# 深圳市威科达科技有限公司

**SHENZHEN VECTOR SCIENCE AND TECHNOLOGYCO., LTD.**

# VA Motion Controller

# Programming Manual

**Version2.0**

Tel / Fax: 0769-22235716

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

# Copyright Statement

# Foreword

Thank you for purchasing VA motion controller! VA motion-controlled PLC is the company developeDA high-performance general-purpose controller with an integrated function motion controller and PLC controller. This programming manual movement control described VA type PLC and motion control software use. Please read carefully before using the user understand the VA motion control PLC type of use.

**safety warning**

Note of warning, in order to avoid injury to the operator and other personnel, to prevent damage to the machine.

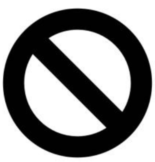
■ The following "Danger" and "Warning" symbol is in accordance with the degree of risk that the accident marked.

Danger

It Indicates a potentially hazardous situation that, if not avoided, will result in death or serious injury.

warning

It indicates a potentially hazardous situation that, if not avoided, will result in minor or moderate injury, or material damage.



This symbol indicates a prohibiteDAction.

This symbol indicates that the operatioNShould be noted.

**General Safety Summary**

**Please review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it. To avoid potential danger, follow the instructions to use the product in detail. Use a power cord that meets national standards. Properly connecteDAnd disconnected. The electrical wiring is correct before the control requirements, the power up sequence requires the user to control the opening movement, the servo drive is turned on, power-off sequence for the first servo drive off, the motion controller off (refer to VADetails motion controller hardware specification Chapter IX of this manual or quick start wiring instructions)**

**When a suspected fault do not operate and if you suspect the product is damaged. Please have a qualified service person check it.**

- ☞ **Do not operate in a wet / moist environment.**
- ☞ **Do not operate in an explosive atmosphere.**
- ☞ **Keep the products surface clean and dry.**

**Prevent electrostatic damage. Electrostatic discharge (ESD) may cause the motion controller and its attachment elements in damage. To prevent ESD, Be careful of control member, do not touch the controller components. Do not place the controller on the surface of static electricity may be generated. Transport and storage controllers in protective static bags or containers.**

**Product's range of applications**

**VA motion controller has a wide range of applications in traditional mechanical numerical control industry, but also plays an irreplaceable role in the emerging manufacturing electronics manufacturing and information products.**

# table of Contents

# Ⅰ MULTIPROG Overview

MULTIPROG is universal PLC programming system for control applications in large-scale development, it is widely used in machinery manufacturing, automotive and process automation industry. The tool is based on Microsoft's COM / DCOM technology architecture, Suitable for XP, Vista, win 7, win8 and win10 Windows operating systems. Its engineering structures fully compliant with IEC61131-3 standard, supports standard which defines five programming languages anDAllows users to customize database and data structures, and supports third-party development tools. System (ProConOS eCLR) running the programming tool and KW-Software can be launched either supporting the use, can also be applied to existing control systems, and can be unified configuration, programmeDAnd downloaded programs to multiple distributed controllers PLC .

MULTIPROG provides a wealth of operational commanDAnDAn excellent man-machine interface, drag and drop support, full keyboard operation. It provides online monitoring variables, mandatory and coverage feature that allows the program to set breakpoints and single-step debugging. And it comes with a logic analyzer, the recording can be easily input and output waveforms. For special occasions, provided the source code protection and non-stop online download function. For programmers habits of different countries, provids multi-language support, including variable names.

# II Features of the Software and Hardware Requirements

The manual describes how to use the programming software--MULTIPROG Express5.51. Users can write your own programs on the Vector motion controller according to this manual.

## 2.1 Basic Knowledge Required

Users who are familiar with this manual need to have general knowledge of automation technology, understand the Windows operating system, and have read the “Vector VA Motion Controller Programming Manual”.

## 2.2 MULTIPROG Express5.51 Features

- Support IEC61131-3 programming languages --FBD, LD, IL, ST and SFC.
- Clear project structure, intuitive programming language.
- Support cross-compilation between FBD, LDAnd IL. Support mixed programming.
- Multi-user programming, shorten the project programming time.
- Guide, cross references and other resources can be efficiently programmed.
- Compatible version can be centrally managed.

## 2.3 Computer Hardware Requirements

| device | lowest | recommend |
|---|---|---|
| CPU | 500MHz | 1GHz |
| RAM | 256MB | 1GB |
| hard disk | 500MB | 1GB |
| Monitor | 1024 × 768 | |
| operating | WindowsXP Pro, 2000, Vista, Win7, Win8, Win10, IE5.0 above | |
| communicati | TCP / IP, RS232 | |

## 2.4 MULTIPROG Express5.51 Support

| content | Quantity |
|---|---|
| Project node in the tree | 8000 |
| Configuration / Resource engineering tree | 1/1 |
| Each instance of the program resources | 1000 |
| Each resource tasks | 1 |

| | |
|---|---|
| Each task program instances | 500 |
| Each global variables POU / local variables | 1 5000/1 5000 |
| The library contains | 32 |
| The number of POU a project (including multiple | 2000 |
| A project supported by the I / O count | 256 Byte |
| I / O group | 200 |

# III PLC Working Principle

## 3.1 PLC Executing the Program Written by the User

Users download the program to the PLC and run PLC. The motion controller can cycle through the user's program. The program is not executed when the CPU is stopped. Each time the PLC executes the user's program, it is calleDA scan cycle, and the following work will be performed in one scan cycle:

A] Read Input: digital, analog input signals read into the input mapping area.

B] Execution program: executing a user's instruction and the intermediate, the final data stored in the various memory areas.

C] Processing any communications requests: to monitor communications.

D] CPU self-diagnosis: Check the firmware, the program memory working condition.

E] Write output: the stored data is copied to the physical output point in the output of the mapping area.

Note: If you want to implement a periodic task when you execute a program at regular intervals, instead of using a timer to trigger a periodic pulse in the cyclic scan task, the former should be accurate.

## 3.2 PLC Data Access

1) The users' data can be stored in different memory cells of the PLC, each cell has a unique address. The following table lists the different sizes of data that can be represented range of values;

| Numerical | Boolean (X) | Byte (B) | Word (W) | Double word (DW) |
|---|---|---|---|---|
| Unsigned integer | 0-1 | 0 to 255 | 0-65535 | 0-4294967295 |
| Signed integer | - | -128 to 127 | -32768 to +32767 | -2147483648 to +2147483647 |
| 32-bit floating point | - | - | - | + 1.175495E-38 to 34,022,823<br>-1.175495E-38 to 34,022,823 |

2) For data access, you must specify the address, the address starts with %, followed by the location prefix, the size prefix, and the byte address with an integer. The decimal point "." is added to the integer to indicate the bit. For example, %IX0.0 means the input mapping area is 0. The 0th bit of the byte, the following table is the address characteristics of the data.

| No. | Prefix | | definition | Conventions Data Types |
|---|---|---|---|---|
| 1 | Location prefix | I | Input mapping area | |
| 2 | | Q | Output mapping area | |
| 3 | | M | Intermediate variables mapping area | |
| 4 | The size prefixes | X | Place | BOOL |
| 5 | | B | Byte (8 bits) | BYTE |
| 6 | | W | Word (16 bits) | WORD |
| 7 | | D | Double word (32) | DWORD |
| 8 | | L | Long (64-bit) | LREAL |

Examples of variable address

DI / DO bit input and output opera tions:

%IX0.0 represents the 0th bit in the 0th byte of the digital input mapping area, indicating the definition of the input terminal DI0 in the motion controller;

%IX0.7 represents the 0th bit in the 1st byte in the digital input mapping area, which means that the input terminal DI7 in the motion controller is defined;

%QX0.0 represents the 0th bit in the 0th byte in the digital output mapping area, indicating the definition of the input terminal DO0 in the motion controller;

%QX1.0 represents the 0th bit in the 1st byte in the digital output mapping area, indicating the definition of the input terminal DO10 in the motion controller;

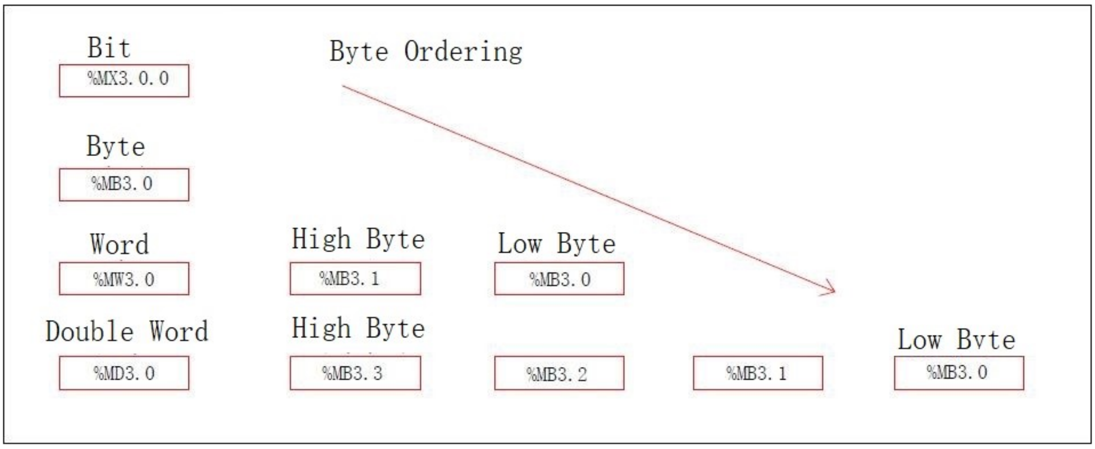
DI/DO input and output byte operations:

%IB0 represents the 8-bit status in the 0th byte of the digital input mapping area, indicating the input terminals DI0~DI7 in the motion controller;

3) Relationship between the motion controller address

In order to facilitate data exchange with peripherals (such as: human-machine, drive, other controllers), the motion controller opens the 10,000-byte address of %MB3.0000~%MB3.9999 to communicate with peripherals, of which %MB3.2000 ~%MB3.3000 is the data address saved by power-down. Users need to pay attention to the fact that the data exchanged with the peripherals does not need to fill in the %MB3.XXXX address, just fill in the variable name and select the correct data type to automatically assign the address.

The address relationship between bytes, words, and double words is ADouble word containing two words, or four bytes, below %MX3.0.0, %MB3.0, %MW3.0, and %MD3.0 For example, the address relationship and data arrangement between bytes, words, and double words are shown in the following figure:




For example, if a hexadecimal number 16#1234 is stored in %MW3.0, then 16#34 exists in %MB3.0, and 16#12 is stored in %MB3.1. If the bit operation in the program affects the byte, worDAnd double word of the bit, the reverse is also true.

Example of a variable address

%MX3.0.0 represents the 0th bit of the 0th byte in the intermediate variable area;

%MB3.0 represents the 0th byte in the intermediate variable area;

%MW3.0 represents the 0th word in the intermediate variable area;

%MD3.4 represents 1 double word starting from the 4th byte in the intermediate variable area

Note: (The 5.3 version of 1 byte can only define 1 bit, such as %MX3.0.0, (indicating that one byte has been occupied), the user can no longer use %MX3.0.1~%MX3.0.7, if you need to define One bit, the user needs to start with %MX3.1.0.)

The PLC supports the input of constant values. The constants can be binary numbers, decimal numbers, hexadecimal numbers, strings, ASCII codes or real numbers. The input format is shown in the table below.

| Number System | Format | For example |
|---|---|---|
| Binary | 2#___ | 2 # 001 |
| Decimal | ______ | 67 |
| Hex | 16 #__ | 16 # AC43 |
| String | '___' | 'VECTOR' |
| ASCII code | ASCII value | 16 # 30 |
| Real | REAL #___ | REAL # 3.1415926 |

## 3.3 PLC Save Data

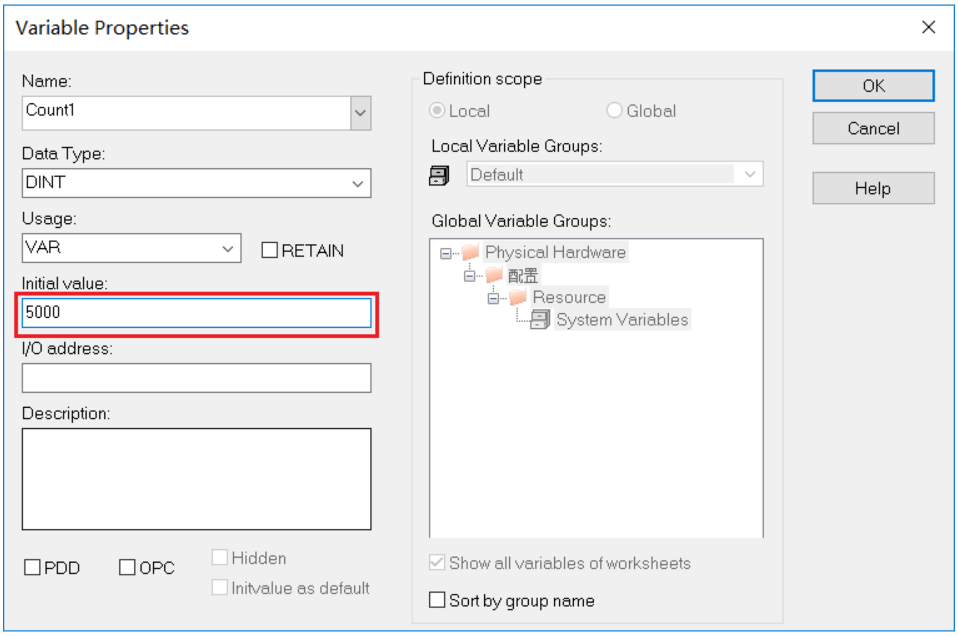
1) First, let's take a look at the approximate workings of the motion controller's internal memory. The internal memory of the motion controller is divided into two types: RAM random access memory and ferroelectric permanent memory. We all know that the data of the RAM memory must be maintained by the power supply. When the power supply of the memory chip is interrupted, the data stored therein does not exist anymore. The RAM memory is mainly useDAs a real-time access space for the program and program data of the motion controller program. The ferroelectric memory is a charged rewritable memory, and its data can be stored

for a long time under complete power-off. The motion controller loads the downloaded program, the data to be saved (optional), and the resource configuration (optional) into the RAM storage area each time the program is downloaded, and the CPU automatically copies it to the ferroelectric memory. In order to achieve permanent preservation. During the use of the motion controller, the PLC will restore the program and resource configuration from the ferroelectric memory area to the RAM memory area each time the power is turned on.

📖 The way of Motion controller saving program data while power off;

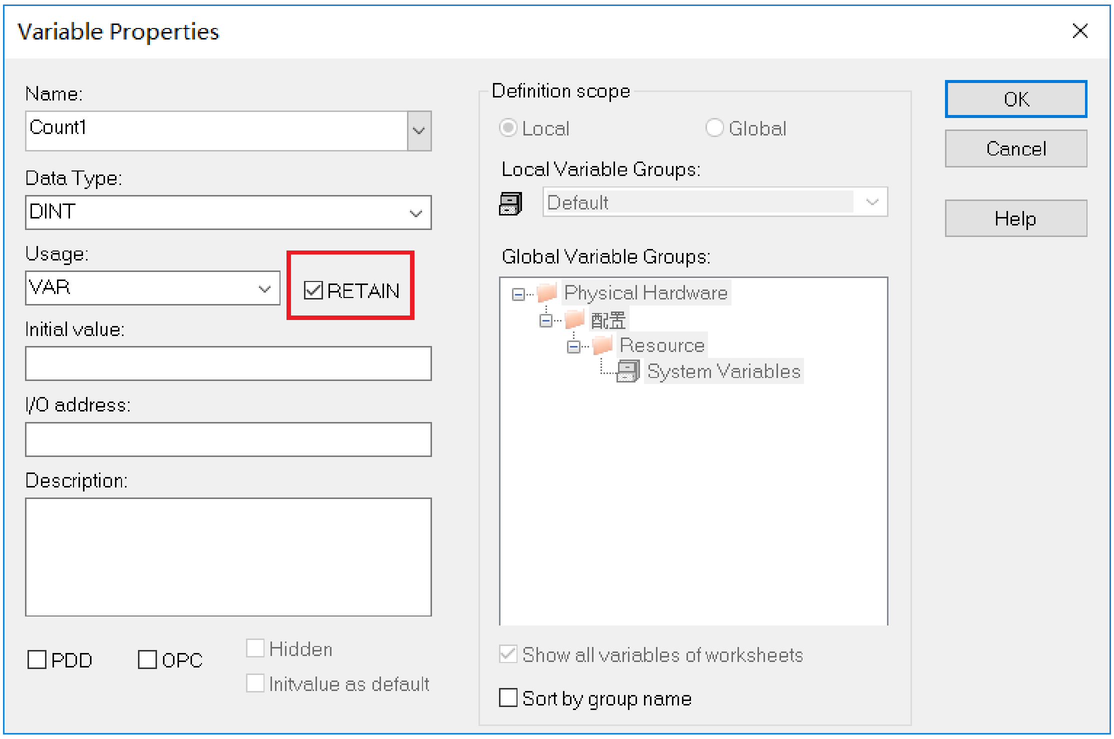
1) variable initial values stored data

When programming, fill in the corresponding initial value under the variable “initial value”. When the motion controller is powered on again, the variable saves the initial value data as shown in the figure below;




2) Variable is set to save the data

When programming, users caNSelect the variable that needs to keep the data after power off. Under the variable attribute “Usage”, check “RETAIN” to indicate that the variable can keep the current value of the variable before the power is turned off after the motion controller is powered off value. Generally, it is used when not communicating with peripherals. (Users need to pay attention to the “RETAIN” when the variable is checked. No natural number can be filled in the “initial value”. Otherwise, the value of the variable is still the data of the initial value after repowering, instead of The data modified before power down) as shown.




NOTE: The difference between the two is the "variable initial value save data", when the motion controller on again, only the initial value of this variable is maintained even when the program is running value of this variable is modified; and "variable set to save data "is data that variable before remain off.

3) Special power-dowNSave data address

In order to facilitate data exchange with peripherals, the %MB3.2000~%MB3.3000 data address is saved by power-down. The user does not neeDAny settings, just fill in the address to achieve, especially pay attention to the variable hook. "RETAIN", otherwise the compiler can't pass. (For more details oNSpecial registers, please refer to: Annex V Register Description)

# IV Data Types

## 4.1 Basic Data Types

The program includes two parts: code and data. The code can be any one of five programming languages: IL, ST, FBD, LD, SFC, or a combination of several languages. The data is divided into three types: basic data type, derived data. Types and user-defined data types, data must exist in the form of variables, the data type determines the format of the variable, the number of bits, the initial value of the range of possible values.

Declared in the basic data type

| type of data | description | Bit length | range | The default initial value |
|---|---|---|---|---|
| BOOL | Boolean | 1 | 0 or 1 (ture / falase) | 0 |
| SINT | Short integer | 8 | -128 to +127 | 0 |
| INT | Integer | 16 | -32768 to +32767 | 0 |
| DINT | DINT | 32 | -2147483648 to +2147483647 | 0 |
| USINT | Unsigned short | 8 | 0 to 255 | 0 |
| UINT | Unsigned int | 16 | 0-65535 | 0 |
| UDINT | Unsigned double integer | 32 | 0-4294967295 | 0 |
| REAL | Real | 32 | -3.402823466 E + 38 to -1.175494351 E-38 and +1.175494351 E-38 to +3.402823466 E + 38 NOTE: scientific notation of decimal 7 | 0.0 |
| LREAL | Long Real | 64 | -11.798E + 308 to -2.225 E-308 as well as + 2.225E-308 to +1.798 E + 308 | 0.0 |
| TIME | Time | 32 | 0 to 4294967295 ms | T # 0S |
| BYTE | Byte | 8 | 0 to 255 (16 # 00..16 # FF) | 0 |
| WORD | Word | 16 | 0 Dao 65535 (16 # 00 ... 16 # FF) | 0 |

| | | | | |
|---|---|---|---|---|
| DWORD | Double Word | 32 | 0 to 4294967295 (16 # 00..16 # FFFFFFFF) | 0 |

## 4.2 Generic Data Type

The generic data type is to group the basic data types hierarchically, with ANY as the prefix of the data type. For example, ANY_INT indicates that all integer data including SINT, INT, DINT, USINT, UINT, and UDINT are included. If the input or output of a function block is connected to ANYINT, it means that this function block can handle variables of integer data such as SINT, INT, DINT, USINT, UINT and UDINT.

Generic data types are organizeDAs follows:

| ANY | | | | |
|---|---|---|---|---|
| ANY_NUM | | ANY_BIT | STRING | TIME |
| ANY_REAL | ANY_INT | BOOL<br>BYTE<br>WORD<br>DWORD | | |
| REAL<br>LREAL | SINT<br>USINT<br>INT<br>UINT<br>DINT<br>UDINT | | | |

## 4.3 User-defined data types

User-defined data types must be inserted into the user-defined data type in the project "data type", which must be done with the “TYPE ...END_TYPE” declaration block. The middle part of the declaration block is the defined derivative data, and the derived data type can be the structure,or an array.

◆ Array

An array is a collection of single data type objects. Like a basic data, it has a unique name. A single object is not named, but the user can access it through its position in the array. An example of an array is as follows:

```
TYPE
graph: ARRAY [0 ... 23] OF INT;
END_TYPE
```

Note: the lowest byte array ARRAY graph is graph [0]

◆ structure

A structure is a collection of objects of different data types. Like a basic data, it has a unique name. A member of a structure is a basic data type or an array type, or it can be another structure, or nested. An example of declaring a structure is as follows:

```
TYPE
machine:
STRUCT
x_pos: INT;
y_pos: INT;
depth: INT;
rp: INT;
END_ STRUCT;
END _TYPE
```

◆ String

A string is a finite sequence of multiple characters. Each character occupies one byte. The data type of the string is STRING. When a string is declared, its length is set in parentheses after the data type, anDA string is declared example as follows:

```
TYPE
STRING10: STRING (10);
END_TYPE
```

In this example, the length of the string is 10, i.e. STRING10 is a string containing 10 characters. 1 is the shortest string length, the longest string length of 32,766.

## 4.4 constant data representation

| type of data | description | Bit length | Examples representation |
|---|---|---|---|
| BOOL | Boolean | 1 | BOOL # 0 |
| SINT | Short integer | 8 | SINT # - 128 |
| INT | Integer | 16 | INT # -32768 |
| DINT | DINT | 32 | DINT # -2147483648 |
| USINT | Unsigned short | 8 | US INT # 255 |
| UINT | Unsigned int | 16 | UINT # 65535 |
| UDINT | Unsigned double integer | 32 | UDINT # 4294967295 |
| REAL | Real | 32 | REAL # 3.1415629 |
| LREAL | Long Real | 64 | LREAL # 3.1415629 |
| TIME | time | 32 | T # 10MS, T # 10S, T # 10M, T # 10H, T # 10D, T # 1D_10H |
| DATE | date | ~ | D # 2011-07-24 |
| TIME OF DATE | time | | TOD # 15: 23: 4555 |
| TIME and DATE | Date and time | | ADT # 2011-07-24 15: 23: 4555 |
| BYTE | byte | 8 | BYTE # 16 # FF |
| WORD | word | 16 | WORD # 16 # FFFF) |
| DWORD | Double | 32 | DWORD # 16 # FFFFFFFF) |

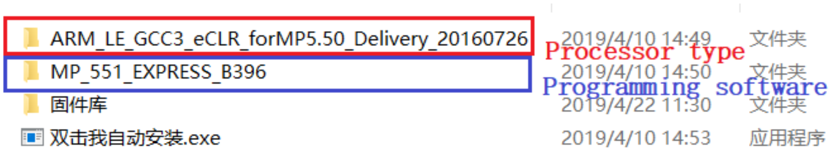
| | | | |
|---|---|---|---|
| | Word | | |
| STRING | String | | 'VECTOR' |

# Ⅴ Software Installation and Introduction

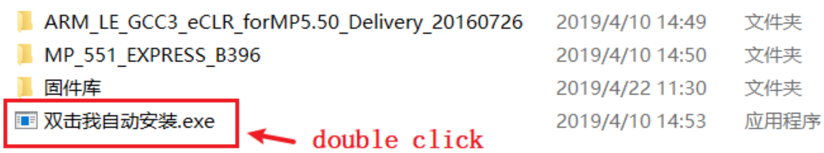
Thanks to MULTIPROG excellent man-machine interface, just a few easy steps to create a project. This section describes how to install the software MULTIPROG description, all the software interface to the end-use configuration MULTIPROG introduced.

## 5.1 MULTIPROG software installation and startup

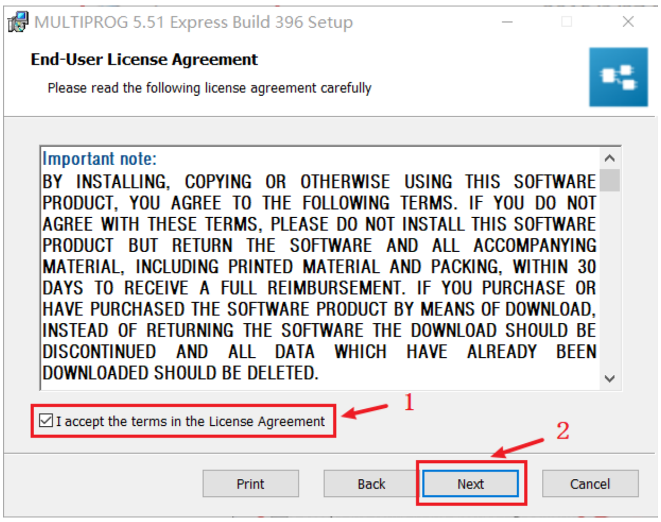
1) Decompress the installation file of MULTIPROG. The "X" and "Y" in the folder name are numbers. After decompression, a file named "MULTIPROGX.XXBuIldYYY" will be generated, indicating the version number of the installation package. Open the folder, which will appear as shown



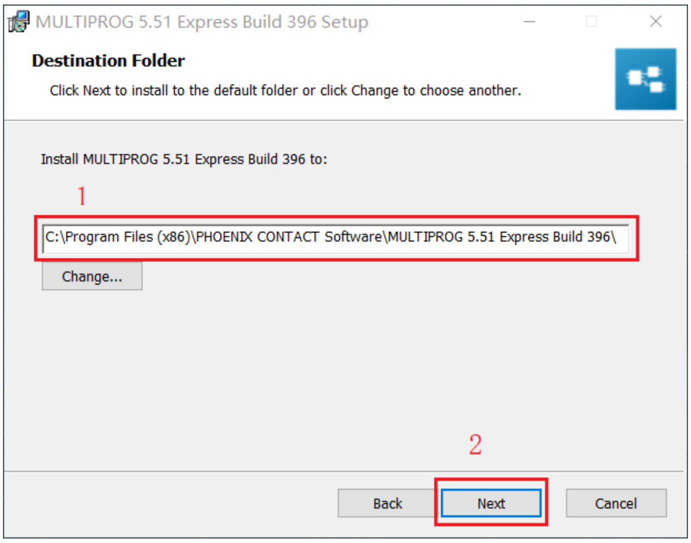

2) Double click to open "双击我自动安装" as shown



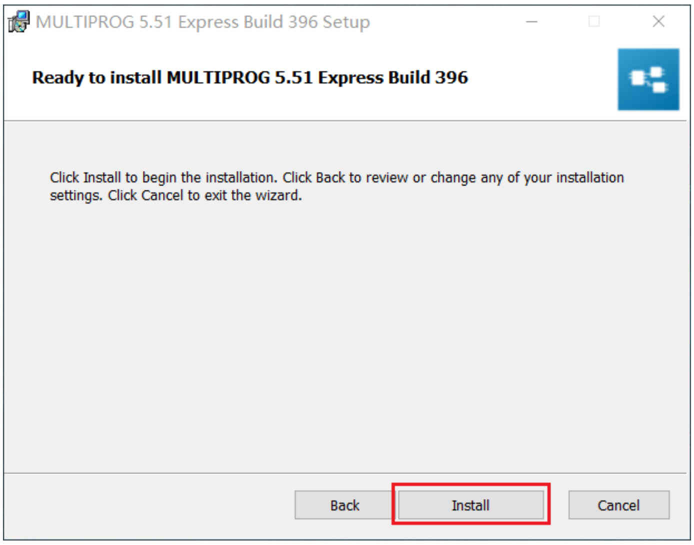

3) In the pop-up dialog box, check "I accpet.........." as shown.



