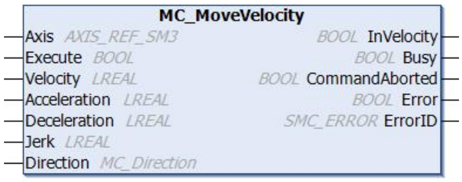
4) Timing diagram


Execute
Done
Busy
CommandAborted
Error
ErrorID
0
Error Code

7.3.5 MC_MoveVelocity

Simulated velocity control using the drive position control mode, where the Velocity assignment controls the speed of the drive if the axis is enabled and the command is valid.

1) Command Format

Instructions	Name	Graphical representation	ST Performance
MC_MoveVelocity	Speed control commands		MC_MoveVelocity(Axis:= , Execute:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , InVelocity=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Velocity	Speed	LREAL	Data range	0	This data is the speed run value for this instruction
Acceleration	Acceleratio n	LREAL	Data range	0	Acceleration value as the speed becomes greater
Deceleration	Deceleratio n	LREAL	Data range	0	Deceleration value as speed becomes smaller
Jerk	Leap	LREAL	Data range	0	The value of the slope change of the acceleration and deceleration curve
Direction	Direction	MC_Direction	1：positive	current	Command operation

	of travel		-1: negative 2: current		for the direction of travel

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
InVelocity	Set speed flag reached	BOOL	TRUE,FALSE	FALSE	The set running speed has been reached, set to TRUE
Busy	Instruction being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is being executed, set to TRUE
CommandAbort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs
ErrorID	Error code	SMC_ ERROR	Refer to SMC_ ERROR	0	Error code is output when an exception occurs

3) Function Description

Changes the Velocity parameter for the analog speed control of the drive.

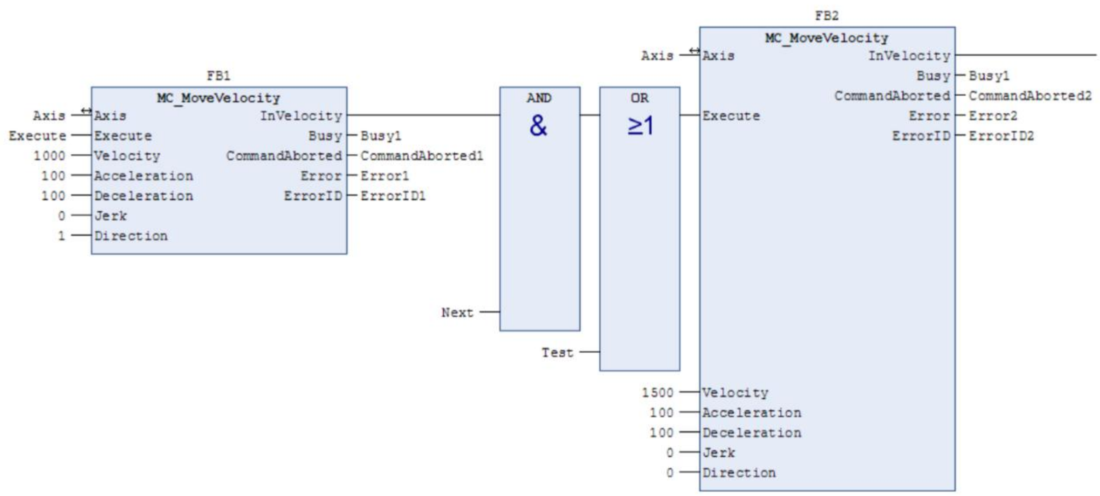
◆ Timing diagram

Execute of the function block must have a rising edge condition

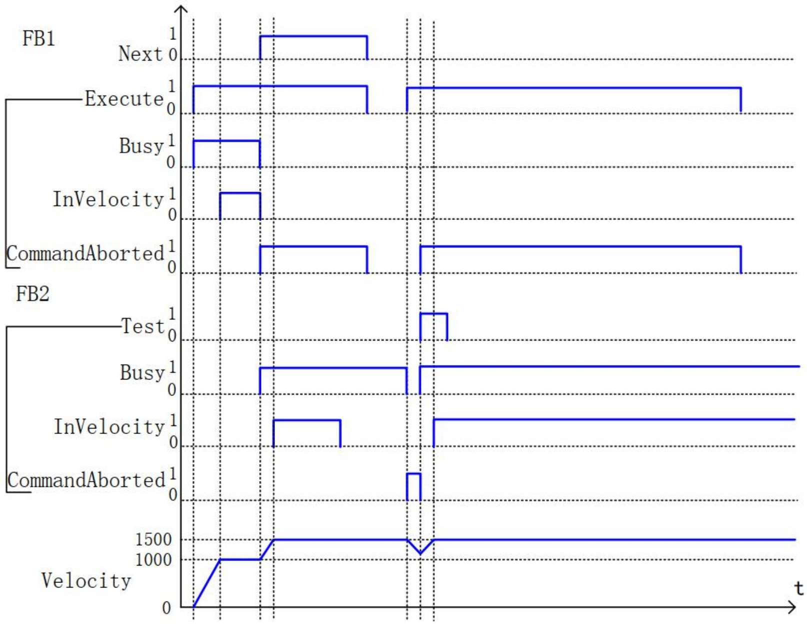
InVelocity of the function block indicates that the running speed of the instruction has reached the set value.

Busy of a function block indicates that the function block is currently being executed.

◆ Examples



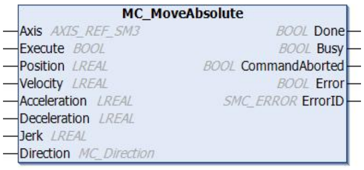
◆ Timing instructions :


FB1
Next
Execute
Busy
InVelocity
CommandAborted
FB2
Test
Busy
InVelocity
CommandAborted
Velocity
1500
1000
0
t

7.3.6 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. If no further actions are pending, the execution ends with velocity 0.

1) Command Format

<table>
<tr><th>Instructions</th><th>Name</th><th>Graphical representation</th><th>ST Performance</th></tr>
<tr><td>MC_MoveAbsolute</td><td>Axis absolute position control commands</td><td>
</td><td>

```
MC_MoveAbsolute(
    Axis:= ,
    Execute:= ,
    Position:= ,
    Velocity:= ,
    Acceleration:= ,
    Deceleration:= ,
    Jerk:= ,
    Direction:= ,
    Done=> ,
    Busy=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> );
```

</td></tr>
</table>

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will start the function block
Position	Axis arrival position	LREAL	Data range	0	This position is the absolute position data of the axis
Velocity	Operating speed	LREAL	Data range	0	Maximum speed at which the axis runs to the target position
Acceleration	Acceleratio n	LREAL	Data range	0	Acceleration value as speed increases
Deceleration	Deceleratio n	LREAL	Data range	0	Value of deceleration as speed becomes smaller
Jerk	Leap	LREAL	Data range	0	Value of the change in slope of the acceleration/deceleration curve
Direction	Direction of command	MC_ DIRECTION	Negative, shortest Positive,	shortest	Negative: Reverse movement ; Shortest: Choose the direction based on the shortest path ;

			current, fastest		Positive: move in the positive direction ; Current: Move in the current direction; Fastest: automatically selects the fastest direction of travel (This function is available in rotation mode)

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Instruction execution in progress	BOOL	TRUE,FALSE	FALSE	Execution of current instruction is in progress, set to TRUE
Command Abort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs
ErrorID	Error code	SMC_ ERROR	Refer to SMC_ERROR	0	Error code is output when an exception occurs

3) Function Description

◆ This function block is an absolute axis positioning instruction, and the Position data is the absolute position of the axis.

This function block is in Standstill, and the state of the command is Discrete Motion.

A complete running process must control the different motion states of the axes.

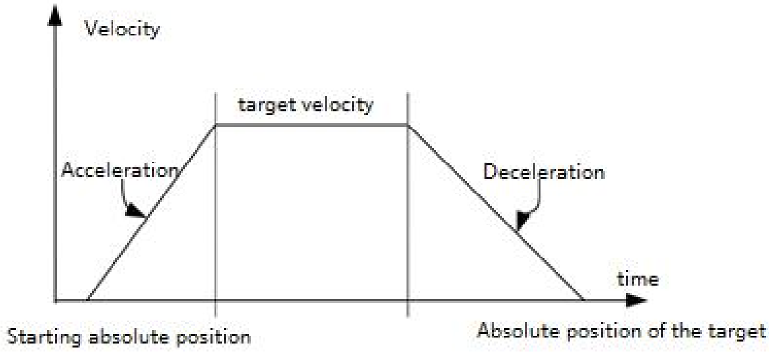
This instruction is valid for repeated rising edges in Discrete Motion, each time refreshing the latest position.

This command is valid on the rising edge of Discrete Motion.

If Acceleration or Deceleration is zero, the command will run in an abnormal state, but the state of the axis is Discrete Motion.

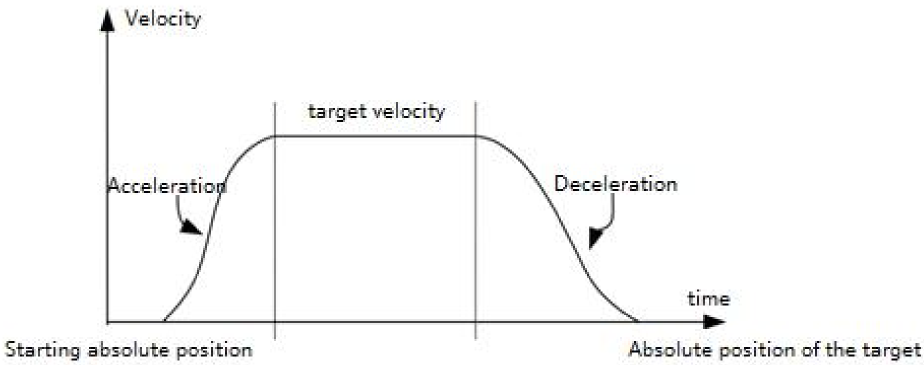
◆ Trapezoidal acceleration and deceleration actions

Velocity, Acceleration and Deceleration have data; and Jerk is 0;

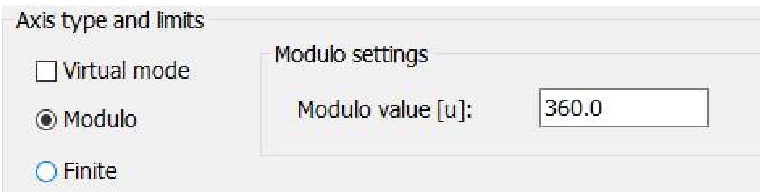


◆ S-curve acceleration and deceleration manoeuvres

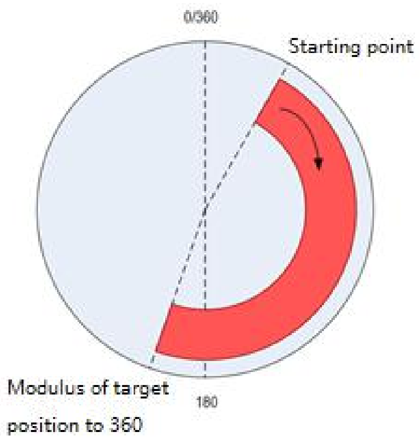
Velocity, Acceleration, Deceleration and Jerk all with data；



◆ **Absolute positioning of the axes in cyclic mode**



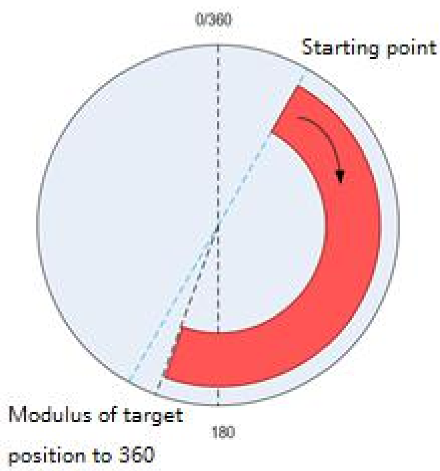
① The axis rotation period is set to 360 and the Direction is set to Positive. When the modulus of Position to 360 (Position/360 is rounded off, e.g. Position 380 is modulus 20 to 360, Position 350 is modulus 350 to 360) > Start absolute position, then the axis runs in the forward direction (modulus of Position to 360 - Start absolute position) by a distance.



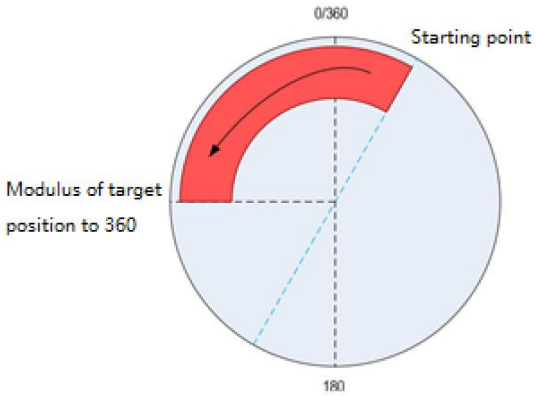
When the modulus of Position to 360 (Position/360 is remainder, e.g. Position is 380 then modulus to 360 is 20) < Start absolute position, then the axis runs in the forward direction (360 - Start absolute position + modulus of Position to 360) by a distance.

② The axis rotation period is set to 360 and the Direction is set to shortest or fastest. The modulus of Position to 360 is XPosition

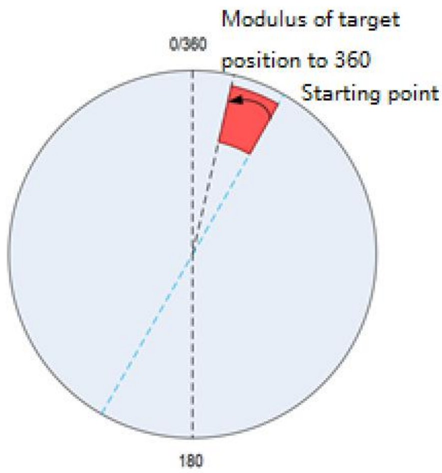
When 0 < XPosition - Start absolute position < 180, the distance of the axis in the forward direction (XPosition - Start absolute position).



When 180 < XPosition - starting absolute position, the axis runs in the opposite direction 360 - XPosition + distance from the starting absolute position.

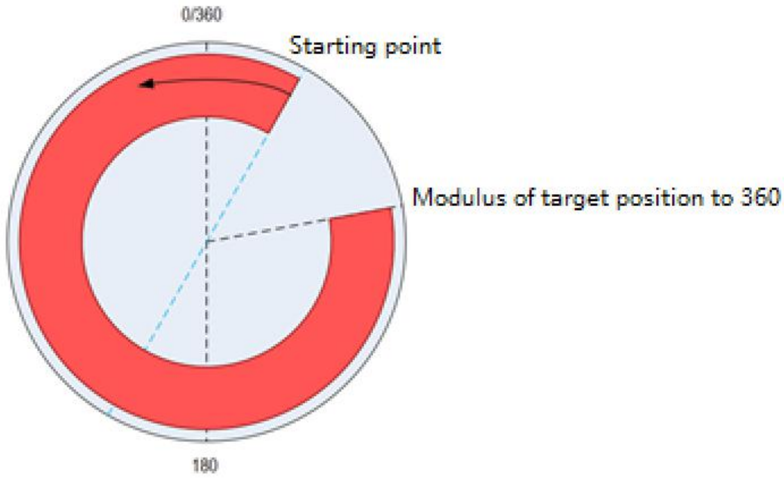


When XPosition< Start Absolute Position, the axis runs in the opposite direction Distance from Start Absolute Position – Xposition.

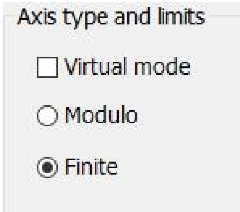


③ The axis rotation period is set to 360 and the Direction is set to shortest or Negative. The modulus of Position to 360 is XPosition

Axis runs in reverse direction Distance from absolute position + 360-XPosition.



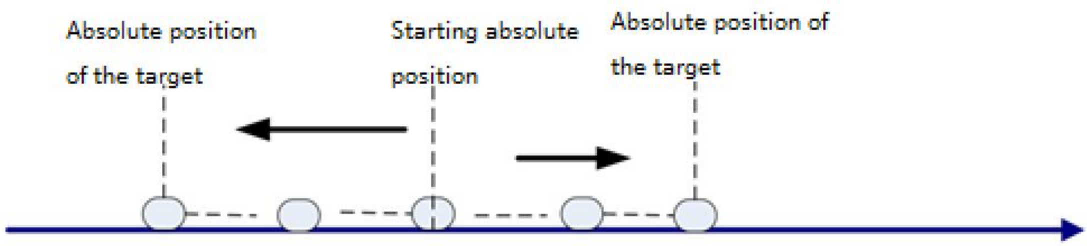
◆ **Absolute positioning of axes in linear mode**



When the absolute position of the target > the starting position, move the target forward (absolute position - distance from the starting position)

When the target position < Start position, move the target in the opposite direction (Start position - Distance from target position)

The running direction set in linear mode does not determine the axis running direction, i.e. Direction is invalid.



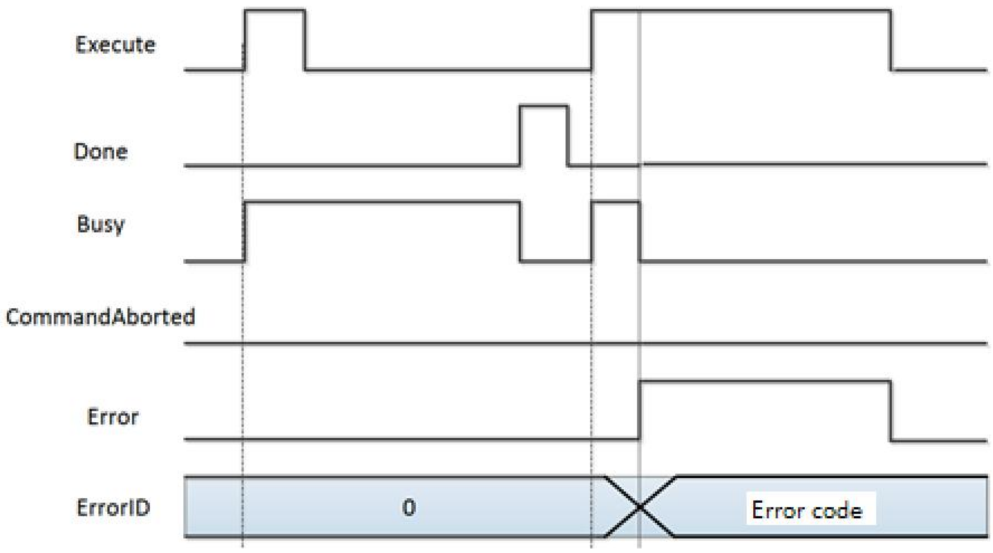
4) Timing diagram

Axis must be in the Standstill state for the instruction to run.

Execute of a function block must have a rising edge condition.

Done of a function block indicates that the instruction has completed normal execution.

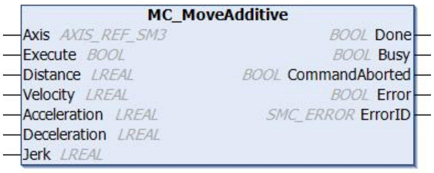
Busy of a function block indicates that the function block is currently being executed;



7.3.7 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.

1) Command Format

Instructions	Name	Graphical representation	ST Performance
MC_Move Additive	Superimposed absolute motion commands		MC_MoveAdditive(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

◆ Input and output variables

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Distance	Axis arrival position	LREAL	Data range	0	This data is the superimposed position data
Velocity	Operating speed	LREAL	Data range	0	Maximum velocity of the axis running to the target position
Acceleration	Acceleration	LREAL	Data range	0	Acceleration value as speed increases
Deceleration	Deceleration	LREAL	Data range	0	Deceleration value as speed becomes smaller
Jerk	Leap	LREAL	Data range	0	Slope change of the acceleration/deceleration curve

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Instruction execution in progress	BOOL	TRUE,FALSE	FALSE	Execution of current instruction is in progress, set to TRUE
Command Abort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs
ErrorID	Error code	SMC_ ERROR	Refer to SMC_ERROR	0	Error code is output when an exception occurs

3) Function description

◆ This function block is a superimposed position command, and the Distance data is the superimposed data of the axis.

◆ If this function block runs in Discrete Motion state, the CommandAbort of other commands will be set in position when it is used.

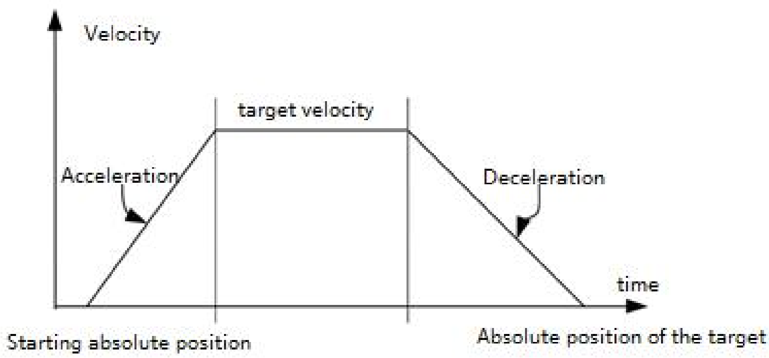
◆ In standstill state, this command can be run independently to achieve relative positioning requirements.

◆ If Acceleration or Deceleration is zero, the instruction runs in an abnormal state, but the state of the axis is Discrete Motion.

◆ The start instruction is the rising edge of Execute.

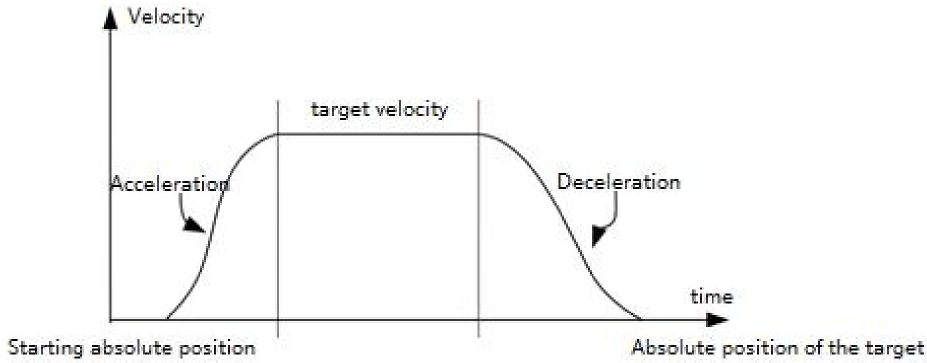
◆ Trapezoidal acceleration and deceleration actions

Velocity, Acceleration and Deceleration have data; and Jerk is 0;



◆S-curve acceleration and deceleration

Velocity, Acceleration, Deceleration and Jerk all have data



4) Timing diagram

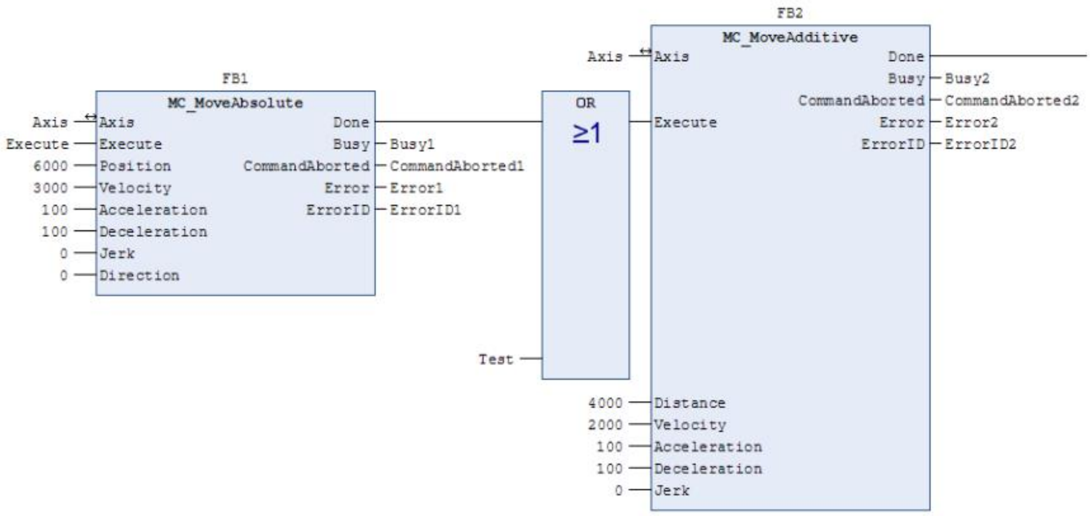
Axis must be in the Standstill state for the instruction to run.

Execute of a function block must have a rising edge condition.

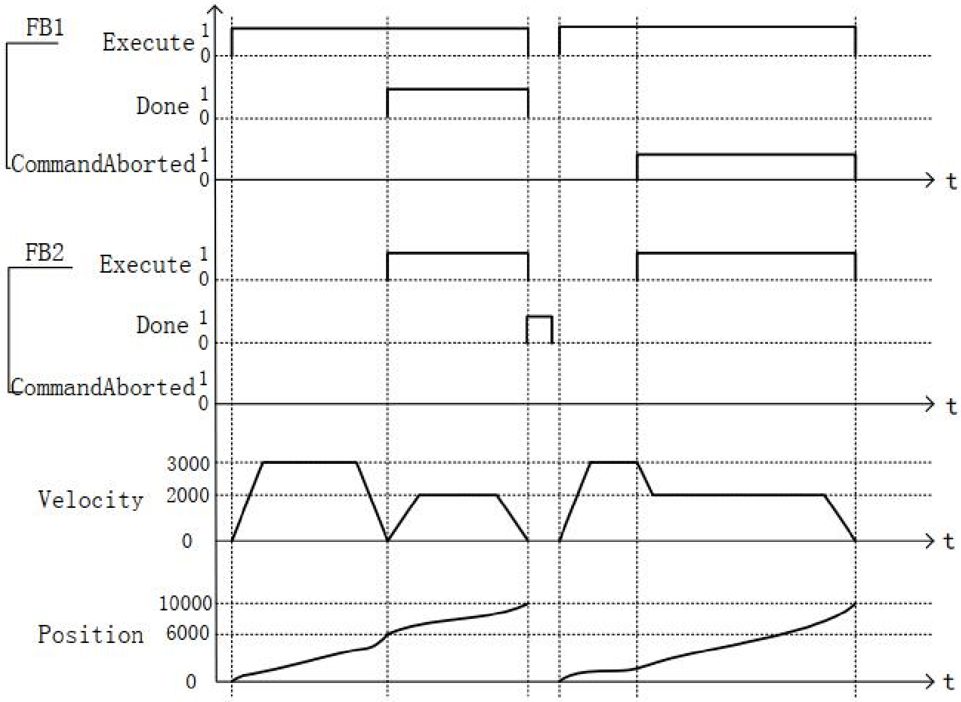
Done of a function block indicates that the instruction has completed normal execution.

Busy of the function block indicates that the function block is currently being executed.

◆ Example



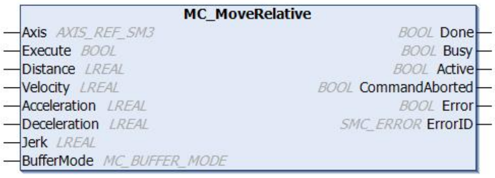
◆ Timing instructions :


FB1
Execute
Done
CommandAborted
FB2
Execute
Done
CommandAborted
Velocity
3000
2000
0
Position
10000
6000
0
t
1
0

7.3.8 MC_MoveRelative

The axes run in relative position. The relative position is specified by Distance (units are set by axis). Set the relevant parameters before running this command, Acceleration、Deceleration、Velocity 、Jerk and BufferMode.

1) Command format

Instructions	Name	Graphical representation	ST Performance
MC_ MoveRelative	Axis relative positioning commands		MC_MoveRelative(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Distance	Relative position of movement	LREAL	Data range	0	This data is the relative position of the movement
Velocity	Running speed	LREAL	Data range	0	Maximum velocity of the axis running to the target position
Acceleration	Acceleratio n	LREAL	Data range	0	Acceleration value as the velocity becomes greater
Deceleration	Deceleratio n	LREAL	Data range	0	Value of deceleration as speed becomes smaller
Jerk	Leap	LREAL	Data range	0	The value of the change

	degree				in slope of the acceleration and deceleration curve
BufferMode	Buffer mode	MC_BUFFER_MODE	Aborting; Buffered; BlendingLow; BlendingPrevious; BlendingNext; BlendingHigh;	Aborting	Defines the time sequence of this FB relative to the previous function block. If the function block is Busy, then only BufferMode=Aborting is allowed.

BufferMode (available only with CODESYS SoftMotion version 4.8.0.0)	Introduction
Aborting	Without buffering, the previous motion function block is immediately aborted and this function block is started immediately (default mode)
Buffered	This function block is started again after the previous motion command has completed its movement
BlendingLow	When switching, after the previous motion command has completed its movement, pass the end position of the first motion command at the lower speed of the two preceding and following motion commands
BlendingPrevious	When switching, after the previous motion command has completed its movement, the end position of the previous motion command is passed at the speed of the previous motion command
BlendingNext	When switching, after the previous motion command has completed its movement, pass the end position of the previous motion command at the speed of the latter motion command
BlendingHigh	When switching, after the previous motion command has completed its movement, pass the end of the first motion command at the higher speed of the two preceding and following motion commands

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Instruction execution in	BOOL	TRUE,FALSE	FALSE	Execution of current instruction is in progress,

	progress				set to TRUE
CommandAbort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs
ErrorID	Error code	SMC_ ERROR	Refer to SMC_ ERROR	0	Error code is output when an exception occurs

3) Function description

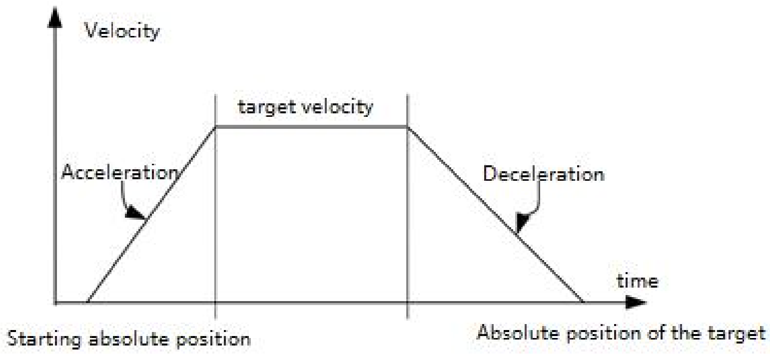
This function block runs in Standstill and the state of the instruction is Discrete Motion, so that the execution of the instruction can be focused on the running state of the axis to avoid interrupting other instructions of the axis or being interrupted by other instructions.

The start instruction is the rising edge of Execute, this instruction can be repeated on the rising edge of Discrete Motion to refresh the latest Position position each time.

Acceleration or Deceleration is zero, the instruction runs in an abnormal state, but the state of the axis is Discrete Motion.

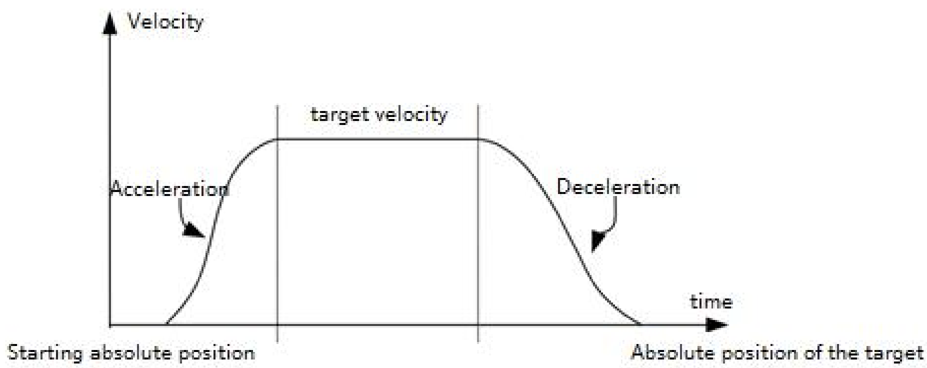
◆ Trapezoidal acceleration and deceleration movements

Velocity, Acceleration and Deceleration have data; and Jerk is 0;



◆S-shaped acceleration and deceleration movements

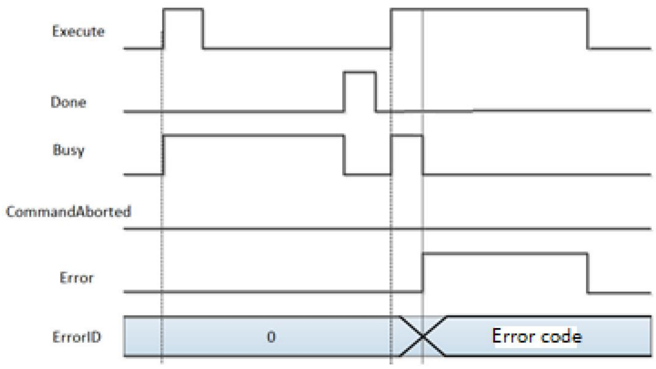
Velocity, Acceleration and Deceleration and Jerk all have data;



4) Timing diagram

Execute of a function block must have a rising edge condition.

Done for a function block indicates that the instruction has been executed normally.
Busy of a function block indicates that the function block is currently being executed;

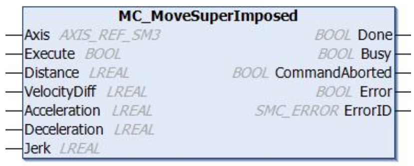


7.3.9 MC_MoveSuperImposed

Axis in the original instruction speed and position on the basis of the superimposed acceleration and position data in the running instruction, the entire original instruction execution time model no change; through this instruction can solve our actual operation of some similar by the belt and gear clearance error compensation, can ensure the consistency of the movement;

The command runs with the parameters Distance, VelocityDiff, Acceleration, Deceleration and Velocity; a value of 0 for Acceleration or Deceleration is an error. MC_ MoveSuperImposed is equivalent to the MC_MoveRelative instruction in the standstill state.

1) Command format

Instructions	Name	Graphical representation	ST Performance
MC_ MoveSuperImposed	叠加相对 运动指令		MC_MoveSuperImposed(Axis:= , Execute:= , Distance:= , VelocityDiff:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description

Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
Distance	Axis arrival position	LREAL	Data range	0	This data is the superimposed position data
VelocityDiff	Stack acceleration	LREAL	Data range	0	Axis running superimposed acceleration
Acceleration	Acceleration	LREAL	Data range	0	Acceleration value as speed increases
Deceleration	Deceleration	LREAL	Data range	0	Deceleration value as speed increases
Jerk	Leap	LREAL	Data range	0	Slope change of curve acceleration/deceleration

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction execution completed	BOOL	TRUE,FALSE	FALSE	Axis instruction execution complete, set to TRUE
Busy	Instruction execution in progress	BOOL	TRUE,FALSE	FALSE	Execution of current instruction is in progress, set to TRUE
CommandAbort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs
ErrorID	Error code	SMC_ ERROR	Refer to SMC_ ERROR	0	Error code is output when an exception occurs

3) Function Description

This function block is for superimposing position and velocity commands, VelocityDiff and Distance for superimposing velocity and position on other commands, respectively.

MC_MoveSuperImposed can be superimposed on any other command in motion mode.

MC_MoveSuperImposed can also be interrupted by MC_MoveSuperImposed.

in the state StandStill, the function block MC_MoveSuperimposed acts similarly to MC_MoveRelative.

The start instruction is the rising edge of Execute.

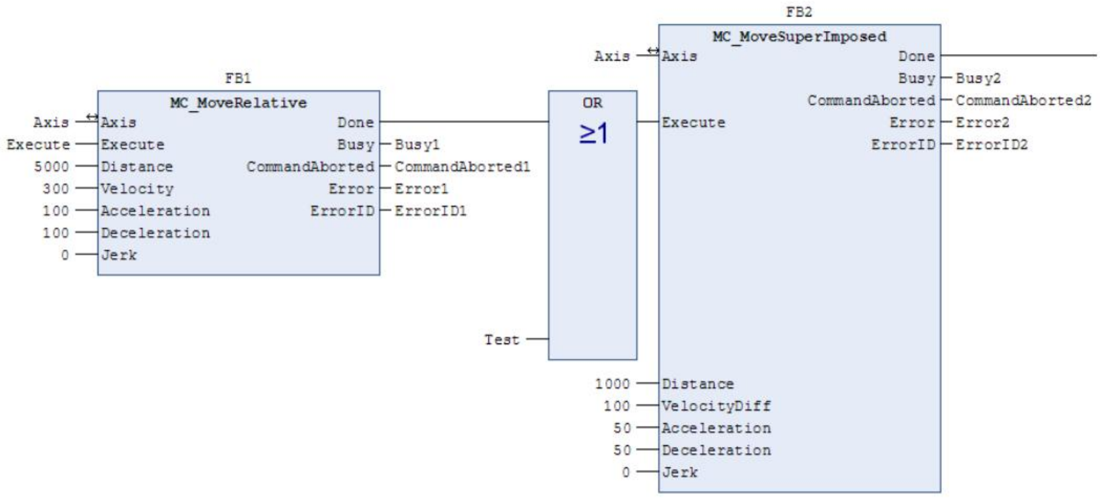
4) Timing diagram

the function block Execute must have a rising edge condition.

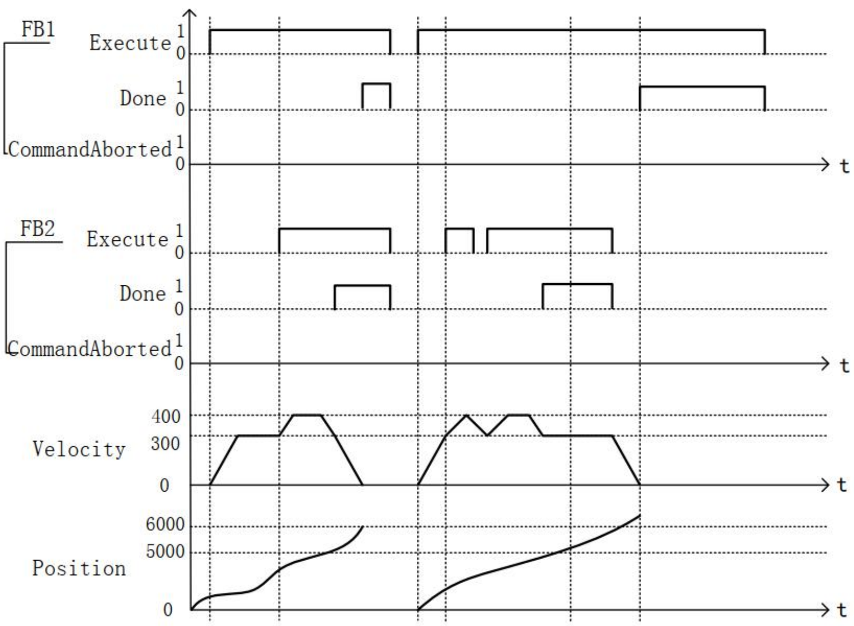
Done of the function block indicates that the instruction has been executed normally.

Busy of the function block indicates that the function block is currently being executed.

◆ Example



◆Timing operating instructions :



7.3.10 MC_PositionProfile

This function block is designed to command time-position locked motion profiles.

1) Command format

Instructions	Name	Graphical representation	ST Performance
MC_ PositionProfile	Position profile command		

2) Related variables

◆ Input and output variables

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	Axes	AXIS_REF	—	—	Mapping to an axis, i.e. an instance of AXIS_REF_SM3
TimePosition	Axis position running time and position description	MC_TP_REF			Description of the axis position runtime and position data , data consisting of multiple data sets

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Execution conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block
ArraySize	Dynamic arrays	INT	Data range	0	Number of arrays used in the run profile
PositionScale	Synthesis factors	LREAL	" Positive numbers " + "0"	1	Scale factor of the position in MC_TP_REF
Offset	Offset	LREAL		0	Overall offset value of the position

◆ Output Variables

Output Variables	Name	Data Type	Effective range	Initial value	Description
Done	Instruction	BOOL	TRUE,FALSE	FALSE	Axis instruction

	execution completed				execution complete, set to TRUE
Busy	Instruction execution in progress	BOOL	TRUE,FALSE	FALSE	Execution of current instruction is in progress, set to TRUE
CommandAbort	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	If the current instruction is interrupted, set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	Set to TRUE if an exception occurs
ErrorID	Error code	SMC_ERROR	Rerfer to SMC_ERROR	0	Error code is output when an exception occurs

3) Function description

This function block is a contouring motion model for time periods and positions, with a running mode of Discrete Motion, based on the data set by the user in the TimePosition variable. The running state of this function block is in Standstill, the state of the instruction is Discrete Motion, and no other state can be run. The start instruction is the rising edge of Execute, and the instruction is repeated in Discrete Motion.

TimePosition is the MC_TP_REF data type;

MC_TP_REF is described as follows :

Members	Type	Initial value	Description
Number_of_pairs	INT	0	Number of segments of the profile path
IsAbsolute	BOOL	TRUE	Absolute motion (TRUE) and relative motion selection
MC_TP_Array	ARRAY[1..N] OF SMC_TP		Arrays of times and positions

SMC_TP The specific description is as follows:

Members	Type	Initial value	Description
delta_time	TIME	TIME#0ms	Time of position segment
position	LREAL	0	Current position value

Note: Any change in speed according to the set position data will be adjusted according to the S curve.

◆ Timing diagram

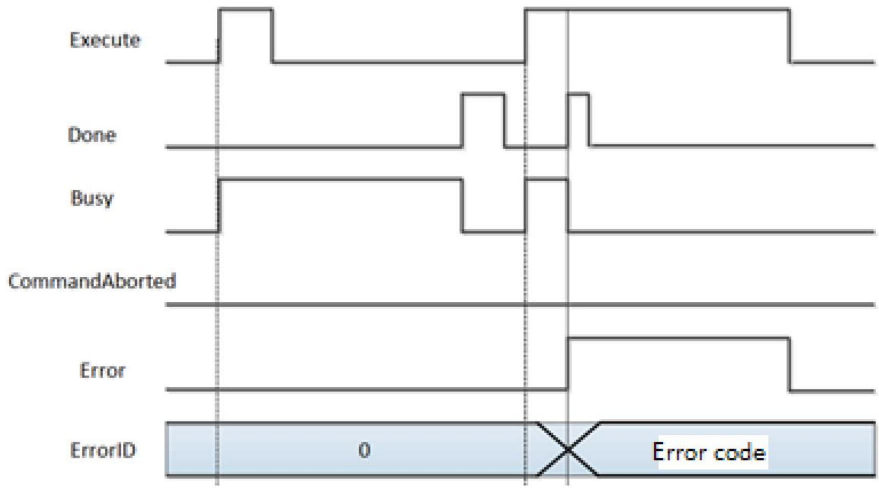
Conditions MC_TP_Array can only be run if the position profile command has been set by other means.

The axis must be in the Standstill state for the instruction to run.

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the instruction has been executed normally.

Busy of a function block indicates that the function block is currently being executed;



4) Error description

An error occurs when the axis state is not a parameter error in the start-up instruction or instruction system in Standstill, and the axis error can only be cleared before operation starts.

Note]: Please read "Appendix C Error Code Descriptions" for a description of the relevant error codes.

7.3.11 MC_Reset

By resetting all errors associated with the internal axes, the function block is designed to stop from status error to stop. This does not affect the output of the function block instance.

1) Command format

Instructions	Name	Graphical representation	ST Performance
MC_Reset	Axis error status reset command		MC_Reset(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

◆ **Input and output variables**

Input and output variables	Name	Data Type	Effective range	Initial value	Description
Axis	AXIS	AXIS_REF	—	—	Mapped to an axis, i.e. an instance of AXIS_REF_SM3

◆ Input variables

Input variables	Name	Data Type	Effective range	Initial value	Description
Execute	Implementation conditions	BOOL	TRUE,FALSE	FALSE	A rising edge of the input will initiate the processing of the function block

◆The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The execution of the instruction is complete	BOOL	TRUE,FALSE	FALSE	The axis instruction execution is complete and is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

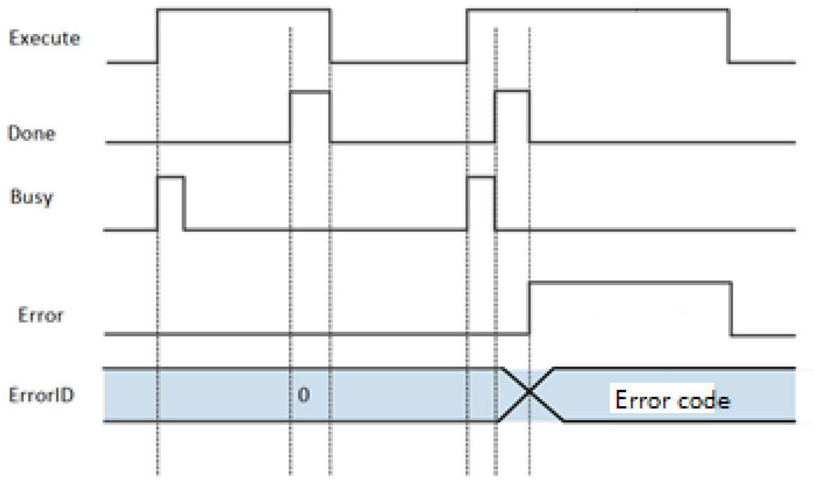
3) Description of the function

This function block changes the axis state to Standstill in the case of normal axis communication, and the abnormal state of the axis to a normal and operational state;

Axis.bCommunication is FLASE state when the axis errorstop cannot be reset, and the communication between the main station and the from the station axis must be re-established;

The Busy flag bit in the instruction has a very short time to connect, please note when using;

Time series chart



7.3.12 MC_ReadActualPosition

The instruction reads the actual location where the drive is running and is saved in a variable cell that it defines.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ ReadActualPosition	The actual location reads the instruction		MC_ReadActualPosition(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current position of the servo for the TRUE state

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Valid	Location data is available for flags	BOOL	TRUE,FALSE	FALSE	The correct location of the drive is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
Position	Gets to the axis position	LREAL	Axis position	0	The axis position data read out of the instruction

3) Function description

The actual position command in the drive is read by means of this instruction, which is an Enable level enable effect. The instruction can be used repeatedly without affecting each other.

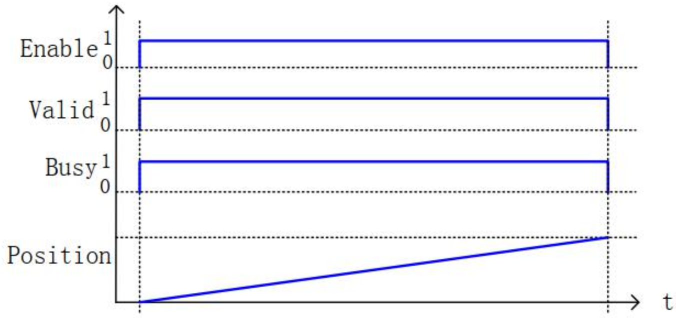
◆ Timing diagram

The condition that Enable of the function block must be TRUE.

Valid of the function block indicates that the Position read is a valid data value.

Busy of the function block indicates that the function block is currently being executed.

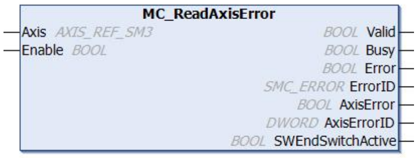
Timing operation description :



7.3.13 MC_ReadAxisError

The error case in which the instruction reads the axis and is saved in a variable cell that it defines.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ReadAxisError	The wrong state of the reading axis		MC_ReadAxisError(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , AxisError=> , AxisErrorID=> , SWEndSwitchActive=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current position of the servo for the TRUE state

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Valid	The error data gets the flag	BOOL	TRUE,FALSE	FALSE	The error data of the axis can be obtained and placed as TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is

					set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
AxisError	The axis is incorrectly marked	BOOL	TRUE,FALSE	FALSE	The read-out axis is an error, corresponding to the indicated position
AxisErrorID	Axis error code	DWORD	0	The read-out axis is an error code	
SWEnd SwitchActive	The soft limit switch is valid	BOOL	TRUE,FALSE	FALSE	In instruction read, check the status of the soft limit switch

3) Function description

Reads the error code in the drive via MC_ReadAxisError, the instruction is the Enable level enable effect. The instruction can be used repeatedly without affecting each other.

◆ Timing diagram

The condition that Enable of the function block must be TRUE.

Valid of the function block indicates that the AxisError and AxisErrorID read is a valid data value.

Busy of the function block indicates that the function block is currently being executed;

7.3.14 MC_ReadBoolParameter

The instruction reads the bit parameters of the drive shaft and saves them in a variable unit that it defines.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ ReadBoolParameter	Read the bit parameters of the axis		MC_ReadBoolParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current position of the servo for the TRUE state
ParameterNumber	The serial number of the axis argument	DINT		0	Access the indexes and sub-indexes and serial numbers of axis parameters

Note:P arameterNumber (DINT)=

-DWORD_TO_DINT (SHL (USINT_TO_DWORD (usiDataLength), 24) (data length in object dictionary)

+SHL (UINT_TO_DWORD (uiIndex), 8) (index -16BIT inobject dictionary)

+usisubIndex(sub-index -8BIT in object dictionary).

usiDataLength: Filled in by bytes; 1 byte is 16'01; 2 bytes is 16'02; 4 bytes is 16'04, etc.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe

Valid	Location data is available for flags	BOOL	TRUE,FALSE	FALSE	The correct location of the drive is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
Value	Gets to the axis status	BOOL	TRUE,FALSE	FALSE	The axis status of the instruction read out

3) Function description

The bit data status in the drive is read via MC_ReadBoolParam, the instruction is an Enable level enable effect. The instruction can be used repeatedly without affecting each other.

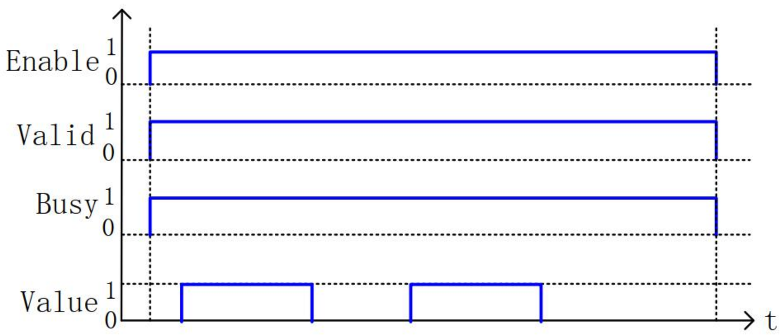
◆ Timing diagram

The condition that Enable of the function block must be TRUE.

Valid of the function block indicates that the read Valid is a valid bit status data.

Busy of the function block indicates that the current function block is being executed.

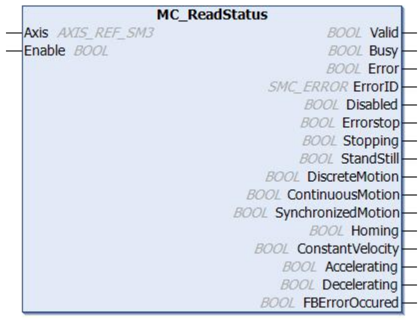
◆ Timing operation description :



7.3.15 MC_ReadStatus

The instruction reads the state data of the axis and saves it in its own defined variable cell.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ReadStatus	The status of the reading axis		MC_ReadStatus(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Disabled=> , Errorstop=> , Stopping=> , StandStill=> , DiscreteMotion=> , ContinuousMotion=> , SynchronizedMotion=> , Homing=> , ConstantVelocity=> , Accelerating=> , Decelerating=> , FBErrorOccured=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current position of the servo for the TRUE state

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Valid	Axis status Gets the flag	BOOL	TRUE,FALSE	FALSE	When true, the representative axis state is available
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_	See SMC_	0	When an exception occurs, the

		ERROR	ERROR		error code is output
Disabled	The axis is not in an enabled state	BOOL	TRUE,FALSE	FALSE	The axis is true in the no-enabled state;
Errorstop	Axis error status	BOOL	TRUE,FALSE	FALSE	The axis is true in the error operating state;
Stoping	The axis stops the process state	BOOL	TRUE,FALSE	FALSE	The axis is TRUE during the stop process
StandStill	Axis standard status	BOOL	TRUE,FALSE	FALSE	The axis is TRUE in the standard (operational) state
Discrete Motion	The discrete motion state of the axis	BOOL	TRUE,FALSE	FALSE	The axis is TRUE in a discrete motion state
Continuous Motion	The continuous motion of the axis	BOOL	TRUE,FALSE	FALSE	The axis is TRUE in a continuous motion state
Synchronized Motion	The axis runs in sync	BOOL	TRUE,FALSE	FALSE	The axis is TRUE in the synchronized motion state
Homing	The axis returns to the origin state	BOOL	TRUE,FALSE	FALSE	The axis is TRUE in the back-to-origin state
Constant Velocity	The shaft runs at a speed of arrival	BOOL	TRUE,FALSE	FALSE	True when the shaft reaches run speed
Accelerating	The axis accelerates the process state	BOOL	TRUE,FALSE	FALSE	The axis acceleration process status is TRUE
Dccelerating	Axis deceleration process status	BOOL	TRUE,FALSE	FALSE	The axis deceleration process status is TRUE
FBError Occured	A flag appears for an error in the axis function block	BOOL	TRUE,FALSE	FALSE	The axis function block error flag is TRUE

3) Function description

The various states of the corresponding axes are indicated by MC_ReadStatus, the command is the Enable level enable effect. The command can be used several times without affecting each other.

The Enable condition of the function block must be TRUE.

Valid of the function block indicates that the various data of the next status flag are read out.

The Busy of a function block indicates that the function block is currently being executed.

7.3.16 MC_ReadParameter

The instruction reads the parameters of the drive shaft and saves them in a variable unit that you define yourself.

1) Instruction format

Instructions	Name	Graphical performance	ST performance

MC_
ReadParameter
	Read the parameters of the axis		`MC_ReadParameter(`
			`Axis:= ,`
			`Enable:= ,`
			`ParameterNumber:= ,`
			`Valid=> ,`
			`Busy=> ,`
			`Error=> ,`
			`ErrorID=> ,`
			`Value=> );`

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current position of the servo for the TRUE state
Parameter Number	The serial number of the axis argument	DINT		0	Access the indexes and sub-indexes and serial numbers of axis parameters

Note:P arameterNumber (DINT)=

-DWORD_TO_DINT (SHL (USINT_TO_DWORD (usiDataLength), 24) (data length in object dictionary)

+SHL (UINT_TO_DWORD (uiIndex), 8) (index -16BIT inobject dictionary)

+usisubIndex(sub-index -8BIT in object dictionary).

usiDataLength: Filled in by bytes; 1 byte is 16'01; 2 bytes is 16'02; 4 bytes is 16'04, etc.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe

Valid	Location data is available for flags	BOOL	TRUE,FALSE	FALSE	The correct location of the drive is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
Value	Gets the axis parameters	LREAL		0	The axis parameters read out of the instruction

3) Description of the function

The MC_ReadParam the bit data state in the drive by using the computer, instructing the Enable level enable effect. Instructions can be reused multiple times without affecting each other.

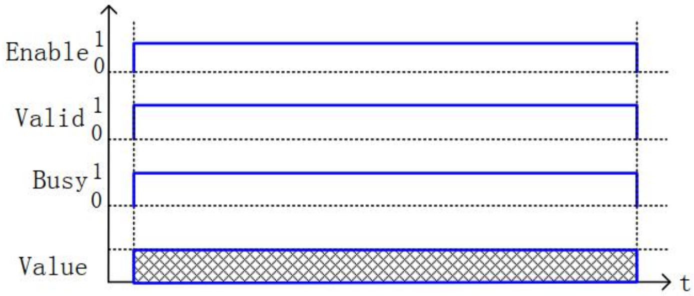
◆Timing diagram

the condition that Enable of the function block must be TRUE.

Valid of a function block indicates that the read Valid is a valid bit status data.

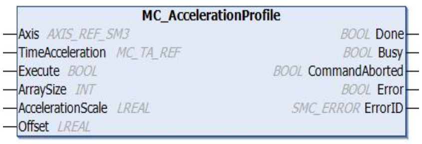
Busy of a function block indicates that the current function block is being executed.

Timing operation description :



7.3.17 MC_AccelerationProfile

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ AccelerationProfile	Acceleration profile instruction		MC_AccelerationProfile(Axis:= , TimeAcceleration:= , Execute:= , ArraySize:= , AccelerationScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

Input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property
TimeAcceleration	Axis acceleration time and acceleration description	MC_TA_ REF			Axis acceleration time and acceleration data description, acceleration data consists of multiple sets of data

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	An up-edge of the input will initiate the processing of the function block
ArraySize	Dynamic array	INT	The range of data	0	The number of arrays used in the run profile
AccelerationScale	Synthesis factor	LREAL	"Positive" and "0"	1	MC_TA_REF factor of acceleration or destoation in the system
Offset	Offset	LREAL		0	The overall offset value of the acceleration and decrease speed

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The execution of	BOOL	TRUE,FALSE	FALSE	The axis instruction execution is

	the instruction is complete				complete and is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Command Abort	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	The current instruction is interrupted and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Description of the function

This function block is a profile motion model for time periods and deceleration, running in Discrete Motion, based on the data set by the user in the TimeAcceleration variable. This function block runs in Standstill, the instruction runs in Discrete Motion, and other states cannot run. The startup instruction is the up-edge start of Execute, and this instruction repeats the speed at Discrete Motion on the last overlay, which is prone to system failure.

TimeAcceleration is MC_TA_REF data type;

MC_TA_REF description is as follows:

Members	Type	The initial value	Describe
Number_of_pairs	INT	0	The number of segments of the profile path
IsAbsolute	BOOL	TRUE	Absolute motion (TRUE) and relative motion selection
MC_TA_Array	ARRAY[1..N] OF SMC_TA		An array of time and acceleration values

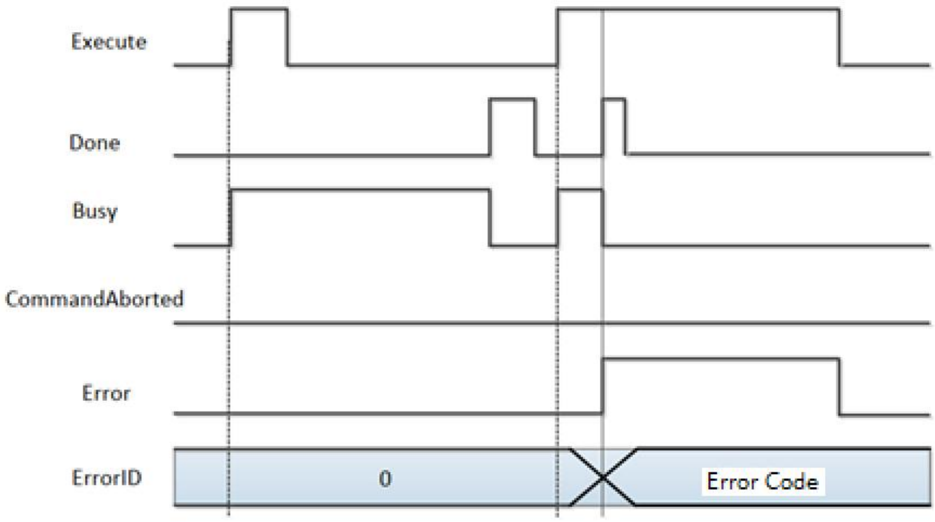
SMC_TA description is as follows:

Members	Type	The initial value	Describe
delta_time	TIME	TIME#0ms	The time of the acceleration period
acceleration	LREAL	0	The current acceleration value

Note : The set acceleration is reflected in the change in velocity, all acceleration changes in the way the S curve changes, from the final result to the acceleration data of the starting acceleration isA, the termination acceleration is B(A-B)/2 is reflected in the final velocity;

4) Time series chart

Condition MC_TA_Array has been set by other means; the axis must
be in the Standstill state instruction to run; the Execute of the function block must have conditions on the rising
edge; the Done of the function block indicates that the instruction is executed
normally; and the Busy of the function block indicates that the current function block is in the process of
executing;



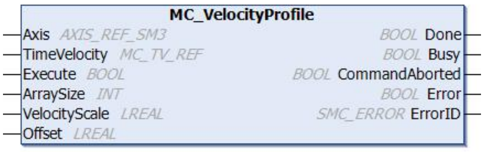
5) Error description

An error occurs when the axis state is not a parameter error in the start-up instruction or instruction system in Standstill, and the axis error can only be cleared before operation starts.

[Note]: Please read "Appendix C Error Code Descriptions" for a description of the relevant error codes.

7.3.18 MC_VelocityProfile

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ VelocityProfile	Speed profile instructions		MC_VelocityProfile(Axis:= , TimeVelocity:= , Execute:= , ArraySize:= , VelocityScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property
TimeVelocity	Axis speed run time and speed description	MC_TV_ REF			Axis speed run time and speed data description, consisting of multiple sets of data

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	An up-edge of the input will initiate the processing of the function block
ArraySize	Dynamic array	INT	The range of data	0	The number of arrays used in the run profile
VelocityScale	The speed factor	LREAL	"Positive", "0"	1	The scale factor of the speed
Offset	Offset	LREAL		0	The overall offset value of the velocity value

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The execution of the instruction is complete	BOOL	TRUE,FALSE	FALSE	The execution of the instruction is complete and is set to TRUE

Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Command Abort	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	The current instruction is interrupted and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output

3) Description of the function

This function block is an outline motion model for time periods and speeds, running in Continuous Motion, based on data set by the user in the TimeVelocity variable.

This function block runs in Standstill, the instruction runs in Discrete Motion, and other states cannot run.

The startup instruction starts on the rising edge of Execute, and this instruction runs repeatedly in Discrete Motion.

TimeVelocity is MC_TV_REF data type;

MC_TV_REF description is as follows:

Members	Type	The initial value	Describe
Number_of_pairs	INT	0	The number of segments of the profile path
IsAbsolute	BOOL	TRUE	Absolute motion (TRUE) and relative motion selection
MC_TV_Array	ARRAY[1..N] OF SMC_TV		An array of time and speed

SMC_TV description is as follows:

Members	Type	The initial value	Describe
delta_time	TIME	TIME#0ms	The time of the speed value segment
Velocity	LREAL	0	The speed value of the current record

Note: The entire velocity process is the way the S curve is deceleration, and each profile is calculated as an overlay; Speed is also superimposed when instructions are repeated, avoiding speed oversleed when instructions are used, and repeated runs must return the state of this

axis to the Standstill state.

◆Timing diagram

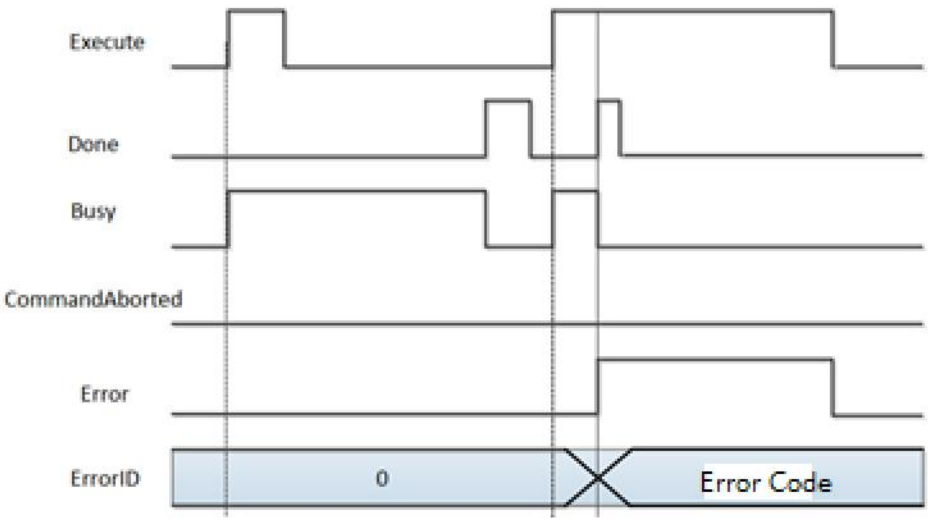
condition MC_TV_Array has been set by other means in order to run the position profile instruction.

the axis must be in the Standstill state for the instruction to run.

Execute of the function block must have a rising edge condition.

Done of the function block indicates that the instruction has been executed normally.

Busy of a function block indicates that the function block is currently being executed.;



4) Error description

An error occurs when the axis state is not a parameter error in the start-up instruction or instruction system in Standstill, and the axis error can only be cleared before operation starts.

[Note]: Please read "Appendix C Error Code Descriptions" for a description of the relevant error codes.

7.3.19 MC_WriteBoolParameter

The instruction sets the bit parameters of the drive shaft.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ WriteBoolParameter	Set the bit parameters for the axis		MC_WriteBoolParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	Set the operation once for the rising edge operation
Parameter Number	The serial number of the axis argument	DINT		0	Access the indexes and sub-indexes and serial numbers of axis parameters
Value	Set the value	BOOL	TRUE,FALSE	FALSE	Set the bit parameter value

Note:P arameterNumber (DINT)=

-DWORD_TO_DINT (SHL (USINT_TO_DWORD (usiDataLength), 24) (data length in object dictionary)

+SHL (UINT_TO_DWORD (uiIndex), 8) (index -16BIT inobject dictionary)

+usisubIndex(sub-index -8BIT in object dictionary).

usiDataLength: Filled in by bytes; 1 byte is 16'01; 2 bytes is 16'02; 4 bytes is 16'04, etc.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The setup	BOOL	TRUE,FALSE	FALSE	The setup operation was

	operation was successful				successfully set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Function description

The bit parameter of the axis is set via MC_ WriteBoolParameter and the instruction is Execute rising edge triggered. The instruction can be used several times without affecting each other.

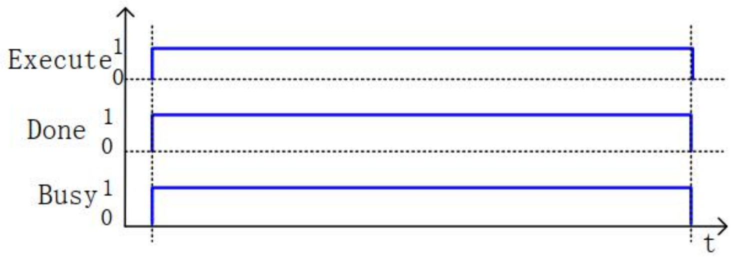
◆ Timing diagram

Execute of the function block must be a rising edge triggering condition.

Done of the function block means that the setting operation is successful.

Busy of the function block indicates that the current function block is being executed.

◆ Description of the timing operation :



7.3.20 MC_WriteParameter

Instructions write parameters to the drive axis and are stored in their own defined variable units.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ WriteParameter	Set the axis parameters		MC_WriteParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	Set the operation once for the rising edge operation
Parameter Number	The serial number of the axis argument	DINT		0	Access the indexes and sub-indexes and serial numbers of axis parameters
Value	Set the value	LREAL			Set the bit parameter value

Note:P arameterNumber (DINT)=

-DWORD_TO_DINT (SHL (USINT_TO_DWORD (usiDataLength), 24) (data length in object dictionary)

+SHL (UINT_TO_DWORD (uiIndex), 8) (index -16BIT inobject dictionary)

+usisubIndex(sub-index -8BIT in object dictionary).

usiDataLength: Filled in by bytes; 1 byte is 16'01; 2 bytes is 16'02; 4 bytes is 16'04, etc.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe

Done	The setup operation was successful	BOOL	TRUE,FALSE	FALSE	The setup operation was successfully set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Function description

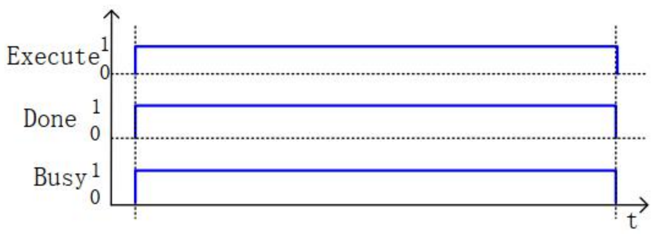
The bit parameter for the axis is set via MC_ WriteParameter and the instruction is Execute rising edge triggered. The instruction can be used several times without affecting each other.

- Timing diagram

Execute of the function block must be a rising edge trigger condition.

Done of the function block means that the setting operation is successful.

Busy of the function block means that the function block is currently being executed;



7.3.21 MC_AbortTrigger

The function block terminates the associated characteristics of the input latch-related events and is used MC_Touchprobe with the user.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_AbortTrigger	The function block terminates the event association		MC_AbortTrigger(Axis:= , TriggerInput:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property
TruggerInput	Trigger signal	TRIIGGER_REF	—	—	Description of trigger signals, trigger properties, etc

The TRIIGGER_REF description:

Structure	Elements	The data type	The initial value	Describe
TRIIGGER_REF	iTrigger Number	INT	-1	Which one of the lock functions is locked in drive mode. 0: Probe 1 Rising Edge Latch 1: Probe 1 Falling Edge Latching 2: Probe 2 Rising Edge Latching 3: Probe 2 Falling Edge Latching
	bFastLatching	BOOL	TRUE	Specifies the type of latch trigger: TRUE: Drive mode FALSE: Controller mode
	bInput	BOOL		bFastLatching is triggered by the controller Input signal when flasE
	bActive	BOOL		A valid signal that is triggered

Enter variables

Enter the	Name	The data type	Effective range	The initial	Describe

variable				value	
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	Set the operation once for the rising edge operation

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The setup operation was successful	BOOL	TRUE,FALSE	FALSE	The setup operation was successfully set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output

3) Function description

The MC_AbortTrigger function block terminates a trigger signal or property and its associated trigger instruction. Execute of the function block must be a rising edge trigger condition; Done of the function block indicates a successful setup operation; Busy of the function block indicates that the function block is currently being executed;

7.3.22 MC_ReadActualTorque

The instruction reads the current torque value that the drive runs, and the current torque value that is read is saved in a variable unit that you define yourself.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ReadActualTorque	The current torque value reads the instruction		MC_ReadActualTorque0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Torque=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current position of the servo for the TRUE state

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Valid	The current torque value gets the flag	BOOL	TRUE,FALSE	FALSE	The torque value of the drive is correctly obtained and placed as TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
Torque	The current torque	LREAL	The torque	0	The current torque data read out

	value obtained		value (s)		by the instruction

3) Function description

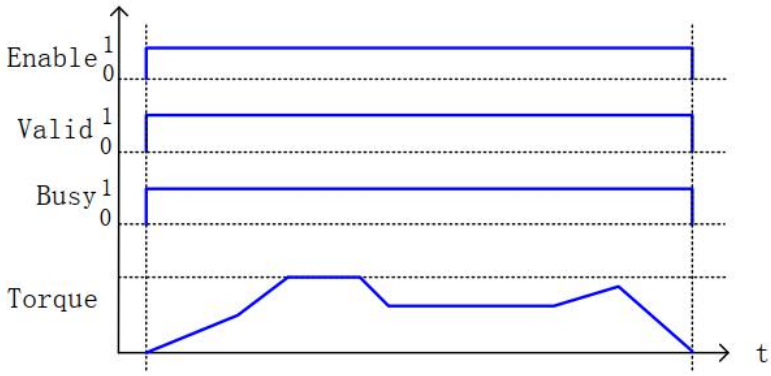
The command to read the current torque value in the drive via MC_ReadActualTorque is the Enable level enable effect. The instruction can be used several times without affecting each other.

◆Timing diagram

The condition that Enable of the function block must be TRUE.

Valid of the function block indicates that the Torque read out is a valid data value.

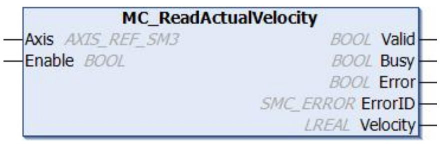
Busy of the function block indicates that the current function block is being executed.;



7.3.23 MC_ReadActualVelocity

The instruction reads the current speed value at which the drive runs, and the current speed value of the read is saved in a variable cell that it defines.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ ReadActualVelocity	Current speed Read instructions		MC_ReadActualVelocity0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	The execution condition	BOOL	TRUE,FALSE	FALSE	Read the current axis speed for the TRUE state

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Valid	The current speed value gets the flag	BOOL	TRUE,FALSE	FALSE	The speed value of the drive is correctly obtained and set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
Velocity	The current speed value obtained	LREAL	The speed value	0	The current speed data read out by the instruction

3) Function description

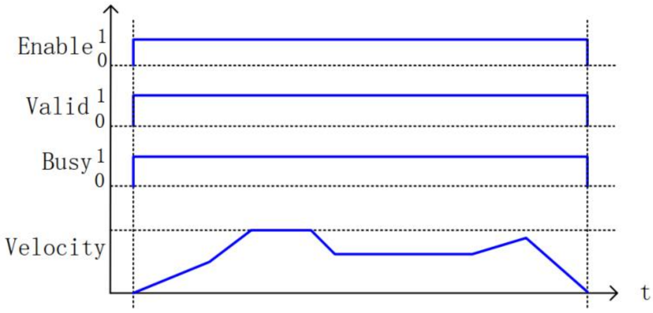
The command to read the current velocity value in the drive via MC_ReadActualVelocity is an Enable level enable effect. The instruction can be used several times without affecting each other.

◆Timing diagram

The condition that Enable of the function block must be TRUE.

Valid of the function block indicates that the Velocity read out is a valid data value.

Busy of the function block indicates that the current function block is being executed;



7.3.24 MC_SetPosition

Setting the position data in the instruction to the position data of the current axis does not cause any displacement movement to the set position data operation, which is used to produce displacement of the coordinate system.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ SetPosition	Read the parameters of the axis		MC_SetPosition0(Axis:= , Execute:= , Position:= , Mode:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	Set the operation once for the rising edge operation
Position	Axis position data	LREAL		0	Location data
Mode	Set the value	BOOL	TRUE,FALSE	FALSE	Position mode; TRUE: Relative Position (RELATIVE); FALSE: Absolute Position (ABSOLUTE);

The output variable

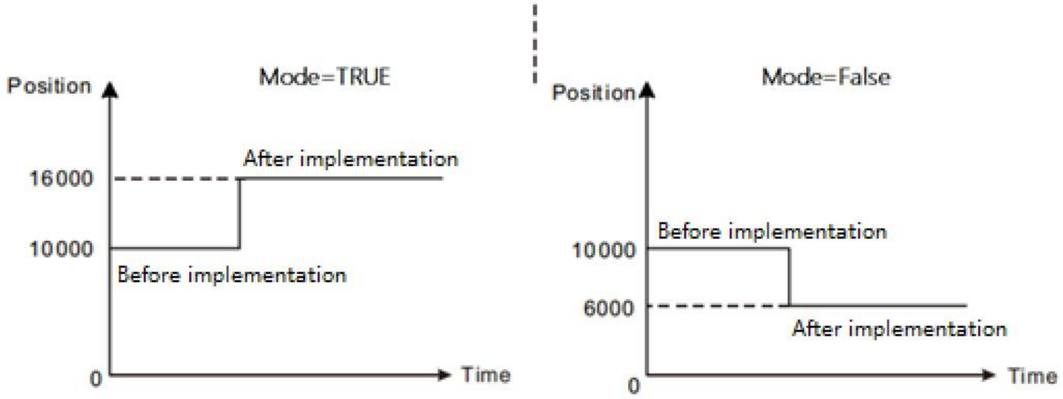
The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The setup	BOOL	TRUE,FALSE	FALSE	The setup operation was

	operation was successful				successfully set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is executing in and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output

3) Function description

a. The axis position parameter is set by MC_ SetPosition, which does not produce any displacement but creates a coordinate offset; the command is triggered by the rising edge of Execute; the command can be used repeatedly without affecting each other.

b. Relationship with the reference position. When Mode=TRUE, Position is relative to the reference position, and the value of Position=Reference Position+Position; when Mode=FALSE, Position is absolute to the reference position, and the value of Position=Position. When the input parameter Relative is a different value, the corresponding execution effects are shown in the lower left and lower right figures respectively.

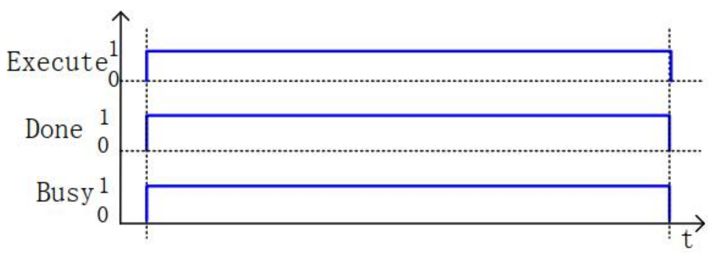


◆ Timing diagram

Execute of the function block must be a rising edge trigger condition.

Done of the function block indicates that the setting operation was successful.

Busy of a function block indicates that the current function block is being executed;



7.3.25 MC_TouchProbe

The instruction is triggered by an external signal to save the position data of the current axis.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_TouchProbe	Enable external locking		MC_TouchProbe(Axis:= , TriggerInput:= , Execute:= , WindowOnly:= , FirstPosition:= , LastPosition:= , Done=> , Busy=> , Error=> , ErrorID=> , RecordedPosition=> , CommandAborted=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property
TruggerInput	Trigger signal	TRIIGGER_ REF	—	—	Associated properties such as trigger signals or trigger properties

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	Set the operation once for the rising edge operation
WindowOnly	Trigger the window	BOOL	TRUE,FALSE	FALSE	
FirstPosition	The start position of the trigger	LREAL	—	0	Specify where to start the receive trigger
LastPosition	The end position of the trigger	LREAL	—	0	Specify the end position where the receive trigger is received

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	The setup operation was	BOOL	TRUE,FALSE	FALSE	The setup operation was successfully set to TRUE

	successful				
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
RecordedPosition	The location where the record was triggered	LREAL	—	0	The current position at which the trigger occurred
CommandAborted	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	The current instruction is interrupted and is set to TRUE

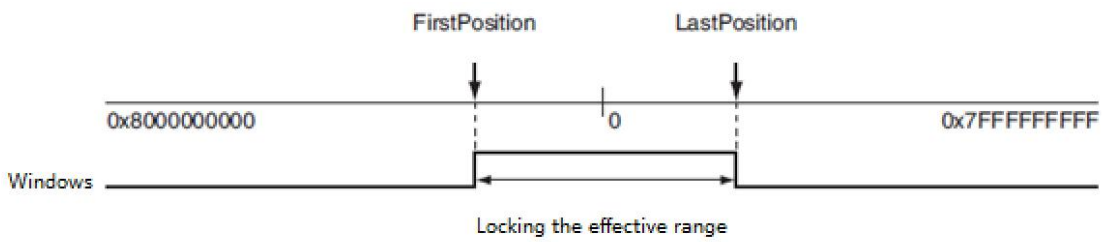
3) Function descriptions

Description of probe functions

- The probe function is designed to enable position control based on the occurrence of a trigger signal, such as a sensor input, and to record (lock) the axis position when a trigger signal occurs. Normally 2 trigger points can be set for each axis at the same time.
- The MC_TouchProbe (enable external locking) command can be used to specify "trigger input conditions" and "enable window" for the axis to be locked. The trigger signal can specify a variable that can be used by the user program in addition to the signal to which the servo drive is connected. To terminate the locking function, use the MC_AbortTrigger command.
- Locking function available for VC servo drives and other servos, encoders, etc. that support the probe function.
- When WindowOnly is used, the trigger signal is only detected within the range of the start and end points. The ranges for the different counting modes are shown below.

Linear mode

- Detectable only if FirstPosition ≤ LastPosition.
- If FirstPosition > LastPosition is specified, an exception will be thrown.
- An exception is thrown when a position range is specified beyond the linear mode

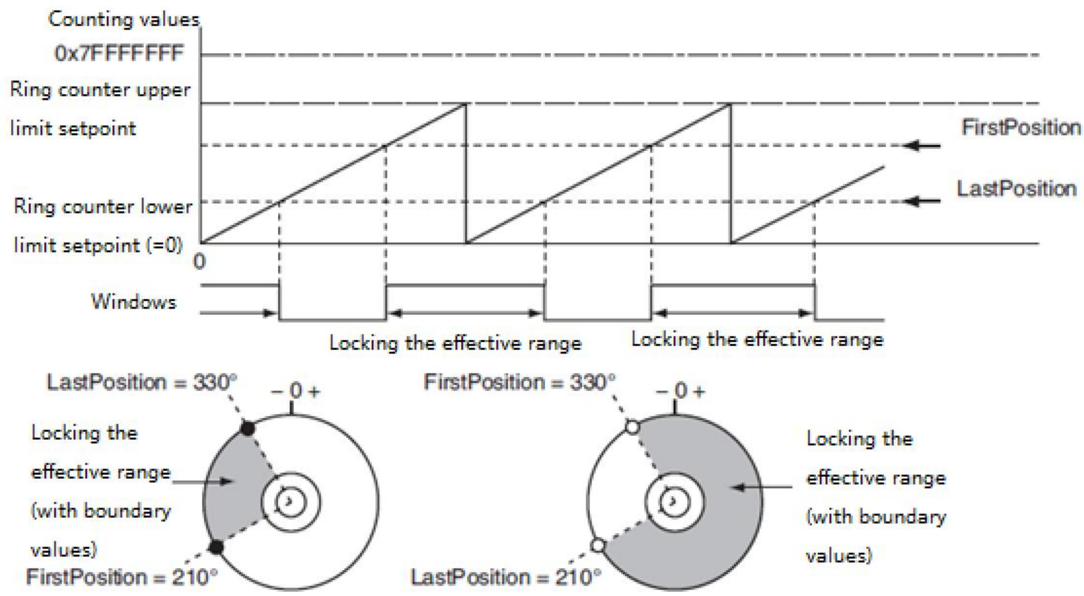


Rotation mode

Both FirstPosition ≤ LastPosition and FirstPosition > LastPosition can be specified. When the latter is specified, it is set to cross the lower limit setting of the ring counter.

If you specify beyond the upper or lower limit of the ring counter, the command will cause an exception.

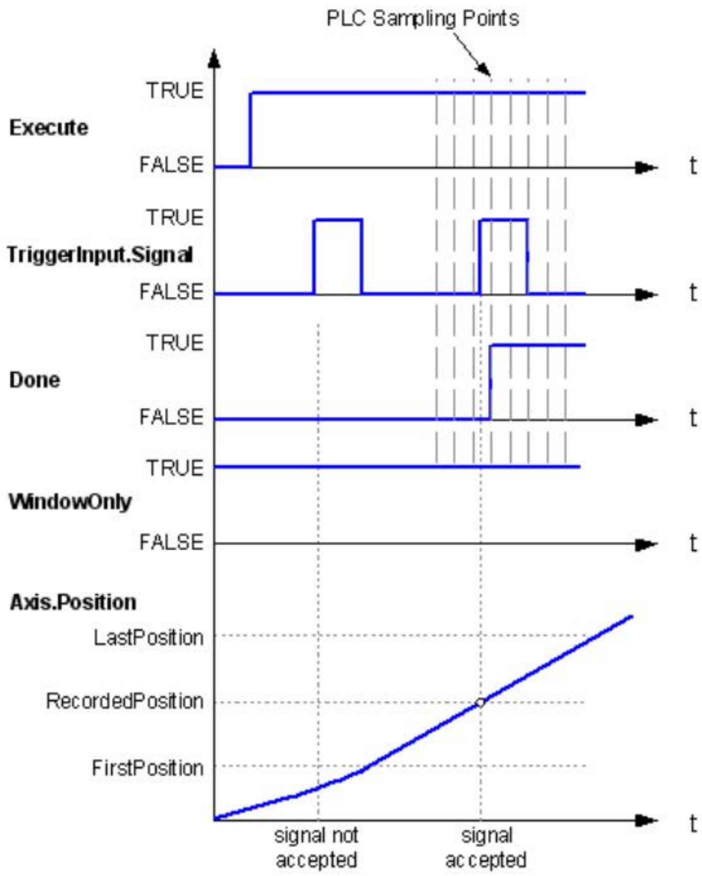
	First Position ≤ Last Position	First Position > Last Position
Effective range	FirstPosition ~ LastPosition	LastPosition ~ FirstPosition



There are two methods of obtaining the latch position, each of which is described below:

- **MC_TuochProbe command get**

The current position of the running axis is recorded when triggered by a signal from the MC_TouchProbe function block TruggerInput. execute Execute on rising edge, drive latch: the drive picks up the latch signal at the recorded position and then transmits it to the controller.



Structure data type TRIIGGER_REF describes the shaft driver used by the probe input and determines which probe number corresponds to which hardware probe.

The name of the member	The data type	The initial value	Describe
iTriggerNumber	INT	-1	Trigger channel; defined by the driver (only used when bFastLatching'.'
bFastLatching	BOOL	TRUE	TRUE: The lock is present in the drive and is completed using the probe function defined by the servo shaft 60B8 (precise). FALSE: In the motion task cycle,bInput is latched (inaccurate).
bInput	BOOL		The internal latch signal, which is valid when bFastLatching is false.
bActive	BOOL	FALSE	The probe status, true, states that the probe is active

When bFastLatching: s TRUE, take the Wykoda BusServo VECServo asan example, the relationshipbetween the iTriggerNumber number and the servo probe is as follows:

iTriggerNumber	The hardware DI and edges of the servo probe
0	Servo DI9 rising edge latch

1	Servo DI9 drops along the latch
2	Servo DI10 rising edge latch
3	Servo DI10 drops along the latch

Familiarisation with VC servo-probe function

Index 16#60B8

Bit	Function	
0	Probe 1 enabled. 0 - Probe 1 not enabled 1 - Probe 1 enabled	Bit0~Bit5: Probe 1 related settings ◆Note. Once the probe 1 enable signal (rising edge of bit0 of 60B8h) is valid, the function settings of probe 1 (trigger mode, trigger signal, valid latching edge) cannot be changed, and bit0 of 60B8h must remain valid during the action of probe 1. DI9 can enable both its rising and falling edges when used as the probe 1 trigger signal.
1	Probe 1 trigger mode 0-Single trigger, triggered only when the trigger signal is valid for the first time 1-Continuous triggering	
2	Probe 1 trigger signal selection 0-DI9 input signal 1-Z signal	
3	RES	
4	Probe 1 rising edge enable 0 - no latching on rising edge 1 - rising edge latched	
5	Probe 1 falling edge enable 0 - no latching on falling edge 1 - falling edge latch	
6-7	RES	
8	Probe 2 enabled. 0 - Probe 2 not enabled 1 - Probe 2 enabled	Bit8~Bit15: Probe 2 related settings ◆Note: Once the probe 2 enable signal (the rising edge of bit 8 of 60B8h) is valid, the function settings of probe 2 (trigger mode, trigger signal, valid latch edge) cannot be changed, and while the probe 2
9	Probe 2 trigger mode 0-Single trigger, triggered only when the trigger signal is valid for the first time 1 - Continuous trigger	

10	Probe 2 trigger signal selection 0-DI10 input signal 1-Z signal	is working, bit 8 of 60B8h must Keep it effective. When DI10 is used as the trigger signal of probe 2, its rising edge and falling edge can be enabled at the same time.
11	RES	
12	Probe 2 rising edge enable 0 - no latching on rising edge 1 - rising edge latched	
13	Probe 2 falling edge enable 0 - no latching on falling edge 1 - falling edge latch	
14-15	RES	

Configuring PDO

The following PDO must be configured to use the probe function.

Output

16#60B8 (probe function)

Inputs

16#60B9 (probe status)

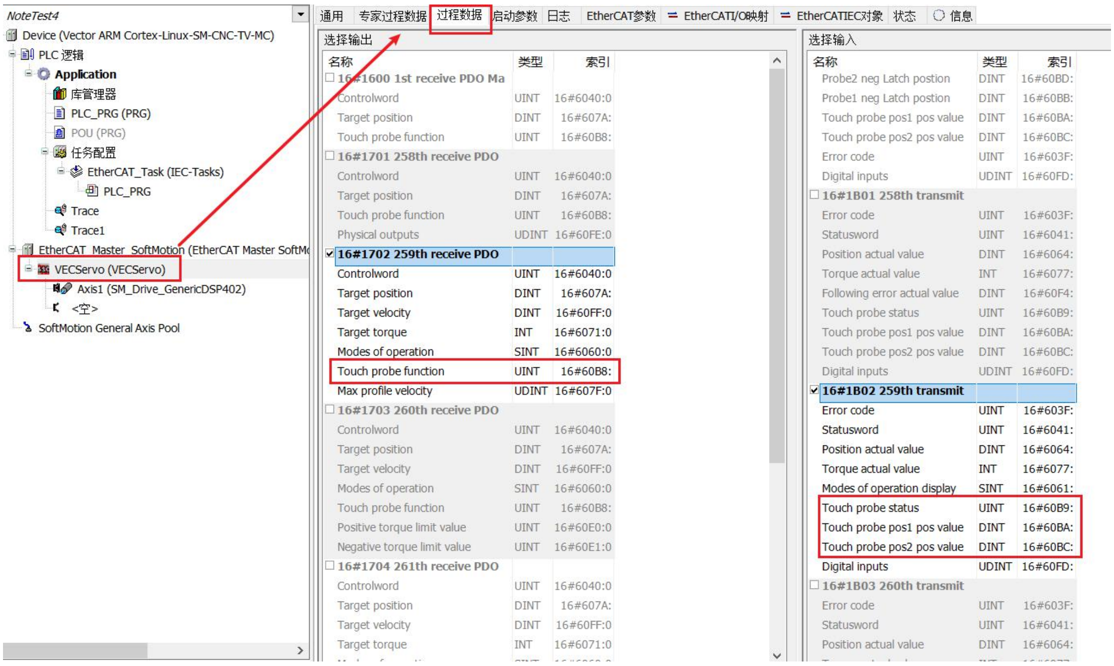
16#60BA (Probe 1 rising edge position latch) // selected according to 60B8 value

16#60BB (Probe 1 falling edge position latch) //selected according to 60B8 value

16#60BC (Probe 2 rising edge position latch) //selected according to 60B8 value

16#60BD (Probe 2 falling edge position latching)//selected according to 60B8 value

Configure the appropriate probe function according to actual needs, as shown below. Or configure the probe related PDO in the 16#1600 and 16#1A00 groups.



- Calling the MC_TouchProbe command

表达式	类型	值	准备值	地址	注释
Tri_input	SM3_Basic.TRIGGER_REF				
iTriggerNumber	INT	0			Trigger channel; defi... by driver (only use...
bFastLatching	BOOL	TRUE				``TRUE``: Latching...done in drive (preci...
bInput	BOOL	FALSE			Trigger signal when ``bFastLatching`` = `...
bActive	BOOL	FALSE			Internal variable

Note that a value needs to be assigned to TrInput.iTriggerNumber. Enable command, trigger signal.

```
Tri_input.iTriggerNumber 0 :=inumber1 0 ;
Tri_input.bFastLatching TRUE :=bFastLatching1 TRUE ;
Tri_input.bInput FALSE :=binput1 FALSE ;
bactive1 FALSE :=Tri_input.bActive FALSE ;

TouchProbe1(
    Axis:= Axis1,
    TriggerInput:= Tri_input,
    Execute TRUE := TouchProbe1_Excute TRUE ,
    WindowOnly FALSE := TouchProbe1_WindowOnly FALSE ,
    FirstPosition 0 := TouchProbe1_FirstPosition 0 ,
    LastPosition 0 := TouchProbe1_Lastposition 0 ,
    Done TRUE => TouchProbe1_Done TRUE ,
    Busy FALSE => TouchProbe1_Busy FALSE ,
    Error FALSE => TouchProbe1_Error FALSE ,
    ErrorID SMC_NO_ERR => TouchProbe1_ErrorID SMC_NO_ERR ,
    RecordedPosition 91.9 => TouchProbe1_Position 91.9 ,
    CommandAborted FALSE => TouchProbe1_Abt FALSE );
```

Note: When using the MC_TuochProbe command to capture a position, only a single capture can be made, not a continuous capture, if you need to capture a position continuously, use the method of directly modifying the PDO

- **Direct modification of PDO**

  Object index 16 s 60B8

  MC_TuochProbe instructions are only supported

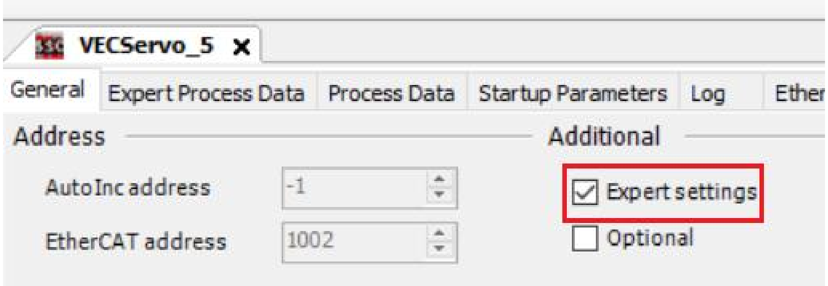
  0: Probe index 60B8 s 16 s 11;

1: Probe index 60B8 s 16 s 21;

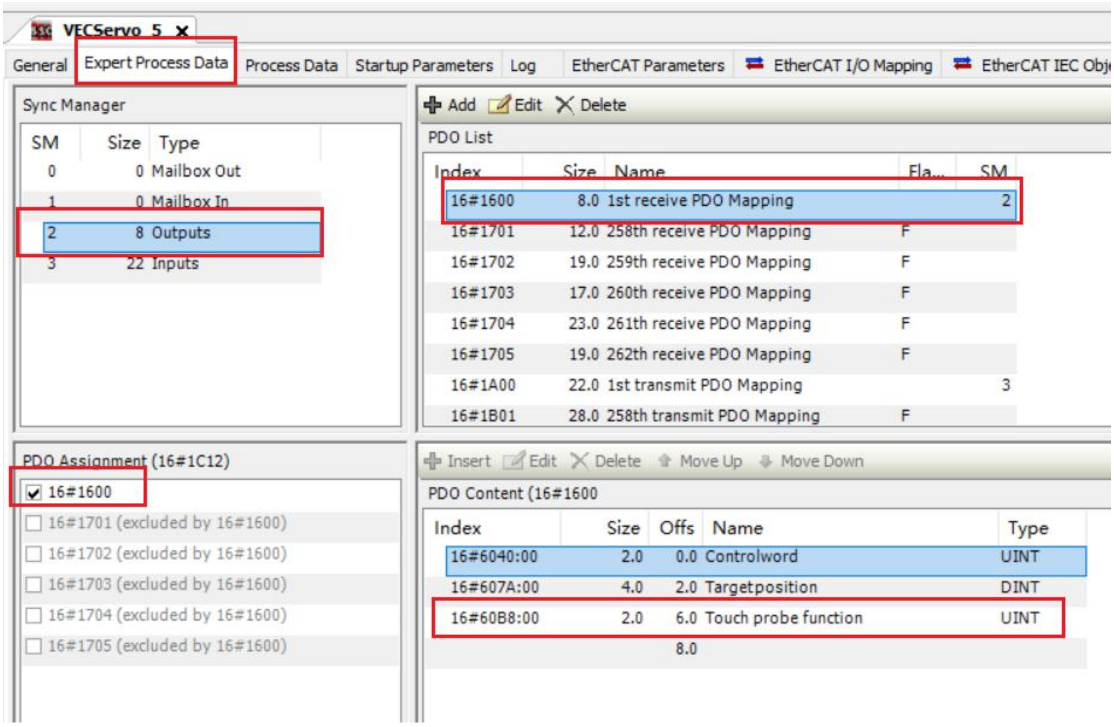
2: Probe index 60B8 s 16 s 1100;

3: Probe index 60B8 s 16 s 2100 four modes. The VC supports more modes, using a probe function other than the four above. You can set the index directly. The setting method is as follows.

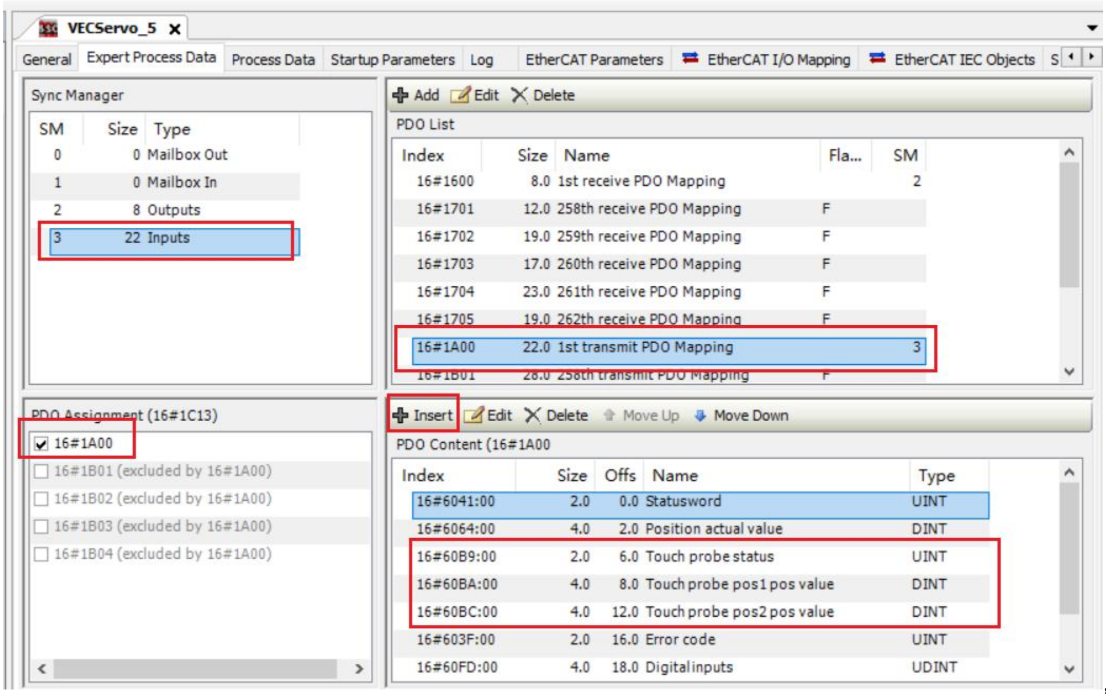
- Select the from which you want to control and check Enable Expert Mode:



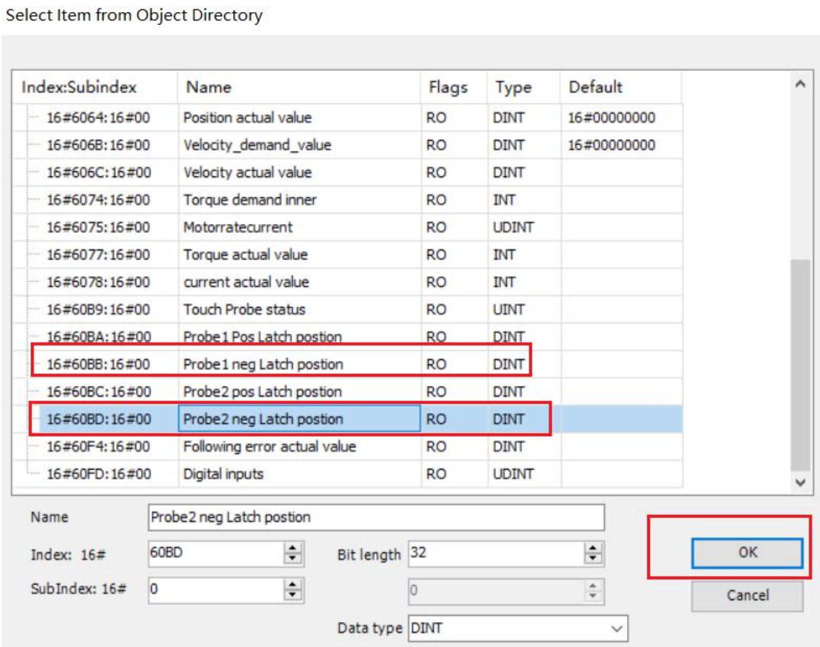
(2) Convert to the "Expert Process Data" interface,click "Output" in the synchronizationmanager, check "16 1600"in the P DO assignment, you can see that there is already"16 s 60B8 probe function" in the1st RPDO at this time, without modification:



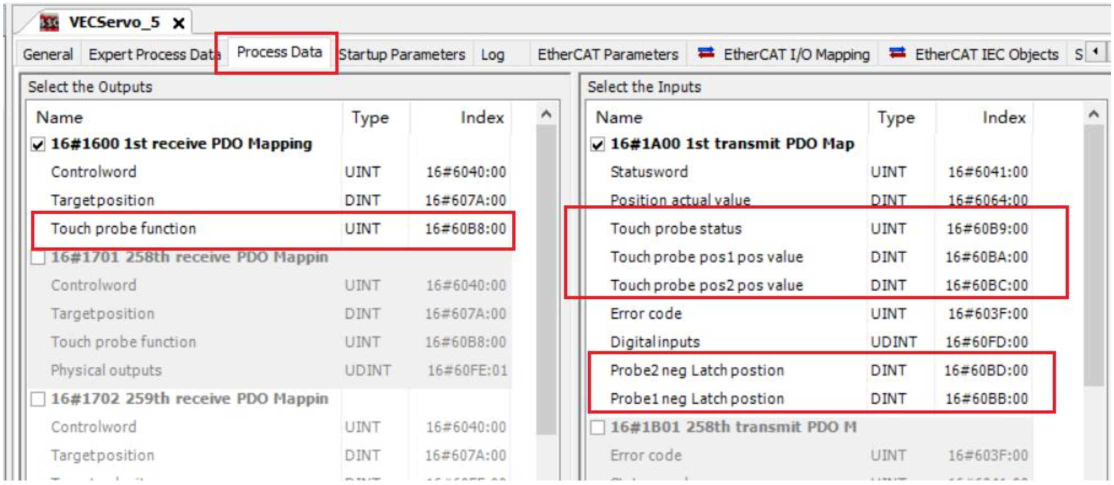
(3) Sync Manager selects "Input", PDO assignmentclicks 1 6 1A00, selects 1 st TPDOwith SM on the right, and can see that there is only "probe status, probe one or two rising edge latch position" in the group at this time, click "Insert" if you want to add the falling edgelatch position



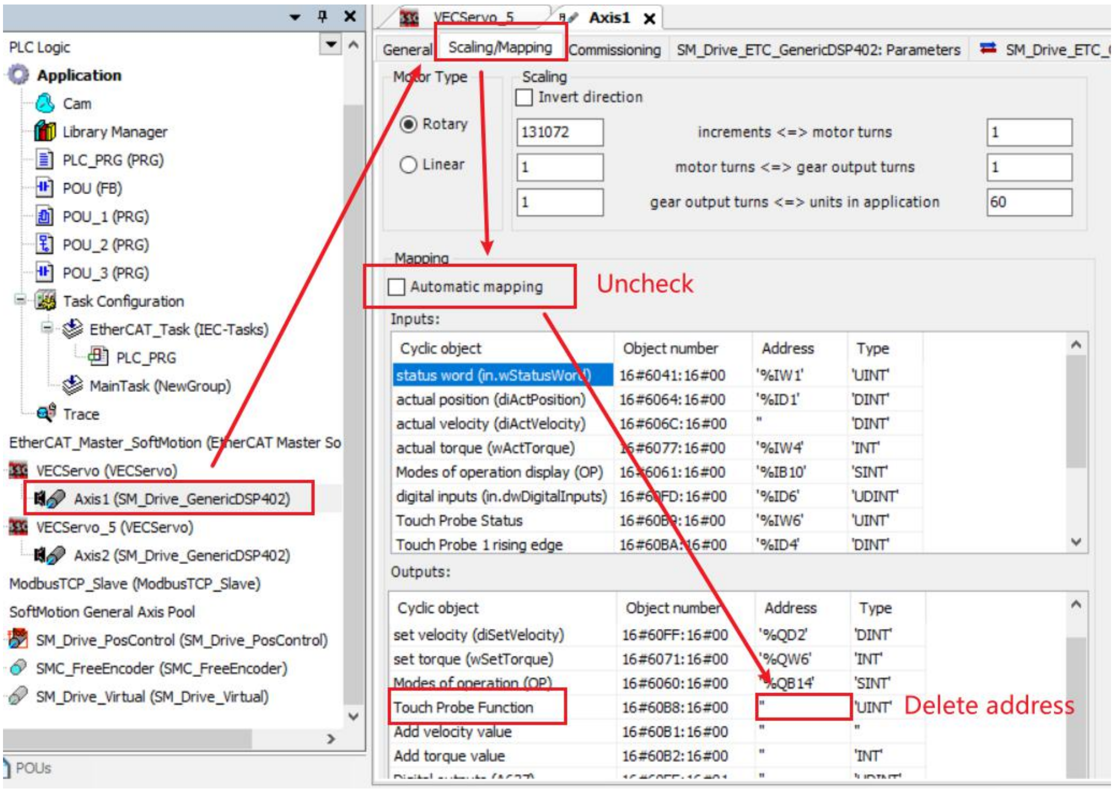
(4) Find 60BB and 60BD,select Click OK toadd:



(5) Click on the process data and you will now see the PDO required for the probe function you have just added:



(6) Modify the axis parameter setting, do not select automatic mapping, and delete the address corresponding to the output output parameter probe function.

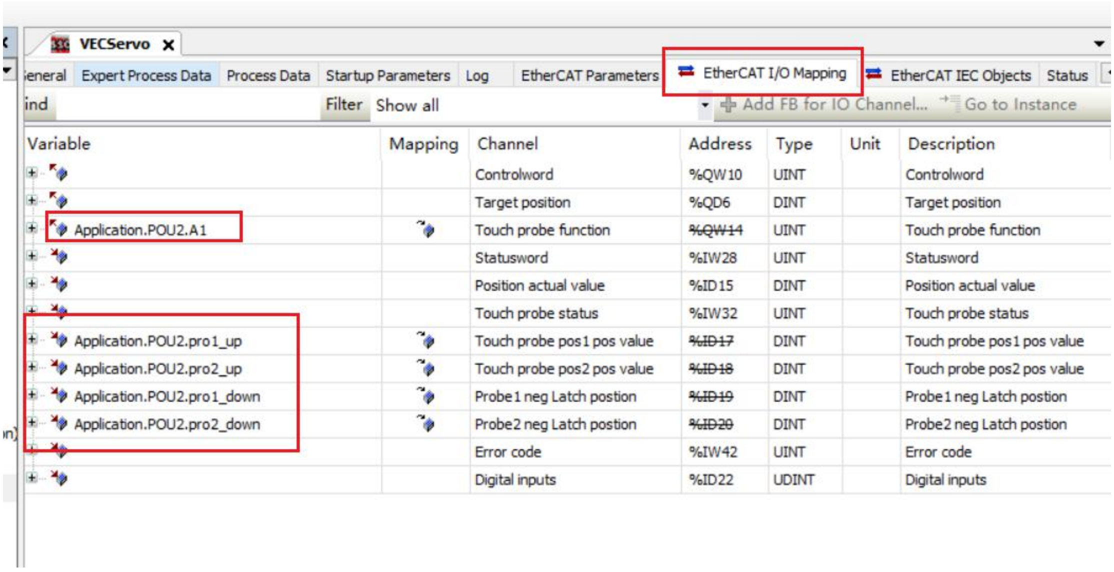


(7) Set the probe function communication address in the program. Refer to VC manual introduction 16#60B8 for specific setting values. here configured as 16#60B8 = 2# 0011 0011 0011 0011 = 13107 with the following functions.

Probe 1 enable, continuous latching, rising edge latching, falling edge latching, latching via DI9

Probe 2 enable, continuous latch, rising edge latch, falling edge latch, latch via DI10

Scope	Name	Address	Data type	Initialization
VAR	ALL		UINT	13107
VAR	pro1_up		DINT	
VAR	pro1_down		DINT	
VAR	pro2_up		DINT	
VAR	pro2_down		DINT	



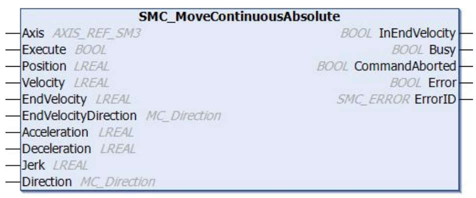
Trigger configuration DI9, DI10, the latch position is saved in the variable and the latch result is as follows

INT60B8	UINT	13107
Pos1_Up	DINT	15309
Pos1_Down	DINT	9678
Pos2_Up	DINT	16110
Pos2_Down	DINT	23561

7.3.26 SMC_MoveContinuousAbsolute

The axes run continuously in absolute position (units are set by axis), the absolute position is specified by Position, and the last running speed, EndVelocity, is run; the relevant parameters, Acceleration, Deceleration and Velocity, are set before this instruction is run. Velocity; an assignment of 0 to Acceleration or Deceleration is an error; during operation, it is important to pay attention to the complete operation of this instruction to avoid interruptions by other instructions from the user program's design point of view.

1) Command format

Instructions	Name	Graphical performance	ST performance
MC_Move ContinousAbsolute	The absolute position of the axis continuously controls the instructions		

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	An up-edge of the input will initiate the processing of the function block
Position	The motion is absolutely right for the position	LREAL		0	This data is the absolute position of the motion
Velocity	The speed at which it is run	LREAL		0	The maximum speed at which the axis runs to the target position
EndVelocity	The end speed of the run	LREAL		0	The speed at which the instruction is executed
EndVelocity Direction	The direction of the end speed	MC_Direction	positive, negative, current;	Current	Can be used: positive, negative, current; Not available: shortest, fastest

Acceleration	Acceleration	LREAL		0	Acceleration value as the velocity increases
Deceleration	Reduce the speed	LREAL		0	Speed changes by hours and decreases the speed value
Jerk	Acceleration rate of change	LREAL		0	Acceleration
Direction	The direction of operation	shortest		shortest	For linear / linear axes: positive, negative; For rotation / circumferon axis: positive, negative, current, shortest, fastest

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
InEndVelocity	The command position arrives	BOOL	TRUE,FALSE	FALSE	The axis command execution position arrives and is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
CommandAbort	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	The current instruction is interrupted and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Function description

This function block is an absolute axis positioning command, where the Distance data is the absolute position of the axis.

The running state of this function block is in Standstill, the state of the instruction is Discrete Motion, a complete running process must control the different motion states of the axis.

The start command is the rising edge of Execute. This command can be repeated on the rising edge of Discrete Motion, refreshing the latest Position position each time.

Acceleration or Deceleration is zero, the instruction runs in an abnormal state, but the state of the axis is Discrete Motion；

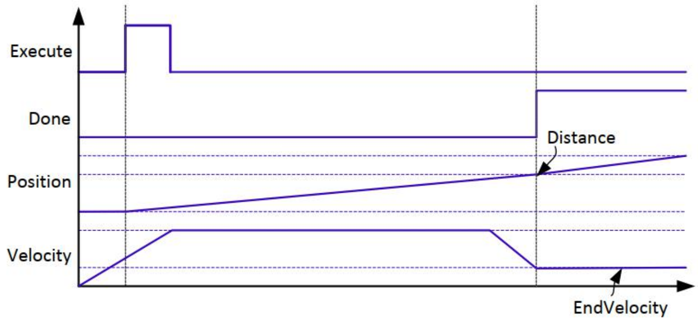
◆ Timing diagram

the axis must be in the Standstill state for the instruction to run.

Execute of a function block must have a rising edge condition.

Done of a function block indicates that the instruction has completed normal execution.

Busy of a function block indicates that the function block is currently being executed；



Execute
Done
Distance
Position
Velocity
EndVelocity

7.3.27 SMC_MoveContinuousRelative

The axes run continuously in relative position (units are set by axis), the relative position is specified by Distance, and the final running speed, EndVelocity, is run; the relevant parameters, Acceleration, Deceleration and Velocity, are set before the instruction is run. Velocity; a value of 0 for Acceleration or Deceleration is an error; during operation, it is important to pay attention to the complete operation of this instruction to avoid interruptions by other instructions from the user program's design point of view.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_Move ContinuousRelative	The axis is relative to the positioning instruction		

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The execution condition	BOOL	TRUE,FALSE	FALSE	An up-edge of the input will start the processing of the function block
Distance	The relative position of the motion	LREAL	The range of data	0	This data is the relative position of the motion
Velocity	The speed at which it is run	LREAL	The range of data	0	The maximum speed at which the axis runs to the target position
EndVelocity	The end speed of the run	LREAL	The range of data	0	The speed at which the instruction is executed
EndVelocity Direction	The direction of the end speed	MC_Direction	positive, negative,	Current	Can be used: positive, negative,

			current;		current; Not available: shortest, fastest
Acceleration	Acceleration	LREAL	The range of data	0	Acceleration value as the velocity increases
Deceleration	Reduce the speed	LREAL	The range of data	0	Speed changes by hours and decreases the speed value

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
InEndVelocity	The command position arrives	BOOL	TRUE,FALSE	FALSE	The axis command execution position arrives and is set to TRUE
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
CommandAbort	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	The current instruction is interrupted and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Function description

This function block runs in Standstill, and the state of the instruction is Discrete Motion, so as to avoid interrupting the execution of other instructions in this axis or being interrupted by other instructions.

The start instruction is the rising edge of Execute, which can be repeated on the rising edge of Discrete Motion to refresh the latest Position each time.

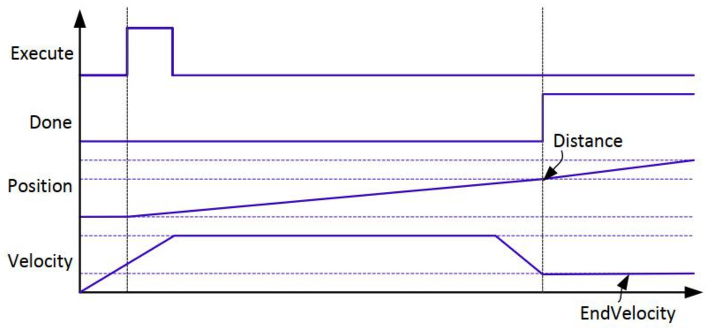
Acceleration or Deceleration is zero, the instruction runs in an abnormal state, but the state of the axis is Discrete Motion;

- Timing diagram

Execute of a function block must have a rising edge condition.

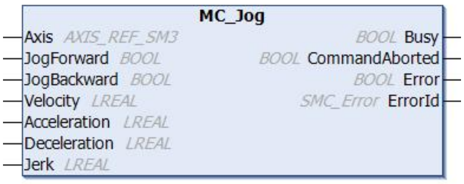
Done of a function block indicates that the instruction has been executed normally.

Busy of a function block indicates that the current function block is being executed.



7.3.28 MC_Jog

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_Jog	Pivot point command		MC_Jog(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

2) Related variables

input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
JogForward	Positive is valid	BOOL	TRUE,FALSE	FALSE	Set to TRUE to start moving forward, and set to FALSE to stop moving forward
JogBackward	Negative is valid	BOOL	TRUE,FALSE	FALSE	Set to TRUE to start moving in reverse; Set to FALSE to stop the reverse movement
Velocity	Target speed	LREAL	Positive or "0"	0	Specify the target speed. Unit: (Instruction unit /s)
Acceleration	Acceleration	LREAL	Positive or "0"	0	Specifies acceleration. In:
Deceleration	Reduce the speed	LREAL	Positive or "0"	0	Specifies a reduction in speed. In:
Jerk	Acceleration	LREAL	Positive or "0"	0	The rate of change in the command acceleration. In:(instruction unit /s^3. .

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Busy	In action	BOOL	TRUE,FALSE	FALSE	When the instruction is received, it is set to TRUE
CommandAborted	The execution is interrupted	BOOL	TRUE,FALSE	FALSE	When the instruction is aborted, it is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE

ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

Input and output variables

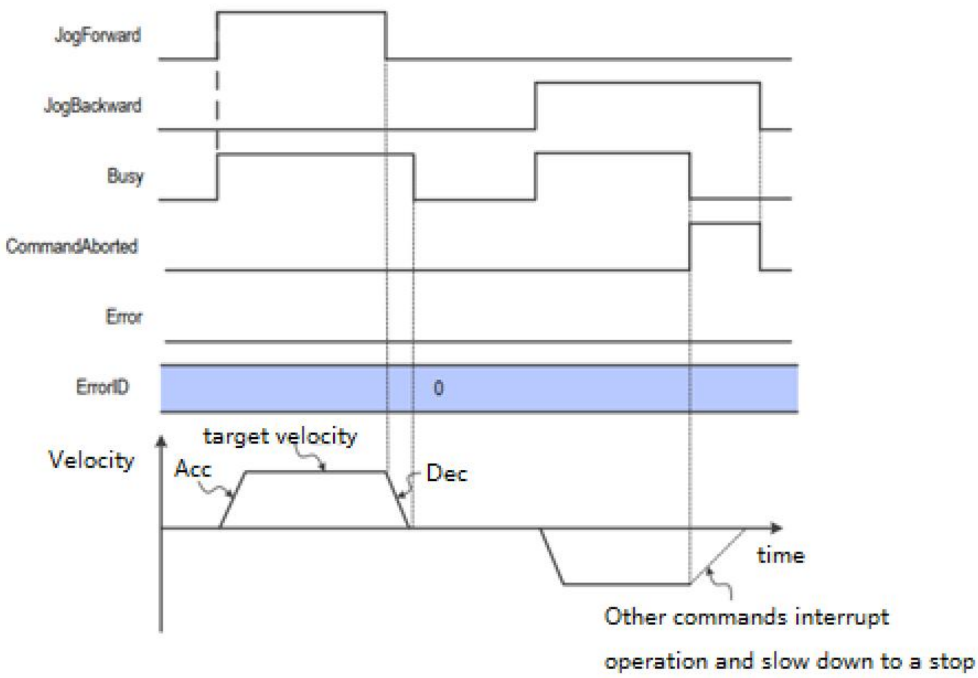
Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

3) Function description

Performs a jog according to the specified Velocity.

If a forward run is required, set JogForward to TRUE; if a reverse run is required, set JogBackward to TRUE.

By setting both JogForward (valid for forward running) and JogBackward (valid for negative running) to TRUE, no movement will occur. If the command speed setting of the MC_Jog instruction exceeds the pointing maximum speed in the axis parameter, it will be executed at the pointing maximum speed.

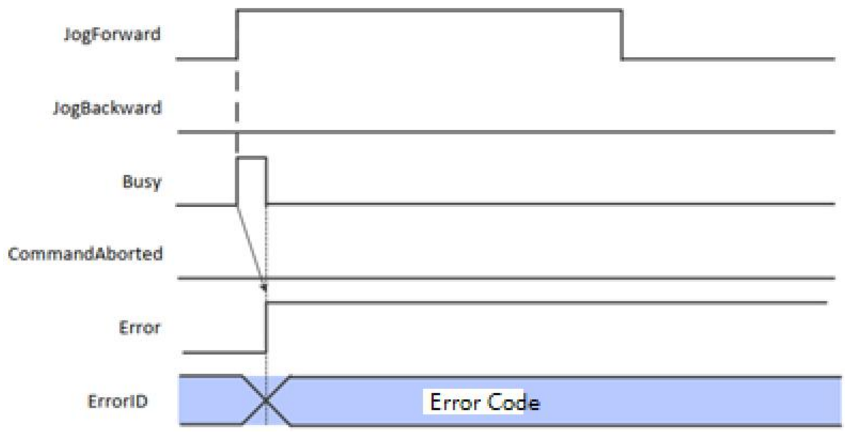


4) Error description

When an exception occurs during the execution of this instruction, Error becomes TRUE and the axis stops moving.

You can check the output value of ErrorID (error code) to understand the cause of the exception.

◆ Timing diagram when an exception occurs



Please read "Appendix C Error Code Descriptions" for a description of the error codes that occur with the command.

7.3.29 SMC_Inch

Axis step-by-step motion control, through the program can achieve step-by-step step control.

1) Instruction format

Instructions	Name	Graphical performance	ST performance

SMC_Inch	The axis is relative to the positioning instruction		`SMC_Inch0(`
			`Axis:= ,`
			`InchForward:= ,`
			`InchBackward:= ,`
			`Distance:= ,`
			`Velocity:= ,`
			`Acceleration:= ,`
			`Deceleration:= ,`
			`Jerk:= ,`
			`Busy=> ,`
			`CommandAborted=> ,`
			`Error=> ,`
			`ErrorId=> );`

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
InchForward	Is executing in the right way	BOOL	TRUE, FALSE	FALSE	If InchForward is TRUE, the axis will move at a given speed (Velocity, Speed, Deceleration) in a positive direction until the distance is reached. The input must be specified as FALSE before the motion is started again for TRUE. If InchForward is set to FALSE before it reaches its position, the axis will immediately decelerate to 0 and Busy will be set to FALSE. If the input InchBackward is set to TRUE in simulation, no motion will be generated.
InchBackward	Reverse execution	BOOL	TRUE, FALSE	FALSE	If InchBackward is TRUE, the axis will move at a given speed value (Velocity, Speed, Deceleration) in reverse motion to

					the set position. The input must then be set toFALSE, and then set to TRUE to start another motion. If the input signal InchForward is also set to TRUE, there will be no axis motion.
Distance	The distance moved	LREAL	The range of data	0	This data is the distance of motion
Velocity	The speed at which it is run	LREAL	The range of data	0	The maximum speed at which the axis runs to the target position
Acceleration	Acceleration	LREAL	The range of data	0	Acceleration value as the velocity increases
Deceleration	Reduce the speed	LREAL	The range of data	0	Speed changes by hours and decreases the speed value

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Busy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
CommandAbortand	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	The current instruction is interrupted and is set to TRUE
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

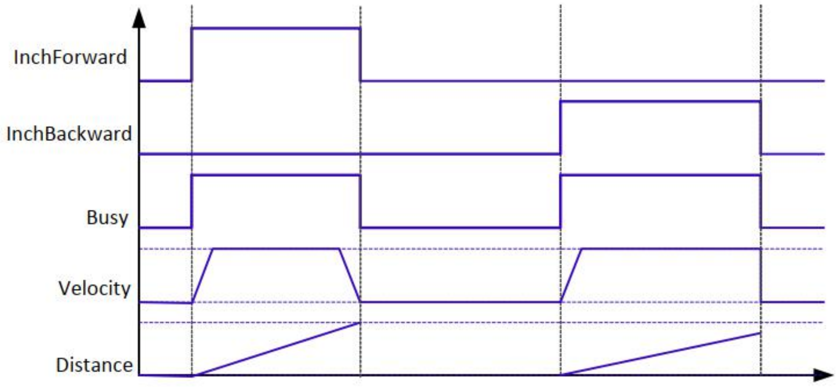
3) Function Description

This function block runs in Standstill, and the state of the instruction is Discrete Motion, so as to avoid interrupting other instructions of the axis or being interrupted by other instructions during the execution of the instruction. state, but the state of the axis is Discrete Motion.

◆ Timing diagram

InchForward/InchBackward of the function block must have the condition TRUE/FALSE.

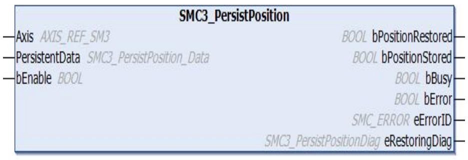
Busy of a function block means that the block is currently being executed;


InchForward
InchBackward
Busy
Velocity
Distance

7.3.30 SMC3_PersistPosition

This instruction is used to maintain the position of the recorded solid-axis absolute value encoder (the position record value before the power-off is restored after the controller is restarted). If the servo motor is using an absolute value encoder, use this function block in conjunction.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC3_ PersistPosition	The axis position is maintained		SMC3_PersistPosition0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
PersistentData	Keep the data	SMC3_PersistPosition_Data			The power-off-hold data structure that stores location information

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	True function blocks are executed, false does not execute function blocks, and to restore the last stored location during initialization, the value must be set to true when the application starts

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bPosition Restored	Location recovery	BOOL	TRUE,FALSE	FALSE	TRUE, position recovery completes after axis restart
bPosition	Location	BOOL	TRUE,FALS	FALSE	TRUE, the save location is

Stored	save		E		complete after calling the function block
bBusy	FB in action	BOOL	TRUE,FALSE	FALSE	TRUE, the function block is not executed
bError	Error	BOOL	TRUE,FALSE	FALSE	TRUE, an exception occurs
eErrorID	The error code	SMC_ERROR	SMC_NO_ERROR	When an exception occurs, the error code is output	
eRestoringDiag	Restore diagnostics	SMC3_PersistPositionDiag		SMC3_PersistPositionDiag.SMC3_PPD_RESTORING_OK	Diagnostic information in location recovery: SMC3_PPD_RESTORING_OK: Location Recovery SMC3_PPD_AXIS_PROP_CHANGED: The axis parameters have changed and position cannot be recovered SMC3_PPD_DATA_STORED_DURING_WRITING: The function block copies the data from the axis parameter data structure instead of from the PersistentData data. Possible causes: non-synchronous persistent variables, controller crash crash

3) Function description

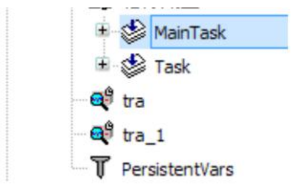
The PLC restart bEnable signal is TRUE, the bPositionRestroed output is TRUE.

Dummy axes and logical axes are not supported.

The actual position of the axes in the VE controller is: Offset + Coded feedback position (command unit Plus) * Scale, the position recorded by the absolute encoder is the command unit value. This function block is therefore required to restore the "actual position" before the power failure after the PLC has been restarted and to record the "actual position" of the axis before the power failure, the SMC3_PersistPosition_Data must be set to the continuous type variable".

Usage (when the real axis encoder is a multi-turn absolute value)

④ Declare the SMC3_PersistPosition_Data type in PersistentVars



Device PersistentVars

```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData1: SMC3_PersistPosition_Data;
END_VAR
```

⑤ Called from the PLC main task (EthCat task)

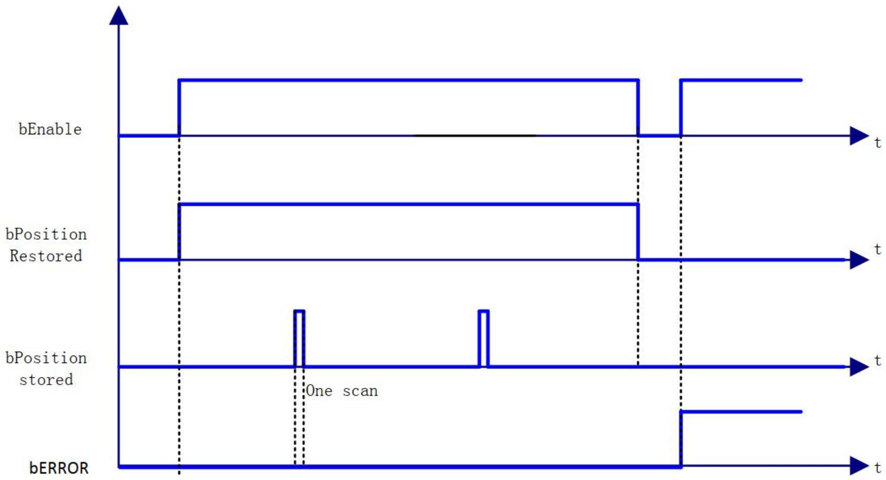
◆ Affirmation section.

```
VAR
    SMC3_PersistPosition_1:SMC3_PersistPosition;
END_VAR
```

Program section:

```
//绝对值位置保存
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

◆ Time-series diagram



4) Error description

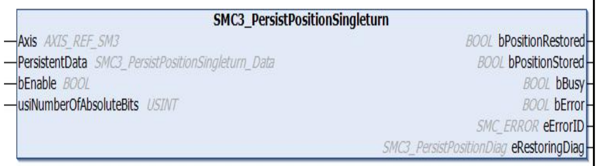
An input axis that is a virtual axis or a logical axis will result in an error output; an error in an axis will result in an error output.

[Note]: Please read "Appendix C Error Code Descriptions" for error code descriptions.。

7.3.31 SMC3_PersistPositionSingleturn

This instruction is used to maintain the position of the recorded solid-axis absolute value encoder (single-turn absolute value) (after the power-off restarts the controller, the pre-power-off position record value is restored).
If the servo motor is using a single-turn absolute value encoder, use this function block in conjunction.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC3_ PersistPosition Singleturn	The axis position is maintained		SMC3_PersistPositionSingleturn_0(Axis:= , PersistentData:= , bEnable:= , usiNumberOfAbsoluteBits:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
PersistentData	Keep the data	SMC3_ PersistPosition Singletrun_Data			The power-off-hold data structure that stores location information

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	True function block execution, false does not execute function block To restore the last stored location during initialization, the value must be set to true when the application starts
usiNumberofAbsoluteBitesusiNumb	The number	UINT		16	How many bits of absolute value encoder (e.g. 20 bits, 24 bit encoder,

erofAbsoluteBites	of digits				etc.)

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bPositionRestored	Location recovery	BOOL	TRUE,FALSE	FALSE	TRUE, position recovery completes after axis restart
bPositionStored	Location save	BOOL	TRUE,FALSE	FALSE	TRUE, the call function is done quickly after saving the location
bBusy	FB in action	BOOL	TRUE,FALSE	FALSE	TRUE, the function block is not executed
bError	Error	BOOL	TRUE,FALSE	FALSE	TRUE, an exception occurs
eErrorID	The error code	SMC_ERROR	SMC_NO_ERROR	When an exception occurs, the error code is output	
eRestoringDiag	Restore diagnostics	SMC3_ PersistPositionDiag		SMC3_PersistPositionDiag. SMC3_PPD_RESTORING_OK	Diagnostic message in position recovery SMC3_PPD_RESTORING_ OK: Position successfully recovered SMC3_PPD_AXIS_PROP_ CHANGED: Axis parameter changed, position cannot be recovered SMC3_PPD_DATA_STORED_DURING_WRITING: Function block copied from axis parameter data structure instead of copying from PersistentData data. Possible causes: Non-synchronous persistent variable, controller crash dead

3) Function description

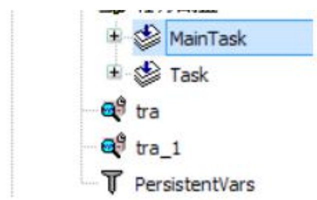
The PLC restart bEnable signal is TRUE, the bPositionRestroed output is TRUE.

Dummy axes and logical axes are not supported.

This function block is required to restore the "actual position" before the power failure after the PLC has been restarted and to record the "actual position" of the axis before the power failure, the SMC3_
PersistPositionSingleTurn_Data as a continuous variable".

Usage (when the real axis encoder is a multi-turn absolute value)

① Declare SMC3_PersistPositionSingleTurn_Data in PersistentVars




```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData2: SMC3_PersistPositionSingleTurn_Data;
END_VAR
```

② Called from the PLC main task (EthCat task)

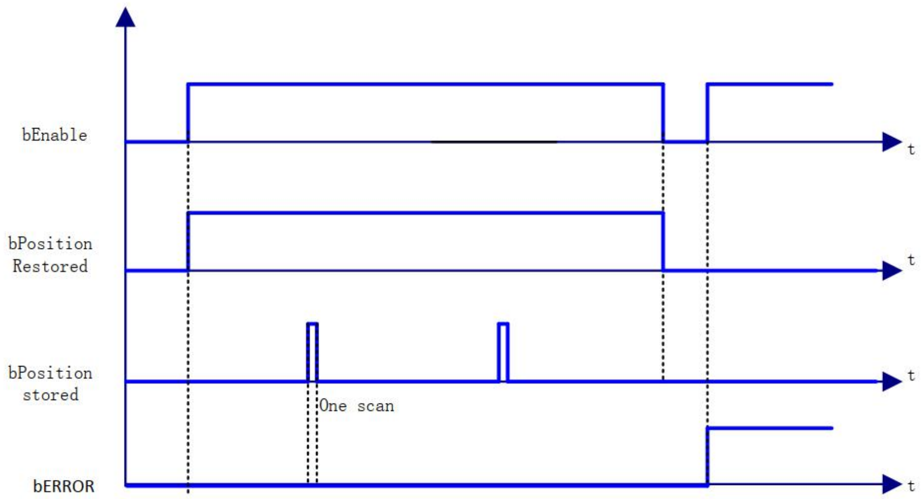
◆ Affirmation section.

```
VAR
    SMC3_PersistPosition_2: SMC3_PersistPositionSingleTurn_Data;
END_VAR
```

◆ Program section:

```
//绝对值位置保存
SMC3_PersistPosition_2(Axis:=Y_Axis , PersistentData:=persistentData2 ,bEnable:=TRUE );
```

4) Time-series diagram



5) Error description

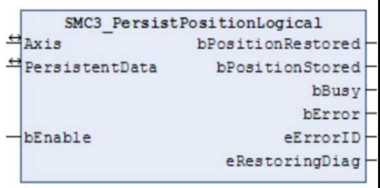
An input axis that is a virtual axis or a logical axis will result in an error output; an error in an axis will result in an error output.

[Note]: Please read "Appendix C Error Code Descriptions" for error code descriptions.。

7.3.32 SMC3_PersistPositionLogical

This command is used to keep track of the position of the logical axes (right click on the real or imaginary axis to "add device" to select the logical axis to be added) (after a power failure and restart of the controller, the value of the position recorded before the power failure is restored).

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC3_ PersistPosition Logical	Axis position holding		SMC3_PersistPositionLogical0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF_ LOGICAL_SM3	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
PersistentData	Maintain data	SMC3_PersistPositionLogical_ Data			The power-off-hold data structure that stores location information

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	执行	BOOL	TRUE,FALSE	FALSE	True function blocks are executed, false function blocks are not executed To restore the last stored location during initialisation, this value must be set to true from application start-up

The output variable

The	Name	The data	Effective	The initial	Describe

output variable		type	range	value	
bPosition Restored	Location Recovery	BOOL	TRUE, FALSE	FALSE	TRUE, position recovery complete after axis restart
bPosition Stored	Position saving	BOOL	TRUE, FALSE	FALSE	TRUE, position saved after function call completed
bBusy	FB Execution in progress	BOOL	TRUE, FALSE	FALSE	TRUE, function block not completed
bError	Error	BOOL	TRUE, FALSE	FALSE	TRUE, exception occurred
eErrorID	Error code	SMC_ERROR	SMC_NO_ ERROR	Output an error code when an exception occurs	
eRestoring Diag	Recovery diagnosis	SMC3_ PersistPosition Diag		SMC3_ PersistPositionDia g.SMC3_PPD_ RESTORING_OK	Diagnostic messages in position recovery SMC3_PPD_RESTORING_OK: Position successfully recovered SMC3_PPD_AXIS_PROP_ CHANGED: Axis parameters have been changed, position cannot be recovered SMC3_PPD_DATA_STORED_DU RING_WRITING: The function block copies data from the axis parameter data structure instead of from Persistent Data data instead of copying from Persistent Data. Possible causes: Non-synchronous persistent variables, controller crash dead

3) Function description

The PLC restart bEnable signal is TRUE, the bPositionRestroed output is TRUE.

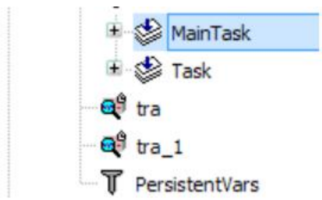
The dummy axis and the real axis are not supported.

This function block is required to restore the "position" before the power failure after the PLC has been restarted and to record the "actual position" of the axis before the power failure, the SMC3_

PersistPositionLogical_Data as a continuous variable".

Usage (when the real axis encoder is a multi-turn absolute value).

Declare SMC3_PersistPositionLogical_Data in PersistentVars




```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData3: SMC3_PersistPositionLogical_Data;
END_VAR
```

Invoked in the PLC main task (EthCat task)

◆ Affirmation section.

VAR

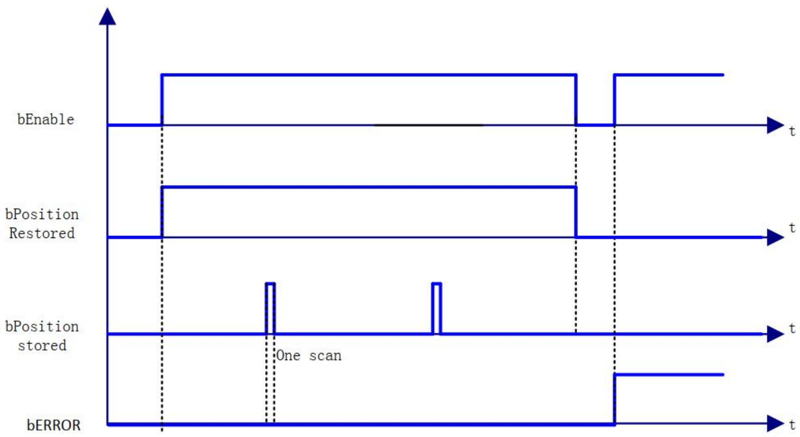
SMC3_PersistPosition_3:SMC3_PersistPositionLogical;

END_VAR

◆ Program section

```
//绝对值位置保存
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

◆ Time-series diagram



4) Error description

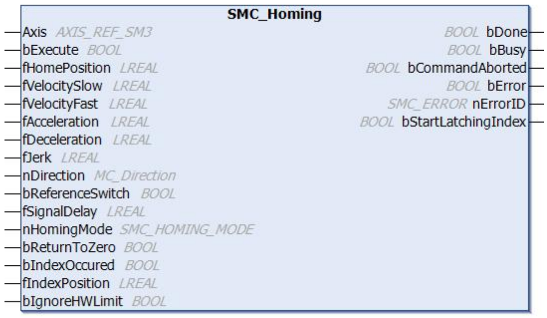
An input axis that is virtual or real will result in an error output; an error in an axis will result in an error output.

[Note]: Please read "Appendix C Error Code Descriptions" for error code descriptions.

7.3.33 SMC_Homing

The axis return to zero command differs from MC_Homing, which sets the return to zero method at the axis configuration, in that this command is a controller-controlled return to zero method.

1) Command format

Instructions	Name	Graphical performance	ST performance
SMC_Homing	Axis back to zero		SMC_Homing0(Axis:= , bExecute:= , fHomePosition:= , fVelocitySlow:= , fVelocityFast:= , fAcceleration:= , fDeceleration:= , fJerk:= , nDirection:= , bReferenceSwitch:= , fSignalDelay:= , nHomingMode:= , bReturnToZero:= , bIndexOccured:= , fIndexPosition:= , bIgnoreHWLimit:= , bDone=> , bBusy=> , bCommandAborted=> , bError=> , nErrorID=> , bStartLatchingIndex=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	—	—	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Execution	BOOL	TRUE, FALSE	FALSE	True function block execution, false no function block execution
fHomePosition	Home set position	LREAL		0	Home set position after return to zero, in user calibrated units
fVelocitySlow	Slow	LREAL		0	Slow setting speed after leaving the reference switch
fVelocityFast	Fast	LREAL		0	Fast setting speed when leaving the

					reference switch set
fAcceleration	Acceleration	LREAL		0	Acceleration setpoint
fDeceleration	Deceleration	LREAL		0	Deceleration setting value
fJerk	Acceleration derivative	LREAL		0	Jerk in [u/s3]
nDirection	Return to zero direction	MC_ DIRECTION		negative	Direction of start of return to zero, reference MC_DIRECTION
bReference Switch	Reference switch	BOOL	TRUE, FALSE	FALSE	Reference switch connected, TRUE: reference switch triggered, FALSE: reference switch closed
fSignalDelay	Delay	LREAL		0	Transfer time of the reference switch to compensate for the dead time. The unit is seconds.
nHomingMod	Return to zero mode	SMC_HOMING_ MODE			Reference SMC_HOMING_MODE
bReturnTozero	Return to zero	BOOL	TRUE, FALSE	FALSE	TRUE: the axis runs to position zero when the return to zero is complete (note: if fHomePosition=10, the axis position becomes 10 when the return to zero is complete. If fHomePosition=10, then the axis position becomes 10 and bReturnTozero is ture then the axis goes backwards 10 units to position 0 after the return to zero is complete)
bIndexOccured		BOOL	TRUE, FALSE	FALSE	Flag pulse recording, effective when zero return mode is FAST_BSLOW_I_S_STOP, FAST_SLOW_I_S_STOP
fIndexPosition		LREAL		0	Position recorded at the time of the flag pulse
bIgnoreHWLimit	Ignore hard limits	BOOL	TRUE, FALSE	FALSE	TRUE, sets the hardware limit switch enable to false, if the same physical switch is used for the hardware limit switch and the reference switch, then the hardware control will be set to

					false

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone		BOOL	TRUE,FALSE	FALSE	True, return to zero complete
bBusy		BOOL	TRUE,FALSE	FALSE	True, the function block is in effect
bCommand Aborted		BOOL	TRUE,FALSE	FALSE	True, the block is interrupted by another action instruction
Error		BOOL	TRUE,FALSE	FALSE	True, an error has occurred
ErrorID		SMC_ERROR		0	Error code, enumerated variable, see help smc_error for specific alarm code
bStartLatching Index		BOOL	TRUE,FALSE	FALSE	Generated by "bIndexOccured" and "fIndexPosition" together

Back to zero mode (SMC_HOMING_MODE)

Enumeration name	Type	Initial value	Description
FAST_BSLOW _S_STOP	SMC_HOMING _MODE	0	Walk towards the home switch at a fast speed in the set direction, hit the home switch and leave the home switch at a slow speed in the reverse direction, after leaving, first execute MC_setPosition to set the current position to the fHomePosition set value, then execute MC_stop
FAST_BSLOW _STOP_S	SMC_HOMING _MOD	1	Walk towards the home switch in the set direction with a fast speed, hit the home switch and leave the home switch in the reverse direction with a slow speed, after leaving, first execute MC_stop to stop the axis and then execute MC_setPosition to set the current position to the fHomePosition set value
FAST_BSLOW_I _S_STOP	SMC_HOMING _MOD	2	The axis moves rapidly in the set direction towards the home switch and then leaves the home switch at a slow speed after hitting the home switch. bIndexOccured signal is executed when MC_setPosition is reached and then MC_stop is executed.
FAST_SLOW _S_STOP	SMC_HOMING_ MOD	4	The MC_setPosition is executed first when the set direction is fast towards the home switch, and then when the home switch is touched, it leaves the home switch at a slow speed.
FAST_SLOW _STOP_S	SMC_HOMING_ MOD	5	Walk towards the home switch in the set direction and leave the home switch at a slow speed after hitting the home switch, then execute MC_stop and then MC_setPosition to set the current position to the fHomePosition setting.
FAST_SLOW_I _S_STOP	SMC_HOMING_ MOD	6	The current position is set to the value set by fHomePosition. The home switch is approached in a fast direction, touched and left at a slow speed, and the bIndexOccured signal is

			followed by MC_setPosition and then MC_stop.

3) Function description

After SMC_HOMING has been started by the rising edge of bExecute, the axis will start moving at speed fVelocityFast and in the direction defined by nDirection until bReferenceSwitch = FALSE. The axis will then stop slowly and leave the reference switch in the opposite direction at speed fVelocitySlow leaves the reference switch in the opposite direction. After bReferenceSwitch = TRUE the return to zero is complete. After enabling the return to zero command the state of bReferenceSwitch is ON->OFF->ON and the return to zero is complete on the rising edge of OFF->ON, setting the reference position.

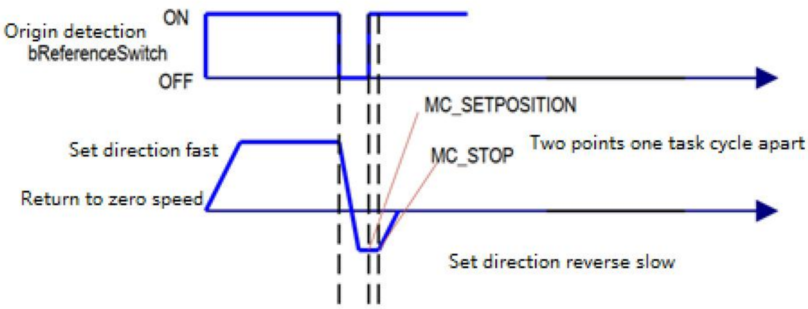
Reference position = fHomePosition + ((fSignalDelay*1000+1 DC clock cycle) /1000) * fVelocitySlow actually compensates for the set bReferenceSwitch sampling delay and one communication cycle shift delay.

If bReturnToZero=TRUE, the state of bReferenceSwitch sets the reference position on the rising edge of OFF->ON to fHomePosition+((fSignalDelay*1000+1 DC clock period)/1000) *fVelocitySlow, and then runs at the speed

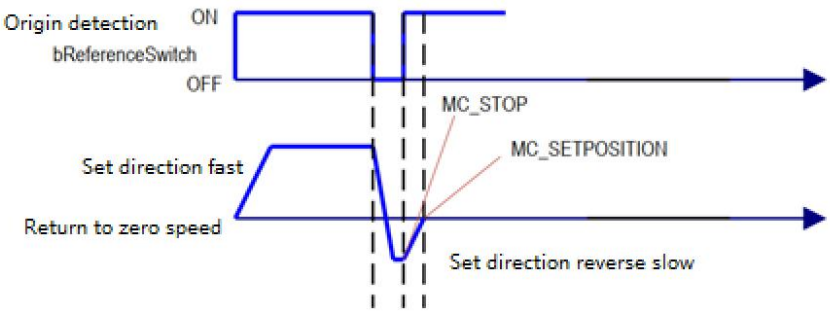
fVelocityFast runs to position 0.

Note: After the Done signal, the axis position is set to: fHomePosition. the timing of the setting is related to the nHomingMode (see SMC_HOMING_MODE for details). The following diagrams show several modes of return to zero:

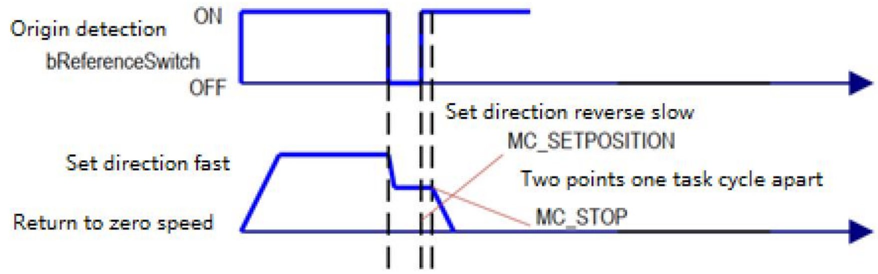
① When returning to zero mode "0"



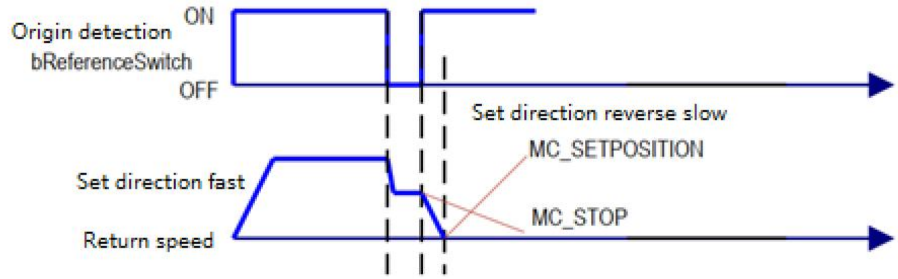
② When returning to zero mode "1"



③ When returning to zero mode "4"

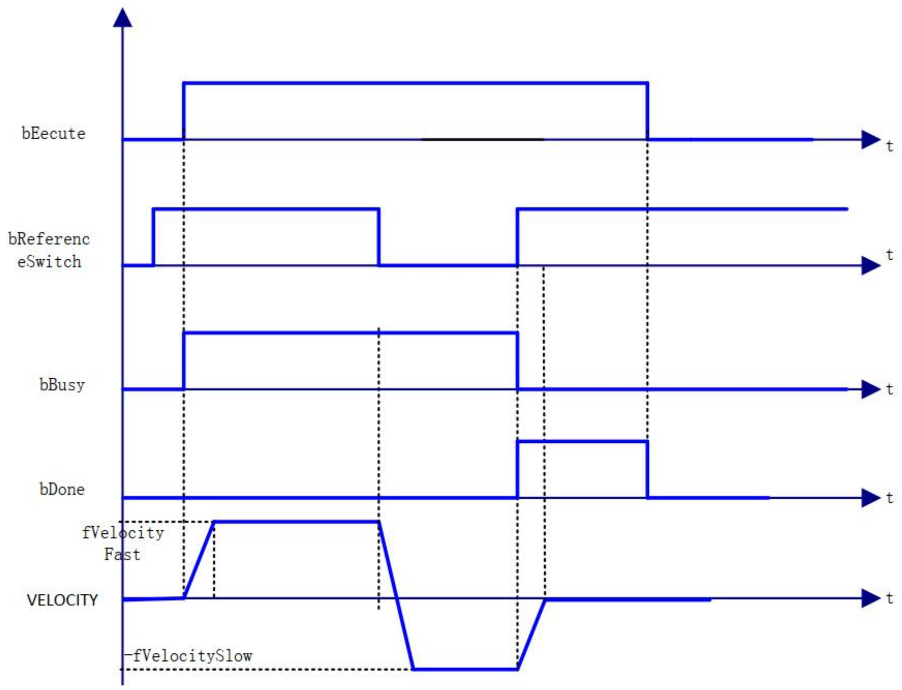


④ When returning to zero mode "5"

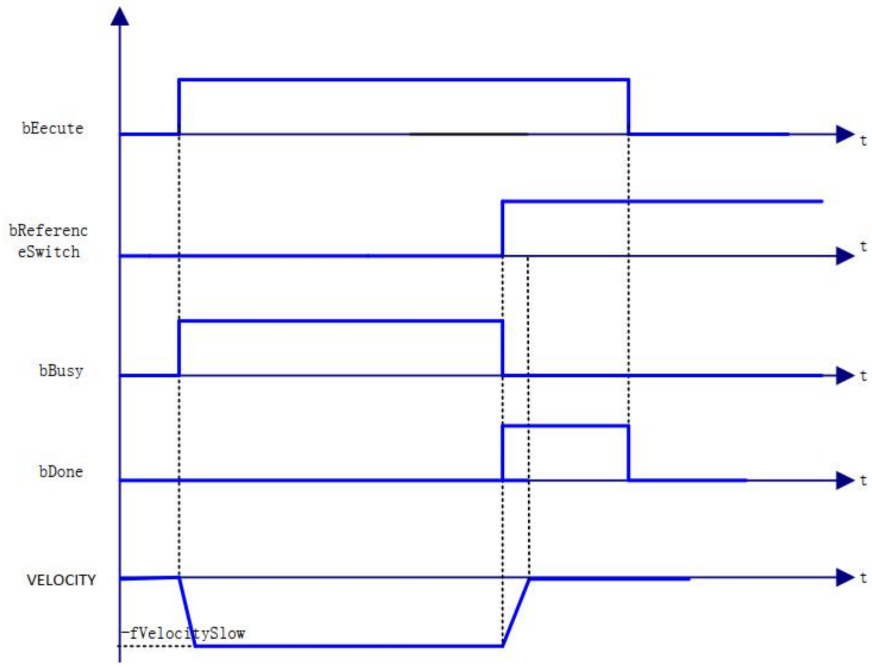


- Timing diagram

① Instruction execution when bReferenceSwitch TRUE



② When the instruction is executed bReferenceSwitch FALSE



4) Error description

Error in input axis type.

Axis has error.

Axis is not enabled

The speed or acceleration is invalid.

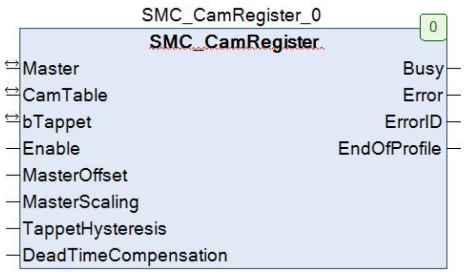
[Note]: Please read "Appendix C Error Code Descriptions" for the descriptions of the relevant error codes.

7.4 Axis group instructions (primary/from-axis instructions).

7.4.1 SMC_CamRegister

Enable cam lift control (cam switch). Cam editing can not edit the main shaft curve, simply configure the tap bar table can be through this function block to achieve the tap control.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ CamRegister	Cam lift bar control		SMC_CamRegister0(Master:= , CamTable:= , bTappet:= , Enable:= , MasterOffset:=0 , MasterScaling:= 1, TappetHysteresis:= , DeadTimeCompensation:= , Busy=> , Error=> , ErrorID=> , EndOfProfile=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
CamTable	Cam table	MC_CAM_REF			Maps to an electronic cam, an instance of an electronic cam
bTappet	Stick output	ARRAY [1..MAX_NUM_TAPPETS] OF BOOL			The output of the lift point

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	Perform	BOOL	TRUE,FALSE	FALSE	True function blocks are executed, false does not perform function blocks
Masteroffset	Spindle offset	LREAL		0	Spindle offset

MasterScaling	Spindle ruler	LREAL		1	The spindle linear scaling factor
TappetHysteresis	Stick damping	LREAL		0	The lift bar controls the damping coefficient
DeadTime Compensation	Dead zone time compensation	LREAL		0	The dead-zone compensation time unit is S, which can be positive or negative depending on the spindle's current velocity linear compensation stick output

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Busy	In action	BOOL	TRUE,FALSE	FALSE	TRUE, function block in action
Error	Error	BOOL	TRUE,FALSE	FALSE	TRUE, an exception occurs
ErrorID	The error code	SMC_ ERROR		SMC_NO_ERROR	When an exception occurs, the error code is output
EndofProfile	Curve Cycle Completes	BOOL	TRUE,FALSE	FALSE	True, the spindle position is greater than or equal to the set period

3) Function description

◆ The Enable signal is TRUE, if there is no error output then the Busy output is TRUE and tappet control is performed.

This control function block is not related to the slave axis in the electronic cam, only the spindle cycle and tappet table need to be configured.

◆ "bTappet" is a one-dimensional boolean structure (MAX_NUM_TAPPETS=512) and bTappet[i] corresponds to the output of the i-th tappet point.

tappet[i] corresponds to the output of the i-th tappet point.

The unit of DeadTimeCompensation is S/sec. If set to a positive value, the tappet signal is overrun, if set to a negative value, the tappet signal is lagged. For example, if the setting is 0.02 seconds and the Ethcat task cycle is set to 4ms, then the tappet output position is P according to the linear speed of the spindle v. The tappet will be output at the spindle set position = P-V*0.02. Conversely, if the setting is -0.02 seconds, the tappet signal is output with a lag of five cycles after the spindle set position is greater than or equal to P.

Example of the use of this function block.

Variable declaration:

```
VAR
    TPP:ARRAY[1..MAX_NUM_TAPPETS] OF BOOL;
    SMC_CamRegister0: SMC_CamRegister;
END_VAR
```

Procedure section:

```
SMC_CamRegister0(
    Master:=Virtual_X ,
```

```
CamTable:=Cam,
bTappet:=TPP ,
Enable:=TRUE ,
MasterOffset:=0 ,
MasterScaling:= 1,
TappetHysteresis:= 0,
DeadTimeCompensation:=0 ,
Busy=> ,
Error=> ,
ErrorID=> ,
EndOfProfile=> );
```

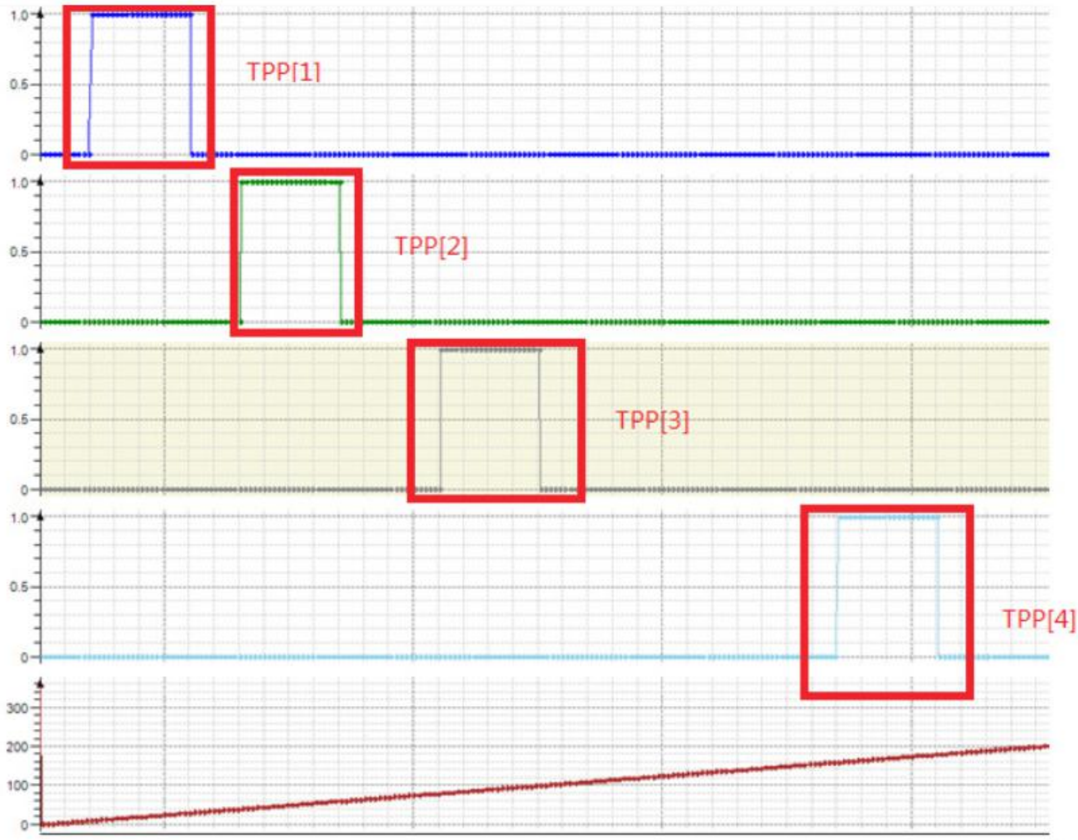
Cam edited the following image：

Cam | Cam table | Tappets | Tappet table

	Track ID	X	positive pass	negative pass
	1			
		10	switch OFF	switch OFF
		30	switch ON	switch OFF
	2			
		40	switch OFF	switch OFF
		60	switch ON	switch OFF
	3			
		80	switch OFF	switch OFF
		100	switch ON	switch OFF
	4			
		160	switch OFF	switch OFF
		180	switch ON	switch OFF

Start Virtual_X-axis:

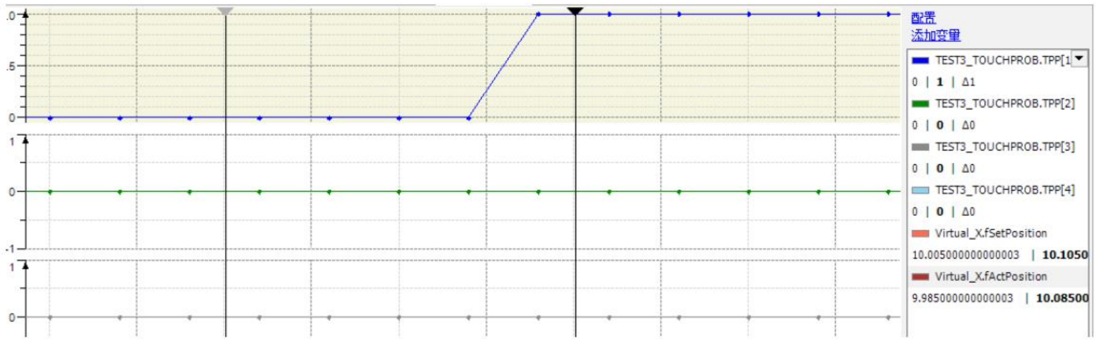
The monitoring curve is shown below：



When the deadband compensation time is set to -0.02 seconds

```
SMC_CamRegister0(
    Master:=Virtual_X ,
    CamTable:=Cam,
    bTappet:=TPP ,
    Enable:=TRUE ,
    MasterOffset:=0 ,
    MasterScaling:= 1,
    TappetHysteresis:= 0,
    DeadTimeCompensation:=-0.02 ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    EndOfProfile=> );
```

The tappet output lags by five task cycles (4ms task cycle) as shown in the diagram below:



4) Error description

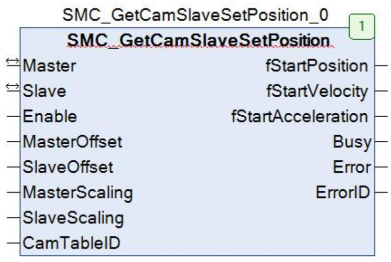
There is an error in the axis, the axis is not enabled, or the offset or scale value is set outside the spindle range.