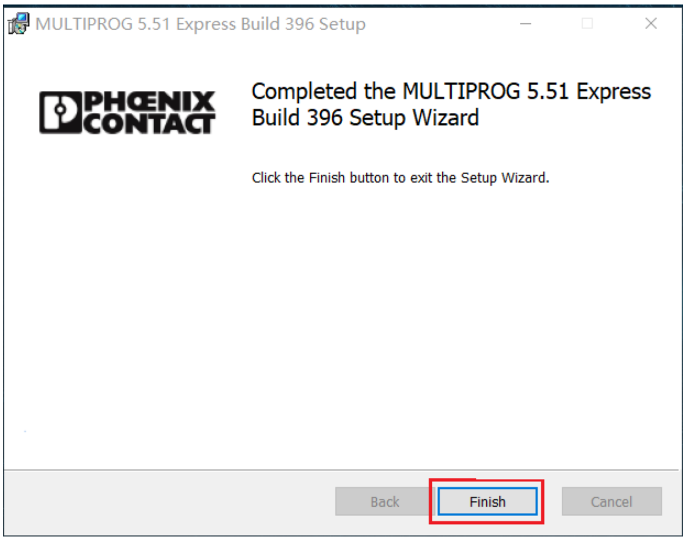

4) Select the installation path, the user can choose it, here choose the default, click "Next" as shown




5) Click the Install, then wait for the completion of the installation, the installation process may take some time, as shown.



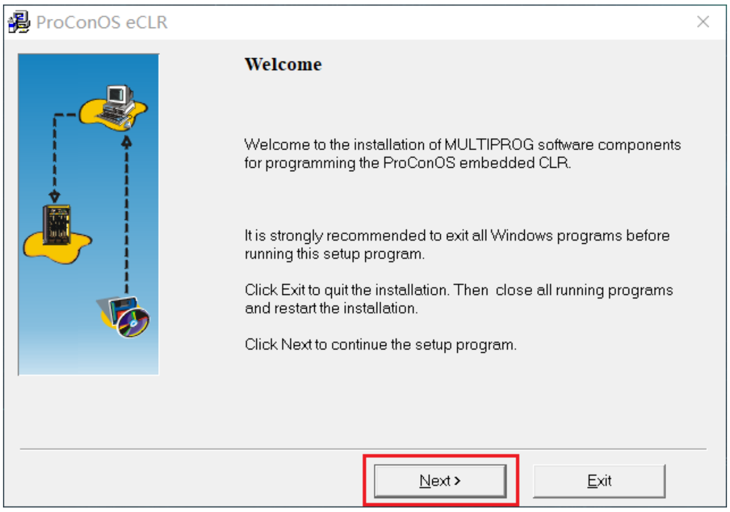

6) Click "Finish" to complete the installation of the MULTIPROG programming software as shown.



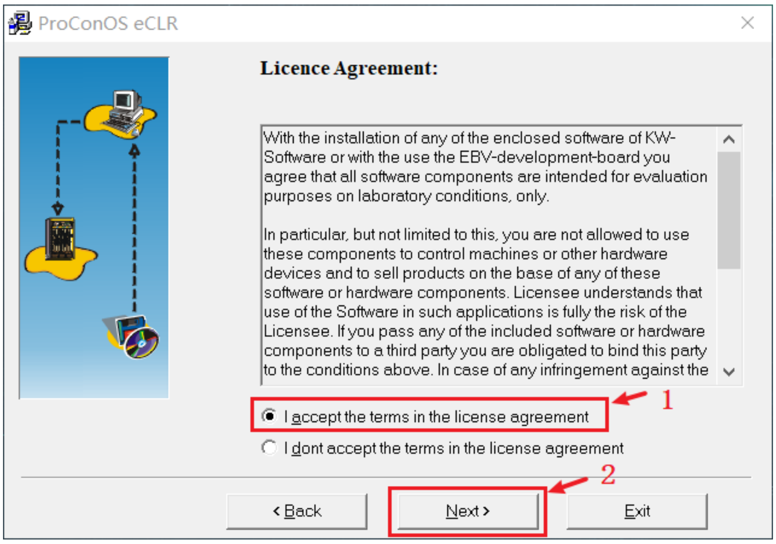
7) Double-click the icon to start MULTIPROG. If the icon does not appear on your desktop, you can also use the "Start Menu"→"program"→"PHOENIX CONTACT Software"→"MULTIPROG 5.51 Express" to start.

## 5.2 Processor type software installation

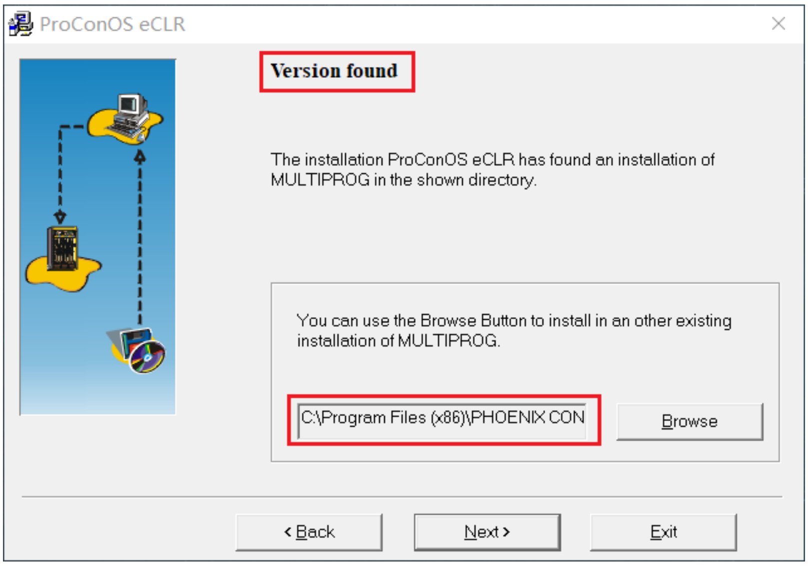
1) After installing the mmmm software, continue to install the processor type software. When the installation dialog appears, click "Next" as shown



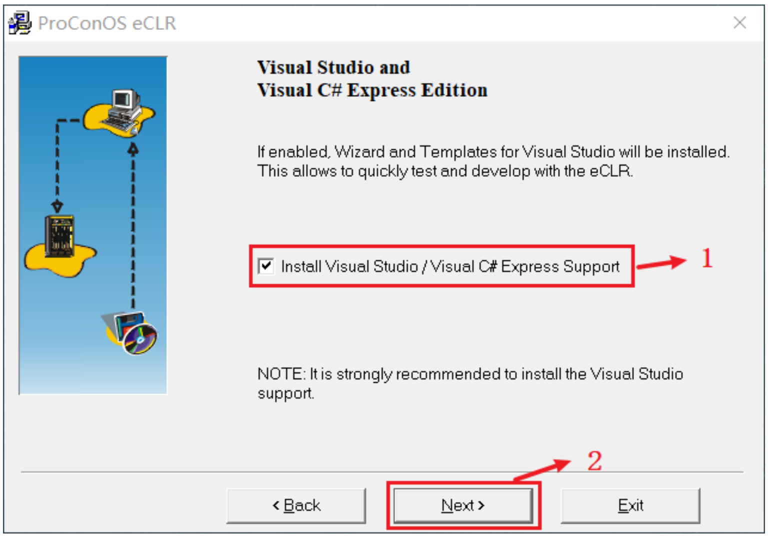

2) Select "I accept ......", click on "Next" as shown.



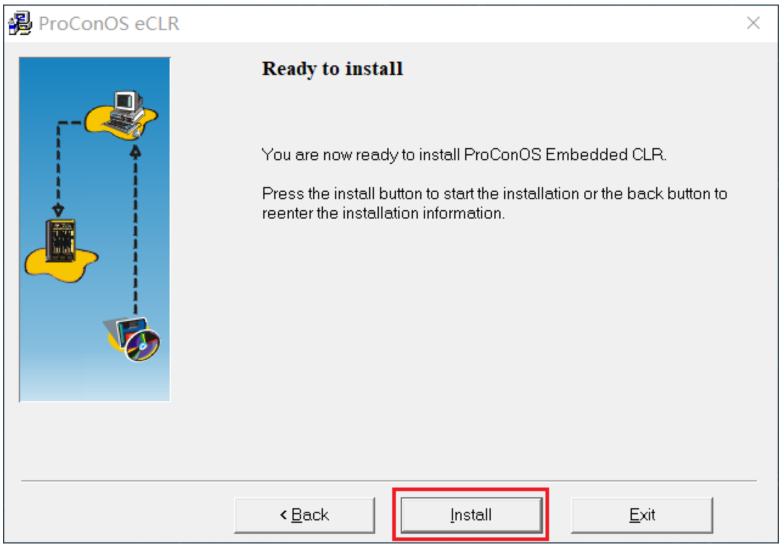

3) Select the installation path, here select the default, display "Version found", click "Next" as shown in the figure (special attention; the processor type software must be consistent with the installation path of the MULTIPROG programming software, the processor type software defaults with the installation The installation path of MULTIPROG programming software is the same, no need to change. If “Version not found” is displayed, the installation path needs to be changed to be the same as the installation path of MULTIPROG programming software.




4) Select "Istall Visual Studio .." click "Next" as shown.



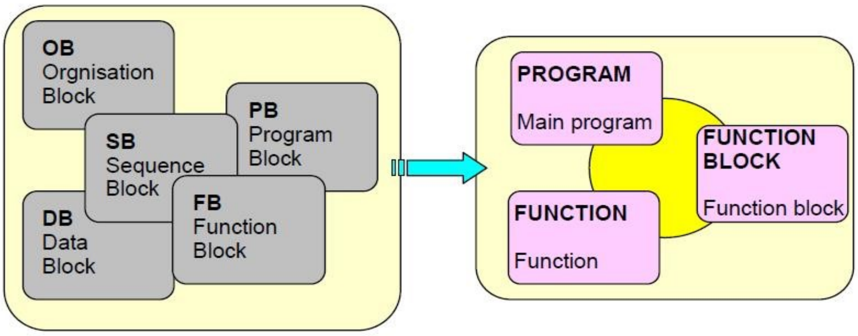

5) Click the "Install" installation, the installation process will take some time, as shown.



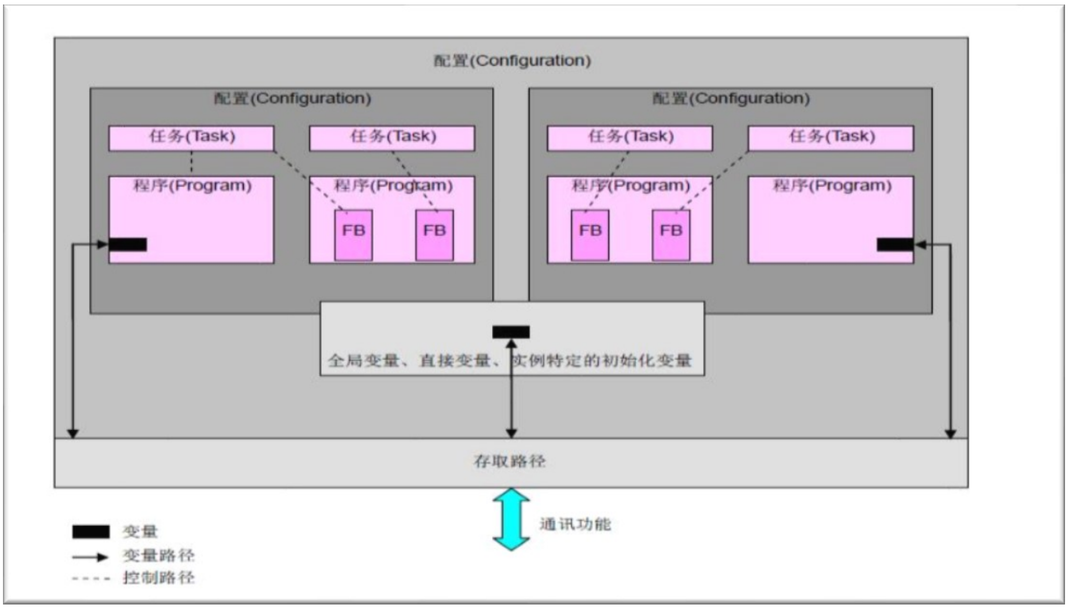
6) After installation is complete, restart the computer, the processor type of software and programming software will builDA relationship.

## 5.3 programming model with standard IEC61131-3

1) The MULTIPROG programming software used by the VA motion controller has a programming language and program structure in accordance with the IEC 61131-3 programming system. In IEC 61131-3, the establishment of programs and projects is done in the Program Organization Unit (POU). The unit POU consists of three parts: PROGRAM, FUNCTIONBLOCK and FUNCTION. It replaces the five functional blocks OB, PB, DB, SB and FB of the traditional PLC programming language. More efficient and more concise as shown.

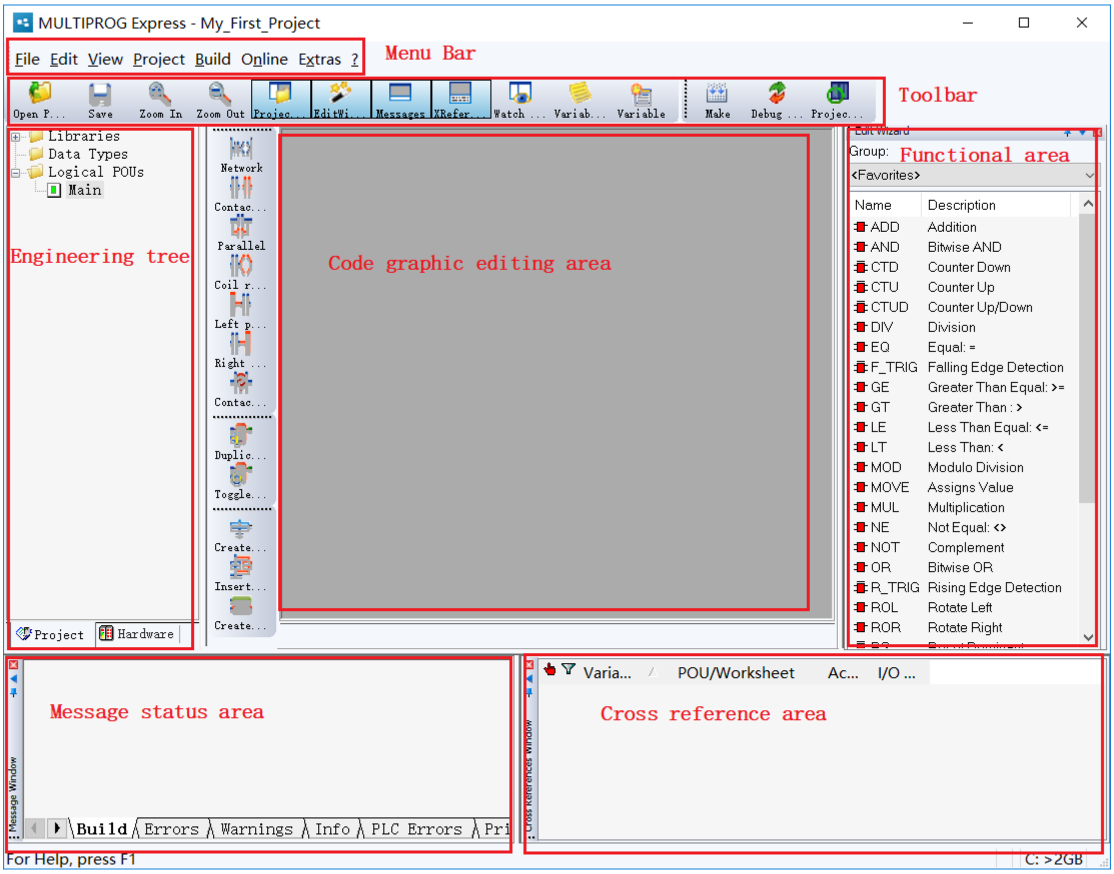



2) IEC 61131-3 programming software model is represented by a hierarchical structure, as shown below, the software model describes the relationship between the various parts, including the configuration, resource, task, program organization unit, global variables, I / O configuration, etc. . Programming process can program a complex program, or divided into a number of small modules can also be simultaneously downloaDA plurality of separate programs, running, or the program into a plurality of tasks to perform, improve the modularity and operational procedures efficiency as shown.



# 5.4 MULTIPROG programming interface presentation

After opening MULTIPROG, you caNSee that it has only one main boundary. According to the function, it is divided into different areas, as shown.




## 5.4.1 Introduction partition function

(1) The toolbar area contains commands for code editing and debugging special functions;

(2) The engineering tree is used to display the structure of the project and the configuration properties of the hardware; the project tree includes two parts, "hardware" and "engineering", which respectively correspond to the hardware and software parts of the established project.

(3) The code graphic editing area is used to edit text or graphic code in the editing state, and is used to display the value of the variable and the running state of the program in the debugging mode;

(4) The message status area is used to display various information when creating a project, online debugging, and running a program;

(5) Cross reference area, you caNSee the current status anDAddress of the variable;

(6) Variable monitoring window, there may be a lot of variables in the large engineering project, you can add the variables you need to monitor, so that you can quickly view the current state of the variables, and facilitate user debugging;

## 5.4.2 hardware

Open the project has been established, on the left side of the " Project Tree " window, click the " Hardware " tab, you caNSee the "physical hardware " , " physical hardware " can display the software model of the structure, the user can view each of the layers , settings.

### (1) Physical hardware

The " physical hardware " tree can reflect the program structure conforming to IEC 61131-3 . It is the entire configuration file of the entire project and is responsible for managing its next layer - " configuration " . Currently, the MULTIPROG Express version only supports one " configuration " . Insert multiple configurations, but you can delete " Configuration " or copy a " Configuration " from another project .

### (2) Configuration

“ Configuration ” is the first layer in the software model. The next layer of “ physical hardware ” is equivalent to the programmable controller system and is responsible for managing its next layer of “ resources ” . Currently, the MULTIPROG Express version only supports one “ Resources " , you cannot insert multiple resources, but you can delete " resources " or copy a " resource " from another project . The type of programmable controller can be viewed by right-clicking “ Configuration ” to select the attribute . It is “ eCLR ” in the PLC type drop-down list in the PLC/ Processing tab and cannot be changed.

### (3) Resources

" Resources " is the second layer in the software model. In the next layer of

" configuration " , it is equivalent to the processor of the programmable controller, responsible for managing its next layer - " Tasks ", " Globales_Variables ", " I /0_ Configuration ” , all three cannot be deleted. The type of processor can be selected by right-clicking on " Resources " , selecting Properties, and selecting " ARM_LE GCC3 " in the Processor Type drop-down list in the PLC/ Processing tab . When no PLC is connected , it is optional, " eCLR Simulation" " Processor settings can be viewed by right-clicking on " Resources " and selecting " Settings " . You caNSee the communication protocol, IP address, processor version, etc. of the programmable controller .

### (4) Tasks

1 ) " Tasks " is the third layer in the software model. In the next layer

of " Resources " , multiple tasks can be inserted under " Tasks " , which can be cyclically scanned or cycled. If the inserted task is For periodic scans, different scan cycles and priorities can be set , which is the multi-tasking feature of MULTIPROG .

2 ) When creating a project, MULTIPROG automatically declares a " Tasks " and the first task in this directory. Users can right-click " Tasks " to insert a new task, or copy a task from another project, insert The task type defaults to " DEFAULT " , which is a cyclic scan. You can also select " CYCLIC " , which is a periodic scan. After selecting " CYCLIC " , you need to set the time interval, priority and monitoring timing. Right-click " Task " and select " Properties " and " Settings " in the pop-up dialog box to view the type and scan period of the modified task .

3 ) After inserting the new " task " , right-click the newly inserted task name, select " Program Instance " to specify the program instance name and instance type, and the instance type is the program inserted in the POU ( PROGRAM ).

In a project, only one task of type DEFAULT is allowed . Others default to CYCLIC type tasks; under one task, multiple program instances can be inserted, and the execution order of multiple program instances is executed in the order in which they appear under the task

. .

**(5) Global_Variables - global variables**

" Global_Variables " is the third layer in the software model. In the next layer of " Resources " , it is juxtaposed with " Tasks " . Global_Variables cannot be copieDAnd pasted. The global variable is a variable table, including system variables provided by MULTIPROG and user-created variables. User-created variables will only appear in this table if they are specifieDAs VAR_GLOBAL .

**(6) IO_Configuration - IO configuration**

" IO_Configuration " is the third layer in the software model. Next to " Resources " , alongside " Tasks " and " Globa . l_Variables " ,IO_Configuration cannot be copied or pasted. Double-click " IO_Configuration " open the I / O configuration dialog, which is used to edit I / O configuration worksheet, including the INPUT (input), the OUTPUT (output) , VARCONF property settings, the user simply set the INPUT , the OUTPUT to In INPUT , there is the name of INPUT , the default " IN " , the task to which it belongs, the logical start address, the driver parameters of the board /IO module. In the driver parameters, the user needs to specify the driver name. The default is DUMMYIO. , the user needs to be changed to " KWIO " .

For the configuration procedure, see " 7 .4 IO configuration."

## 5 . 4 .3 Project

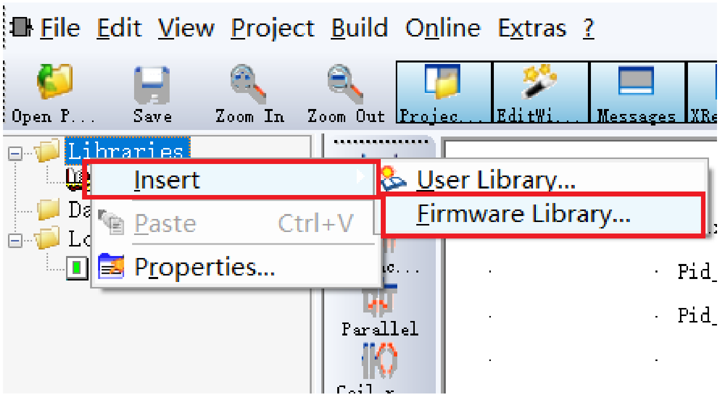
The project consists of three parts: library, data type, and logical POU , which form a complete and powerful program.

### (1) Library

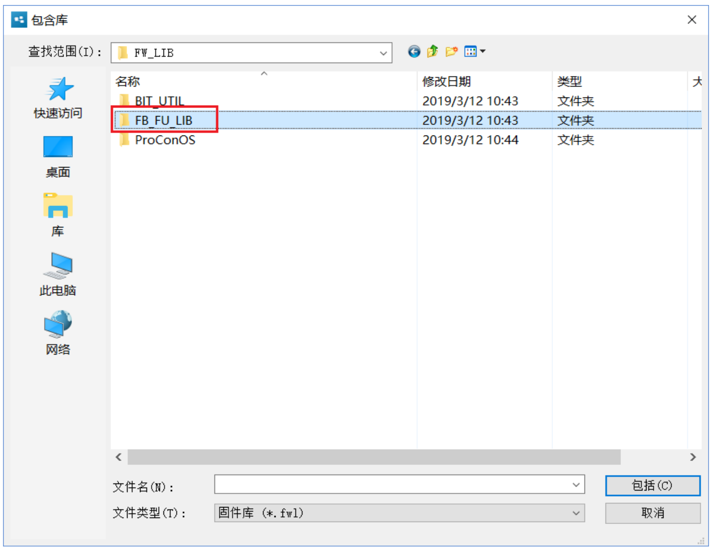
1 ) The library provides function blocks, functions, programs, and data types. After inserting a library, the user can use the functions and function blocks in the library as if they were IEC function blocks. Right click on " Library " to insert " User Library " and " Firmware Library " . These two libraries are not required, and users should choose to insert according to the needs of their own programs. Library users are other projects created by the user, the user library file extension name * .mwto firmware library function is a special function, function block, requires the user to insert a separate work process, firmware library file name extension * .fwl ,

The following example shows how to insert a firmware library.

1> Right click on the "Library" in the project tree window, select "Insert" and select "Firmware Library" as shown




2> In the pop-up " Include Library" window, open the " FB_FU_LIB " folder (you need to copy the folder to the default directory before this operation) , then click the file " FB_FU_LIB.FWL ", then click Including, as shown in the figure , the firmware library insertion can be completed.




### (2) Data type

1 ) If a user to define their own data types ( eg : arrays, structures, etc. ) , these data types must be in the " data type " declaration. Right-click " Data Type " , select Insert " Data Type " , specify the name of the data type worksheet, double-click the generated data type work order, enter the editing area, type the following characters as shown

example:

```
TYPE
    DATA1:ARRAY [1..100] OF INT;
END_TYPE
```

Above code defines a containing 100 th the IN T are array variables, array name DATAl .

### (3) Logical POU

1 ) The program organization unit POU is a language element of the PLC program. They contain the prograMCode is the most small, independent software units. The name of the POU must be unique within the project. The right-click " Logical POU " can be inserted into the following three program organization units :

A: 〗 program ( PROGRAM )

B: 〗 function block ( FUNCTION BLOCK )

C: 〗 function ( FUNCTION )

2 ) Each POU consists of two different parts : the variable work order and the code ontology, which are all variables that appear in the POU in the variable worksheet . A POU code works with single-user IL, ST , FBD, the LD, the SFC five programming languages are written one, where IL is an instruction list programming language ( Instruction List ) ST is a structured text programming language ( Structured the Text ) ; FBD is a functional block diagram programming language ( function block Diagram ) , the LD is the ladder programming language ( Relay ladder Logic Diagram ) ; the SFC is a sequential function programming language ( the sequential function the Chart ) .

🕮·function

" Function " , abbreviateDAs FU , is a program organization unit POU with multiple inputs and one output . Similar to functions in high-level programming languages, the return value of " function " can be simple data types such as BOOL, IN T, etc. " function " internal can call another " function " , but can not be called " functional blocks " or " program " does not allow recursive call. When declaring a " function, you must declare input and output variables, intermediate variables, and external variables in the variable worksheet of this " function " .

List of features supported by MULTIPROG :

| | |
|---|---|
| Type conversion function | Such as : IN_TO_REAL |
| Numerical function | Such as : ABS and LOG |
| StandarDArithmetic function | Such as : ADDAnd MUL |
| Bit string function | Such as : ANDAnd SHL |

| | |
|---|---|
| Selection and comparison function | Such as : SEL and GE |
| String function | Such as : RIGHT and INSER T |
| Time data type function | Such as SUB with TIME data type |

**·function block**

" Function block " , abbreviateDAs the FB , is a program with multiple inputs and multiple outputs organizational units the POU, " functional blocks " can call another internal " function block " or " function " , but can not be called " program " , Recursive calls are not allowed. All " function blocks, " ( IEC defined, library library FB and user-defined FB ) can be easily inserted into the user's " function block " or " program " . When declaring a " function block " , you must declare input and output variables, intermediate variables, and external variables in the variable worksheet of this " function block " .

List of function blocks supported by MULTIPROG :

| | |
|---|---|
| Bistable element, | Such as SR and RS |
| Edge detection function block | Such as : R_RIG and F_TRIG |
| counter | Such as : CTU and CTD |
| Timer function block | Such as : TON and TOF |

**·program**

A " program " is a combination of prograMCode that contains functions and function blocks. The behavior and use of a " program " is similar to a function block. It can have input and output parameters, can have internal storage, but does not allow recursive calls. When creating a project, MULTIPROG automatically declares a " program " . When a new " program " is declared ,

MULTIPROG also generates a variable worksheet for the " program " ( double-click the newly declared program, then click on the variable work order) this variable may enter worksheet program ) , and load it into tasks first task in the directory, this can cut and paste into another application tasks. As describeDAbove, in the Tasks →·insertion program instance at a task to be input program name and type instances, this program is a program instance type →·logic POU inserted under the " program " , so that, in a plurality of tasks can be inserted Multiple program instances, the names of these program instances can be different, but can be the same program instance type, that is, a program instance type can be executed in multiple tasks. The " program " must be linked to the task.

# Ⅵ MULTIPROG Programming Language

MULTIPROG supports IL, ST , FBD, LD, SFC five programming languages, of which IL and S T belong to the text programming language, FBD, LDAnd SFC belong to the graphic programming language. A program with independent functions is divided into code part and data part. The code is written in one or several languages of IL, ST , FBD, LD, SFC , and the data is declared in the variable work order. This chapter describes how to declare variables and how to program them in these five programming languages.

〖1 〗IL is an abbreviation of the Instruction List ;

〖2 〗ST is an abbreviation of Structured Text structured text;

〖3 〗FBD is an abbreviation of Function Block Diagram function chart;

〖4 〗LD is an abbreviation of Ladder Diagram ladder diagram;

〖5 〗SFC is SequentialFunction Chart is an abbreviation of sequential function chart;

In a graphics-like programming language, programs are scanned from top to bottom and left to right. In a text-based programming language, programs are scanned from top to bottom.

# 6 .1 Variable Worksheet

IL, ST , FBD, LD, SFC Five programming languages need to declare variables in the variable worksheet corresponding to each program organization unit POU . The user selects the program, function or function block in the POU directory in the project tree , clicks the menu. Variable work order on the bar, enter the variable work order, select the first line as follows , select the additional variable, MULTIPROG automatically insert ADefault variable, (you can also create a variable set, fill in the name with the name plus # ; such as m# ; fill iNStart and stop addresses and select data types, click OK to create multiple variables) as shown

| | Name | Type | Usage | Description | Address | Init | Retain | P... | O... |
|---|---|---|---|---|---|---|---|---|---|
| 1 | ⊟ **Default** | | | | | | | | |
| 2 | MC_AXIS_REF_1 | MC_AXIS... | VAR | | | | ☐ | ☐ | ☐ |
| 3 | Axis0 | USINT | VAR | | | 0 | ☐ | ☐ | ☐ |
| 4 | ControlMode | INT | VAR | | | 0 | ☐ | ☐ | ☐ |
| 5 | Moter_Max_V | DINT | VAR | | | 3000 | ☐ | ☐ | ☐ |

The name column is the name of the variable, the default NewVarl , the user can modify the variable name, the variable name must start with a letter, can contain letters, numbers and underscores ; the type column is the data type, the user can directly type the user name, or select through the drop-down menu The usage bar, which indicates the scope of the inserted variable :

. 1 ➢insertion procedure, only VAR and VAR the EXTERNAL two options, VAR represents the internal variables, var_ EX T ERNAL represents an external variable ;

2 For the inserted function, there are only two options VA R and VAR_INPUT , VAR for internal variables and VAR_INPUT for input variables ;

3 For inserted function blocks, there are VAR, VAR_EXTERNAL, VARIN_OUT, VAR_ INPUT and VAR _OUTPUT, VAR for internal variables, VAR_EXTERNAL for external variables, VAR_IN OUT for input and output variables ;

⮙·Description column : is the text description input by the user ; the address bar is the input, output, and intermediate variables of the variable ;

⮙·address bar : indicate the address of the variable ;

⮙·initial value column : In the PLC program, the first time the variable is used, the initial value indicated here will be used ;

⮙·Hold column : In the case of PLC power failure, the value of this variable is still saved, after the warm start, the last value of the variable will be used ;

⮙·PDD column : Indicates that the variable has been written to the process datADirectory ( PDD ) , and is checked only when the user accesses the variable name corresponding to an address on the PLC .

⅄·OPC column : indicates variable has written OPC server file, only when the user wishes via OPC only access the variable client check on it.

FBD, LDAnd SFC programming language compile time course, the variables may be inserted in the editing area, the variable is inserteDAfter the variable is automatically included in the worksheet. VAR_INPU T represents the input variable and VAR_OUTPUT represents the output variable.

## 6 .2 IL Instruction List Programming Language

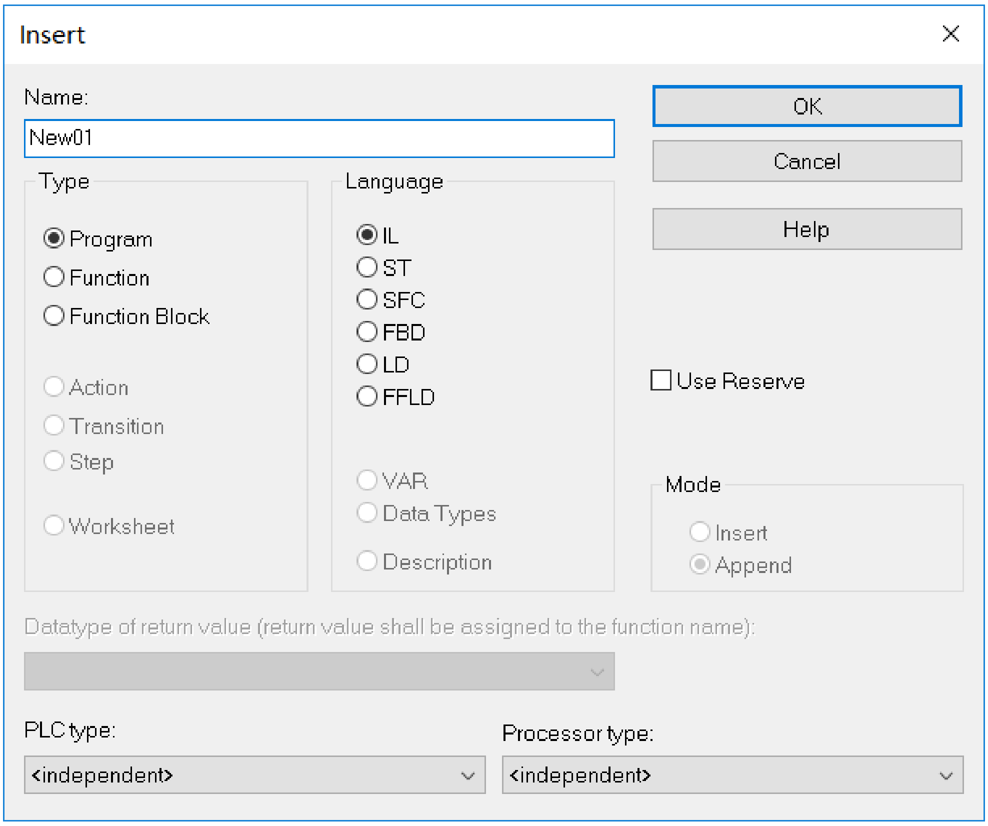
The basic statement of the instruction list programming language is the instruction list, which is an underlying language that uses machine-oriented operators and is relatively easy to convert to machine code of a programmable controller. Because of the lack of effective tools, the instruction list programming language is suitable for small The control program is not suitable for large and complex control tasks.

### 6 .2.1 Creating an IL program

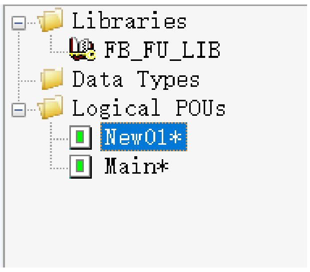
Using IL programming, the user can type code in the editing area to the instruction input directly or in the " Edit Wizard " in drag refers to the editing area. The following is an example of writing an A+B=OUT ( * addition * ) program to illustrate the creation of an IL programming project :

1 > Create a project

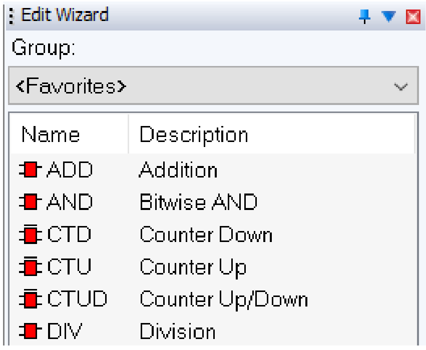
2 > Right-click the " Logical POU " in the project tree to insert the "Program" pop-up dialog box, enter " IL_Test " in the dialog box, select IL in the language bar , and click OK as shown .




3> After completion of the step, there is a program in the project tree POU one of IL _ the Test , as shown .




4 > Double-click " logic POU under" of IL _Test , theNSelect Edit Wizard region "all FU and the FB " functions and function blocks all appear as shown .




5> Double-click " the ADD module" , or dragging " the ADD module" into the editor, " the ADD module" function can occur in the editing area, as shown .

```
LD   (* IN1 as ANY_NUM *)
ADD  (* IN2 as ANY_NUM *)
ST   (* Result as ANY_NUM *)
```

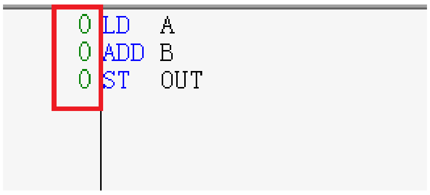
6> The figures were changed to the green portion of the font A, B , and OUT ( the user himself caNSelect the variable name, not necessarily the IN and OUT ), as shown .

```
LD  A
ADD B
ST  OUT
```

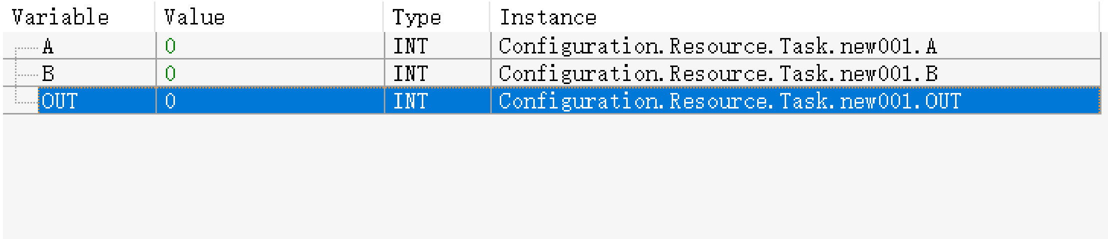
7> Click " Variable Work Order " on the menu bar to declare three variables A, B and OUT with type INT . After the usage is VA R (local variable) , return to IL programming interface as shown.

| | Name | Type | Usage |
|---|---|---|---|
| 1 | ⊟ **Default** | | |
| 2 | A | INT | VAR |
| 3 | B | INT | VAR |
| 4 | OUT | INT | VAR |

8> click "Create" no error, click "download", and then click the "debugging", as shown .

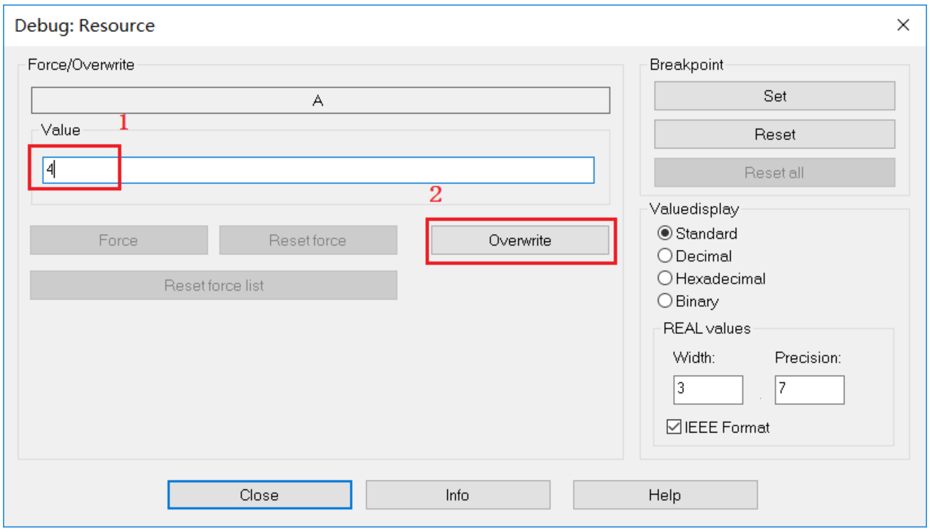



9> Click the variable work table, use the mouse to select the three operands, right-select "Add to watch window", the user can debug and online monitoring variables such as drawing shown .

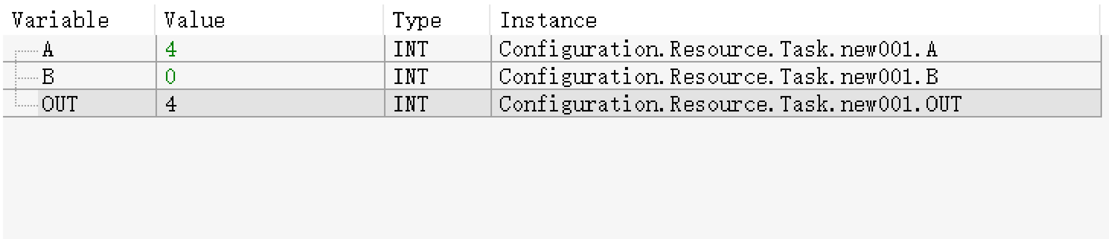


| Variable | Value | Type | Instance |
|---|---|---|---|
| A | 0 | INT | Configuration.Resource.Task.new001.A |
| B | 0 | INT | Configuration.Resource.Task.new001.B |
| OUT | 0 | INT | Configuration.Resource.Task.new001.OUT |

10> are double tap monitoring window A and B variables were assigned 4 and 5 , and then click the cover, as shown .




11> in the editing area can be displayed on the left view window state variable values, as shown . .



| Variable | Value | Type | Instance |
|---|---|---|---|
| A | 4 | INT | Configuration.Resource.Task.new001.A |
| B | 0 | INT | Configuration.Resource.Task.new001.B |
| OUT | 4 | INT | Configuration.Resource.Task.new001.OUT |

## 6 .2.2 IL statement

In the editing area of the instruction list language IL , the statements of the IL programming language can be written line by line , each statement occupies one line, and the front ( non-editing area ) of each line of the statement is a line number, and each line of the instruction is without a semicolon. .

The format of the IL statement is as follows :

| Line number | OperatioNSymbol / instruction | Operand | ( * * ) |
|---|---|---|---|
| 1<br>.<br>.<br>N. | IL operator or instruction | Operators can have 0, 1 or more constant variables, or instruction parameter tables | Comment |

The IL statement distinguishes the parts in different colors : the operator / function is blue, the operand is black, and the comment is green.

Instruction example : Take the ADD instruction as an example to further explain

Find the suMCode of IN and IN2 as follows :

LD IN ( *IN1 as ANY_ NUM* )

ADD IN2 ( *IN2 as ANY_NUM* )

ST OUT ( * result as ANY_ NUM* )

LD is an operator. The variable IN after LD is an operand. The function of LD is to load the following operand IN into the accumulator . ADD is the addition instruction. After adding the value, the result is loaded into the accumulator ; behind ST The variable OUT is also an operand, which is used to assign the data in the accumulator to the subsequent operands ; the green font part is a comment, prompting the user to declare a variable as an operand, and the data type is AN Y_NUM . Table shows the user declares a variable as an operand, the data type of ANY_NUM .

## 6 .2.3 of IL operator

IL programming language in addition to the instructioNSet of instructions is available, to IEC 61131-3 standards following 24 Species instruction as standard instructions.

| Operator | Modifier | Operand | Description |
|---|---|---|---|
| LD | N | ANY | Load the following operands into the accumulator |
| ST | N | ANY | Assign the data in the accumulator |

| | | | |
|---|---|---|---|
| | | | to the subsequent operands |
| S | | BOOL | Operand set 1 |
| R | | BOOL | Complex operand 0 |
| AND | N, ( | ANY _ BIT | Logic and |
| & | N, ( | ANY _ BIT | Logic and |
| OR | N, ( | ANY _ BIT | Logical or |
| XOR | N, ( | ANY _ BIT | Logical XOR |
| NOT | ( | ANY_BIT | Logical negation |
| ADD | ( | ANY_NUM | plus |
| SUB | ( | ANY_NUM | Less |
| MUL | ( | ANY_NUM | Multiply |
| DIV | ( | ANY_NUM | except |
| MOD | ( | ANY_INT | Molding |
| GT | ( | ANY_NUM, ANY_BIT | Compare, greater than, > |
| GE | ( | ANY_NUM, ANY_BIT | Comparison, greater than or equal to, >= |
| EQ | ( | ANY_NUM, ANY_BIT | Compare, equal, = |
| NE | ( | ANY_N UM, ANY_BIT | Comparison, not equal, <> |
| LE | ( | ANY_NUM, ANY_BIT | Comparison, less than or equal to <= |
| LT | ( | ANY_NUM, ANY_BIT | Comparison, less than < |
| JMP | C, N | LABAL | Jump to the instruction at the label |
| CAL | C, N | NAME | Call function block |
| RET | C, N | | Return from the called function, function block |
| ) | | | End of delay |

Note :

1> The modifier N indicates that the inverse operation is performed, such as ANDN, which indicates that the operand is inverted ;

2> modifier C means that it is executed only when the current operation result is true, such as :

```
LD IN1
AND IN2
ST OUT1
JMPC M2      // Execute jump when M2 is true
M2:
LD IN3
ST OUT2
```

# 6 .3 ST structured text programming language

Structured text programming language S T is a high level language, similar to the Pascal programming language, it does not use low-level machine-oriented operators, but using a similar date statements often language to describe control commands, sophisticateDAlgorithms can be described. A structured text language prograMConsists of statements consisting of expressions and keywords with the following characteristics :

◆·There is no jump statement, and the conditional statement is used to implement the branch of the program ;

◆·Each statement ends with a semicolon " ; " ;

◆·with ( ** ) add comments to the program, comments can not be nested, such as ( * ( ** ) * );

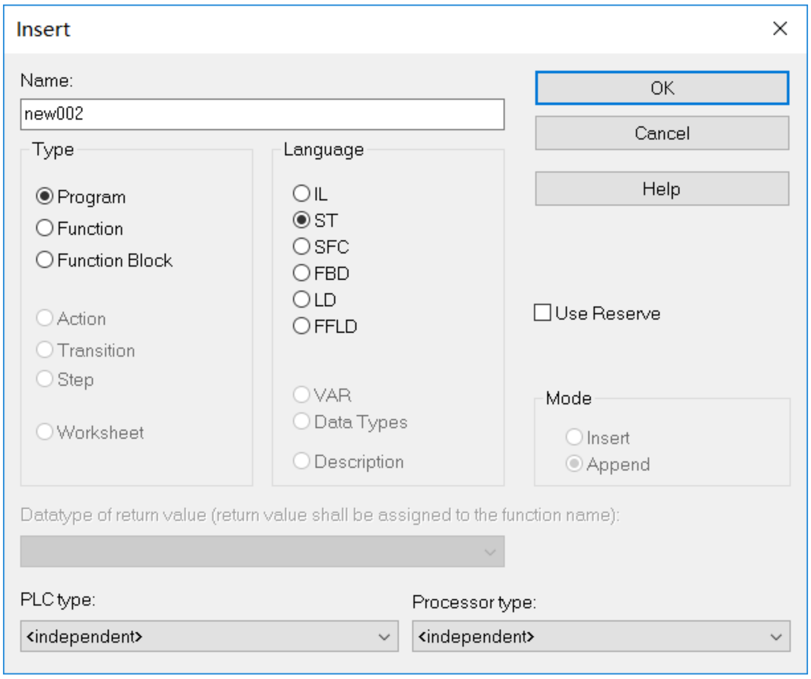
◆·Need to declare input, output, internal, external, global variables in the variable worksheet corresponding to the POU ;

## 6 .3.1 create aNST program

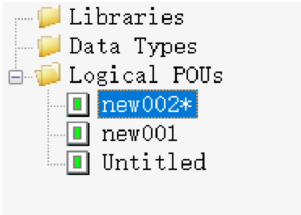
Using ST language programming, the user can type a statement in the editing area, or drag and drop the command directly into the editing area in the " Edit Wizard " , and then type the operanDAnd the ending character " ; " below with an A*B=OUT ( multiplication ) program as an example ST programmed pass procedure :

1> Create a project

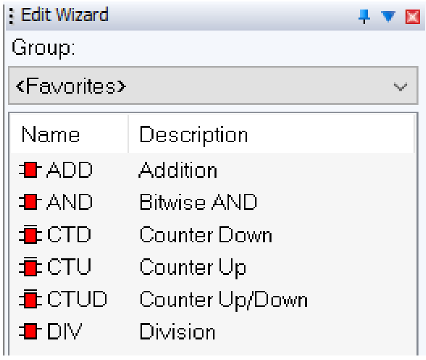
2> Right Project tree " series of the POU " , select Insert a program, pop-up dialog box, type the program name in the dialog box, in this case ST_Text , select the language bar ST , click " OK " as shown.




3> there is a project tree in the new procedure the POU , as shown.




4> Double -click " logic the POU " under " ST_Text " , theNSelect Edit Wizard region " all FU and the FB " , anDAll functions appears function can block, as shown.




5> Double-click " MUL " , or dragging MUL into the editor, MUL function appears in the editing area, as shown.

```
(* Result as ANY_NUM *) := (* IN1 as ANY_NUM *) * (* IN2 as ANY_NUM *);

```

6> The figures were changed to green font part OUT and C and D ; ( user himself caNSelect the variable name, not necessarily the IN and OUT ), as shown.

```
OUT1:= C * D;
```

| | Name | Type | Usage |
|---|---|---|---|
| 1 | ⊟ **Default** | | |
| 2 | C | INT | VAR |
| 3 | D | INT | VAR |
| 4 | OUT1 | INT | VAR |

7> Click " Make " , and "Download" , then click " Debug Switch " , you caNSee the current result on the left side of the edit interface, as shown

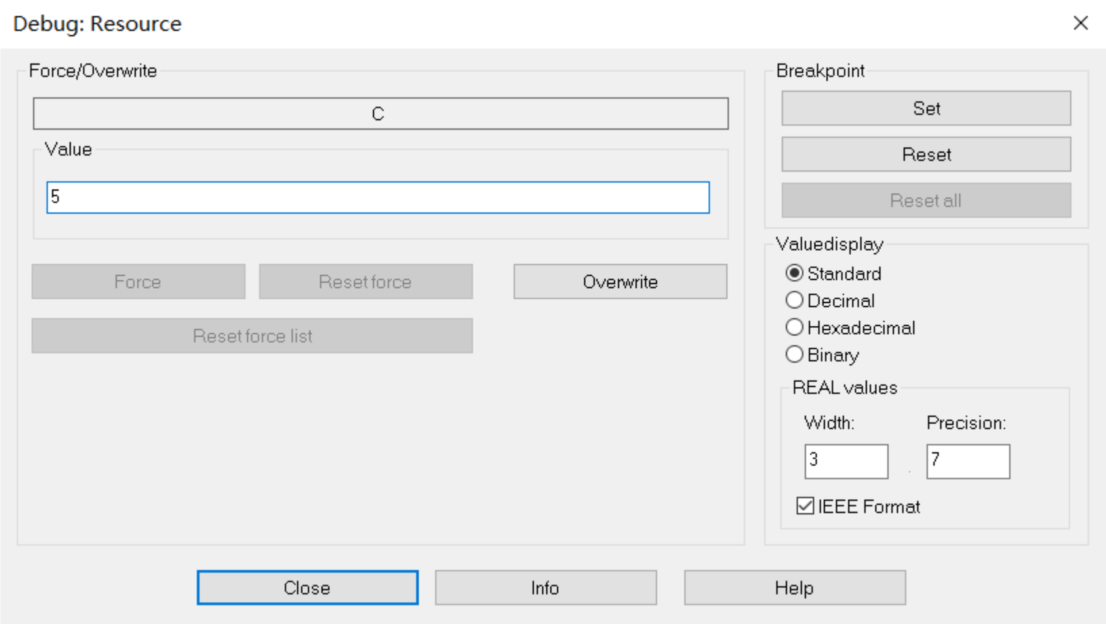
```
0 OUT1:= C * D;
```

8> selected with the mouse are three operands, the right choice to add to the Watch window, the monitor window as shown.

| Variable | Value | Type | Instance |
|---|---|---|---|
| C | 0 | INT | Configuration.Resource.Task.new002.C |
| D | 0 | INT | Configuration.Resource.Task.new002.D |
| OUT1 | 0 | INT | Configuration.Resource.Task.new002.OUT1 |

Watch 1 | Watch 2 | Watch 3 | Watch 4 | Watch 5 | Watch 6 | Watch 7 | Watch 8

9> Double-click the C and D variables in the watch window , write 5 and 3 respectively , and then click "Overwrite" as shown.