[Note]: Please read "Appendix C Error Code Descriptions" for descriptions of the relevant error codes.。

7.4.2 SMC_GetCamSlaveSetPosition

Reads cam gauge slave position, speed and acceleration information.

1) Command format

Instructions	Name	Graphical representation	ST Performance
SMC_GetCamSlaveSetPosition	Get cam slave position		SMC_GetCamSlaveSetPosition0(Master:= , Slave:= , Enable:= , MasterOffset:= , SlaveOffset:= , MasterScaling:= , SlaveScaling:= , CamTableID:= fStartPosition=> , fStartVelocity=> , fStartAcceleration=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Map to an axis
Slave	From the axis	AXIS_REF	-	-	Map to an axis

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	Perform	BOOL	TRUE,FALSE	FALSE	True function blocks are executed, false does not perform function blocks
Masteroffset	Spindle offset	LREAL		0	Cam table spindle offset
Slaveoffset	Offset from the axis	LREAL		0	The cam table is offset from the axis
MasterScaling	Spindle zoom	LREAL		1	Cam table spindle scaling factor
SlaveScaling	Zoom from the axis	LREAL		1	The cam table scales the factor from the axis
CamTableID	Cam ID	MC_CAM_ID			Cam meter ID

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
fStartPosition	From the axis position	LREAL		0	The position of the axle obtained based on the cam table and the current spindle information
fStartVelocity	Speed from the axis	LREAL		0	The axle speed obtained based on the cam meter and the current spindle information
fStart Acceleration	Acceleration from the shaft	LREAL		0	The axle acceleration obtained from the cam meter and the current spindle information
busy	In action	BOOL	TRUE, FALSE	FALSE	TRUE, which indicates that the function block is executing
Error	Error	BOOL	TRUE, FALSE	FALSE	TRUE, an exception occurs
ErrorID	The error code	SMC_ERROR		SMC_NO_ERROR	When an exception occurs, the error code is output

3) Function description

The output value calculated by this instruction is: Y = (cam((cam start spindle table position + Masteroffset)* MasterScaling) + slaveoffset)* SlaveScaling, the

Cam is a cam table function. Example: cam start spindle position is 0, master and slave scaling is 1, masteroffset is

100 and slaveoffset is 0, the output of the function block is the slave position corresponding to the cam table at 100.

The function block reads the slave position only if the cam table is built successfully, there is no requirement for the master and slave axes to be running, for example.

Statement:

```
SMC_GetCamSlaveSetPosition0: SMC_GetCamSlaveSetPosition;
ENABLE: BOOL;
    MC_CamTableSelect0: MC_CamTableSelect;
```

Program:

```
MC_CamTableSelect0(
    Master:=Virtual_X ,
    Slave:=Virtual_Y ,
    CamTable:=Cam ,
    Execute:= ,
    Periodic:=TRUE ,
    MasterAbsolute:=0 ,
    SlaveAbsolute:=0 ,
    Done=> ,
    Busy=> ,
    Error=> ,
```

```
    ErrorID=> ,
    CamTableID=> );
SMC_GetCamSlaveSetPosition0(
        Master:= Virtual_X,
        Slave:= Virtual_Y,
        Enable:=ENABLE ,
        MasterOffset:= 100,
        SlaveOffset:=0 ,
        MasterScaling:=1 ,
        SlaveScaling:= 1,
        CamTableID:=MC_CamTableSelect0.CamTableID,
        fStartPosition=> ,
        fStartVelocity=> ,
        fStartAcceleration=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );
```

Enable	BOOL	TRUE
MasterOffset	LREAL	100
SlaveOffset	LREAL	0
MasterScaling	LREAL	1
SlaveScaling	LREAL	1
CamTableID	MC_CAM_ID	
fStartPosition	LREAL	33.580246913580254

4) Error Description

Error output is True, the instruction error is output.

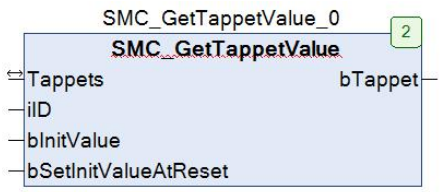
Refer to ErrorID,SMC_ERROR to determine the cause of the error.

[Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.。

7.4.3 SMC_GetTappetValue

Use MC_CamIn command to get the current stick output value.

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
SMC_ GetTappetValue	Gets the stick output value		SMC_GetTappetValue0(Tappets:= iID:=, bInitValue:= , bSetInitValueAtReset:= , bTappet=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Tappets	Stick	SMC_TappetData	-	-	Map to a stick

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
iID	Stick group number	INT		0	The group ID of the stick
bInitValue	The initial value	BOOL			The lift bar initializes the value on the first call of the function block
bSetInitValue AtReset	BOOL				TRUE, MC_CamIn the bar output value will be initialized to the bInitValue set value FALSE when the function block restarts, and the lift bar output value will remain when the MC_CamIn function block restarts.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bTappet	Stick output	BOOL		FALSE	Lifting bar value

3) Function Description

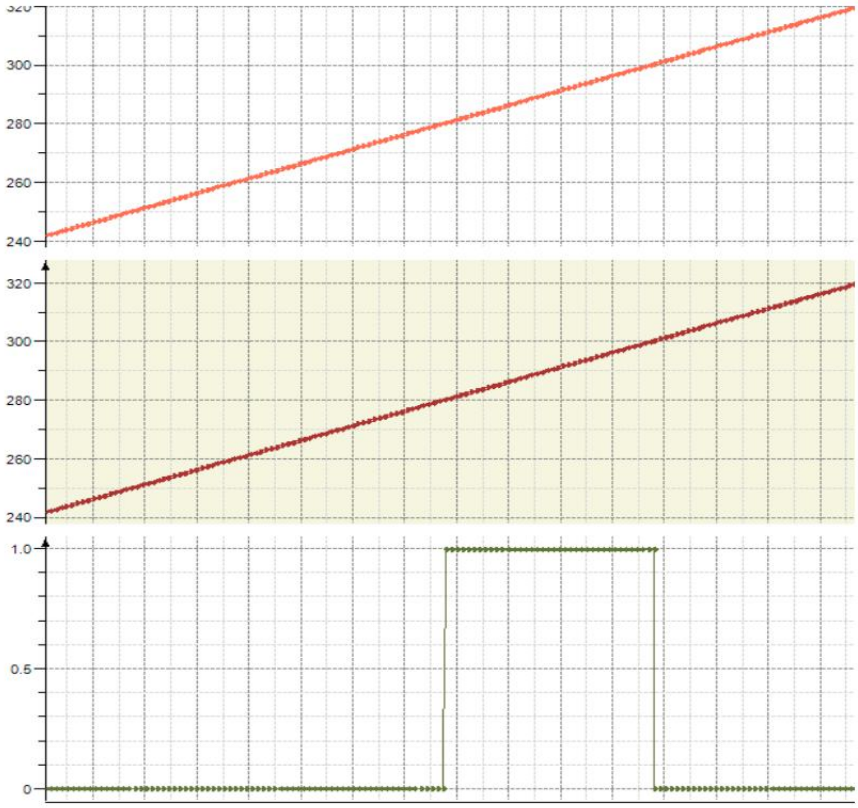
◆ This function block needs to be used in conjunction with the MC_CamIn command.

◆ This function block reads the tappet output as well as the SMC_CamRegister function, but there is a conflict between the two, so that the tappet output is read in the same cam. tappet table

The function block is used in conjunction with the MC_CamIn command.

Example of use：

```
MC_CamIn0(
    Master:=Virtual_X ,
    Slave:= Virtual_Y,
    Execute:= ,
    MasterOffset:= 0,
    SlaveOffset:= 0,
    MasterScaling:=1 ,
    SlaveScaling:= 1,
    StartMode:= 1,
   CamTableID:= MC_CamTableSelect0.CamTableID,
   VelocityDiff:= ,
   Acceleration:= ,
   Deceleration:= ,
   Jerk:= ,
   TappetHysteresis:= ,
   InSync=> ,
   Busy=> ,
   CommandAborted=> ,
   Error=> ,
   ErrorID=> ,
   EndOfProfile=> ,
   Tappets=> );
SMC_GetTappetValue0(
    Tappets:= MC_CamIn0.Tappets,
    iID:=2,
    bInitValue:= false,
    bSetInitValueAtReset:=true ,
    bTappet=> );
```



4) Error description

Axis has error ;

Axis not enabled ;

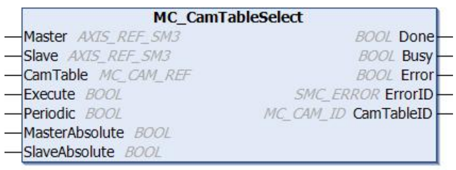
CamTable ID does not point.

[Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.

7.4.4 MC_CamTableSelect

The MC_CamTableSelect function block, which specifies the cam table, is used in conjunction with the MC_CamIn instruction. This function block is used to correlate the relationship between the master, slave and cam table and to set the period of cam operation, the position mode of the master and slave (absolute or relative position), etc. It is a managed instruction, i.e. after triggering the instruction and executing it only once, the relevant master and slave axes can continue to operate according to this characteristic; if the cam table needs to be changed or the master and slave axes need to be changed, the execution of this function block needs to be triggered again If the cam table needs to be changed or the master and slave axis needs to be changed, the function block needs to be triggered again.

1) Command format

Instructions	Name	Graphical representation	ST Performance
MC_ CamTableSelect	Cam table designation		MC_CamTableSelect0(Master:= , Slave:= , CamTable:=), Execute:= , Periodic:= , MasterAbsolute:= , SlaveAbsolute:= , Done=> , Busy=> , Error=> , ErrorID=> , CamTableID=>);

2) Related variables

Input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Main shaft	AXIS_REF	-	-	Mapping to a master axis, i.e. an instance of AXIS_REF_SM3
Slave	Slave shafts	AXIS_REF	-	-	Mapping to a slave axis, i.e. an instance of AXIS_REF_SM3
CamTable	Selection table	MC_CAM_REF	-	-	Mapping to a CAM table description, i.e. an instance of MC_CAM_REF

Notes on use.

The master and slave axes must not be specified as the same axis, otherwise an error will be output. The cam table corresponding to the CamTable must be edited correctly, otherwise it will also cause an error to be reported in the command. The master and slave axes can be real or imaginary axes.

Input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe

Execute	Execution	BOOL	TRUE,FALSE	FALSE	Rising edge signal, execute command
Periodic	Repeat mode	BOOL	TRUE,FALSE	FALSE	Specifies whether the specified cam table is to be executed repeatedly or only once TRUE: Repeat False: not repeated
MasterAbsolute	Spindle absolute	BOOL	TRUE,FALSE	FALSE	Specify whether the spindle tracking distance coordinate system is based on absolute or relative position 1: Absolute position, 0: Relative position
SlaveAbsolute	Mode	BOOL	TRUE,FALSE	FALSE	In combination with the StartMode in the MC_CamIn instruction, this specifies whether the current command position of the slave axis is absolute (the current spindle position corresponds to the cam table output) or relative (the cam table output value is superimposed on the slave axis position at the start of the command). The current command position is either absolute (the cam table output corresponding to the current spindle position) or relative (the cam table output value superimposed on the slave position at the start of the command) 1: absolute position, 0: relative position

Precautions for use:

Improper selection of MasterAbsolute and SlaveAbsolute may cause the electronic cam output to jump, so please make sure to set the cam curve working method before setting.

Output variables

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	Complete	BOOL	TRUE,FALSE	FALSE	TRUE when completion is selected,
Busy	Execution in	BOOL	TRUE,FALSE	FALSE	TRUE if no completion in

	progress				selection
Error	Error	BOOL	TRUE,FALSE	FALSE	TRUE when an exception occurs
ErrorID	Error code	SMC_ERROR	Refer to SMC_ ERROR	0	Output error code when an exception occurs
CamTableID	Effective	MC_CAM_ID	-	-	Cam_ID of the selection, used in conjunction with the CamTableID in the MC_CamIn instruction

Notes on use:

When Error occurs, please check the SMC_ERROR in the help against the ErrorID.

3) Function Description

◆ This instruction specifies the cam table required for electronic cam operation, so the cam table must be edited (by the cam editor or online) before using this instruction.

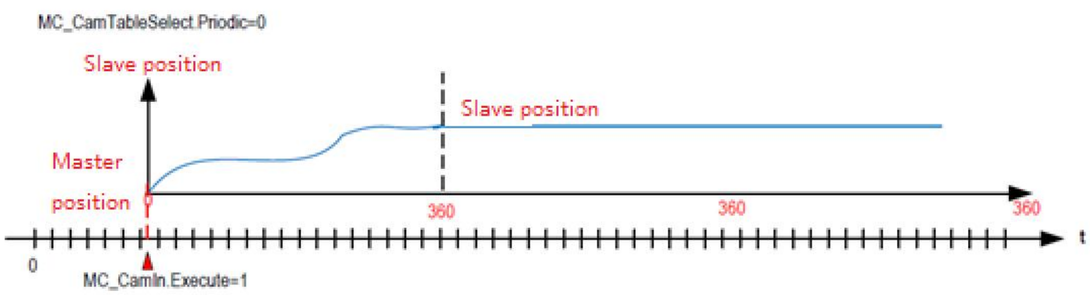
◆ Excute rising edge, execute the specified cam table, or refresh the specified cam table after the cam table is updated.

If the output of Done signal is TRUE, the output variable "CamTableID" is generated and takes effect.

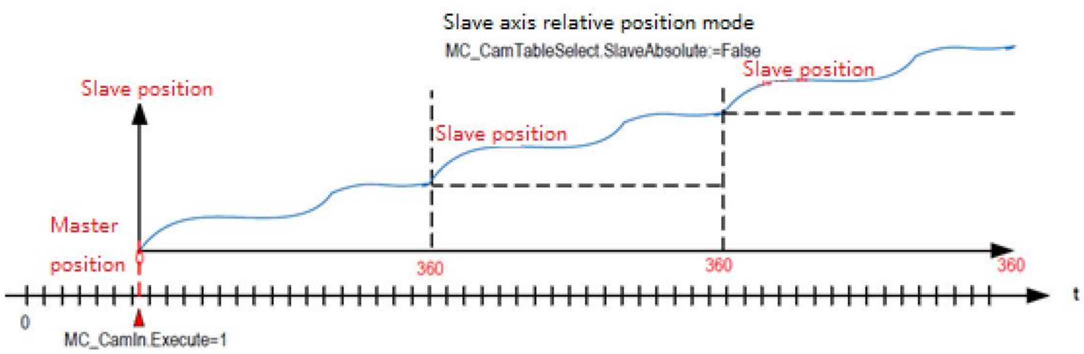
When the Busy signal is TRUE, the Done signal is TRUE and the Busy signal is FALSE.

◆Periodic parameters

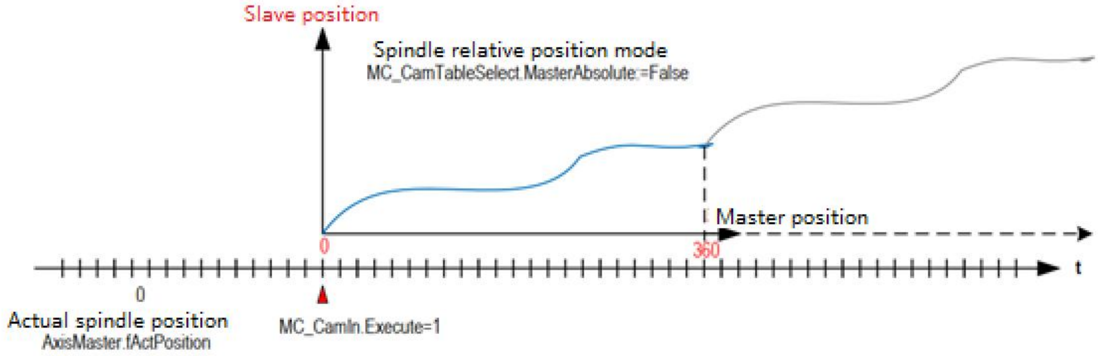
The following figure shows the effect of single-cycle cam operation. When the cam table is selected in single cycle mode (Periodic:=0), the slave axis is released from cam operation after one cam table cycle has been run.



When the cam table is selected in Periodic mode (Periodic:=1), after running one cam table cycle, the slave axis starts the next cam cycle again until a user program commands it to exit the cam running state, as follows:



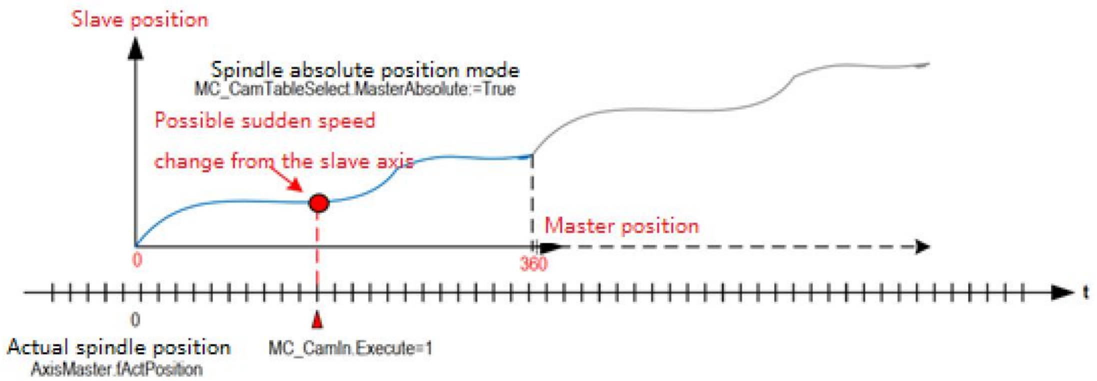
◆Operating characteristics when both spindle and slave axes are in relative position mode



When the master axis is in relative position mode, the cam module will operate with the current position as the starting point X=0 of the master axis when entering CAM.

When the slave axis is in relative position mode, the cam module uses the current position as the starting point Y0 for the slave axis when entering CAM, and the CAM output is superimposed on this thereafter.

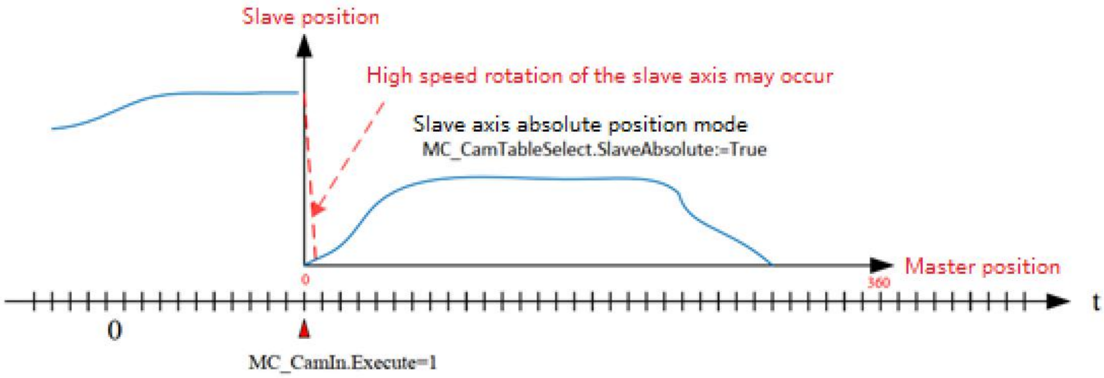
◆ Major axis is in absolute position mode, slave axis is in relative position mode



When the spindle is in absolute position mode, when entering the CAM, the cam operation module obtains the axle position at the current spindle position, so:

- High-speed rotation from shaft position when entering CAM operation, resulting in shock or damage to the equipment;
- If the current position is outside the valid range of the CAM table, the axle does not move and an alarm is issued;
- If the CAM table is in cycle mode, the continuous running of the next CAM cycle begins when the current cycle is completed.

◆Main axis in relative to position mode, slave axis in absolute position mode



When the slave axis is in absolute position mode, it will be adjusted to the position required by CAM when it enters CAM operation, and if the deviation is relatively large, automatic adjustment of the high-speed movement will occur.

Countermeasures according to application characteristics.

For equipment where alignment operation is necessary, such as fixed length cutting rotary knives as cam slave must be in absolute position, programming with attention to the zero position operation of the rotary knives before the first rotary cutting action.

setting the spindle position range of the cam table reasonably to avoid position adjustment of the cam in the opposite direction at the start of the next cycle.

Run SMC_GetCamSlaveSetPosition to set the slave position of the cam entry point to the current coordinates of the slave axis.

For applications where relative position mode can be used, try to use relative position mode as far as possible:.

MC_CamTableSelect.SlaveAbsolute:=False; or set MC_CamIn.StartMode:=1; (relative mode)

Caution.

When the slave axis is set to absolute mode with "limited length", the controller will select a closer direction to return to zero when making zero adjustments, if either left or right rotation is possible. When designing the range of the cam table, particular care should be taken not to allow the range of the cam table to exceed the actual range of operation required, otherwise instantaneous high speed rotation of the servo slave axis may occur, resulting in mechanical shock.

4) Explanation of errors

◆ The master axis and slave axis cannot be specified as the same axis, otherwise there will be an error output.

◆ The CamTable must be edited correctly, otherwise it will be outputted incorrectly.

Note]: Please read "Appendix C Error Code Descriptions" to understand the error code descriptions.

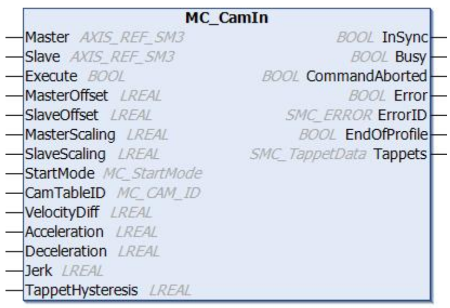
7.4.5 MC_CamIn

It puts the cam slave axis into synchronous operation with the cam spindle and controls the adjustment of the cam slave axis to the corresponding target point according to the current position of the spindle and the position relationship of the cam table; the execution of this command has no effect on the spindle. The master-slave axis offset value, scaling ratio and operating mode can be specified according to the application requirements.

Once MC_CamIn has been triggered, the slave axis follows the position of the spindle according to the position correspondence in the cam table, note that it is a position correspondence and not a speed correspondence.

Once in cam operation, each EtherCAT interrupt parses the CAM cam table, calculates the next target point for the slave axis based on the current position of the spindle and sends the next target position to the slave axis to make it run.

1) Command format

Instructions	Name	Graphical representation	ST Performance
MC_CamIn	Start of cam action		MC_CamIn0(Master:= , Slave:= , Execute:= , MasterOffset:= , SlaveOffset:= , MasterScaling:= , SlaveScaling:= , StartMode:= , CamTableID:= , VelocityDiff:= , Acceleration:= , Deceleration:= , Jerk:= , TappetHysteresis:= , InSync=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , EndOfProfile=> , Tappets=>);

2) Related variables

. . . input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
Slave	From the axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Precautions for use:

The spindle and the from axis cannot be specified as the same axis, otherwise there will be an error output.

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	Perform cam work to enter the energy block	BOOL	TRUE, FALSE	FALSE	Up the edge, perform the electronic cam
MasterOffset	Spindle bias	LREAL	Negative, positive, 0	0	Moves the phase of the spindle with the specified offset value
SlaveOffset	Biased from the axis	LREAL	Negative, positive, 0	0	Moves the phase from the axis with the specified offset value
MasterScaling	Spindle pre-edited shift ratio	LREAL	>0.0	1	Zoom in/out of the phase of the spindle at a specified scale
SlaveScaling	Move the scale from axis pre-editing	LREAL	>0.0	1	Zoom in/out the phase from the axis at a specified scale
StartMode	The output mode from the shaft relative to the cam	MC_ StartMode		absolute	0: absolute absolute position : 1: relative relative position: 2: ramp_in (slope cut). 3: ramp_in_pos (front ramp cut) 4: ramp_in_neg reverse ramp cut in
CamTableID	Table number	MC_CAM_ID			Defines the use of cam tables, MC_CamTableSelect with the output point CamTableID of the computer
VelocityDiff		LREAL			The ramp_in different from the maximum speed
Acceleration		LREAL			ramp_in at the time of the change
Deceleration		LREAL			ramp_in at the time of the change
Jerk		LREAL			ramp_in acceleration of the car
Tapped Hysteresis		LREAL			The damping coefficient of the lift bar

◆ The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe

InSync	The cam is in effect	BOOL	TRUE, FALSE	FALSE	After the spindle and the axle establish cam relations, InSync is positioned and InSync is reset when the execution condition of the instruction is OFF.
Busy	Running synchronously	BOOL	TRUE, FALSE	FALSE	When Execute enters the rising edge, position TRUE, true indicates that the cam relationship coupling requires a Cam_out command reset, and the instruction execution condition reset cannot reset the state.
Command Aborted	The instruction is interrupted	BOOL	TRUE, FALSE	FALSE	The output from the axis is interrupted by other control instructions as TRUE
Error	Error	BOOL	TRUE, FALSE	FALSE	If an error is detected, the Error bit is set, and when the execution condition of the instruction OFF is OFF, the Error bit is reset.
ErrorID	The error code	SMC_ ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output
EndOf Profile	The curve is complete	BOOL		FALSE	If the periodic parameter is 0 (non-periodic) when the MC_CamTableSelect instruction is executed, the EndOfProfile bit is positioned once the cam curve is executed, and the EndOfProfile bit is reset when the execution condition of the instruction IS OFF.
Tappets	SMC_ TappetData				An associated cam lift bar can be read MC_GetTappetValue command

3) Function description

Execute rising edge, no error is reported on the axis, this instruction is activated if the cam table is selected correctly.

To call a cam curve in a cam system, first call the MC_CamTableSelect instruction to select the corresponding cam table and then execute MC_CamIn; if the cam curve is to be changed, call the MC_CamTableSelect instruction again to reselect the cam table.

The Camout instruction is used to break the cam coupling between the master and slave axes. When this instruction is executed, the cam relationship between the slave axis and the master axis will be released and the CommandAborted output will be TRUE when the slave axis of this instruction executes another motion command.

4) Command details

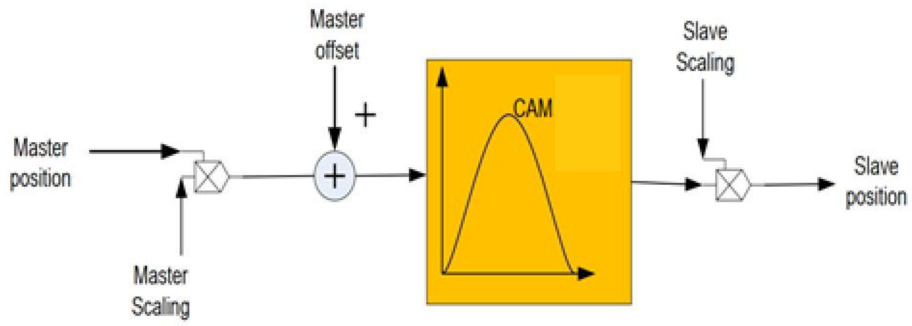
The following is a detailed description of the command:

◆ **Command start conditions**

This command can be activated in any state during spindle stop, position control, speed control or synchronous control

Note: The cam follower position setting must be within the software limit value, otherwise it will result in an incorrect output command.

◆ Calculation of the point of contact in a cam curve



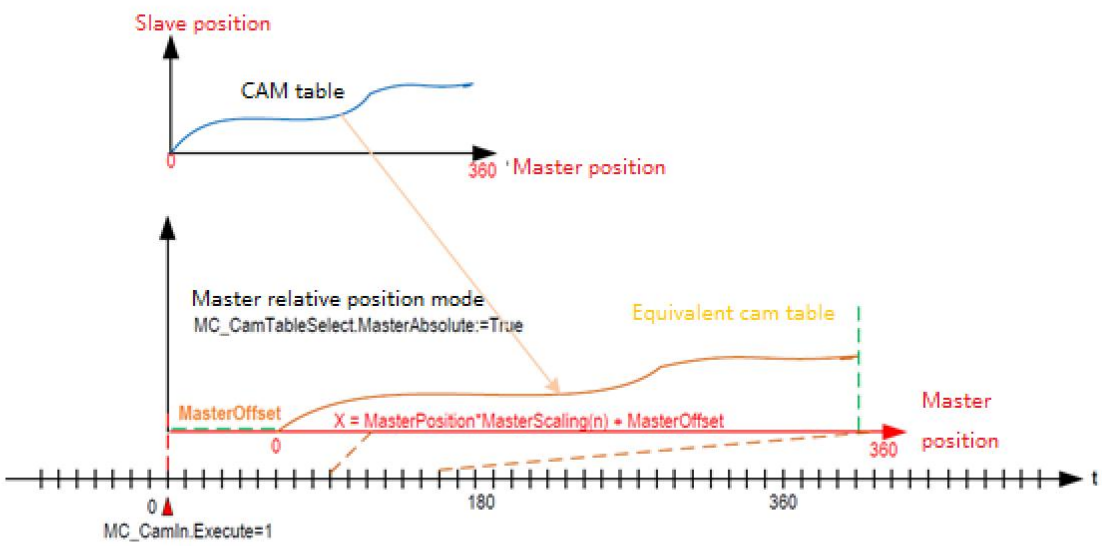
The calculation from the figure above is as follows:

Position_Slave = SlaveScaling*CAM(MasterScaling*MasterPosition + MasterOffset) + SlaveOffset

The spindle position and the axle position in the formula do not represent the position of the actual physical axis, but rather the spindle position associated with the cam function curve. The relationship between the main-axis position and the main-from solid axis position is described in detail.

◆The **cam spindle MasterScaling calculation**

By default, the system is on MasterScaling1,and if the user program modifies the variable:



The proportional SCALE value is set for the cam spindle, and the position of the spindle can be scaled linearly so that its corresponding position relationship with the cam table meets the requirements of the expectation.

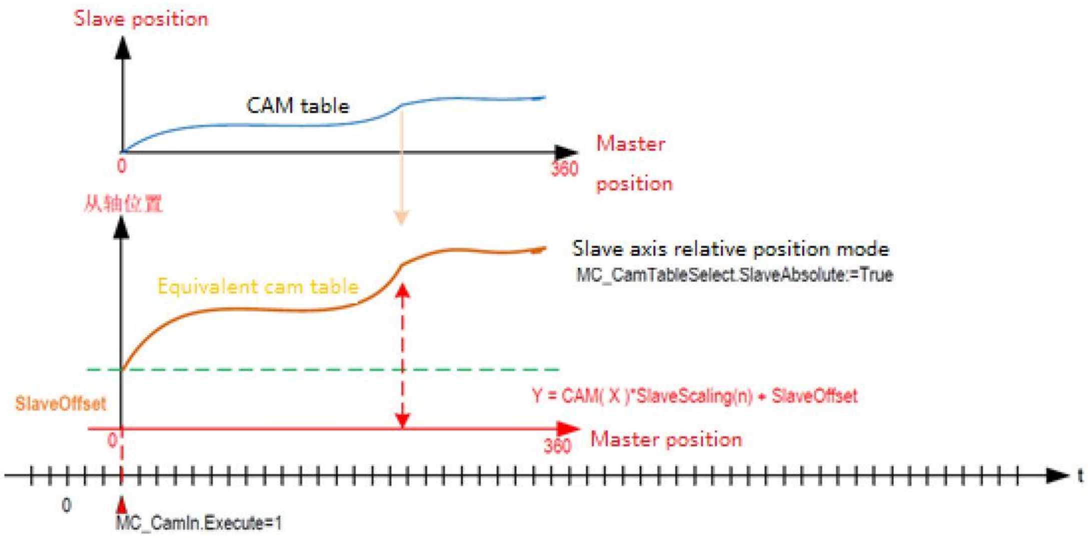
If the offset setting for the spindle is taken into account, the calculated position of the spindle(X)in the cam table will be:

X - MasterPosition-MasterScaling (n) - MasterOffset

This parameter can be used to fine-tune the dimensions of the equipment machining work pieces.

◆The **cam is calculated from** the shaft **SlaveScaling**

By default, the system is on SlaveScaling1,and if the user program modifies the variable:



Set the proportional SlaveScaling value for the cam slave from the shaft to scale linearly from the position of the slave shaft so that the output of the cam control meets the desired position of motion from the shaft.

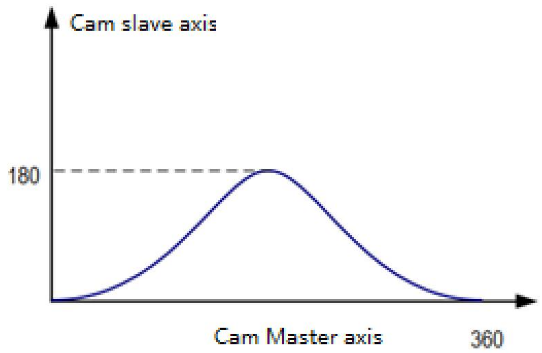
If the offset setting from the axis is taken into account, the output position of the cam from the shaft(Y)is:

$$Y = CAM(X) * SlaveScaling(n) + SlaveOffset$$

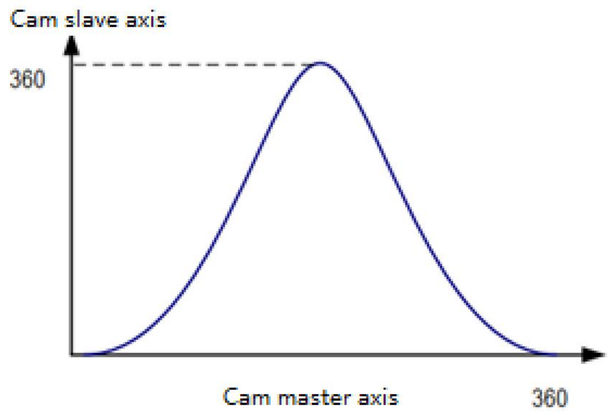
This parameter can be used to fine-tune the dimensions of the equipment machining work pieces.

Examples of usage:

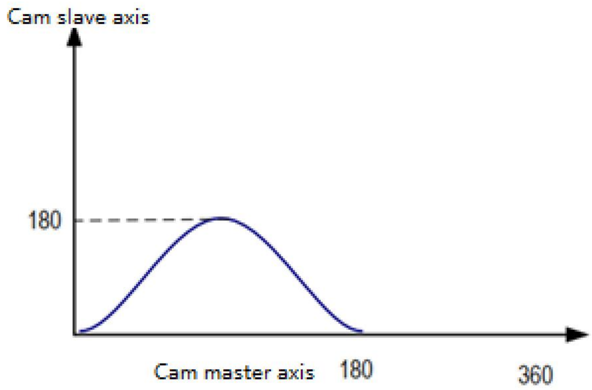
When MasterScaling is 1.0, SlaveScaling is 1.0, MasterOffset is 0, slaveOffset is 0, the cam curve is the planned cam curve as shown in the following illustration:



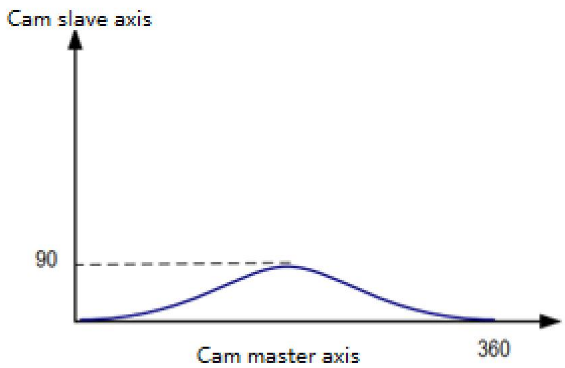
When MasterScaling=1.0, SlaveScaling=2.0, MasterOffset=0, SlaveOffset=0, the cam curve is as shown below:



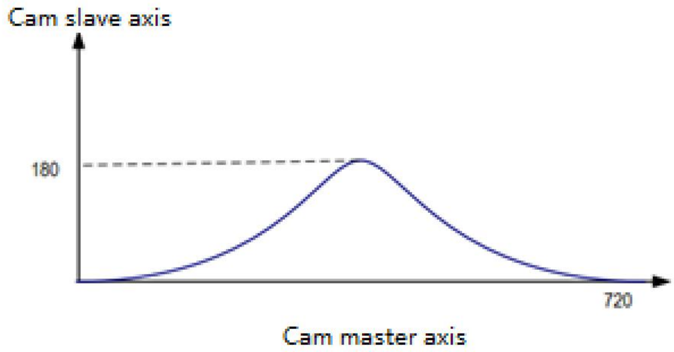
When MasterScaling is 2.0, SlaveScaling is 1.0, MasterOffset is 0, slaveOffset is 0, the cam curve is shown in the following image:



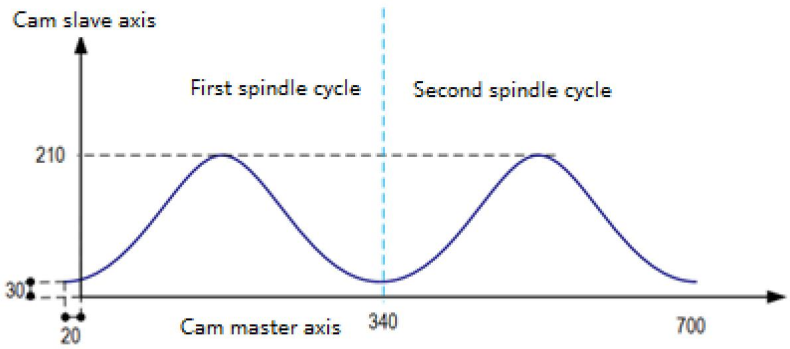
When MasterScaling is 1.0, SlaveScaling is 0.5, MasterOffset is 0, SlaveOffset is 0, the cam curve is shown in the following image:



When MasterScaling is 0.5, SlaveScaling is 1, MasterOffset is 0, slaveOffset is 0, the cam curve is shown in the following image:



When MasterScaling is 1, SlaveScaling is 1, MasterOffset is 20, and Slave Is 30, the cam curve is shown in the following image:



Offset, Scale usage featuresand considerations for cam operation:

(1) Spindle position mode, from the station position mode, in addition to the special requirements of the application system, it is recommended to use relative mode as far as possible, so that simple programming, the possibility of mechanical system impact is relatively small;

(2) Cam meter spindle startand endrange, Offset,Scale and other settings, can make up for the design deviation of the CAM table, it is recommended to refer to the default settings as far as possible, so that easy to debug and maintenance, the chance of running errors can also be reduced;

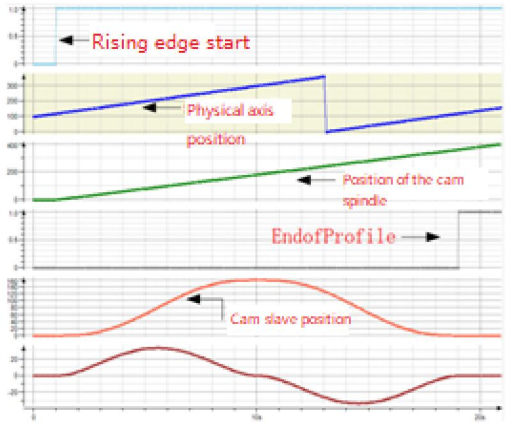
(3) When the CAM cam table cycle is completed /or exited/or switched, the MC_CamIn re-entry is performed again, and the system clears the settings of Offset,Scale, etc. in the memory and reverts to the default values, requiring attention.

◆ Periodic mode in relation to EndOfProfile

Periodic mode Non-periodic mode determines whether the electronic cam is to be performed again after the spindle has reached the end position.

① Non-periodic mode: MC_CamTableSelect instruction Periodic selects False

In non-periodic mode, the cam completes the EndofProfile signal with True, and the EndofProfile output is FALSE if FALSE is entered.



Note: The spindle period refers to the range of the electronic cam spindle position from the start position to the end position.

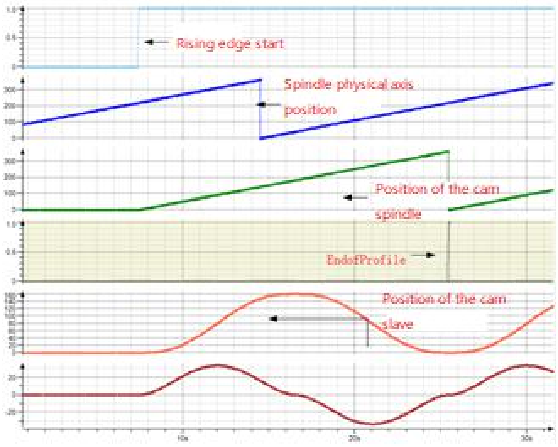
②Periodic mode: MC_CamTableSelect command Periodic select TRUE

In this case the cam completes one spindle cycle and the next cycle is executed, and the EndofProfile signal TRUE is output for only one task cycle.

Caution:

When the cam spindle position is greater than or equal to the cam end position, the EndofProfile signal output is TRUE and the cam spindle position is updated to: cam start position + partly greater than end position.

For example, if the electronic cam spindle start position is 0, the end position is 360, the master and slave axis scaling is set to 1, the master and slave axis offset value is set to 0, the task cycle is 2 ms and the spindle speed is 100, when the cam spindle position is 359.99 in one task cycle, then the EndofProfile output is TRUE in the next cycle and the spindle position becomes The start and end positions of the cam profile in cycle mode should ideally be smooth, otherwise there will be jumps. For example, if the start speed is 0 and the end speed is not 0, this will cause the spindle to jump at the end of the cycle and at the start of a new cycle.。



◆ Relationship between StartMode and the absolute relative mode of the master and slave axes in MC_CamTableSlect

Absolute mode: At the start of a new e-cam cycle, the e-cam is calculated independently of the current slave axis position. If the starting position of the slave axis with respect to the master axis is different from the ending position of the slave axis with respect to the master axis, this will cause a jump.

Relative mode: The new electronic cam changes according to the current slave axis position; i.e. the position of the slave axis at the end of the previous electronic cam cycle is calculated as a "slave axis offset" by the current electronic cam movement. However, if the slave position corresponding to the starting position of the main axis is not 0 in the electronic cam definition, this will cause a jump.

Ramp input: The potential jump at the start of the electronic cam is prevented by adding a compensating motion (motion based on the limit value VelocityDiff, acceleration, deceleration). Thus, as long as the slave axis is in a rotating mode, the forward ramp input option compensates only in the forward direction, while the reverse ramp input compensates only in the reverse direction. For slave axes with linear motion, the direction of compensation can be achieved automatically, i.e. the forward ramp input and the reverse ramp input can be interpreted in the same way as the ramp input.)

The relationship table is shown in the following table：

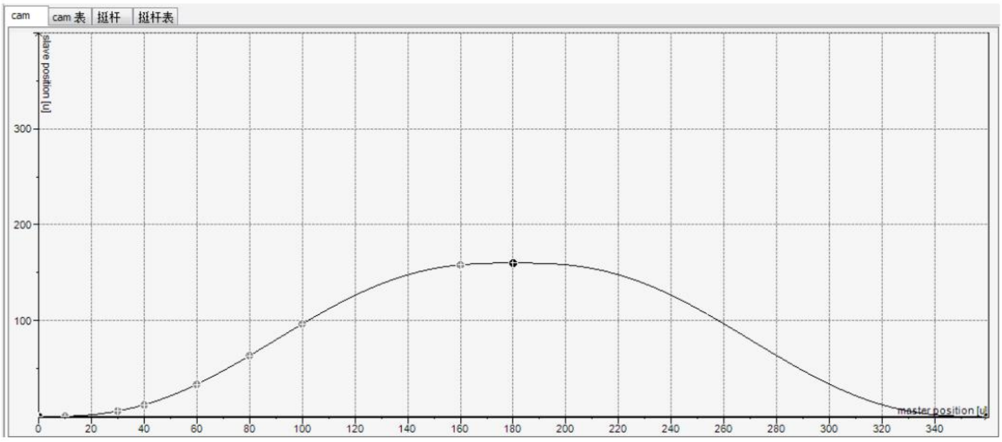
MC_CamTableSelect.MasterAbsolute	Spindle mode
absolute	Absolute mode
relative	Relative mode

MC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	Slave mode
absolute	TRUE	Absolute mode
absolute	FALSE	Relative mode
relative	TRUE	Relative mode
relative	FALSE	Relative mode

ramp_in	TRUE	Slope-cut absolute mode
ramp_in	FALSE	Slope cut relative mode
ramp_in_pos	TRUE	Forward slope cut absolute
ramp_in_pos	FALSE	Forward slope entry relative mode
ramp_in_neg	TRUE	Reverse slope entry absolute
ramp_in_neg	FALSE	Reverse slope entry relative mode

The detailed relationships are described as follows.

Cam master range (0-360), cam slave range (0-180), cycle mode, master-slave offset value 0, master-slave scaling ratio 1. The designed cam table is shown in the following figure:

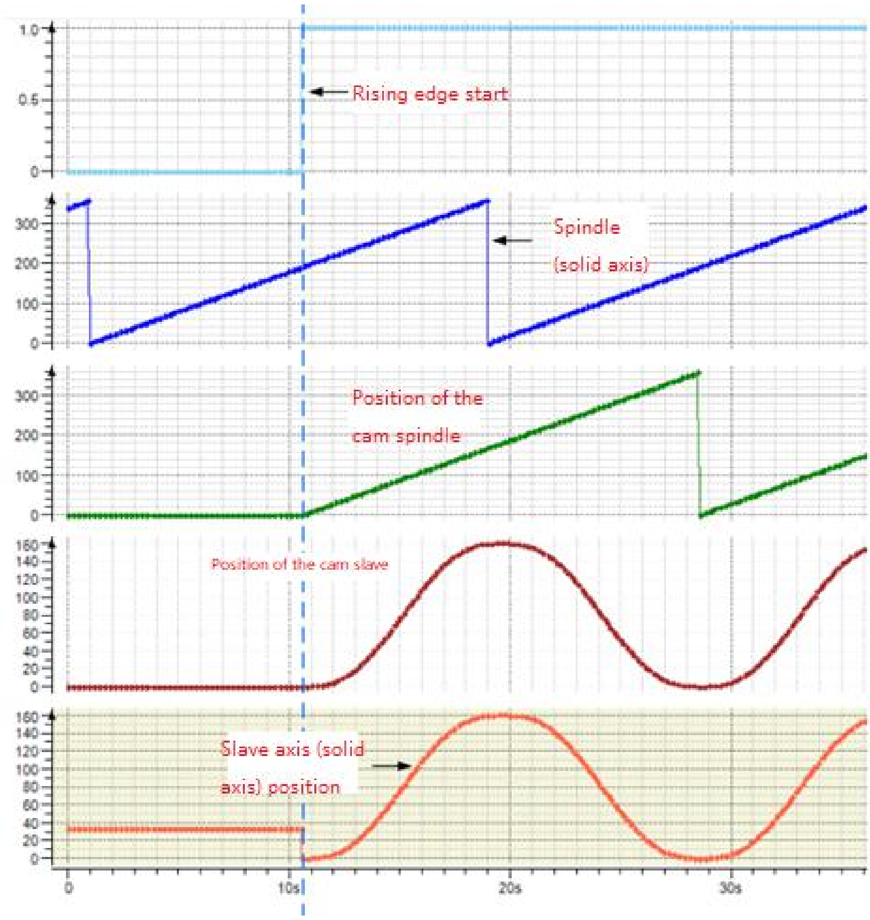


■ StartMode is 0 (absolute mode)

① When the MC_CamTableSlect command MasterAbsolute is set to FALSE and SlaveAbsolute is set to TRUE.

The master axis is in relative mode and the slave axis is in absolute mode. When the cams are activated along the Excute rise, the camshaft starts at the cam table "start position" (0) and the camshaft slave calculates the output according to the "Cam table tooth combination formula" as described above, with the slave real axis command position being equal to the tooth combination calculated output value. If, for example, the camshaft start position is 0 and the camshaft real axis position is 20 at the start of the camshaft, then the start of the camshaft real axis position command is 0, resulting in a jump.

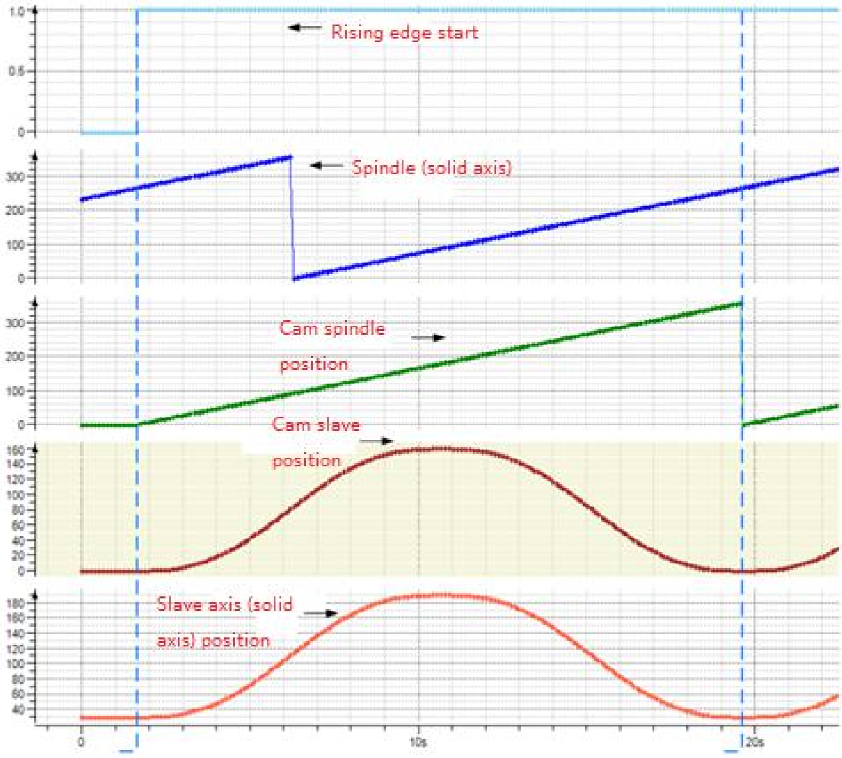
Note: In this case the start position of the slave axis (real axis) is not in the cam slave start position and a jump will occur.



② When the MC_CamTableSlect command MasterAbsolute is set to FALSE and SlaveAbsolute is set to FALSE

The master axis is in relative mode and the slave axis is in relative mode. When the cam is activated along the Excute rise, the camshaft starts at the cam table "start position" (0) and the camslave calculates the output according to the "cam table gearing formula" as described above, with the slave real axis command position equal to the gearing calculated output (camslave position) + start. camshaft position) + camshaft real position at start-up.

If, for example, the cam starts with a solid slave shaft position of 20 and the cam table starts with a slave shaft position of 0, then the cam starts with a solid slave shaft position command of 20, which is followed by 20 + the cam table calculated value, with a peak value of 20 + the maximum cam table calculated value (180 in this case) = 200.



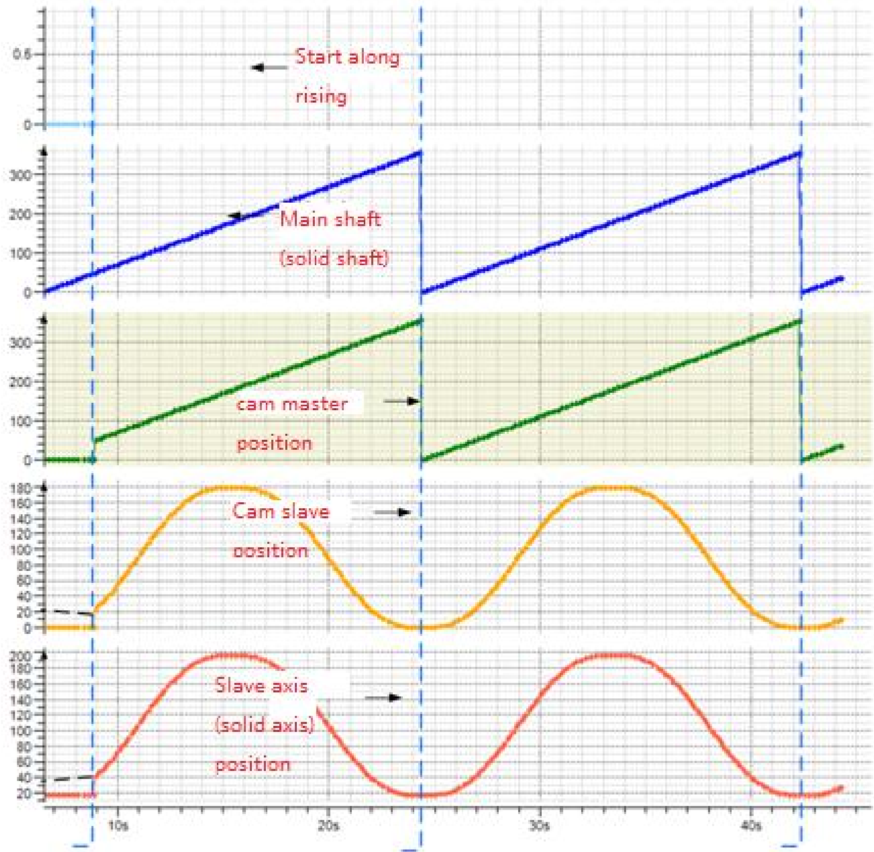
③When the MC_CamTableSlect command MasterAbsolute is set to TRUE and SlaveAbsolute is set to FALSE

The master axis is in absolute mode and the slave axis is in relative mode. When the cam is activated along the Excute rise, the camshaft starts from the current "Master axis real position" and the Slave axis real position command = the calculated value of the cam table tooth fit (cam slave position) + the slave axis position at start-up.

Caution:

1 If the spindle (solid axis) start position is not at the camshaft start position in this case, a jump will occur.

2 The master axis position should be within the camshaft position range.



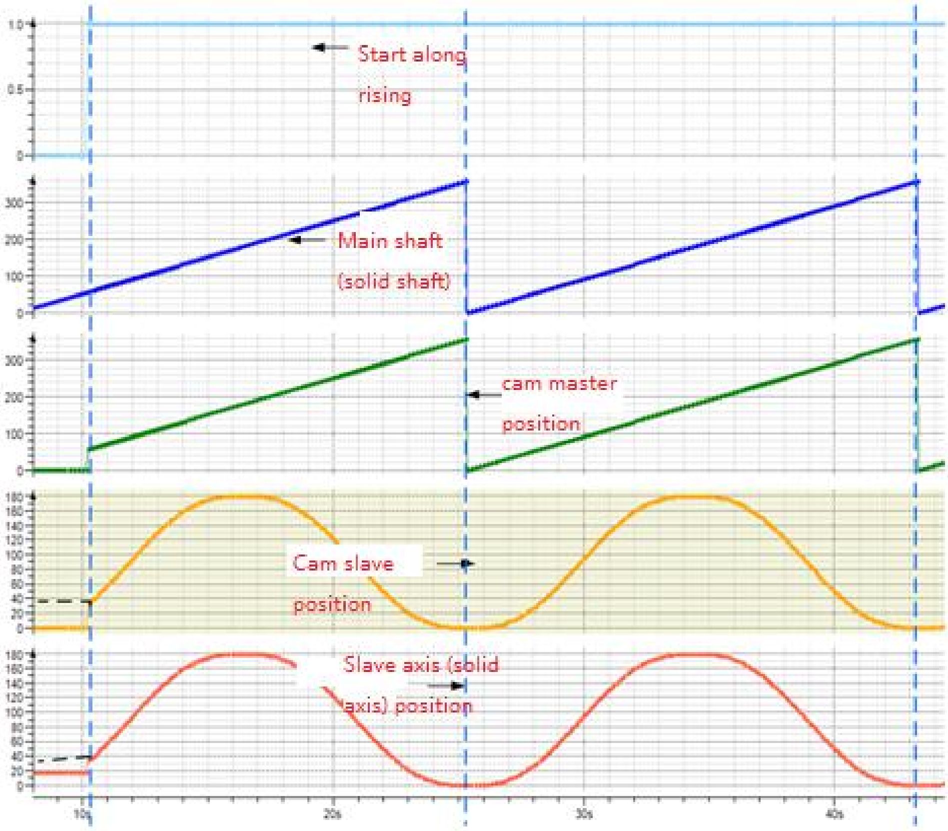
④ When the MC_CamTableSlect command MasterAbsolute is set to TRUE and SlaveAbsolute is set to TRUE

The master axis is in absolute mode and the slave axis is in absolute mode. When the Excute rises, the camshaft starts from the current "MasterAbsolute position" and the SlaveAbsolute command = the calculated value of the cam table tooth fit (cam slave position).

Caution.

1 If the starting position of the main shaft (solid axis) in this case is not at the starting position of the cam main shaft and the slave axis position is not at the starting position of the cam slave axis, a jump will occur.

2 The main shaft position should be within the range of the cam main shaft position.。



■ StartMode is 1 (relative mode)

① When the MC_CamTableSlect command MasterAbsolute is set to FALSE and SlaveAbsolute is set to TRUE or False

The master axis is in relative mode and the slave axis is in relative mode. When Excute rises, the camshaft starts from the "Cam table start position" and the slave real axis position command = Cam table tooth fit calculated value + Cam table tooth fit calculated value (cam slave position).

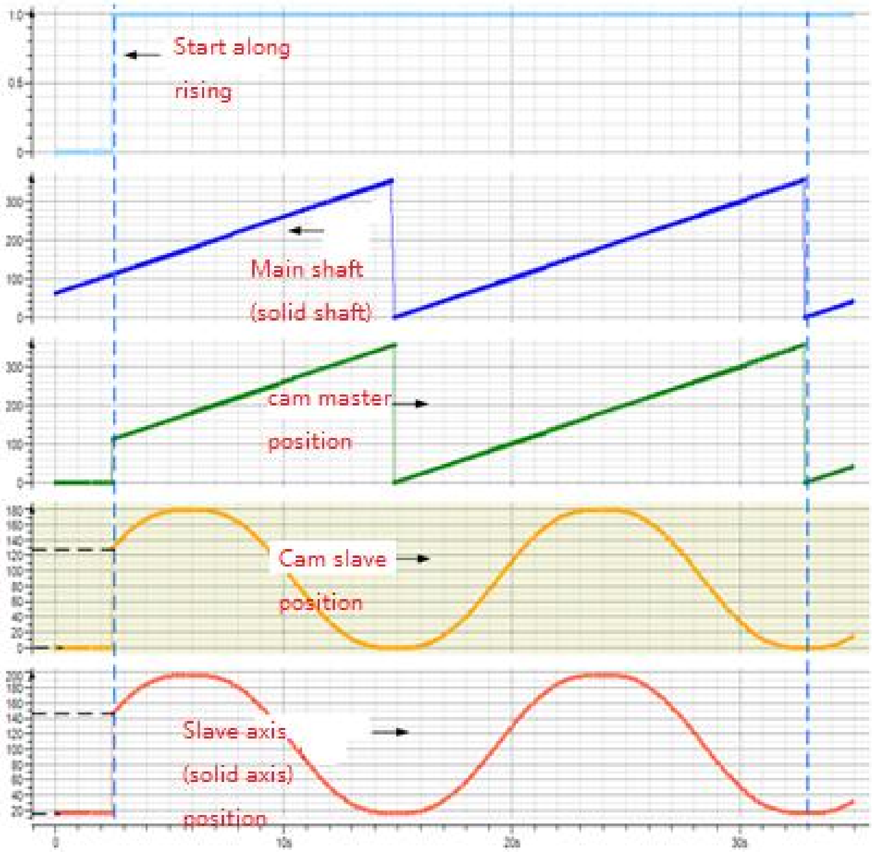
② When the MC_CamTableSlect command MasterAbsolute is set to TRUE and SlaveAbsolute is set to TRUE or False

The master axis is in absolute mode and the slave axis is in relative mode. When the Excute rises, the cam spindle starts from the "current position of the spindle" when the cam is activated and the slave real axis position command = slave position at start + cam table tooth fit calculation (cam slave position).

Caution.

1 If the spindle (solid axis) start position in this case is not at the cam spindle start position then a jump will occur.

2 The master axis position should be within the camshaft position range

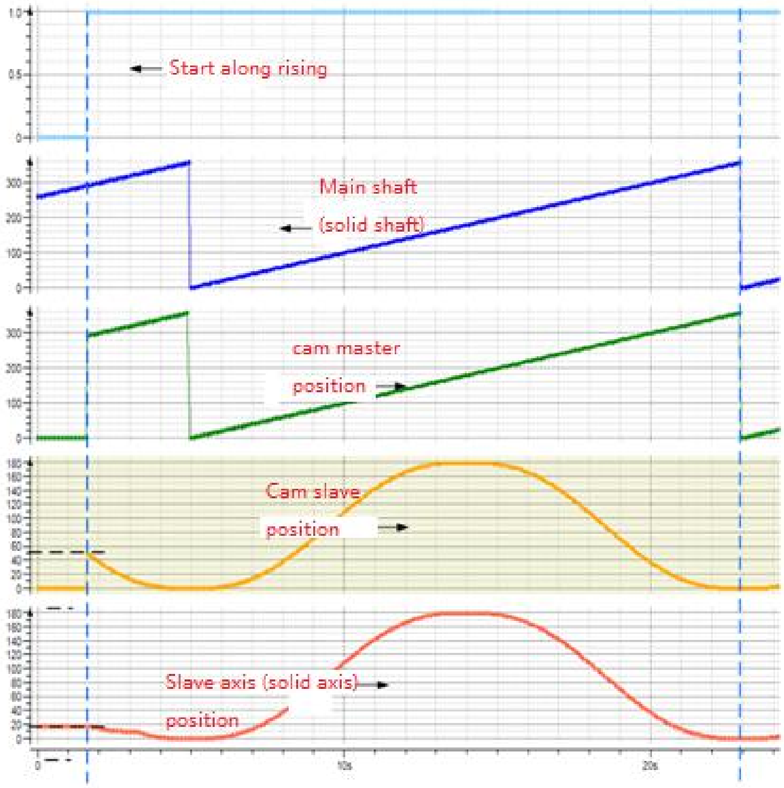


■ StartMode is 2 (rampin ramp-in mode)

① When the MC_CamTableSlect command MasterAbsolute is set to TRUE and SlaveAbsolute is set to TRUE

The master axis is in absolute mode and the slave axis is in absolute mode. When the cam is activated along the Excute rise, the cam spindle starts at the "current position of the spindle" and the slave axis adds a compensating movement to avoid potential jumps during cut-in by setting VelocityDiff, Acceleration, Deceleration.

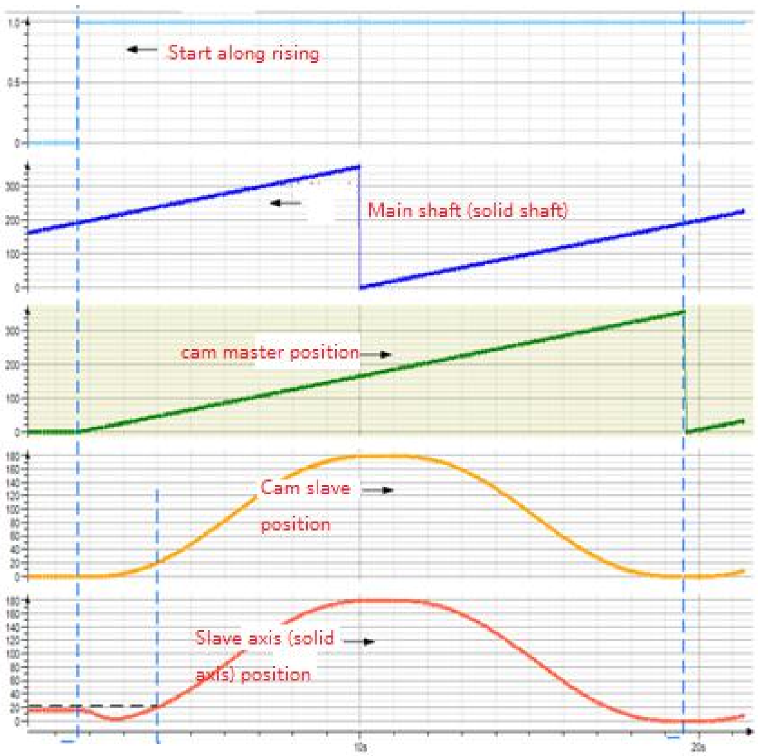
Slave real axis position command = cam table tooth fit calculation (cam slave position) + f(VelocityDiff,Acceleration,Deceleration)



② When the MC_CamTableSlect command MasterAbsolute is set to FALSE and SlaveAbsolute is set to TRUE

The master axis is in relative mode and the slave axis is in absolute mode. When the cam is activated along the Excute rise, the cam spindle starts from the "cam spindle start position" and the slave adds a compensating motion to avoid potential jumps during cut-in by setting VelocityDiff, Acceleration, Deceleration. The slave axis adds a compensating motion to avoid potential jumps during cut-in by setting VelocityDiff, Acceleration, Deceleration.

Slave real axis position command = calculated cam table tooth fit (cam slave position) + f(VelocityDiff,Acceleration,Deceleration).

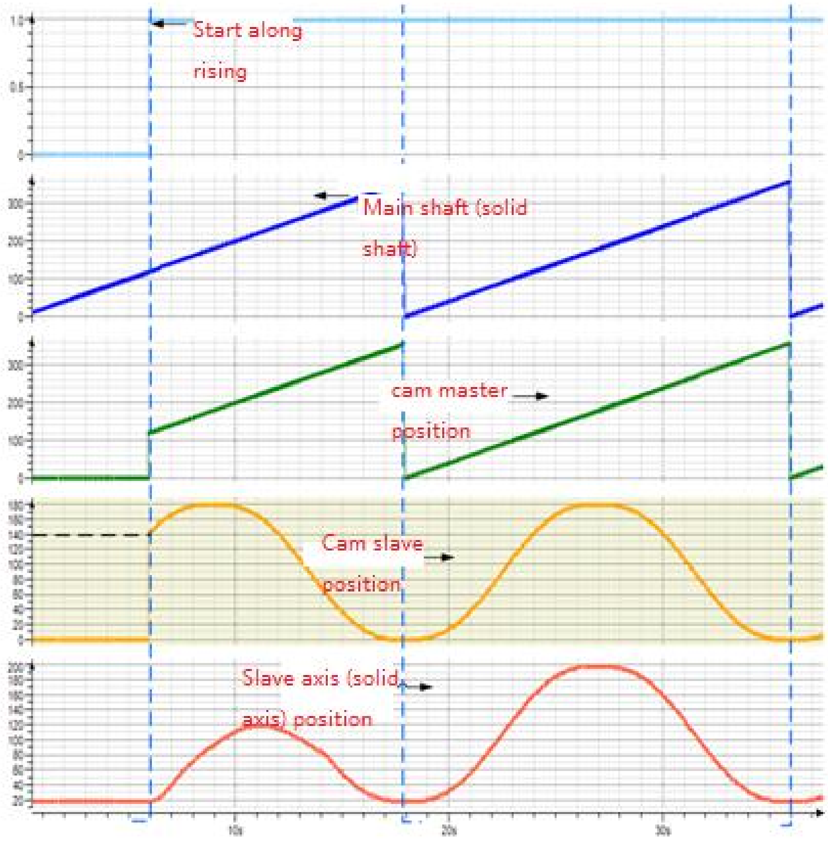


③When the MC_CamTableSlect command MasterAbsolute is set to TRUE and SlaveAbsolute is set to FALSE, the spindle is in absolute mode and the slave is in relative mode. When the Excute rises, the cam starts

The cam spindle starts at the "current position of the spindle" and the slave axis adds a compensating motion to avoid potential jumps during cut-in by setting VelocityDiff, Acceleration, Deceleration.

Slave real axis position command = Slave current position + Cam table tooth fit calculation value (cam slave position) + f(VelocityDiff,Acceleration,Deceleration).

Note: The cam curve may vary significantly from the design curve during the first spindle cycle in this method

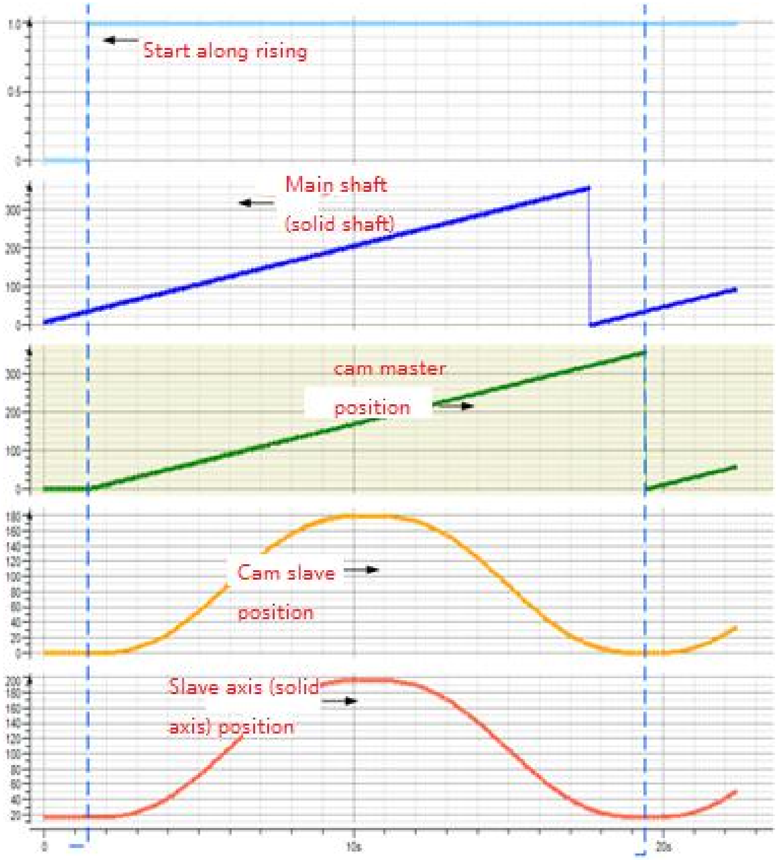


④ When the MC_CamTableSlect command MasterAbsolute is set to FALSE and SlaveAbsolute is set to FALSE, the master axis works in relative mode and the slave axis works in relative mode. When the Excute rises, the cam spindle starts from the "cam spindle start position" and the slave axis adds a compensating motion to avoid potential jump during cut-in by setting VelocityDiff, Acceleration, Deceleration. The slave axis adds a compensating motion to avoid potential jump during cut-in by means of the set VelocityDiff, Acceleration, Deceleration.

Slave real axis position command = current position of the slave axis + calculated value of the cam table tooth fit (cam slave position)

+f(VelocityDiff,Acceleration,Deceleration).

Note: The cam curve may vary significantly from the design curve during the first spindle cycle in this method



■ StartMode is 3, 4 (forward ramp in ramp_in_pos, reverse ramp in ramp_in_neg)

When the axis is in "rotary mode" ramp_in_pos only compensates in the direction of forward axis movement and ramp_in_neg only compensates in the direction of reverse axis movement, when the axis is in linear mode ramp_in_pos, ramp_in_neg and ramp_in are automatically adjusted for the direction of compensation, i.e. if the axis is set to work in linear mode the ramp_in_pos, ramp_in_neg and ramp_in start modes work in the same way.

◆ **Electronic cam restart**

Basically, the two e-cams can be switched at any time, but there are a number of cases to consider: in the e-cam editor, the position of the slave is defined as the calculated output of the e-cam function, which is calculated on the basis of a master position within the range of the master axis, and can thus be illustrated by the following simple formula.

SlavePosition = CAM(MasterPosition)

Since the actual period of the spindle drive is generally different from the spindle range defined by the electronic cam, the spindle position must be scaled to the function definition in order to satisfy the correct input to the electronic cam function.

SlavePosition = CAM(MasterScale*MasterPosition + MasterOffset)

In a similar way, if an electronic cam is started in absolute mode and produces an upward jump, the function output (i.e. the virtual slave position) will also be corrected proportionally: the

SlavePosition = SlaveScale*CAM(MasterPosition) + SlaveOffset

In the worst case, both of these scaling corrections must be applied, so that in fact the slave position (SlavePosition) is calculated by the more complex formula

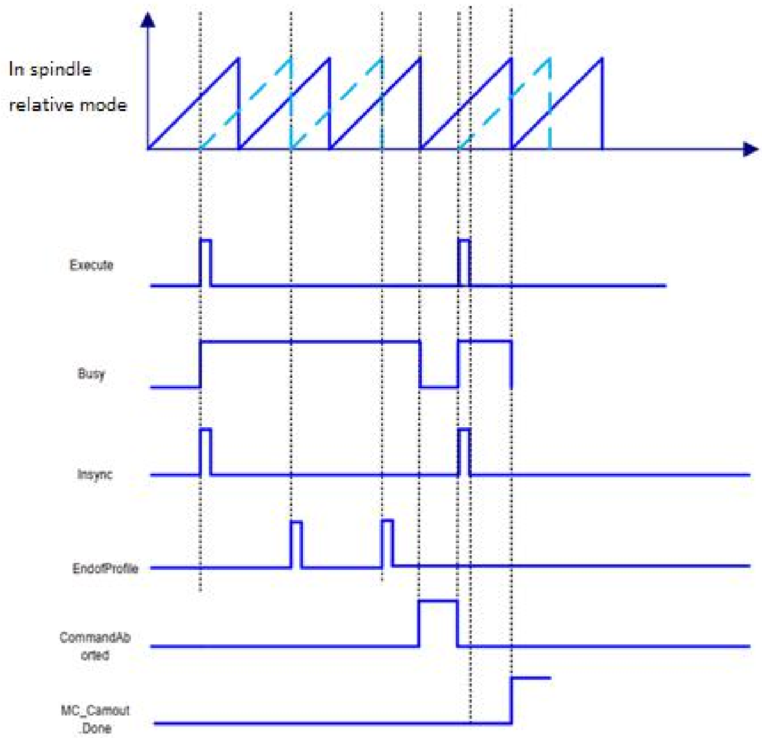
Slaveposition = SlaveScale*CAM(MasterScale*Masterposition + MasterOffset) + SlaveOffset

At the end of each e-cam cycle, the scale and offset can be changed to obtain more suitable parameters. Unfortunately, the restart of the MC_CamIn module of the electronic cam will delete its memory and include the scale and offset values. As a result, the defined electronic cam function will be adapted to the different slave values in general. For this reason, it is recommended to restart MC_CamIn-FB only if a different electronic cam needs to be processed.

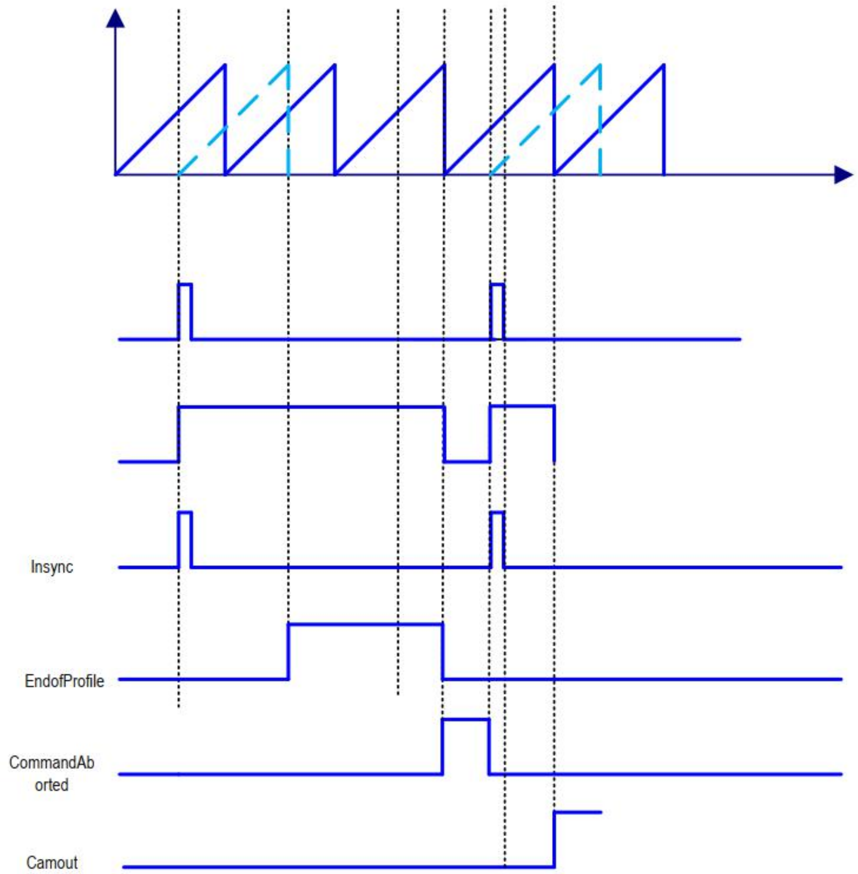
5) Timing diagram.

Periodic mode (MC_CamTableSelect.Periodic set to TRUE) is shown below.

Note: The MC_Camout instruction only cuts off the cam coupling between the master and slave axes, if the slave axis speed is not 0 when it is cut off, the slave axis will not automatically decelerate to 0.



The non-periodic mode (MC_CamTableSelect.Periodic set to FALSE) is as follows:



6) Error Description

The command setup information does not match the Camslect command setup information.

The axis is not enabled.

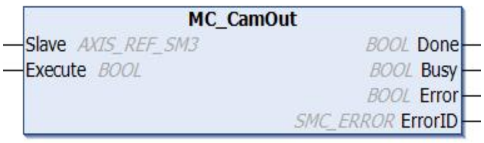
When an exception is detected by starting this instruction, Error becomes TRUE.

See the output of ErrorID (error code) and read "Appendix C Error Code Descriptions" for a description of the error code.

7.4.6 MC_CamOut

Disconnect the cam coupling relationship from the shaft. When the slave is running on the cam, triggering the execution of the function block causes Slave to exit the cam run state from the shaft and enter a continuous running state (Continuios_Motion, i.e. Axis.nAxisState 5), and the execution of the instruction has no effect on the spindle. Note: After executing this instruction, the axis continues to run at the pre-separation speed, so it needs to be used with instructions such as MC_Stop.