10> in the editor window can be displayeDAnd the monitor variable state value, as shown.

| Variable | Value | Type | Instance |
|---|---|---|---|
| C | 5 | INT | Configuration.Resource.Task.new002.C |
| D | 3 | INT | Configuration.Resource.Task.new002.D |
| OUT1 | 15 | INT | Configuration.Resource.Task.new002.OUT1 |

At this point, a complete simple ST program is completed.

## 6 .3.2 ST statement

In the ST programming language, a prograMConsists of statements, and statements consist of expressions and keywords. In the editing area of the structured text programming language ST , the statements of the ST programming language can be written line by line , each statement ends with a semicolon, and multiple statements can occupy one line, and the front ( non-editing area ) of each line of statements is one Line number. Such as an assignment statement, which consists of a variable, an assignment keyword, anDAn expression, which is used to assign the result of the expression to the operation.

Assign the variable to the left of the keyword :
Variable name : = expression ;
The data types on both sides of the assignment keyword must be the same.

## 6 . 3 .3 ST expression

An expression consists of an operanDAnDAn operator, and the operand can be ADirect quantity, a variable, or a function name. The operators that can be useDAre as follows :

The operators that can be useDAre as follows

| Operator | Example | Example value | description | priority |
|---|---|---|---|---|
| ( ) | ( 2+3 ) * ( 4+5 ) | 45 | brackets | Most high |
| ** | 3.0**4 | 81.0 | Power operation | |
| - | - 10 | - 10 | Find the opposite | |
| NOT | NOT TRUE | False | Bitwise negation | |
| * | 10*3 | 30 | Multiplication | |
| / | 6/2 | 3 | Division operation | |
| MOD | 17 MOD 10 | 7 | Modulo operation | |
| + | 2+3 | 5 | Addition | |
| - | 4-2 | 2 | Subtraction | |
| < , > , <= , >= | 4>12 | False | Comparison | |
| = | T#26h 二 T#1 d2h | True | equal | |
| <> | 8 <>16 | True | not equal | |
| & , AND | TRUE&FALSE | False | Boolean | |
| XOR | TRUE XOR FALSE | True | Boolean or | |
| OR | TRUE OR FALSE | True | Boolean or | lowest |

| Description | Keyword | Example | description |
|---|---|---|---|
| Assignment operator | := | OUT:=IN | Assign IN to OUT |
| return | RETURN | RETURN; | Exit the called function, function block or program and return to the statement that called it. |

| | | | |
|---|---|---|---|
| select | IF | IF a<b THEN c:=1;<br>ELSIF a=b THEN c:=2;<br>ELSE c:=3;<br>END_IF: | When the expression ' a<b ' after IF is TRUE , execute a statement after THEN ( with a semicolon as the boundary ) , otherwise the statement after THEN is not executed , and continue to judge ELSIF or ELSE . |
| | CASE | CASE f OF<br>1:a:=3;<br>2: a:=4;<br>3: a:=2;<br>ELSE<br>a:=0;<br>END_CASE; | According to CASE value of the expression after the keyword, a group of statements. The variable or expression ' f ' must be an INT data type. |
| cycle | FOR | FOR a:=1 TO 10 BY 3 DO<br>f[a] :=b;<br>END_FOR; | The variable ' a ' starts at 1 , and the statements of FOR and END_FOR are executed repeatedly . For each execution, a increases by 3 and ends with 10 . All values must have an ANY_INT data type. |
| | WHILE | WHILE b>1 DO<br>b:=b/2;<br>END_WHILE; | When the value of the expression `b>1 ' is TRUE , the statements of WHILE and END_WHILE are executed repeatedly until the value of ' b>1 ' is FALSE . |
| | REPEAT | REPEAT<br>a:=a*b;<br>UNTIL a <10000<br>END_REPEAT; | The statements of REPEAT and END_REPEAT are executed repeatedly until the value of the expression ' a<10000 ' is TRUE . |
| End of cycle | EXIT | FOR a:=1 TO 2 DO<br>IF flag THEN EXIT;<br>END IF<br>SUM:= SUM+a<br>END_FOR | An exit statement can be used to abort the execution of a loop statement. |
| End of statement | ; | | Putting it after the statement indicates the end of the statement, or it can exist separately. |

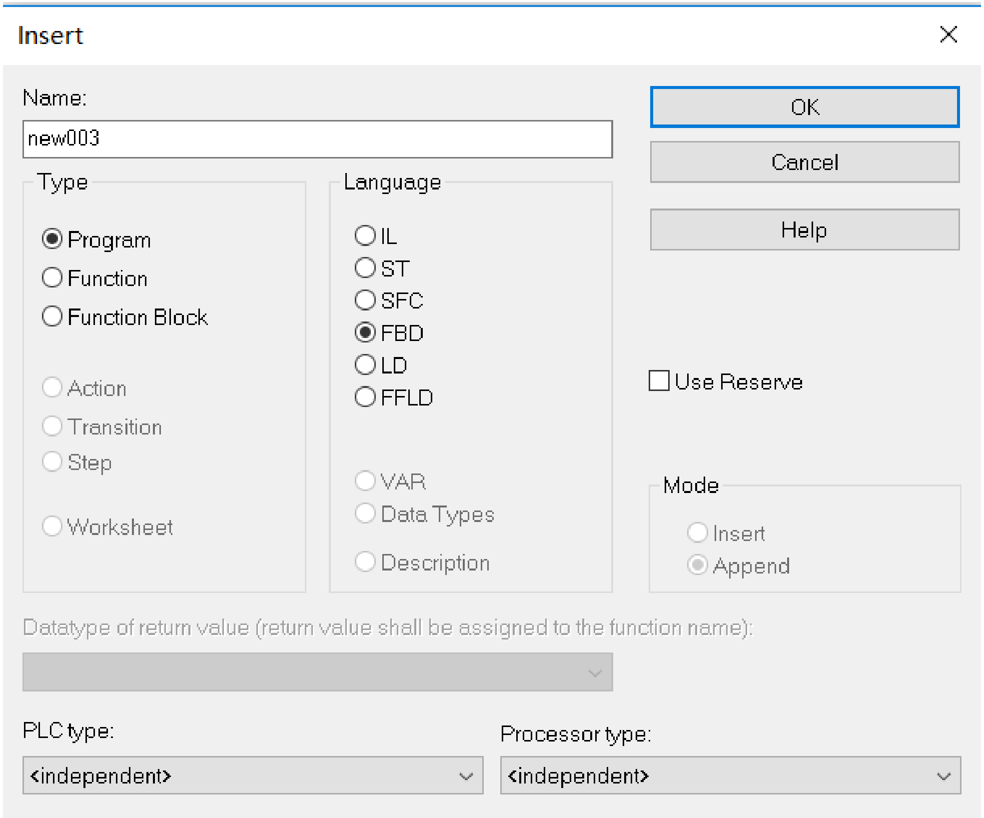
**ST language common keywords**

# 6 .4 FBD Function Block Diagram language programming

The function block diagram programming language is derived from the field of signal processing. It is the basis of the IEC 61499 standard. A function block diagram programming language prograMConnects various function blocks. The elements of the programming language are functions, function blocks and connectioNSymbols.

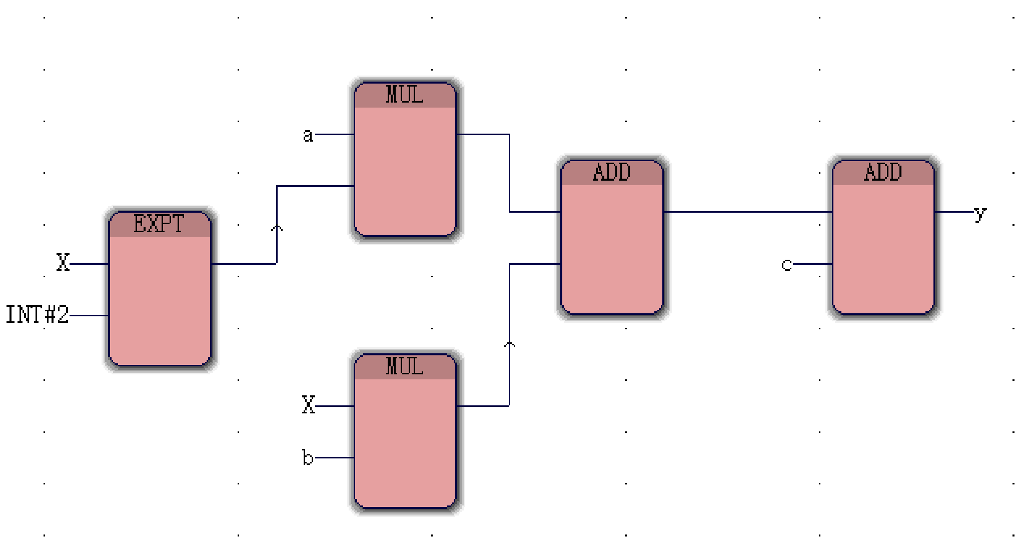
## 6 . 4 .1 Create an FBD program

1 ) creates a Ge new post-project, right- project tree " logic of the POU " , choose Insert →·program, the pop-up dialog box, type the program name FBD_Test , select the type and programming languages, as shownure




2 ) Create a tree project FBD_Test , in editing wizard were founDADD, MUL and EXPT function blocks, will they drag FBD_TesT the editing area, we use them to complete a one dollar function quadratic function ;

$y=ax^2+bx+c$ , the established FBD program is shown




3 ) The function block EXPT completes the square operation of x , the function block MUL completes the multiplication of a and $X^2$, the other MUL completes the multiplication of b and x , the function block ADD completes the addition of $ax^2$ and bx and c , click " produce " Then "bottom" , then click " debug switch " , add the variables a, b, c, x, y to the watch window, assign the variables a, b, c, x to 3.0, 4.0, 6.0,2.0. Respectively , the program automatically calculates the result of $ax^2$ + bx + c = y , as shown

| Variable | Value | Type | Instance |
|---|---|---|---|
| X | 2.0000000E+000 | LREAL | Configuration.Resource.Task.new003.X |
| a | 3.0000000E+000 | LREAL | Configuration.Resource.Task.new003.a |
| b | 4.0000000E+000 | LREAL | Configuration.Resource.Task.new003.b |
| c | 6.0000000E+000 | LREAL | Configuration.Resource.Task.new003.c |
| y | 2.6000000E+001 | LREAL | Configuration.Resource.Task.new003.y |

Watch 1 | Watch 2 | **Watch 3** | Watch 4 | Watch 5 | Watch 6 | Watch 7 | Watch 8

In FBD , there is no need to create additional variables in the variable worksheet, so after inserting the variables and double-clicking the variables, they appear in the variable worksheet. The FBD function block diagram programming language is similar to the LD ladder diagram programming language.

At this point, the completion of a FBD language programming .

## 6 .4.2 EN and ENO Description

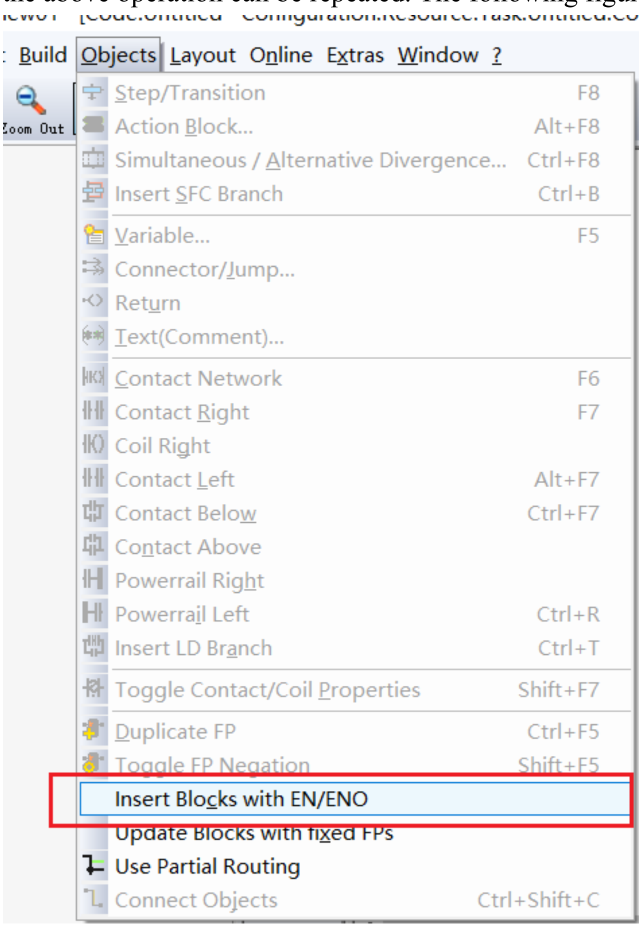
**1 : The difference between the logic instruction with EN and ENO and without EN and ENO ;**

( 1 ) When the user uses an instruction with EN and ENO , if the parameter value of the EN pin is FALSE ( 0 ) , the function defined by the instruction will not be executed, and the output value of the output pin of the instruction will not be refreshed. . Conversely, if the EN pin parameter value defined by the instruction is TRUE ( 1 ) , the function defined by the instruction will be executeDAnd the value of the instruction output pin will be refreshed.

( 2 ) ENO pin output and the EN input pins consistent, EN pin is TRUE , ENO piNSimultaneously becomes TRUE ; EN pin is FALSE , ENO piNSimultaneously becoming FALSE . When the instruction is a function block ( FB ), if the function block ( FB ) is executed, EN changes from TRUE to FALSE , the function block ( FB ) continues to execute, but the value of the output block of the function block ( FB ) does not Was refreshed.

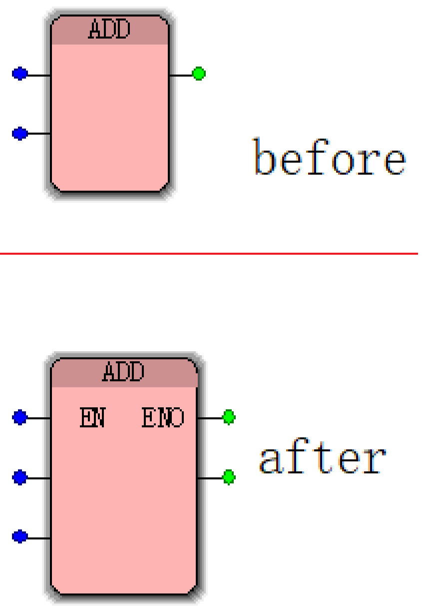
**2 : How to set logic instructions with EN and ENO without EN and ENO**

( 1 ) Put the mouse under the programming interface, the menu bar will automatically add the "Object" menu, click -> " Object " -> Select " Insert Block with EN/ENO " , and later from the "Edit Wizard" The function module that is dragged out will be self- EN/ENO ; when the function is not used, the above operation can be repeated. The following figure illustrates




( 2 ) The function block without EN/ENO has been called . If you want to increase the EN/ENO function, the mouse selects the programming interface. At this time, the menu bar will automatically add the “Object” menu, click -> “Object” -> select "insert with EN / ENO blocks", and theNSelect the need to add EN / ENO functional blocks, right -> select >

- update the FB / FU ( E ), this time with a function block are automatically added ENO EN / pin . As shown below



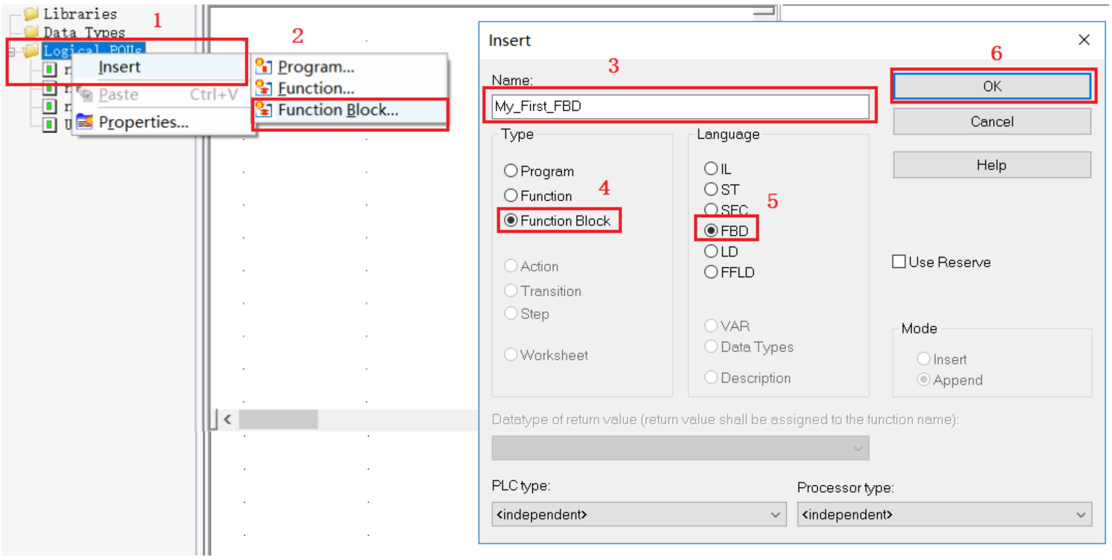

## 6 .4.3 Creating a User Library

User library function: Generally, the user library is built into a function block by encapsulating it into a function block. In other projects, the same specific function is also required. The user can directly call the packaged function block without rewriting it. Shorten the program development cycle and facilitate the modular management of the program.

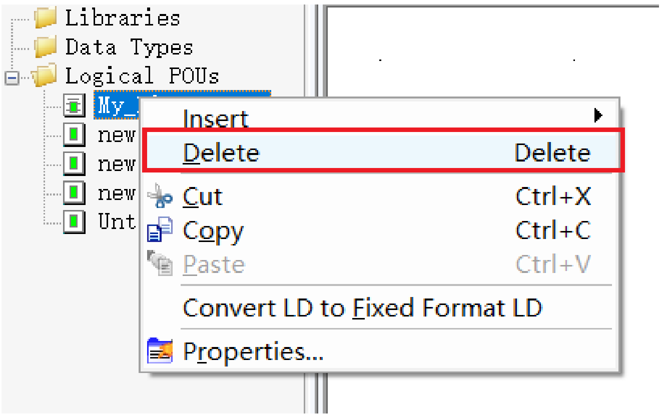
**Create user library and call steps**

1. Creating a Project User Reference (Chapter 3 3.3 Creating a Project ) will not be repeated here.

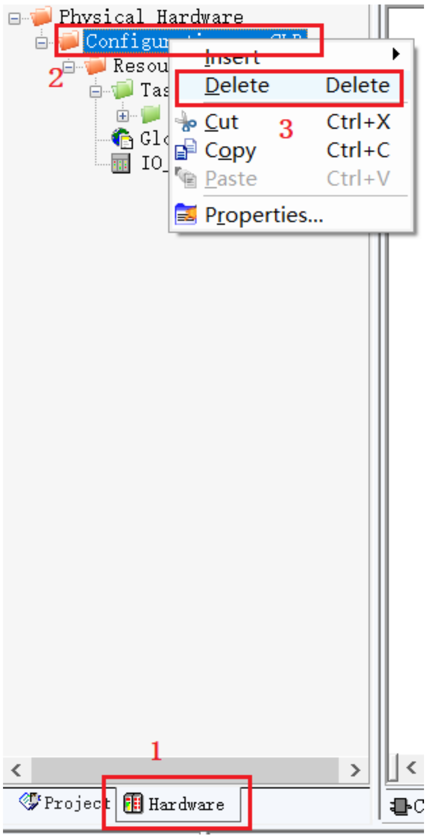
2. In the project tree select -> Logic POU-> Right -> insert -> Function Block ( E ), then pop up a "Insert" dialog box, name will be inserted into the block. (This case is named My_First_FBD ), select the programming language (this case is selecteDAs LD language) Click -> OK, a function block named “My_First_FBD” will be created under the logical PUO file , as shown.



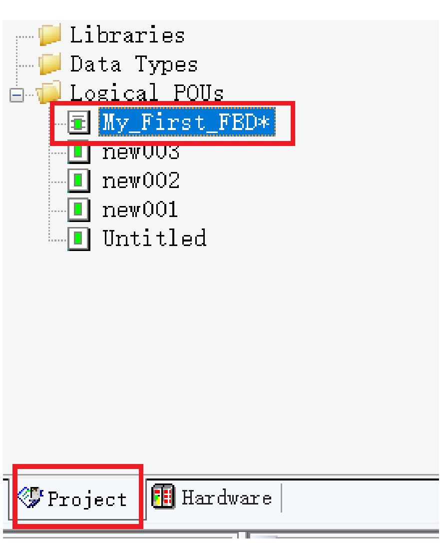

3. Delete the block under POU logic, only keep the function block named My_First_FBDAnd select the block to be deleted -> right click -> delete , as shown below

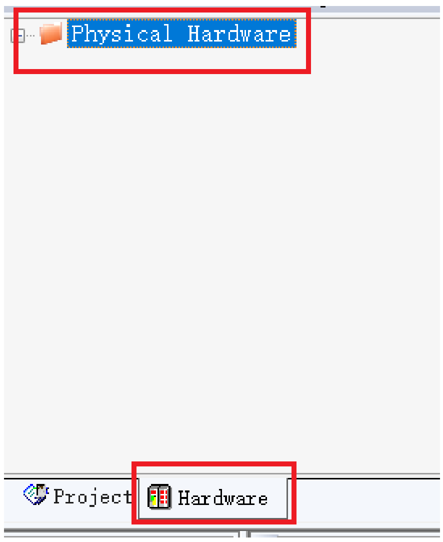



4. Delete the configuration information under hardware, click -> Hardware -> Select Configuration -> Right -> Check -> Delete to retain only the physical hardware as shown below




5. After the completion, the "Engineering" and "Hardware" under the project tree are as shown below.

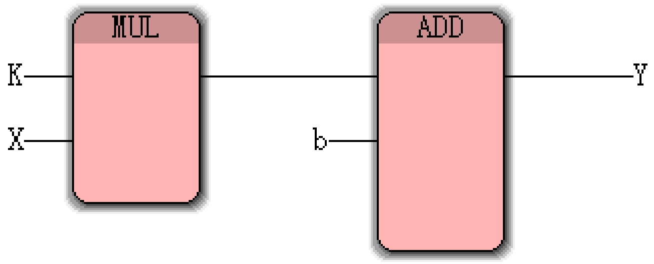


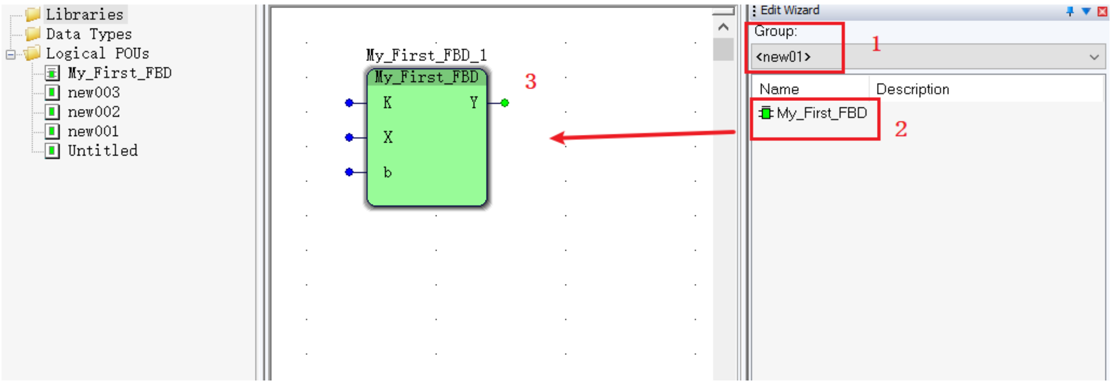

6 . After completing the above operation, start programming and packaging of the function block.

Example: Encapsulating a one-on-one function instruction;

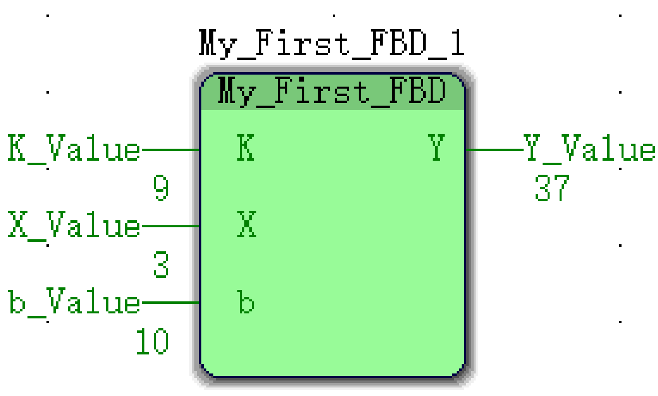
Call a multiply instruction, an addition instruction, and theNSet the four variable names to K, X, b, Y , and select the data type as INT (in this case, select the INT type) . Three variables in the K, X, b usage. Select the variable of type VAR_INPUT and Y to select VAR_OUTPUT . Click "Make" after completion without error warning. Click Save and close the project. At this point, programming and packaging are all completed. As shown below (special reminder: package function block variables use internal variables as much as possible to improve the use of the package block)




7. Find the user library in the group and find the function block of the above package, as shown in the figure.




8: Under the main project of the project, drag the My_First_FBD function block to the main program and set the variable name for it. The data type is INT type and the usage is internal variable (VAR). Click "Make" after compiling without error message, click "Download", then click "Cold Start", select to open "Debug Switch", online assignment variable K_Value =9, X_Value =3, b_Value = 10 final output result Y_Value =37. As shown in the following figure (from the color can distinguish the user library and firmware library, pink for the firmware library, blue for the user library, green for the functional blocks packaged under this project)



At this point a user library package and call has been completed.