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
MC_CamOut	Disconnect cam coupling		MC_CamOut(Slave:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Slave	From the axis	AXIS_REF	-	-	Map to the axis, which AXIS_REF_SM3 instance of the property

Input

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	Execute the instruction	BOOL	-	-	The rising edge signal executes the instruction

The output

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	Complete	BOOL	TRUE,FALSE -	- FALSE	Complete the cam coupling disconnect from the spindle
Busy	In action	BOOL	TRUE,FALSE	FALSE	The instruction is executed
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output

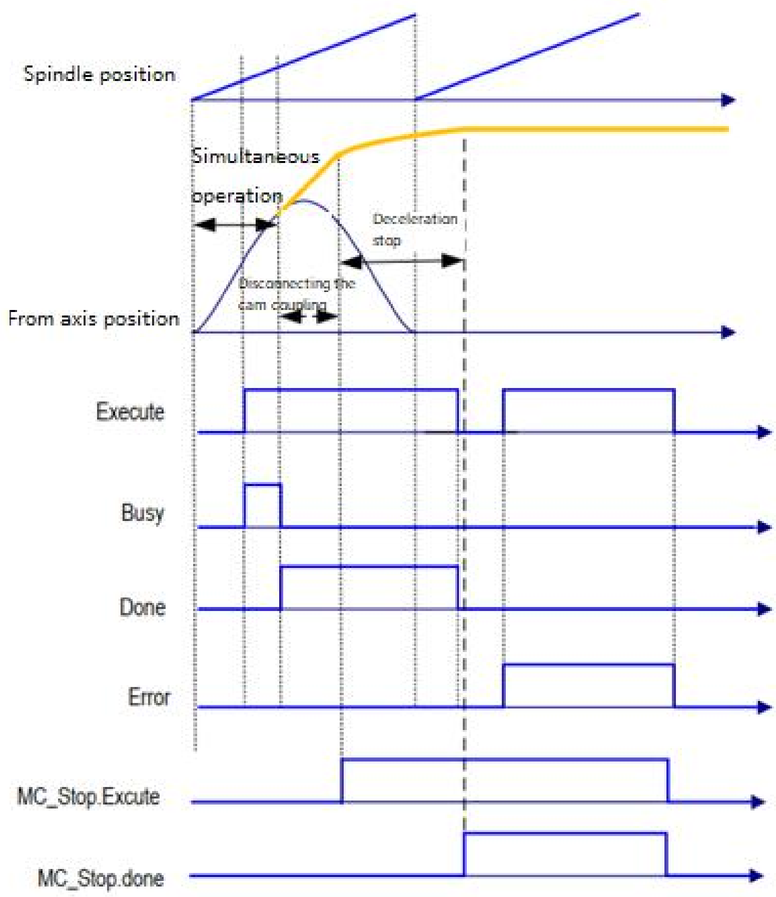
3) Function Description

Execute this instruction to disarm the cam coupling relationship from the shaft,excute the cam coupling relationship from the shaft is broken whenexcute rises, and the cam relationship does not necessarily stop after the cam relationship is disconnected;

If the speed of the from the shaft is not 0 before the instruction is executed, the cam coupling relationship is broken after the instruction DONE signal is completed but runs at the pre-cut speed from the shaft at will;

If the execution is performed from the axis without a cam coupling relationship, the ERROR output.

4) Timing diagram

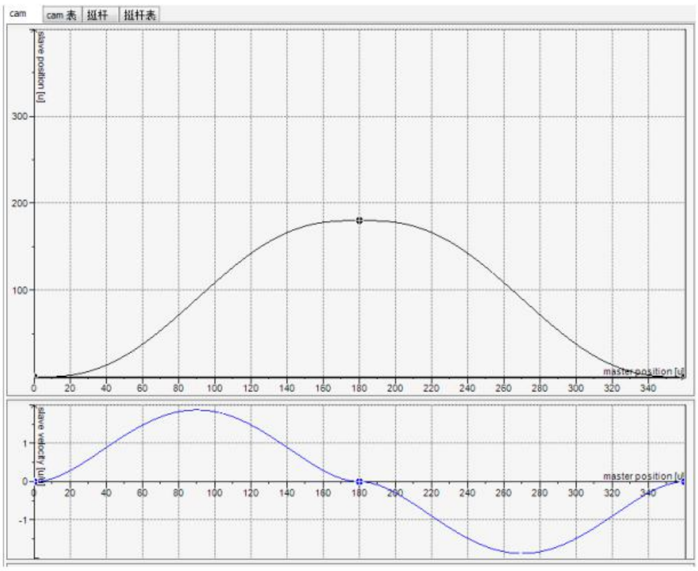


5) Example of use

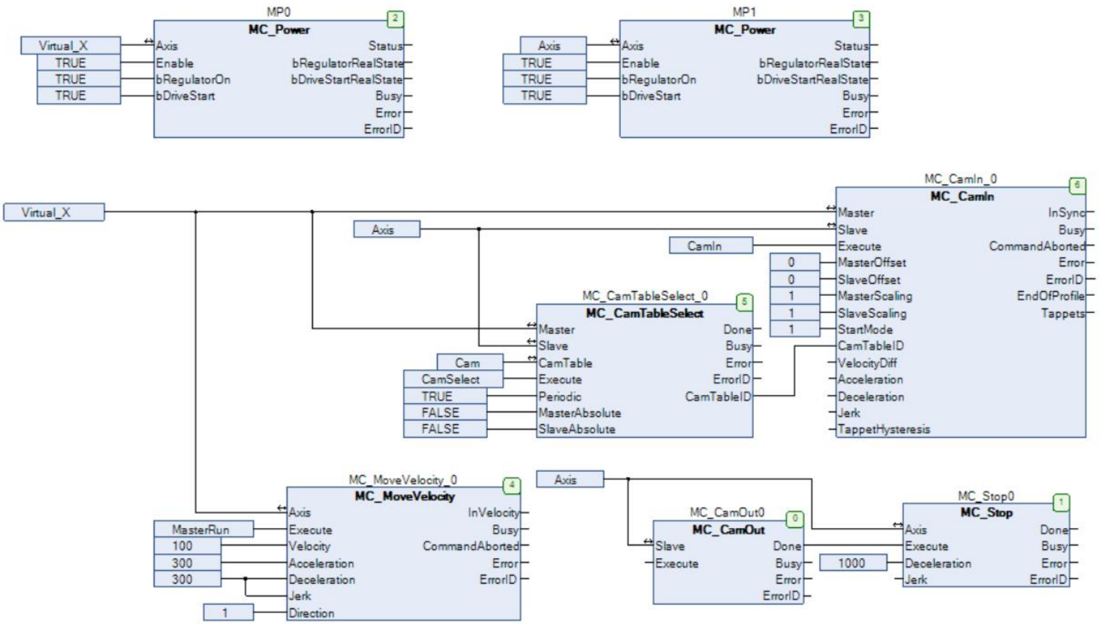
This example applies cam-related commands to introduce the creation of cam relationships and the relevant motion states of the axes when running and disengaging

The cam editor creates the following cam table (cam):

	X	Y	V	A	J	节类型	min(P...	max(P...	max(|V...	max(|A...
	0	0	0	0	0					
						Poly5	0	180	1.8749...	0.0320...
	180	180	0	0	0					
						Poly5	0	180	1.8749...	0.0320...
	360	0	0	0	0					



Program：



Master and slave axes are automatically enabled after power-up, MasterRun is set to TRUE to run the spindle at 100 speed

CamSelect is set to True to select the cam table, then CamIn is set to True to start the electronic cam.

When the electronic cam needs to be disconnected set MC_CamOut0.Execute to True.

6) Error description

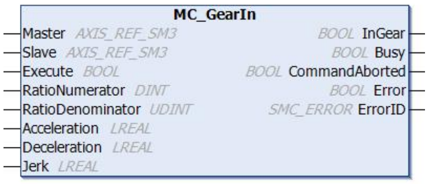
If an error occurs when starting this command, the Error output is True.

See ERRORID and refer to "Appendix C Error Code Descriptions" for SMC_ERROR error codes.

7.4.7 MC_GearIn

Set the gear ratio between the shaft and the spindle for electronic gear action.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_GearIn	Electronic gear function block		MC_GearIn0(Master:= , Slave:= , Execute:= , RatioNumerator:= , RatioDenominator:= , Acceleration:= , Deceleration:= , Jerk:= , InGear=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Map to the axis, AXIS_REF_SM3 instance of the map
Slave	From the axis	AXIS_REF	-	-	Map to the axis, AXIS_REF_SM3 instance of the map

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	Perform	BOOL	TRUE- Cutter	FALSE	Rise the edge and start executing the instructions
RatioNumerator	Gear ratio molecules	DINT	Positive, negative	1	Gear ratio molecules
RatioDenominator	Gear score mother	UDINT	Positive	1	Gear score mother
Acceleration	Acceleration	LREAL	Positive or 0		Specifies acceleration
Deceleration	Reduce the speed	LREAL	Positive or 0		Specifies a reduction in speed
Jerk	the degree	LREAL	Positive or 0		Acceleration

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
InGear	gear ratio arrived	BOOL	TRUE- Cutter	- FALSE	True, the target speed is reached from the axis

Busy	In action	BOOL	- TRUE,FALSE	FALSE	True, the instruction is being executed
Command Aborted	Interrupt	BOOL	TRUE,FALSE	FALSE	True, interrupted by other control instructions
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Function description

Execute rising edge to start the electronic gear action.

To uncouple the electronic gear after execution, the GearOut command must be used.

This instruction is a speed e-gear function and the loss of synchronisation distance caused during acceleration is not automatically compensated.

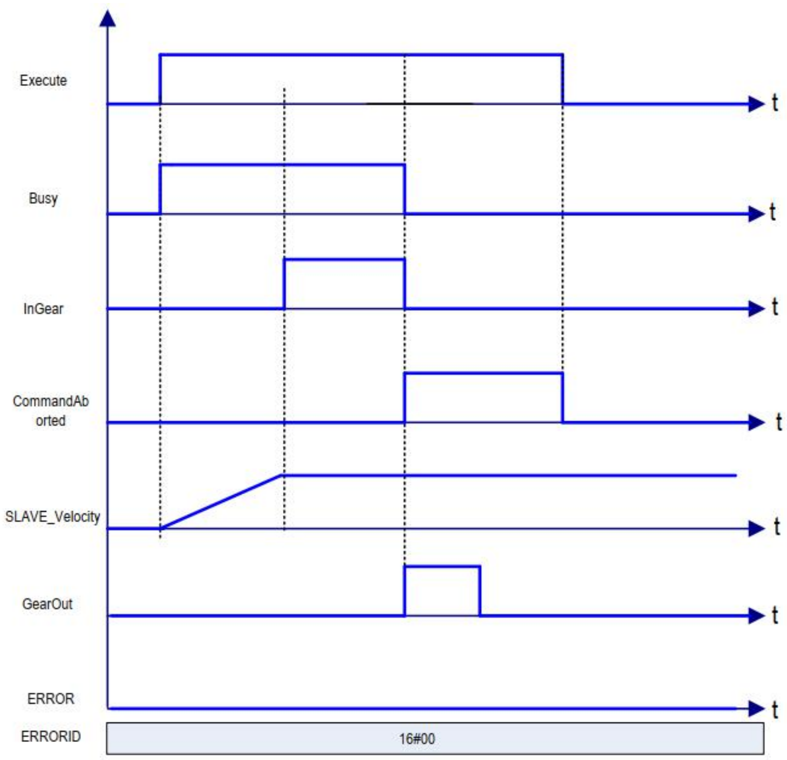
If the Busy signal is TRUE during the execution of the instruction, the new rising edge of Execute will not affect the target speed of the slave axis if it is not reached.

If the Busy signal is TRUE during instruction execution, the new rising edge of Execute will not affect it if the target speed of the slave axis is reached. When the target speed is reached, InGear is TRUE and the slave axis travel = master axis travel * RatioNumerator/RatioDenominator.

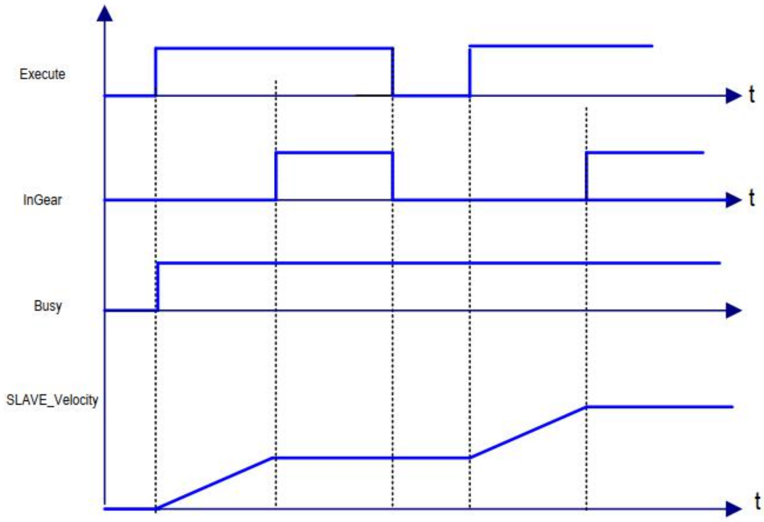
Please take care when using this command if the spindle speed is changing in real time.

Note: Do not use the MC_SetPosition instruction during the execution of the instruction to avoid accidents caused by the motor running rapidly.

◆ Timing diagram.:



The timing diagram for the restart command after changing the gear ratio parameter is as follows:



4) Error description

An error is output when the ERROR is TRUE for a start-up command.

Please read "Appendix C Error Code Descriptions" for a description of the relevant error codes.

7.4.8 MC_GearOut

To terminate an MC_GearIn in MC_GearInPos order.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_GearOut	The electronic gear coupling is broken		MC_GearOut0(Slave:= Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Slave	From the axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	Perform	BOOL	TRUE- Cutter	Fakse	Rise the edge and start executing the instructions

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	Complete	BOOL	TRUE- Cutter	FALSE	True, the coupling between the shaft and the spindle electronic gear is broken
Busy	In action	BOOL	TRUE,FALSE	FALSE	True, the instruction is in the process of being executed
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ ERROR	0	When an exception occurs, the error code is output

3) Function description

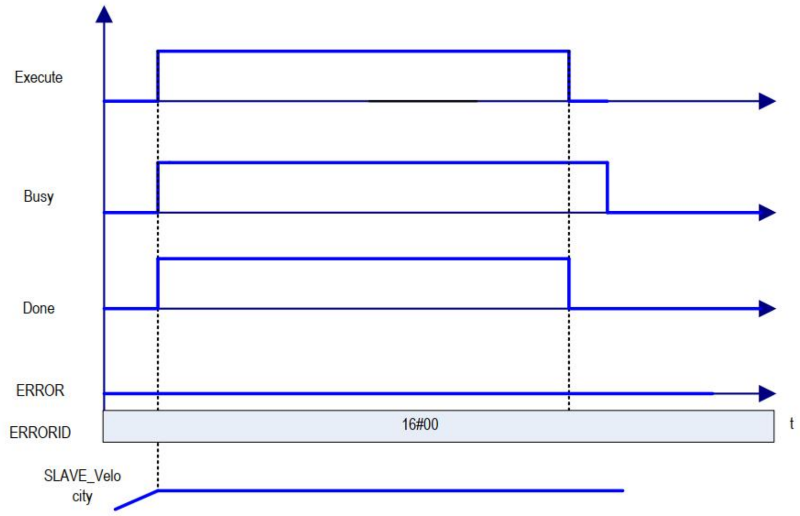
Execute rising edge, executes cut-out electronic gear action.

If Excute is TRUE and ERROR is FALSE, the Busy output is TRUE and the Done output is TRUE.

When the electronic gear is cut out, the speed of the slave axis is the speed before it is cut out, so the slave axis must be stopped with the MC_Stop command.

Execute is FALSE, Done is FALSE

MC_Stop instruction executes the reset Busy signal



4) Error description

An error in the setting of the relevant parameter will result in a command alarm.

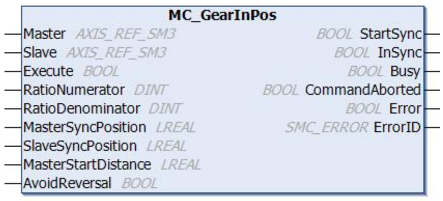
Axis not enabled will cause the command to alarm.

Note]: Please read "Appendix C Error Code Descriptions" for the description of the relevant error codes.

7.4.9 MC_GearInPos

Set the ratio of electronic gears between the spindle and the axle to perform electronic gear movements.
Specify the spindle position at which synchronization begins, the synchronization distance from the axis position, and the spindle to complete the cut-in electronic gear movement.

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
MC_GearInPos	The specified position is cut into the electronic gear coupling		MC_GearInPos0(Master:= Slave:= Execute:= , RatioNumerator:= , RatioDenominator:= , MasterSyncPosition:= , SlaveSyncPosition:= , MasterStartDistance:= , AvoidReversal:= , StartSync=> , InSync=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
Slave	From the axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The instruction is executed	BOOL	TRUE TICK	-FALSE	Rise the edge and start executing the instructions
Ratio Numerator	Gear ratio molecules	DINT	-	1-	The molecule of the spindle velocity ratio
Ratio Denominator	Gear score mother	DINT		1	The denominator of the spindle velocity ratio
Master SyncPosition	The spindle synchronization position	LREAL			The spindle position when the spindle gear ratio is coupled

Slave SyncPosition	Synchronize position from axis	LREAL			The position from the shaft when the spindle gear ratio is coupled
Master StartDistance	Performs the synchronization spindle position	LREAL			A smooth curve is calculated from the axis according to the position value and the MasterSyncPosition and SlaveSyncPosition values so that the axle is synchronized with the spindle gear at SlaveSync, with a curve spindle range of "MasterStartDistance, MasterSyncPosition"
AvoidReversal	Reversal is prohibited	BOOL	TRUE Fakse	FALSE	Set to FALSE if reversed from the physical position of the axis ahead. Set to TRUE if the reversal is not physically possible from the axis or causes a hazard. Only under the modal axis. If the reversal cannot be avoided, the axis stops incorrectly.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
StartSync	Start coupling processing	BOOL	TRUE- FALSE	ALSE	True, start the electronic gear coupling process
InSync	coupling	BOOL	TRUE- FALSE	FALSE	True, electronic gear coupling is complete and the spindle gear ratio is coupling
Busy	The instruction is in process	BOOL	TRUE,FALSE	FALSE	True, the instruction is in process
Command Aborted	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	Interrupted by other control instructions
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output

3) Function description

Execute The rising edge signal starts the execution of the command. After the start of the action, the Slave takes the speed of the Master multiplied by the gear ratio as the target speed and accelerates and decelerates.

The process from the start of synchronisation to the end of synchronisation is essentially an electronic cam in which the slave follows the master axis during the synchronisation interval. range (MasterSyncPosition - MasterStartDistance, MasterSyncPosition), the slave range (current position, SlaveSyncPosition), and the slave range (current position).

The command will automatically design a cam curve based on the set gear ratio and the three parameters mentioned above, so that the slave axis follows the master axis during synchronisation.

Note that if the master and slave axes are in linear mode, it is recommended that the master and slave axes are in cyclic mode, as the above parameters must be set correctly otherwise the gear action will not be carried out correctly. Example.

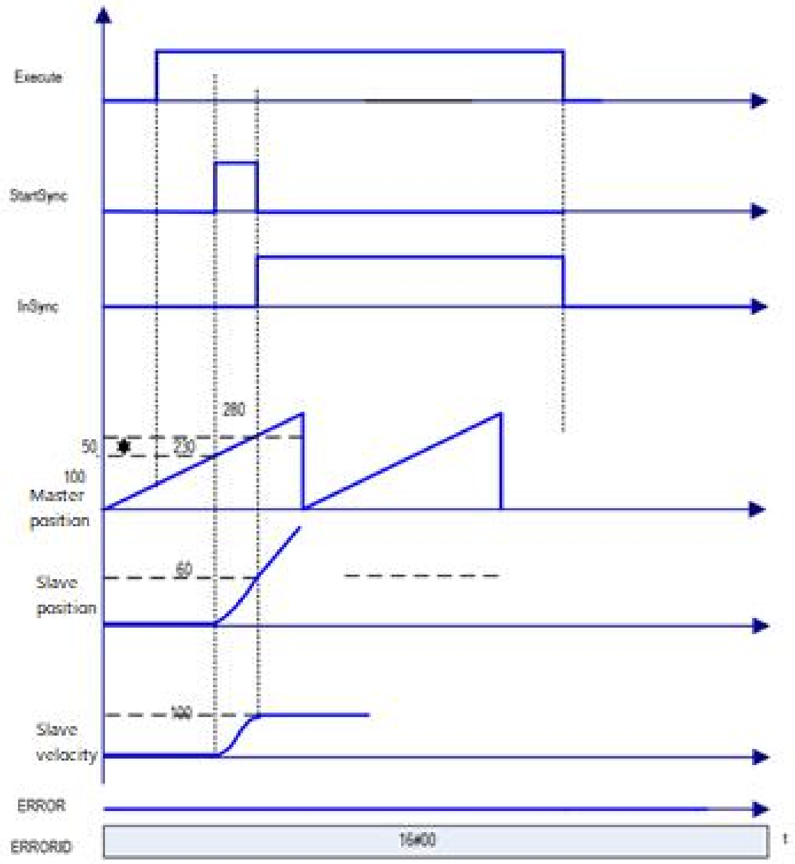
The master and slave axes work in linear mode both in forward motion, if the command is executed with

Master position > MasterSyncPositionMasterStartDistance, or Slave position > SlaveSyncPosition, the electronic gear action cannot be cut in.

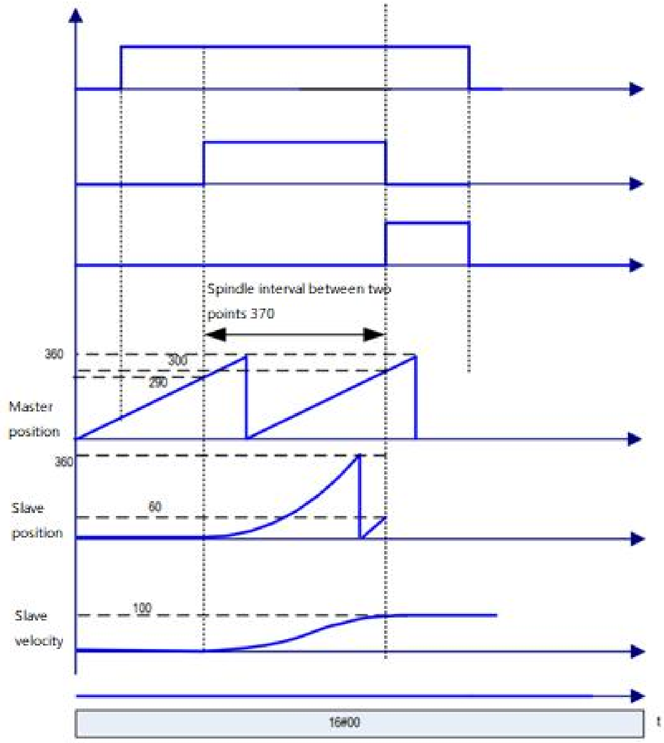
Sample timing diagrams for several different parameters are given below.

When the master axis is operating in cyclic mode (360 cycles) and the slave axis in cyclic mode (360 cycles):

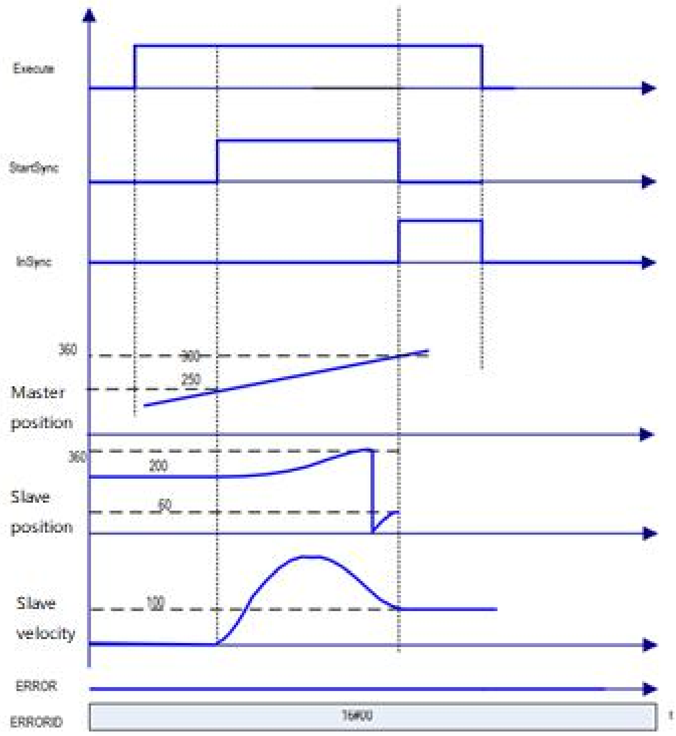
① MasterSyncPosition=280, MasterStartDistance=50, SlaveSyncPosition=60, spindle speed is 50, spindle speed is 50.
AvoidReversal=FALSE



② MasterSyncPosition=300, MasterStartDistance=370, SlaveSyncPosition=60, spindle speed is 50, AvoidReversal=FALSE



③MasterSyncPosition=300, MasterStartDistance=50, SlaveSyncPosition=60, spindle speed 50, AvoidReversal=FALSE, slave start position greater than 60

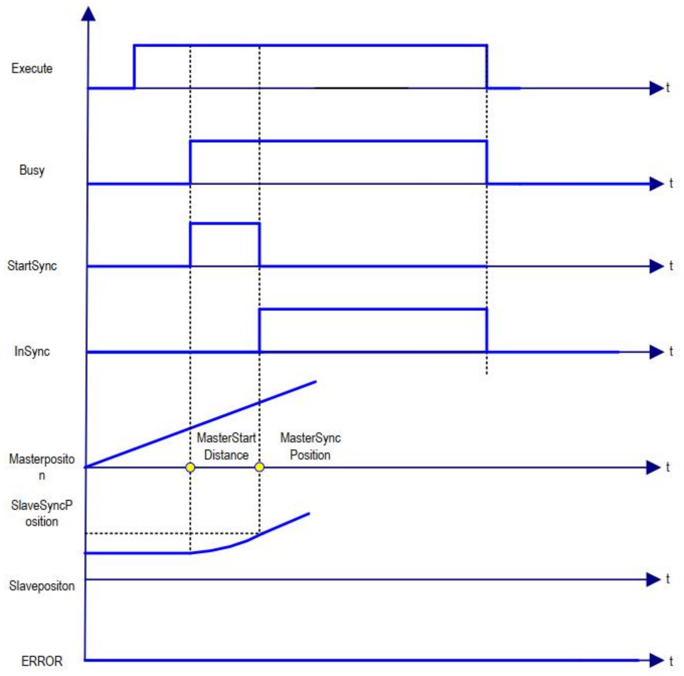


The target speed is reached at the same time as the synchronisation is completed (InSync is TRUE), after which

Slave axis travel = Master axis travel * RatioNumerator/RatioDenominator.

For AvoidReversal: MC_GearInPos tries to avoid the reversal of the slave axis if the slave axis is a modal axis and the spindle speed (multiplicative relationship of gear ratios) is not relative to the speed relationship of the slave axis. It tries to "stretch" the slave motion by adding 5 slave cycles. If this "stretch" is not effective, then an error occurs and the slave axis stops incorrectly. If the slave axis speed is related to the main axis speed (a multiple of the gear ratio), then an error will occur and the axis will stop incorrectly. If the slave axis is a linear axis mode axis, then an error will occur when the rising edge of the Execute input is processed.

4) Timing diagram:



5) Error description

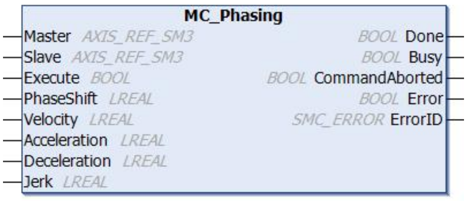
- An error in the setting of the relevant parameter will lead to a command alarm.
- The command alarm will be caused if the axis is not enabled.

[Note]: Please read "Appendix C Error Code Descriptions" for the description of the relevant error codes.

7.4.10 MC_Phasing

Specifies the phase deviation between the spindles.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
MC_ Phasing	The primary is offset from the axis phase		MC_Phasing0(Master:= Slave:= Execute:= , PhaseShift:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
Slave	From the axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter the relevant variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Execute	The instruction is executed	BOOL	TRUE- Cutter	FALSE	Rise the edge and start executing the instructions
PhaseShift	The value of the main phase deviation from the axis	LREAL		0	The main phase deviation value of the spindle, and the positive number represents the lag from the axis.
Velocity	Speed	LREAL		0	The maximum velocity value when the phase offset is performed
Acceleration	Acceleration	LREAL		0	The maximum acceleration value when the phase offset is performed
Deceleration	Reduce the speed	LREAL		0	The maximum deslevel value when the phase offset is performed

Jerk	Velocity secondary conductor	LREAL		0	The maximum Jerk value when the phase offset is performed

Output related variables

The output variable	Name	The data type	Effective range	The initial value	Describe
Done	Complete	BOOL	TRUE- FALSE	FALSE	True, if the phase offset is complete
Busy	The instruction is in process	BOOL	TRUE,FALSE	FALSE	True, the instruction is in process
Command Aborted	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	Interrupted by other control instructions
Error	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
ErrorID	The error code	SMC_ ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output

3) Function description

Execute the phase shift on the rising edge, the slave axis automatically calculates a smooth curve and completes the phase shift of the slave axis to the main axis, the phase difference between the master and slave axis is the PhaseShift value of the input signal, a positive value means that the slave axis lags behind the main axis.

The Done signal is output as True after the offset is completed.

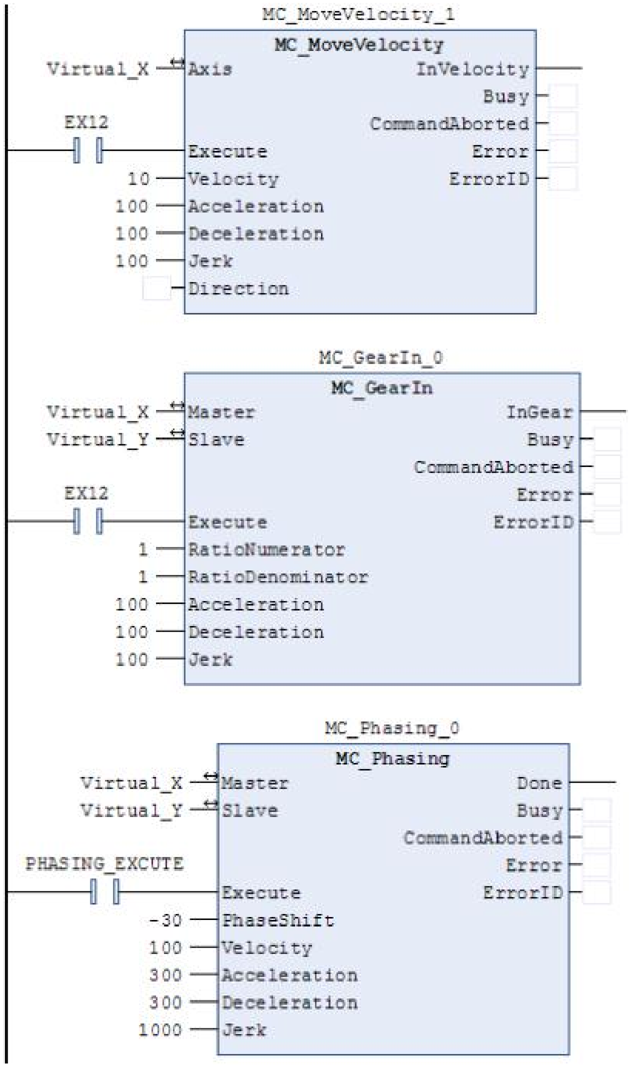
The master-slave phase difference is compensated according to the set PhaseShift, Velocity, Acceleration and Deceleration.

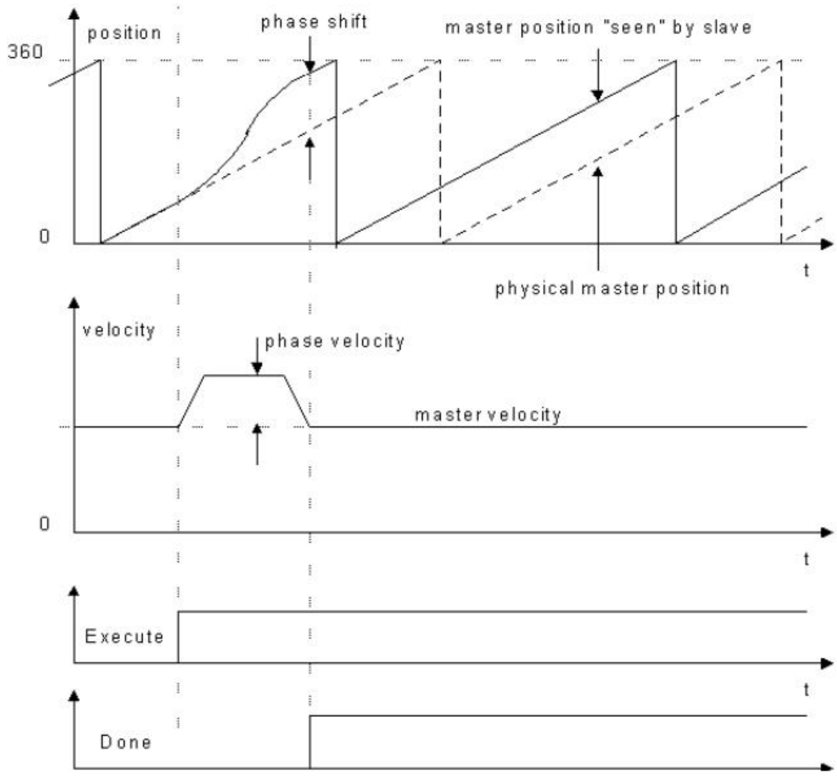
When the phase difference between the master and slave axes reaches PhaseShift, the Done signal is output.

The spindle command position and the feedback position remain unchanged during the execution of the command, while the slave axis is adjusted.

The final result of this instruction is a phase shift between the given values of the axes, so the actual feedback value of the real axis may not match the final shift.

This instruction can be used in conjunction with the MC_GearIn instruction as follows: the spindle is Virtual_x, the slave is Virtual_y, the EX12 rising edge performs the spindle speed control and the master and slave electronic gear action, then the phase shift is performed. It can also be used with the electronic cam, where the slave axis acts as the "electronic cam spindle" to achieve the electronic cam spindle phase shift effect.

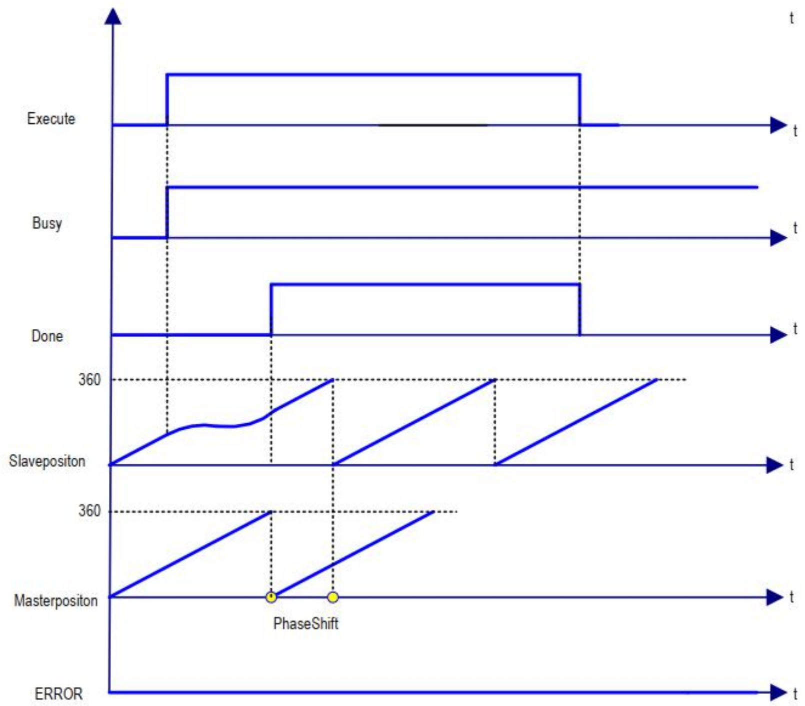

MC_MoveVelocity_1
MC_MoveVelocity
Virtual_X
Axis
InVelocity
Busy
CommandAborted
EX12
Execute
Error
10
Velocity
ErrorID
100
Acceleration
100
Deceleration
100
Jerk
Direction
MC_GearIn_0
MC_GearIn
Virtual_X
Master
InGear
Virtual_Y
Slave
Busy
CommandAborted
EX12
Error
Execute
ErrorID
1
RatioNumerator
1
RatioDenominator
100
Acceleration
100
Deceleration
100
Jerk
MC_Phasing_0
MC_Phasing
Virtual_X
Master
Done
Virtual_Y
Slave
Busy
CommandAborted
PHASING_EXCUTE
Error
Execute
ErrorID
-30
PhaseShift
100
Velocity
300
Acceleration
300
Deceleration
1000
Jerk



4) Timing diagram

With the master and slave axes moving in 360 cycles, the rising edge of the Execute signal performs the adjustment and the phase deviation between the slave axis and the main axis after the adjustment is completed is

The value set by PhaseShift



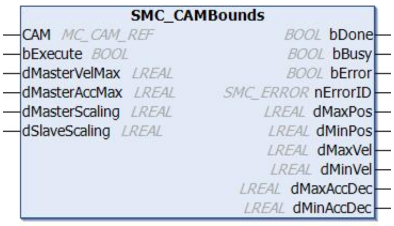
5) Error Description

- If the error output is TRUE when starting the command, an error has occurred.
- Check the ErrorID, check SMC_ERROR in the help to determine the alarm information, please read "Appendix C Error Code Description" for the related error code description.

7.4.11 SMC_CAMBounds

When the slave axis is coupled to the spindle cam this function block can be used to calculate the maximum position, speed and acceleration of the slave axis.
The spindle moves under the input maximum speed, acceleration and deceleration limits. This instruction is used to check the correctness of the curve when designing the cam table, provided
maximum acceleration and deceleration of the spindle, speed, etc. are known

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_CAMBounds	Cam upper and lower limits		MC_Phasing0(Master:= Slave:= Execute:= , PhaseShift:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
CAM	Cam	MC_CAM_REF	-	-	Maps to the cam, which is MC_CAM_REF of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	The instruction is executed	BOOL	TRUE- Cutter	FALSE	Rise the edge and start executing the instructions
dMasterVelMax	Maximum speed	LREAL	-	1	Maximum spindle speed in absolute mode.
dMasterAccMax	Maximum acceleration	LREAL	-	0	Maximum spindle acceleration in absolute mode
dMasterScaling	Ruler factor	LREAL	-	1	The spindle cam applies the ruler factor
dSlaveScaling	Ruler factor	LREAL	-	1	Apply the ruler factor from the

					shaft cam

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	Complete	BOOL	TRUE- FALSE	FALSE	True, if the calculation is complete
bBusy	The instruction is in process	BOOL	TRUE,FALSE	FALSE	True, the instruction is in process
bError	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
nErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output
dMaxPos	The maximum position	LREAL		0	The maximum position from the shaft is calculated from the cam table
dMinPos	The minimum position	LREAL		0	The minimum position from the shaft is calculated from the cam table
dMaxVel	Maximum speed	LREAL		0	Calculate the maximum speed
dMinVel	Minimum speed	LREAL		0	The minimum speed is calculated
dMaxAccDec	Maximum acceleration	LREAL		0	The maximum acceleration is calculated
dMinAccDec	Minimum acceleration	LREAL		0	The minimum acceleration is calculated

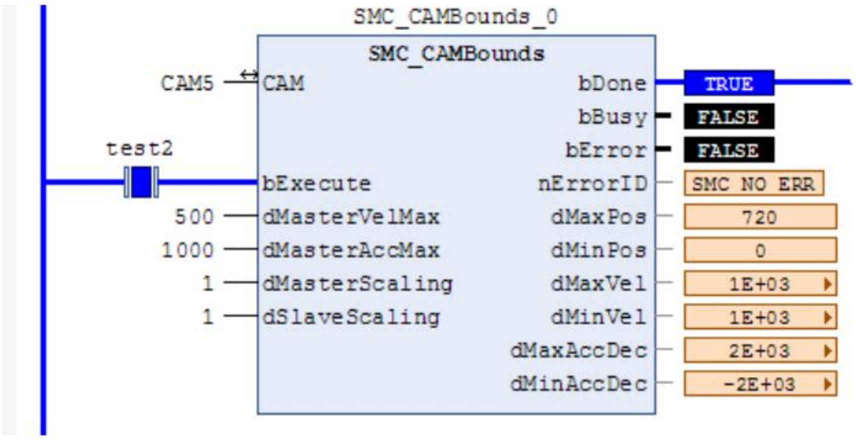
3) Function description

bExecute rising edge to combine the input variables "dMasterVelMax", "dMasterAccMax", "dMasterScaling", "dSlaveScaling", etc. with the cam table data to calculate the minimum position of the slave axis "max position".

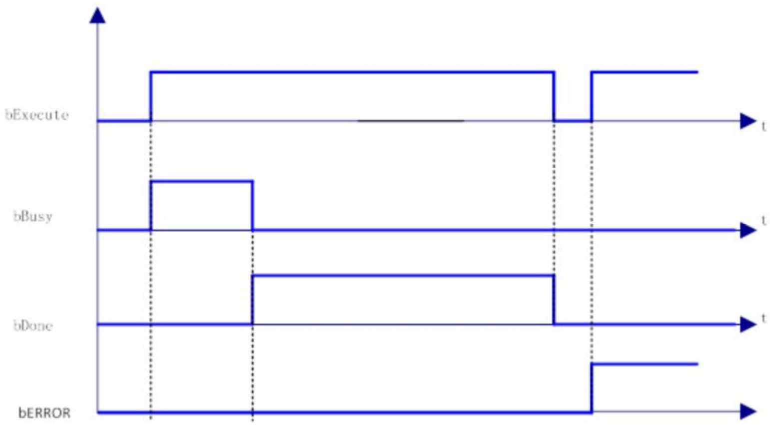
The "dSlaveScaling" values and the cam table data are used to calculate the "maximum position" and the minimum position of the slave axis. Example: spindle period 360, cam table A straight line with a slope of 2 is calculated as shown in the figure below.

This command can be used when the spindle is running in absolute mode or when the spindle is set to cycle mode and the modulus is set to the spindle cycle.

The cam table is XYVA (valid in polynomial mode), not valid for 1D arrays, 2D arrays etc.



4) Timing diagram



5) Error description

Cam table format is not polynomial mode.

The cam table MC_CAM_REF setting does not match the actual cam table.

[Note]: Please read "Appendix C Error Code Descriptions" for the relevant error code descriptions.

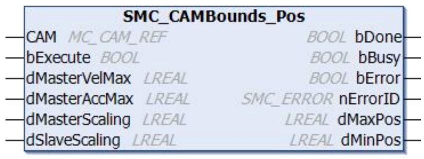
7.4.12 SMC_CAMBounds_Pos

When the from the shaft is coupled with the spindle cam, the maximum position of the from the shaft, with the minimum position, can be calculated by this function block. This function block is SMC_

calculations such as maximum acceleration compared to camBounds, and the other functions are consistent.

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance

SMC_ CAMBounds_Pos	The upper and lower limits of the cam position		`SMC_CAMBounds_Pos0(`
			`CAM:= ,`
			`bExecute:= ,`
			`dMasterVelMax:= ,`
			`dMasterAccMax:= ,`
			`dMasterScaling:= ,`
			`dSlaveScaling:= ,`
			`bDone=> ,`
			`bBusy=> ,`
			`bError=> ,`
			`nErrorID=> ,`
			`dMaxPos=> ,`
			`dMinPos=> );`

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
CAM	Cam	MC_CAM_REF	-	-	Maps to the cam, which is MC_CAM_REF of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	The instruction is executed	BOOL	TRUE- Cutter	FALSE	Rise the edge and start executing the instructions
dMasterVelMax	Maximum speed	LREAL		1	Maximum spindle speed in absolute mode.
dMasterAccMax	Maximum acceleration	LREAL		0	Maximum spindle acceleration in absolute mode
dMasterScaling	Ruler factor	LREAL		1	The spindle cam applies the ruler factor
dSlaveScaling	Ruler factor	LREAL		1	Apply the ruler factor from the shaft cam

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	Complete	BOOL	TRUE- FALSE	FALSE	True, if the calculation is complete
bBusy	The instruction is in process	BOOL	TRUE,FALSE	FALSE	True, the instruction is in process
bError	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
nErrorID	The error code	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the error code is output
dMaxPos	The maximum position	LREAL		0	The maximum position from the shaft is calculated from the cam table
dMinPos	The minimum position	LREAL		0	The minimum position from the shaft is calculated from the cam table

3) Function description

bExecute rising edge to combine the input variables "dMasterVelMax", "dMasterAccMax", "dMasterScaling", and

The "dSlaveScaling" values and the cam table data are used to calculate the "maximum position" and the minimum position of the slave axis.

This command can be used when the spindle is running in absolute mode or when the spindle is set to cycle mode and the module value is set to the spindle cycle.

The cam table is XYVA (valid in polynomial mode), not valid for 1D arrays, 2D arrays etc.

4) Error description

The cam table format is not polynomial mode; the set value of cam table MC_CAM_REF does not match the actual cam table.

Note]: Please read "Appendix C Error Code Descriptions" for the relevant error code descriptions.

7.4.13 SMC_WriteCAM

The program runs to store the edited cam table as a file. so that it can be MC_CamIn such as instructions. The resulting file contains a content reference called Cam Format.

This instruction can be used SMC_ReadCAM with other users.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_WriteCAM	Cam upper and lower limits		

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
CAM	Cam	MC_CAM_REF	-	-	Maps to the cam, which is MC_CAM_REF of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	The instruction is executed	BOOL	TRUE,Tick	FALSE	Rise the edge and start executing the instructions
sFileName	Filename	STRING	''		File names in ASCII format that contain cam descriptions can be viewed by helping to "Cam Format" inside.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	Complete	BOOL	TRUE- FALSE	FALSE	True, if the cam is written into the file to complete
bBusy	The instruction is in process	BOOL	TRUE,FALSE	FALSE	True, the execution of the instruction was not completed
bError	Error	BOOL	TRUE,FALSE	FALSE	When an exception occurs, it is set to TRUE
nErrorID	The error	SMC_ERROR	See SMC_ERROR	0	When an exception occurs, the

	code				output is erred

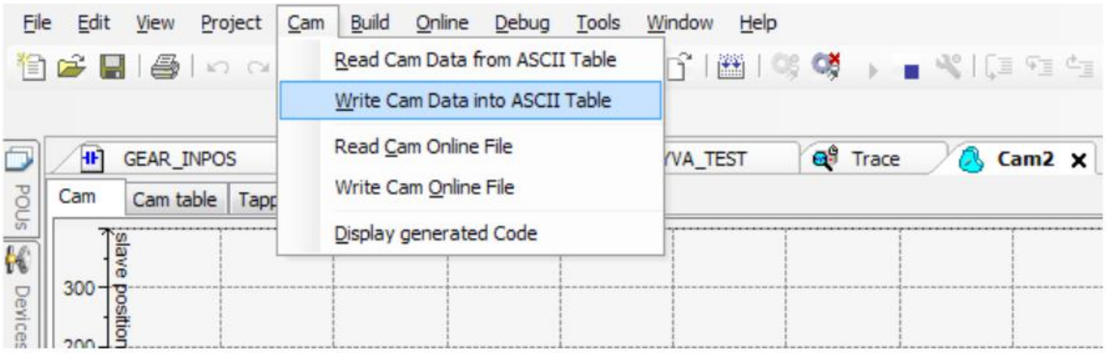
3) Function description

◆ bExecute rising edge, command execution - stores the cam information of the "CAM" connection in a file connected by the file name "sFileName".

The bDone signal is output as true.

◆ The stored cam table information is limited by the hardware memory.

◆ Note: This function is performed while the program is running and the cam table information can also be stored manually in the offline information.



4) Error description

◆ This command can only complete the cam table in XYVA polynomial mode, one-dimensional, two-dimensional, etc. will cause an error output

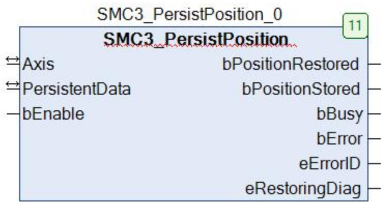
◆ sFileName The connected file name does not exist or the information is wrong.

[Note]: Please read "Appendix C Error Code Descriptions" to understand the error code descriptions.

7.4.14 SMC3_PersistPosition

This instruction is used to maintain the position of the record solid-axis absolute value encoder (after the power-off restarts the controller, the position-recording value before the power-off is restored). If the servo motor is using an absolute value encoder, use this function block in conjunction.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC3_ PersistPosition	The axis position is maintained		SMC3_PersistPosition0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related variables

. . . input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Map to the axis, AXIS_REF_SM3 instance of the property
PersistentData	Keep the data	SMC3_Persist Position_Data			The power-off-hold data structure that stores location information

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	True function blocks are executed, false does not execute function blocks, and to restore the last stored location during initialization, the value must be set to true when the application starts

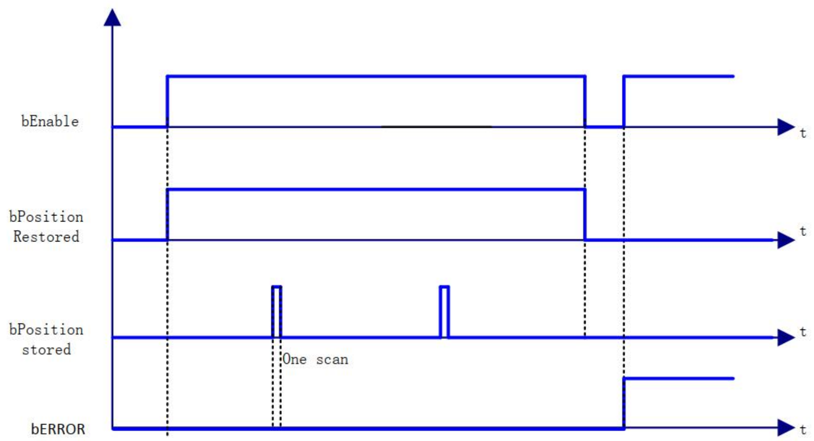
The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bPosition Restored	Location recovery	BOOL	TRUE,FALSE	FALSE	TRUE, position recovery completes after axis restart
bPosition Stored	Location save	BOOL	TRUE,FALSE	FALSE	TRUE, the call function is done quickly after saving the location

bBusy	FB in action	BOOL	TRUE,FALSE	FALSE	TRUE, the function block is not executed
bError	Error	BOOL	TRUE,FALSE	FALSE	TRUE, an exception occurs
eErrorID	The error code	SMC_ ERROR	SMC_NO_ERROR		When an exception occurs, The error code is output
eRestoringDiag	Restore diagnostics	SMC3_PersistPositionDiag	SMC3_ PersistPositionDiag. SMC3_PPD_ RESTORING_OK		Diagnostic information in location recovery SMC3_PPD_RESTORING_OK: Location recovery SMC3_PPD_AXIS_PROP_CHANGED: The axis parameters have changed and bit SMC3_PPD_DATA_STORED_DURING_WRITING cannot be recovered: the function block copies the data from the axis parameter data structure instead of from the PersistentData data. Possible causes: non-synchronous persistent variable, controller crash crash

3) Function Description

- If the PLC restart bEnable signal is TRUE, the output of bPositionRestroed is TRUE.
- The dummy axis is not supported to follow the logic axis.
- Timing diagram



4) Error description

- An input axis that is a virtual axis or a logical axis will result in an error output.
- There is an error in the axis.

Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.

7.4.15 SMC_FollowVelocity

Set the speed directly to the shaft without doing any checks. This instruction is different from MC_MoveVelocity in that each task cycle gives the axis speed instruction (the MC_MoveVelocity instruction speed changes andmust be refreshed to takeeffect).

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
SMC_ FollowVelocity	The axis speed is given		SMC_FollowVelocity_0(Axis:= , bExecute:= , fSetVelocity:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE,FALSE	FALSE	The rising edge executes the function block
fSetVelocity	Set the speed	LREAL		0	The speed set by the axis

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True: In the execution of instructions, where the axis is synchronized, as in the cam MC_CamIn instruction runtime axis state, the bBusy state can be cleared with MC_Camout instructions
bCommand Aborted	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	True: The axis is interrupted by other control commands (when bExecute is True).
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated

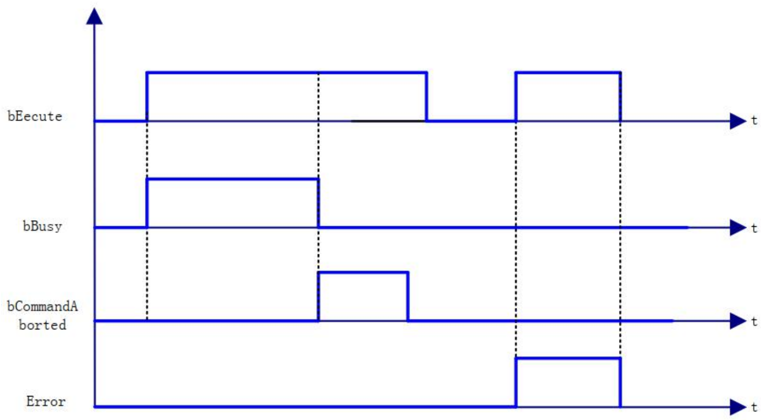
iErrorID	The error code	SMC_ ERROR			Reference SMC_Error

3) Function description

After SMC_FollowVelocity has been started by the rising edge of bExecute, the axis sends a velocity command to the axis every task cycle. bBusy is the same as the MC_CamIn command when the axis is in synchronous operation, and can be cleared with the MC_CamOut command.

When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when the command is interrupted by another control command.

- Timing diagram



4) Error description

bExecute on rising edge.

Axis variable connected to a non-AXIS_REF_SM3 type structure variable, Error output.

Axis is not enabled, Error is output.

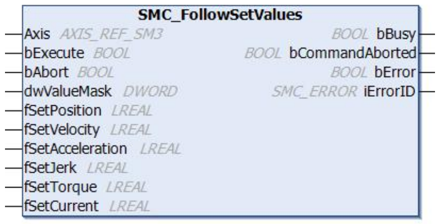
If the instruction is running and the axis is wrong, the error is output.

[Note]: Please read "Appendix C Error Code Descriptions" for related error code descriptions.

7.4.16 SMC_FollowSetValues

Like the other SMC_Follow functions, it is a direct axis command. However, this command includes not only the other SMC_Follow commands but also acceleration, current, torque and other control signals, so it can be considered a comprehensive version. The DwValueMask value is used to select the desired command.

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
SMC_ FollowSetValues	Axis-related commands given		SMC_FollowSetValues_0(Axis:= , bExecute:= , dwValueMask:= , fSetPosition:= , fSetVelocity:= , fSetAcceleration:= , fSetJerk:= , fSetTorque:= , fSetCurrent:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Implementation	BOOL	TRUE,FALSE	FALSE	Rising edge execution function block
DwValue Mask	Control Management	DWORD		0	Bite0:TRUE:fSetPosition active FALSE: ignored
bAbort	Interrupt	BOOL	TRUE,FALSE	FALSE	Bite1:TRUE: fSetVelocity active FALSE: ignored
fSetPosition	Set position	LREAL		0	Bite2:TRUE: fSetAcceleration active FALSE: ignored
fSetVelocity	Set speed	LREAL		0	Bite3:TRUE: fSetJerk active FALSE: Ignored
fSetAccel eration	Set acceleration	LREAL		0	Bite4:TRUE: fSetTorque active FALSE: Ignored
fSetJerk	Set jump value	LREAL		0	Bite5:TRUE: fSetCurrent active FALSE: ignored
fSetTorque	Set torque	LREAL		0	Rising edge interrupt function block
fSetCurrent	Set current	LREAL		0	Axis set position (calibrated units)

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	During execution	BOOL	TRUE,FALSE	FALSE	True- During the execution of the instruction, the
bCommand Aborted	Instruction interrupted	BOOL	TRUE,FALSE	FALSE	(the axis is in a synchronized state, the same as when the cam MC_CamIn instruction is running), the bBusy state can be cleared with the MC_Camout instruction
bError	Error	BOOL	TRUE,FALSE	FALSE	True- The axis is interrupted by another control command
iErrorID	Error code	SMC_ERROR			True, exception generated

3) Function description

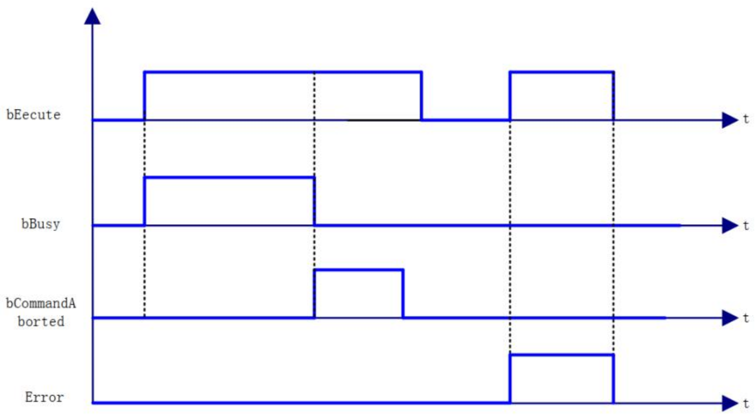
◆ After SMC_FollowSetValues has been started by the rising edge of bExecute, the axis sends the selected parameter command to the axis every task cycle.

◆ When the bBusy signal comes, the state of the axis is synchronous and the state of the slave axis is the same as when the MC_CamIn instruction is in effect, which can be cleared by the MC_CamOut instruction.

When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when the command is interrupted by another control command.

◆ The control parameter is selected by the DwValueMask value, for example, if DwValueMask is 1, the position is sent for each task cycle, just like the SMC_FollowPosition instruction. A DwValueMask of 2 is the output of a separate speed command. A DwValueMask of 3 is the output of a position velocity command. A DwValueMask of 7 is the output of a position, velocity, acceleration command, etc.

◆ Timing diagram



4) Error description

bExecute on rising edge.

Axis variable is connected to a non-AXIS_REF_SM3 type structure variable, Error output

Axis is not enabled, Error output.

The instruction is running, the axis is wrong, Error is output.

Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.

7.4.17 SMC_SetControllerMode

Set the current operating mode of the servo, default to the synchronization cycle position control, control mode related settings please refer to the servo control mode.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ SetControllerMode	Set the shaft control mode		`SMC_SetControllerMode0( Axis:= , bExecute:= , nControllerMode:= , bDone=> , bBusy=> , bError=> , nErrorID=> );`

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE,FALSE	FALSE	The rising edge executes the function block
nController Mode	Control mode	SMC_CONTR OLLER_MODE		SMC_ Position	Shaft control mode 1: torque control mode: SMC_torque 2: speed control mode,: SMC_Velocity 3:position control mode : SMC_Position 4: current control mode, SMC_Current

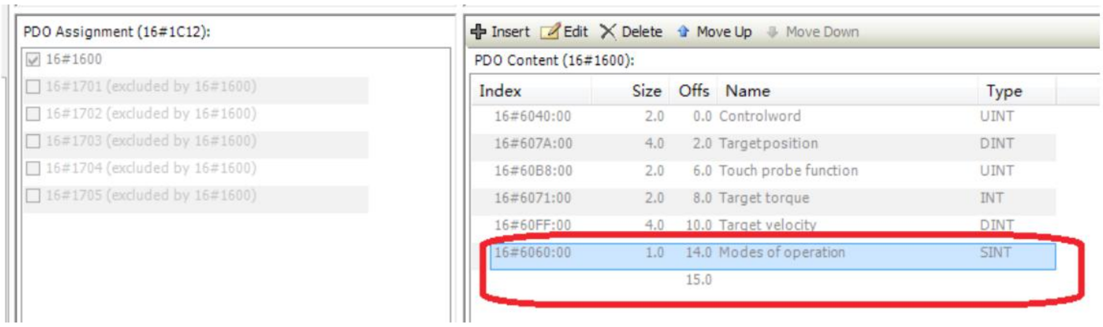
The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	The mode setting is complete	BOOL	TRUE,FALSE	FALSE	True, the mode setting is complete
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True - In the execution of instructions,
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated

iErrorID	The error code	SMC_ERROR			Reference SMC_Error

3) Function description

◆ SMC_SetControllerMode, which gives the servo drive the control mode command after the rising edge of bExecute is started, or the control mode can be set by the Axis.out.byModesofOpreation value after axis configuration (object dictionary 6060h has to be added to the process data).



◆ Conditions to be met for the use of the function block.

1: The axes must meet these control conditions, e.g. dummy axes cannot use the function block.

2: The synchronisation period supported by each mode must be the same (refer to the manual of "EtherCAT Servo" of WKD)

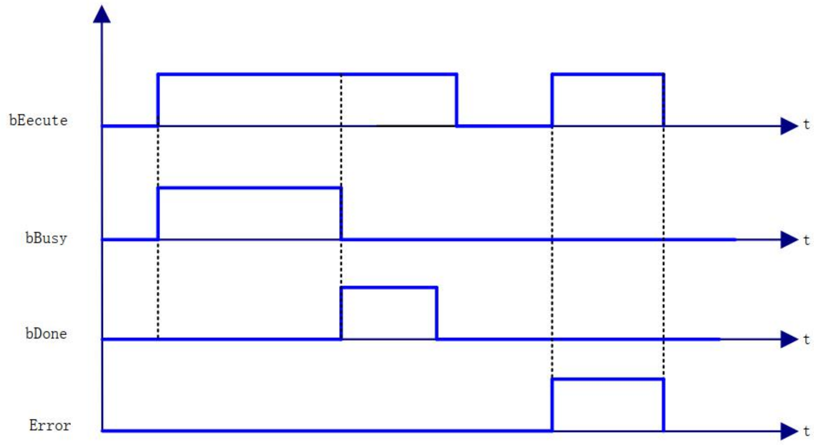
3: The axes must be in a state other than "errorstop", "stopping" or "homing" when the command is executed, otherwise an error will occur.

◆If the axis does not change to the set control mode after 1000 cycles of the command, the command will report an error and bError will change from false to true.

◆ When the axis control mode changes from low level to high level (torque -> velocity, torque->position, velocity->position), the function block will calculate the set value of the high level mode. For example, when there is a change from torque mode to position mode, the function block will superimpose an expected position distance (calculated by the current actual velocity and the time offset in the task cycle) to compensate for the time lag between the actual and set values.

◆ When the bDone signal is triggered after the command has been executed, the axis will still run during the time between the command trigger and the bDone signal trigger. However, if the bDone signal is triggered and there is no other control command to continue setting values for the axis, the axis will stop immediately and an error will be reported, so the rising edge of the bDone signal should be used to trigger commands such as MC_Halt, MC_MoveVelocity or MC_MoveAbsolute to smooth out the axis.

4) Timing diagram



5) Error description

bExecute On rising edge.

Axis invalid

Axis state is invalid.

The axis does not satisfy the control mode.

Axis error is reported and Error is output.

【Note】: Please read "Appendix C Error Code Descriptions" for descriptions of the relevant error codes.

7.4.18 SMC_CheckLimits

The instruction function is to check whether the current drive setting value exceeds the maximum value configured by the controller.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ CheckLimits	Axis Limits Check		SMC_CheckLimits0(Axis:= , bEnable:= , bCheckVel:= , bCheckAccDec:= , bBusy=> , bError=> , iErrorID=> , bLimitsExceeded=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE, FALSE	FALSE	TRUE: In the execution of the check
bCheckVel	Speed check	BOOL	TRUE, FALSE	FALSE	TRUE: Speed check, false: Do not perform speed check
bCheckAccDec	Add a deceleration check	BOOL	TRUE, FALSE	FALSE	TRUE: Perform a deceleration check, false: Do not perform a deceleration check

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True - Perform axis check, False: Do not perform axis check
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
iErrorID	The error code	SMC_ERROR			Reference SMC_Error
bLimits Exceeded	Check Limit Output	BOOL	TRUE,FALSE	FALSE	TRUE: Currently set speed, or add or decelerate over Axis.fSWMaxVelocity, Axis.fSWMaxAcceleration

					Axis.fSWMaxDeceleration

3) Function Description

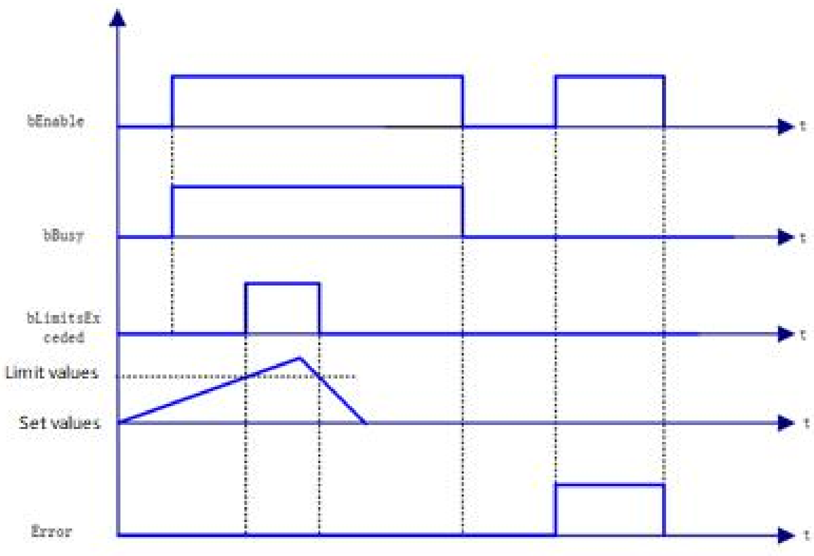
bEnable is TRUE, bBusy outputs TRUE. bEnable checks the axis velocity and acceleration.

If the set speed or acceleration/deceleration of the current axis exceeds the set values of Axis.fSWMaxVelocity, Axis.fSWMaxAcceleration, Axis.

fSWMaxDeceleration, the bLimitsExceeded signal is output as TRUE

Note: This function only checks that the current command speed or acceleration/deceleration exceeds the set limit value, it does not stop the axis.

4) Timing diagram



5) Error description

bExecute on rising edge.

Axis error reported, Error output.

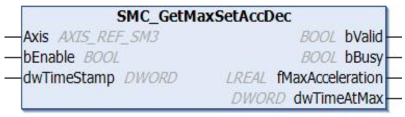
Invalid axis input, Error output.

[Note]: Please read "Appendix C Error Code Descriptions" for the relevant error code descriptions.

7.4.19 SMC_GetMaxSetAccDec

The command reads the maximum acceleration and deceleration of the axis.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ GetMaxSetAccDec	Maximum axis increase and deceleration		SMC_GetMaxSetAccDec_0(Axis:= , bEnable:= , dwTimeStamp:= , bValid=> , bBusy=> , fMaxAcceleration=> , dwTimeAtMax=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Map to the axis, AXIS_REF_SM3 instance of the map

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	TRUE: Perform a read
dwTimeStamp		Dword			Optional timestamp input;

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bValid	Effective	BOOL	TRUE, FALSE	FALSE	True, the instruction execution is valid
bBusy	In action	BOOL	TRUE, FALSE	FALSE	True, reading data
fMaxAcceleration	Maximum plus-deceleration value	LREAL		0	Maximum plus-deceleration value (positive acceleration, negative deceleration, plus-deceleration absolute maximum value is final)
dwTimeAtMax	The maximum value corresponds to the timestamp	Dword		0	The dwTimeStamp value corresponding to the maximum acceleration (e.g. when the acceleration continues to increase, the value is updated with dwTimeStamp, the fMaxAcceleration value is also updated,

					and once the acceleration reaches the maximum, the fMaxAcceleration record maximum, and the dwTimeStamp corresponding to the maximum value is also recorded)

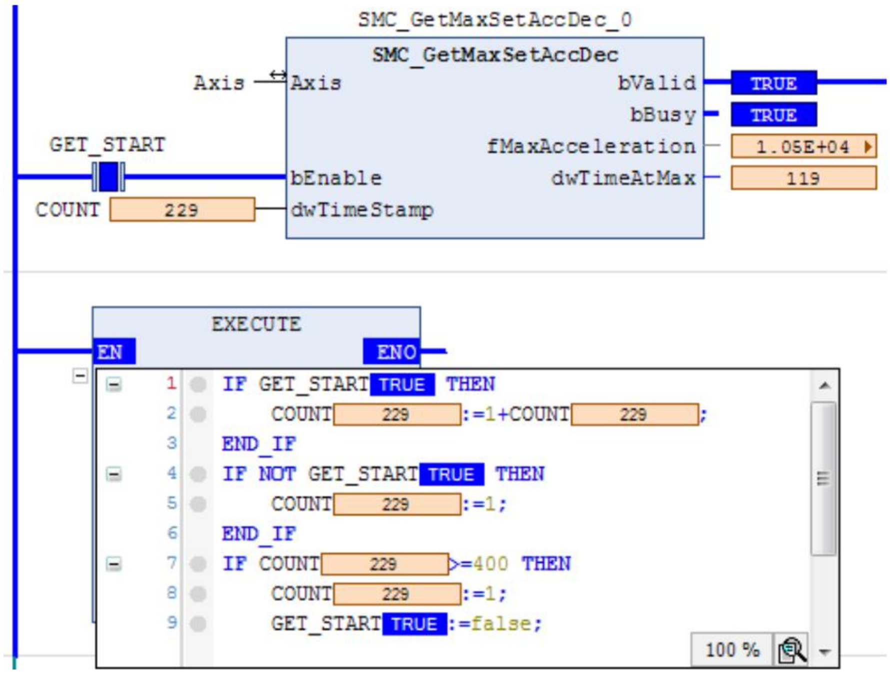
3) Function description

bEnable is TRUE, no error, bValid outputs TRUE.

When the absolute value of acceleration/deceleration is greater than the previous value, fMaxAcceleration and dwTimeAtMax will be refreshed.

The dwTimeAtMax value is the corresponding dwTimeStamp value for the maximum acceleration and deceleration, so dwTimeStamp should be set to a variable value, e.g. with the task period or a fixed time period. (see sample program)

◆ Sample program



7.4.20 SMC_GetMaxSetVelocity

The instruction function is: Read the maximum speed of the axis.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_GetMax SetVelocity	Maximum axis increase and deceleration		SMC_GetMaxSetVelocity_0(Axis:= , bEnable:= , dwTimeStamp:= , bValid=> , bBusy=> , fMaxVelocity=> , dwTimeAtMax=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	TRUE: Perform a read
dwTimeStamp		Dword			Optional time stamp input;

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bValid	Effective	BOOL	TRUE,FALSE	FALSE	True, the instruction execution is valid
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True, reading data
fMaxVelocity	The maximum speed value	LREAL		0	Maximum velocity value (positive, negative reverse, absolute maximum final)
dwTime AtMax	The maximum value corresponds to the	Dword		0	The dwTimeStamp value at maximum speed (e.g., when the speed continues to increase, the value is updated with dwTimeStamp, the fMaxVelocity value is also updated, and once the maximum

	timestamp				speed is reached, the fMaxVelocity record maximum, and the dwTimeStamp corresponding to the maximum value is also recorded)

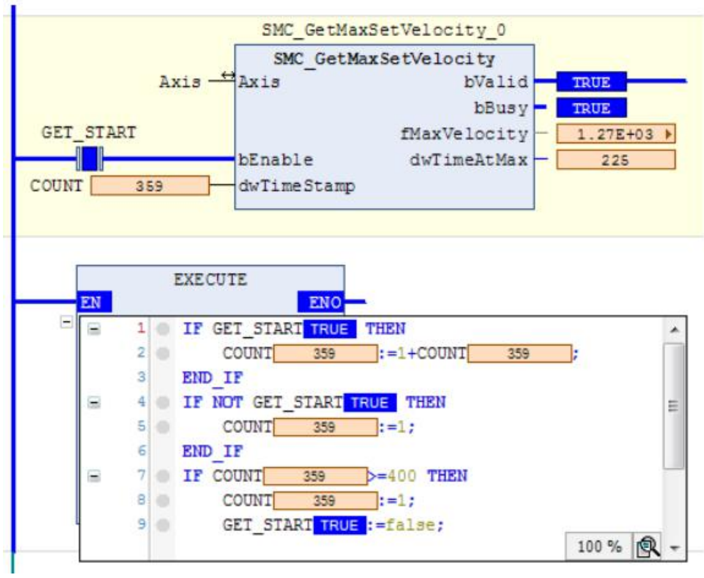
3) Function description

bEnable is TRUE, no error, bValid outputs TRUE.

fMaxVelocity and dwTimeAtMax are refreshed when the absolute value of the velocity is greater than the previous value.

The dwTimeAtMax value corresponds to the dwTimeStamp value for maximum velocity, so dwTimeStamp should be set to a variable value, e.g. with the task cycle or a fixed time period. (see sample program)

◆ Sample program



7.4.21 SMC_InPosition

The instruction function is to monitor the deviation between the current axis set position value and the actual value, and to determine whether the axis is within the required deviation range through the set deviation window.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_InPosition	Axis deviation monitoring		SMC_InPosition0(Axis:=Axis , bEnable:= , fPosWindow:= , fPosTime:= , fTimeOut:= , bInPosition=> , bBusy=> , bTimeOut=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE, FALSE	FALSE	TRUE: Perform a read
fPosWindow	Bias window	LREAL		0	Set the window for deviation monitoring, fPosWindow and Distance (the deviation between the instruction position and the feedback position), and the true position is based on the fPosTime time output bPositionInPosition
fPosTime	The trigger time	LREAL		0	The deviation is within the window range time and is used to trigger bInPosition units in S (seconds)
fPosTiOut	Time-out time	LREAL		0	Deviation timeout in S (seconds)

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bInPosition	The	BOOL	TRUE,FALSE	FALSE	True, the deviation is within the set

	deviation is normal				window
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
bTimeOut	Timeout	LREAL	TRUE,FALSE	FALSE	Current deviation detection related to byDeaTimeCycles values

3) Function Description

bEnable is TRUE, if the detected deviation is less than the set window fPosWindow and lasts fPosTime seconds then bInPosition triggers TRUE. bInPosition outputs FALSE as soon as the detected deviation is greater than the set window.

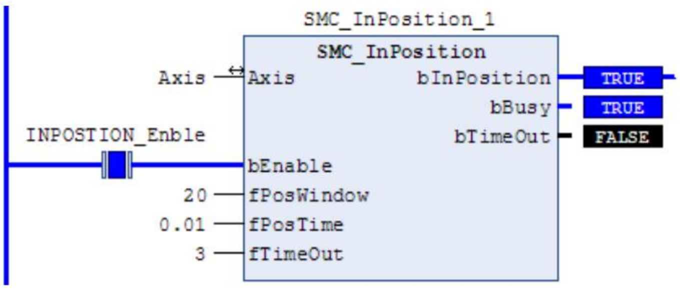
Note: The fPosTime must be set at a reasonable time otherwise bTimeOut will be triggered (e.g. for a cam with a 2 second cam period and a continuous deviation not exceeding the set window of 1.5 seconds, fPosTime set to greater than 1.5 seconds will cause bInPosition not to be triggered).

bEnable is TRUE, bBusy output is true.

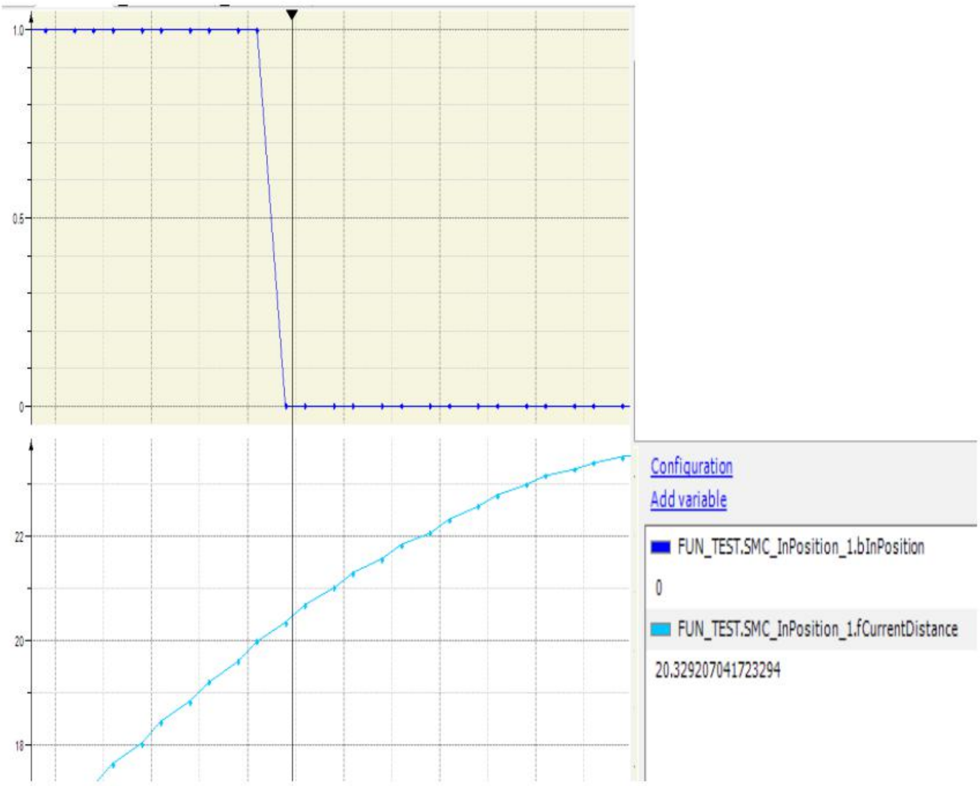
The deviation value monitors the data fCurrentDistance in the SMC_InPosition structure.

If bEnable is TRUE, bInPosition is not triggered TRUE after the time set by fPosTime, then bTimeOut is triggered TRUE.

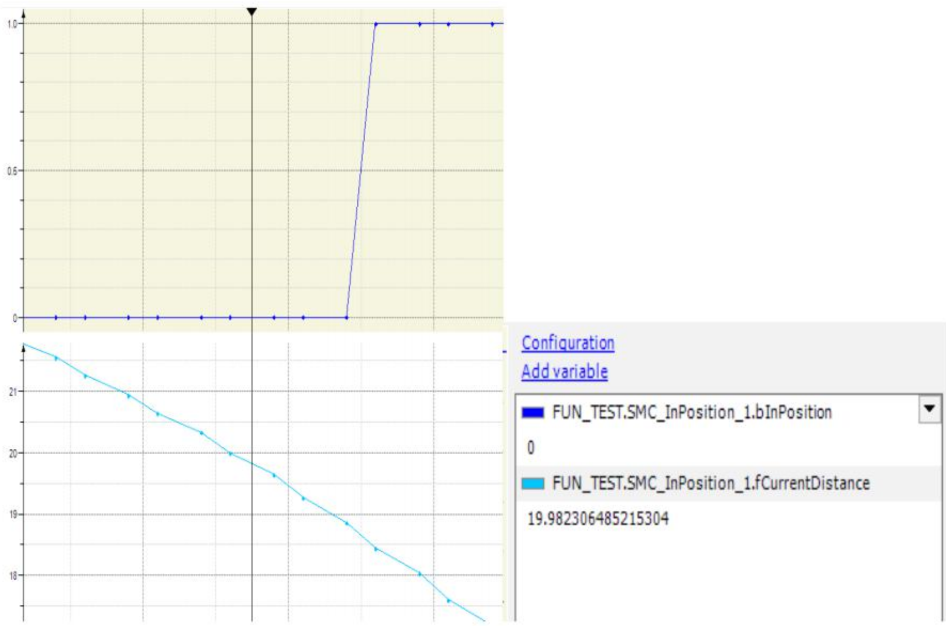
◆ Timing chart sample program



◆ Sample program

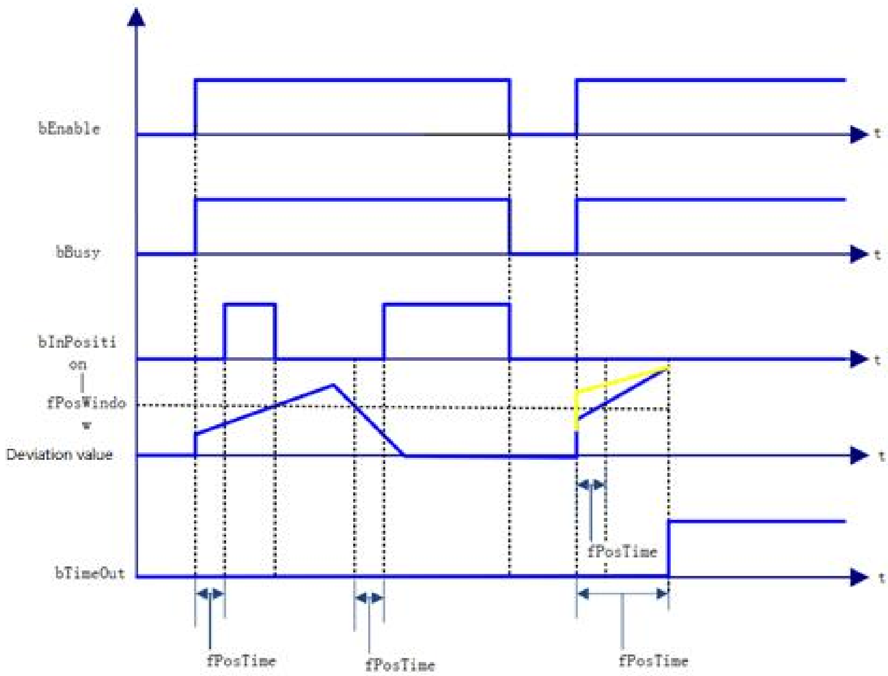


Larger than window setting bInPosition immediately changes from true to FALSE



bInPosition becomes TRUE after 4 task cycles (2.5ms) within the setting window, which corresponds to the program setting of 0.01S

4) Timing diagram


bEnable
bBusy
bInPosition
|
fPosWindow
Deviation value
bTimeOut
t
fPosTime
fPosTime
fPosTime
fPosTime

7.4.22 SMC_ReadSetPosition

The instruction function is to read the instruction position of the axis (the converted user unit).

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ ReadSetPosition	Read the axis command position	 	SMC_ReadSetPosition0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
Enable	Perform	BOOL	TRUE,FALSE	FALSE	TRUE: Perform a read

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
Valid	Effective	BOOL	TRUE,FALSE	FALSE	True, read valid
Busy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
Error	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
ErrorID	The error code	SMC_ERROR			Reference SMC_Error
Position	The position of the instruction	LREAL		0	The command position of the current task cycle

3) Function description

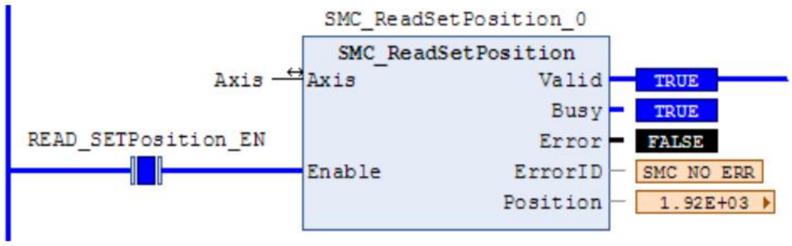
Enable is TRUE, Valid if no error, Busy output is TURE.

Position is the value of Axis.fSetPosition.

If Enable is FALSE, , then Valid and Busy output is FALSE. Position stays at the value before

FALSE.

◆ Timing chart sample program



4) Error description

bExecute on rising edge: Axis error, Error output; Invalid axis input, Error output.

[Note]: Please read "Appendix C Error Code Descriptions" for the relevant error code descriptions.

7.4.23 SMC_SetTorque

The instruction function is to set the shaft torque (valid when in torque control mode).

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_SetTorque	Torque setting		SMC_SetTorque0(Axis:= , bEnable:= , fTorque:= , bBusy=> , bError=> , nErrorID=>);

2) Related variables

Enter the output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Map to the axis, AXIS_REF_SM3 instance of the map

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	Rise the edge and set the shaft torque
fTorque	Set the torque	LREAL		0	The unit is 0.1

The output variable

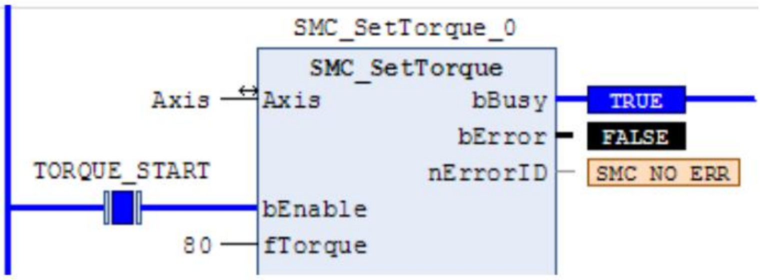
The output variable	Name	The data type	Effective range	The initial value	Describe
Busy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
Error	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
ErrorID	The error code	SMC_ERROR			Reference SMC_Error

3) Function description

bEnable Rising edge, no error then bBusy output is TURE.

This instruction only sets the torque value for the axis and is not a torque control function. The axis control mode is valid in the torque control mode, i.e. you need to use the SMC_SetControllerMode instruction to set the servo to torque mode first and then execute this instruction.

◆ Timing diagram sample program



4) Error description

bExecute on rising edge.

Axis error reported, Error output; invalid axis input, Error output.

Axis control mode error, Error output, error code SMC_ST_WRONG_CONTROLLER_MODE

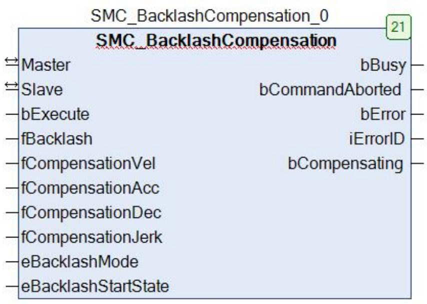
[Note]: Please read "Appendix C Error Code Descriptions" for descriptions of the relevant error codes.

7.4.24 SMC_BacklashCompensation

The instruction function is: to compensate for the main shaft gap, for example, the virtual axis in the belt transfer is the main axis, the axis is a virtual axis synchronization mirror, due to external reasons, there is a gap between the position of the shaft and the spindle, the instruction can be used to compensate for this gap.

This instruction function is similar to the phase offset instruction (MC_Phasing) and its phase depends on the direction in which the spindle operates.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_BacklashCompensation	Gap compensation		SMC_BacklashCompensation0(Master:= , Slave:= , bExecute:= , fBacklash:= , fCompensationVel:= , fCompensationAcc:= , fCompensationDec:= , eBacklashMode:= , eBacklashStartState:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=> , bCompensating=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Master	Spindle	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
Slave	From the axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE, FALSE	FALSE	Rise edge, set offset
fBacklash	LREAL	0			Compensate for the gap
fCompensationVel	LREAL	0			The speed at which compensation is made
fCompensationAcc	LREAL	0			Acceleration at compensation
fCompensationD	LREAL	0			Reduce the speed when

ec					compensating
eBacklashMode	SMC_ BACKLASH _MODE	SMC_BL _AUTO			Compensation mode: SMC_BL_AUTO: Spindle direction determines compensation direction SMC_BL_POSITIVE: Forward compensation, independent of spindle direction SMC_BL_NEGATIVE: Reverse compensation, independent of spindle direction SMC_BL_OFF: No compensation
eBacklash StartState	SMC_ BACKLASH_ STARTSTATE	SMC_BL_ START_ NEGATIVE			Describes the working state of the axis at which the instruction works. SMC_BL_START_NEGATIVE: The motion from the axis is pulled in the negative direction and does not need compensation in the negative direction, once the forward motion is called twice the fBacklash to establish compensation SMC_BL_START_POSITIVE: the forward motion is pulled in the positive direction, no compensation is required in the positive direction, and once the reverse motion needs to be compensated by twice the fBacklash SMC_BL_START_NONE: the distance compensation of the fBacklash value is generated in the positive or opposite direction movement.

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
bCommandAborted	The instruction is	BOOL	TRUE,FALSE	FALSE	True - interrupted by other control

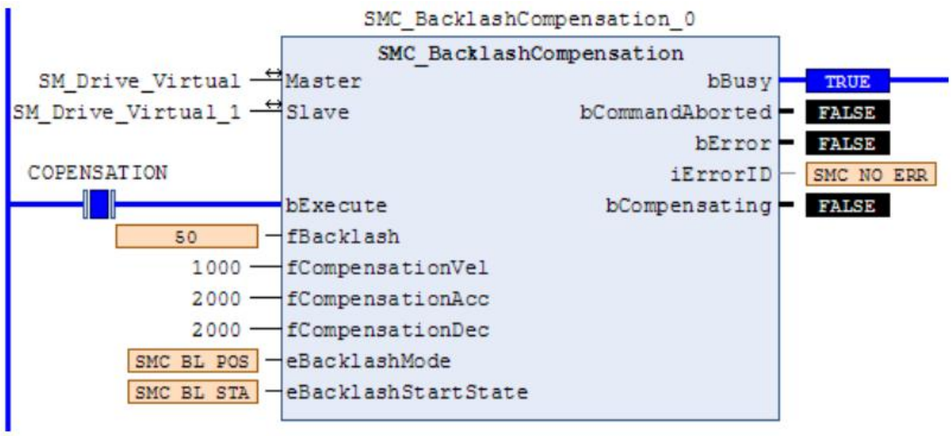
	interrupted				commands
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
iErrorID	The error code	SMC_ERROR	Reference SMC_Error		
bCompsating	compensation	BOOL	TRUE,FALSE	FALSE	

3) Function Description

bEecute rising edge, no error, bBusy output is TURE, bCompsating output is true, bCompsating output is false when compensation is complete.

The mode of operation is: eBacklashMode - the compensation direction is "positive", eBacklashStartState is "positive", fBacklash is positive. Before the bBusy signal comes, it is better that the master and slave axes are in the same position, otherwise the slave axes will be adjusted to spindle phase synchronization after the bEecute rising edge comes, and if the bBusy signal is already present then the bEecute rising edge is refreshed, please observe.

◆ Timing diagram sample procedure



◆ Sample program

4) Error description

bExecute on rising edge.

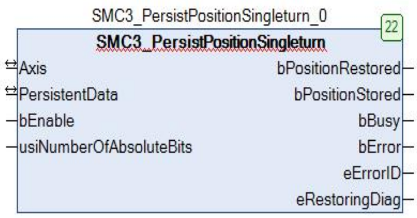
Axis error, Error output; Invalid axis input, Error output.

[Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.

7.4.25 SMC3_PersistPositionSingleturn

This instruction is used to maintain the position of the recorded solid-axis single-turn absolute value encoder (after the power-off restarts the controller, the pre-power-off position record value is restored).

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC3_ PersistPosition Singleturn	The axis position is maintained		SMC3_PersistPositionSingleturn_0(Axis:= , PersistentData:= , bEnable:= , usiNumberOfAbsoluteBits:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property
PersistentData	Keep the data	SMC3_ PersistPosition Singletrun_Data			A map to a recorded location structure SMC3_ PersistPosition_Data of a structure variable that must be power-off hold

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE, FALSE	FALSE	True function block execution, false does not perform function block PLC restart after the need for true to restore the pre-restart storage location.
usiNumberof AbsoluteBites	The number of digits	UINT		16	How many bits of absolute value encoder (e.g. 20 bits, 24 bit encoder, etc.)

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bPositionRestored	Location	BOOL	TRUE,	FALSE	TRUE, position recovery

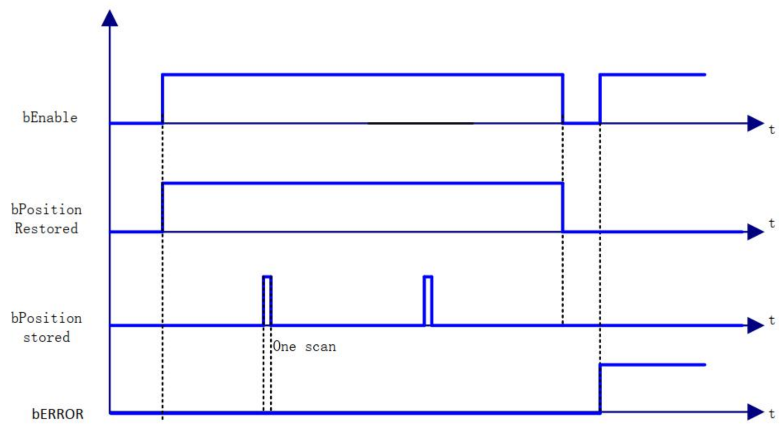
	recovery		FALSE		completes after axis restart
bPositionStored	Location save	BOOL	TRUE, FALSE	FALSE	TRUE, the call function is done quickly after saving the location
bBusy	FB in action	BOOL	TRUE, FALSE	FALSE	TRUE, the function block is not executed
bError	Error	BOOL	TRUE, FALSE	FALSE	TRUE, an exception occurs
eErrorID	The error code	SMC_ERROR		SMC_NO_ ERROR	When an exception occurs, the error code is output
eRestoringDiag	Restore diagnostics	SMC3_PersistPositionDiag		SMC3_Persist PositionDiag. SMC3_PPD_ RESTORING_OK	Diagnostic information in location recovery

3) Function description

The PLC restart bEnable signal is TRUE, the bPositionRestroed output is TRUE.

The dummy axis is not supported to follow the logic axis.

◆ Timing diagram



4) Error description

An input axis that is a virtual axis or a logical axis will result in an error output.

There is an error in the axis.

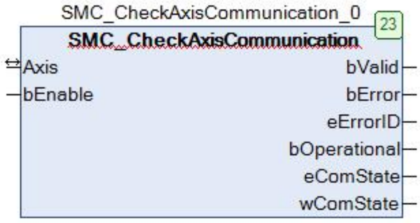
[Note]: Please read "Appendix C Error Code Descriptions" for error code descriptions.

7.4.26 SMC_CheckAxisCommunication

The instruction function is to check the current drive traffic status.

1) Instruction format

Instructions	Name	Graphical performance	ST performance

SMC_ CheckLimits	Axis limit check		`SMC_CheckAxisCommunication0(`
			`Axis:= ,`
			`bEnable:= ,`
			`bValid=> ,`
			`bError=> ,`
			`eErrorID=> ,`
			`bOperational=> ,`
			`eComState=> ,`
			`wComState=> );`

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	TRUE: In the execution of the check

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bValid	In action	BOOL	TRUE, FALSE	FALSE	True, the instruction execution is valid
bError	Error	BOOL	TRUE, FALSE	FALSE	True, exceptions are generated
eErrorID	The error code	SMC_ERROR			Reference SMC_Error
bOperational	Communication is normal	BOOL	TRUE, FALSE	FALSE	True, communication is normal (code 100) Operation False, communication is not normal, not axis operation
eComState	The communication status	SMC_COMMUNICATIONSTATE			Contains: SMC_COMSTATE_NOT_STARTED, communication does not start SMC_COMSTATE_VARIABLE_INITIALIZATION, communication variable

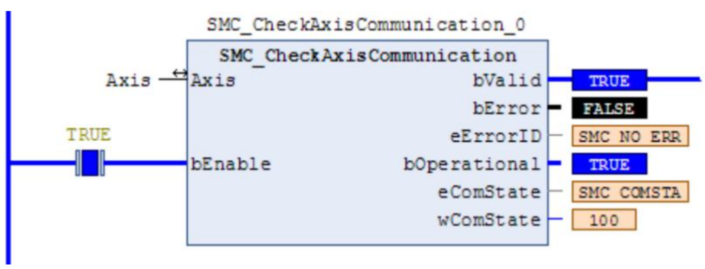
					initialization SMC_COMSTATE_BASE_COM_INITIALIZATION, basic port initialization SMC_COMSTATE_DRIVE_INITIALIZATION, communication driver initialization SMC_COMSTATE_DRIVE_WAITING_FOR_SYNC, synchronous warning SMC_COMSTATE_INITIALIZATION_DONE, initialization complete SMC_COMSTATE_OPERATIONAL, communication can be used normally SMC_COMSTATE_REINITIALIZATION, communication re-initialization SMC_COMSTATE_ERROR, communication errors SMC_COMSTATE_UNKNOWN the communication status is unknown
wComState	The communication code	WORD			Same as the axis structure variable in the input and output: Axis .wCommunicationState value, the code that represents the current communication state, refer to AXIS_REF_SM3 reference 1013

3) Function description

bEnable is TRUE, no error, bValid is TRUE.

When the bValid output is TRUE, the axis communication status is checked and the bOperational output is TRUE when the eComState output is SMC_COMSTATE_OPERATIONAL.

◆ Sample program



4) Error description

bExecute on rising edge: axis error reported, Error output.

Invalid axis input, Error output.

Note]: Please read "Appendix C Error Code Descriptions" for the relevant error code descriptions.

7.4.27 SMC_FollowPosition

The instruction function is to set the position directly to the shaft without doing any checks. This instruction differs from MC_MoveAbsolute in that each task cycle gives the axis position command regardless of the axis's state after the up-edge model is executed. (The user can write the cam function using the instruction instead of using MC_CamIn, etc.).

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ FollowPosition	The axis position is given		SMC_FollowPosition_0(Axis:= , bExecute:= , fSetPosition:=SET_POSITION , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE,FALSE	FALSE	The rising edge executes the function block
fSetPosition	Set the position	LREAL		0	The position set by the axis

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True - In the execution of the instruction, the axis is synchronized and, like the cam MC_CamIn instruction runtime axis state, the bBusy state can be cleared with the MC_Camout instruction
bCommand Aborted	The instruction is	BOOL	TRUE,FALSE	FALSE	True - The axis is interrupted by other control commands

	interrupted				
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
iErrorID	The error code	SMC_ERROR			Reference SMC_Error

3) Function description

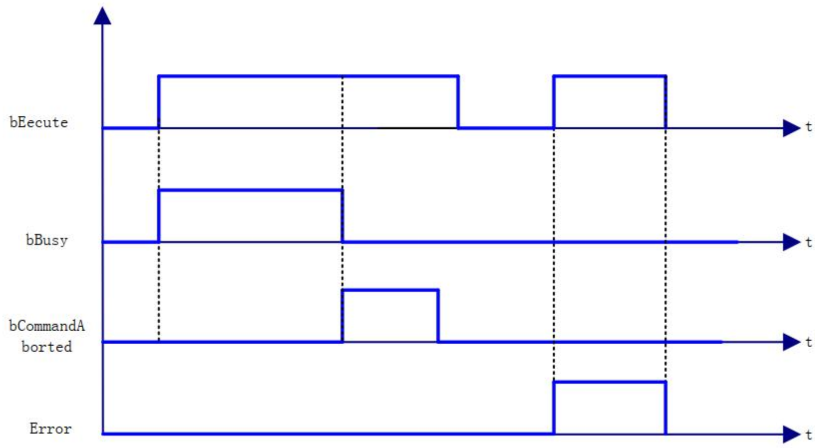
After SMC_FollowPosition has been started by the rising edge of bExecute, the axis sends position commands to the axis every task cycle. bBusy signal comes with the axis in the same synchronous state as the MC_CamIn instruction and can be cleared with the MC_CamOut instruction.

Axis velocity - calculated from the position increment of the difference between the two task cycles of the axis, velocity: ΔL / Δt , ΔL current task cycle

The difference between fSetVelocity and fSetVelocity of the previous task cycle, Δt is the scan time.

When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when another control command interrupts the instruction.

◆ Timing diagram



4) Error description

bExecute on rising edge.

Axis variable connected to a non-AXIS_REF_SM3 type structure variable, Error output.

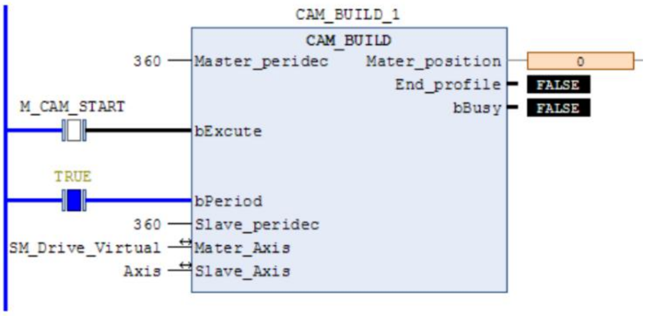
If the axis is not enabled, Error is output.

The instruction is running, the axis is wrong, Error is output.

Note]: Please read "Appendix C Error Code Descriptions" for the description of the relevant error codes.

5) Example

Use SMC_FollowPosition to implement the electronic cam function.



Function block variable definition section.

FUNCTION_BLOCK CAM_BUILD

VAR_INPUT// Input variable definition

Master_peridec:REAL; // master_cycle

bExcute:BOOL; // instruction execution

bPeriod:BOOL; // Cam period execution, false Single cycle execution

Slave_peridec:REAL; // Slave cycle

END_VAR

VAR_OUTPUT// Output variable definition

Mater_position:LREAL;// spindle position (spindle position calculated after the start of command execution)

End_profile:BOOL; // curve completion output flag bit

bBusy:BOOL; // Execution in progress

END_VAR

VAR// Function block intermediate variable definition

SMC_FollowPosition_0: SMC_FollowPosition;

SET_POSITION: LREAL;

SET_POSITIONOLD: LREAL;

Mater_positionOLD:LREAL;

bExcute_old:BOOL;

INC:LREAL;

Y:LREAL;

X5:LREAL;

X4:LREAL;

X3:LREAL;

X2:LREAL;

X1:LREAL;

MC_Stop0: MC_Stop;

STOP:BOOL;

COUNTNUM:DINT;

SET_INC:LREAL;

YOLD:LREAL;

SMC_FollowPositionVelocity_0: SMC_FollowPositionVelocity;

K:REAL;

K_OUT:REAL;

MC_CamOut_0: MC_CamOut;

```
VAR_IN_OUT// Input and output variable definitions
Mater_Axis:AXIS_REF_SM3;
Slave_Axis:AXIS_REF_SM3;
END_VAR
Program section.
IF bExcute AND NOT bExcute_old THEN // Rising edge initialization parameter
Mater_position:=0;
Mater_positionOLD:=Mater_Axis.fActPosition;
End_profile:=FALSE;
SET_POSITION:=Slave_Axis.fActPosition;
SET_POSITIONOLD:=Slave_Axis.fActPosition;
COUNTNUM:=0;
YOLD:=0;
K:=0;
ELSE

IF bExcute_old THEN
INC:= Mater_Axis.fActPosition-Mater_positionOLD;// Spindle task cycle increment
IF INC<0 THEN // Spindle encoding position past zero (when axis is set to modulo- modulo mode)
INC:= Mater_Axis.fActPosition-Mater_positionOLD+Mater_Axis.fPositionPeriod;
END_IF
Mater_position:=INC+Mater_position;// current spindle position
Mater_positionOLD:=Mater_Axis.fActPosition;
//******** judge curve completion *********//
IF Mater_position>=Master_peridec THEN
End_profile:=TRUE;
ELSE
End_profile:=FALSE;
END_IF
IF bPeriod THEN
IF Mater_position>=Master_peridec THEN
Mater_position:=Mater_position-Master_peridec;
END_IF
END_IF
END_IF
END_IF
IF bExcute_old THEN
X1:=(Mater_position/Master_peridec);
X2:=X1*X1;
X3:=X2*X1;
X4:=X3*X1;
X5:=X4*X1;
Y:=(6*X5-15*X4+10*X3)*Slave_peridec;// From axis position, curve
```

```
K:=(30*X4-60*X3+30*X2)*Slave_peridec/Master_peridec;// Slope of the curve
SET_INC:=Y-YOLD;
IF SET_INC<0 THEN
SET_INC:=Slave_peridec-YOLD+Y;
END_IF
YOLD:=Y;
IF bPeriod THEN
SET_POSITION:=SET_POSITION+SET_INC;
ELSE
IF End_profile THEN
SET_POSITION:=SET_POSITIONOLD+Slave_peridec;
ELSE
SET_POSITION:=SET_POSITION+SET_INC;
END_IF
END_IF
IF SET_POSITION>=Slave_Axis.fPositionPeriod THEN
SET_POSITION:=SET_POSITION-Slave_Axis.fPositionPeriod;
END_IF
END_IF
SMC_FollowPosition_0(
Axis:=Slave_Axis,
bExecute:=bExcute,
fSetPosition:=SET_POSITION ,
bBusy=>bBusy ,
bCommandAborted=> ,
bError=> ,
iErrorID=> );
IF NOT bExcute AND bExcute_old THEN
STOP:=TRUE;
END_IF
MC_CamOut_0(
Slave:=Slave_Axis,
Execute:= STOP,
Done=> ,
Busy=> ,
Error=> ,
ErrorID=> );
MC_Stop0(
Axis:=Slave_Axis,
Execute:= MC_CamOut_0.Done OR MC_CamOut_0.Error ,
Deceleration:=20000 ,
Jerk:= 20000,
Done=> ,
Busy=> ,
```

```
Error=> ,
ErrorID=> );
IF MC_Stop0.Done OR MC_Stop0.Error THEN
STOP:=FALSE;
END_IF
IF NOTbExcute_old THEN
End_profile:=FALSE;
END_IF
bExcute_old:=bExcute;
```

7.4.28 SMC_FollowPositionVelocity

The instruction function is the SMC_FollowPosition the same as the use function, but the speed setting is increased. Note: The speed setting to meet the position setting change is: the speed setting is set by setting a difference between the
task cycle position and a guide to the time. For example:
if the two inter-cycle positions are set consistently, the speed should be set to 0, otherwise the motor will vibrate violently.

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
SMC_ FollowPositionVelocity	The axis position and speed are given		SMC_FollowPositionVelocity_0(Axis:= , bExecute:= , fSetPosition:= , fSetVelocity:= , bBusy=> bBusy, bCommandAborted=> , bError=> , iErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE,FALSE	FALSE	The rising edge executes the function block
fSetPosition	Set the position	LREAL		0	The position set by the axis
fSetVelocity	Set the speed	LREAL		0	The position set by the axis

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True - In the execution of

					instructions, the bBusy state can be cleared with MC_Camout instructions when the axis is in sync, as is the status of the cam MC_CamIn instruction when it is run
bCommand Aborted	The instruction is interrupted	BOOL	TRUE,FALSE	FALSE	True - The axis is interrupted by other control commands
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
iErrorID	The error code	SMC_ ERROR			Reference SMC_Error

3) Function description

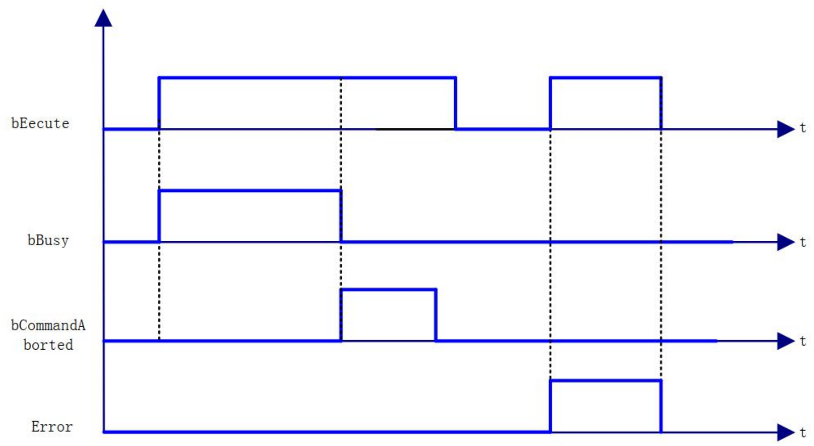
SMC_FollowPositionVelocity After starting by the rising edge of bExecute, the axis will send set position and set velocity commands to the axis every task cycle.

When the bBusy signal comes in, the axis is in the same state as when the MC_CamIn instruction is in effect and can be cleared with the MC_CamOut instruction.

The set speed of the axis must be the same as the set position: fSetVelocity= $\Delta L / \Delta t$, ΔL is the difference between the fSetVelocity of the current task cycle and the fSetVelocity of the previous task cycle, Δt is the scan time.

When the bExecute signal is TRUE, bBusy changes from TRUE to FALSE when another control command interrupts the instruction.

◆ Timing diagram



4) Error description

bExecute on rising edge.

Axis variable connected to a non-AXIS_REF_SM3 type structure variable, Error output.

Axis is not enabled, Error is output.

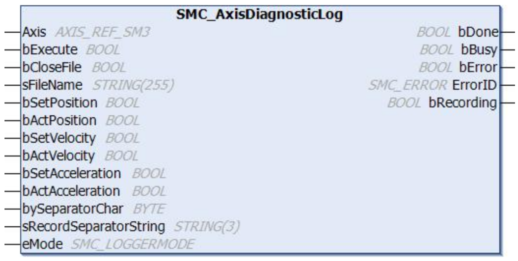
If the instruction is running and the axis is wrong, the error is output.

[Note]: Please read "Appendix C Error Code Descriptions" for related error code descriptions.

7.4.29 SMC_AxisDiagnosticLog

The instruction function is to periodically write a parameter of the axis to the file.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ AxisDiagnosticLog	Axis parameters are written to the file		SMC_AxisDiagnosticLog(Axis:= , bExecute:= , bCloseFile:= , sFileName:= , bSetPosition:= , bActPosition:= , bSetVelocity:= , bActVelocity:= , bSetAcceleration:= , bActAcceleration:= , bySeparatorChar:= , sRecordSeparatorString:= , eMode:= , bDone=> , bBusy=> , bError=> , ErrorID=> , bRecording=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE,FALSE	FALSE	Up the edge, perform the function block
bClosefile	Close the file	BOOL	TRUE,FALSE	FALSE	TRUE, the instruction immediately closes the file
sFileName	Filename	STRING(80)		‘’	The stored file name (before the path.)
bSetPosition	Record the set position	BOOL	TRUE,FALSE	FALSE	TRUE, record the set position when executing the instruction
bActPosition	Record the actual location	BOOL	TRUE,FALSE	FALSE	TRUE, record the actual location when executing the instruction
bSetVelocity	Record the set speed	BOOL	TRUE,FALSE	FALSE	TRUE, record the set speed when executing the instruction

bActVelocity	The actual speed of several rounds	BOOL	TRUE,FALSE	FALSE	TRUE, the actual speed is recorded when the instruction is executed
bSetAcceleration	Record the set acceleration	BOOL	TRUE,FALSE	FALSE	TRUE, record the set acceleration when executing the instruction
bActAcceleration	Record the actual acceleration	BOOL	TRUE,FALSE	FALSE	TRUE, record the actual acceleration when executing the instruction
bySeparatorChar		BYTE		9	ASCII code value, written between two different values
sRecord SeparatorString				'RN'	The string written at the end of the date
eMode		SMC_ LOGGERMODE	LOG_ CONTINUOUS		log_continuous: Record to file continuously log_at_close: Record continuously to the buffer (10kbyte). When bclosefile is true, the buffer's data is written to the file

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	Complete	BOOL	TRUE,FALSE	FALSE	True, the save is complete
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
ErrorID	The error code	SMC_ERROR			Reference SMC_Error
bRecording	recorded	BOOL	TRUE,FALSE	FALSE	True, the argument is being saved in the record

3) Function description

This function block is used to write a set of parameter values belonging to an axis cyclically to a file. This output file is ideally suited for diagnostic purposes. As it usually takes some time to write data on the data media, this block stores the collected data in a buffer of 10kbyte size and the data is not written until the module action WriteToFile is called. To prevent interference with the actual action task and the action itself, this action call should be placed in a slower (~50 ms) lower priority task. Once the buffer has been exceeded, the module will create an error output.

4) Error description

bExecute on rising edge: axis error reported, Error output.

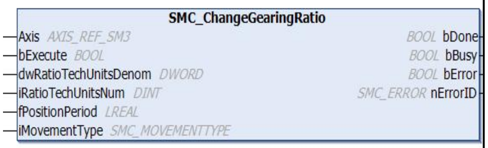
Invalid axis input, Error output.

Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.

7.4.30 SMC_ChangeGearingRatio

The instruction function is: to change the user-set electronic gear ratio (pulse-to-user unit ratio) and drive type. Note: The function block rear axle needs to be restarted SMC3_ReinitDrive to ensure that the variable

1) instruction format can be initialized correctly

Instructions	Name	Graphical performance	ST performance
SMC_ ChangeGearingRatio	Change the gear ratio		SMC_ChangeGearingRatio0(Axis:= , bExecute:= , dwRatioTechUnitsDenom:= , iRatioTechUnitsNum:= , fPositionPeriod:= , iMovementType:= , bDone=> , bBusy=> , bError=> , nErrorID=>);

2) Related variables

input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the . The gear ratio will be changed to the shaft

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	Perform	BOOL	TRUE,FALSE	FALSE	Up the edge, perform the function block
dwRatioTechUnitsDenom		DWORD		0	Pulse units converted to application units (eg:mm)
iRatioTechUnitsNum		DINT		0	The dwRatioTechUnitsDenom value corresponds to the desired application unit
fPositionPeriod		LREAL			Position cycle (mould value) is only valid for rotating motors
iMovementType		INT			0: modulo axis (module axis), 1: finite axis (limited long axis).

The output variable

The output	Name	The data type	Effective range	The initial value	Describe

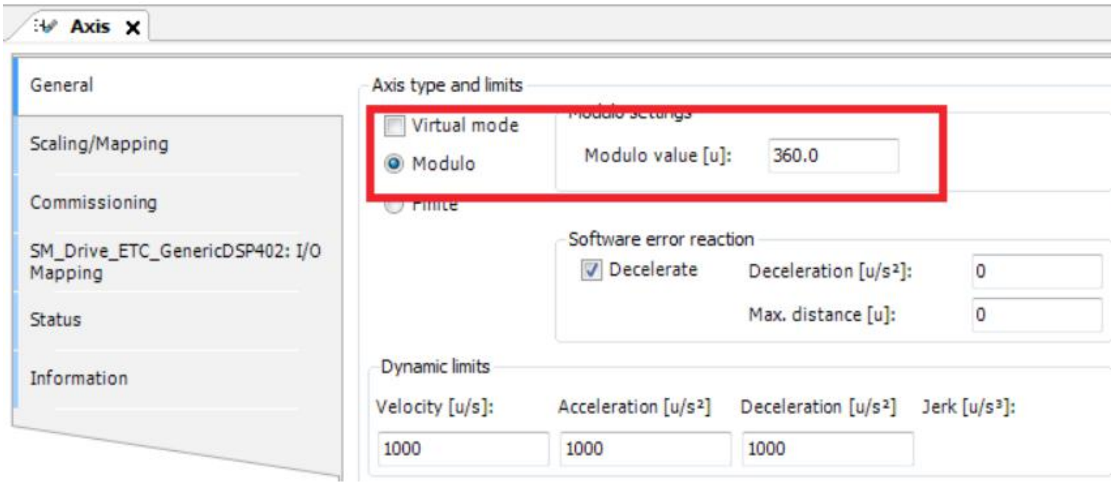
variable					
bDone	Complete	BOOL	TRUE,FALSE	FALSE	True, the execution set is complete
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
bError	Error	BOOL	TRUE,FALSE	FALSE	True, exceptions are generated
nErrorID	The error code	SMC_ERROR			Reference SMC_Error

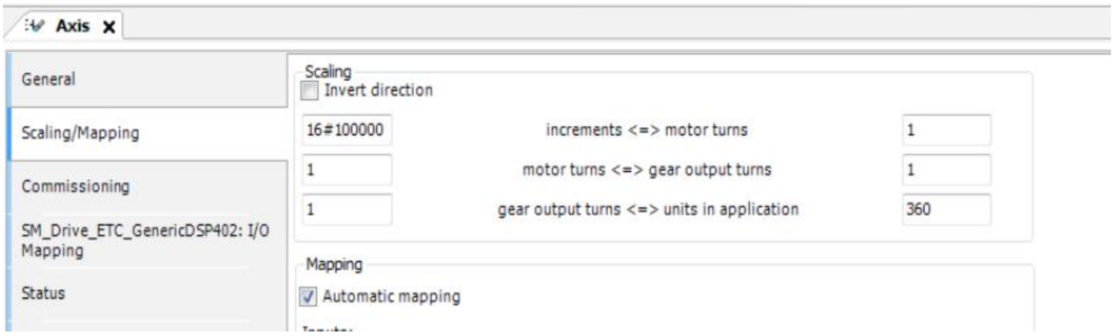
3) Function Description

bEecute rising edge, no error then, bBusy output is TURE, finish bDone output is true, bBusy output is false.

For example, if a 20-bit encoder servo motor with a 10:1 reduction ratio drives a screw (10mm pitch), the motor turns 10 revolutions and the screw moves 10mm, setting dwRatioTechUnitsDenom 1048576*10 and iRatioTechUnitsNum to 10.

The function block serves to dynamically modify for the program the parts shown in the following diagram:





4) Error description

bExecute On rising edge.

◆ The axis reports an error, Error output.

◆ Invalid input value, Error output , Error code SMC_CGR_ZERO_VALUES

◆ Axis in command-controlled operation, Error output , Error code SMC_CGR_DRIVE_POWERED

◆ The input modulus value is invalid (eg: <0), Error output , Error code

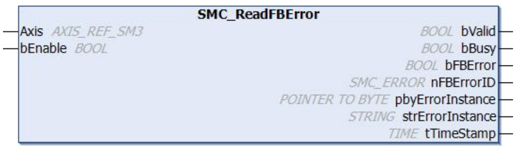
SMC_CGR_INVALID_POSPERIOD

Note]: Please read "Appendix C Error Code Descriptions" for the error code descriptions.

7.4.31 SMC_ReadFBError

The instruction function is: MC, SMC function block error .

1) Instruction format

Instructions	Name	Graphical performance	**ST** performance
SMC_ ReadFBError	Read function block error		SMC_ReadFBError(Axis:= , bEnable:= , bValid=> , bBusy=> , bFBError=> , nFBErrorID=> , pbyErrorInstance=> , strErrorInstance=> , tTimeStamp=>);

2) Related variables

. . . input and output variables

Enter the output variable	Name	The data type	Effective range	The initial value	Describe
Axis	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

Enter variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bEnable	Perform	BOOL	TRUE,FALSE	FALSE	TRUE: Perform a read

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bValid	Effective	BOOL	TRUE,FALSE	FALSE	True, read valid
bBusy	In action	BOOL	TRUE,FALSE	FALSE	True, in action
bFBError	Error	BOOL	TRUE,FALSE	FALSE	True, there is an FB error
nFBErrorID	The error code	SMC_ERROR			Reference SMC_Error
pbyErrorInstance		POINTER TO BYTE			The function block of the output point is misaled
strErrorInstance		STRING			Point to error function blocks (programs, sub-programs, function blocks)
tTimeStamp		TIME			The timestamp at which the error occurred

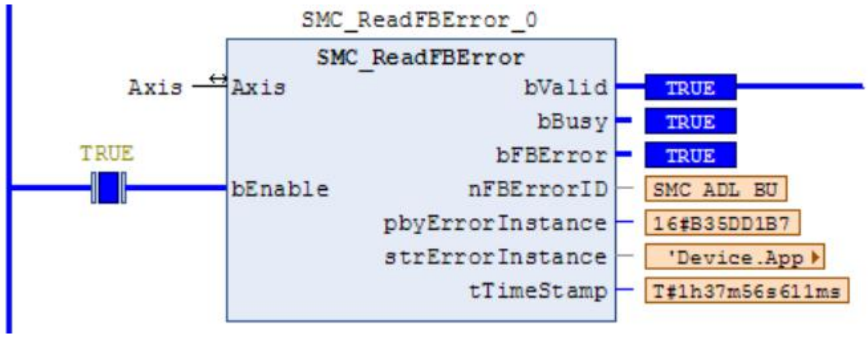
3) Function description

Enable is TRUE, no error is Valid, Busy is TURE.

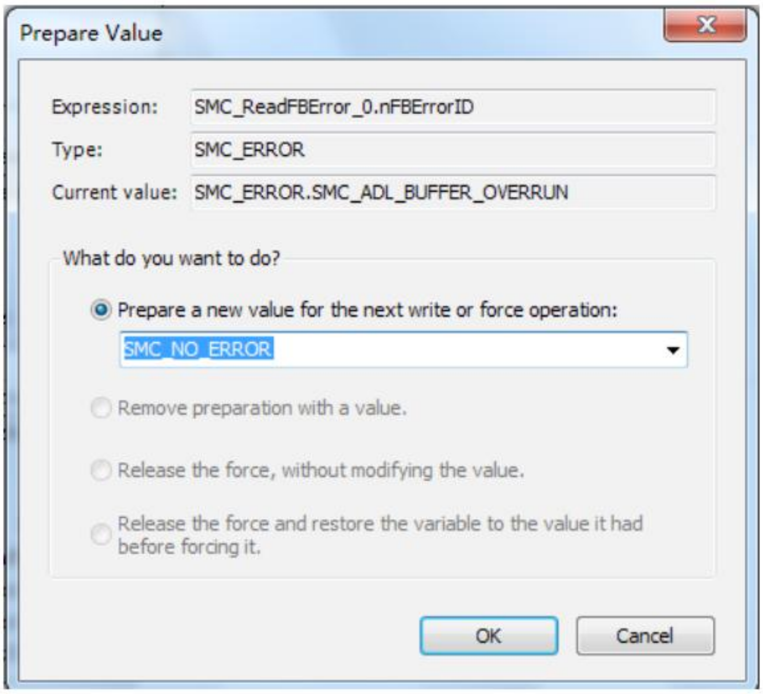
If there is a function block alarm, bFBError is true.

If Enable is FALSE, , then Valid, Busy output is FALSE.

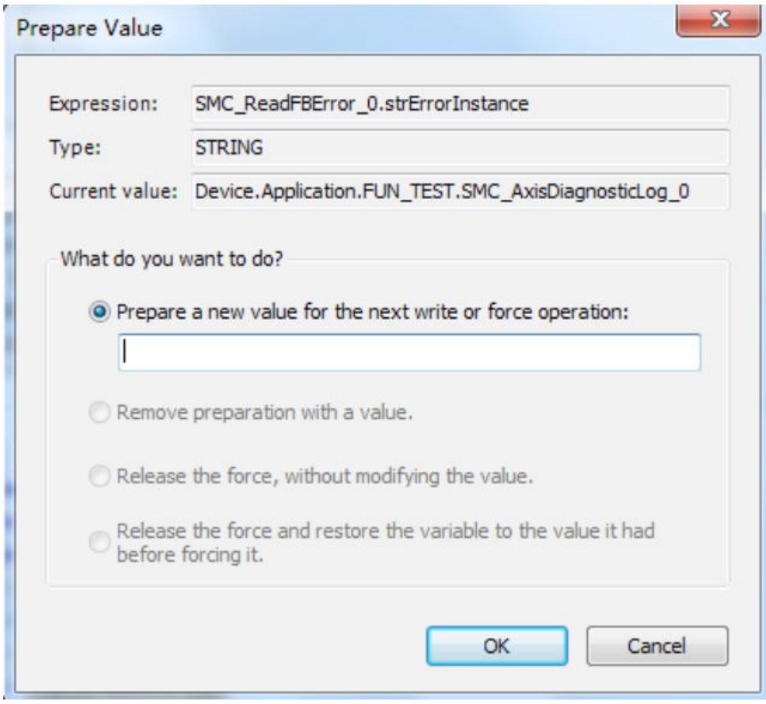
◆ Timing diagram sample program

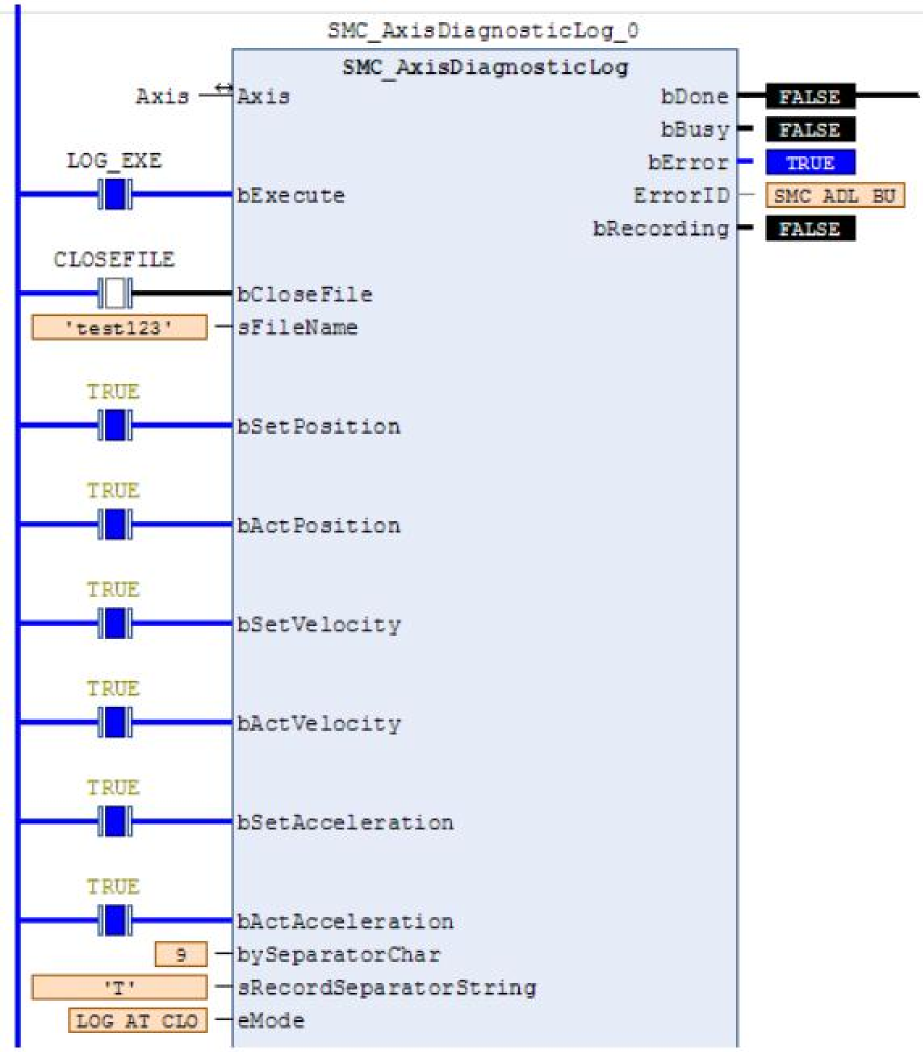


◆ Sample program



◆ Error ID





The function block where the error occurred

4) Error description

bExecute on rising edge.

Axis error reported, Error output.

Invalid axis input, Error output.

Note]: Please read "Appendix C Error Code Descriptions" for the relevant error code descriptions.

7.4.32 SMC_ClearFBError

The instruction function is to clear FB errors from the function block.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ ClearFBError	Clears the function block error		TEST:=SMC_ ClearFBError(pDrive:=ADR(Axis));

2) Related variables

. . . input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
pDrive	Axis	AXIS_REF	-	-	Maps to the axis, AXIS_REF_SM3 instance of the property

The output variable

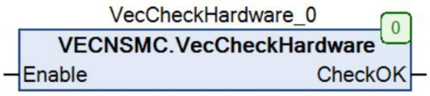
The output variable	Name	The data type	Effective range	The initial value	Describe
SMC_ClearFBError	Clear the error	BOOL	TRUE,FALSE	FALSE	True, clear

7.5 Vector special instructions

7.5.1 VECNSMC. VecCheckHardware

The instruction function is: Check that the controller hardware ID is correct. (The library NSMCLib needs to beinstalled.)

1) Instruction format

Instructions	Name	Graphical performance	ST performance
VECNSMC. VecCheckHardware	Hardware ID detection function block		VecCheckHardware(Enable:= , CheckOK=>);

2) Related variables

Enter variables

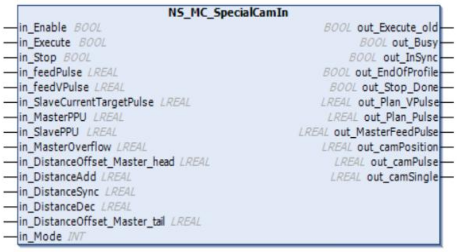
Enter the variable	Name	The data type	Effective range	The initial value	Describe
in_Enable	Enable function blocks	BOOL	TSTREET,FALSE	FLASE	Enable hardware ID to detect function blocks

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
CheckOK	The detection was successful	BOOL	TRUE,FALSE	FALSE	When hardwareI D is detected to be correct, it is placed asTRUE

7.5.2 VECNSMC.NS_MC_SpecialCamIn

The instruction function is: for establishing a special cam relationship between the two axes. (Library NSMCLib1) instruction format needs to be installed

<table>
<tr><th>Instructions</th><th>Name</th><th>Graphical performance</th><th>ST performance</th></tr>
<tr><td>VECNSMC.
NS_MC_
SpecialCamIn</td><td>Special
cam
function
block</td><td>
</td><td>NS_MC_SpecialCamIn(
in_Enable:= ,
in_Execute:= ,
in_Stop:= ,
in_feedPulse:= ,
in_feedVPulse:= ,
in_SlaveCurrentTargetPulse:= ,
in_MasterPPU:= ,
in_SlavePPU:= ,
in_MasterOverflow:= ,
in_DistanceOffset_Master_head:= ,
in_DistanceAdd:= ,
in_DistanceSync:= ,
in_DistanceDec:= ,
in_DistanceOffset_Master_tail:= ,
in_Mode:= ,
out_Execute_old=> ,
out_Busy=> ,
out_InSync=> ,
out_EndOfProfile=> ,
out_Stop_Done=> ,
out_Plan_VPulse=> ,
out_Plan_Pulse=> ,
out_MasterFeedPulse=> ,
out_camPosition=> ,
out_camPulse=> ,
out_camSingle=>);</td></tr>
</table>

2) Related variables

. . . input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
in_Enable	Enable function blocks	BOOL	TSTREET,FALSE	FLASE	Enable special cam function blocks
in_Execute	The execution condition	BOOL	TSTREET,FALSE	FLASE	An up-edge of the input will initiate the processing of the function block
in_Stop	Stop the cam	BOOL	TSTREET,FALSE	FLASE	One of the rising edges of the input will lift the cam and stop from the axis after the current cam cycle is complete
in_feedPulse	Spindle position	LREAL	The range of data	0	Associate the spindle and enter the given position of the spindle, which isf Setposition
in_feedVPulse	Spindle speed	LREAL	The range of data	0	Associate the spindle and enter the given speed of the spindle, whichis f Setvelocity
in_SlaveCurrentTargetPulse	From the axis target position	LREAL	The range of data	0	From the target position of the axis, that is, from the axis off Setposition

in_MasterPPU	Spindle electronic gear	LREAL	The range of data	0	Electronic gears for spindles (default 1)
in_SlavePPU	Electronic gear from shaft	LREAL	The range of data	0	Electronic gear from shaft (default setting 1)
in_MasterOverflow	The number of cam cycle spindle units	LREAL	The range of data	0	The module of the spindle is to be paired with this value in the user unit
in_DistanceOffset_Master_head	Spindle front offset	LREAL	The range of data	0	How many positions the spindle moved before the cam action was made from the axis
in_DistanceAdd	Accelerate the distance from the axis	LREAL	The range of data	0	The acceleration distance from the axis from rest to the synchronization zone
in_DistanceSync	Synchronize the distance from the axis	LREAL	The range of data	0	The synchronous running distance from the axis to the sync zone
in_DistanceDec	Slow down the distance from the shaft	LREAL	The range of data	0	The deceleration distance from the axis from the synchronization zone to rest
in_DistanceOffset_Master_tail	Spindle rear offset	LREAL	The range of data	0	How many more positional offsets do the spindles take after the cam action is made from the axis?
in_Mode	Mode	INT	The range of data	0	Working mode

Note:

(1) Modulus is used for both master and slave axis types in the cam module.

(2) No other motion control can be performed on the slave axes bound in the cam function block.

(3) When modulus is used for both master and slave axes, the slave axis modulus = in_DistanceAdd+ in_DistanceSync+ in_DistanceDec and the master axis modulus = in_DistanceAdd*30/16+ in_DistanceSync+ in_DistanceDec*30/16;

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
out_Execute_old	Execute signal output	BOOL	TRUE,FALSE	FALSE	The function block outputs TRUE when it receives the Ex ecute signal
out_Busy	The instruction	BOOL	TRUE,FALSE	FALSE	The current instruction is

	is being executed				in execution and is set to TRUE
out_InSync	Reach a uniform speed from the shaft	BOOL	TRUE,FALSE	FALSE	The constant speed is reached from the shaft and is set to TRUE
out_EndOfProfile	The cam action completes the signal	BOOL	TRUE,FALSE	FALSE	After each cam action is completed, it is set to TRUE
out_Stop_Done	Stop completing the signal	BOOL	TRUE,FALSE	FALSE	When the stop is complete, set to TRUE
out_Plan_VPulse	Speed from the axis	LREAL	The range of data	0	The speed from the axis, in userunits/S
out_Plan_Pulse	From the axis position	LREAL	The range of data	0	Encoder position in pulses
out_MasterFeedPulse	Feedback on the number of cam cycle spindle units	LREAL	The range of data	0	Feedback on the position of the spindle once cam cycle walk (in user units)
out_camPosition	Feedback cam cycle from the number of shaft units	LREAL	The range of data	0	The position (in user units) where the feedback 1 cam cycle d'a walks fromthe axis;
out_camPulse	The number of pulses from the shaft for the feedback cam cycle	LREAL	The range of data	0	Feedback on the position of one cam cycle walk from the shaft (in pulses)
out_camSingle	Cam output	LREAL	The range of data	0	Used to determine the position relationship between the cam and the cut point, the initial position is 0 before the shaft passes through the cutpoint, and after the cut point the cut point is0 (units: units).

◆ Example: The imaginary axis (Axis_Master) is used as the main axis with the real axis (Axis_Slave) running cam movement：

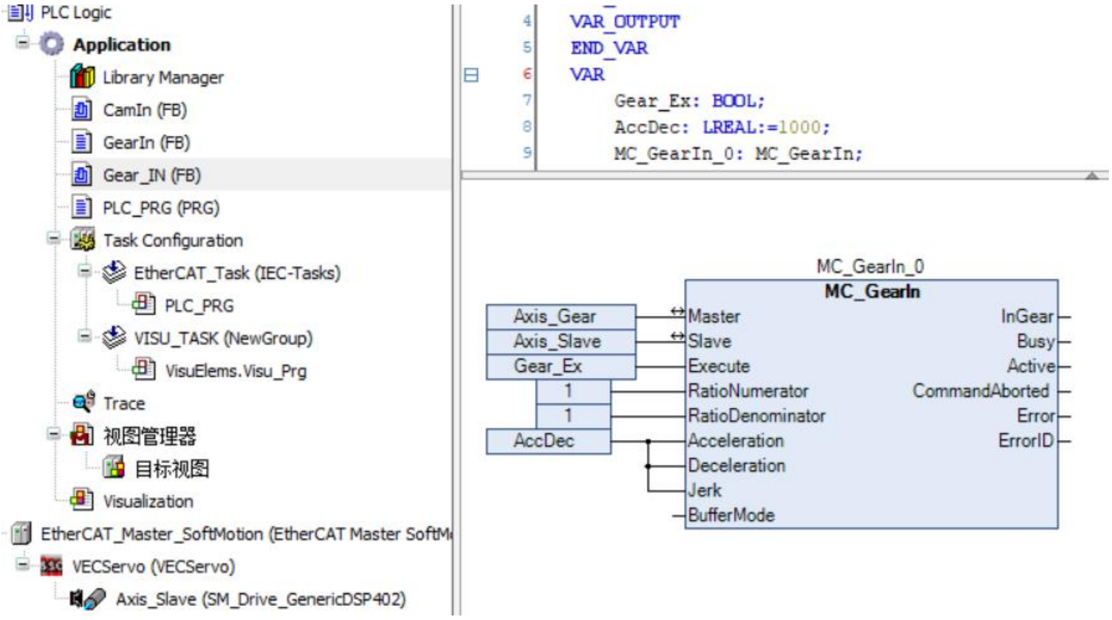
Cam spindle: Axis_Master (dummy axis).

Cam slave and gear spindle: Axis_Gear (dummy axis).

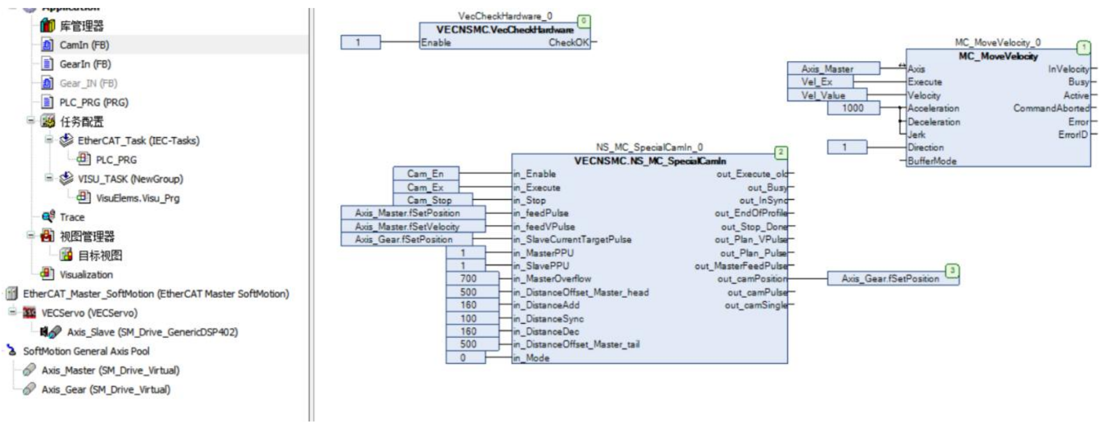
Gear slave: Axis_Slave (real axis)

Note：

(1) The special cam function block must be used in conjunction with the VecCheckHardware function block. The special cam function block can only be used when the CheckOK output of the VecCheckHardware function block is TRUE.
(2) Since we cannot assign a value to the fSetPosition of the real axis directly, but can assign a value to the fSetPosition of the imaginary axis, we need to use the MC_GearIn function block to establish the gear relationship, so that the imaginary axis Axis_Gear is the main axis of the gear and the real axis Axis_Slave is the slave of the gear; as follows.

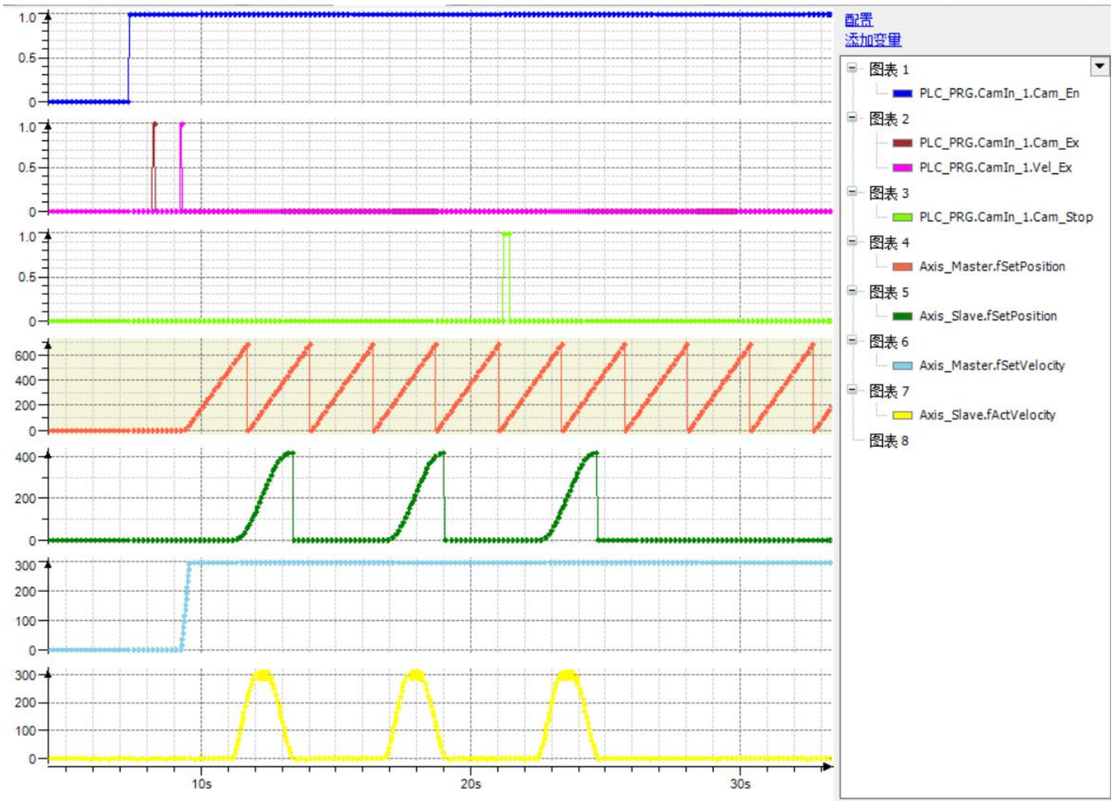


The cam function is then used as follows: The main axis Axis_Master and the imaginary axis Axis_Gear are then allowed to establish a cam relationship so that the real axis Axis_Slave can be cammed with the imaginary axis Axis_Gear via the gear.
The cam function is used by first enabling (in_Enable) the cam module and then triggering (in_Execute) the cam
module



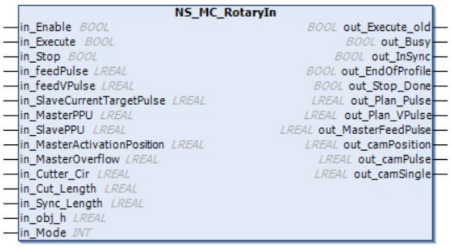
◆Timing diagram
The following diagram shows the timing diagram of the routine. When using the cam function block, you need to set in_Enable to TRUE first, then give in_Execute a rising edge trigger, and then trigger the spindle speed to carry out the cam action (triggering the spindle

speed first and then triggering in_Execute of the cam module is also feasible); when you need to stop the cam, trigger in_Stop on the rising edge, and the slave cam action will stop after completing the current cam cycle.;



7.5.3 VECNSMC.NS_MC_RotaryIn

The instruction function is: used to establish a wheel-cutting relationship between the two axes. (Library NSMCLib1) instruction format needs to be installed

Instructions	Name	Graphical performance	ST performance
VECNSMC. NS_MC_RotaryIn	Wheel-cut function block		NS_MC_RotaryIn(in_Enable:= , in_Execute:= , in_Stop:= , in_feedPulse:= , in_feedVPulse:= , in_SlaveCurrentTargetPulse:= , in_MasterPPU:= , in_SlavePPU:= , in_MasterActivationPosition:= , in_MasterOverflow:= , in_Cutter_Cir:= , in_Cut_Length:= , in_Sync_Length:= , in_obj_h:= , in_Mode:= , out_Execute_old=> , out_Busy=> , out_InSync=> , out_EndOfProfile=> , out_Stop_Done=> , out_Plan_Pulse=> , out_Plan_VPulse=> , out_MasterFeedPulse=> , out_camPosition=> , out_camPulse=> , out_camSingle=>);

2) Related variables

. . . input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
in_Enable	Enable function blocks	BOOL	TSTREET,FALSE	FLASE	Enable special cam function blocks
in_Execute	The execution condition	BOOL	TSTREET,FALSE	FLASE	An up-edge of the input will initiate the processing of the function block
in_Stop	Stop the round cut	BOOL	TSTREET,FALSE	FLASE	One of the rising edges of the input will lift the cam and stop from the shaft at 180 degrees at the cut point
in_feedPulse	Spindle position	LREAL	The range of data	0	Associate the spindle and enter the actual position of the spindle, which isf Setposition
in_feedVPulse	Spindle speed	LREAL	The range of data	0	Associate the spindle and enter the given speed of the spindle, whichis f Setvelocity
in_SlaveCurrentTargetPulse	From the axis target position	LREAL	The range of data	0	From the target position of the axis, that is, from the axis off Setposition

in_MasterPPU	Spindle electronic gear	LREAL	The range of data	0	The electronic gear of the spindle, write 1 by default
in_SlavePPU	Electronic gear from shaft	LREAL	The range of data	0	From the electronic gear of the shaft, write 1 by default
in_MasterActivationPosition	The spindle start distance	LREAL	The range of data	0	The first time the knife to the cut point, the spindle walks the distance, Mode 1 is available (requires more than half of the knife's perelong)
in_MasterOverflow	The number of spindle units for the wheel-cutting cycle	LREAL	The range of data	0	The module of the spindle is to be paired with this value in the user unit
in_Cutter_Cir	Knife per per se	LREAL	The range of data	0	Knife perigle (corresponding to the die from the shaft)
in_Cut_Length	Cut long	LREAL	The range of data	0	Cut length (corresponding to the die of the spindle)
in_Sync_Length	Synchronize the distance from the axis	LREAL	The range of data	0	The synchronization distance from the axle cut action
in_obj_h	The thickness of the material	LREAL	The range of data	0	Material thickness, using the initial value by default
in_Mode	Mode	INT	The range of data	0，1	Working mode

Caution:

(1) Modulus is used for both master and slave axis types in the wheel-cutting block.

(2) No other motion control can be performed on the slave axes bound in the wheel-cutting function block.

(3) When modulus is used for both master and slave axes, slave modulus = in_Cutter_Cir, master modulus = in_Cut_Length；

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
out_Execute_old	Execute is a valid output	BOOL	TRUE,FALSE	FALSE	The function block outputs TRUE when it receives the Ex ecute signal
out_Busy	The instruction	BOOL	TRUE,FALSE	FALSE	The current

	is being executed				instruction is in execution and is set to TRUE
out_InSync	Reach a uniform speed from the shaft	BOOL	TRUE,FALSE	FALSE	The constant speed is reached from the shaft and is set to TRUE
out_EndOfProfile	The cam action completes the signal	BOOL	TRUE,FALSE	FALSE	After each cam action is completed, it is set to TRUE
out_Stop_Done	Stop completing the signal	BOOL	TRUE,FALSE	FALSE	When the stop is complete, set to TRUE
out_Plan_VPulse	Speed from the axis	LREAL	The range of data	0	The speed from the axis, in userunits/S
out_Plan_Pulse	From the axis position	LREAL	The range of data	0	Encoder position in pulses
out_MasterFeedPulse	Feedback on the number of cam cycle spindle units	LREAL	The range of data	0	Feedback on the position of the spindle's one-wheel cut cycle walk (units)
out_camPosition	Feedback cam cycle from the number of shaft units	LREAL	The range of data	0	Feedback is provided on the position (units) that go from the axis to the wheel-cutting cycle at onetime;
out_camPulse	The number of pulses from the shaft for the feedback cam cycle	LREAL	The range of data	0	Feedback on the position of one wheel-cutting cycle from the axis (in pulses)
out_camSingle	Cam output	LREAL	The range of data	0	Used to determine the position relationship between the cam and the cut point, the initial position is 0 before the shaft passes through the cutpoint, and after the cut point the cut point is0 (units:

					units).

◆ Example procedure: Imaginary axis for spindle with real axis slave running wheel tangent motion:

Cam spindle: Axis_Master (dummy axis).

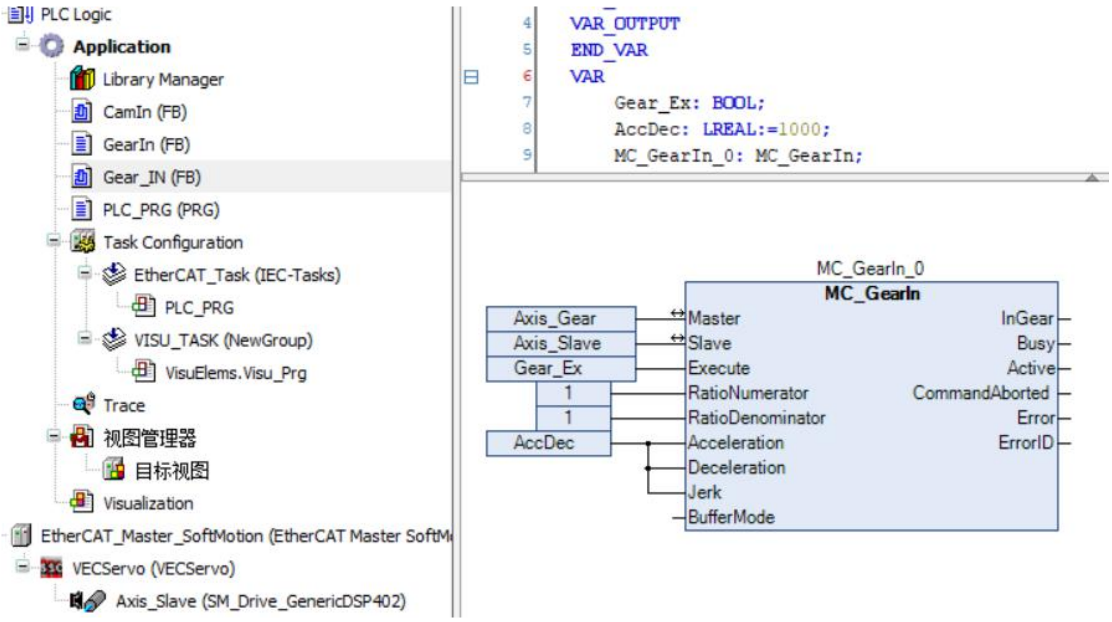
Cam slave and gear spindle: Axis_Gear (dummy axis).

Gear slave: Axis_Slave (real axis)

Note:

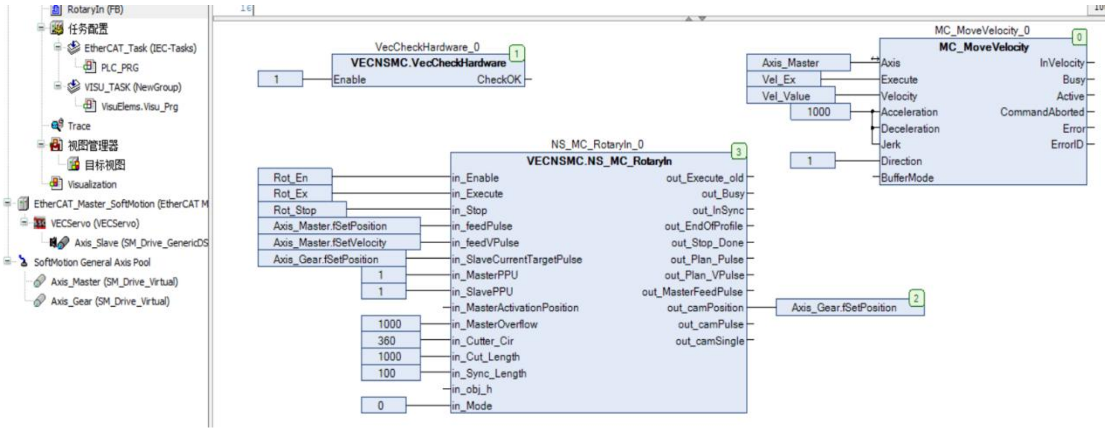
(1) The wheel cut function block must be used in conjunction with the VecCheckHardware function block and the special cam function block can only be used when the CheckOK output of the VecCheckHardware function block is TRUE.

(2) Since we cannot assign a value to the fSetPosition of the real axis directly, but can assign a value to the fSetPosition of the imaginary axis, we need to use the MC_GearIn function block to establish the gear relationship, so that the imaginary axis Axis_Gear is the main axis of the gear and the real axis Axis_Slave is the slave of the gear; as follows:



The cam's main axis Axis_Master and the imaginary axis Axis_Gear are then allowed to establish a wheel-cutting relationship so that the real axis Axis_Slave can be wheel-cut with the imaginary axis Axis_Gear by means of a gear.

To use the wheel-cutting function, first enable (in_Enable) the wheel-cutting module and then trigger (in_Execute) the wheel-cutting module

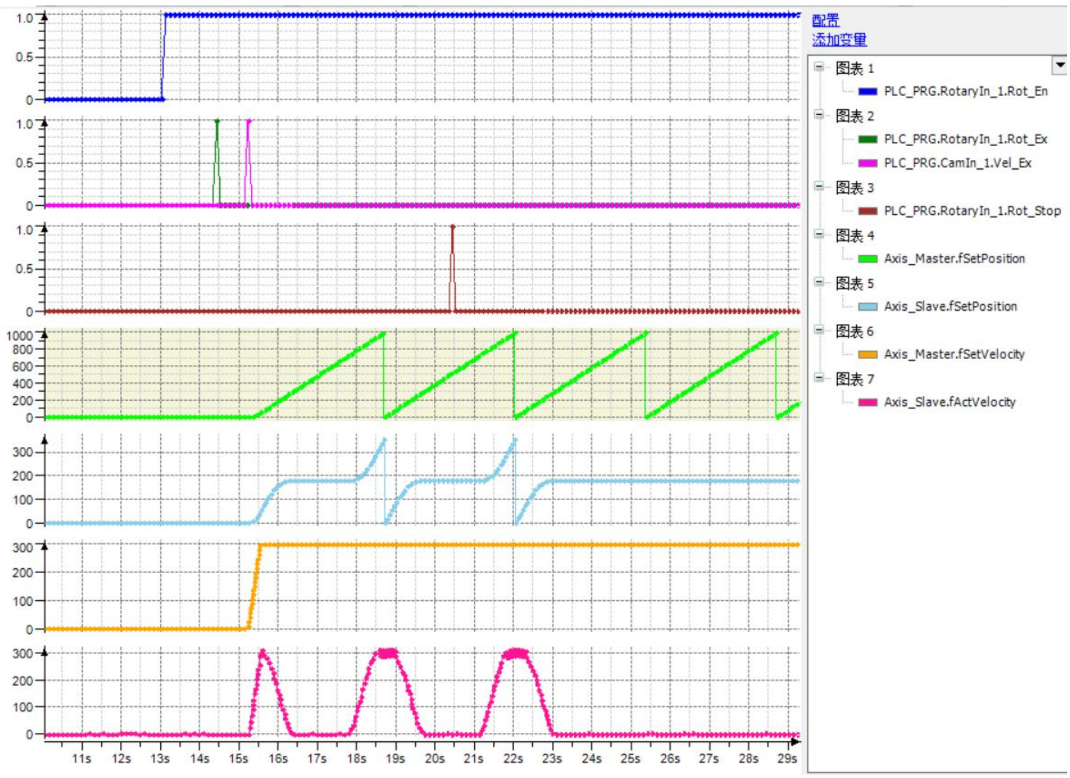


◆ Model description

in_Mode	Model description
0	Point 0 from axis in line with tangent point
1	Point 0 from axis at 180° from the tangent point

◆Timing diagram

The following diagram shows the timing diagram of the routine. When using the wheel-cutting function block, you need to set in_Enable to TRUE first, then give in_Execute a rising edge trigger, and then trigger the spindle speed to carry out the cam action (triggering the spindle speed first and then triggering in_Execute of the wheel-cutting module is also feasible); when you need to stop the cam, trigger in_Stop on the rising edge, and the slave axis will stop at 180° of the tangent point;



7.6 CNC Instructions

7.6.1 SMC_ReadNCFile2

The instruction function is: Read and write C NC files withG code.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_ReadNCFile2	Read theC NC file function block		SMC_ReadNCFile2(bExecute:= , sFileName:= , pvl:= , fDefaultVel:= , fDefaultAccel:= , fDefaultDecel:= , fDefaultVelFF:= , fDefaultAccelFF:= , fDefaultDecelFF:= , b3DMode:= , bStepSuppress:= , aSubProgramDirs:= , bParenthesesAsComments:= , bDisableJumpBuffer:= , bBusy=> , bError=> , ErrorID=> , errorPos=> , ErrorProgramName=> , sentences=> , adwFileSize=> , adwPos=>);

2) Related variables

input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	The execution condition	BOOL	TSTREET,FALSE	FLASE	An up-edge of the input will initiate the processing of the function block
sFileName	Filename	STRING(255)	The range of data		The file path for the file that contains the g code (e.g._cnc/CNC_3.cnc').
pvl	The list of variables	POINTER TO SMC_VARLIST	TSTREET,FALSE	FLASE	The list of variables defines the type and address of each variable that can be used from the g code. If there are no variables in the g code, this input is not used
fDefaultVel	The default speed	LREAL	The range of data	0	Use this input if speed (F) is not specified in

					the CNC file. Note: Only for main programs, not sub-programs.
fDefaultAccel	The default acceleration	LREAL	The range of data	0	Use this input if acceleration (E-plus) is not specified in the CNC file. Note: For main programs only, not for sub-programs
fDefaultDecel	The default speed reduction	LREAL	The range of data	0	Use this input if you do not specify speed reduction (E -)in the CNC file. Note: For main programs only, not for sub-programs
fDefaultVelFF	G0 default speed	LREAL	The range of data	0	The default speed of G0(FF),which is used if the speed is not specified in the CNC file. Note: For main programs only, not for sub-programs
fDefaultAccelFF	G0 the default acceleration	LREAL	The range of data	0	G0 defaults to acceleration degree EF plus. Use this input if acceleration is not specified in the CNC file. Note: For main programs only, not for sub-programs
fDefaultDecelFF	G0 slows down by default	LREAL	The range of data	0	G0 Default Speed Reduction EF -. Use this input if the reduction speed is not specified in the CNC file. Note: For main programs only, not for sub-programs
b3DMode	3D mode	BOOL	TSTREET,FALSE	TSTREET	If true, the G17 command (activates 3D mode) is implicitly executed
bStepSuppress	Comment	BOOL	TSTREET,FALSE	FLASE	When this input is

	processing				TRUE, the line that the CNC program begins with "/" is ignored. For FALSE, it will be processed
aSubProgramDirs	Sub-program	ARRAY [0..4] OF STRING(174)			Tasks that are frequently repeated, such as cavity milling, hole drilling, and tool changes, can be replaced with G code subp programs and called from here; For subprograms named "SUB", start with the directory aSubProgramDirs, and search for the file "sub.cnc" (in small case) in ascending order. The first matching file is used. The first empty directory name ends the search.
bParenthesesAsComments	Parenthesis is a multi-line comment	BOOL	TSTREET,FALSE	TSTREET	Set T RUE,which uses parentheses in G code to be considered multi-line comments, and FALSE, which can be used in expressions ('a-b)*c') for sub-program calls in g code such as ('N10 sub(17)').
bDisableJumpBuffer	Jump buffer	BOOL	TSTREET,FALSE	FALSE	True, this input disables the internal jump buffer, which is used to improve the performance of g code processing with jump (G20).