**Special Note:**

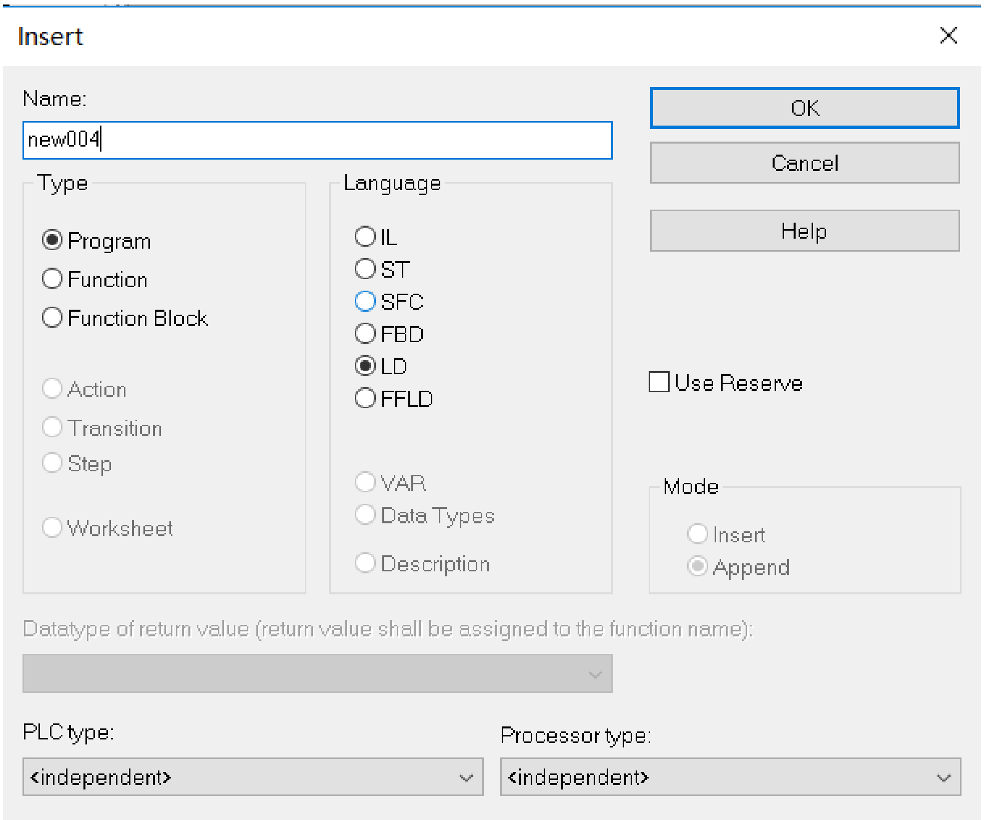
The called user library "double-click" function block can view the internal programming content, but does not allow modification or adding new functions.

# 6 .5 the LD Ladder Logic programming language

The ladder programming language is one of the oldest programming languages. The ladder diagram is derived from the logic control diagram of the electrical system. The logic diagram uses relays, contacts, coils and logic diagrams to represent the logical relationship between them. Ladder programming language graphical elements may be employed there ladder chart network, power rail, connecting wires, contacts, coils , function and other function blocks, the data type can be BOOL, BYTE, WORD , and DWORD .

## 6 .5.1 Create a LD Program

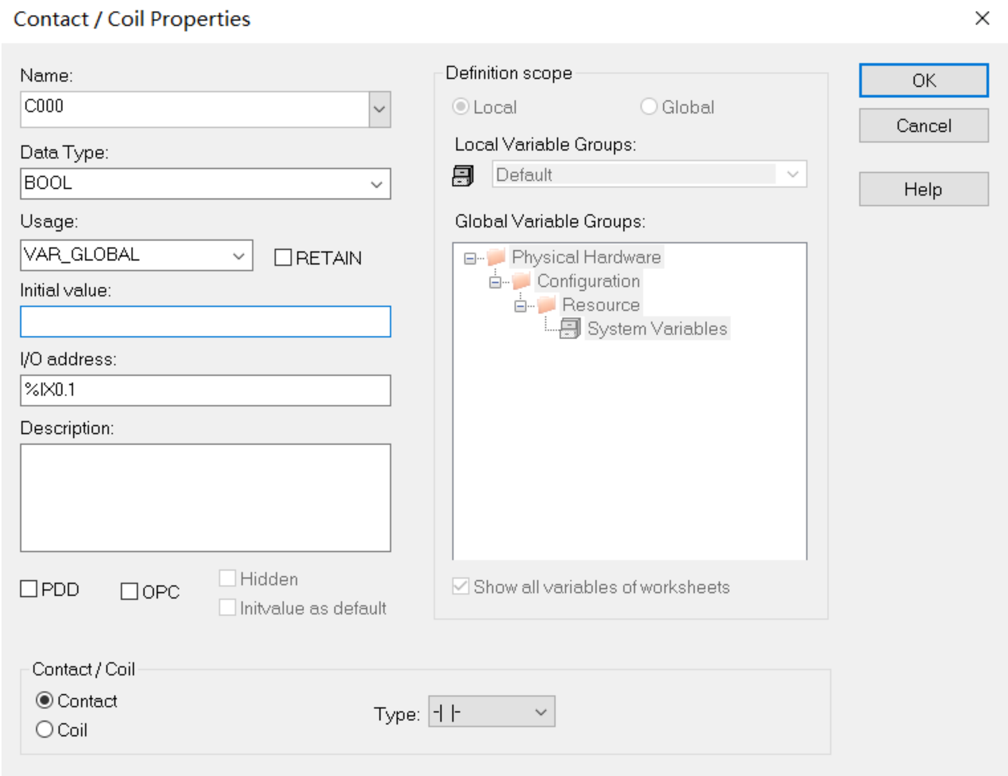
1> After creating a project, right- project tree " logic of the POU ", choose Insert →·program, the pop-up dialog box, type the program name LD_Test , select the type and programming languages, as Figure 8-25 shown



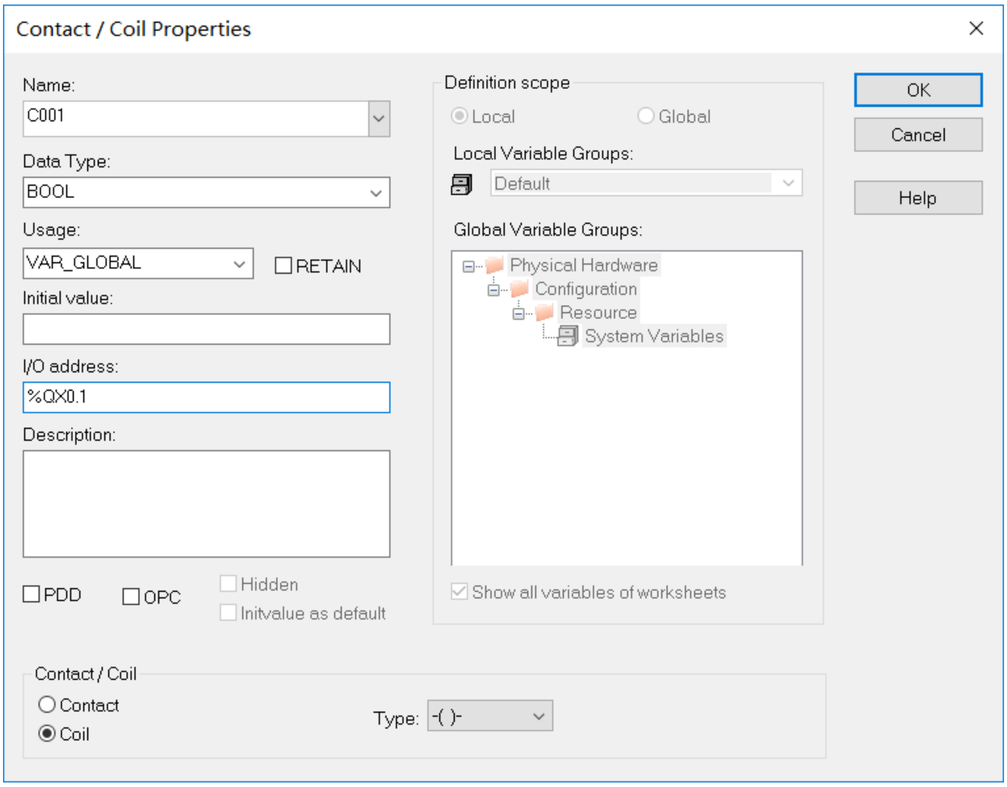

2> Click the editing area of the MULTIPROG programming software, then click " " on the toolbar on the left side of the editing area .

3> A simple network with a ladder diagram appears in the editing area. On the left is the left power rail 001 anDA normally open contact C000 . On the right is a coil C001 and the right power rail as shown.


4> Double-click the normally open contacts C000 , contact may / Coil Properties dialog box, the I / O to the address field, enter % IX0.0 , represents PLC of the machine ADigital input channel, click " OK " , shown in FIG.




5> double -click ladder coil C001 , contact may / Coil Properties dialog box, the I / O address field, enter % QX0.0 , represents PLC a first digital output channels of the machine, click " OK " as in FIG. FIG .

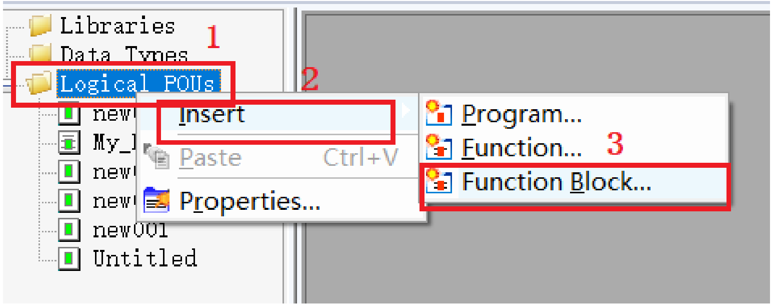



6> In LD , there is no need to create additional variables in the variable worksheet. Therefore, after inserting a ladder network and double-clicking the contacts and coils respectively, these two variables appear in the variable worksheet . You can insert a variable by right- clicking in the editing area and selecting " Variable ( V )" . This inserted variable must be connected to the function block pin. In the above figure, the variable working range ( usage ) of C000 and C001 is set to VAR GLOBAL , indicating that these two variables are global variables and can be used in other programs in this project.

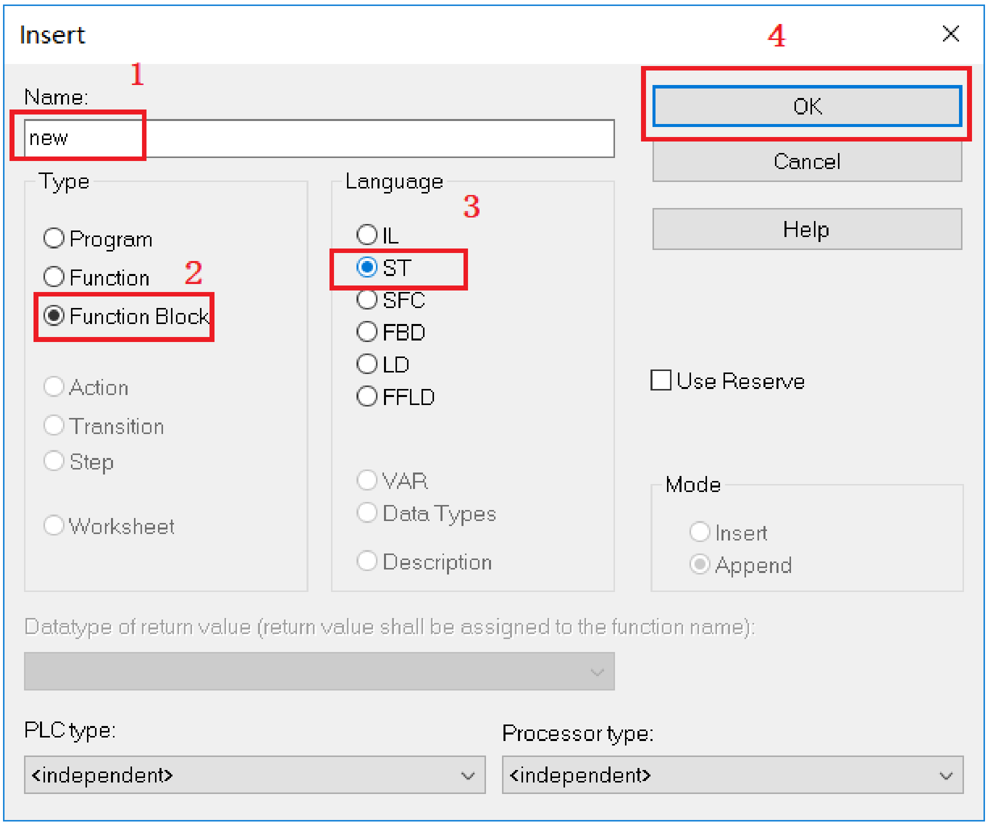
At this point, a complete LD program is completed. When the contact connected to the first digital input channel % I X 0.0 is closed, the coil connected to the first digital output channel % Q X 0.0 is turned on.

## 6 .5.2 in LD created in a FB

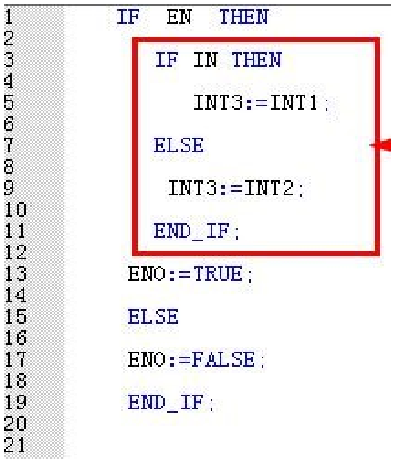
Use LD programming, sometimes neeDA special function block, and editing wizard is not integrated , in which case the user may LD create the users own function blocks, the following explains how LD create one of the MOVE block. Right-click on the created LD program name in the project tree and select Insert →·function block as shown



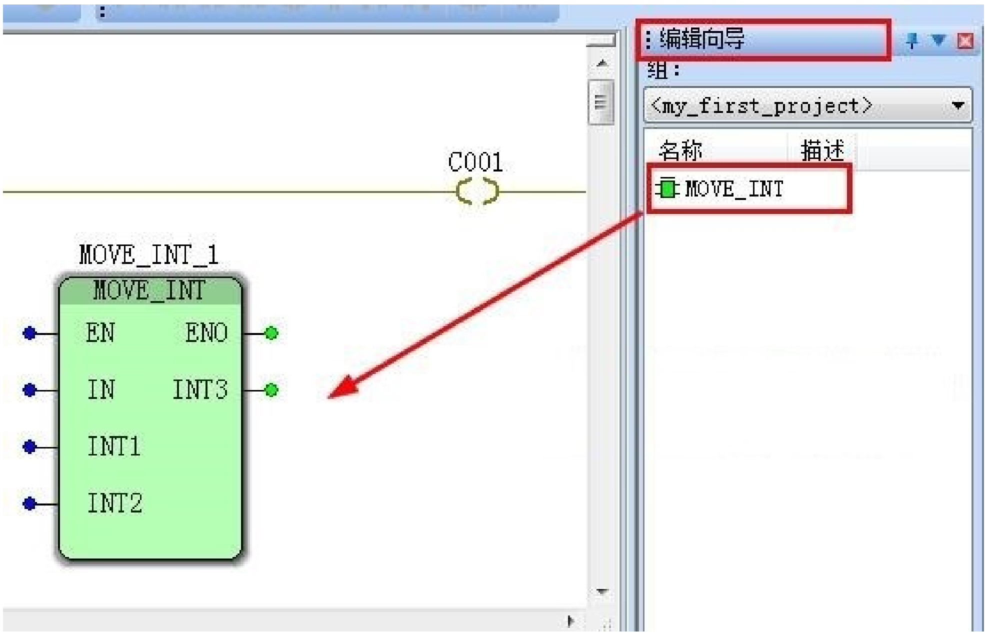

Pop up ADialog box, you want to create in this dialog box FB name, and used to develop the type of programming language, here are the MOVE _INT , function blocks and ST , click OK, then enter the ST language programming of Figure Shown .



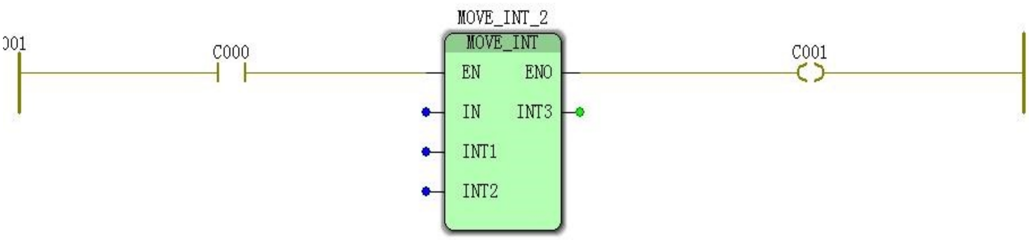

The MOVE function block created by has input pins EN, IN, INT1, INT2 with output pins ENO, and INT3 . This function block is used in the LD prograMCreated earlier . Using ST create the same program, to establish a first variable worksheet MOVE_INT variables used in the function block comprises a variable name, data type, pin functions return after completing ST programming interface for programming as follows as FIG illustrated




When finished, click " production " compiled by back LD editing area, in the editor wizard creates MOVE_INT function block drag LD edit area as in FIG illustrated




➢Connection of the following figure made by this MOVE_INT function block is as shown




🕮·Example: Controlling the start and stop of ADevice, it has two working modes, the contact " C000 " controls the start stop, and the contact " C002 " controls the operation mode (such as different speed operation).

Equipment :

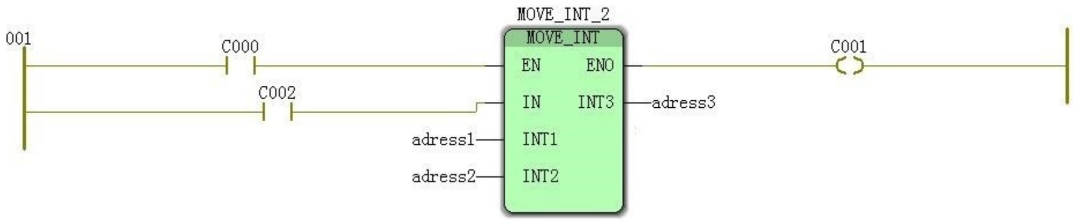
C000 is started for " TRUE " device and " FALSE " is stopped

C002 is " FALSE " operating mode selection " adress2 "

C002 is " TRUE " operating mode selection " adress1 "

ADress3 is used to monitor the current operating mode;

The final ladder diagram is shown



# 6 .6 SFC Sequential Function Chart Programming Language

The SFC Sequential Function Chart programming language divides a complex control program into several small tasks, each of which is executed sequentially. In the SFC sequential function chart programming language, each small task is calleDA " step " . The relationship between " step " and " step " is called " conversion " . Each " step " carries an action, " step " . , " Conversion " and " Action " are connected by " wiring " . A " step " can be associated with multiple actions. An action consists of an action body anDAn action qualifier that describes how the action is associated with the step. When the SFC step becomes active, the associateDAction is executeDAccording to the action qualifier. Actions can be either a Boolean variable or an IL, ST , LD, FBD program ( called ' details' ) . The conversion represents the case where processing continues through the next step. If a transition becomes TRUE , the previous step is executeDAgain and the next step becomes active. The conversion can be either a Boolean variable or ADirectly connected Boolean expression written in FBD or LD . You can also edit the code to be executed in another prograMCalled Detail.

The collection of connected objects is calleDANSFC network. ANSFC network must have an initial step, which is the first step to be executed when the SFC POU is called . Parallel branches ( executing synchronously ) or selecting branches can be inserted within the SFC network .

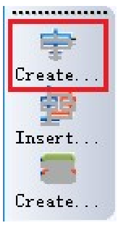
## 6 .6.1 Creating a SFC program

Below we create a control program for traffic lights.

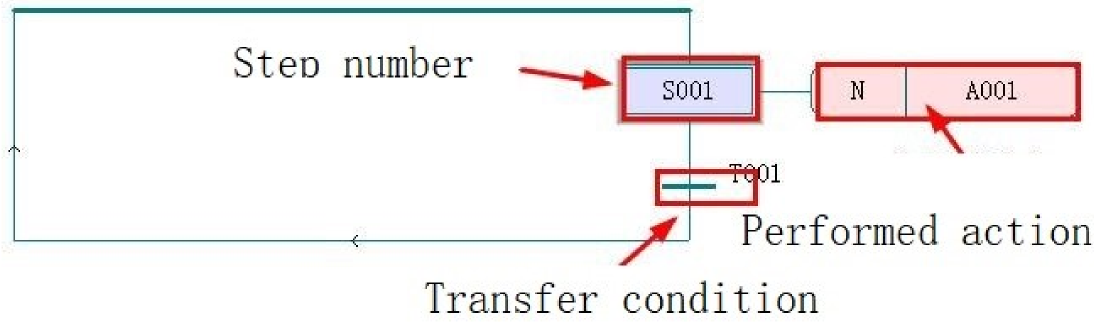
## 6 .6.2 Creating SFC network

( 1 ) Create a new project. During the creation process, the program language is SFC and the program name is TrafficLight .

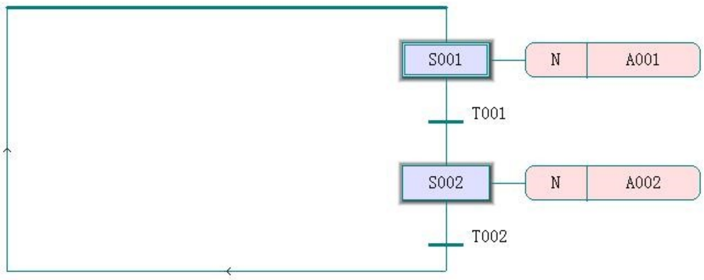
Entering TrafficLight edit area, and then click on the left side of the editing area , " to create a step switch sequence " as shown.



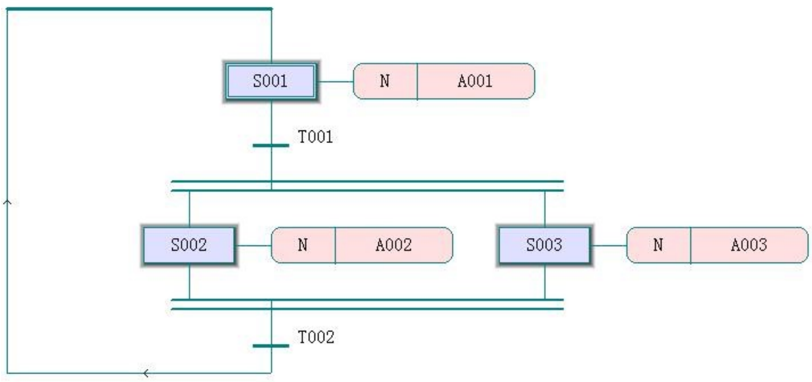

( 2 )appears in the editing area a SFC of " step " S 001 , "action" A001, "conversion" T001 as shown.



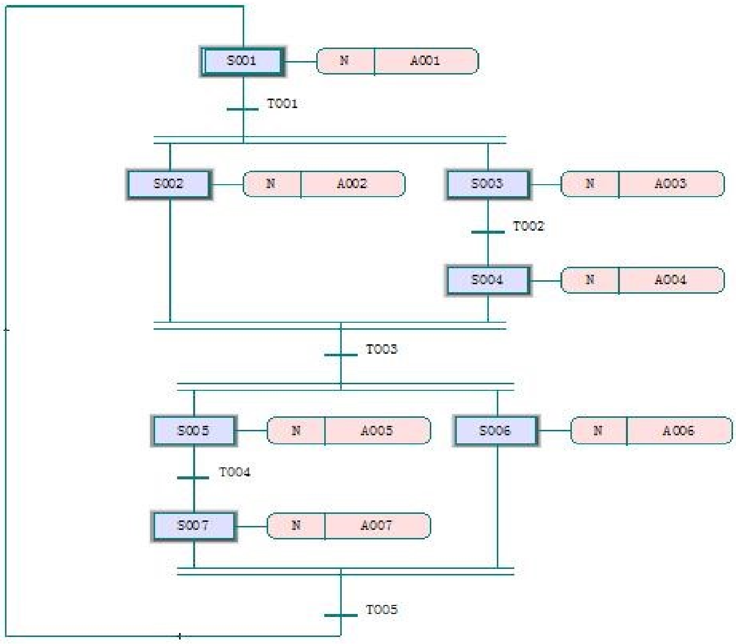

( 3 ) In the above figure, each " step " , " conversion " and " action " have unique names. Then, click " convert " T00 1 in the above figure , and then click " create step conversioNSequence " . in the following step adds a " step " S00 2 , as shown.




( 4 ) Click " Step " S002 , then click " Insert SFC Branch " , then insert a branch on the right side of " Step " S002 , the branch has a " Step " S002 , as shown




✧▪Repeat the operation described on the final traffic light SFC function diagram as shown



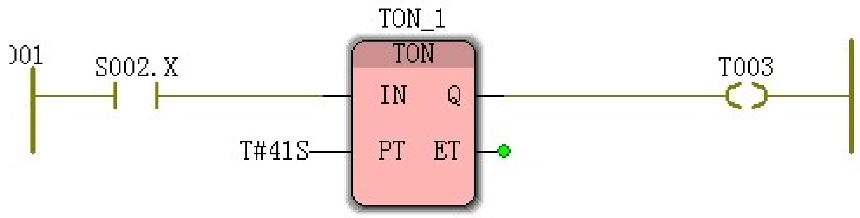

At this point, aNSFC network is created, in which S001 is the initial step , the user can use the initialization data, such as the counter to clear the initial value and other operations, and then according to the requirements of the "step", "conversion" to write code and The setting of the property.

1 ) Conversion condition : Double-click " Convert " T 001 and select LD programming language in the pop-up dialog box.

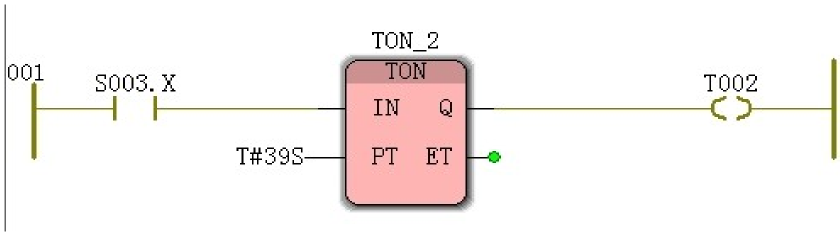

2 ) Click OK to enter the programming of T001 , insert a network in the editing area, and double-click the variable name of C001 . The variable name of the coil is changed to T001 , as shown below.


This conversion condition is to start the SFC program because " step " S001 is the starting step and is always 1 . When the global variable C000 is set to 1 , "convert" T0 01 is 1 , and "step" S002 and S003 are activated .

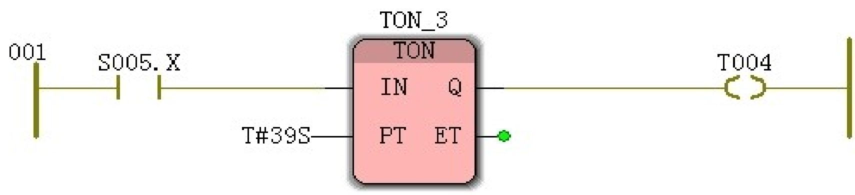
3 ) Double-click " Convert " T00 3 to select the LD programming language. This conversion condition is used for timing. When " Step " S00 2 is activated for 41 seconds, " Convert " T00 3 is set to 1 to activate "Step". S005 and S006 are inserted into the following program.