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bBusy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
bError	Error	BOOL	TRUE,FALSE	FALSE	When it isT RUE, the function block reports an error
ErrorID	The error code	SMC_ERROR			The error code that was output when the function block reported an error
errorPos	The wrong location	SMC_NC_SourcePosition			The wrong source location was detected
ErrorProgramName	The name of the wrong program	STRING			The wrong program name was detected
sentences	Sentence queue	SMC_GSentenceQueue			You can enter SMC_NCInterpreter sentence queue in the file
adwFileSize	The file size	ARRAY[0..(NUM_PARSER_CHAINS - 1)] OF DWORD	The range of data		File size, in bytes
adwPos	Read the location	ARRAY[0..(NUM_PARSER_CHAINS - 1)] OF DWORD	The range of data		The cursor is in its current position in the file

7.6.2 SMC_NCInterpreter

The instruction function is: the G code read to the read file function block is interpreted as a list SMC_GEOINFO the file.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_NCInterpreter	Decode the function block		SMC_NCInterpreter(sentences:= , bExecute:= , bAbort:= , bAppend:= , piStartPosition:= , vStartToolLength:= , nSizeOutQueue:= , pbyBufferOutQueue:= , bEnableSyntaxChecks:= , eOriConv:= , dCircleTolerance:= , pInterpreterStack:= , nInterpreterStackSizeBytes:= , bDone=> , bBusy=> , bError=> , wErrorID=> , errorPos=> , poqDataOut=> , iStatus=> , iLineNumberDecoded=> , GCodeText=> , CallstackInfo=> , aActivePrograms=>);

2) Related variables

input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
sentences	Sentence queue	SMC_GSentenceQueue			The input queue for the g code statement
bExecute	The execution condition	BOOL	TSTREET,FALSE	FALSE	An up-edge of the input will initiate the processing of the function block
bAbort	Abort processing	BOOL	TSTREET,FALSE	FLASE	If true, the current processing of this feature block is aborted
bAppend	Additional data	BOOL	TSTREET,FALSE	FLASE	If true, triggering bExecute does not result in a reset of the queue. Instead, the newly entered data is written out of the end of the queue
piStartPosition	The starting position	SMC_POSINFO	The range of data	0	The starting position of the path
vStartToolLength	The starting	SMC_Vector3d	The range of data	0	The starting tool length

	tool length				
nSizeOutQueue	The size of the buffer	UDINT	The range of data	0	This variable contains the size of the data buffer and is written to SMC_GEOINFO list of structured objects. The buffer must hold at least five SMC_GEOINFO objects. Otherwise, the function block will not perform any action at all. Its size can be predefined, but can only be modified later during the reset. It is recommended to create a buffer as follows: BUF: Array .50 of SMC_GeoInfo; The operator is then used to retrieve the appropriate buffer size, size size (BUF);
pbyBufferOutQueue	Point to the store	POINTER TO ARRAY [0..0]OF SMC_GEOINFO	The range of data	0	This input must point to the SMC_OUTQUEUE assigned to the store of the structure. This area must be at least as large as defined in NSizeOutQueue Asdefined: BUF: ARRAY (1..50) OF SMC_GEOINFO Then the ADR(BUF)points to thisinput;
bEnableSyntaxChecks	Grammar detection	BOOL	TSTREET,FALSE	TRUE	Turn on syntax detection, which detects an invalid G code and stops in this case as an error.
eOriConv	Directional explanation	SMC_ORI_CONVENTION		SMC_ORI_CONVENTION. ADDAXES	Defines how the direction in the A/B/C word is interpreted.
dCircleTolerance	The definition of a circle	LREAL			Tolerances determine whether the definition of a circle makes sense. Case is defined by the target position and radius: If the distance between the start and end positions (both projected onto the

					circle plane) is greater than , the circle is converted to a straight line. 2 s the radius and MAX (fCircle Tolerance, 1e-06) case definition by target position and center position: the maximum value of the distance between the starting and center position x and the distance between the target location and the center position (projecting everything onto the circular plane). If these distance differences are greater than , the circle is converted to a straight line. MAX(fCircleTolerance, 0.1 * x)
pInterpreterStack	Stack buffer	POINTER TO BYTE			Provides a buffer for the interpreter stack. If it is 0, the default buffer with a size of 10240 bytes is used. A given buffer will be used if the buffer is 0. This input is read per cycle.
nInterpreterStackSizeBytes	The size of the stack buffer	UDINT			The size of the buffer that pInterpreterStack points to. Note that the size should be at least 1024 bytes

The output variable

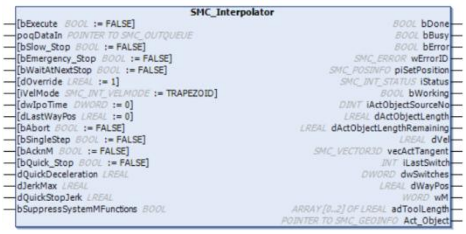
The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	The execution of the instruction is complete	BOOL	TRUE,FALSE	FALSE	The current instruction execution is complete and is set toT RUE
bBusy	The instruction is being	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE

	executed				
bError	Error	BOOL	TRUE,FALSE	FALSE	When the function block is reported as an error, it isT RUE
wErrorID	The error code	SMC_ERROR			The error code that was output when the function block reported an error
errorPos	The wrong location	SMC_NC_SourcePosition			The wrong source location was detected
poqDataOut	Position data output	POINTER TO SMC_OUTQUEUE			A SMC_OUTQUEUE to a structure that manages decoded SMC_GEOINFO objects
Status	The current state	SMC_DEC_STATUS			The current state
iLineNumberDecoded	The line number	DINT	The range of data		The currently decoded G-line number
GCodeText	G code text	SMC_GCODE_TEXT	The range of data		G code text
CallstackInfo	Stack information	SMC_NCCallstackInfo			
aActivePrograms	Currently active programs and sub-programs	ARRAY[0..(SoftMotion_NC2_Constants.IPR_CALLSTACK_SIZE - 1)] OF STRING			aActivePrograms saves the name of the (sub) program that is currently being interpreted. If it is a subprogram, then aActivePrograms saves the name of the calling (sub) program, and so on. The list of active programs terminates with an empty string.

7.6.3 SMC_Interpolator

The instruction function is: This function block is used to convert the continuous path described by the SMC_GEOINFO object to a discrete path location point, taking into account the defined speed curve and time pattern. These location points are then typically converted through an IEC program (for example, converted to drive shaft positions) and sent to the drive.

1) Instruction format

Instructions	Name	Graphical performance	ST performance
SMC_Interpolator	Interpolation function blocks		SMC_Interpolator(bExecute:= , poqDataIn:= , bSlow_Stop:= , bEmergency_Stop:= , bWaitAtNextStop:= , dOverride:= , iVelMode:= , dwIpoTime:= , dLastWayPos:= , bAbort:= , bSingleStep:= , bAcknM:= , bQuick_Stop:= , dQuickDeceleration:= , dJerkMax:= , dQuickStopJerk:= , bSuppressSystemMFunctions:= , bDone=> , bBusy=> , bError=> , wErrorID=> , piSetPosition=> , iStatus=> , bWorking=> , iActObjectSourceNo=> , dActObjectLength=> , dActObjectLengthRemaining=> , dVel=> , vecActTangent=> , iLastSwitch=> , dwSwitches=> , dWayPos=> , wM=> , adToolLength=> , Act_Object=>);

2) Related variables

input variables

Enter the variable	Name	The data type	Effective range	The initial value	Describe
bExecute	The execution condition	BOOL	TSTREET,FALSE	FALSE	An up-edge of the input will initiate the processing of the function block
poqDataIn	Location data entry	POINTER TO SMC_OUTQUEUE			The variable points to SMC_OUTQUEUE structure object, which contains the object of the SMC_GEOINFO path. Typically it points to the

					poqDataOut of the preprocessed function block output
bSlow_Stop	Stop slowly	BOOL	TSTREET,FALSE	FLASE	If you set this variable to FALSE, the path is passed non-stop. Set to TRUE, SMC_Interpolator will reduce the speed to 0 based on the defined speed curve (byVelMode) and the maximum deceleration (dDecel) of the current SMC_GEOINFO object, and wait until bSlow_Stop to reset it to FALSE
bEmergency_Stop	Stop immediately	SMC_POSINFO	The range of data	0	As soon as the input gets TRUE, SMC_Interpolator stop immediately, which means that the current location is preserved. Therefore, the speed is immediately set to 0
bWaitAtNextStop	Wait for the next stop point	SMC_Vector3d	The range of data	0	As long as this variable is FALSE(the default), thepath ispassed non-stop. Otherwise, SMC_Interpolator stop at the next regular point until bWaitAtNextStop resets it to FALSE
dOverride	Speed factor	LREAL	The range of data	1	This variable can be overwritten online. Less than 0.01 is not allowed. The multiply used to change the interpolation speed, such as dOverride plus 2, doubles the speed. Note: The multiply can be modified at any time, but can only be applied if there is currently no acceleration or deceleration.
iVelMode	Speed mode	SMC_INT_VELMODE		TRAPEZOID	This input defines the speed SMC_INT_VELMODE

					defined in the data set
dwIpoTime	Cycle time	DWORD	The range of data	0	This variable must be set for each call. It represents the cycle time in microseconds
dLastWayPos	The last extension path	LREAL	The range of data	0	This input allows the user to measure the extension of the path protruding from the interpolator. The output of this module, dWayPos, is the same as the distance covered by dLastWayPos and the current period. If dLastWayPos is set to equal the output dWayPos, dWayPos will always increment in the current path segment, resulting in the total length of the travel path. dLastWayPos can be reset to 0 or other values at any time.
bAbort	Abort processing	BOOL	The range of data	FALSE	If true, the current processing of this feature block is aborted
bSingleStep	Stop a cycle	BOOL	The range of data	FALSE	The purpose of this input is that the interpolator stops a cycle at the transition between the two path objects (also at the same transition where the cut is made). If you set its bSingleStep to TRUE during the move, the interpolator stops at the end of the object and can reach that target without exceeding the predetermined deslevel value. If the interpolator should stop at the next possible stop position (i.e. a point with a speed of 0), bWaitAtNextStop must be used.
bAcknM	The M	BOOL	The range	FALSE	This input can be used to

	function is confirmed		of data		confirm the M function. If the input is TRUE, the output wM is cleared and path processing continues
bQuick_Stop	Stop quickly	BOOL	The range of data	FALSE	If this input is TRUE, the interpolator reduces the speed to zero until bQuick_Stop reset it to FALSE. Decelerates according to the defined speed curve (by VelMode) and the deceleration given in the(dQuickDeceleration) path. If secondary speed mode is used, the impact is limited. max(dJerkMax, dQuickStopJerk)
dQuickDeceleration	The desdation value of the quick stop	THEREAL	The range of data	0	The bQuick_Stop used to reduce the value
dJerkMax	The amplitude of the maximum acceleration	THEREAL	The range of data	0	Only for secondary speed mode. It must be positive and cannot be changed while the interpolator is running
dQuickStopJerk	Fast-stopping acceleration	THEREAL	The range of data	0	If one of the secondary velocity modes is selected, the emergency stop uses a sharp amplitude to reduce acceleration
bSuppressSystemMFunctions	Output wM flag bit	BOOL	The range of data	FALSE	If this option is set, the output wM is not set for the internal M feature created by the G75 or G4 commands

The output variable

The output variable	Name	The data type	Effective range	The initial value	Describe
bDone	The	BOOL	TRUE,FALSE	FALSE	The current instruction

	execution of the instruction is complete				execution is complete and is set toT RUE
bBusy	The instruction is being executed	BOOL	TRUE,FALSE	FALSE	The current instruction is in execution and is set to TRUE
bError	Error	BOOL	TRUE,FALSE	FALSE	When the function block is reported as an error, it isT RUE
wErrorID	The error code	SMC_ERROR			The error code that was output when the function block reported an error
piSetPosition	Calculate the set position	SMC_NC_SourceP osition			It reflects the calculated set position and contains the Descartes coordinates for the next position and the status of the attached axis
Status	The current state	SMC_INT_STATUS	IPO_INIT		The enumeration variable reflects the SMC_INT_STATUS of the function blocks defined in the database. Possible state: IPO_UNKNOWN (0): Internal state changes that may not occur after SMC_Interpolator. IPO_INIT (1): Initialized state; IPO_ACCEL (2): Acceleration IPO_CONSTANT (3): Constant motion IPO_DECEL (4): Deceleration IPO_FINISHED (5): Path complete. Any other objects that SMC_GEOINFO will not be processed by poqDataIn. IPO_WAIT (6): Wait for one of the following:bEmergency_Stop sTRUEbSlow_Stop s TRUE and dVel s true and dVels true bWait_At_Next_Stop and dVels 0 IPO_INCREASING_ACCEL (7): Increase acceleration IPO_DECREASING_ACCEL (8): Reduce acceleration

					IPO_INCREASING_DECEL (9): Increase deceleration IPO_DECREASING_DECEL (10): Lower Slow down
bWorking	The current state	BOOL	The range of data	FALSE	T RUE only if list processing has started but has notyet been completed
iActObjectSourceNo	The current interpolation line number	DINT	The range of data	-1	The runtime displays the line number of the current interpolation in real time, which is -1 when bWorking is FALSE
dActObjectLength	The current object length	LREAL	The range of data		The length of the current object is output when bWorking is TRUE
dActObjectLengthRemaining	The remaining length of the current object	LREAL	The range of data		When bWorking is TRUE, the remaining length of the current object is output
dVel	The current path speed	LREAL	The range of data	0	This variable contains the current path speed
vecActTangent	The actual path cut	SMC_VECTOR3D			This structure contains path slices, or unit vectors
iLastSwitch	Last switch	INT	The range of data	0	This output contains the number of the last passed switch. Note: If more than one switch passes in a cycle, only the last one is mentioned
dwSwitches	Multi-switch bit	DWORD	The range of data	0	Describes the current switching statusof all switches 1 - 32. Bit0 means switch1, Bit31 means switch32. Compared to iLastSwitch, this bit field also contains multiple switches in a cycle
dWayPos	The extended path	THEREAL	The range of data		See Enter dLastWayPos
wM	The M	WORD	The range of		If the interpolator passes the M

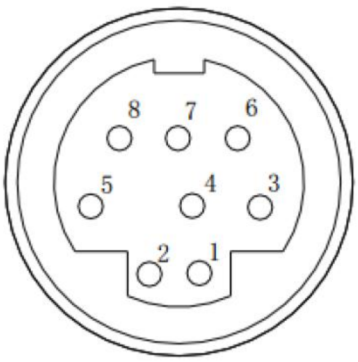
	function associates the value		data		function, this output is set to the value associated with the M function. The interpolator will stop until the M function (bAcknM) is entered
adToolLength	Tool length compensation parameters	ARRAY [0..2] OF LREAL	The range of data		Parameters for tool length compensation (set by G43 I /J/K).
Act_Object	Point to the interpolation path	POINTER TO SMC_GEOINFO			A pointer to the current interpolation path element. It could be 0

8 Comprehensive configuration debugging

8.1 Modbus Communications

8.1.1 ModBusRTU_Slave

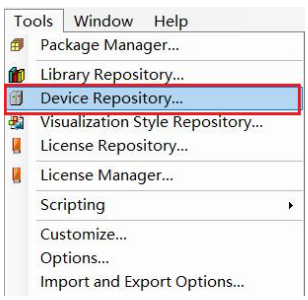
The VE motion controller supports standard ModeBusRTU communication, connecting to the touch screen serial port via a communication serial port. The following step touch screen as an example, through the serial RS232/485, VE controller connected to two touch screens, touch screen as the main station, VE controller as a startingstation, the wiring diagram is as follows:



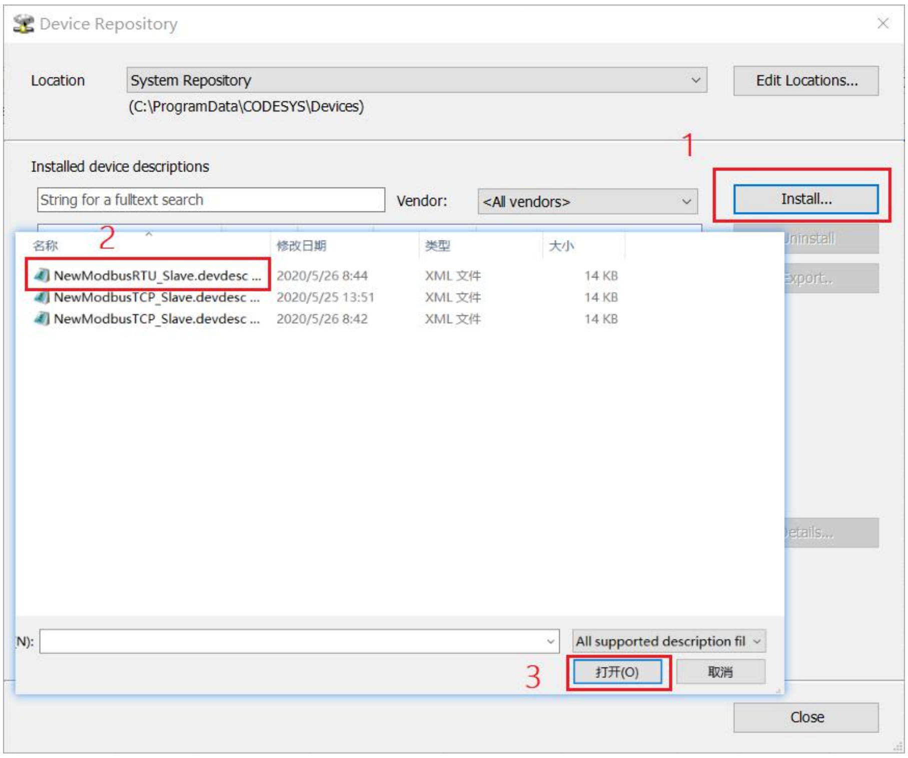
Foot position	Defined
1	RS485 -
2	RS485 +
5	GND

Install ModbusRTU_Slave

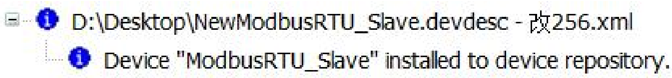
To use the ModeBusRTU_Slave,first install the device by clicking onthe toolbar "Tools→Device Repository"



Then click "Install" to find the device description file"ModBusRTU_Slave.xml" andselect and clickOpen

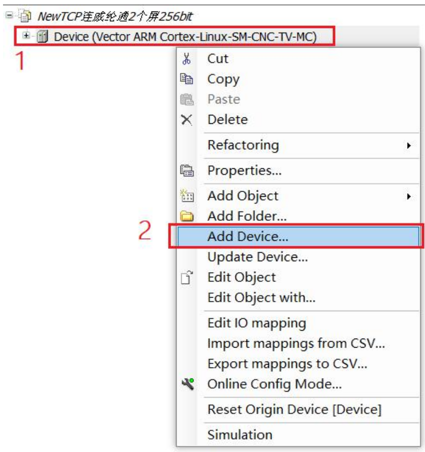


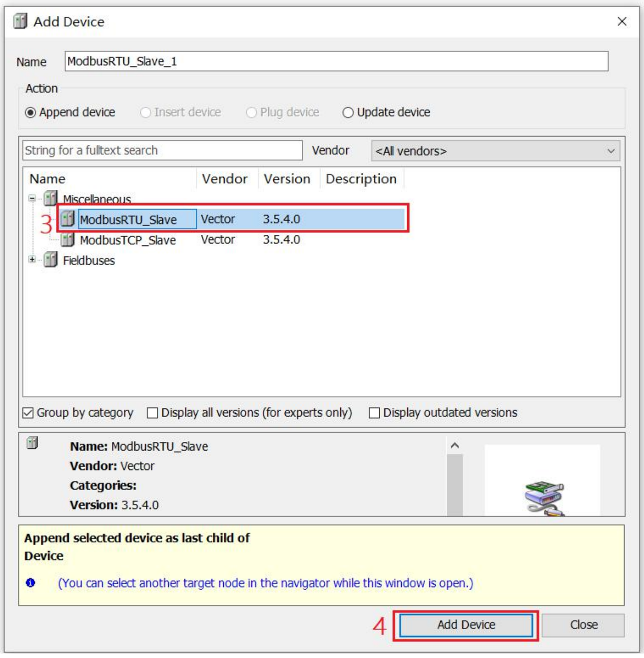
Displays that the installation was successful, which indicates that the device was installed successfully and can be added for use.



Add an RTU device

After the new project is created, select "Right-click → Device Add Device Mod→BUSRTU_SlaveAdd Device" to confirm that the→RTUdevice isadded and that the VE controller will be added to the project as a from the station.





When the addition is complete, double-click to open as follows.

ModbusRTU_Slave

PCI-Bus IEC Objects | Internal Parameters | Internal I/O Mapping | Status | Information

Parameter	Type	Value	Default Value	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'ModbusRTU'	'ModbusRTU'		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
ID	WORD	1	1		ModbusRTU slave ID
Baud	WORD	9600	9600		ModbusRTU baud rate
Data_bits	WORD	8	8		ModbusRTU bata bites
Check	WORD	0	0		ModbusRTU Check. 0:no, 1:odd, 2:even.
Stop bits	WORD	1	1		ModbusRTU Stop bits.

Parameter settings

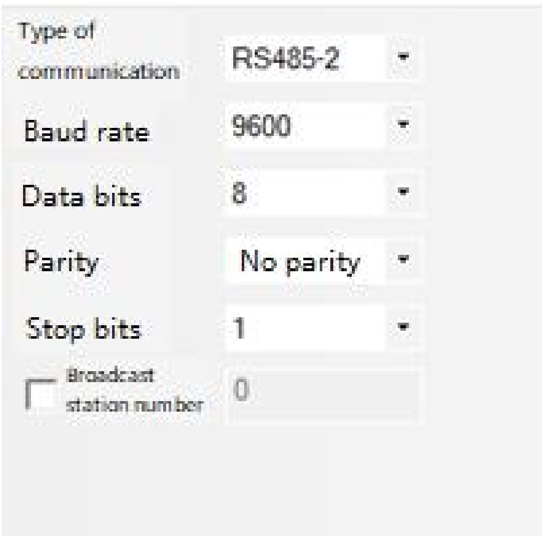
After double-clicking expands, the VE controller sets the parameters as shownbelow, modBusRTU_Slaverelated parameter settings include: station number, baud rate, data bit, parity, stop bit.

ModbusRTU_Slave

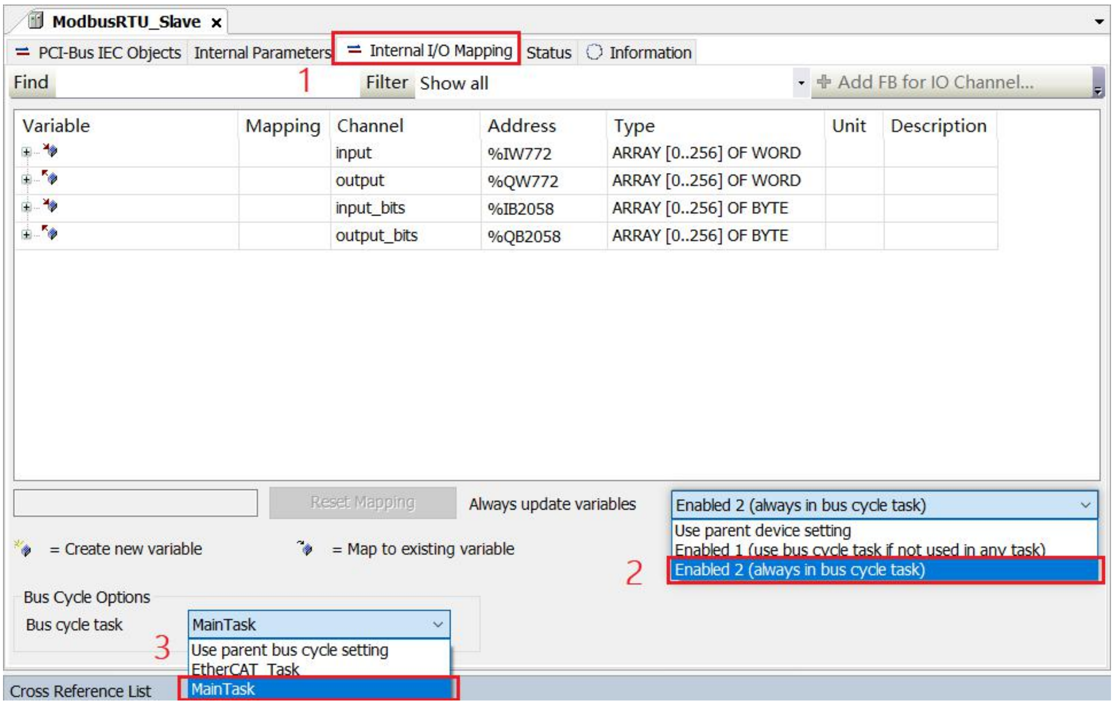
PCI-Bus IEC Objects | Internal Parameters | Internal I/O Mapping | Status | Information

Parameter	Type	Value	Default Value	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'ModbusRTU'	'ModbusRTU'		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
ID Station number	WORD	1	1		ModbusRTU slave ID
Baud Baud rate	WORD	9600	9600		ModbusRTU baud rate
Data_bits Data bits	WORD	8	8		ModbusRTU bata bites
Check Checksum	WORD	0	0		ModbusRTU Check. 0:no, 1:odd, 2:even.
Stop bits Stop bits	WORD	1	1		ModbusRTU Stop bits.

At the same time, the touch screen engineering also needs to set the corresponding station number and related parameters



You also need to set up a data scan refresh cycle, as shown below, selecting Main_Task as the task for the scan, allowing the user to select other tasks (non EtherCAT_Task) and set the task cycle time.



Address-associated variables

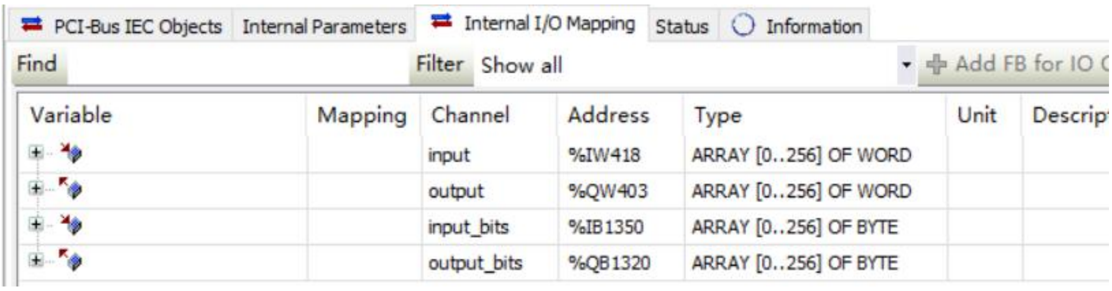
In the ModBusRTU_Slave device, the mapping address is provided as follows:

Type	Channel	Description
input (Address type: 4X).	input[0] ~ input[1023]	Enter the register power-down hold area
	input[1024] ~ input[4095]	The input register power-down does not hold the zone
output (Address type: 3X).	output[0] ~ output[4095]	
input_bit (Address type: 0X).	input_bit[0]~input_bit[1023]	Enter the coil power-down hold area
	input_bit[1024]~input_bit[4095]	Enter that the coil is powered down and does not hold the zone

output_bit (Address type: 1X).	ooutput_bit[0]~output_bit[4095]	

Touch screen address types: 0X, 1X, 3X, 4X, corresponding to ModBusRTU_Slave address channels are:input_bit,output_bit,output,input.

Variables determine the number of channels occupied according to their own data type, such as INT variables occupy one WORD, REAL, DINT variables occupy two WORDS, LREAL, LINT variables occupy 4 WORDS, and so on.

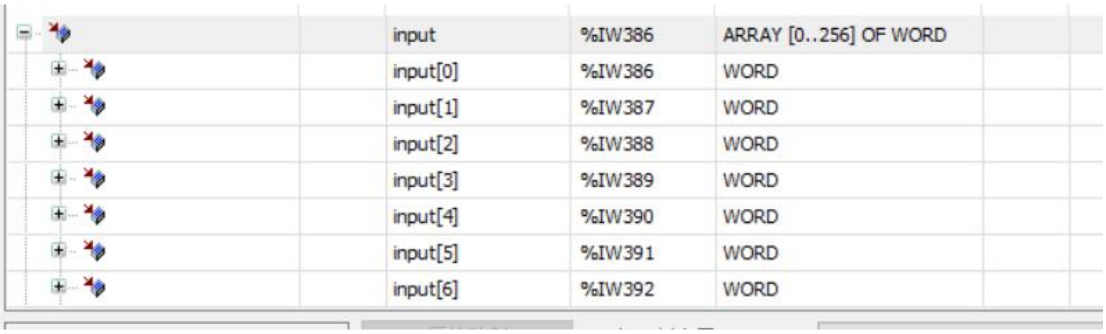


Variable	Mapping	Channel	Address	Type	Unit	Descrip
		input	%IW418	ARRAY [0..256] OF WORD		
		output	%QW403	ARRAY [0..256] OF WORD		
		input_bits	%IB1350	ARRAY [0..256] OF BYTE		
		output_bits	%QB1320	ARRAY [0..256] OF BYTE		

Attention:

1, when the variable type is 32 bits (such as REAL) or 64 bits (such as LREAL), the address map should start from a double address mapping, such as REAL data can not be mapped to the address %IW5, can only be mapped to %IW4 or %IW6 and other double address, otherwise compilation will report errors.

2, the associated address should be associated according to the channel's starting address, as shown below, the variable wants to associate to the channel input



	input	%IW386	ARRAY [0..256] OF WORD
	input[0]	%IW386	WORD
	input[1]	%IW387	WORD
	input[2]	%IW388	WORD
	input[3]	%IW389	WORD
	input[4]	%IW390	WORD
	input[5]	%IW391	WORD
	input[6]	%IW392	WORD

In a device, io addresses are mapped to variables in two ways.

Method 1: Map addresses in variable declarations, as shown below.

Input type:

```
PROGRAM POU1
VAR
    //    input
    A1 AT %IW1:INT; //INT
    A2 AT %IW2:REAL; //REAL
    A3 AT %IW4:UINT;//UINT
    A4 AT %IX5.0:BOOL;//BOOL
    //    output
    B1 AT %QX0.0:BOOL;//BOOL
    B2 AT %QW1:INT:=10;
END_VAR
```

Table type:

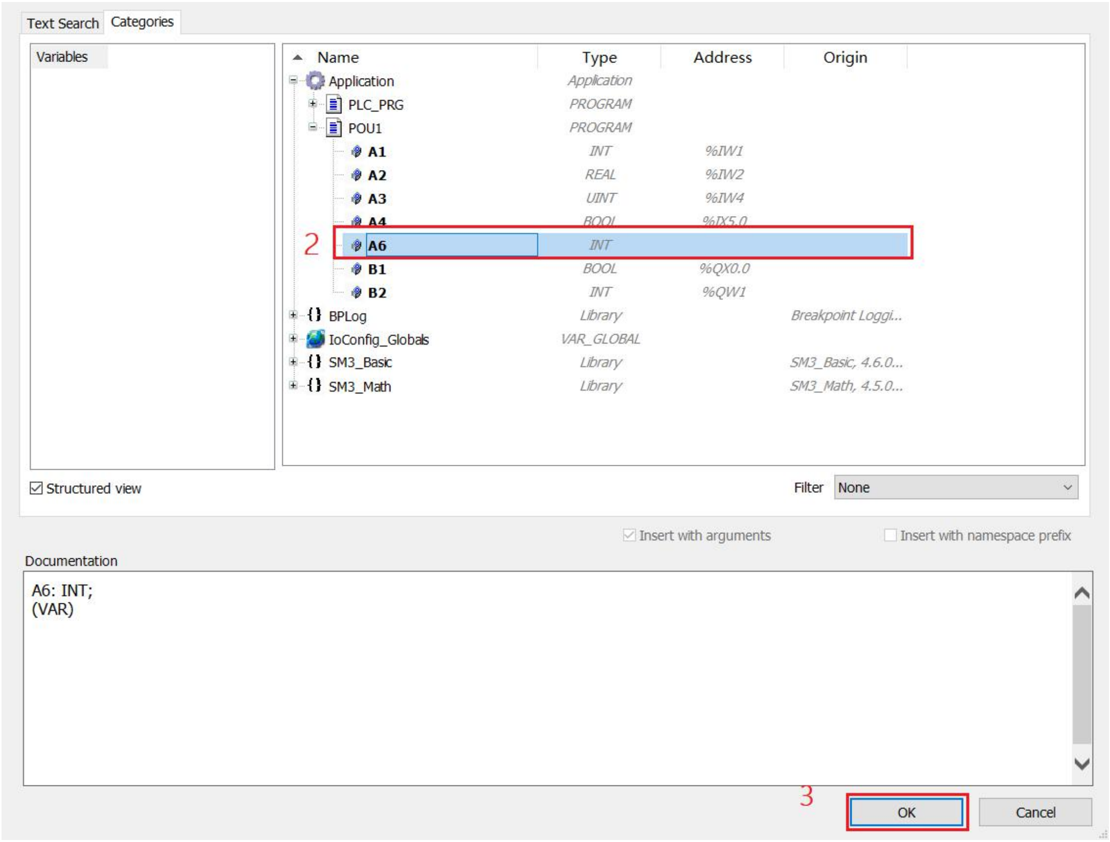
Scope	Name	Address	Data type	Initialization	Comment	Attributes
VAR	A1	%IW1	INT			
VAR	A2	%IW2	REAL			
VAR	A3	%IW4	UINT			
VAR	A4	%IX5.0	BOOL			
VAR	B1	%QX0.0	BOOL			
VAR	B2	%QW1	INT	10		

Method 2: Select a variable in the io mapping list.

Internal Parameters | Internal I/O Mapping | Status | Information

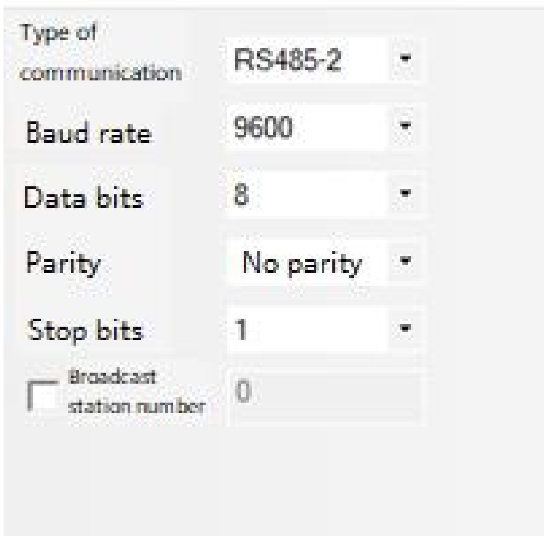
Find | Filter Show all

Variable	Mapping	Channel	Address	Type
1		input[5]	%IW5	WORD
...		input[6]	%IW6	WORD
		input[7]	%IW7	WORD
		input[8]	%IW8	WORD



HMI settings

Taking the step touch screen as an example, the system parameters are set as follows:



The PLC parameters are set as follows:

The VE motion controller corresponds to the human ModBus address as follows:

Motion controller channel address - Human-machine address - 1

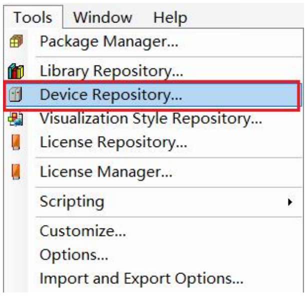
The channel type	The controller address type	The type of human-machine address
inputbit	input_bit[0]	0X 1
	input_bit [1]	0X 2
	input_bit [2]	0X 3
input	input[3]	4X 4
	input[500]	4X 501
outputbit	ooutput_bit[0]	1X 1
output	output[500]	3X 501

8.1.2 ModBusTCP_Slave

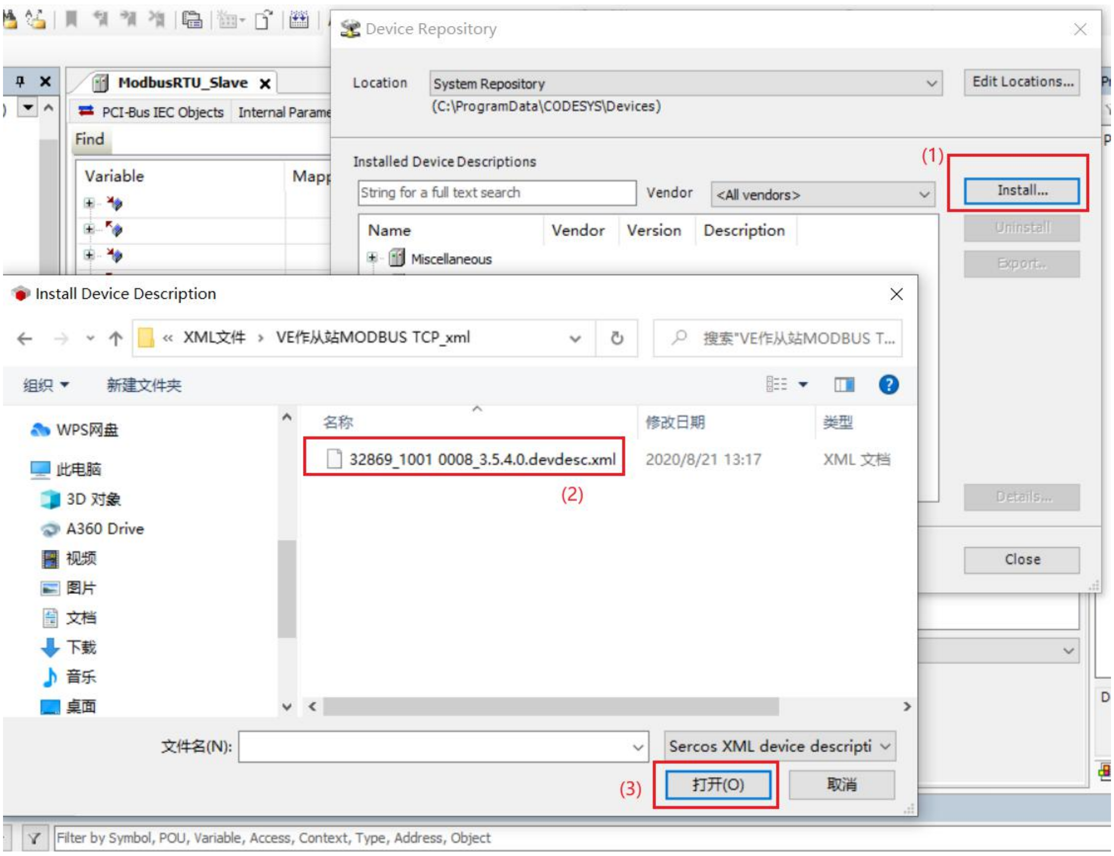
The VE motion controller supports standard ModeBusTCP communication, connected to a touch screen or switch via the EtherNet communication port. The following is an example of the Veronton touch screen, through the switch, VE controller EtherNet network port to connect two touch screens, touch screen as the main station, VE controller as a starting station, the specific operation steps are as follows:

Install ModBusTCP_Slave device

To use the ModeBusTCP_Slave,first install the device by clicking onthe toolbar "Tools→Device Repository"



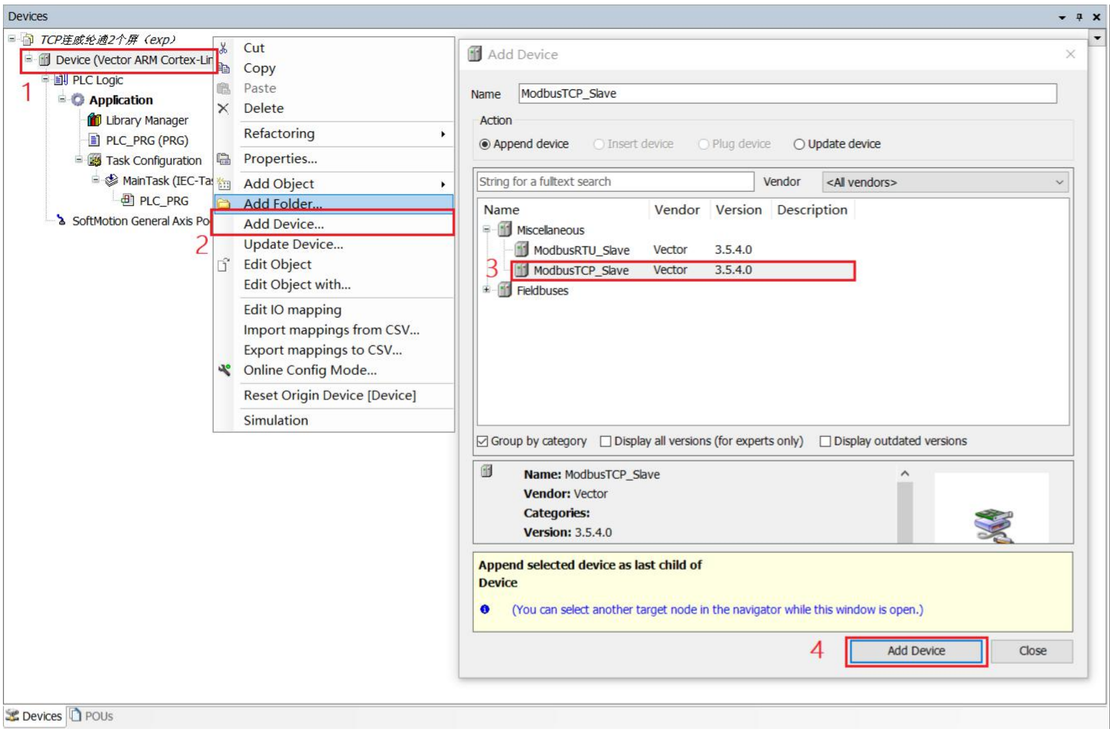
Then click "Install" to find the device descriptionfileVEC_ModBusTCP".xml Slave"and select and click Open



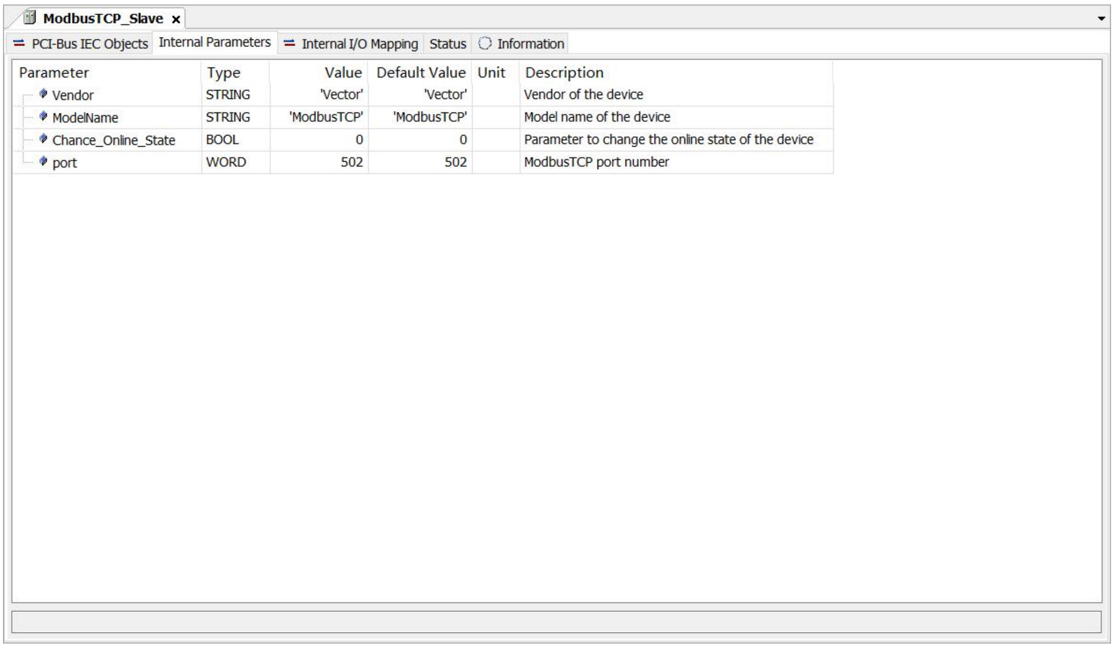
Displays that the installation was successful, which indicates that the device was installed successfully and can be added for use.

Add a TCP device

After the new project is created, select Right-click → Device Add Device→ModBusTCP_SlaveAdd Device to confirm that the→TCPdevice is added and that the VE controller isadded to the project as a from the station.



When the addition is complete,double-click on the following,ModBusTCP_Slave's EtherNet default IP is 192.168.1.123, consistent with the host, and requires the touch screen IP to be set in the same band, distinguished by port number.

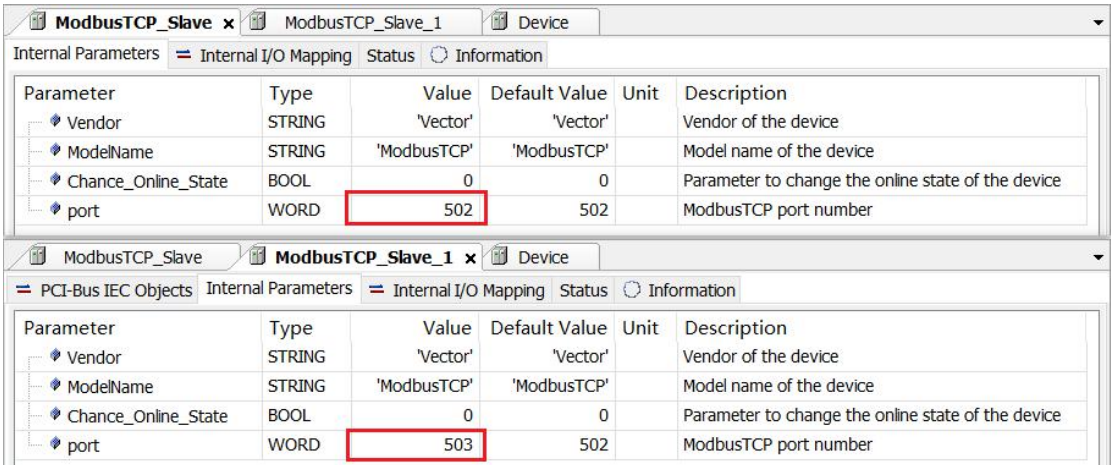


Parameter	Type	Value	Default Value	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'ModbusTCP'	'ModbusTCP'		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
port	WORD	502	502		ModbusTCP port number

In the same way, add a ModBusTCP_Slave device.

Parameter settings

Under the same network segment, the VE controller distinguishes between two touch screen devices by port number. As shownbelow, the modBusTCP_Slave port number is set to "502"andthe ModBusTCP_Slave-1 port number is set to 503

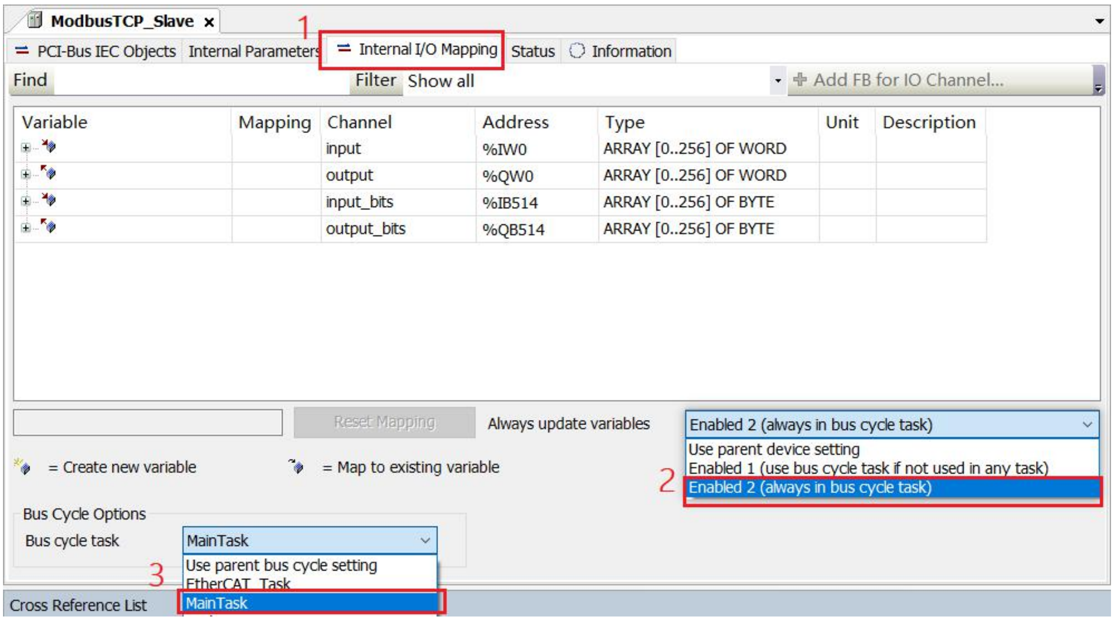


Parameter	Type	Value	Default Value	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'ModbusTCP'	'ModbusTCP'		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
port	WORD	502	502		ModbusTCP port number

Parameter	Type	Value	Default Value	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'ModbusTCP'	'ModbusTCP'		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
port	WORD	503	502		ModbusTCP port number

At the same time, the two touch screen projects also need to set the corresponding IP address and port number

At the same time, you also need to set the data scan refresh cycle, as shown in the figure below, select Main_Task as the scanning task, the user can choose other tasks and set the task cycle.



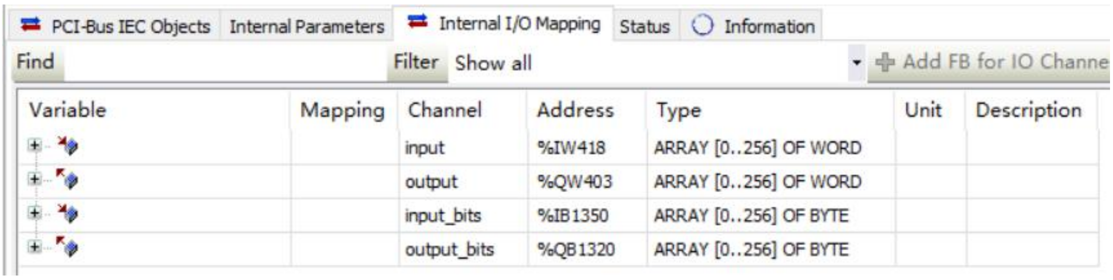
Address-associated variables

In the ModBusTCP_Slave device, the mapping address is provided as follows:

Type	Channel	Description
input (Address type: 4X).	input[0] ~ input[1023]	Enter the register power-down hold area
	input[1024] ~ input[4095]	The input register power-down does not hold the zone
output (Address type: 3X).	output[0] ~ output[4095]	
input_bit (Address type: 0X).	input_bit[0]~input_bit[1023]	Enter the coil power-down hold area
	input_bit[1024]~input_bit[4095]	Enter that the coil is powered down and does not hold the zone
output_bit (Address type:	ooutput_bit[0]~output_bit[4095]	

1X).		

Each channel data size is IN BOOL and WORD, and the address is expressed as %IX and %QX, or %IW and %QW. Variables determine the number of channels occupied according to their own data type, such as WORD variables occupy 16 BOOL-type positions, INT-type variables occupy one WORD, REAL variables occupy two WORD, LREAL variables occupy 4 WORD,and so on.

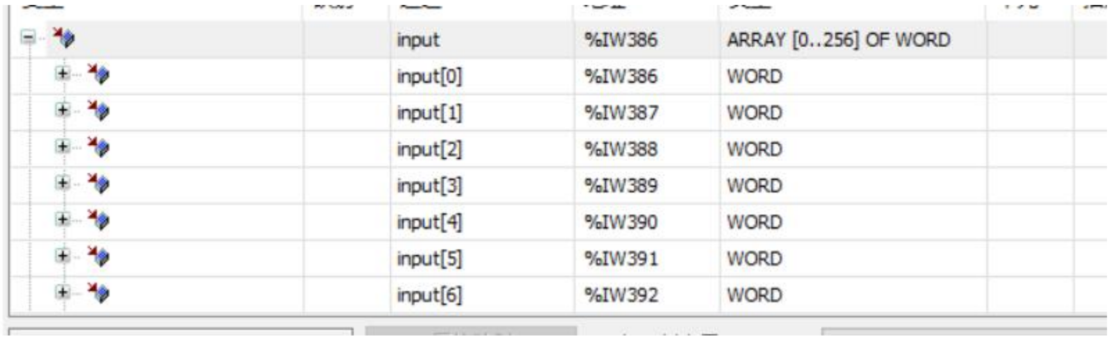


Variable	Mapping	Channel	Address	Type	Unit	Description
		input	%IW418	ARRAY [0..256] OF WORD		
		output	%QW403	ARRAY [0..256] OF WORD		
		input_bits	%IB1350	ARRAY [0..256] OF BYTE		
		output_bits	%QB1320	ARRAY [0..256] OF BYTE		

Note:

1, when the variable type is 32 bits (such as REAL) or 64 bits (such as LREAL), the address map should start from a double address mapping, such as REAL data can not be mapped to the address %IW5, can only be mapped to %IW4 or %IW6 and other double address, otherwise compilation will report errors.

2, the associated address should be associated according to the channel's starting address, as shown below, the variable wants to associate to the channel input



Variable	Mapping	Channel	Address	Type
		input	%IW386	ARRAY [0..256] OF WORD
		input[0]	%IW386	WORD
		input[1]	%IW387	WORD
		input[2]	%IW388	WORD
		input[3]	%IW389	WORD
		input[4]	%IW390	WORD
		input[5]	%IW391	WORD
		input[6]	%IW392	WORD

In a device, io addresses are mapped to variables in two ways.

Method 1: Map addresses in variable declarations, as shown below.

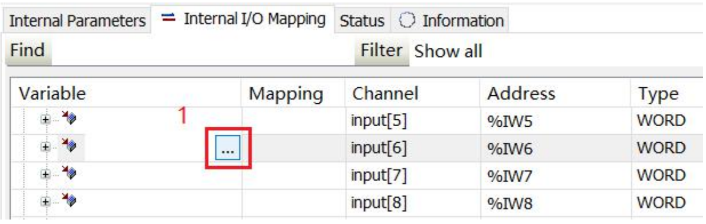
Input type:

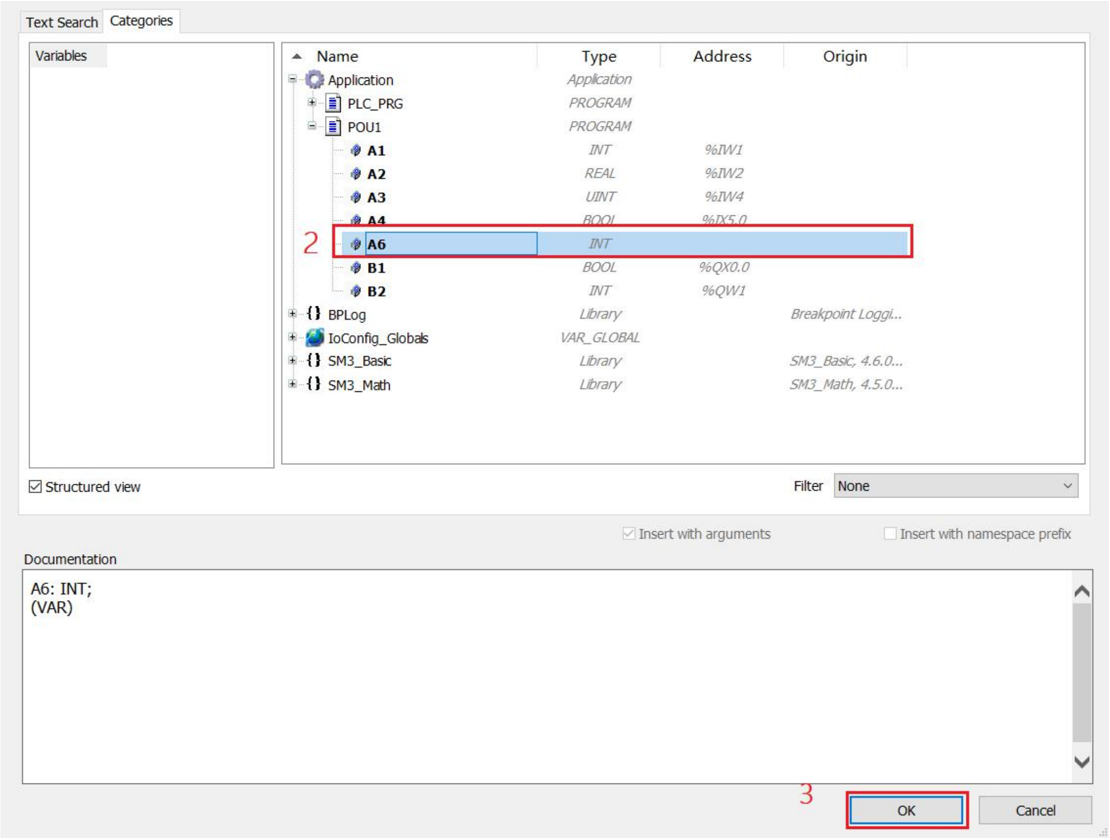
```
PROGRAM POU1
VAR
    //      input
    A1 AT %IW1:INT; //INT
    A2 AT %IW2:REAL; //REAL
    A3 AT %IW4:UINT;//UINT
    A4 AT %IX5.0:BOOL;//BOOL
    //      output
    B1 AT %QX0.0:BOOL;//BOOl
    B2 AT %QW1:INT:=10;
END_VAR
```

Table type:

Scope	Name	Address	Data type	Initialization	Comment	Attributes
VAR	A1	%IW1	INT			
VAR	A2	%IW2	REAL			
VAR	A3	%IW4	UINT			
VAR	A4	%IX5.0	BOOL			
VAR	B1	%QX0.0	BOOL			
VAR	B2	%QW1	INT	10		

Method 2: Select a variable in the io mapping list.

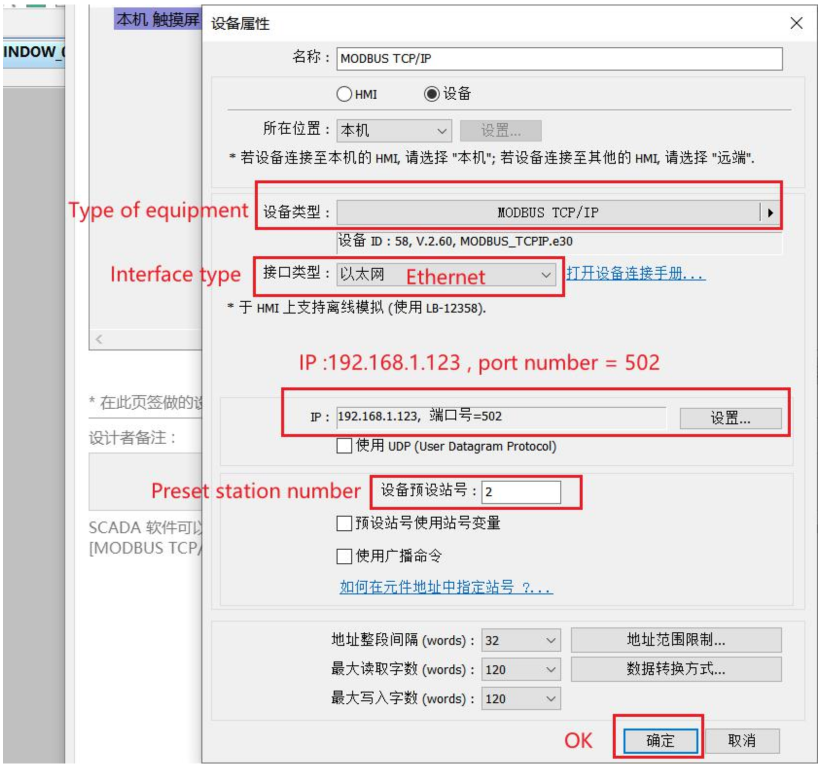




HMI settings

Taking the Willon pass touch screen as an example, the system parameters are set as follows:

ModBusTCP device parameters are set as follows, note the IP address and port number settings, reference parameter settings:



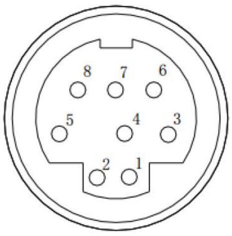
The corresponding relationship between the motion controller and the man-machine ModBus address is as follows:

Motion controller channel address - Human-machine address -1

The channel type	The controller address type	The type of human-machine address
inputbit	%IX0. 0	0X 1
	%IX1. 0	0X 2
	%IX2. 0	0X 3
input	%IW3	4X 4
	%IW500	4X 501
outputbit	%QX0.0	1X 1
output	%QW500	3X 501

8.1.3 ModBusRTU_Master

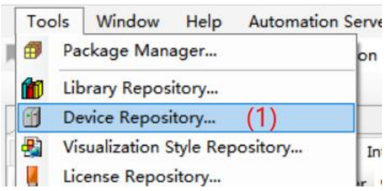
The VE motion controller supports standard ModeBusRTU communication methods, connecting individual stations via the communication serial RS232/485,with the VE controlleras the primary station. The serial wiring diagram is as follows:



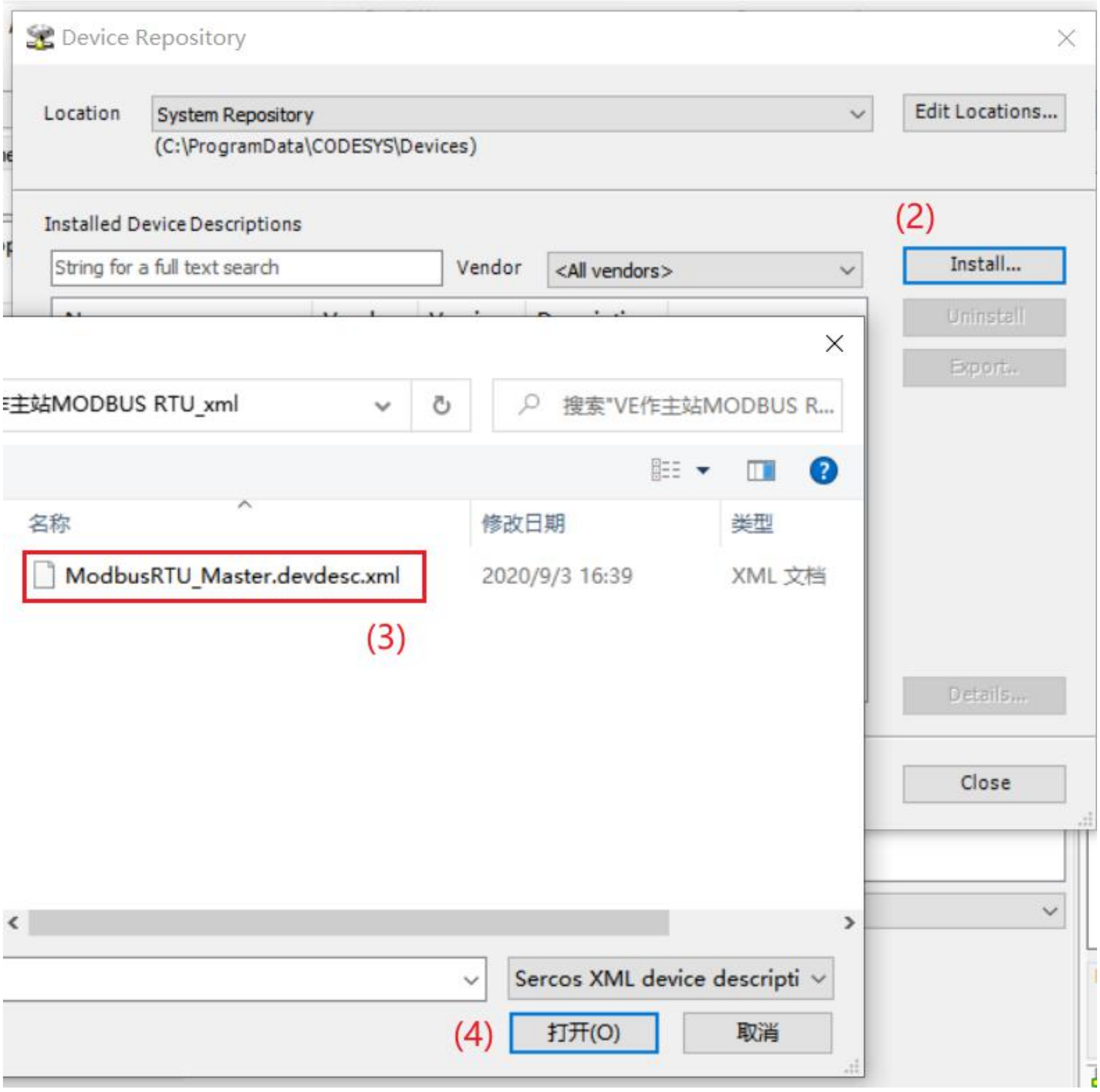
Foot position	Defined
1	RS485 -
2	RS485 +
5	GND

Install ModBusRTU_Master device

(1) Click on the toolbar's "Tools" and click on "Device Store"



- Click "Install"
- Select the ModbusRTU_Master.xml file
- Click Open

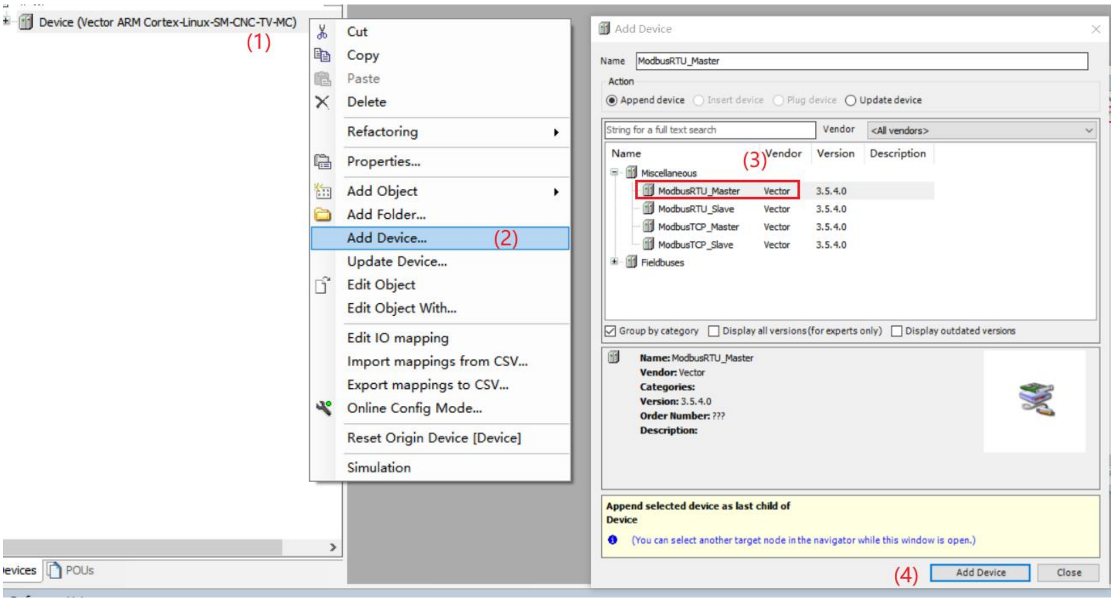


(5) shows that the installation was successful, which indicates that the installation of the device was successful and can be added for use.

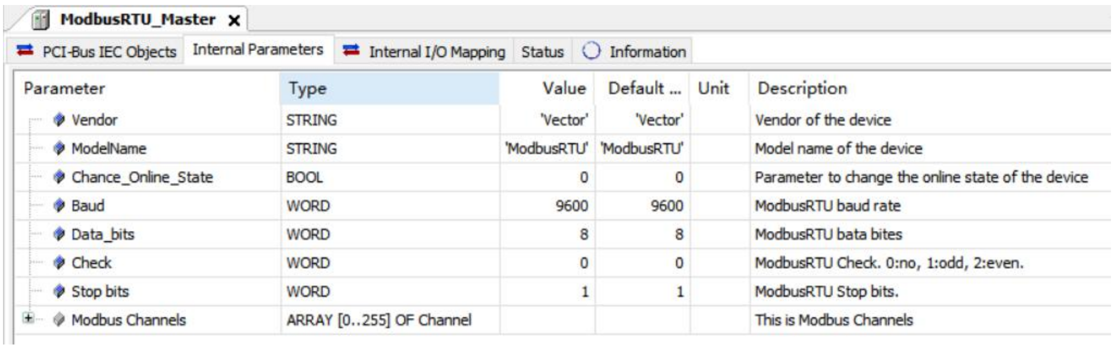
Device "ModbusRTU_Master" installed to device repository

Add ModBusRTU_Master device

After the new project is created, select Right-click Device Add Device → →ModBusRTU-MasterAdd → Device to confirm that the RTUdevice is added and that the VE controller will beadded to the project as the primary station.



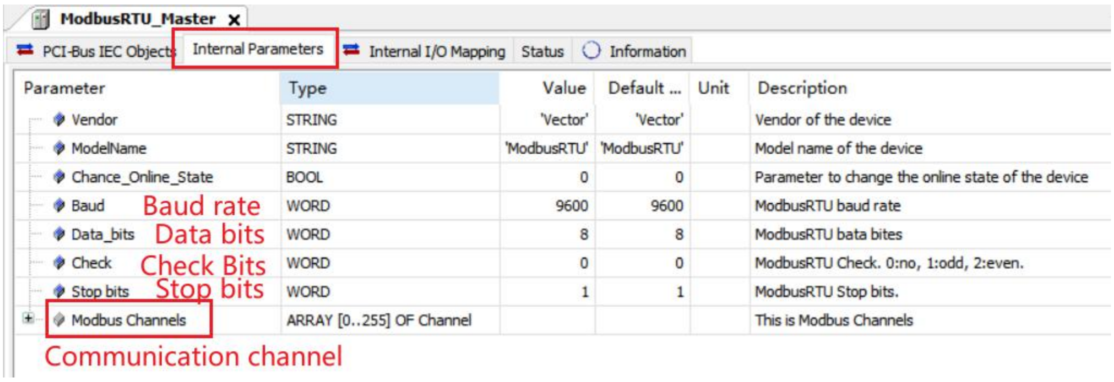
When the addition is complete, double-click to open as follows.



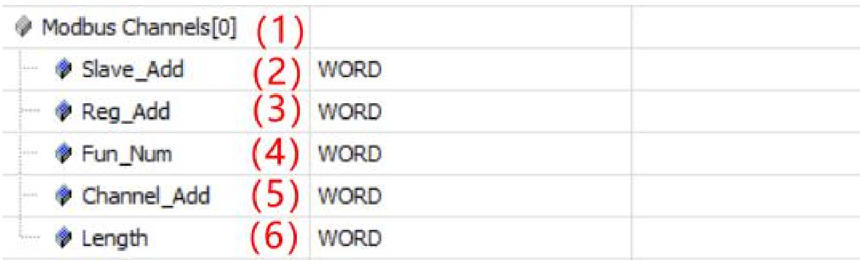
Parameter	Type	Value	Default ...	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'ModbusRTU'	'ModbusRTU'		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
Baud	WORD	9600	9600		ModbusRTU baud rate
Data_bits	WORD	8	8		ModbusRTU bata bites
Check	WORD	0	0		ModbusRTU Check. 0:no, 1:odd, 2:even.
Stop bits	WORD	1	1		ModbusRTU Stop bits.
Modbus Channels	ARRAY [0..255] OF Channel				This is Modbus Channels

Parameter settings ("Internal parameters" introduction)

After double-clicking expand, click "Internal parameters" and the VE controller sets the parameters as follows:



ModBusRTU-Master-relatedparameter settings include Baud rate, data bit, check bit, stop bit, communication channel.



Once the Modbus Channels are expanded, a total of 256 channels are available, with an example of expanding the first channel here:

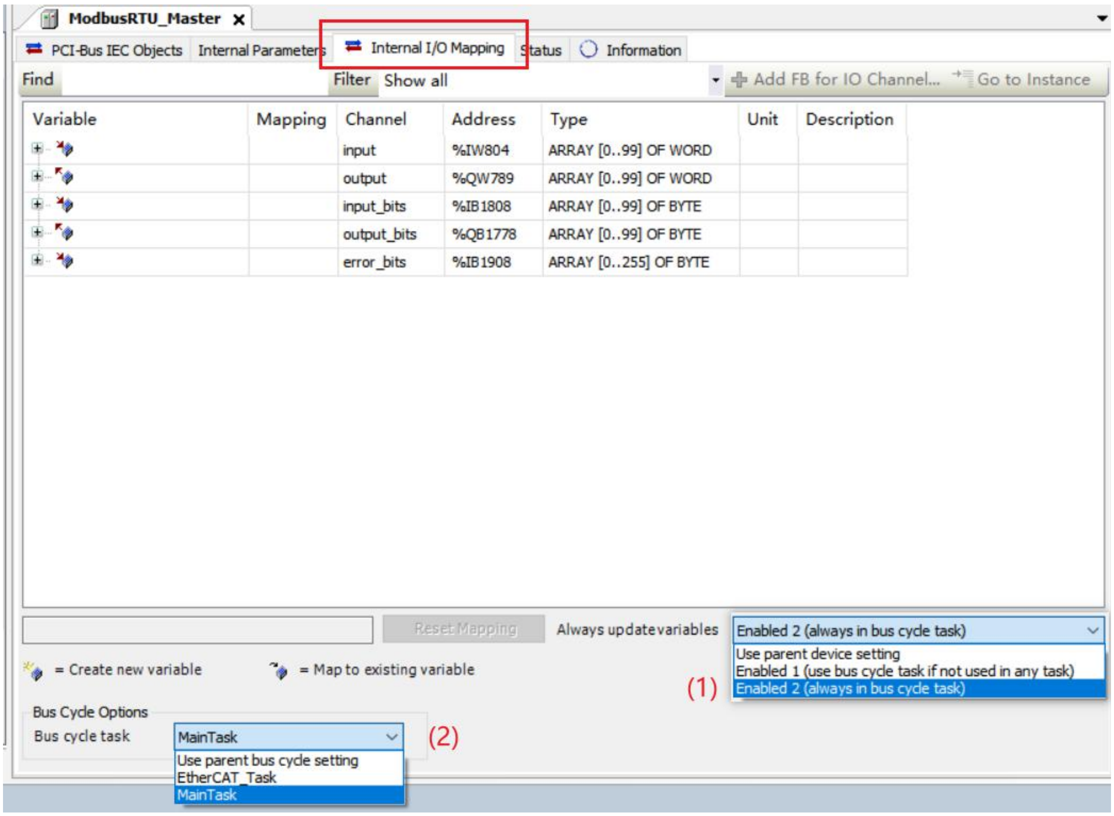
- Channel number: the channel currently in use is Channel 0;
- Slave_Add: from the station number;
- Reg_Add: from the station register address;
- Fun_Num: function code;
- Channel_Add: Transfer data using the first channels in the output/outpu_bit or receive (input/input_bit) of the Internal I/O map;
- Length: Data length (how many channels are occupied);

Note: Every time you change the Internal parameter to download, you need to power it up again to take effect, otherwise you will report an error.

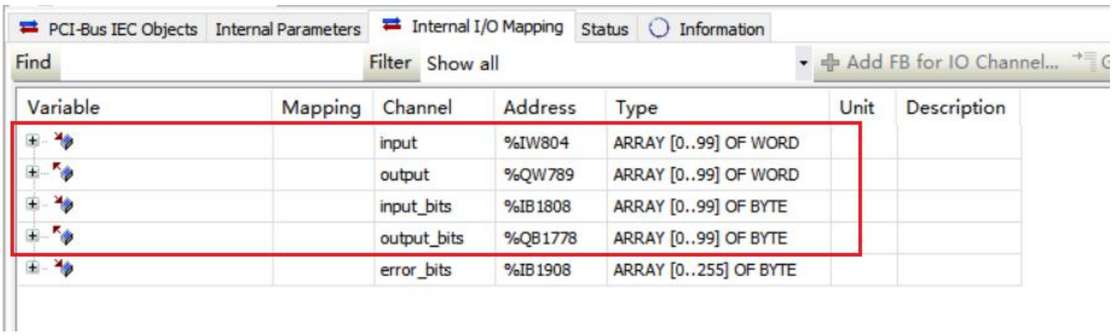
Address Association Variables (Internal I/O Mapping Introduction)

Click "Internal I/O Map" to see 4 IO mapping channels and one error viewing channel (error_bits);

- First, you need to select Enable 2 (always in the bus loop task);
- Then select the bus loop task to use, and the example here selects MainTask;



Here's a look at the IO channel:



In the ModBusRTU_Masterdevice, the mapping address is provided as follows:

Type	Channel
input	input[0] ~ input[2047]
output	output[0] ~ output[2047]
input_bit	input_bit[0]~input_bit[2047]
output_bit	ooutput_bit[0]~output_bit[2047]

- Modbus function codes 1, 2 corresponding to the channel is:nput_bit;
- Modbus function code 3, 4 corresponding channels are: input;
- Modbus function codes 5, 15 corresponding channels are: output_bit;
- Modbus function code 6, 16 corresponding channels are: output;

Attention:

1, variables according to their own data type to determine the number of occupied channels, such as INT-type variables occupy a WORD, REAL, DINT-type variables occupy two WORD, LREAL, LINT-type variables occupy 4 WORDS, and so on.

2, the associated address should be associated according to the channel's starting address, as shown below, the variable wants to associate to the channel input

Variable	Mapping	Channel	Address	Type	Unit	Description
		input	%IW804	ARRAY [0..99] OF WORD		
		input[0]	%IW804	WORD		
		input[1]	%IW805	WORD		
		input[2]	%IW806	WORD		
		input[3]	%IW807	WORD		
		input[4]	%IW808	WORD		
		input[5]	%IW809	WORD		

In a device, the io address is mapped to a variable in two ways:

Method 1: Map addresses in variable declarations, as shown below.

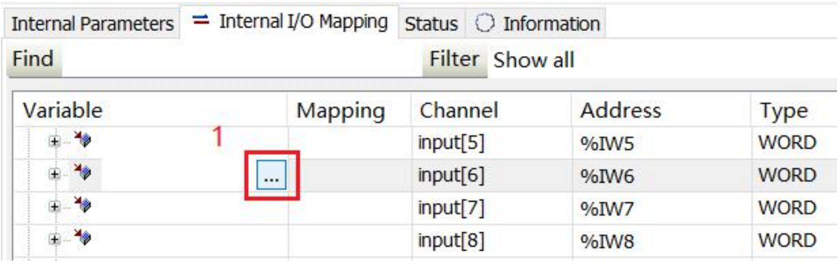
Input type:

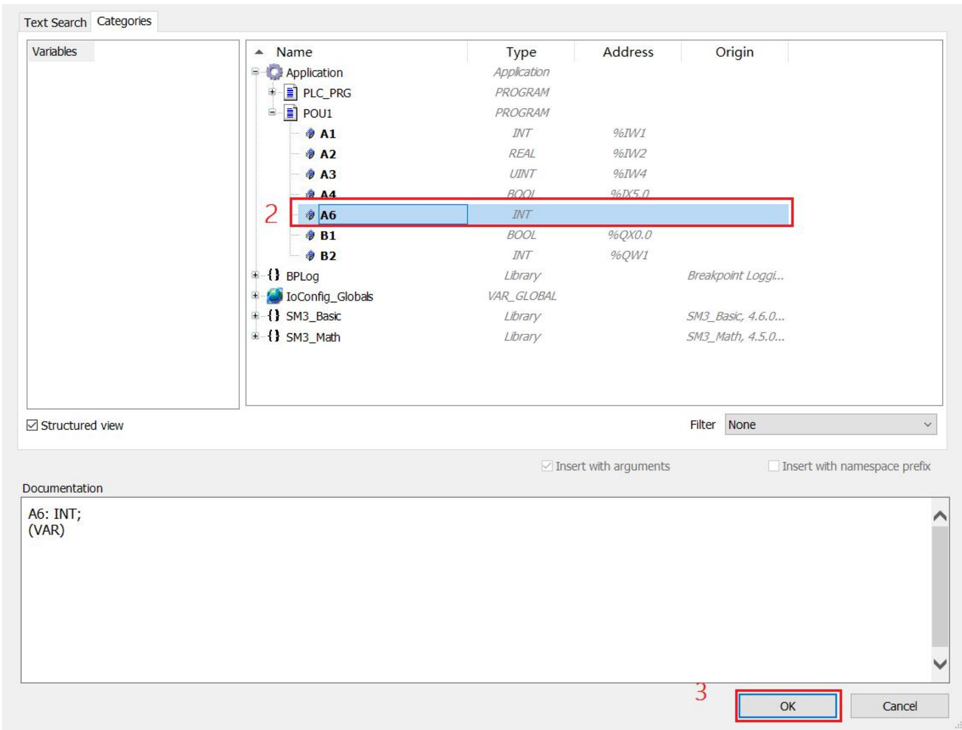
```
PROGRAM POU1
VAR
    //    input
    A1 AT %IW1:INT; //INT
    A2 AT %IW2:REAL; //REAL
    A3 AT %IW4:UINT;//UINT
    A4 AT %IX5.0:BOOL;//BOOL
    //    output
    B1 AT %QX0.0:BOOL;//BOOI
    B2 AT %QW1:INT:=10;
END_VAR
```

Table type:

Scope	Name	Address	Data type	Initialization	Comment	Attributes
VAR	A1	%IW1	INT			
VAR	A2	%IW2	REAL			
VAR	A3	%IW4	UINT			
VAR	A4	%IX5.0	BOOL			
VAR	B1	%QX0.0	BOOL			
VAR	B2	%QW1	INT	10		

Method 2: Select a variable in the io mapping list.





Note: When writing multiple words (above 16 bits) or multiple single-word parameters, be sure to define the variable address (i.e. method one) in your program, and you need to empty the occupied multi-digit channel, otherwise you will report an error.

Example: As shown below, a WORD type (32-bit) variable C3 is established and associated with channel 5;

```
C3 AT %QW5:DWORD:=65537;
```

After running the program, you can see that channel 5 can only put 16 bits of data, if you store 32 bits of variables in this channel need to occupy two channels, that is, channel 6 also occupied.

input	%IW0	ARRAY [0..99] OF WORD	
output	%QW0	ARRAY [0..99] OF WORD	
output[0]	%QW0	WORD	111
output[1]	%QW1	WORD	0
output[2]	%QW2	WORD	0
output[3]	%QW3	WORD	0
output[4]	%QW4	WORD	0
output[5]	%QW5	WORD	1
output[6]	%QW6	WORD	1
output[7]	%QW7	WORD	0

Here's how the Internal I/O map mis-views the channel (error_bits):

input	%IW0	ARRAY [0..99] OF WORD
output	%QW0	ARRAY [0..99] OF WORD
input_bits	%IB200	ARRAY [0..99] OF BYTE
output_bits	%QB200	ARRAY [0..99] OF BYTE
error_bits	%IB300	ARRAY [0..255] OF BYTE

Expand the channel and you can see the communication status of each communication channel (Modbus Channels):

input	%IW0	ARRAY [0..99] OF WORD		
output	%QW0	ARRAY [0..99] OF WORD		
input_bits	%IB200	ARRAY [0..99] OF BYTE		
output_bits	%QB200	ARRAY [0..99] OF BYTE		
error_bits	%IB300	ARRAY [0..255] OF BYTE		
error_bits[0]	%IB300	BYTE	1	ERROR
error_bits[1]	%IB301	BYTE	1	
error_bits[2]	%IB302	BYTE	1	
error_bits[3]	%IB303	BYTE	1	
error_bits[4]	%IB304	BYTE	1	
error_bits[5]	%IB305	BYTE	1	
error_bits[6]	%IB306	BYTE	1	
error_bits[7]	%IB307	BYTE	0	Normal
error_bits[8]	%IB308	BYTE	0	
error_bits[9]	%IB309	BYTE	0	

As shown above, when the current value of the channel is 1, it indicates a channel communication error, and when the current value is 0, it indicates normal communication;

Introduction to the use of function codes

The function codes (e.g.) available when the VE controller is the ModeBus Fun_Num are described below:

Function code (de-order)	Description of the function
1	Read the coil register
2	Read discrete input registers
3	Read hold register
4	Read the input register
5	Write a single coil register
6	Write a single hold register
15	Write multiple coil registers
16	Write multiple hold registers

- Function code 1 (read coil register):

Bit parameters used to read writeable (RW) in the master station, the master to the from the station;

Read a single bit parameter example: When reading a single bit parameter, Length (data length) is 1 because only 1 channel is used;

Slave_Add	WORD	1
Reg_Add	WORD	0
Fun_Num	WORD	1
Channel_Add	WORD	0
Length	WORD	1

Read multiple bit parameter examples: When reading 10 bit parameters with consecutive addresses in the station, Length (data length) is 10 because 10 channels are required;

Slave_Add	WORD	1
Reg_Add	WORD	0
Fun_Num	WORD	1
Channel_Add	WORD	0
Length	WORD	10

- Function code 2 (read discrete input register):

Used to read bit parameters in the master to the station that can only be read (R) from the master;

- Function code 3 (read hold register):

Used to read the word parameters that can be read and writeable (RW) in the master station to the master station;

Read a single word example: when reading a 16-bit single-word parameter, Length (data length) is 1, because only 1 channel is used;

Slave_Add	WORD	1
Reg_Add	WORD	0
Fun_Num	WORD	3
Channel_Add	WORD	0
Length	WORD	1

Read multiple single-word parameters or multi-word parameter examples: read from the station address consecutive 2 16-bit single-word parameters or 1 32-bit double-word parameters, Length (data length) is 2, because the need to occupy 2 channels;

Slave_Add	WORD	1
Reg_Add	WORD	0
Fun_Num	WORD	3
Channel_Add	WORD	0
Length	WORD	2

- Function code 4 (read input register):

Used to read word parameters that can only be read (R) from the master to the station in the master station, and to read multiple consecutive or multi-word parameters in the same way as function code 3.

- Function code 5 (write a single coil register):

A state write for a single bit parameter of a readable writeable (RW) from the master station;

- Function code 6 (write a single hold register):

For the primary station to write data to a single single word parameter of the station readable and writeable (RW);

- Function code 15 (write multiple coil registers):

Used for the primary station to state write to multiple address consecutive bit parameters of the station readable writeable (RW);

- Function code 16 (write multiple hold registers):

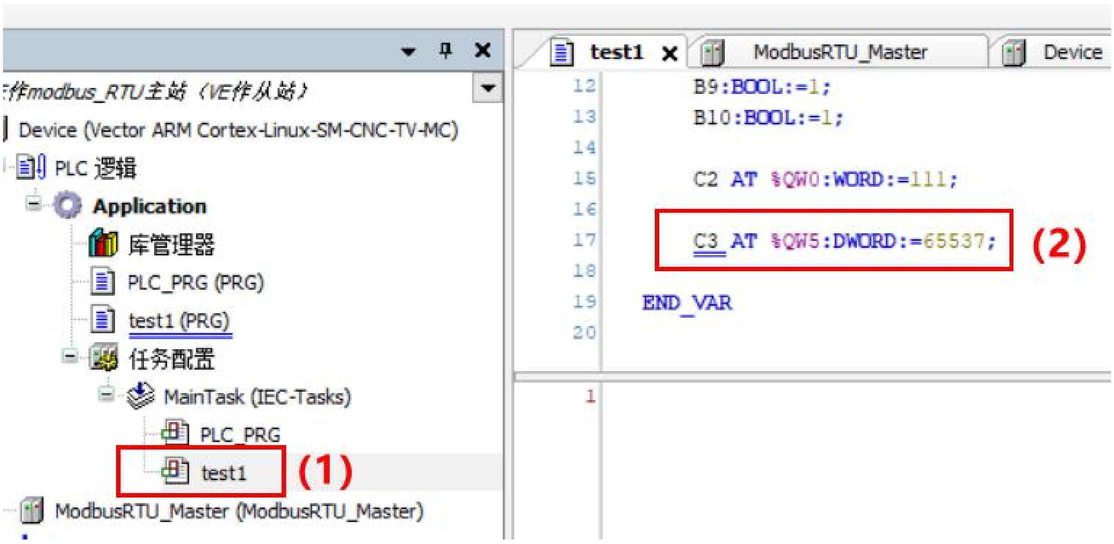
For the main station to write data to multiple address consecutive single-word or multi-word parameters of the readable and writeable (RW) from the station;

Example of use

The following example shows how a master can write a double word parameter to the 7th register address of a slave with station number 1:

(1) Double-click to open the POU to be used.

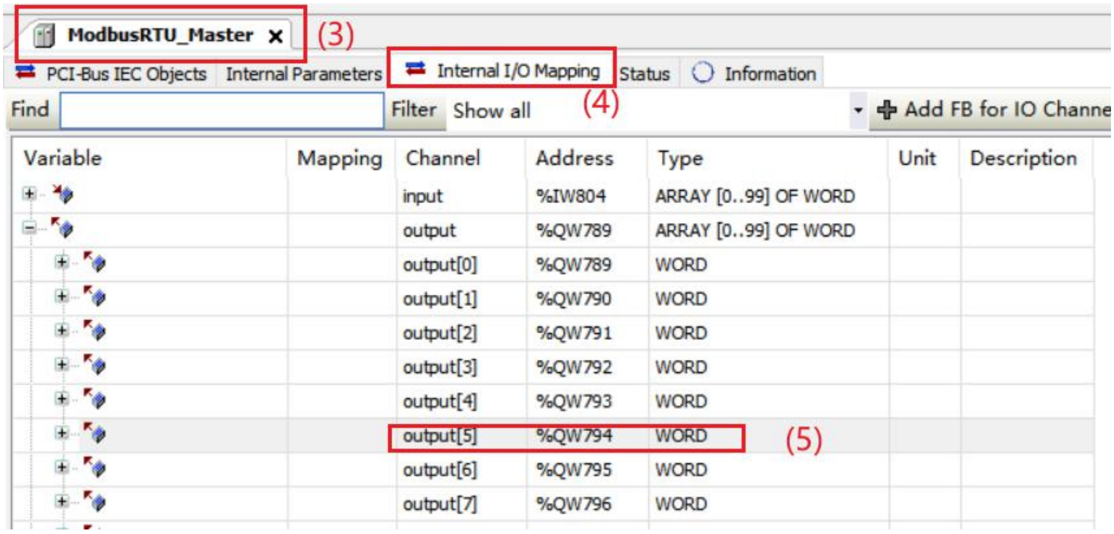
(2) Define a double-word variable and assign it an unoccupied address (here the example %QW5).;



(3) Double-click on "ModbusRTU_Master" to open it.

(4) Click on "Internal I/O mapping".

(5) Find the channel corresponding to the variable address just defined (%QW5) and learn that the channel number is 5 (output[5]);



(6) Click on "Internal parameters".

(7) Expand a communication channel.

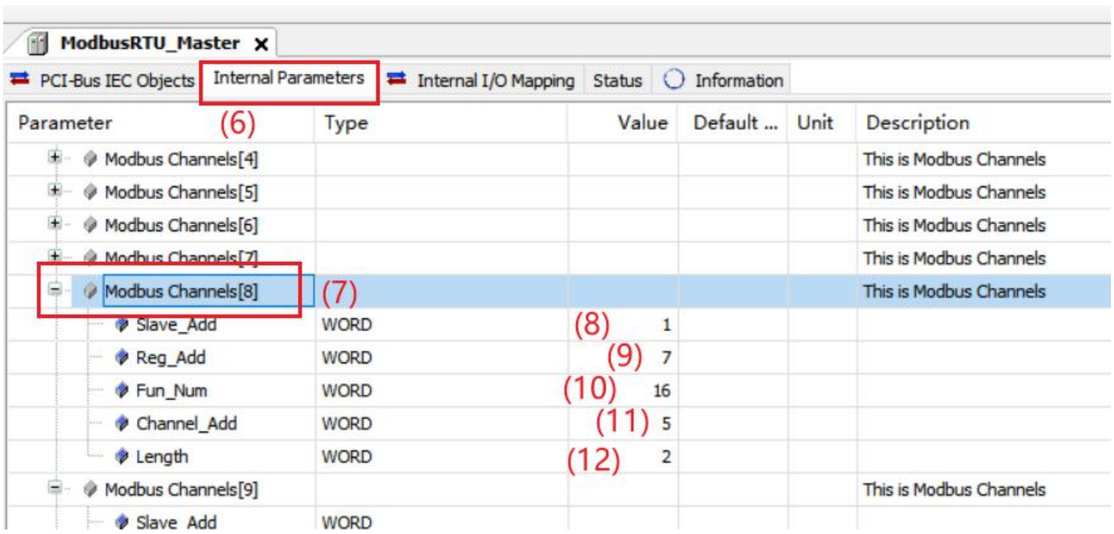
(8) The Slave_Add value is set to 1.

(9) The Reg_Add (slave register address) value is set to 7.

(10) The value of Fun_Num (function code) is set to 16, which means that multiple holding registers are written.

(11) The value of Channel_Add is written to the channel number corresponding to the variable address %QW5.0 queried in step (5): 5.

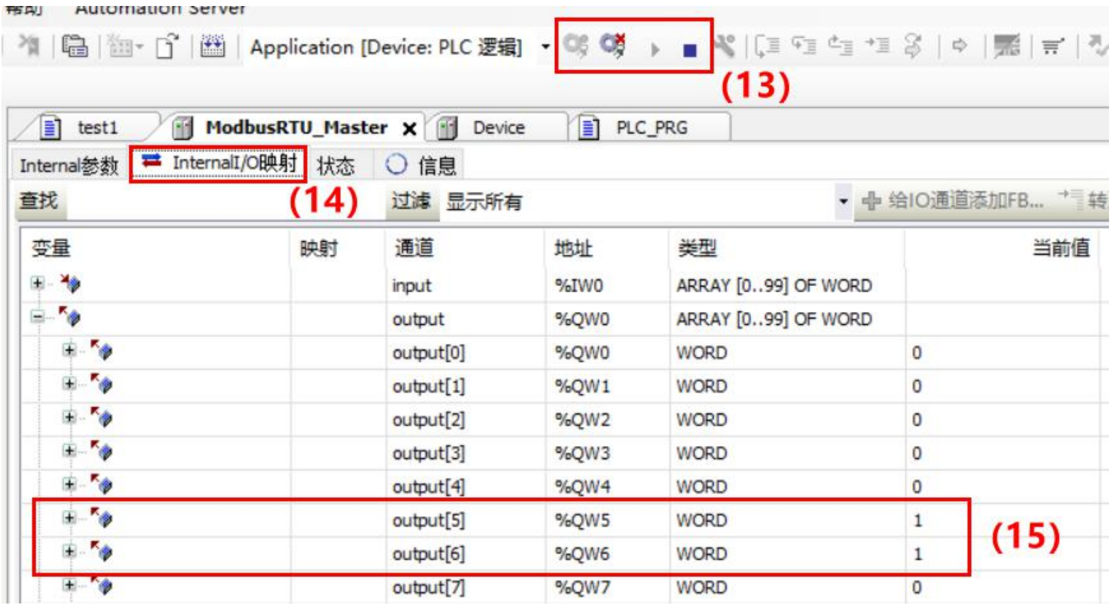
(12) The value of Length is written to 2, because the double word needs to occupy two channels;



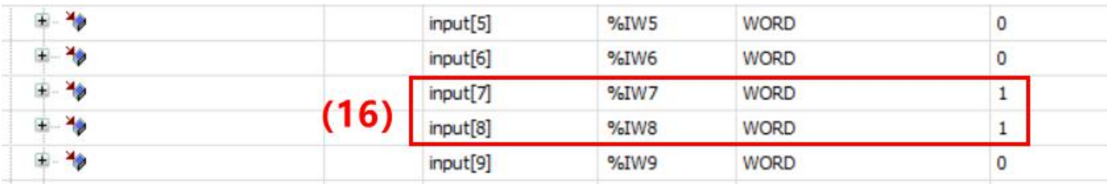
(13) Log in and run.

(14) Click on "Internal I/O Mapping".

(15) You can see that the data for variable C3 has been successfully sent;



(16) Turning on the slave you can see that the correct data has been received in the 7th register of the slave's input;

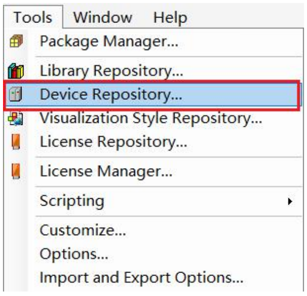


8.1.4 ModbusTCP_Master

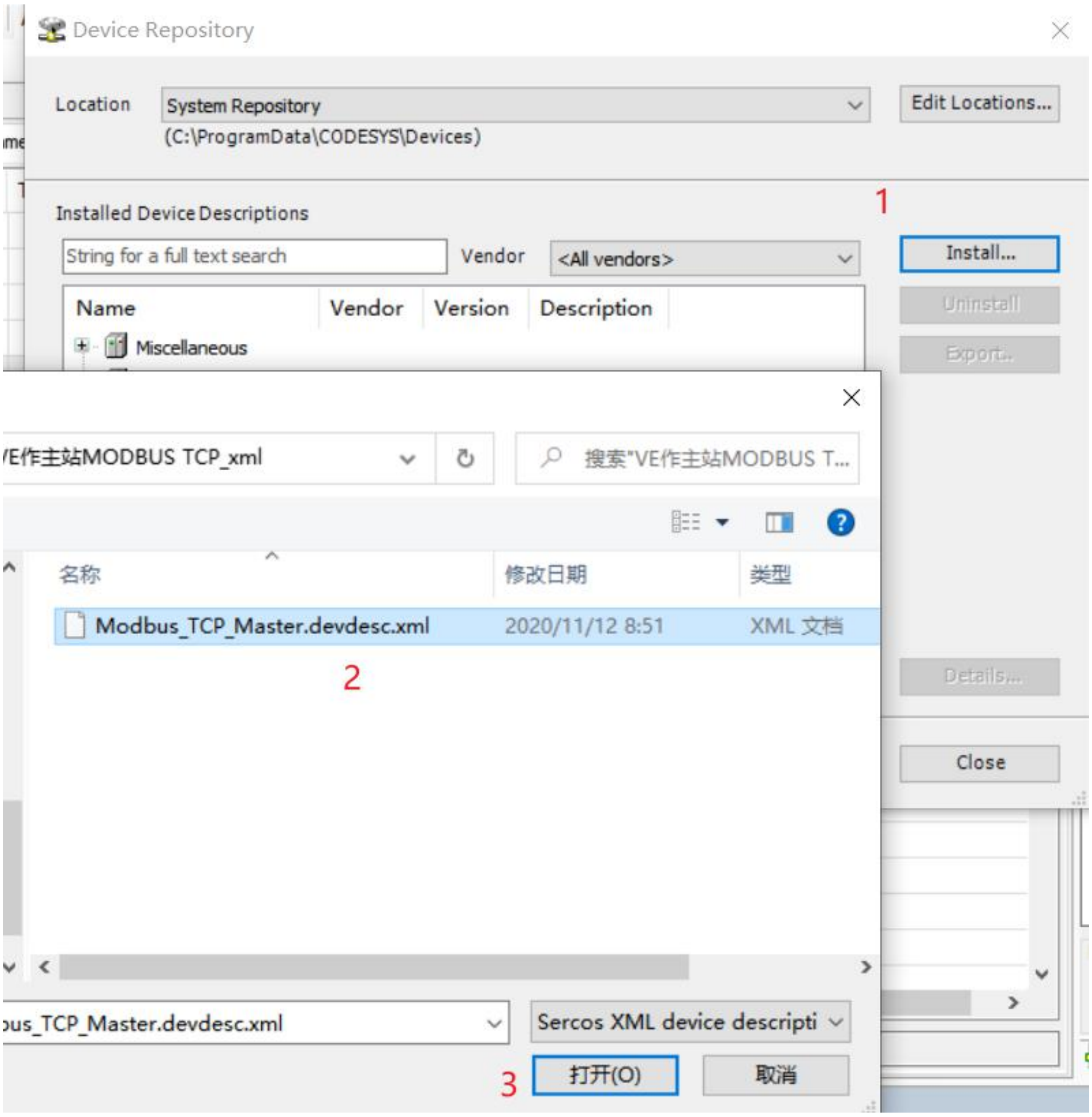
The VE motion controller supports standard ModeBusTCP communication, connected to a touch screen or switch via the EtherNet communication port. The following two VE as an example, through the switch, VE controller one for the host, one for the machine, from the machine installation configuration reference: ModBusTCP_Slave, from the machineIP address to: 192.168.1.122 (modification method reference: product configuration and module instructions), the host installation configuration steps are as follows:

Install the device description file

To use ModBusTCP_Master,first install the device by clicking onthe toolbar "Tools→Device Repository"



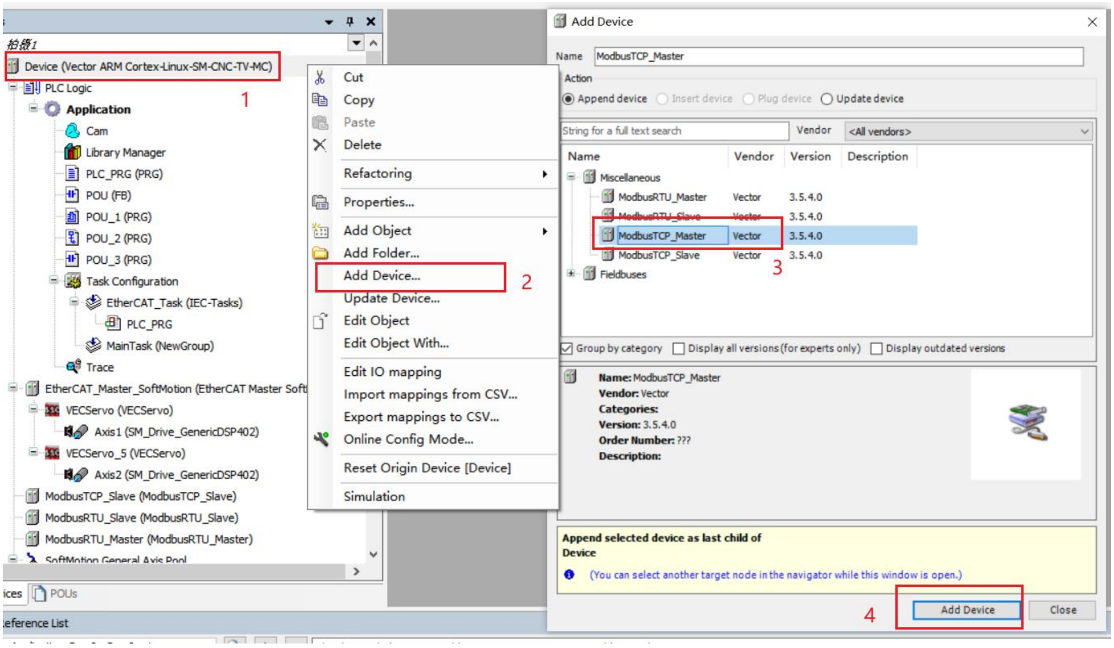
Then click "Install" to find the device description file"VEC ModBusTCP.xml Master"and select andclick Open



Displays that the installation was successful, which indicates that the device was installed successfully and can be added for use.

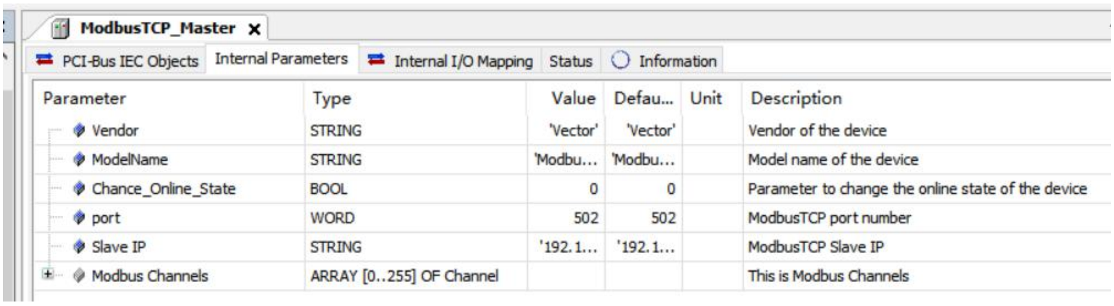
Add a device

After the new project is created, select Right-click → Device Add Device Mod→BusTCP_Master Add Device toconfirmthat the→TCPdevice is added and that the VE controller will beadded to the project as a fromstation.



Host parameter settings

When you're done, double-click to open asfollows,usTCP_the Mod B and Master interfaces are as follows:

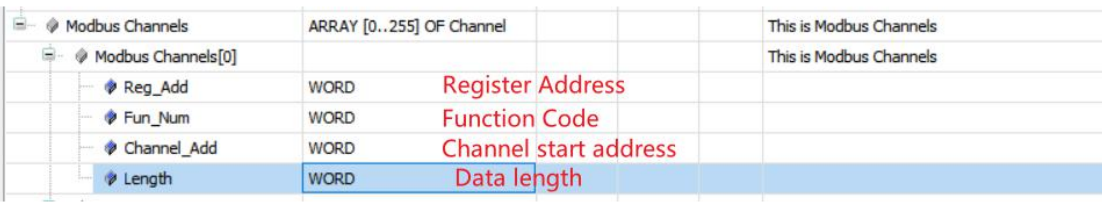


The communication parameters between the device and the from the station include:

(1) From the station port number, ModbusTCP_Slave default port number is 502;

(2) From the station IP address, ModbusTCP_Slave's IP address changed to: 192.168.1.122, if not changed will conflict with the host IP.

The host device consists of 256 Modbus channels, each of which can be set with a separate Modbus function code, register address, channel address, and configuration length in WORD, as shown below.

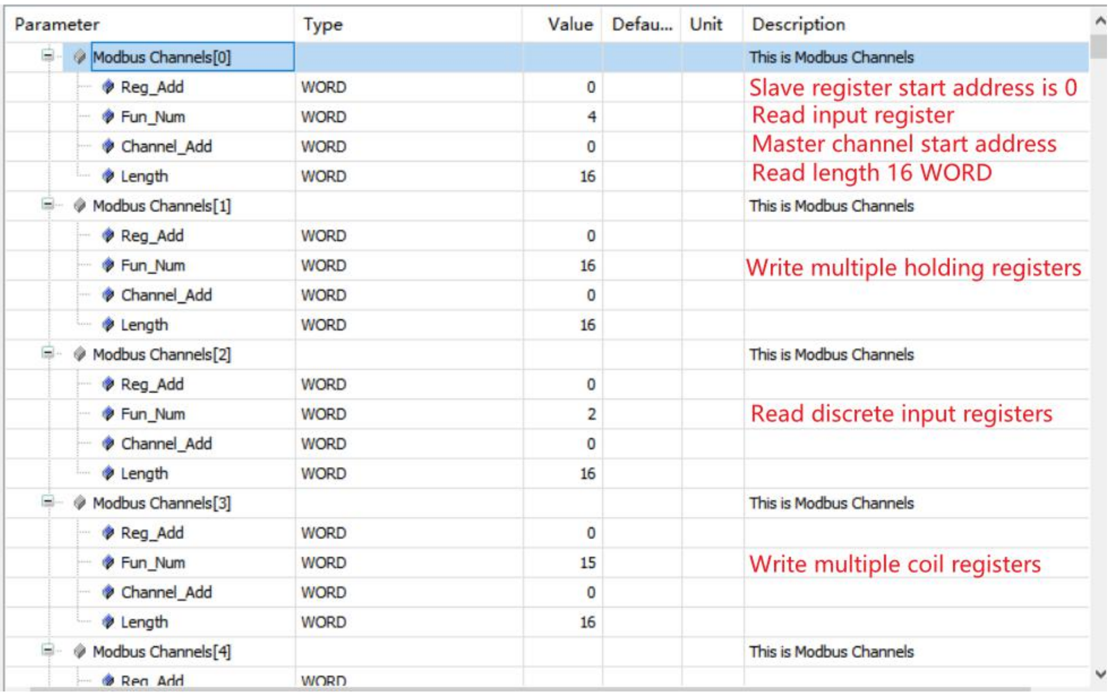


For feature codes, ModbusTCP_supported by Master are as follows:

The function code	Describe	Bit/word operation	The number of

			operations
1	Read the coil register	Bit operation	single or more
2	Read discrete input registers	Bit operation	single or more
3	Read hold register	Word operation	single or more
4	Read the input register	Word operation	single or more
5	Write a single coil register	Bit operation	Single
6	Write a single hold register	Word operation	Single
15	Write multiple coil registers	Bit operation	Multiple
16	Write multiple hold registers	Word operation	Multiple

This example configures the function code and other channel parameters as follows:



Parameter	Type	Value	Defau...	Unit	Description
Modbus Channels[0]					This is Modbus Channels
Reg_Add	WORD	0			Slave register start address is 0
Fun_Num	WORD	4			Read input register
Channel_Add	WORD	0			Master channel start address
Length	WORD	16			Read length 16 WORD
Modbus Channels[1]					This is Modbus Channels
Reg_Add	WORD	0			
Fun_Num	WORD	16			Write multiple holding registers
Channel_Add	WORD	0			
Length	WORD	16			
Modbus Channels[2]					This is Modbus Channels
Reg_Add	WORD	0			
Fun_Num	WORD	2			Read discrete input registers
Channel_Add	WORD	0			
Length	WORD	16			
Modbus Channels[3]					This is Modbus Channels
Reg_Add	WORD	0			
Fun_Num	WORD	15			Write multiple coil registers
Channel_Add	WORD	0			
Length	WORD	16			
Modbus Channels[4]					This is Modbus Channels
Reg_Add	WORD				

Inaddition, the description of the I/O mapping channel is consistent with the "address association variable" ("Internal I/O mapping") description of **8.1.3**

From the machine parameter settings

The from-machine parameters are set as follows, and the default port number for the from-machine is: 502, which is not changed here.

ModbusTCP_Master

PCI-Bus IEC Objects | Internal Parameters | Internal I/O Mapping | Status | Information

Parameter	Type	Value	Defau...	Unit	Description
Vendor	STRING	'Vector'	'Vector'		Vendor of the device
ModelName	STRING	'Modbu...	'Modbu...		Model name of the device
Chance_Online_State	BOOL	0	0		Parameter to change the online state of the device
port Port number	WORD	502	502		ModbusTCP port number
Slave IP	STRING	'192.1...	'192.1...		ModbusTCP Slave IP
Modbus Channels	ARRAY [0..255] OF Channel				This is Modbus Channels

Online monitoring

(1) 4 function code

Variable	Ma...	Channel	Address	Type	Unit
		input	%IW0	ARRAY [0..256] OF WORD	
		output	%QW0	ARRAY [0..256] OF WORD	
		output[0]	%QW0	WORD	1
		output[1]	%QW1	WORD	2
Slave		output[2]	%QW2	WORD	3
		output[3]	%QW3	WORD	4
		output[4]	%QW4	WORD	5
		output[5]	%QW5	WORD	6
		output[6]	%QW6	WORD	0

		input	%IW278	ARRAY [0..99] OF WORD	
		input[0]	%IW278	WORD	1
Master		input[1]	%IW279	WORD	2
		input[2]	%IW280	WORD	3
		input[3]	%IW281	WORD	4
		input[4]	%IW282	WORD	5
		input[5]	%IW283	WORD	6
		input[6]	%IW284	WORD	0

(2) 16 function code

		input	%IW278	ARRAY [0..99] OF WORD	
		output	%QW150	ARRAY [0..99] OF WORD	
		output[0]	%QW150	WORD	7
		output[1]	%QW151	WORD	8
Master		output[2]	%QW152	WORD	9
		output[3]	%QW153	WORD	10
		output[4]	%QW154	WORD	11
		output[5]	%QW155	WORD	0

		input	%IW0	ARRAY [0..256] OF WORD	
		input[0]	%IW0	WORD	7
		input[1]	%IW1	WORD	8
Slave		input[2]	%IW2	WORD	9
		input[3]	%IW3	WORD	10
		input[4]	%IW4	WORD	11
		input[5]	%IW5	WORD	0

(3) 2 function code

Slave

input	%IW0	ARRAY [0..256] OF WORD	
output	%QW0	ARRAY [0..256] OF WORD	
input_bits	%IB514	ARRAY [0..256] OF BYTE	
output_bits	%QB514	ARRAY [0..256] OF BYTE	
output_bits[0]	%QB514	BYTE	1
output_bits[1]	%QB515	BYTE	1
output_bits[2]	%QB516	BYTE	1
output_bits[3]	%QB517	BYTE	0
output_bits[4]	%QB518	BYTE	1
output_bits[5]	%QB519	BYTE	1
output_bits[6]	%QB520	BYTE	1

Master

input	%IW278	ARRAY [0..99] OF WORD	
output	%QW150	ARRAY [0..99] OF WORD	
input_bits	%IB756	ARRAY [0..99] OF BYTE	
input_bits[0]	%IB756	BYTE	1
input_bits[1]	%IB757	BYTE	1
input_bits[2]	%IB758	BYTE	1
input_bits[3]	%IB759	BYTE	0
input_bits[4]	%IB760	BYTE	1
input_bits[5]	%IB761	BYTE	1
input_bits[6]	%IB762	BYTE	1

(4) 15 function code

Master

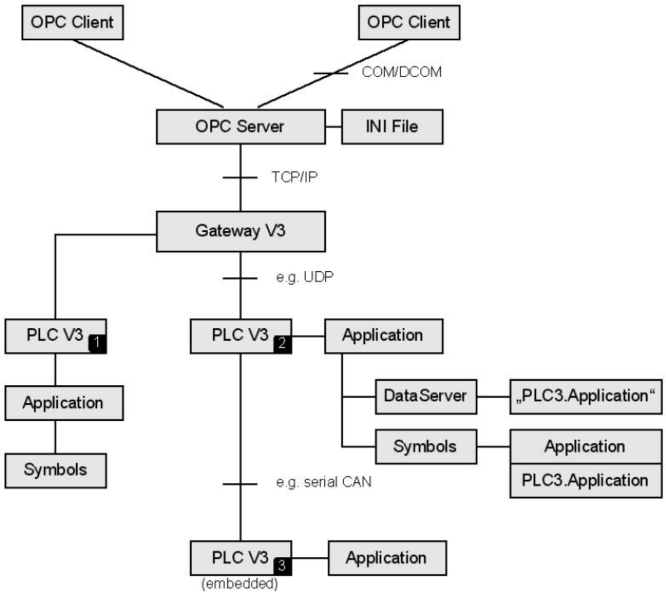
input	%IW278	ARRAY [0..99] OF WORD	
output	%QW150	ARRAY [0..99] OF WORD	
input_bits	%IB756	ARRAY [0..99] OF BYTE	
output_bits	%QB500	ARRAY [0..99] OF BYTE	
output_bits[0]	%QB500	BYTE	1
output_bits[1]	%QB501	BYTE	0
output_bits[2]	%QB502	BYTE	1
output_bits[3]	%QB503	BYTE	0
output_bits[4]	%QB504	BYTE	1
output_bits[5]	%QB505	BYTE	0
output_bits[6]	%QB506	BYTE	1
output_bits[7]	%QB507	BYTE	0
output_bits[8]	%QB508	BYTE	1

Slave

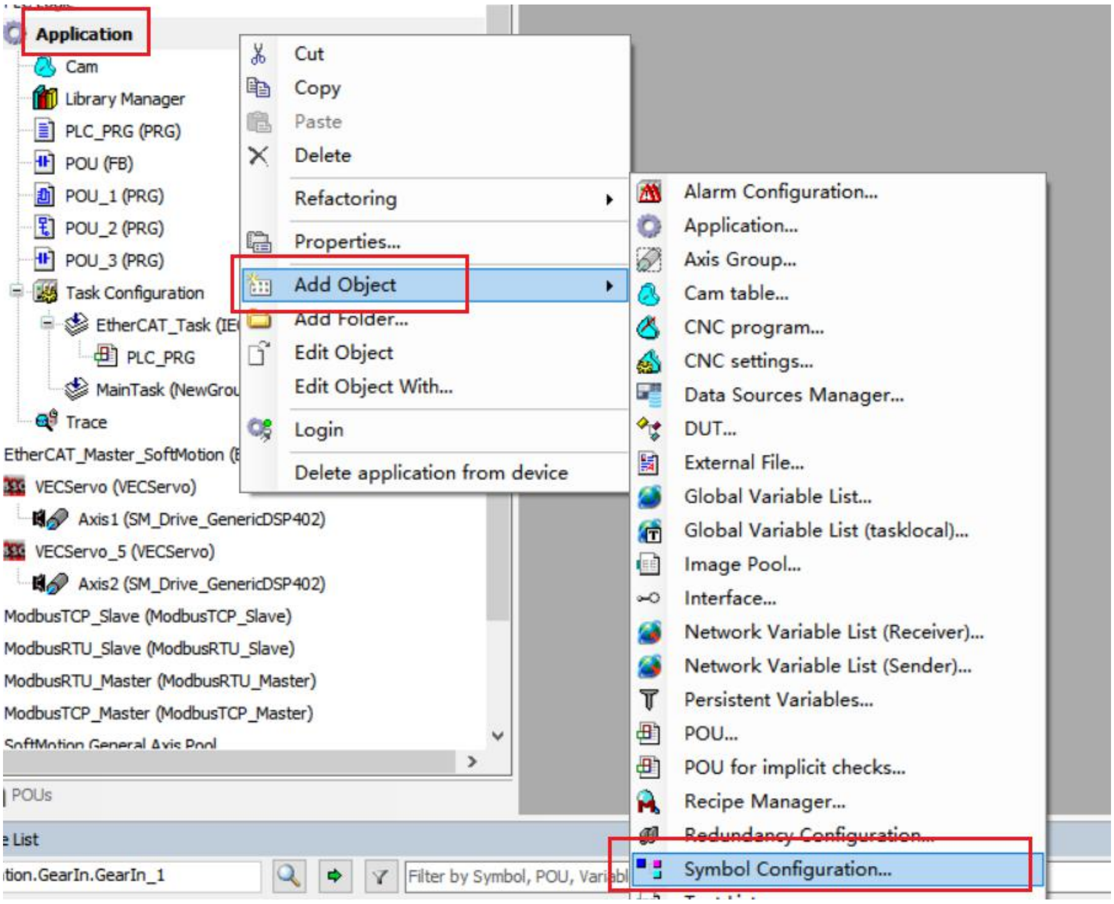
input	%IW0	ARRAY [0..256] OF WORD	
output	%QW0	ARRAY [0..256] OF WORD	
input_bits	%IB514	ARRAY [0..256] OF BYTE	
input_bits[0]	%IB514	BYTE	1
input_bits[1]	%IB515	BYTE	0
input_bits[2]	%IB516	BYTE	1
input_bits[3]	%IB517	BYTE	0
input_bits[4]	%IB518	BYTE	1
input_bits[5]	%IB519	BYTE	0
input_bits[6]	%IB520	BYTE	1
input_bits[7]	%IB521	BYTE	0
input_bits[8]	%IB522	BYTE	1

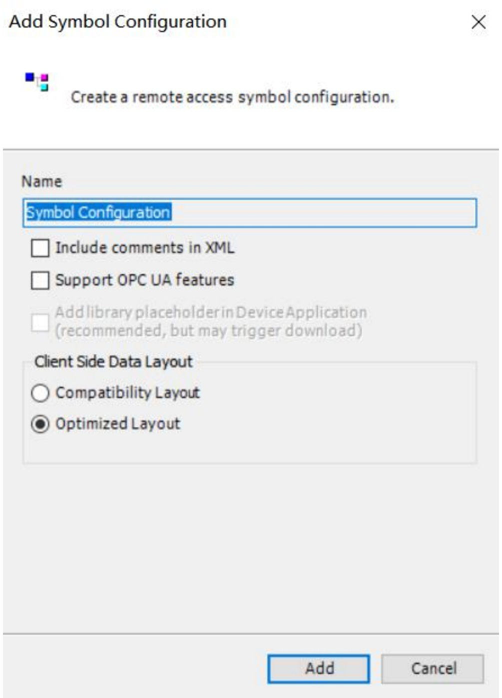
8.1.5 OPCserver

OPC Server Architecture：

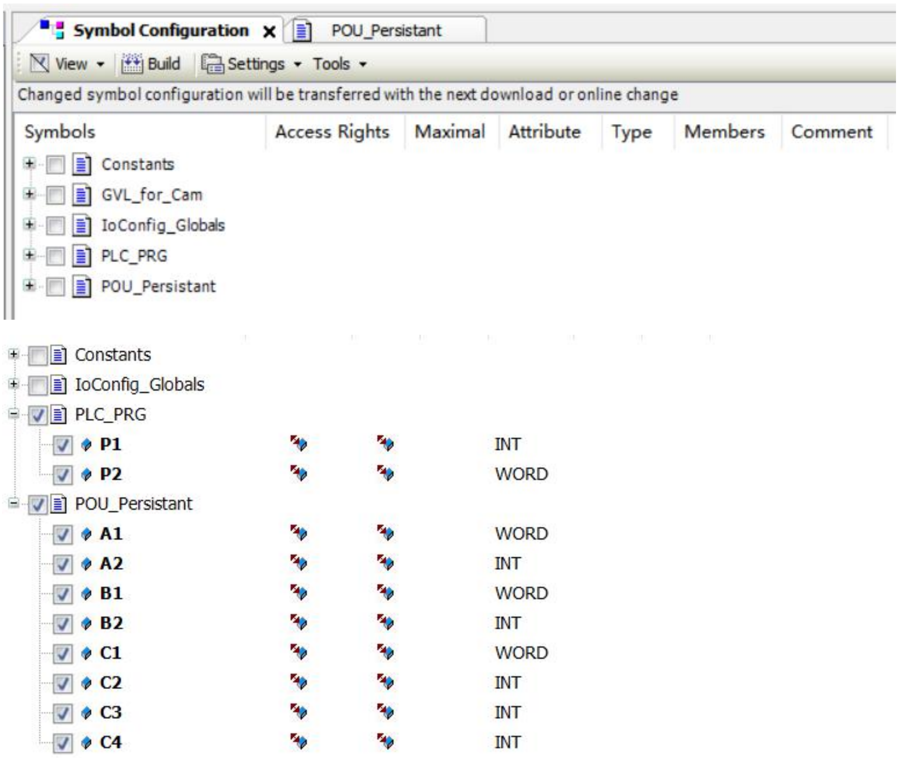


1、Add "symbols configuration" to "Applications" in the background of the program

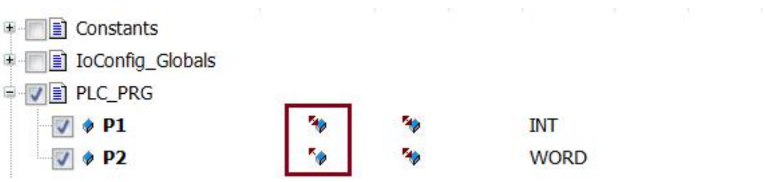




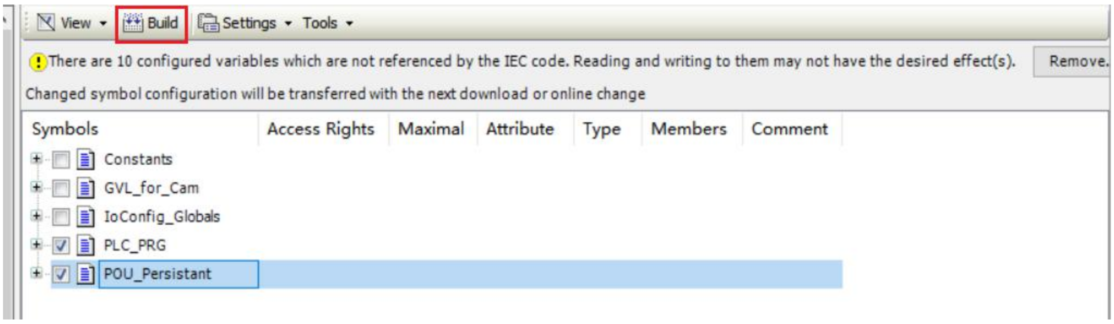
2. Select as POCsever to communicate data with PLC



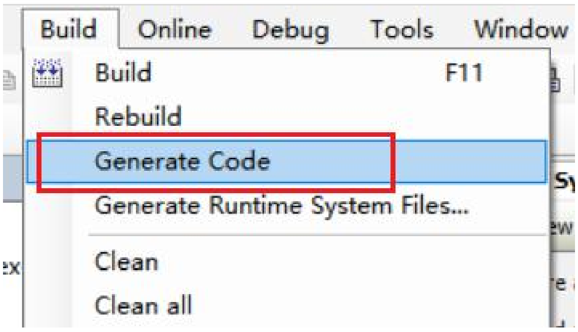
3. Click on the arrow under "Access Permissions" to select the permission



4、After the selection is complete, click "Compile"

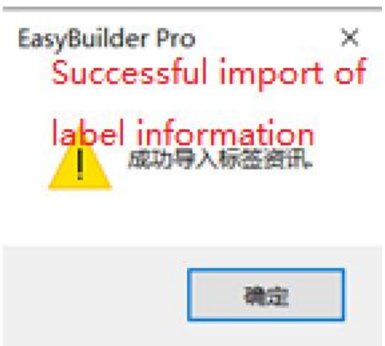


5, after confirmation, click on the toolbar "compile →generated code", the software automatically in the engineering directory to establish the corresponding "project name." Device.Application.xmlfile.

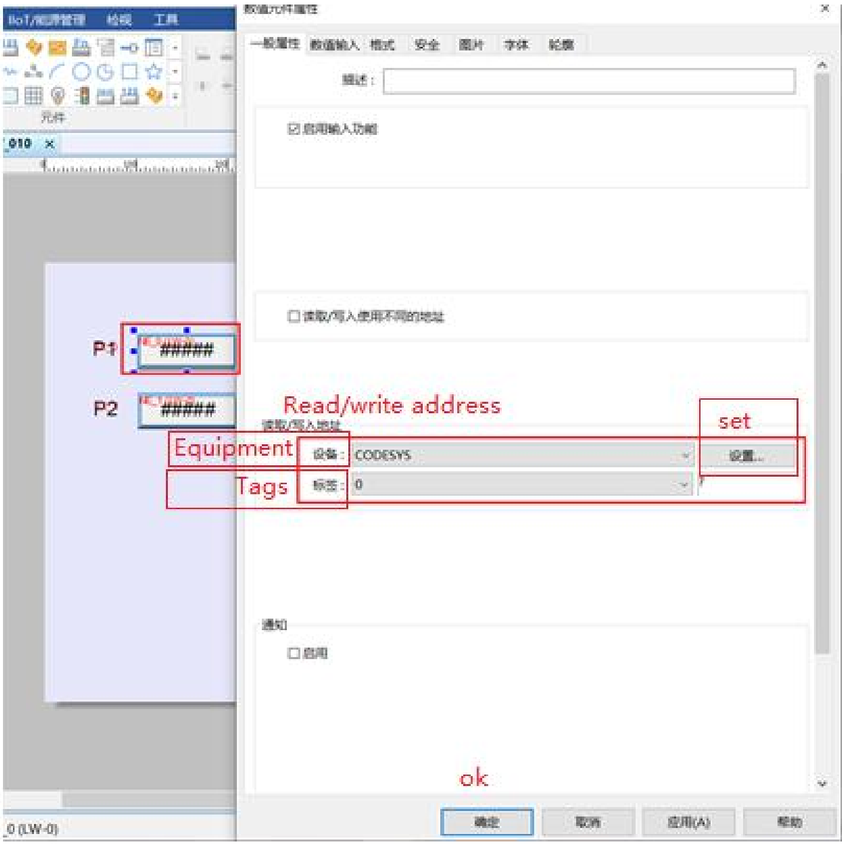


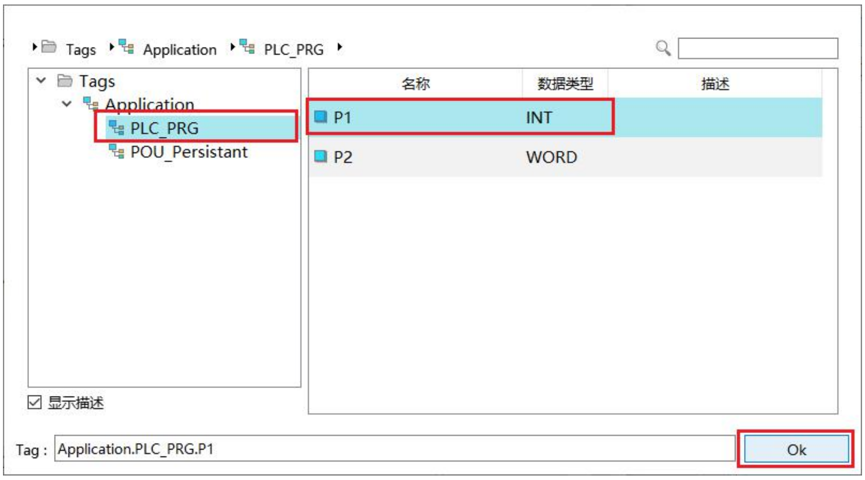
6, connect the VE controller Ethernet and touch screen network port, open the touch screen (support OPCserver) software interface, here to Theron pass touch screen as an example configuration, as follows, the construction project, equipment configuration as follows

7, click on the device, click on the "import label" to find the newly established file "project name. Device.Application.xml,select Import



8, open the touch screen component properties, as follows, the device selects the established "CODESYS", the label selects the corresponding variable, as shown in the following image



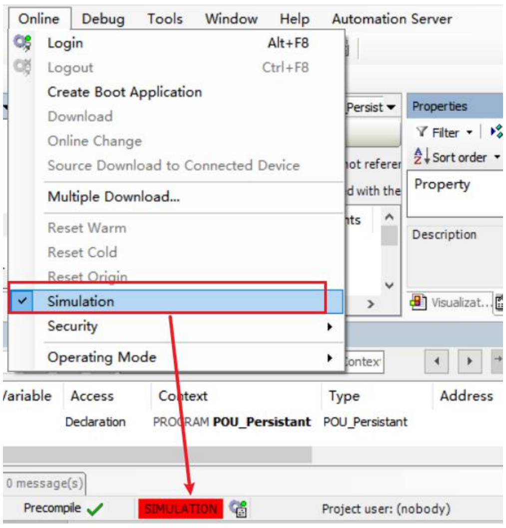


After selecting the corresponding tab, the HMI components can be associated with program variables.

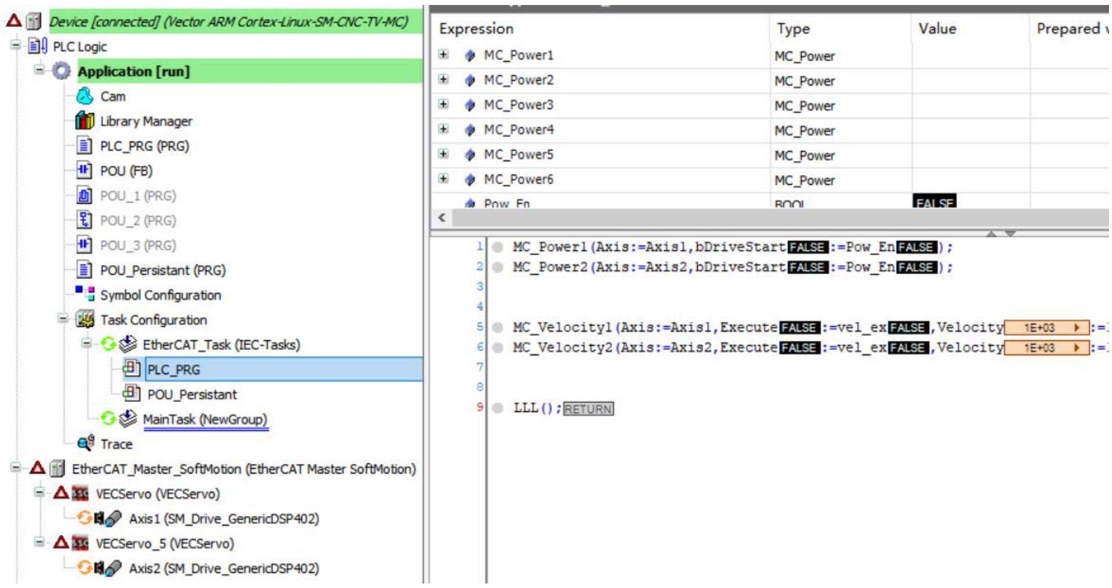
8.2 Simulation and debugging

8.2.1 Simulate the VE controller

When the user is programming debugging, they may not have VE controller hardware at hand, and codeSYS simulation can be used to debug the logic of the user program. The method of turning on the simulation function is as follows, →click on the online simulation, in the simulation state, the simulation status prompt with red font under the programming software.



In the simulation state, you can click "compile" the user program, "log in" to the controller, and then click "start", the user program loaded into the COMPUTER emulator, you can actually access the controller to monitor the user program, force modify the operation of parameters, observe the user program performance, as shown in the following image:



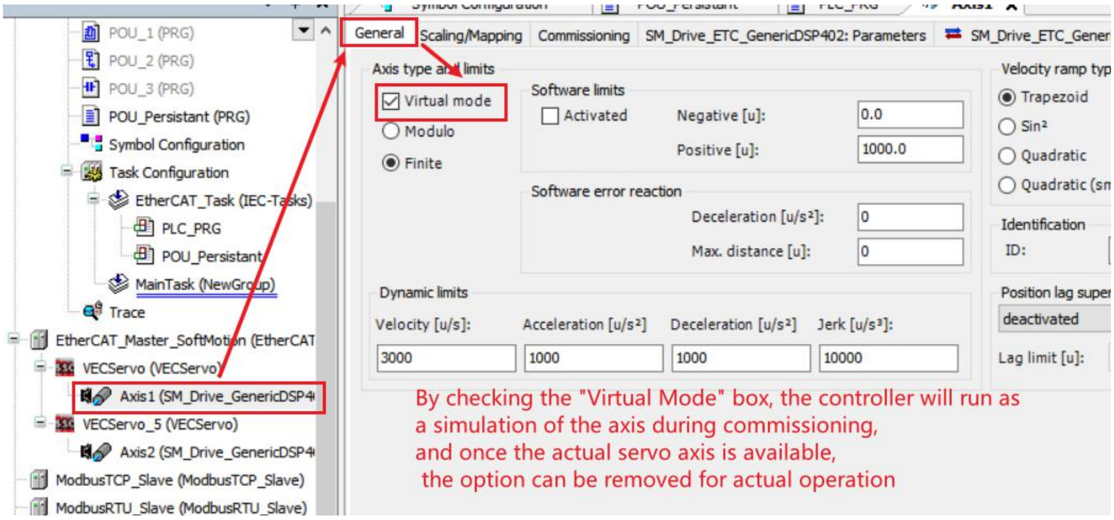
Attention:

1, here can not be the operation of the network bus simulation, but can be forced on the servo axis data structure parameters, you can still observe the execution logic of the program, check the implementation of the program.

2,"login", you can click "run", "stop" to execute the user program, need to modify the user program, to "exit" the login state.

8.2.2 Simulate servo drives

When writing a debugging MC operation application, the programmer has a VE controller on hand, but no servo drive, or does not have a sufficient number of servo drives, to debug debug the user program, you can use the "virtual axis" way to replace the servo drive real axis, as shown in the following illustration:



In programming debugging, if the number of servo axes accessed and the number configured in the user program is different, the system will alarm, can not be normal debugging, if connected to this virtual axis, the system will not alarm, but in the software simulation of the servo way of operation. You can visually see the "running" status of the axis and verify the correctness of our MC control procedures.

The virtual axis is also an axis, although it is a "virtual axis", but the operation logic of the axis state still needs to be programmed according to the state transfer logic in the PLCopen specification, such as the need to run MC_Power before running, the error after the MC_Reset and so on, so that we can debug and exclude logical errors in the user program.

If the actual servo axis is connected, you can simply cancel the "virtual axis mode" of the corresponding axis in the figure above and you will be ready to function properly.

8.3 Security management and user rights settings

Codesys can effectively manage and set up security for engineering files and devices, and this document focuses on the security settings for engineering files, security settings for devices, and permission settings for POU.

8.3.1 Device login permissions settings

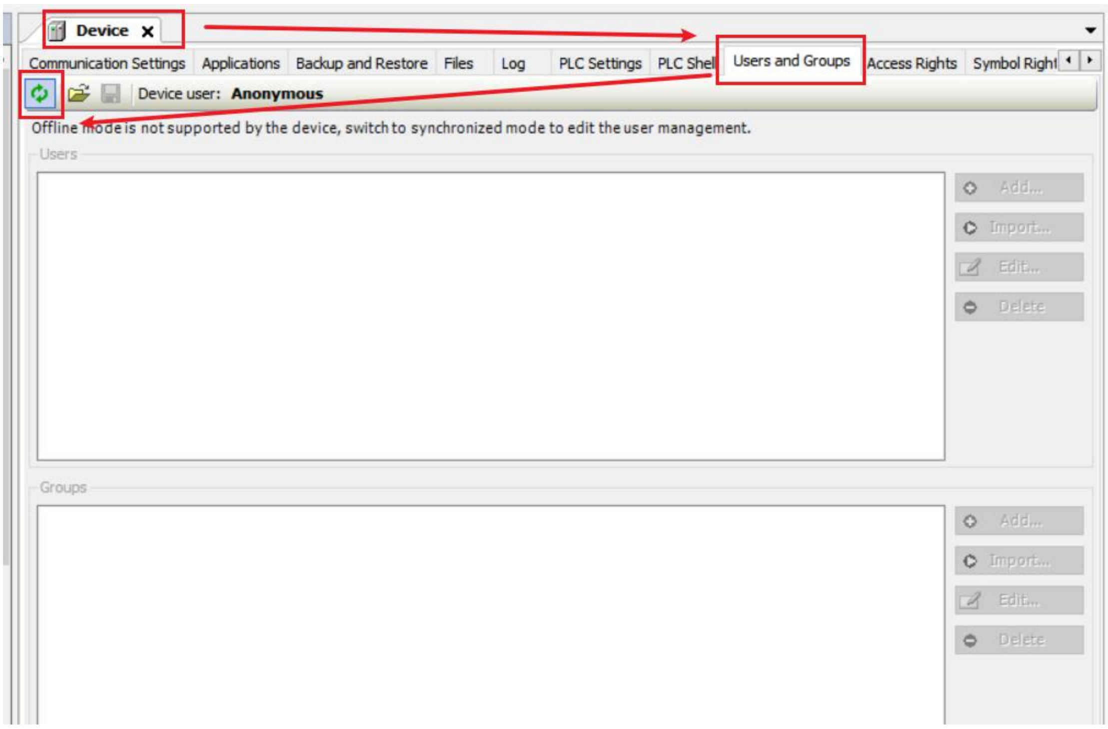
To minimize exposure to PLC and control networks onopen networks and the Internet (letting someone else sign in to your PLC),Codesys can set the device's login password to keep device data secure.

The following describes the device username and password settings, the method of logging on to the device and the method of canceling the device login password.

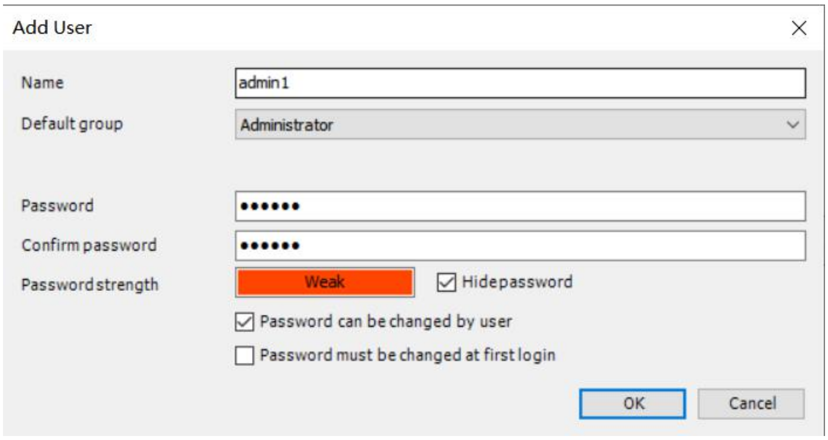
Add the user and password

Mode 1:

(1) Scan the device that needs to set the password first, and click Sync to see the user in the device by clicking "Sync" in the Device directory, as shown.

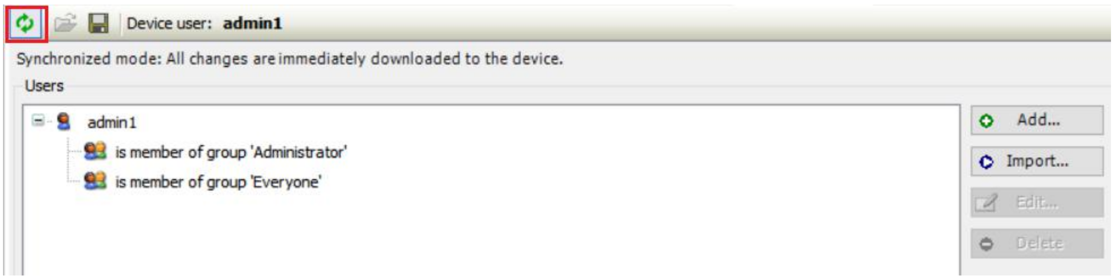


(2) Click Add to enter the username and password, as shown in the figure.



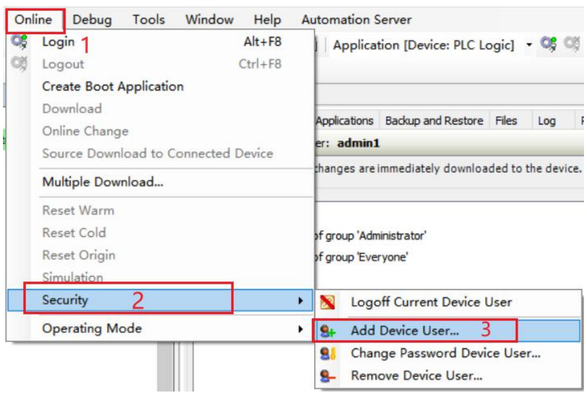
(3) When the user is added, the Everyone user needs to be deleted and click on the "Following" so that the user and password set will take effect, as shown in the figure. Be sure to remember one of the users' passwords before deleting Everyone, or you'll need to re-swipe your machine to sign in to your device.

(4) After the download is complete, you can view the user information status of the device by clicking Sync, as shown in the figure.

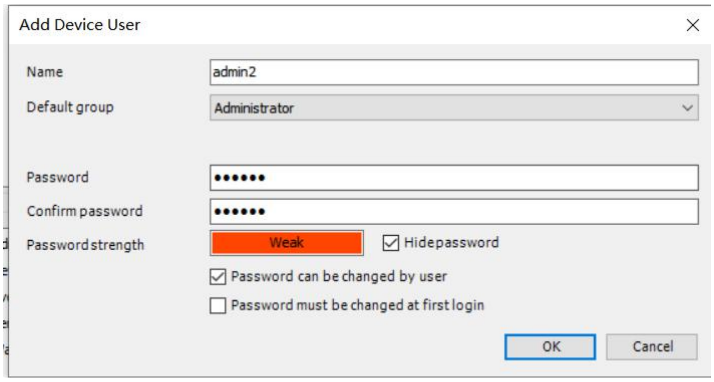


Mode 2:

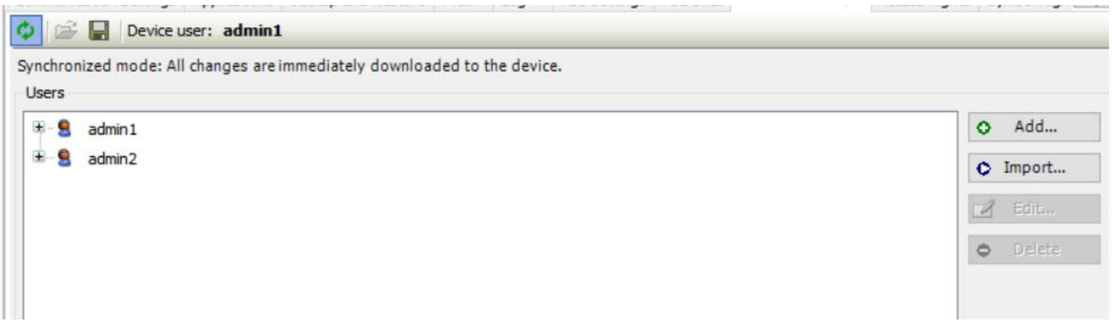
(1) Log in to the device, in the menu bar "Security" select "Add online users" to set the user name and password, as shown in the figure.



(2) Follow the prompt to log in again, the user password is effective, as shown in the figure.

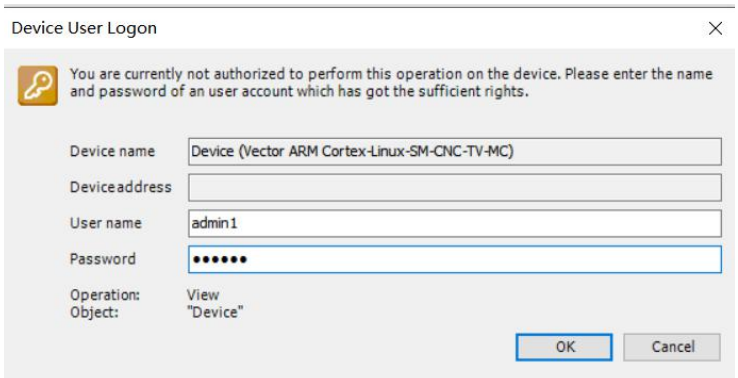


(3) At this point, sync the device userinformation in the same wayas (4) in Mode 1, you can see that the Everyone user has been deleted, as shown in the figure.



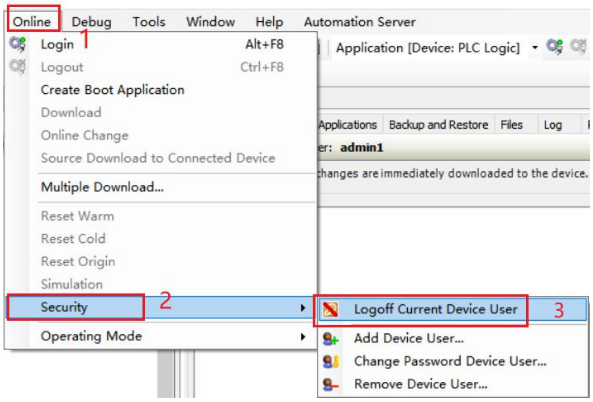
Sign in to the device

After you add user information as described above, you can sign in to the device with any of the added usernames and passwords, as shown in the figure.



Exit the current user

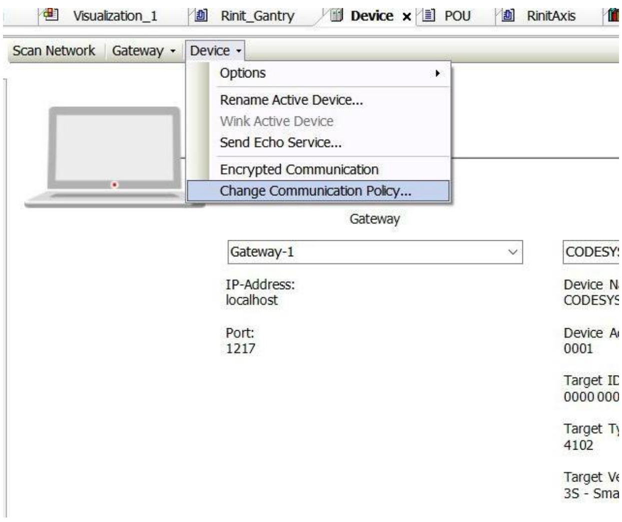
Once you're signed in to your device, you can opt out of the current user by selecting Sign out of the current online user in the menu bar, Security



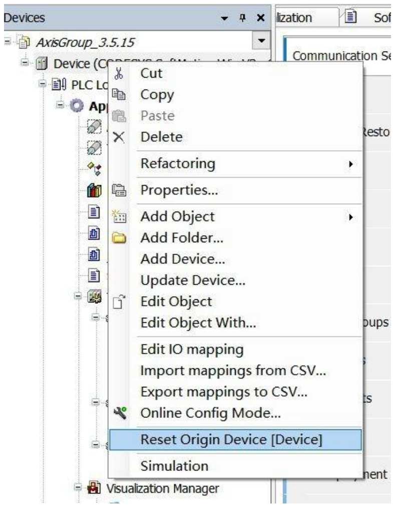
Cancel your account password login

Preferred to log in to your current device with your account password, and then do the following:

(1), click "Change Communication Policy" under "Device" in the "Communication Settings" tab of the device window, change to "Optional XXXXXXXXX" in the window thatappears, and click "OK" to confirm.



2, in the left side of the software DeviceTree window to Device right-click, select "Reset Origin Device", and so on the device program cleared and then reconnect the download program will not need an account and password.

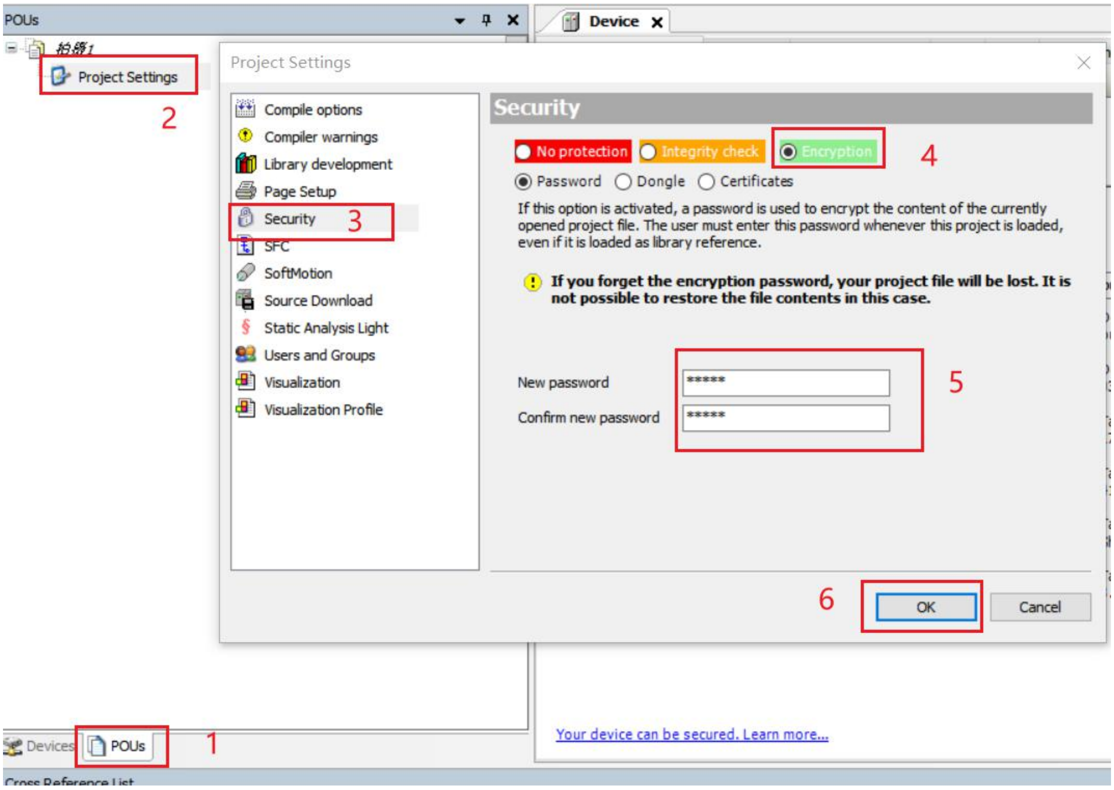


8.3.2 Project file security settings

The written engineering files can be encrypted, only to obtain the correct engineering file password can open the project files, to ensure the security of the project files. The following describes the setting of the project file password and how to cancel the password.

Note: Forgetting your password won't get it back!

(1) Password settings: click on the menu bar "Engineering" → "Engineering Settings" → "Security" to set the project file password, as shown in the figure.



(2) Every time you open an encrypted file, you'll have a prompt like this.

(3) To cancel the password, only need to log in to the project, will (1) in the"enabling engineering file encryption" check to remove it.

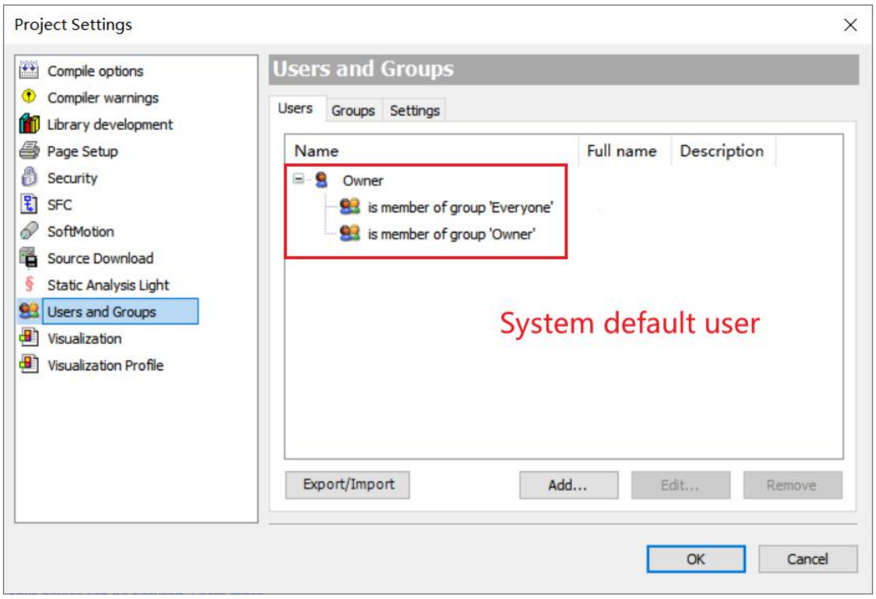
8.3.3 POU permission settings

User and group descriptions

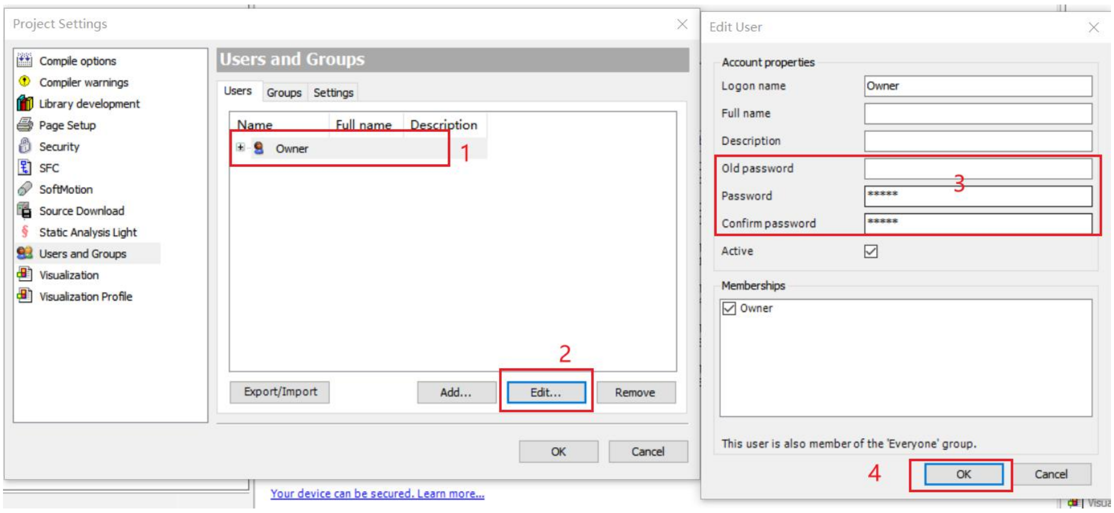
A project file contains multiple OUUs that set different permissions for different users. When setting permissions on a POU, you need to understand the difference between the 'Owner' group and the 'Everyone' group, first by describing the difference between the 'Owner' group and the 'Everyone' group, and then by using an instance of the POU access settings to illustrate how the POU permissions are set.

(1) About 'Owner' and 'Everyone'

(1) Before setting up the access rights of the POU, the grouping settings are set in the menu bar → Engineering Settings→ Users and Groups. The system brings two groups and one user, the Everyone, Owner, and Owner users, as shown. The Owner group user is granted all authorizations, and the figure shows that the Owner user is a member of the group Owner.



Therefore, the password for the 'Owner' user should be set first (the initial password is empty), as shown in the diagram. If no password is set for the 'Owner' user, the 'Owner' user identity can access all POUs with an empty password.



②When a user opens a project file, it is opened by default as a user in the 'Everyone' group. When the access control attribute of POU_1 in the project is set to "Deny", POU_1 cannot be accessed when the project file is opened, and access rights can be obtained through the 'Owner' identity.

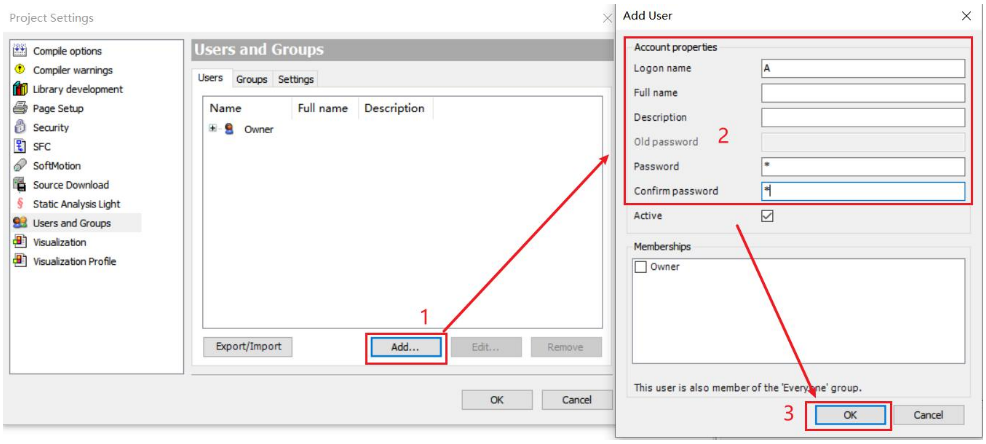
Example of POU permission settings

① Assume there are three users, A, B and C. A project has three POUs, POUA, POUB and POUC, and the relationship between users and POU access rights is shown in Table I.