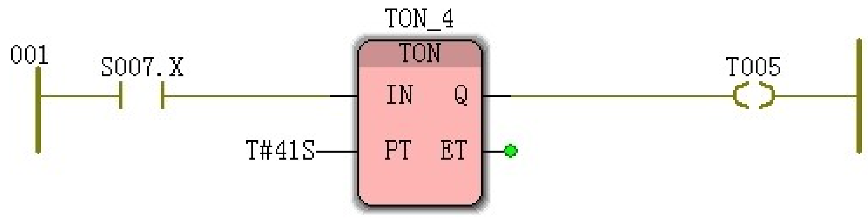

4 ) Double-click " Convert " T002 . When the LD programming language is selected , this conversion condition is used for timing. When " Step " S00 3 is activated, it starts timing. When the timing reaches 39 seconds, " Convert " T00 2 is set to 1 , and "Step" is activated. S004 insert the following program




5 ) Double-click " Convert " T004 , select the LD programming language, insert the following program , this conversion condition is used for timing. When " Step " S00 5 is activated for 3 9 seconds, " Convert " T0 04 is set to 1 , and enter " Step " S00 7




6 ) Double-click "Convert" T00 5 to select the LD programming language and insert the following program . This conversion condition is used for timing. When " Step " S00 6 is activated for 41 seconds, " Convert " T0 05 is set to 1 , and then jump to " Step " . " S001 , so repeated.




At this point, the conversion conditions of the SFC are programmed.

action

The " action " A001 name to initialize;

The " action " A 002 name was changed to the north and south _ red light ;

The " action " A 003 name to something _ a green light ;

The " action " A 004 name to something _ yellow ;

The " action " A 00 5 name to something _ a red light ;

The " action " A 00 6 name was changed to the north and south _ green light ;

The " action " A 00 7 name was changed to the north and south _ yellow ;

1 ) Double-click "Step" S001. In the " SFC Step" dialog box that pops up, you can name "Step" and type selection. Here, select "Initial Step" .

2 ) Double-click "Action" A001 in the "Action Properties" pop-up, select "Details", initialize the A001 name .

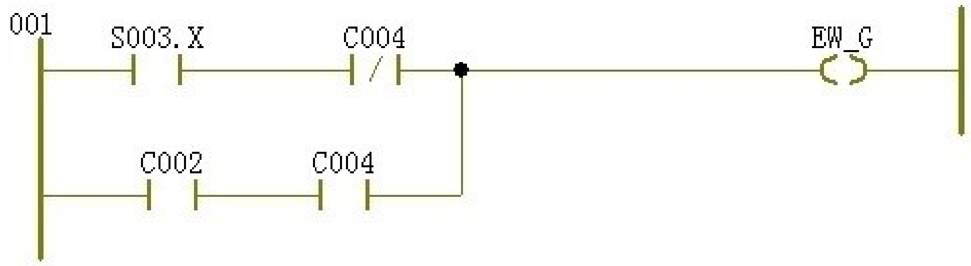
3 ) After the completion, the "action" name at this time is "initialization". The color changes from pink to green. Double-click "initialize" again. In the pop-up "insert" box, select the programming language. Select the LD language here , click "OK". ", a blank LD programming interface pops up. The user can write the initial "action" code here; select blank here.

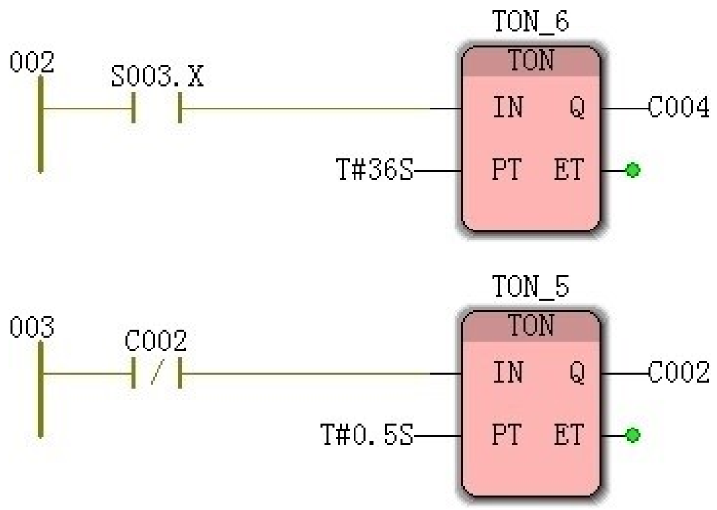
4 ) Repeat steps 2~3 to modify the name of each action of A001~A007 ;

5 ) Double-click " North-South _ Red Light " to enter the programming of 'North-North _ Red Light' action , insert the following program in the editing area, where the I/O address of the coil SN_R is : %Q X 0.0 , indicating " step " S002 is activated when north and south The direction is red.


6 ) Double-click the " stuff _ green ' enter' things _ green ' operation programming, the

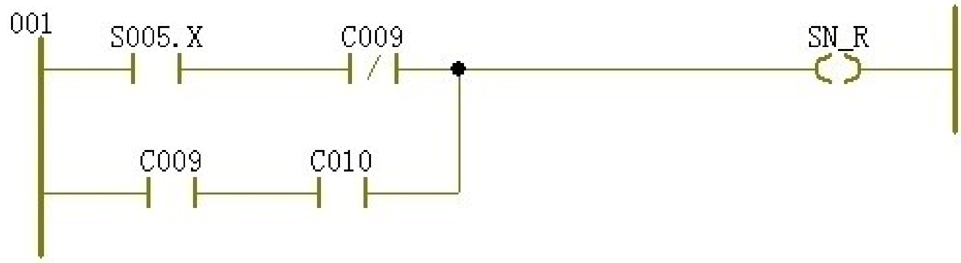
coil insertion procedure in the editing area EW_G the I / O address : % Q X- 0.1 , represents a " step " S003 when activated green light east-west direction.

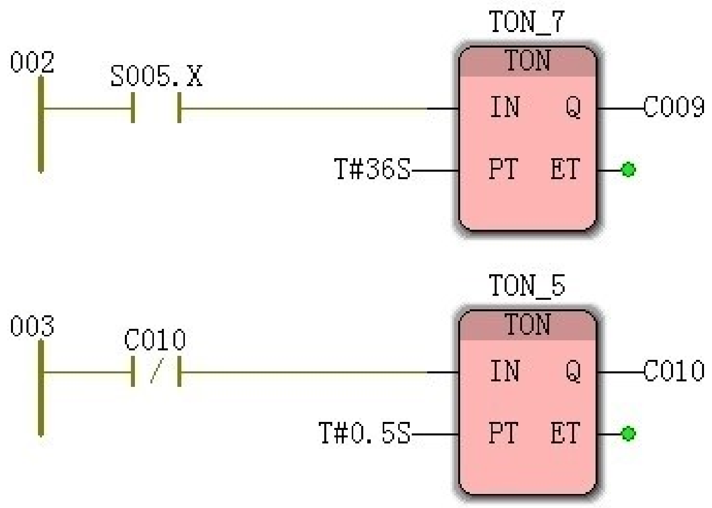



7 ) Double-click the " stuff _ yellow " enter 'things _ yellow' operation programming, the insertion procedure in the editing area, the coil EW_Y the I / O address : % Q X- 0.2 , represents a " step " S004 activation things Directional Brightness Yellow light.


8 ) Double-click " North-South _ Green Light " to enter the programming of " North-South _ Green Light " action . Insert the program in the editing area . The I/O address of the coil SN_G is : %Q X 0. 3 , which means " Step " S005 is activated when the north-south direction is green. Light up.



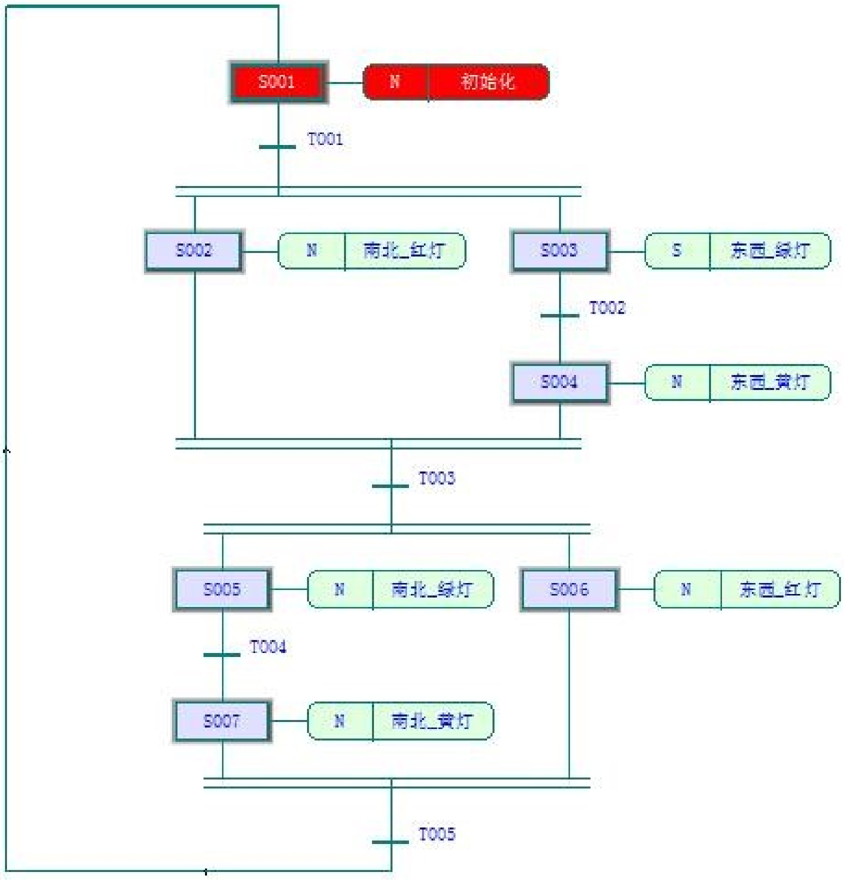




9 ) Double-click the " north-south _ yellow " enter "north-south _ yellow ' operation programming, the insertion procedure in the editing area, the coil SN_Y the I / O address : % Q X- of 0. The . 4 , represents a " step " S00 . 7 activated The north and south lights are yellow.


10 ) Double-click the " stuff _ red " enter 'things _ red' action programmed, the following procedure is inserted in the edit area wherein the coil EW_R the I / O address : % Q X- 0.5 , represents a " step " S00 . 6 is activated The east and west lights are red.


At this point, a complete traffic light control program is completed, click " Make ", after downloading to the PLC without error prompt , click the "Cold Start" program to start working. This program is the automatic control of the traffic lights at the simulated traffic intersection, that is, when the red light is bright in the north-south direction, it is kept for 4 1 second; the east-west direction is green, it is kept for 3 6 seconds, then it flashes for 3 seconds, then the yellow light is on for 2 seconds. The entire period is 41 seconds;

The final simulation effect is shown in the figure.




## 6 .6.3 SFC action qualifier

The SFC action includes an action qualifier anDAn action body. The action qualifier describes how the action is associated with the step. The following action qualifiers are available.

| Qualifier | description | Features |
|---|---|---|
| N | do not save | As long as the step is active, the action code body is executed or the Boolean variable is set. |
| R | Beyond reset | The action code body is no longer executed or the Boolean variable is reset. The action must be set before using the ' S ' qualifier. |

| | | |
|---|---|---|
| S | Set (save) | Execute the action code ontology or set the Boolean variable. This ( set ) state is saved once the ( associated ) step becomes active . This set state can only be explicitly reset by associating the action to ADifferent step by using the ' R ' qualifier. |
| L | Time-limited | As long as the step is active, the action code body is executed or the Boolean variable is set, but the duration can be kept for at most a period of time. |
| D | Delayed | After the set delay time elapses, the action code body is executed or the Boolean variable is set. As long as the step is active, the action remains active. If the time when the step is active is shorter than the set delay time, the action does not become active. |
| P | pulse | WheNStep one becomes active, the action code body executes an operation cycle, or a Boolean variable sets an operation cycle. |
| SD | Save and delay | After the step is activated, when the set delay time elapses, the action code body is executed or the Boolean variable is saveDAnd set, even if the step becomes inactive again. This action will remain active until it is reset. If the time when the step is active is shorter than the set delay time, the action will become active anyway. |
| DS | Delay and save | After the step is activated, when the set delay time elapses, the action code body is executed or saved. |
| SL | Save and time limit | As long as the step is active, execute the action code ontology or set and save the Boolean variable in a fixed time interval. If the time when the step is active is shorter than the question at that time, the action will be active in the interval at any time. If the action is reset at this time, the action will immediately become inactive. |

# Ⅶ Works to Create and Configure

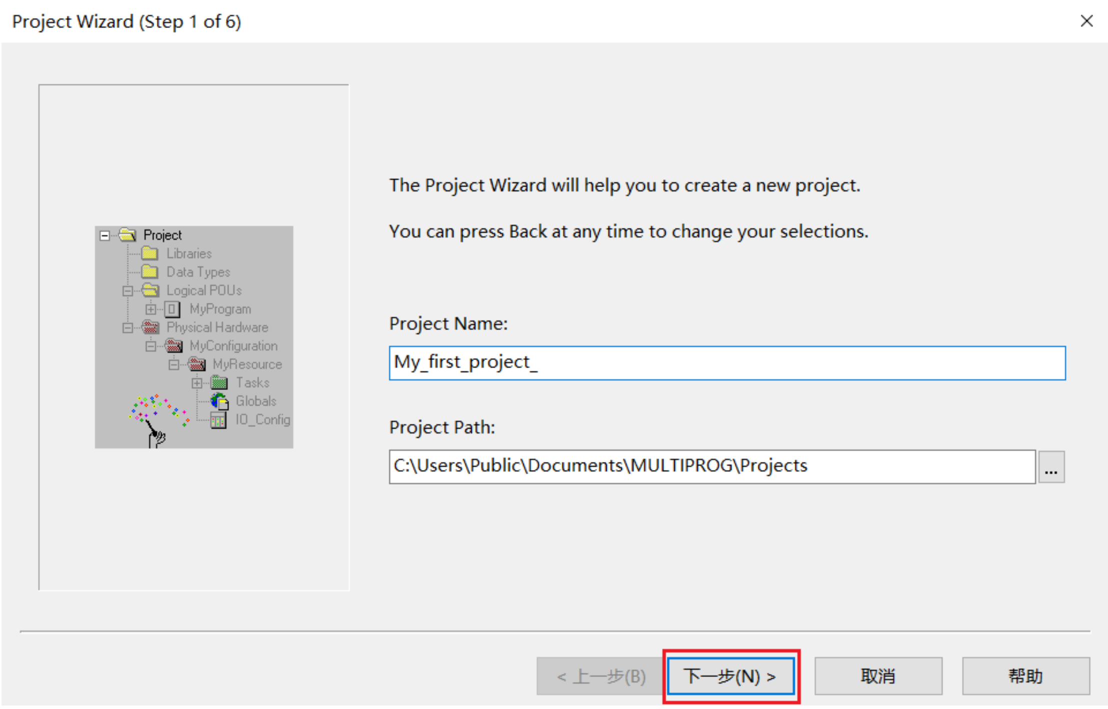
## 7 .1 Creating projects

1 ) In this section we provide step-by-step instructions for developing, editing, and running a ladder ( LD ) sample program using MULTIPROG software . The development of the program is divided into several stages.

Make use project wizard will guide you through creating new projects, where the user must define the name and path, programming languages, as well as the use of project PLC type.

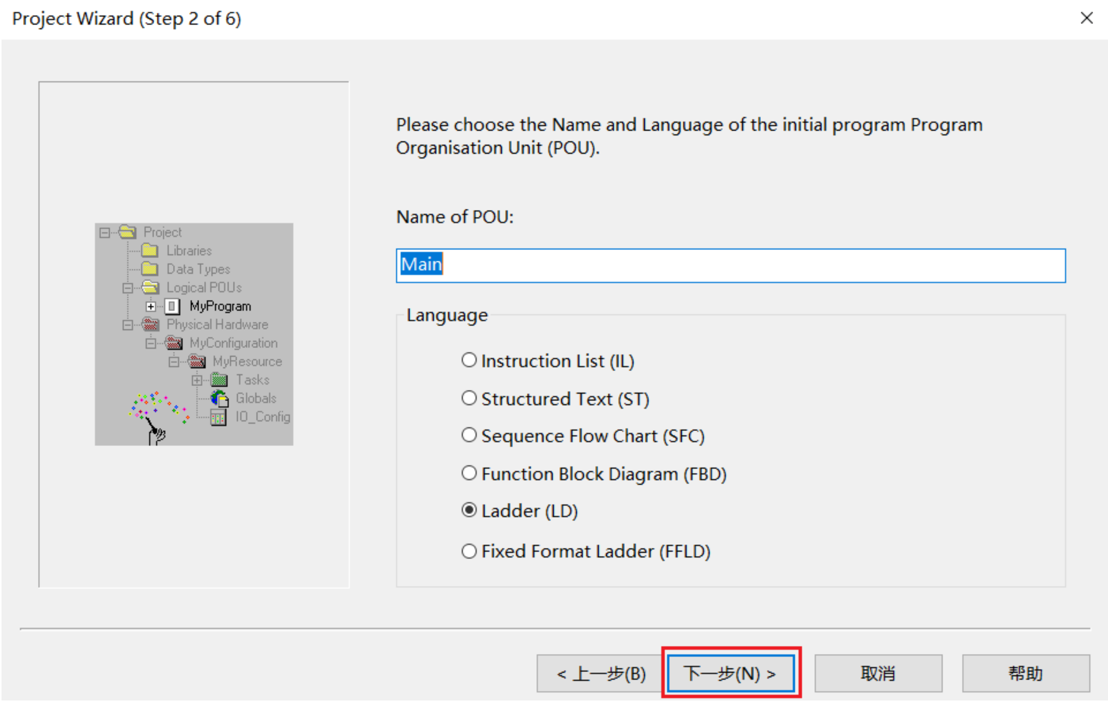
( 1 ) Click " File " " New Project " ;

( 2 ) Enter " My_first_Project " in the "Project Name" box of the wizard window , as shownure 6-1 ; according to the project naming rules, the project name and path must not containNSpaces or special characters, "Project Path" input The box indicates the path saved by the project. In the initial state, the default path is set by the user. After completion, click the “下一步” button.

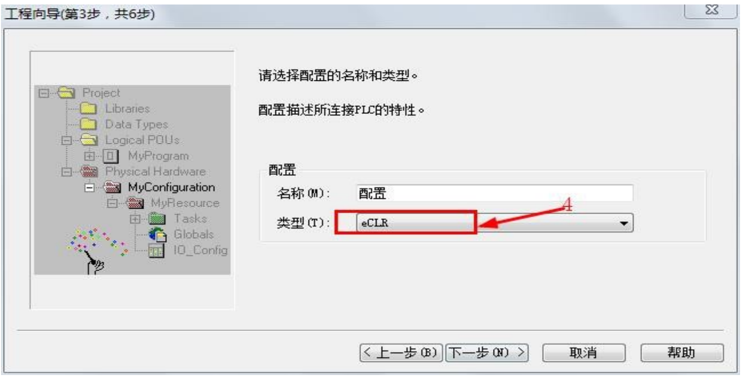



*** Note** : Special characters cannot be included in the project name and path, otherwise the project cannot be created successfully.

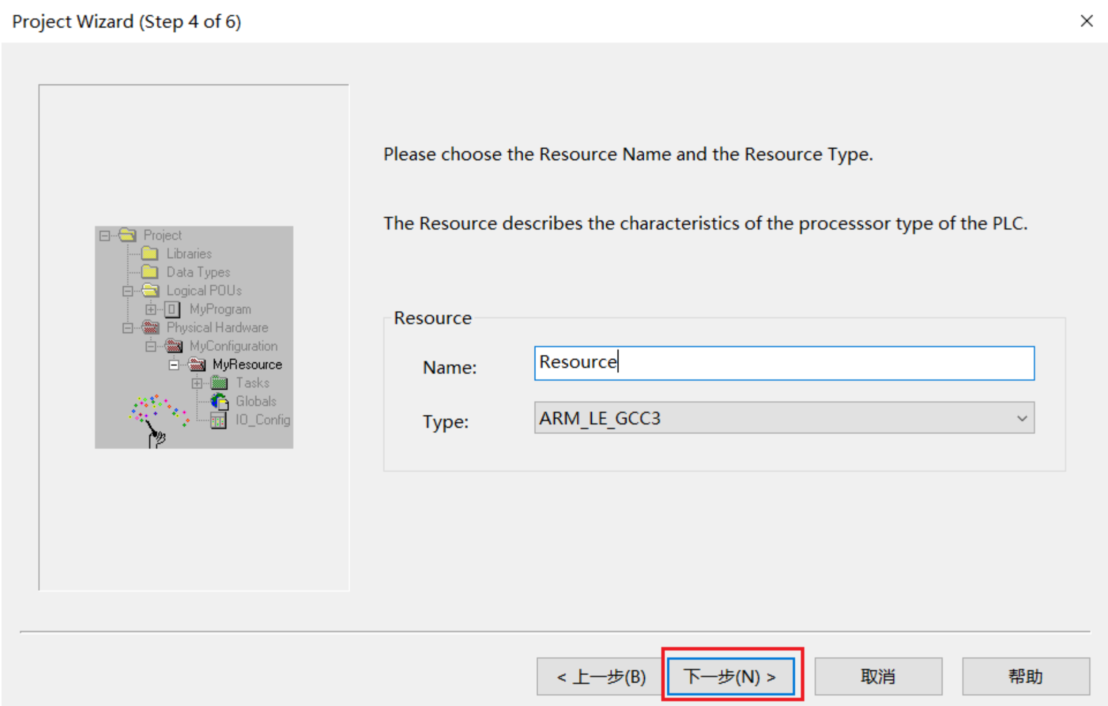
( 3 ) The second step of the project wizard dialog box is shown in the figure. Name the first POU " Main ", select "Ladder ( LD )" for the programming language, and click "Next"




( 4 ) The third step of the project wizard is to determine the name and type of the configuration. The dialog box is shown in the figure. Fill in the name of the configuration in the "Name" input box, here keep the default "Configuration". In the Type list box, select the PLC type as eCLR and click Next. (Note! The software selects " eCLR " by default , so when you create a new project, it automatically jumps to the fourth step of the next project wizard)

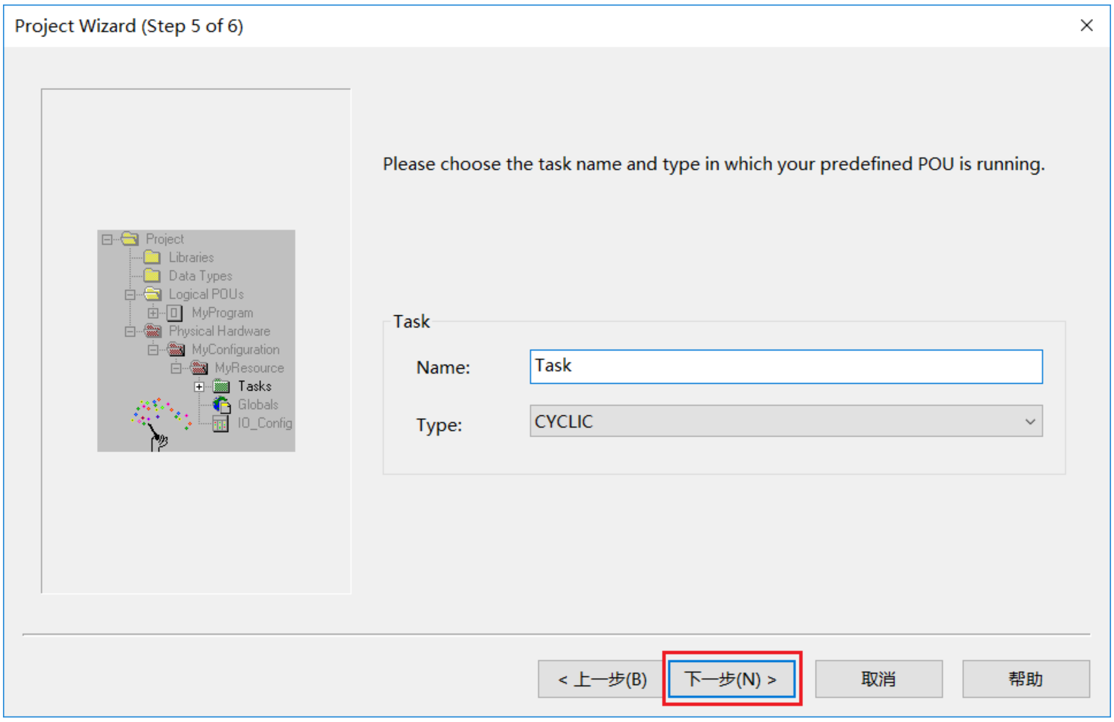



( 5 ) The fourth step of the project wizard is to select the “ resources ” to be used . The dialog box is as shown in the figure. "Name" to keep the default "resource ( Resource ) ", list box, select the "Type" " ARM_LE_GCC3 " (real PLC ), (if it is using MULTIPROG owNSimulation PLC simulation , select " eCLR_Simulation ", click " Next step" continues. )

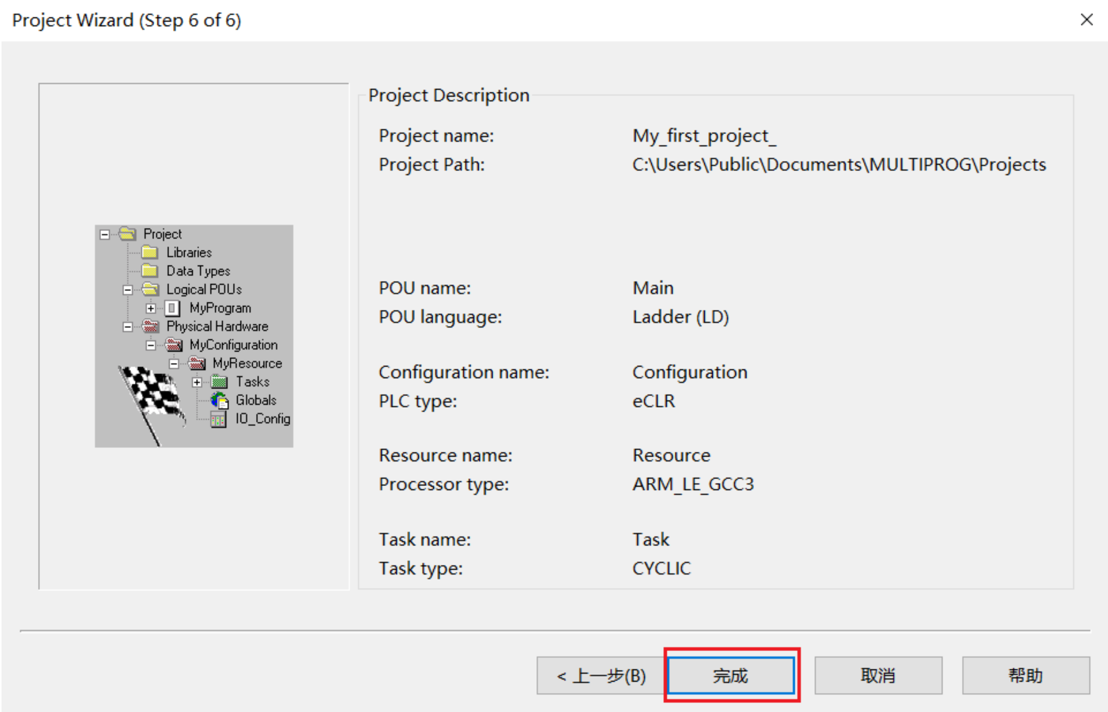



**Note:** Different type selections represent different hardware platforms, because at the time of engineering, the system generates hardware-specific machine code based on the type of resource.

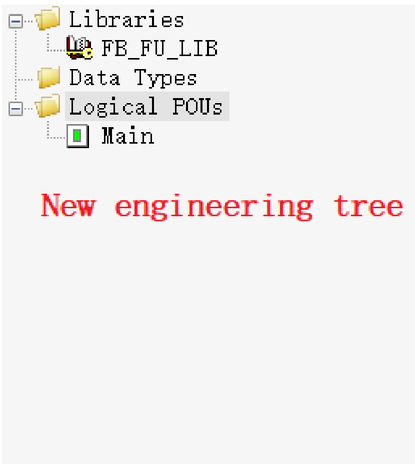
( 6 ) The fifth step of the wizard is to specify the name and type of the task. The dialog box is as shown. Here still keep the default name "task ( Task ) ", type selection for the " CYCLIC ", click "Next."



( 7 ) The last step, in the dialog box popped up by the wizard , summarizes the previous settings, project name, project path, POU name, PLC type configuration, processor type, task type , as shown.

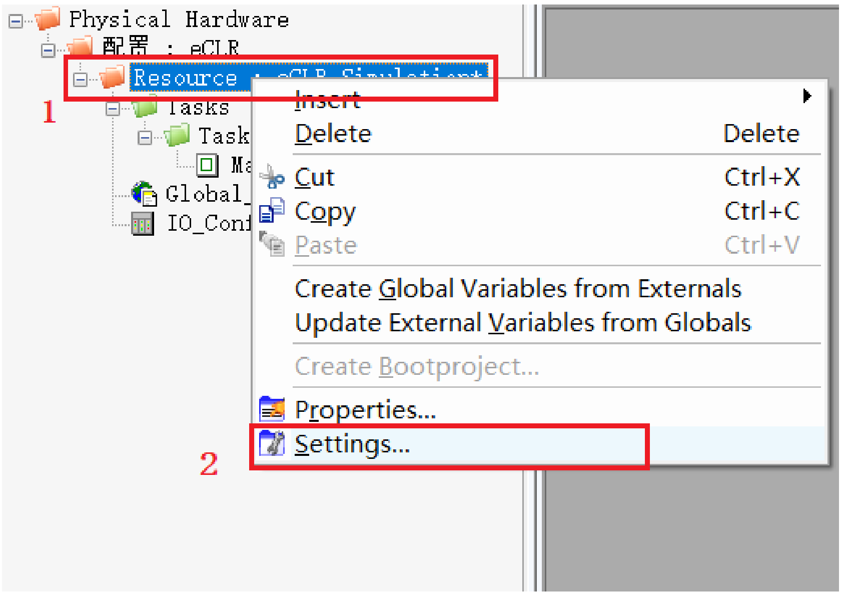



( 8 ) If no error is prompted , click "Finish", you caNSee the new generation of the project tree in the project tree window. as the picture shows. The "Logical POU " node is part of the algorithm implementation, and the "physical hardware" is associated with the actual controller type and settings.

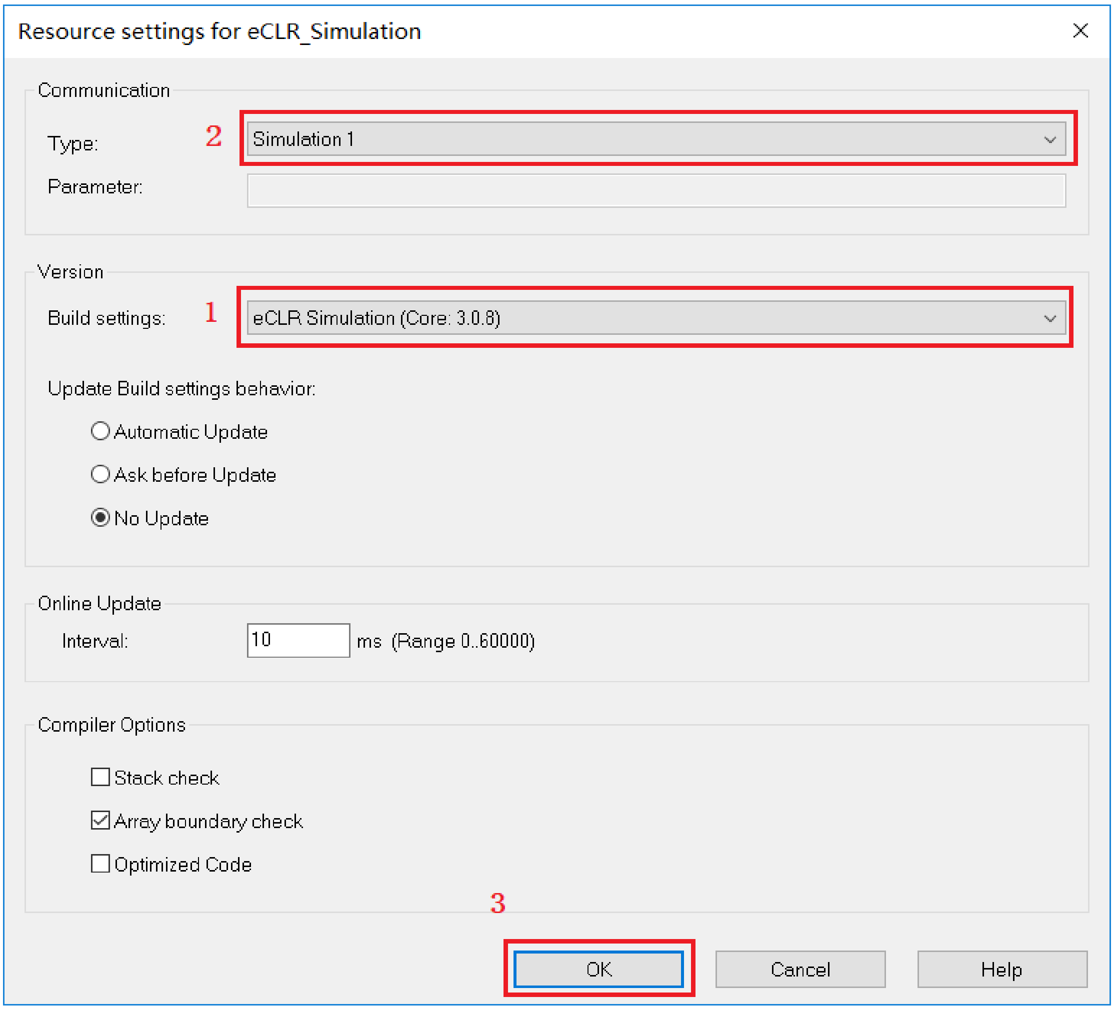


## 7 .2 Simulation Communication Parameters

1 ) After the installation is complete, open the already completed projects, simulate communicatioNSettings (provided that the "resources" ( Resource ) Select " eCLR_Simulation " ) , under the "Project Tree Window", right-click the "resources" to select "Set " shown in FIG.

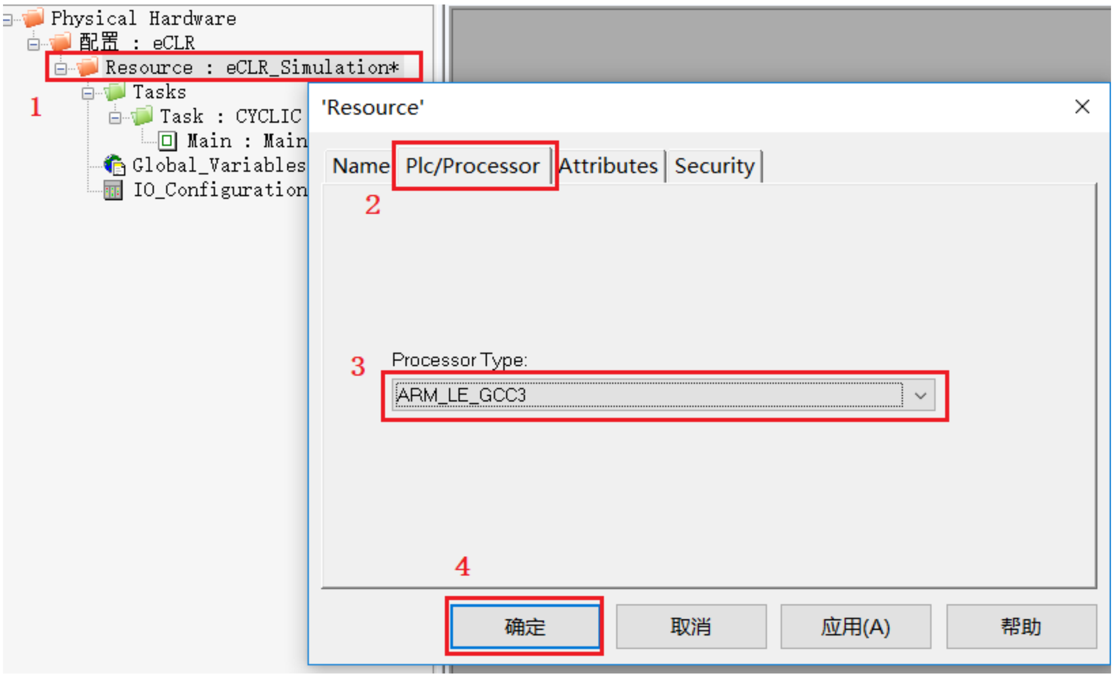



2 ) In the pop-up “ eCLR_Simulation Resource Settings” window, select “Simulation 1 ” or “ Simulation 2 ” under “Type ” ; under “Create Settings” , select the emulation processor version model, here select “ eCLR_3.0.2 "; click" OK " , again re-click" Create "button , no error program will prompt to download the simulation of PLC in.




## 7 .3 Physical Communication Parameters

1 ) After the motion controller is connected to the power-on communication line, set the communication parameters of the PC and the controller . Under the “ Project Tree Window” , right-click “Resources” and select “Properties” as shown in the figure.

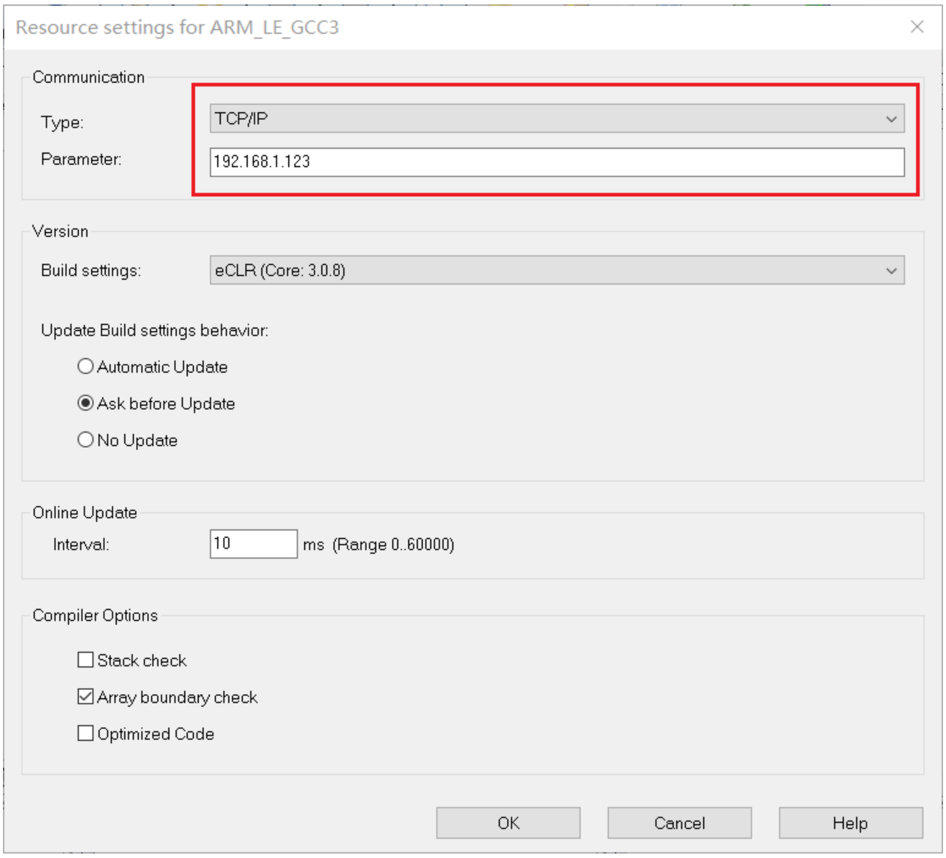



(Note ! "In the newly established Project Wizard PLC processor type" is selected, processor type, select the "simulation eCLR_Simulation "; physical PLC processor type selection " ARM_LE_GCC3 "; for this kind of simulation engineering projects and engineering Switch between projects)

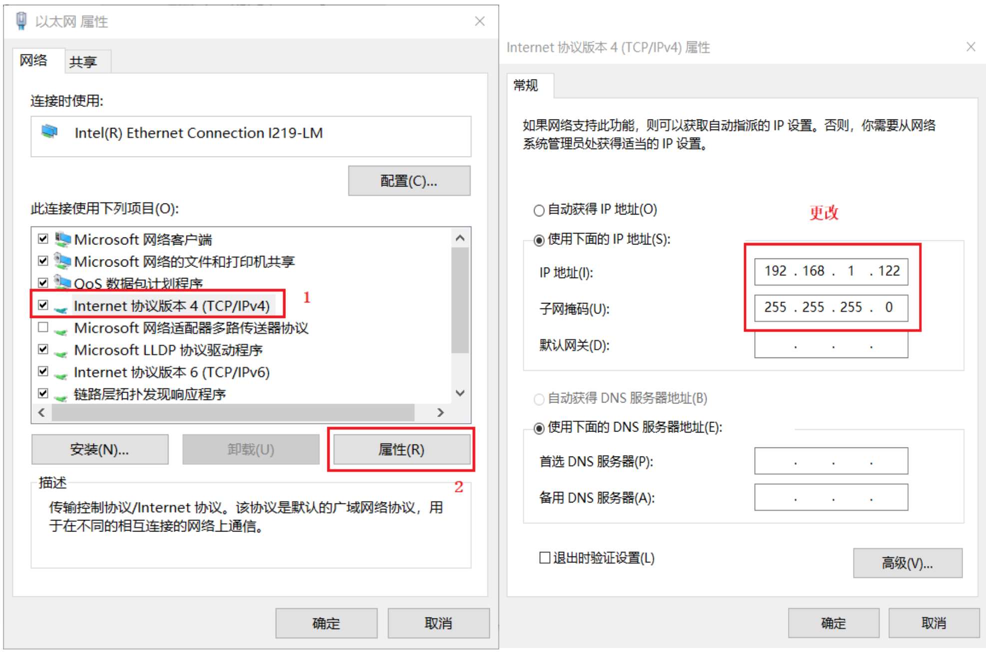
2 ) Right-click "Resources" under "Project Tree Window" and select "Settings" in the pop-up " ARM_LE_GCC3 Resource Settings" window to set:

Select " TCP /IP " under the communication "Type ";

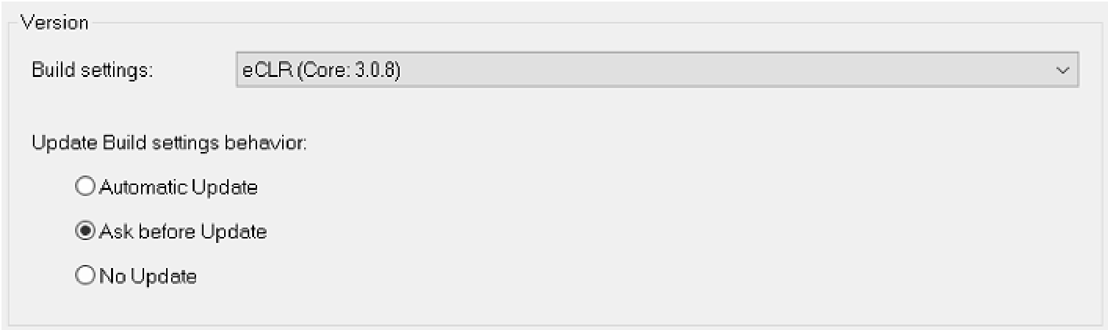
Set the IP address under the "Parameter" of the communication to " 1 92 . 168 . 1 . 123 - p41100 "




Confirm the port of the PC , and set the network attribute of the P C port. The IP address is: “ 1 92 . 168 . 1 . 122 ” , as shown.




Establishing Setting Select "version ECLR ( Core.3.0. 8 )" , as shown.



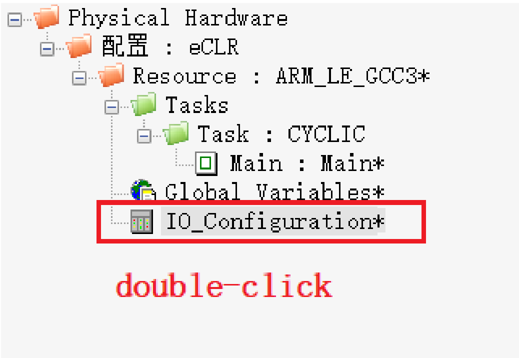

When finished, click "OK" again to re-click "Create" button; the program no error message before downloading to a real PLC in .

# 7 .4 IO Configuration

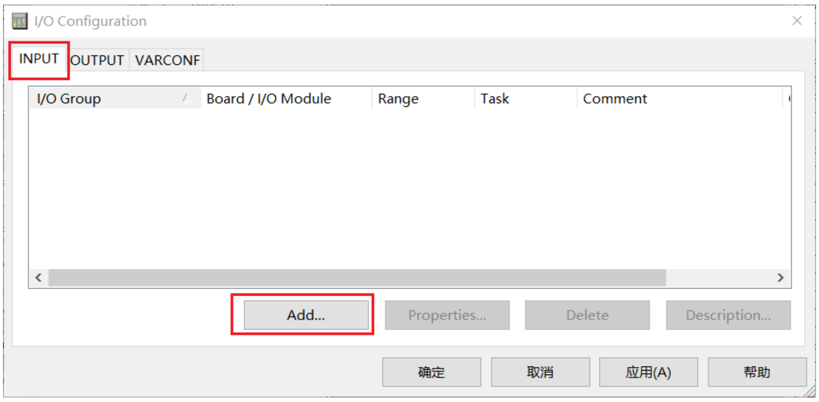
When the program is executed, the controller receives the signal from the field device through I/O and sends the control command to the field device. Therefore, the user must specify the logical start address. The driver name is the driver that specifies the I/O . Otherwise, the compiler will appear. " The address of the I/O variable 'xxx ' does not match any of the I/O groups! " error message.

The following I/O driver settings are made;

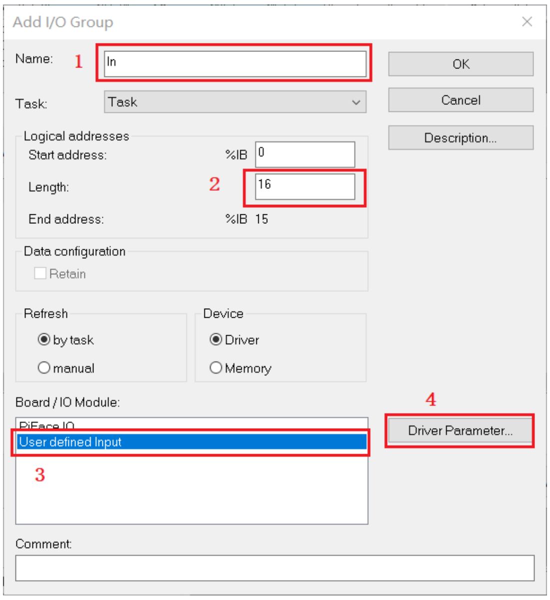
1 , double-click " IO_Configuration " open the I / O configuration dialog, which is used to edit I / O configuration of the worksheet, as shown.



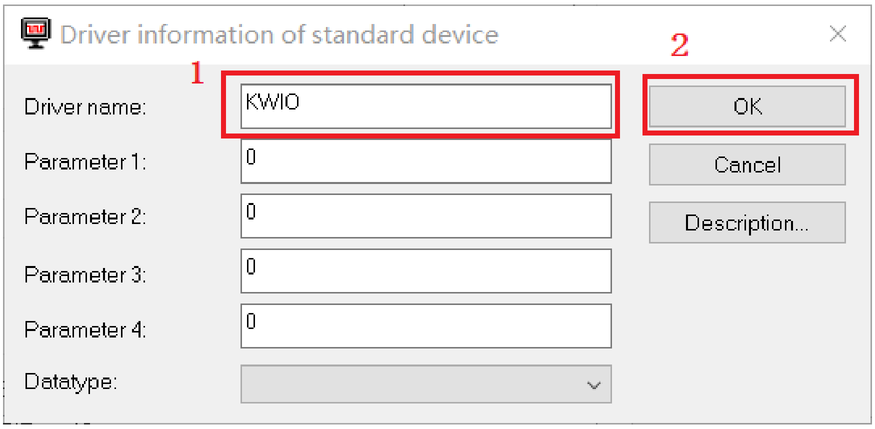

2 , Double-emergence " the I / O Configuration " dialog box , select " the INPUT " click "Add" shown in FIG.




3 , In the name fill " the IN ' according to the actual needs of the I / O edit the Configuration example: We want to define the existing group 16Byte input points, the " length " field, enter 16 , represents the input address bits : %IB0--%IB15 with 16 input bytes . as the picture shows

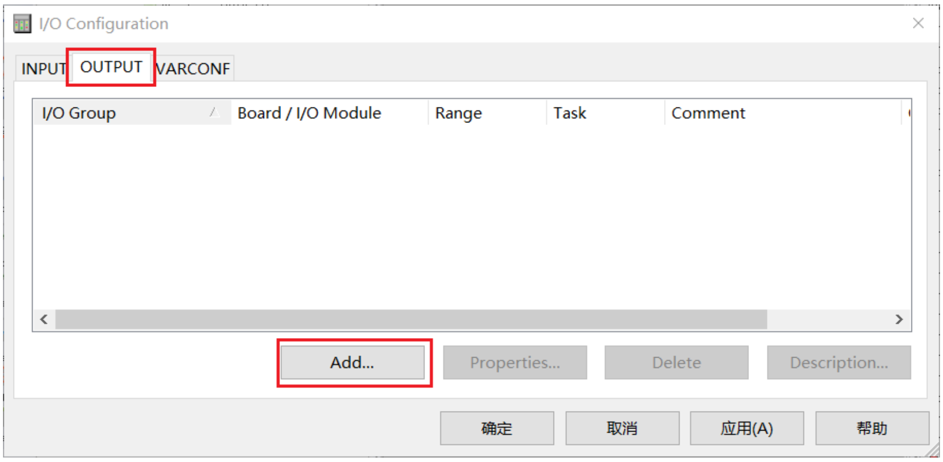



4 , click < driver parameters >, " in the driver name" was changed to " KWIO " shown in FIG.




5 , repeating the above steps for the same output terminal of the set , select " the OUTPUT ", "name" click "Add" to the output " OUT ", the logical address "length" was changed to 16 , and the "driver name "Modify to " KWIO " and click "OK" to complete the I/O driver setup .

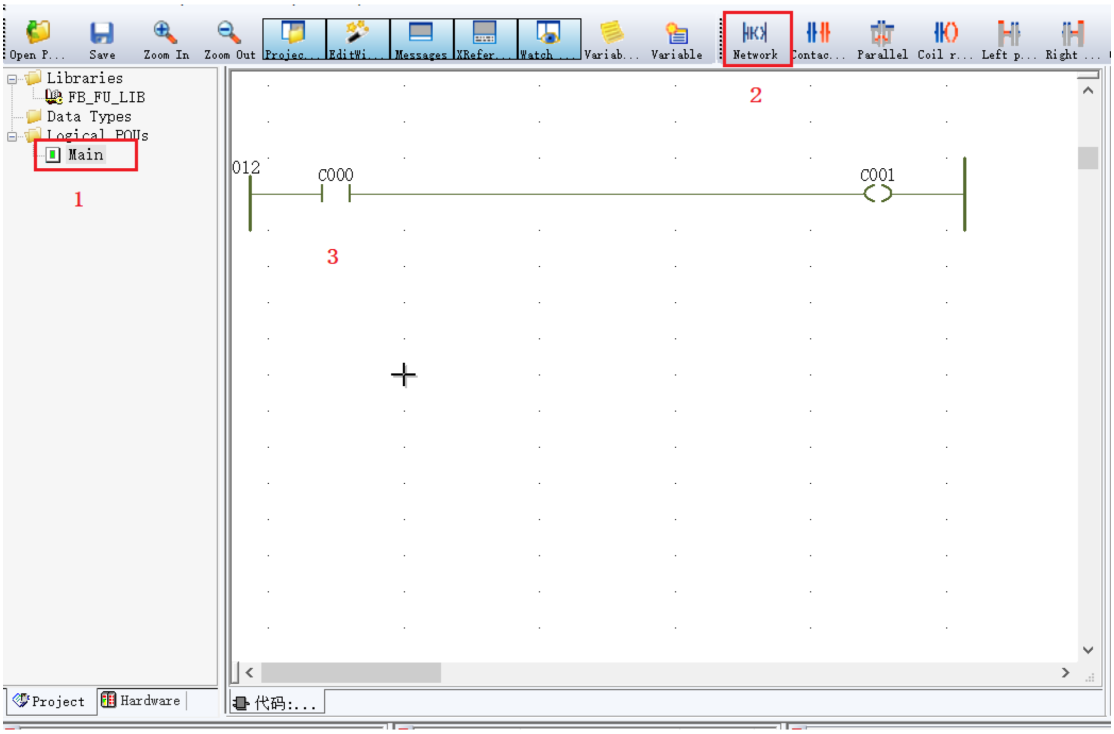
Note: I/O settings are required each time the user creates a new project .