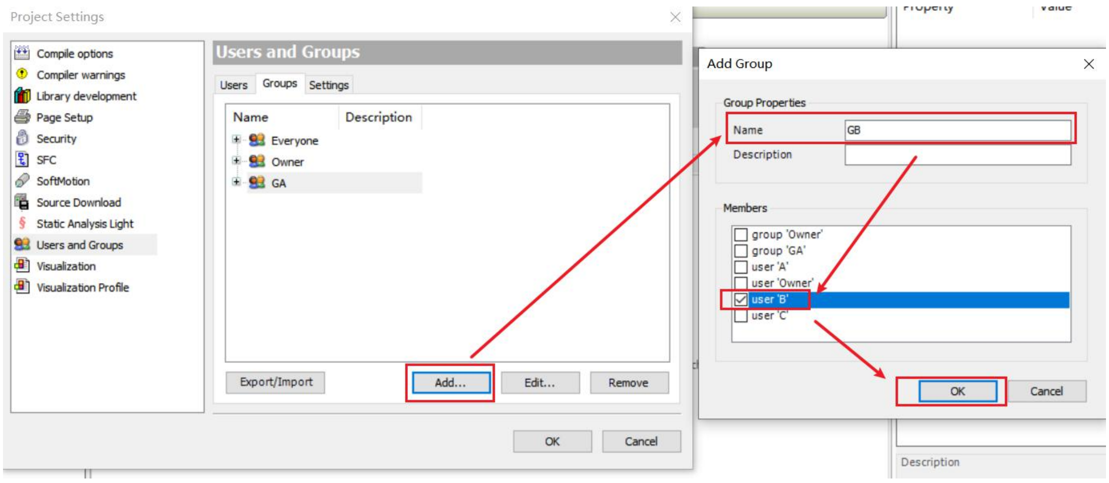
用户	POUA	POUB	POUC
A	√	×	×
B	×	√	×
C	×	×	√

Note: ″√″ means access is possible; ″×″ means no access

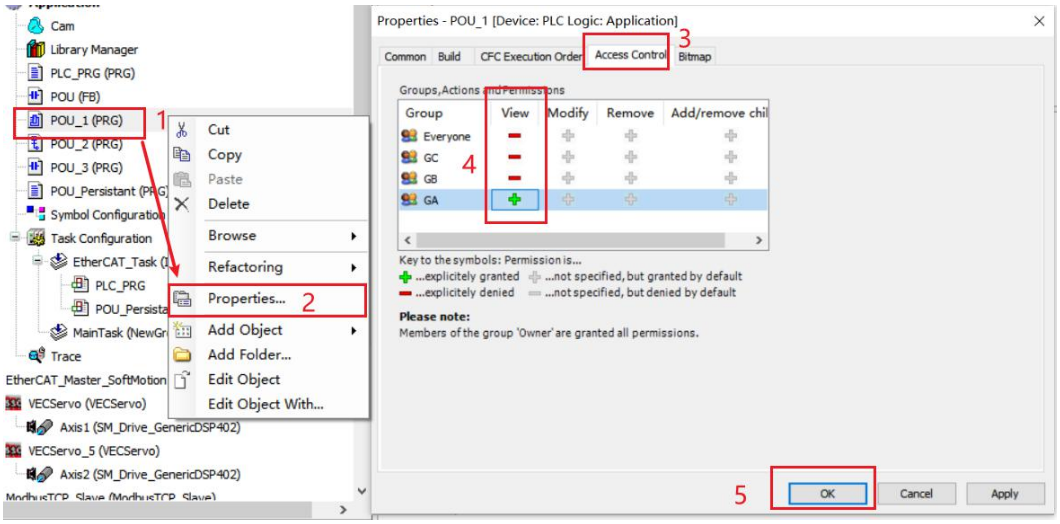
② Add three users A, B and C. When adding them, you need to enter the Owner username and password, as shown in the figure



③ Since the object of setting permissions is groups, you need to add three groups, GA, GB and GC, which contain users A, B and C respectively



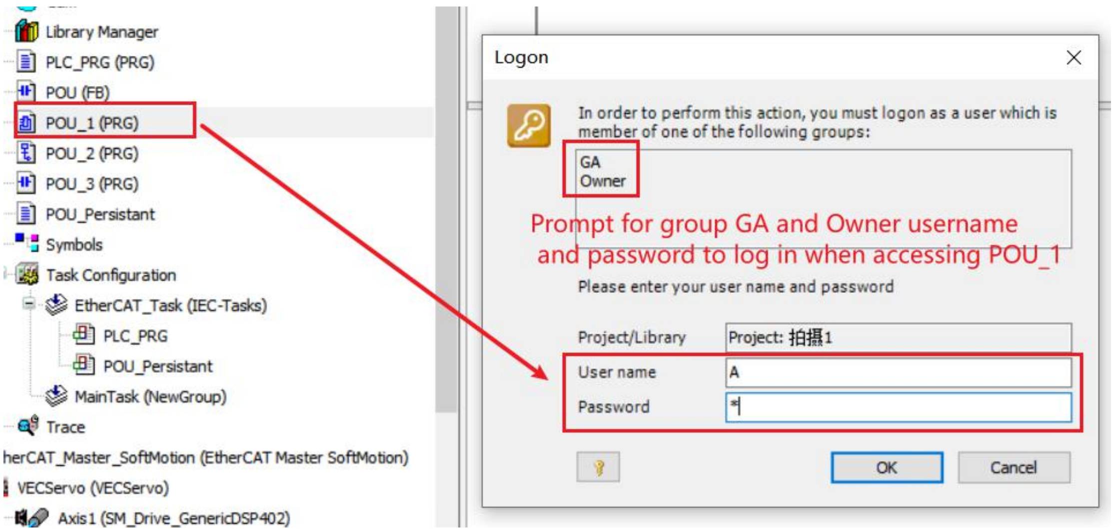
④ The settings for POUA, POUB and POUC respectively are shown in the figure. Since the project files are opened by default by users in the 'Everyone' group, the access properties of the 'Everyone' group in the POUA, POUB and POUC control properties are set to "Deny "



The results are as follows:

User P	POUA	POUB	POUC
Everyone	Refused	Refused	Refused
Ga	Allow	Refused	Refused
GB	Refused	Allow	Refused
GC	Refused	Refused	Allow

⑤ After saving the project file, re-open the project file, then you need to enter the user name and password to access the POUs, where the 'Owner' user has access to all POUs.



⑥ For other permissions set up in a similar way as for access permissions.

Appendix A VECServo supported origin regression models

Zero return mode setting process

Note: If it is an absolute encoder and the Z point is used as the encoder zero point, please first pre-set P03.79 - How many pulses the absolute encoder outputs per week.

(1) Set 6060 = 6 first
(2) Set the return to zero offset 607Ch, the unit of which is the user position unit.
(3) Set return to zero mode 6098h
(4) Set the speed of home switch finding 6099h_01, its unit is rpm
(5) Set the speed of finding the Z point 6099h_02, its unit is rpm
(6) Set the speed of return to zero plus or minus 609Ah, which is in user units/s/s
(7) Set control word 6040h to 6 -> 7 -> 15 -> 31 in order to perform zero return
(8) Read status word 6041h

Zero-back mode-related objects

Back to zero mode 6098h

Index	6098h
Name	Back to zero
The object type	Variable
The data type	There are 8 bits of symbol
PDO mapping	Mapable
Read and write properties	Readable and writeable
The default	0
Set the range	0-35
A detailed description	Set the back to zero mode

Back to zero speed 6099h

Index	6099h
Name	Back to zero speed
The object type	Array object
The data type	Unsigned 32 bits
PDO mapping	Mapable
Read and write properties	Readable and writeable

Index_Child Index	6099h_00
Name	The number of valid sub-indexes of 6099h
The data type	Unsigned 32 bits
PDO mapping	Cannot be mapped
Read and write properties	Read-only
The default	2

Index_Child Index	6099h_01
Name	Look for the speed rpm of the origin switch
The data type	Unsigned 32 bits
PDO mapping	Mapable
Read and write properties	Readable and writeable
The default	P03.53

Index_Child Index	6099h_02
Name	Look for Z-point speed rpm
The data type	Unsigned 32 bits
PDO mapping	Mapable
Read and write properties	Readable and writeable
The default	P03.54

Back to zero acceleration 609Ah

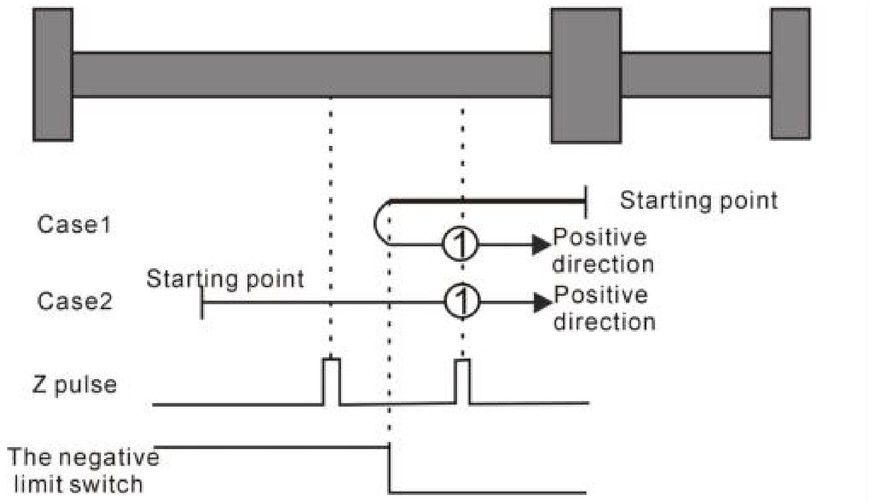
Index	609Ah
Name	Zero acceleration back
The object type	Variable
The data type	Unsigned 32 bits
PDO mapping	Mapable
Read and write properties	Readable and writeable
The default	500000
Set the range	0~4294967295
A detailed description	Zero acceleration, unit user unit/s/s

Back to zero mode

Back to zero is the calibration of a mechanical zero point, after marking, all absolute positions are used as a reference point to move. VEC bus-type servo has a variety of back-zero mode, according to the zero-back mode of 6098h settings, the corresponding back-zero action. Users can choose the appropriate origin back to zero mode according to site conditions and process requirements.

- **Origin back to zero mode 1: Depends on the origin regression of the reverse operating limit switch and Z pulse**

Scenario 1: When the user triggers the execution back to zero, if the reverse running limit switch state is low, then the axis starts to move reverse at the first speed, when the reverse run limit switch is at a high level, the direction of motion changes and the second speed starts to move;
Scenario 2: When the user triggers the execution back to zero, if the reverse running limit switch state is high, then the positive motion is started directly at the second speed, and the position of the first Z pulse encountered when the reverse running limit switch state is low is the origin.

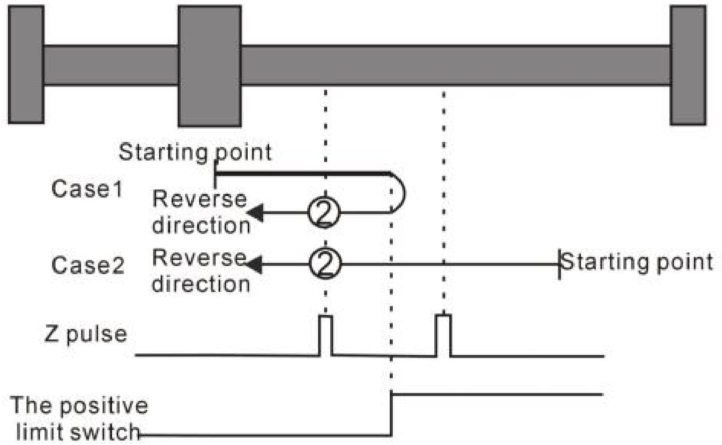


Homing method 1: Homing on the negative limit switch and Z index pulse

- **Origin back to zero mode 2: Dependson the origin regression of the** positive **running limit switch and Z pulse**

Scenario 1: When the user triggers execution back to zero, if the forward running limit switch state is low, then the axis starts to move forward at the first speed, when the forward running limit switch is at a high level, the direction of motion changes and the second speed starts to move, when the forward running limit switch state is low, the position of the first Z pulse is the original position.
Scenario 2: When the user triggers execution back to zero, if the positive running limit switch state is high, then the axis starts the reverse movement directly at the second speed, and the position of the first Z pulse encountered when the positive running limit switch state is low is

the origin position.



Homing method 2: Homing on the positive limit switch and Z index pulse

Mode 3 to Mode 6 depends on the origin switch and **the origin zero** of the **Z pulse**

- **Origin zero model 3**

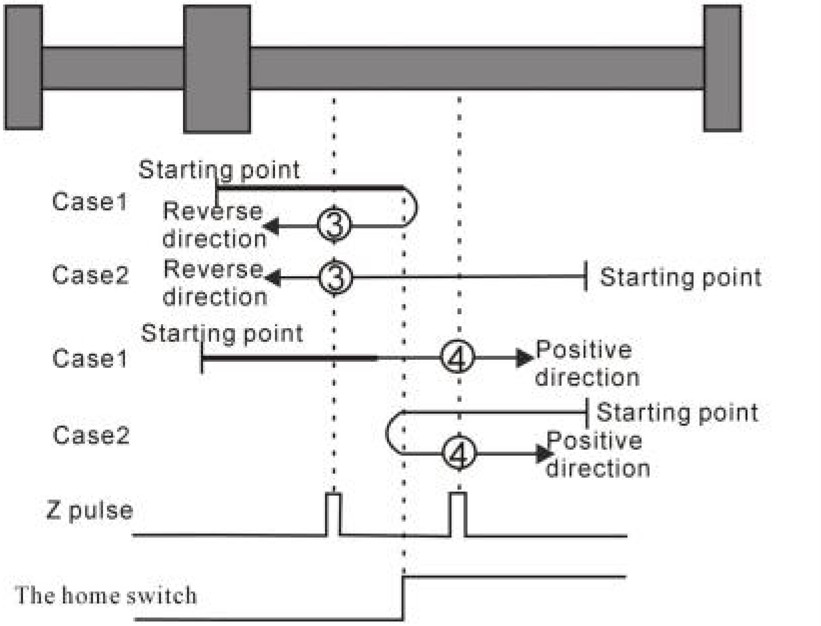
Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, the axis begins to move forward at the first speed, when the origin switch is encountered at a high level, the direction of motion changes and the second speed starts to move, the position of the first Z pulse encountered when the origin switch state is low is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the reverse movement directly at the second speed, and the position of the first Z pulse encountered when the origin switch state is low is the origin position.

- **Origin zero model 4**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move at the first speed, when the origin switch is at a high level, the second speed is moving positively, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the reverse movement directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second segment speed starts to move, the position of the first Z pulse is the origin position.



Homing method 3 ~ 4 Homing on the home switch and the Z index pulse

- **Origin zero model 5**

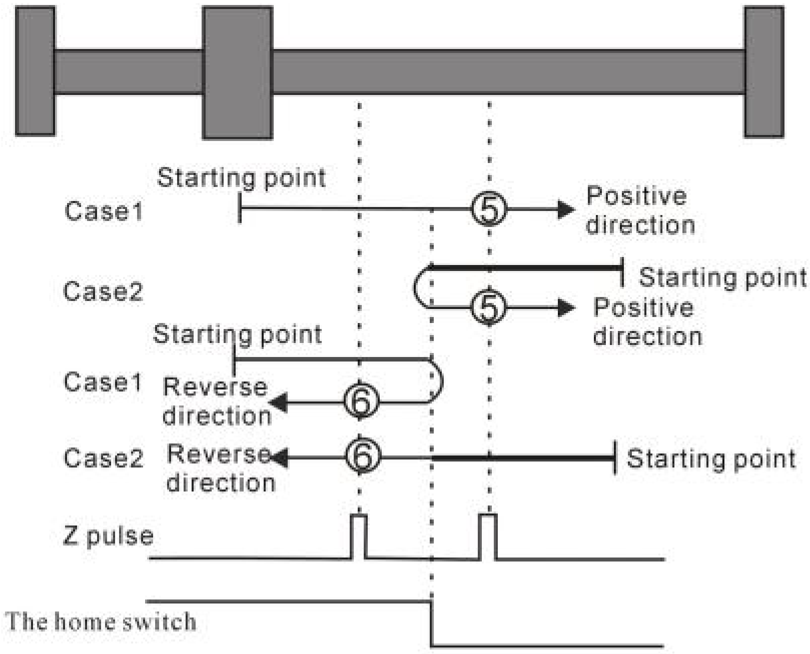
Scenario 1: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the positive motion directly at the second speed, and the position of the first Z pulse encountered when the origin switch state is low is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to reverse at the first speed, when the origin switch is at a high level, the direction of motion changes and the second speed starts to move, when the origin switch state is low, the position of the first Z pulse is the origin position.

- **Origin zero model 6**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is high, then the axis starts forward motion directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, the second speed starts to move, and the position where the first Z pulse is encountered is the origin position.



Homing method 5 ~ 6 Homing on the home switch and the Z index pulse

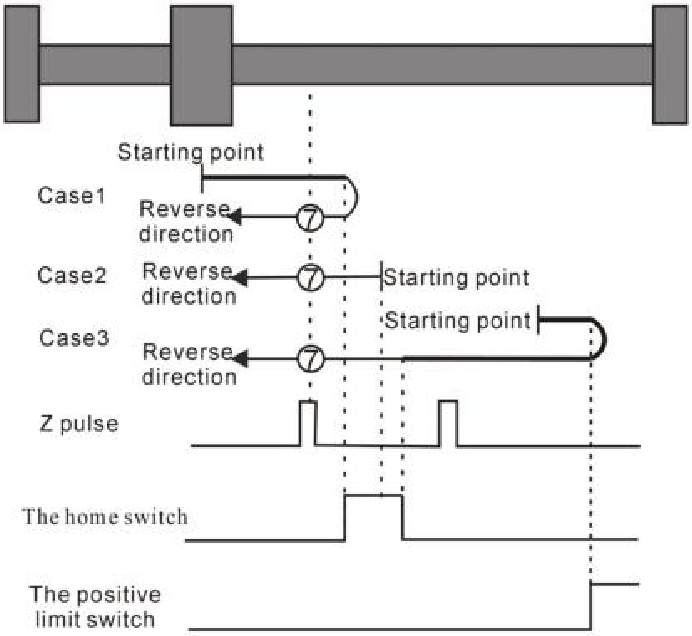
Mode 7 to Mode 10 depends on the origin **switch, the positive operating limit, and** the origin back zero of the **Z pulse**

- **Origin zero model 7**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is at a high level, the direction of motion changes and the second speed starts to move, when the origin switch state is low, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts to reverse at the second speed, and the position of the first Z pulse encountered when the origin switch state is low is the origin position.

Scenario 3: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is at a high level, the second speed starts to move, the position of the first Z pulse is encountered when the origin switch state is low.



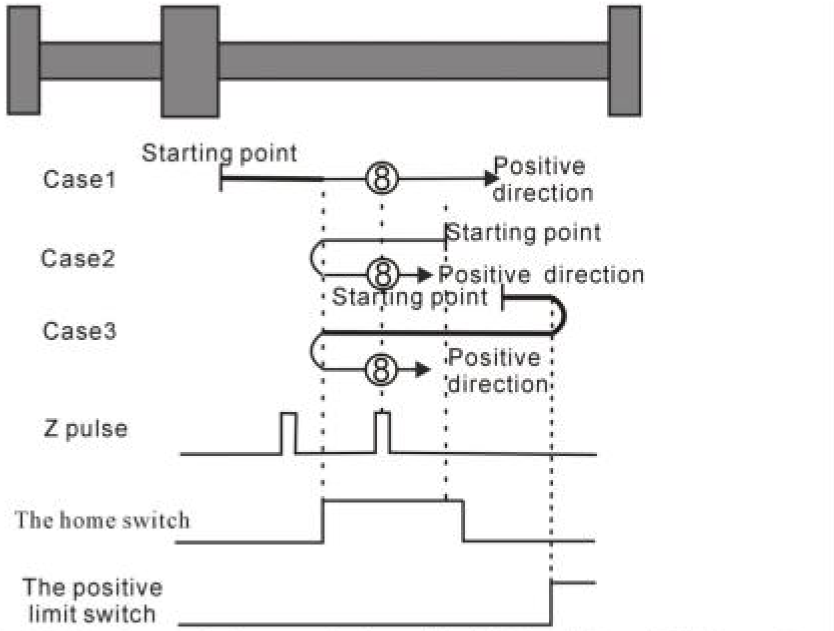
Homing method 7 Homing on the home switch, positive limit switch, and Z index pulse

- **Origin zero model 8**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move at the first speed, when the origin switch is at a high level, the second speed starts to move, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is high, then the axis directly at the second speed to start the reverse movement, when the origin switch is at a low level, the direction of motion changes and the second segment speed to start the movement, encountering the position of the first Z pulse is the origin position.

Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is high, the direction of motion changes and starts at the first speed Motion, when the origin switch is at a high level, is still moving at the first speed, when the origin switch state is low, the direction of motion changes and starts at the second speed, when the origin switch is at a high level, the second speed starts to move, The position where the first Z pulse is encountered is the origin position.



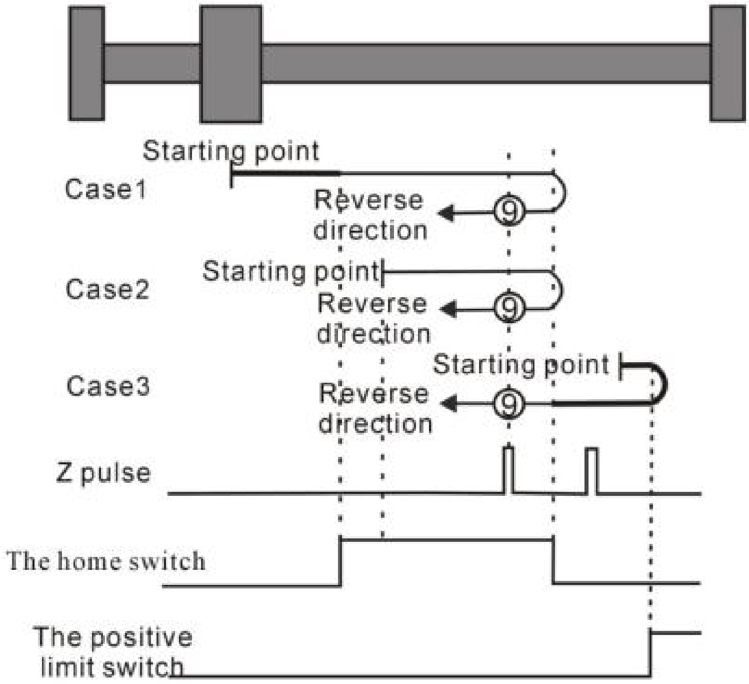
Homing method 8 Homing on the home switch, positive limit switch, and Z index pulse

- **Origin times zero model 9**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is at a high level, the second speed starts to move, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts to move forward at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, the position of the first Z pulse is the origin position.

Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is at a high level, the direction of motion changes and the first speed to start the movement, when the origin switch is at a high level, the second speed to start the movement, the position of the first Z pulse is the original position.



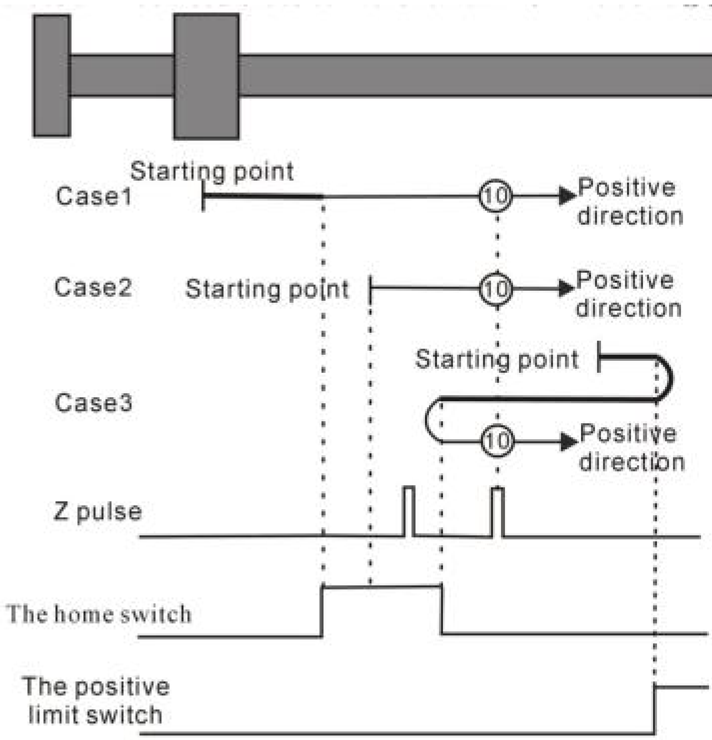
Homing method 9 Homing on the home switch, positive limit switch, and Z index pulse

- **Origin zero model 10**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move at the first speed, when the origin switch is at a high level, the second speed is started, when the origin switch is at a low level, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis begins to move at the second speed, and when the origin switch is encountered at a low level, the position of the first Z pulse is the origin position.

Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is at a high level, the direction of motion changes again and starts at the second speed, when the origin switch is low, the first Z pulse position is encountered.



Homing method 10 Homing on the home switch, positive limit switch, and Z index pulse

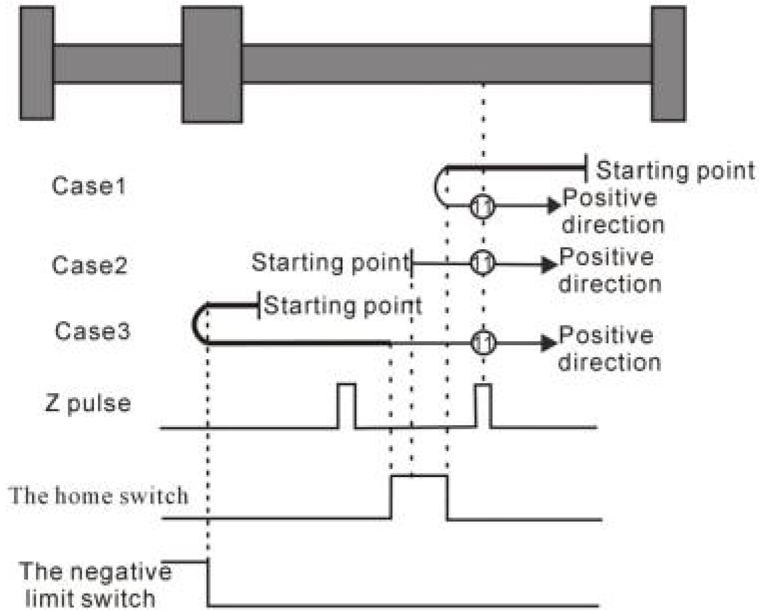
Mode 11 to Mode 14 depends on the origin **switch, the reverse operating limit, and the origin zero of the Z pulse**

- **Origin zero model 11**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, the direction of motion changes and the second speed starts to move, when the origin switch state is low, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the positive motion directly at the second speed, and the position of the first Z pulse encountered when the origin switch state is low is the origin position.

Scenario 3: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is at a high level, the second speed starts to move, the position of the first Z pulse is encountered when the origin switch state is low.



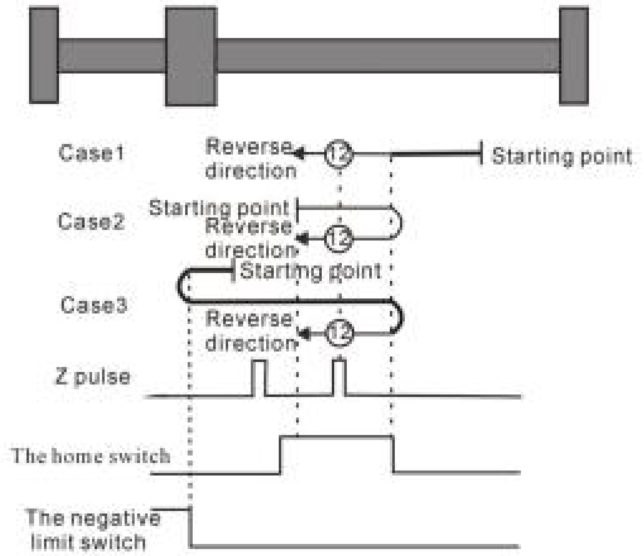
Homing method 11 Homing on the home switch, the negative limit switch and the Z index pulse

- **Origin zero model 12**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, the second speed is the starting point position, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts forward motion directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second segment speed starts to move, the position of the first Z pulse is the origin position.

Scenario 3: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse run limit switch is high, the direction of motion changes and starts at the first speed Motion, when the origin switch is at a high level, still at the first speed, in the origin switch state is low, the direction of motion changes and the first speed to start the movement, when the origin switch is at a high level, the second speed to start the movement, The position where the first Z pulse is encountered is the origin position.



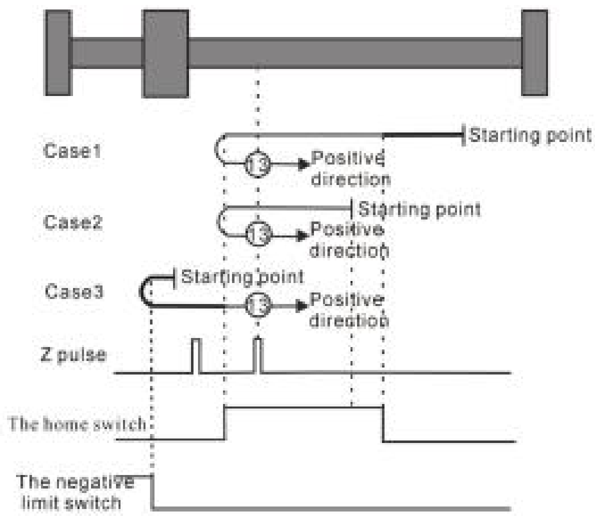
Homing method 12 Homing on the home switch, the negative limit switch and the Z index pulse

- **Origin zero model 13**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, the second speed starts to move, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, the position of the first Z pulse is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is high, then the axis is directly reverse motion at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, the position of the first Z pulse encountered is the origin position.

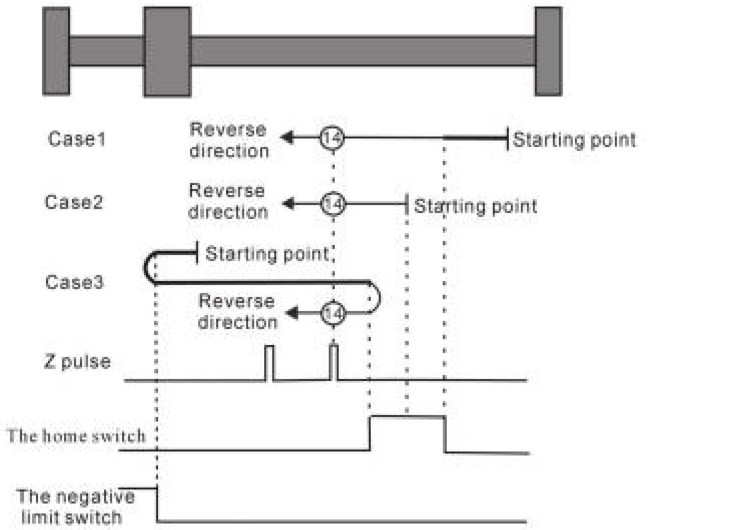
Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse running limit switch is at a high level, the direction of movement changes and the first speed starts to move, when the origin switch is at a high level, the second segment speed starts to move, the position of the first Z pulse is the original position.



Homing method 13 Homing on the home switch, the negative limit switch and the Z index pulse

- **Origin zero model 14**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, when the second speed is encountered, when the origin switch is at a low level, the position of the first Z pulse is the origin position.
Scenario 2: When the user triggers the execution back to zero, if the origin switch state is high, then the axis starts to move reverse at the second speed, and when the origin switch is encountered at a low level, the position of the first Z pulse is the origin position.
Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse running limit switch is at a high level, the direction of motion changes and starts at the first speed, when the origin switch is at a high level, the direction of motion changes again and starts at the second speed, when the original point switch is at a low position, the first Z pulse is encountered.



Homing method 14 Homing on the home switch, the negative limit switch and the Z index pulse

Mode 15 - Mode 16 is reserved

- **Patterns 15 and 16 are retained as origin regression patterns for later development.**

Mode 17 to Mode 30 requires the origin regression of the Z pulse

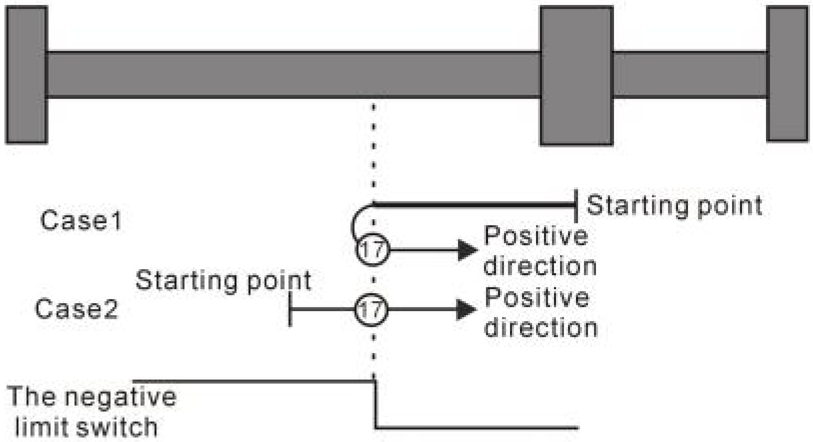
Mode 17to mode 30 is similar to mode 1 to mode 14 mentioned earlier, but the positioning of their origin regression position no longer requires Z pulses, but only according to the relevant origin switch and limit switch state changes to achieve. Mode 17 is similar to mode 1, mode 18 is similar to mode 2, mode 19 and mode 20 are similar to the previous mode 3, mode 21 and mode 22 are similar to the previous mode 5, mode 23 and mode 24 are similar to the previous mode 7, mode 25 and mode 26 are similar to the previous mode 9. Modes 27 and 28 are similar to the previous mode 11, and modes 29 and 30 are similar to the previous mode 13.

- **Origin back to zero mode 17: Depends on the origin back zero of the reverse**

operating limit switch

Scenario 1: When the user triggers execution back to zero, if the reverse running limit switch state is low, then the axis starts to move reverse at the first speed, when the reverse run limit switch is at a high level, the direction of motion changes and the second speed starts to move;

Scenario 2: When the user triggers the execution back to zero, if the reverse running limit switch state is high, then the axis starts the positive motion directly at the second speed, and the position at the reverse running limit switch state is the origin position when the state of the reverse running limit switch is low.

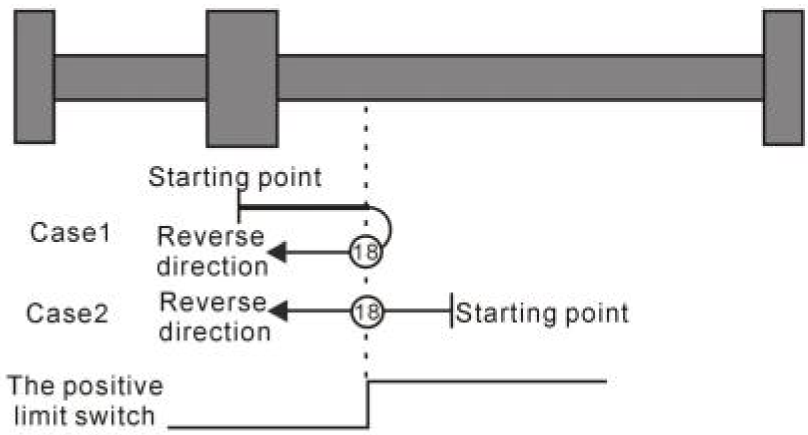


Homing method 17: Homing on the negative limit switch

- **Origin back to zero mode 18: Depends on the origin regression of the positive running limit switch**

Scenario 1: When the user triggers the execution back to zero, if the forward running limit switch state is low, then the axis starts to move forward at the first speed, when the forward running limit switch is at a high level, the direction of movement changes and the second speed starts to move, the position at the forward running limit switch state is low when the position is the original position.

Scenario 2: When the user triggers the execution back to zero, if the positive running limit switch state is high, then the axis starts the reverse movement directly at the second speed, and the position at the positive running limit switch state is the origin position when the state of the positive running limit switch is low.

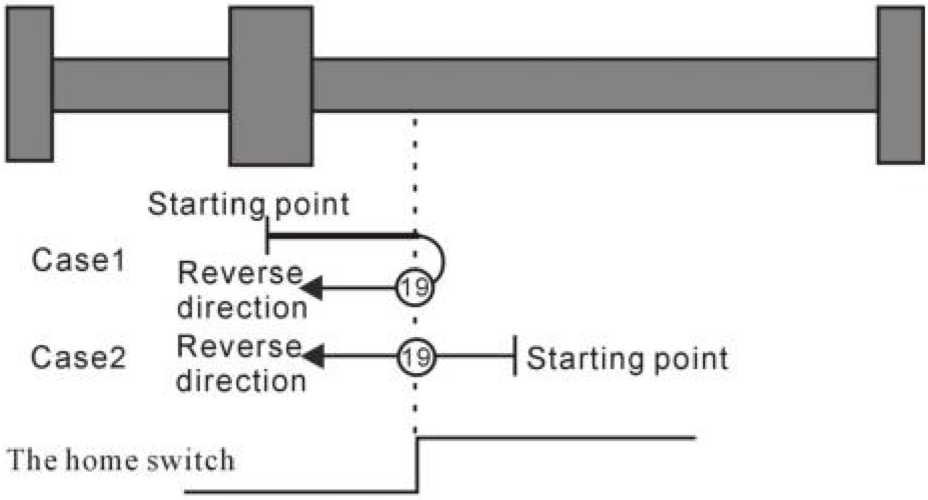


Homing method 18: Homing on the positive limit switch

- **Origin zero model 19**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is at a high level, the direction of motion changes and the second speed starts to move, when the origin switch is at a low position is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the reverse movement directly at the second speed, and when the origin switch is encountered at a low position, the position is the origin position.

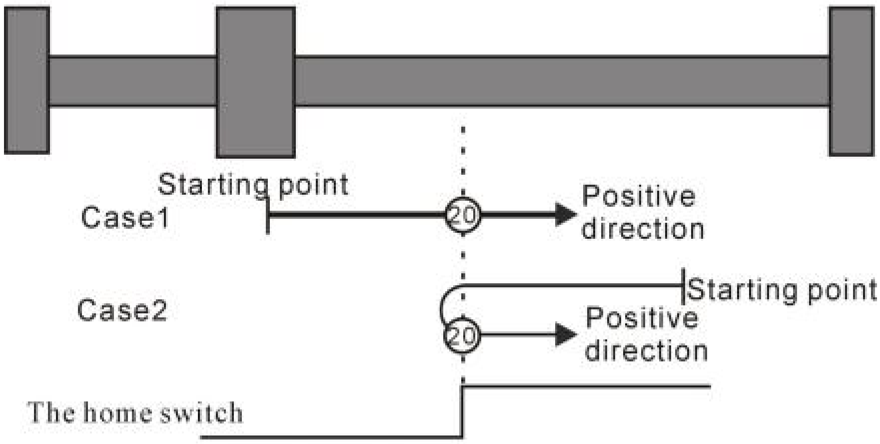


Homing method 19 Homing on the home switch

- **Origin zero model 20**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis begins to move positively at the first speed, and when the origin switch is encountered at a high position, the position is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the reverse movement directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is encountered at a high position is the origin position.

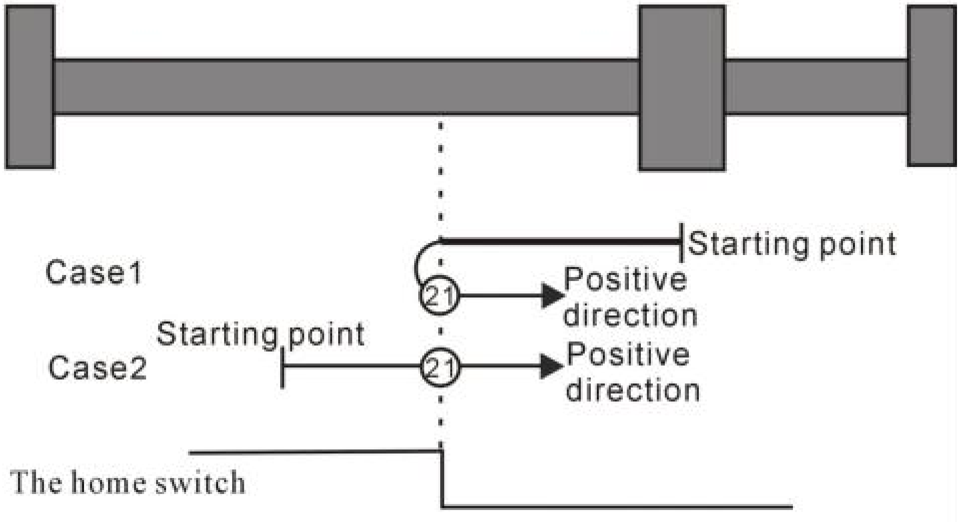


Homing method 20 Homing on the home switch

- **Origin zero model 21**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to reverse at the first speed, when the origin switch is encountered at a high level, the direction of motion changes and the second speed starts to move, when the origin switch is encountered at a low position is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the positive motion directly at the second speed, and when the origin switch is encountered at a low position, the position is the origin position.

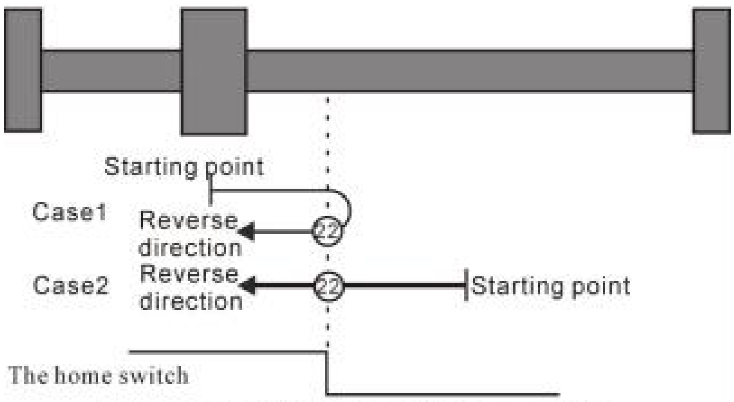


Homing method 21 Homing on the home switch

- **Origin zero model 22**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts forward motion directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is encountered at a high position is the origin position.

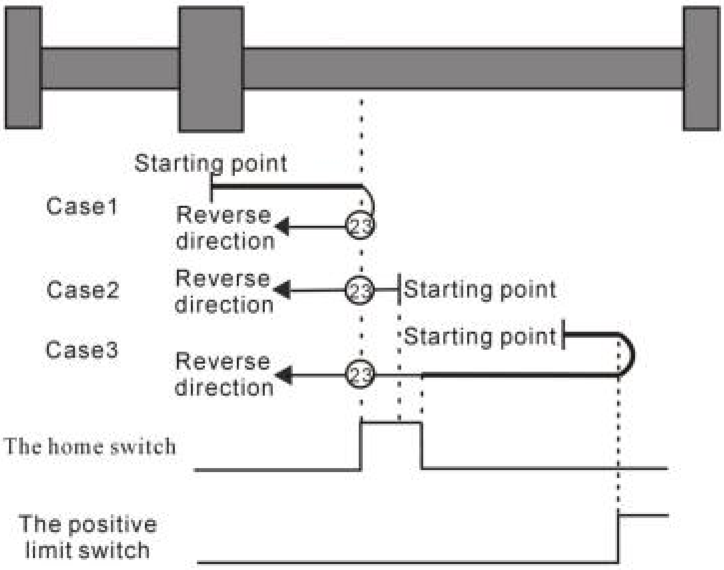
Scenario 2: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to reverse at the first speed, and when the origin switch is encountered at a high level, the position is the origin position.



Homing method 22 Homing on the home switch

- **Origin zero model 23**

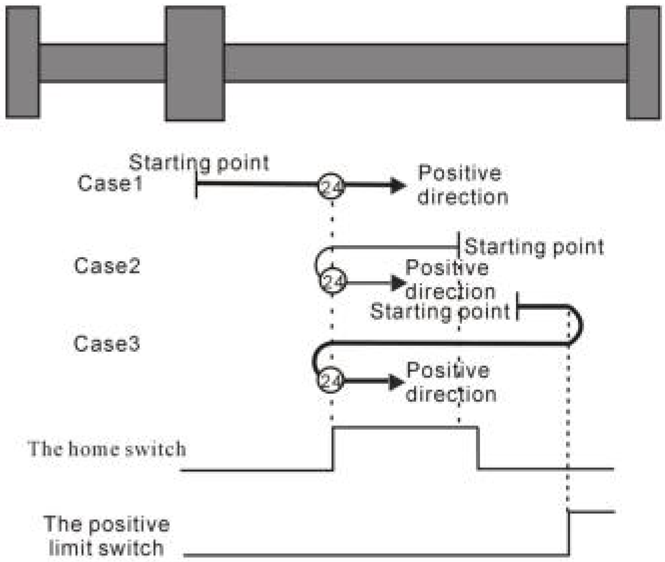
Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is at a high level, the direction of motion changes and the second speed starts to move, in the origin switch state is low when the position is the origin position.
Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the reverse movement directly at the second speed, and the position at the low level of the origin switch state is the origin position.
Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is at a high level, the second speed starts to move, the position at the low level of the origin switch state is the original position.



Homing method 23 Homing on the home switch, positive limit switch

- **Origin zero model 24**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis begins to move positively at the first speed, and when the origin switch is encountered at a high position, the position is the origin position.
Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts the reverse movement directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, the position when the origin switch is at a high level is the origin position.
Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is high, the direction of motion changes and at the first speed Start movement, when the origin switch is at a high level, still at the first speed, in the origin switch state is low, the direction of motion changes and the first speed to start the movement, when the origin switch is encountered at a high position is the origin position.



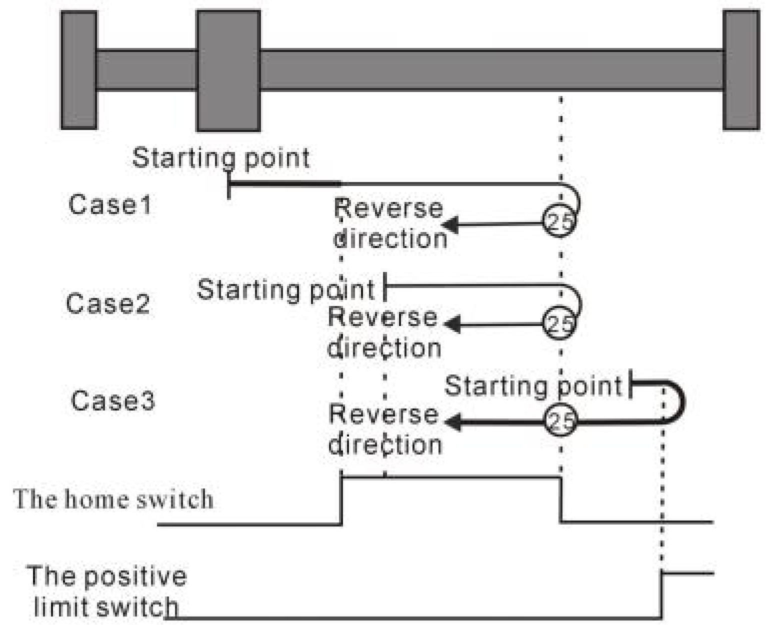
Homing method 24 Homing on the home switch, positive limit switch

- **Origin zero model 25**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is at a high level, the second speed is started, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is at a high position is the origin position.

Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts to move forward at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is at a high position is the origin position.

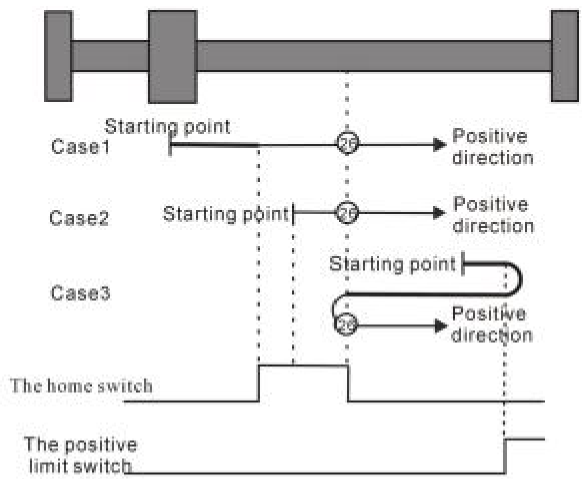
Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is encountered at a high position is the origin position.



Homing method 25 Homing on the home switch, positive limit switch

- **Origin zero model 26**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move at the first speed, when the origin switch is at a high level, the second speed is started, when the origin switch is at a low position is the origin position.
Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts to move at the second speed, and when the origin switch is encountered at a low position, the position is the origin position.
Scenario 3: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move forward at the first speed, when the origin switch is low and the forward running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is at a high level, the direction of motion changes again and starts at the second speed, when the origin switch is at a low position.



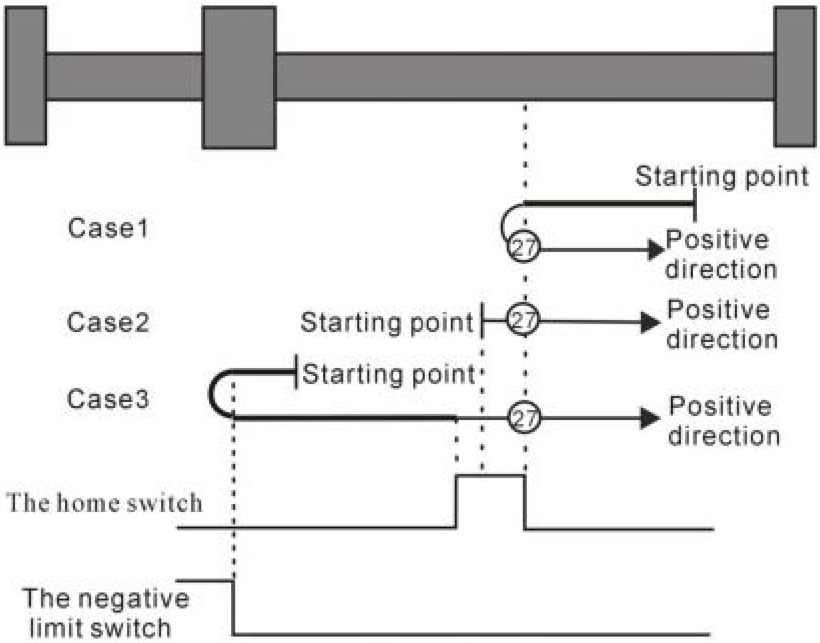
Homing method 26 Homing on the home switch, positive limit switch

- **Origin zero model 27**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is encountered at a high level, the direction of motion changes and the second speed starts to move, the position at which the origin switch state is low is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is high, then the axis starts the positive motion directly at the second speed, and the position at the low level of the origin switch state is the origin position.

Scenario 3: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is at a high level, the second speed starts to move, the position at the origin switch state is low.



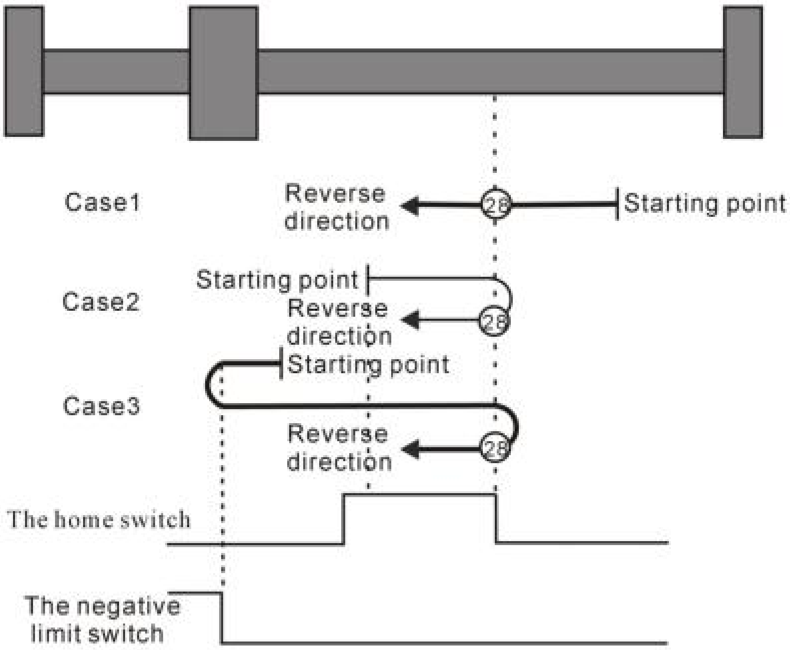
Homing method 27 Homing on the home switch, the negative limit switch

- **Origin zero model 28**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to reverse at the first speed, and when the origin switch is encountered at a high position, the position is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is high, then the axis starts forward motion directly at the second speed, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is encountered at a high position is the origin position.

Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse run limit switch is high, the direction of motion changes and at the first speed Start movement, when the origin switch is at a high level, still at the first speed, in the origin switch state is low, the direction of motion changes and the first speed to start the movement, when the origin switch is encountered at a high position is the origin position.



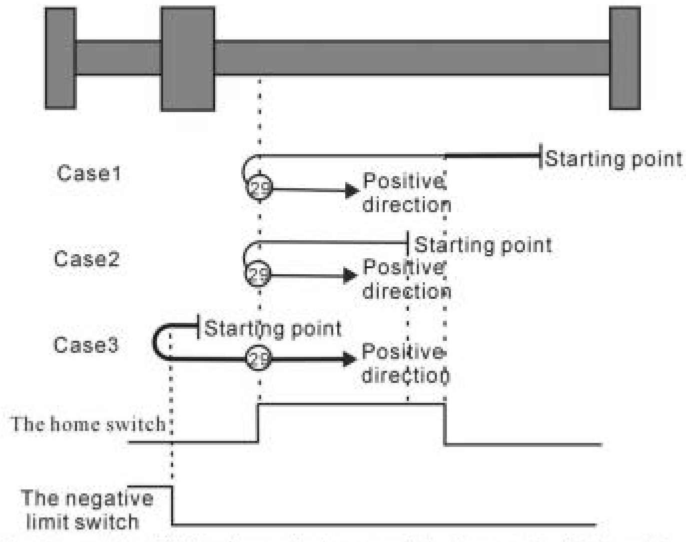
Homing method 28 Homing on the home switch, the negative limit switch

- **Origin zero model 29**

Scenario 1: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, the second speed is started, when the origin switch is at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is encountered at a high position is the origin position.

Scenario 2: When the user triggers the execution back to zero, if the origin switch state is high, then the axis is completely reverse motion at the second speed, when the origin switch is encountered at a low level, the direction of motion changes and the second speed starts to move, when the origin switch is encountered at a high position is the origin position.

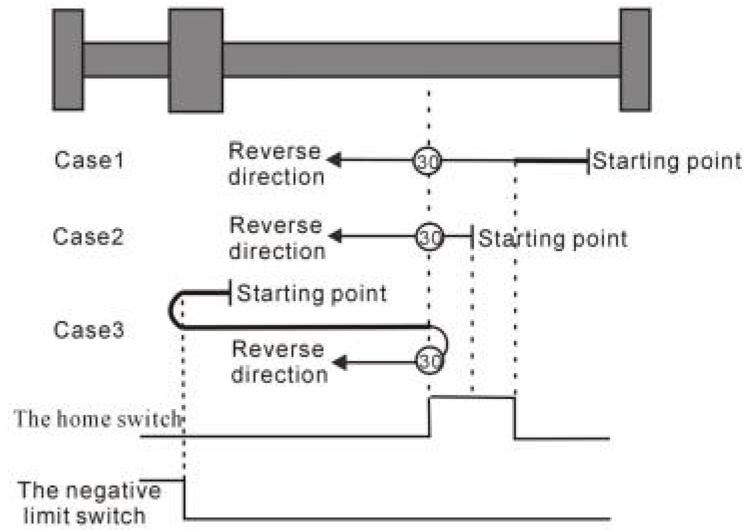
Scenario 3: When the user triggers execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse running limit switch is at a high level, the direction of motion changes and the first speed starts to move, when the origin switch is encountered at a high position is the origin position.



Homing method 29 Homing on the home switch, the negative limit switch

- **Origin zero model 30**

Scenario 1: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is at a high level, the second speed is started, when the origin switch is at a low position is the origin position.
Scenario 2: When the user triggers execution back to zero, if the origin switch state is high, then the axis starts to move reverse at the second speed, and when the origin switch is encountered at a low level, the position is the origin position.
Scenario 3: When the user triggers the execution back to zero, if the origin switch state is low, then the axis starts to move reverse at the first speed, when the origin switch is low and the reverse running limit switch is at a high level, the direction of motion changes and the movement begins at the first speed, when the origin switch is at a high level, the direction of motion changes again and the movement begins at the second speed, when the origin switch is at a low position.



Homing method 30 Homing on the home switch, the negative limit switch

Mode 31 and mode 32 are reserved

- **Patterns 31 and 32 are retained as the origin regression modes for later development.**

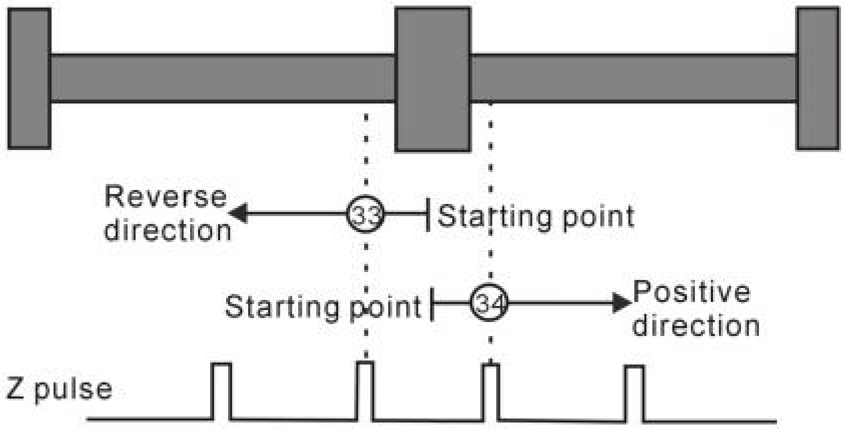
Mode 33 to Mode 34 depends on the origin regression of the Z pulse

- **Origin zero model 33**

In mode 33, when the user triggers execution back to zero, the axis begins to reverse at the second speed, and when the first Z pulse is encountered, the position is the origin position.

- **Origin zero model 34**

In mode 34, when the user triggers execution back to zero, the axis begins to move positively at the second speed, and when the first Z pulse is encountered, the position is the origin position.



Homing method 33 ~ 34 Homing on the Z index pulse

- **Origin back to zero mode 35: Origin regression depending on the current position**

In mode 35, when the user triggers execution back to zero, the axis does not move and the current position of the axis is considered the position of origin regression.

Appendix B Quick reference list of CiA402 common objects supported by VECServo

Index	Sub-indexes	Name	Access	Size	Unit	Setting range	Default value	PDO Mapping
603F	00	Error Code	RO	UINT16	-	TPDO		
This object gives the most recent fault code or warning code of the drive, corresponding to the lower 12 bits. To view the fault log, you can use 200B:22 and 23 to view up to 10 latest fault log codes.								
6040	00	Control words	RW	UINT16	-	0~65535	0	RPDO
Status guidance after servo power-up, command control in each servo mode								
6041	00	Status word	RO	UINT16	-	TPDO		
Reacts to the servo drive operating status.								
6050	00	Slow down time	RW	UDINT32	ms	0-U32MAX	5000	
Servo slow stop time setting								
6051	00	Fast downtime	RW	UDINT32	ms	0-U32MAX	50	
Servo quick stop time setting								
605A	00	Quick stop method selection	RW	INT8	0~7	2	-	
0~7: Selects the drive quick stop method								
605D	00	Suspension stop method selection	RW	INT8	1~3	1	-	
Select drive pause method								
605E	00	Fault response options	RW	INT16	-	0-4	0	
0-4 0 - Direct break enable 1-Fast stop break enable 2-Slow stop break enable 3-Fast stop hold enable 4-Slow stop hold enable								
6060	00	Servo mode selection	RW	INT8	-	0~10	0	RPDO
1- Contour position mode (pp) 3- Profile velocity mode (pv) 4- Contour torque mode (pt) 6- Zero return mode (hm) 8- Periodic synchronous position mode (csp) 9- Cyclic synchronous velocity mode (csv) 10- Periodic synchronous torque mode (cst)								
6061	00	Servo operation mode display	RO	INT8	-	TPDO		

Index	Sub-indexes	Name	Access	Size	Unit	Setting range	Default value	PDO Mapping
Actual mode of operation								
6062	00	Location commands	RO	INT32	Command unit	TPDO		
Position command value per position loop cycle time, command units								
6063	00	Location feedback	RO	INT32	Encoder units	TPDO		
The current position of the motor fed back by the motor encoder.								
6064	00	Location feedback	RO	INT32	Command unit	TPDO		
Position feedback value after inverse gear ratio calculation. 6063=6064× Gear ratio								
6065	00	Excessive position deviation threshold	RW	UINT32	Command unit	0~232-1	3145728	RPDO
The drive reports an excessive position deviation (Er.B00) fault when the position deviation 60F4 is greater than ±6065. This fault can be reset by bit 13=1 of 6041 in contour position mode at the same time.								
6067	00	Position reaches threshold	RW	UINT32	Command unit	0~65535	7	RPDO
When the position deviation 60F4 is less than this value and the time reaches 6068, the DO signal for positioning completion is valid and bit 10 of 6041 = 1. If either of these conditions is not met, the position arrival is invalid.								
6068	00	Location arrival window time	RW	UINT16	ms	0-65535	0	RPDO
When the position deviation 60F4 is less than this value and the time reaches 6068, the DO signal for positioning completion is valid and bit 10 of 6041 = 1. If either of these conditions is not met, the position arrival is invalid.								
606B	00	Speed command value	RO	DINT32	0.1RPM	TPDO	0	
Servo speed command value								
606C	00	Actual speed	RO	INT32	s	TPDO		
This object displays the position feedback per second (command unit)								
606D	00	Speed reaches threshold	RW	UINT32	rpm	0~65535	10	RPDO
When the difference between the motor speed feedback and the speed command is within ±606D and the time reaches 606E, the DO signal for speed arrival is valid and bit10=1 of 6041. if either of these conditions is not met, speed arrival is invalid.								
606E	00	Speed arrival window time	RW	UINT16	ms	0-65535	0	RPDO
When the difference between the speed feedback and the speed command is within ±606D and the time reaches 606E, the DO signal for the speed arrival is valid and bit10=1 of 6041. if either of these conditions is not met, the speed arrival is invalid.								
606F	00	Zero speed threshold	RW	UINT16	0.1rpm	0-65535	50	RPDO
Zero speed threshold for servo								
6071	00	Target torque	RW	INT16	0.1%	-5000~5000	0	RPDO
Target torque setting in torque mode								

Index	Sub-indexes	Name	Access	Size	Unit	Setting range	Default value	PDO Mapping
6072	00	Maximum torque command	RW	UINT16	0.1%	0~5000	0	RPDO
Maximum torque limit value.								
6074	00	Torque command	RO	INT16	0.1%	-5000~5000	0	TPDO
Torque output command after internal calculation of the drive								
6075	00	Motor rated current	RO	UDINT32	mA	-	-	TPDO
Motor rated current								
6077	00	Actual torque	RO	INT16	0.1%	-5000~5000	0	TPDO
Feedback torque value obtained by the drive								
6078	00	Percentage of actual drive torque	RO	INT16	0.1%	-	-	TPDO
Percentage of actual drive torque								
607A	00	Target location	RW	INT32	Command unit	-231-(231-1)	0	RPDO
The target position is given by the upper computer, and according to the position factor, i.e. the control word, the servo motor travels in response to the displacement increment.								
607C	00	Origin deviation	RW	INT32	Command unit	-231-(231-1)	0	RPDO
Position of mechanical origin offset mechanical zero point								
607D	Software absolute location restrictions							
	Sub-indexes	Number of sub-indexes	RO	UINT8	-	2	2	-
	01	Minimum Position Limit	RW	INT32	User Location Units	-231-(231-1)	-231	RPDO
	02	Maximum position limit	RW	INT32	User location units	-231-(231-1)	231-1	RPDO
Once the home return to zero is complete, the minimum and maximum position limit values allowed to operate are set by combining with the 607C. Position commands exceeding this value will stop when the limit is reached.								
607E	00	Command polarity	RW	UINT8	-	0-255	0	RPDO
BIT7- Position command polarity :0- Maintain original polarity, 1- Reverse polarity BIT6- Speed command polarity :0- Maintain original polarity, 1- Reverse polarity BIT5- Torque command polarity :0- maintain original polarity, 1- polarity reversed								
607F	00	Maximum speed	RW	UDINT32	Command unit /s	0~232-1	104857600	RPDO
Maximum speed limit value allowed. Setting method. 607F = maximum permissible motor speed (rpm) * encoder resolution /60								
6080	00	Maximum motor	RW	UDINT32	0.1RPM	-	-	RPDO

Index	Sub-indexes	Name	Access	Size	Unit	Setting range	Default value	PDO Mapping
		speed						
Maximum motor speed								
6081	00	Contour running speed	RW	UINT32	User Units	0~232-1	0	RPDO
In profile position mode, the motor is set to run at a constant speed within this displacement								
6083	00	Contour acceleration	RW	UINT32	Command unit /s2	1~232-1	1747626667	RPDO
Acceleration in pp, csv, pv modes. Default value 1747626667 Command unit /s2 means: acceleration from 0rpm to 1000rpm in 10ms.								
6084	00	Contour deceleration	RW	UINT32	Command unit /s2	1~232-1	1747626667	RPDO
Deceleration in pp, csv, pv modes. Default value 1747626667 Command unit /s2 means that it takes 10ms to accelerate from 0rpm to 1000rpm.								
6085	00	Rapid stop deceleration	RW	UINT32	User acceleration units	1~232-1	1747626667	RPDO
Acceleration of the deceleration section at 605A=2 when the upper unit issues a fast stop command (bit2=0 of 6040). Default value 1747626667 Command unit /s2 means: 10ms for acceleration from 0rpm to 1000rpm.								
6086	00	Operating curve selection	RW	INT16	-	0	0	RPDO
Set the motor running curve in profile position mode. Currently only linear movements are supported.								
6087	00	Torque ramp	RW	UINT32	0.1%/s	0	0xFFFFFFFF	RPDO
Set the torque command increments per second in profile torque mode								
6091	Gear ratio							
	00	Number of sub-indexes	RO	UINT8	2	2		
	01	Motor resolution	RW	UINT32	-	0~232-1	1	RPDO
	02	Load axis resolution	RW	UINT32	-	1-232-1	1	RPDO
Establishing a proportional relationship between encoder units and command units.								
6098	00	Origin reversion method	RW	INT8	-	0-35	0	RPDO
Supports 35 zero return methods as defined by the DS402 protocol								
6099	01	High-speed search for speed bumps	RW	UINT32	Command unit /s	0~232-1	1747626	RPDO
	02	Search home low speed	RW	UINT32	Command unit /s	0~232-1	174762	RPDO
609A	00	Return to zero	RW	UINT32	Command	1~232-1	1747	RPDO

Index	Sub-indexes	Name	Access	Size	Unit	Setting range	Default value	PDO Mapping
		acceleration			unit /s2			
The acceleration of the variable speed segment in home return to zero mode. Default value 1747 Command unit /s2 means: acceleration from 0rpm to 1000rpm in 10ms.								
60B0h	00	Position Offset	RW	INT32	Command units	-231-(231-1)	0	RPDO
60B1h	00	Speed bias	RW	INT32	Command unit /s	-231-(231-1)	0	RPDO
60B2h	00	Torque bias	RW	INT32	0.1%	-5000-5000	0	RPDO
60B8h	00	Probe mode	RW	UINT16	-	0-65535	0	RPDO
60B9h	00	Probe state	RW	UINT16	-	0-65535	0	RPDO
60BAh	00	Probe 1 rising edge position value	RW	INT32	Command unit	-231-(231-1)	0	RPDO
60BBh	00	Probe 1 falling edge position value	RW	INT32	Command unit	-231-(231-1)	0	RPDO
60BCh	00	Probe 2 rising edge position value	RW	INT32	Command unit	-231-(231-1)	0	RPDO
60BDh	00	Probe 2 falling edge position value	RW	INT32	Command unit	-231-(231-1)	0	RPDO
60E0h	00	Forward torque limiting	RW	UINT16	0.1%	0-5000	2000	RPDO
60E1h	00	Reverse torque limiting	RW	UINT16	0.1%	0-5000	2000	RPDO
60E3h	00	Supported zero return methods	RW	UINT16	-	-	-	-
60E6h	00	Position calculation method	RW	UINT16	-	0-1	0	-
60F4h	00	Position deviation	RO	INT32	Command unit	-231-(231-1)	0	TPDO
Position deviation, command units								
60FDh	00	DI Status	RO	UINT32	-	0~232-1	0	RPDO
60FEh	00	DO status	RO	UINT32	-	0~232-1	0	RPDO
60FFh	00	Target speed	RW	INT32	Command unit /s	-231-(231-1)	0	RPDO
Set speed command in synchronous cycle speed mode								
6502	00	Support for drive	RO	UINT32	0000	TPDO		

Index	Sub-indexes	Name	Access	Size	Unit	Setting range	Default value	PDO Mapping
		modes			03ADhex			
Displays the relevant modes supported by the drive.								

Appendix C Error Code Descriptions

SMC_ERROR：Records the error sequence number returned by the motion control function block.

Code	Source of generation	Variable name	Description of the cause of the error
0	All	SMC_NO_ERROR	No errors
1	Drive interfaces	SMC_DI_GENERAL_COMMUNICATION_ ERROR	Communication error (e.g. Sercos ring breakage)
2	Drive Interface	SMC_DI_AXIS_ERROR	Shaft error
10	Drive Interface	SMC_DI_SWLIMITS_EXCEEDED	The soft limit is activated. When bSWLimitEnable is enabled, the current position of the axis is not in the fSWLimitPositive and fSWLimitNegative range
11	Drive Interface	SMC_DI_HWLIMITS_EXCEEDED	Hardware limit switch is activated
13	Drive Interface	SMC_DI_HALT_OR_QUICKSTOP_NOT_ SUPPORTED	Drive status stopped or fast stop not supported
14	Drive interface	SMC_DI_VOLTAGE_DISABLED	Drive is not enabled
15	Drive interface	SMC_DI_IRREGULAR_ACTPOSITION	The drive is currently giving an incorrect position format. Check communication.
16	Drive interfaces	SMC_DI_POSITIONLAGERROR	Position hysteresis error. Limit values exceeded in set and current position
20	All modules created by motion control	SMC_REGULATOR_OR_START_NOT_SET	The controller is not enabled or the holding brake is not open
21	Axis in wrong control mode	SMC_WRONG_CONTROLLER_MODE	Axis is not in a correct control mode
30	Drive interface	SMC_FB_WASNT_CALLED_DURING_MOTION	The module created by the motion control is not called before the end of the motion
31	All modules	SMC_AXIS_IS_NO_AXIS_REF	The given AXIS_REF variable is not of type AXIS_REF
32	Axis in wrong control mode	SMC_AXIS_REF_CHANGED_DURING_ OPERATION	AXIS_REF- The return value of the variable is processed before the module is activated
33	Drive interface	SMC_FB_ACTIVE_AXIS_DIABLED	Axis not activated when moving (MC_Power.bRegulatorOn)
34	All modules created by motion control	SMC_AXIS_NOT_READY_FOR_MOTION	The axis cannot process the current command in the current state
40	Virtual drives	SMC_VD_MAX_VELOCITY_EXCEEDED	Maximum velocity reached (fMaxVelocity)
41	Virtual drives	SMC_VD_MAX_ACCELERATION_EXCEEDED	Maximum acceleration reached (fMaxAcceleration)
42	Virtual drives	SMC_VD_MAX_DECELERATION_EXCEEDED	Maximum deceleration reached (fMaxDeceleration)
50	SMC_Homing	SMC_3SH_INVALID_VELACC_VALUES	Invalid velocity or acceleration value
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLIMIT	Module requires end limit switch (for safety purposes)

Code	Source of generation	Variable name	Description of the cause of the error
70	SMC_ SetControllerMode	SMC_SCM_NOT_SUPPORTED	Mode not supported
71	SMC_ SetControllerMode	SMC_SCM_AXIS_IN_WRONG_STATE	The control mode used in the current mode is not supported
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	Axis is not a correct control mode and needs to be in torque mode
80	SMC_ResetAxisGroup	SMC_RAG_ERROR_DURING_STARTUP	Error at start of axeset
90	SMC_ ChangeGearingRatio	SMC_CGR_ZERO_VALUES	Incorrect variables
91	SMC_ ChangeGearingRatio	SMC_CGR_DRIVE_POWERED	Drive ratio cannot be changed in drive control mode
92	SMC_ ChangeGearingRatio	SMC_CGR_INVALID_POSPERIOD	Improper position cycle (<= 0)
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	Axis does not contain any information during the scan cycle (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	Axis does not have an error reset
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER	Axis does not perform an error reset
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	Error cannot be reset
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	Communication with the axis does not respond
130	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	Parameter serial number position
131	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_REQUESTING_ERROR	An error occurred during the transfer of parameters to the drive. See also Function block example Error in ReadDriveParameter (SM_ DriveBasic.lib)
140	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_PARAM_INVALID	Parameter serial number position or no write operation allowed
141	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_SENDING_ERROR	Refer to the module example WriteDriveParameter for errors (Drive_Basic.lib)
170	MC_Home	SMC_H_AXIS_WASNT_STANDSTILL	Axis not in standard condition
171	MC_Home	SMC_H_AXIS_DIDNT_START_HOMING	An error occurred while performing a zero return
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	Communication error
173	MC_Home	SMC_H_ERROR_WHEN_STOPPING	Execution of zero return error stopped. Check to see if deceleration is set.
180	MC_Stop	SMC_MS_UNKNOWN_STOPPING_ERROR	Unknown error at stop
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VALUES	Unsuitable speed or acceleration value
182	MC_Stop	SMC_MS_DIRECTION_NOT_APPLICABLE	Direction=shortest not available
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	The axis is in an error stop state. Stop cannot be processed.
184	MC_Stop	SMC_BLOCKING_MC_STOP_WASNT_CALLED	An instance of MC_Stop, locked axis

Code	Source of generation	Variable name	Description of the cause of the error
			(Execute=TRUE), cannot be called. Please call MC_ Stop(Execute=FALSE).
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_VALUES	Unsuitable speed or acceleration values
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	Wrong direction
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VALUES	Unsuitable velocity or acceleration value
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	Wrong direction
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VALUES	Unsuitable velocity or acceleration value
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	Wrong direction
276	MC_MoveSuperImposed	SMC_MSI_INVALID_VELACC_VALUES	Unsuitable velocity or acceleration value
277	MC_MoveSuperImposed	SMC_MSI_INVALID_DIRECTION	Wrong direction
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VALUES	Unsuitable velocity or acceleration value
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APPLICABLE	Direction=shortest/fastest not supported
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	Wrong alignment size
326	MC_PositionProfile	SMC_PP_STEP0MS	Step time = t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	Wrong alignment size
351	MC_VelocityProfile	SMC_VP_STEP0MS	Step time = t#0s
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	Wrong alignment size
376	MC_AccelerationProfile	SMC_AP_STEP0MS	Step time = t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	Trigger condition has been activated
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	Drive interface does not support window function
402	MC_TouchProbe	SMC_TP_COMM_ERROR	Communication error
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	Trigger condition has been terminated
426	SMC_ MoveContinuousRelative	SMC_MCR_INVALID_VELACC_VALUES	Unsuitable speed or acceleration values
427	SMC_ MoveContinuousRelative	SMC_MCR_INVALID_DIRECTION	Wrong direction
451	SMC_ MoveContinuousAbsolute	SMC_MCA_INVALID_VELACC_VALUES	Unsuitable velocity or acceleration value
452	SMC_ MoveContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	Wrong direction
453	SMC_ MoveContinuousAbsolute	SMC_MCA_DIRECTION_NOT_APPLICABLE	Direction= fastest not available
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	CAM does not contain any tappets
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	Tappet group ID reaches MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	More than 32 interfaces in one CAM_REF
625	MC_CamIN	SMC_CI_NO_CAM_SELECTED	No CAM selected
626	MC_CamIN	SMC_CI_MASTER_OUT_OF_SCALE	Spindle out of range
627	MC_CamIN	SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	For ramp_in function block speed and acceleration must be specified exactly
628	MC_CamIN	SMC_CI_SCALING_INCORRECT	Scale variable fEditor/TableMasterMin/Max incorrect

Code	Source of generation	Variable name	Description of the cause of the error
640	SMC_CAMBounds, SMC_CamBounds_Pos	SMC_CB_NOT_IMPLEMENTED	Function blocks in the given CAM format are not supported
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator = 0
676	MC_GearIn	SMC_GI_INVALID_ACC	Acceleration is not suitable
677	MC_GearIn	SMC_GI_INVALID_DEC	Acceleration inappropriate
725	MC_Phase	SMC_PH_INVALID_VELACCDEC	Velocity, acceleration, deceleration inappropriate
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD0	Rotation axis fPositionPeriod = 0
750	All modules using MC_ CAM_REF as input	SMC_NO_CAM_REF_TYPE	Given CAM not type MC_CAM_REF
751	MC_CamTableSelect	SMC_CAM_TABLE_DOES_NOT_COVER_ MASTER_SCALE	If the data obtained from the CamTable is not the spindle area (xStart and xEnd) obtained by data transformation.
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION_CHANGE	Spindle changes direction of rotation during slave coupling
800	SMC_ BacklashCompensation	SMC_BC_BL_TOO_BIG	The gear return ratio (fBacklash) is too large (>position periode/2)
1000	CNC 需要授权的功能块	SMC_NO_LICENSE	The target is not authorised for CNC.
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	Path cannot be processed because velocity = 0.
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	Previous path object Vel_End > 0.
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	Warning: GEOINFO- list is processed in DataIn, but the list is not set at the end. Reason: Forgot to set EndOfList in DataIn or SMC_Interpolator is faster than path compiler module
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_STOP	Stop speed > 0.
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	Use too many SMC_Interpolator calls for SoftMotion- errors.
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIES	Input-OutQueue DataIn is not used as the final processing module for SMC_ Final processing module for CHeckVelocities
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	Internal / numerical error error
1008	SMC_Interpolator	SMC_INT_VEL_ACC_DEC_ZERO	Velocity, acceleration or deceleration is null or too low.
1009	SMC_Interpolator	SMC_INT_DWIPOTIME_ZERO	FB call dwIpoTime = 0
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_SMALL	Data buffer too small
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	Path is not fully contained in the queue
1100	SMC_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	Velocity, deceleration or acceleration value is not positive
1120	SMC_Controlaxisbypos	SMC_CA_INVALID_ACCDEC_VALUES	The amount of change in fGapVelocity/fGapAcceleration/fGap. fGapDeceleration is not a positive value
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	Acceleration values are not allowed

Code	Source of generation	Variable name	Description of the cause of the error
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	Deceleration values are not allowed
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	Data below Queue is read and is empty.
1203	SMC_NCDecoder	SMC_DEC_JUMP_TO_UNKNOWN_LINE	Jumped line number cannot be executed because the line number is unknown
1204	SMC_NCDecoder	SMC_DEC_INVALID_SYNTAX	Syntax error
1205	SMC_NCDecoder	SMC_DEC_3DMODE_OBJECT_NOT_SUPPORTED	These objects do not support 3D mode
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	Buffer too small
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	Buffer element type is wrong
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	The current interpolation line cannot be found
1500	All function blocks using SMC_CNC_REF	SMC_NO_CNC_REF_TYPE	The given CNC program is not of type SMC_CNC_REF
1501	All function blocks that use SMC_ OUTQUEUE function blocks	SMC_NO_OUTQUEUE_TYPE	The given OutQueue is not of type SMC_OUTQUEUE
1600	CNC function blocks	SMC_3D_MODE_NOT_SUPPORTED	This function block is only available in the 2D path
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	File does not exist
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	No buffer allocation
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	Buffer is too small
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	Low buffered data in the swap area is read and is empty
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE_REPLACED	Placeholder variables cannot be replaced
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	input pvl cannot point to SMC_VARLIST object
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	File cannot be opened
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	No buffer definition
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	Buffer too small
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	Unknown file endings
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	File cannot be opened
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OVERRUN	Out of range buffering; WriteToFile must be called more often
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	File cannot be opened
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	Too much data saved to CAM
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	Wrong compile mode
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	Wrong file version
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	Unknown document endings
3001	SMC_ WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	SMC_WDPF_TIMEOUT_PREPARING_LIST
3002	SMC_ WriteDriveParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	File cannot be created
3003	SMC_ WriteDriveParamsToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	Error when reading file parameters

Code	Source of generation	Variable name	Description of the cause of the error
3004	SMC_ WriteDriveParamsToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	Wrong time when preparing parameter list
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	The conversion factor of the decoder reference (dwRatioTechUnitsDenom) is 0.
5001	SMC_Encoder	SMC_ENC_AXISUSEDBYOTHERFB	Other modules are processing the decoding axis.
5002	Driver interface	SMC_ENC_FILTER_DEPTH_INVALID	Inappropriate filter selection