# 7 .5 Write Ladder Code

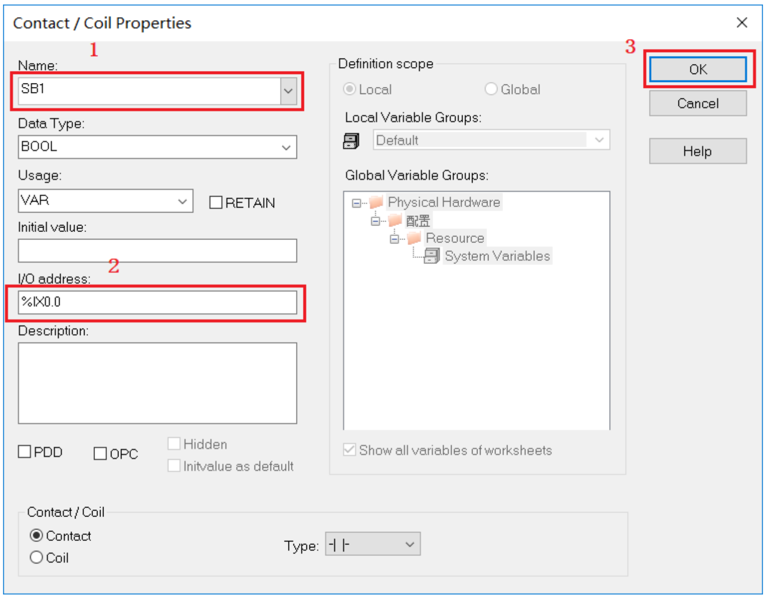
## 7 .5.1 Insert New Network

1 ) Double-click the project tree " main " project name, A blank editing window; click " network" , the occurrence region editing a ladder FIG network , the left side is a normally open contact, variable The right side of the name C000 is a coil with the variable name C001 .

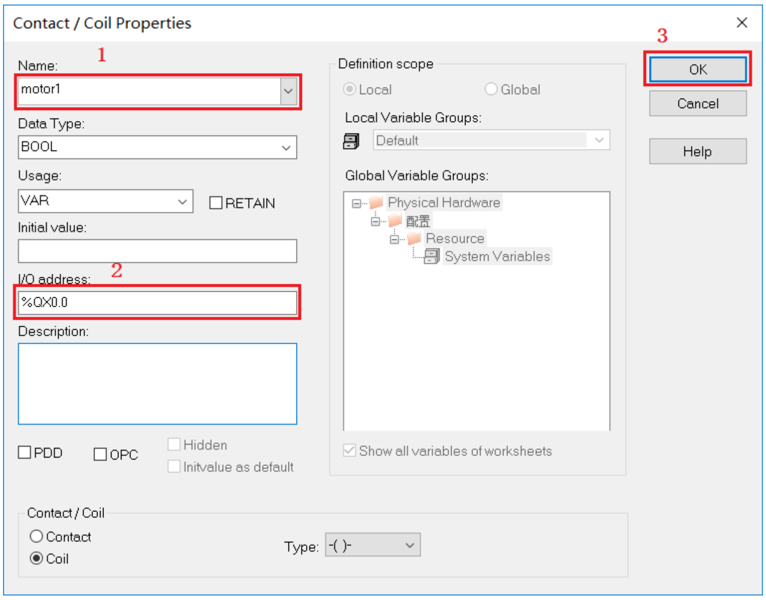



## 7 .5.2 modify variables of the property

1 ) Double-click the normally open contact " C000 " , appears contacts / Coil Properties dialog box, the I / O address ( S ) field, enter % IX0.0 , represents PLC a first digital input channel of the machine, a single click " OK " , as in FIG illustrated

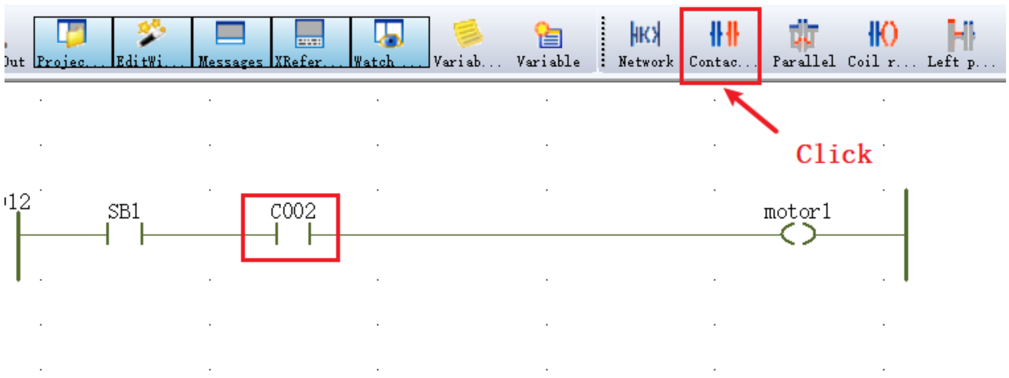



2 ) Double-coil ladder C001, contact may / Coil Properties dialog box, the I / O address ( S ) field, the input field % QX0.0 represents , represents PLC a first digital output channel of the machine, Click "OK " as shown

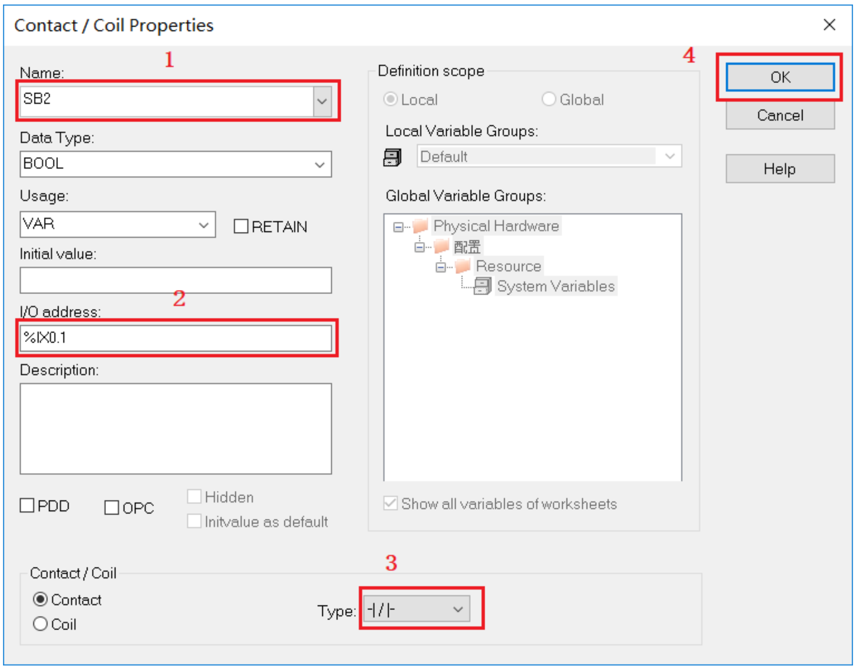



## 7 .5.3 Insert new contacts

1 ) Click " SBl ", iNSelected cases, the contact click on the toolbar as shown, i.e. " SBl " inserted normally open contact C02 , as shown;



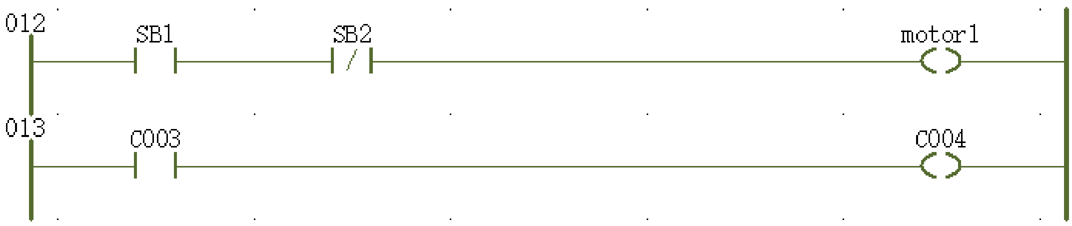

2 ) Double-click " C002 " Modify contacts its properties shown in FIG.




Similarly, the normally closed contact of the thermal relay is inserted:

## 7 .5.4 inserting a second LD network

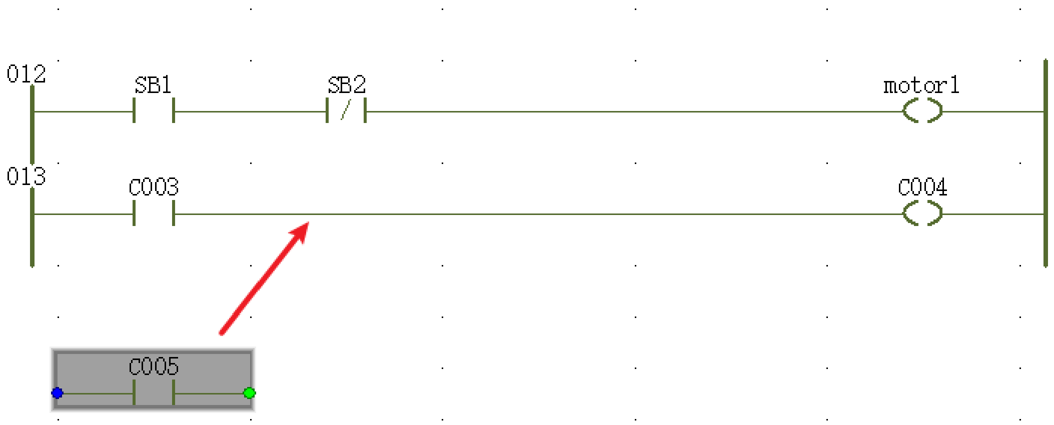
1 ) Move the cursor to the bottom of the existing network, a single click Tools "Network" button to insert a new network , as shown.

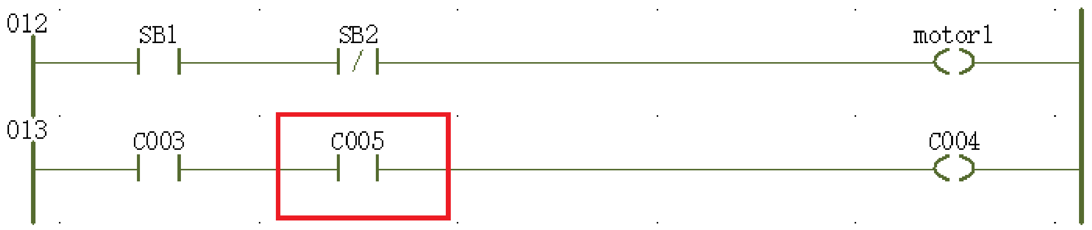


## 7 .5.5 line drawing anDAnnotation connection

1 ) Connect the lines

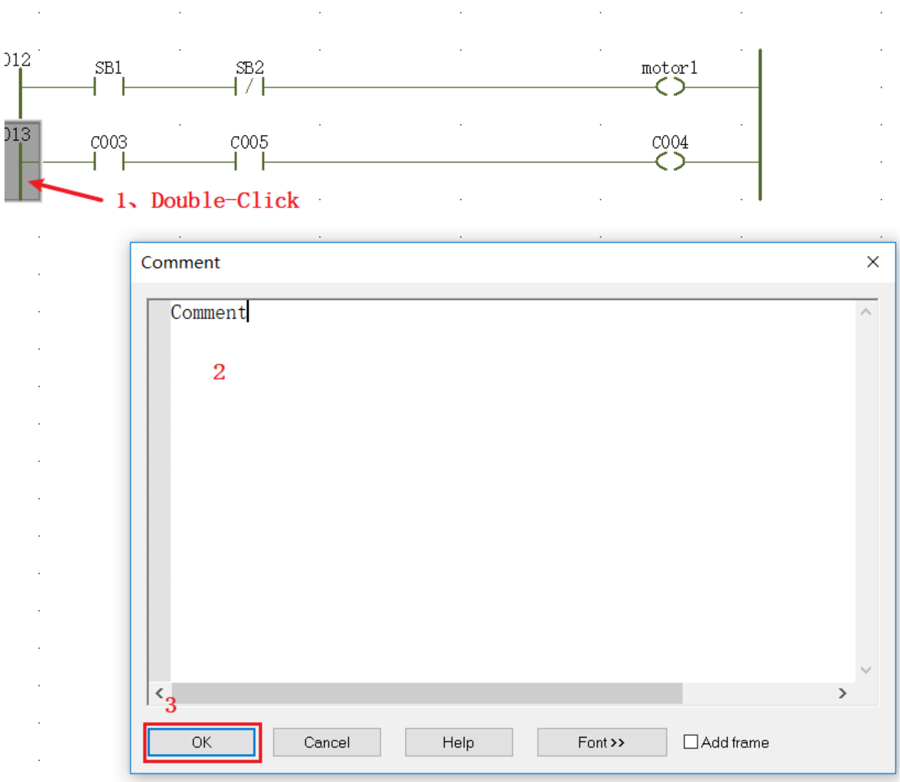
MULTIPROG software, provides a convenient drag and drop feature, click on the toolbar contacts " " , appears a new "normally open contact", double-click the new contact to modify its properties, and then to drag contacts black nodes can occur on the network to be connected , as shown.



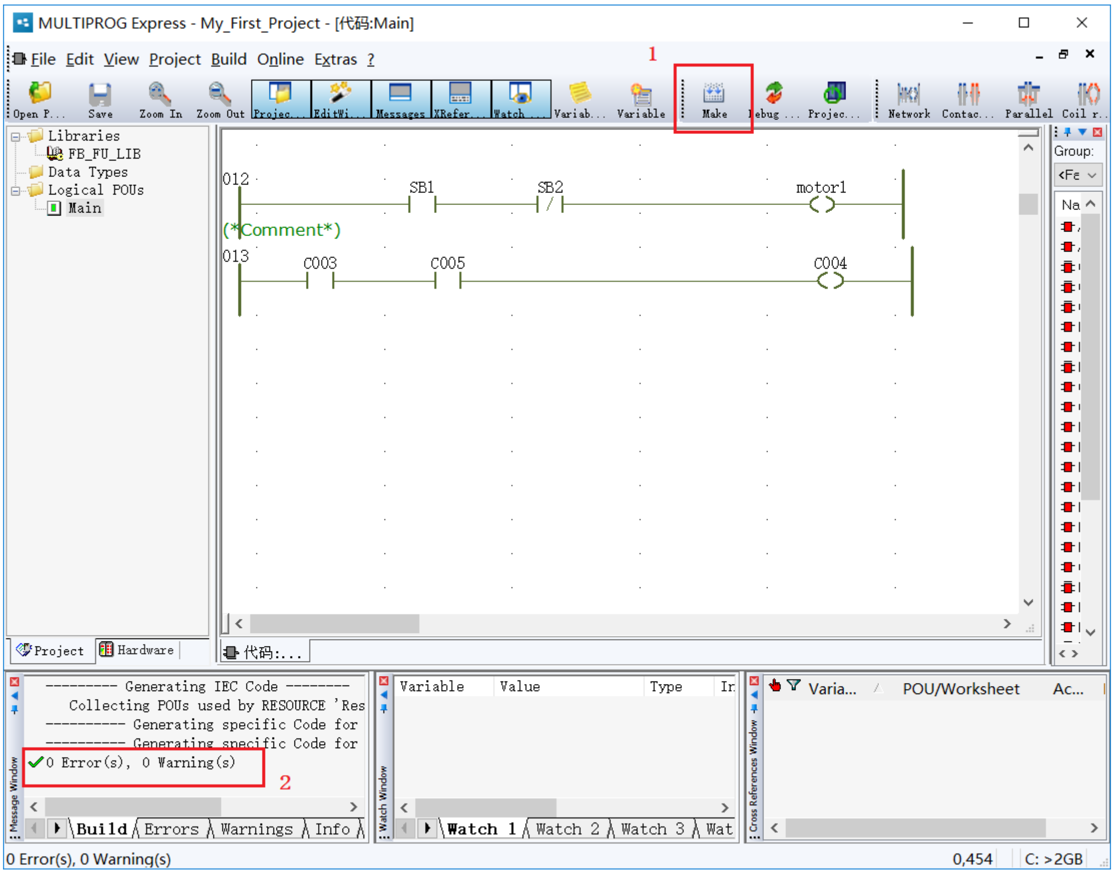

2 ) ladder diagraMComments

Double-click the left busbar label of the network and fill in the prograMComments in the pop-up "Comment" dialog box so that you can understand it later.


012
SB1
SB2
motor1
013
C003
C005
C004
1、Double-Click
Comment
Comment
2
3
OK
Cancel
Help
Font >>
Add frame

## 7. 6 Production and compilation of projects

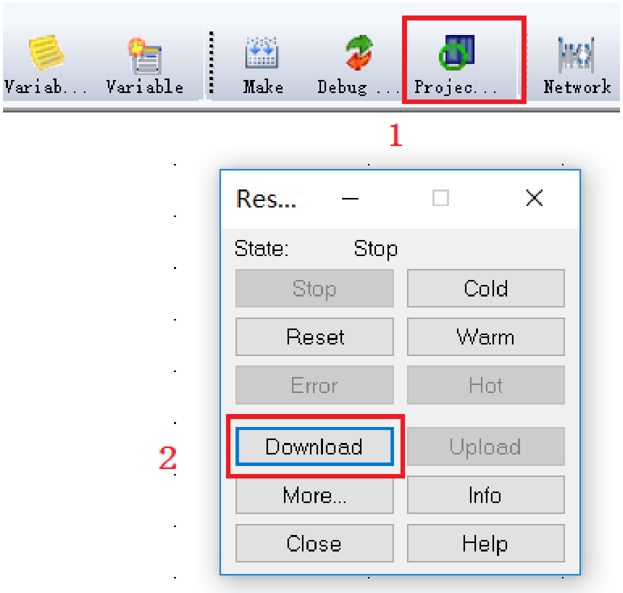
1 ) click Compile " made after" informatioNStatus area no error prompts carried out under a step simulatioNStage, if prompted , double-click on the "wrong" in the message area prompts, the software automatically jump to the wrong place, modify the finished Then click "Make", until the informatioNStatus area has no error prompt, then "Download" , as shown



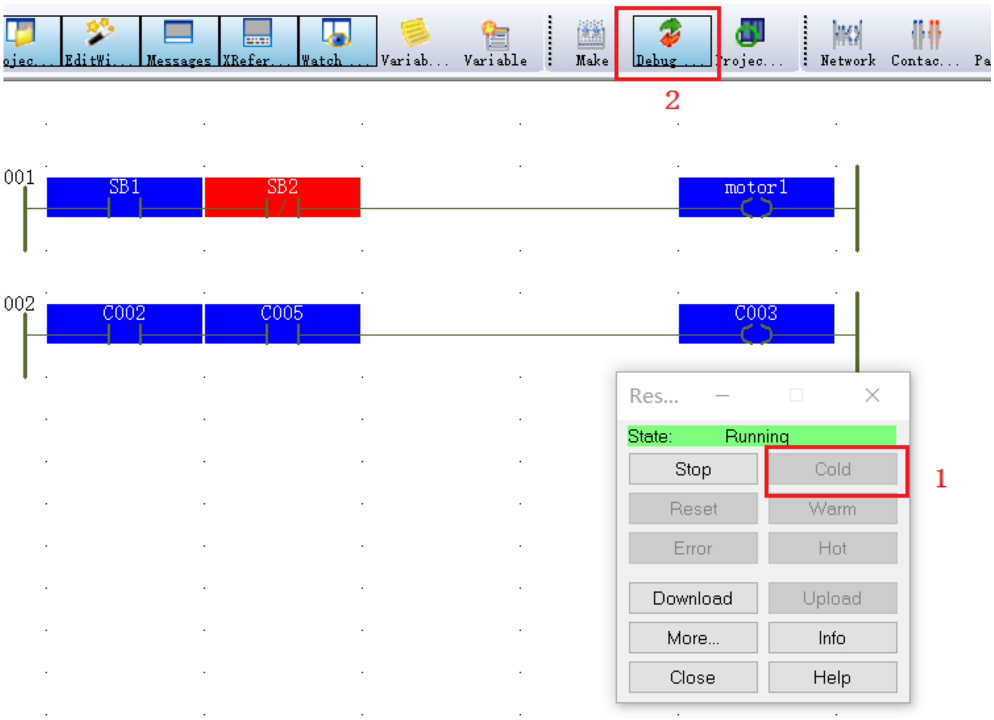

The engineering project created so far is basically completed .

# 7. 7. Program download to PLC / simulation

1 ) In the communication After setting, click the Toolbar " Project control dialog window " , pop-up to "Resources" panel , click download, the already completed projects downloaded to the emulator on the device , downloading process there download progress , as shown.




2 ) After the program "Download" is completed, click the "Cold Start" program on the resource panel to run, and the status display "Run" indicates that the user program is continuously looping; the user can debug online, monitor the status of the variable, click "Debugging on " on the toolbar , Screen effect during simulation , as shown.



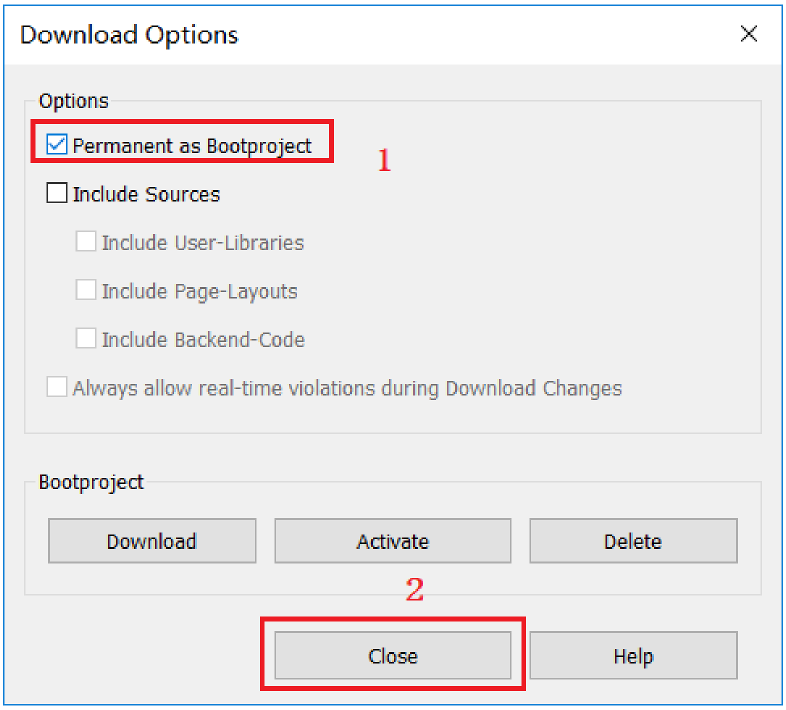
📖·**panel function introduction**

◆·cold start : indicates that the PLC starts executing the program from the initial state, anDAll variables are defined initial values at the start time;

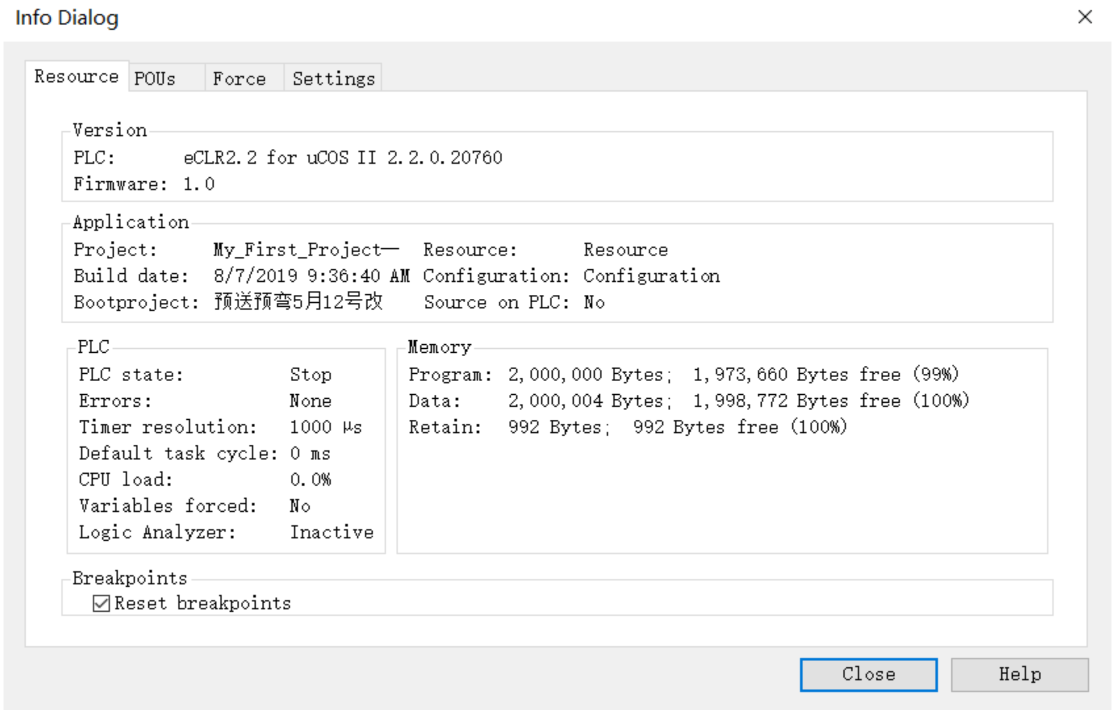
◆·Warm start : Indicates that the hold type variable in the program maintains the state at the last stop, and the other variables are the initial state.

◆·Tip: Click the “Stop” button in the “Run” state to pause the operation of the PLC . All the variables in the program will remain at the moment before the stop. At this time, the “warm start” button is available. Click this button to continue the program. run.

◆·: It is used to choose whether to save the program downloaded by the user (after the VA motion controller is powered off). Before the user program is installed, you need to click the “More” pop-up window to check the box as “Guided Project Permanent Station” . Leave" and then click "Close" as shown in the figure (Note : If unchecked, the PLC will not be able to run the last downloaded program after power-on, that is, the motion controller does not save the last downloading program. Need to reinstall the program)




◆·Info:It is used to view the PLC running status, the bytes occupied by the program memory, the current scanning period setting, etc.; as shown




Note: When an abnormality occurs during PLC operation (for example, the divisor is zero), the PLC will automatically stop, and the “Status” will be displayeDAs “Error” and the background color will be red. The "Error Button" becomes available at this time. Clicking this button causes the cause of the error to be displayed in the " PLC Error" tab of the MULTIPROG message window .

# Ⅷ Online Debugging and Monitoring Procedures

## 8 .1 force and coverage

In online mode, you can " force " or " overwrite " variables. In both cases, a new value is assigned to the corresponding variable.

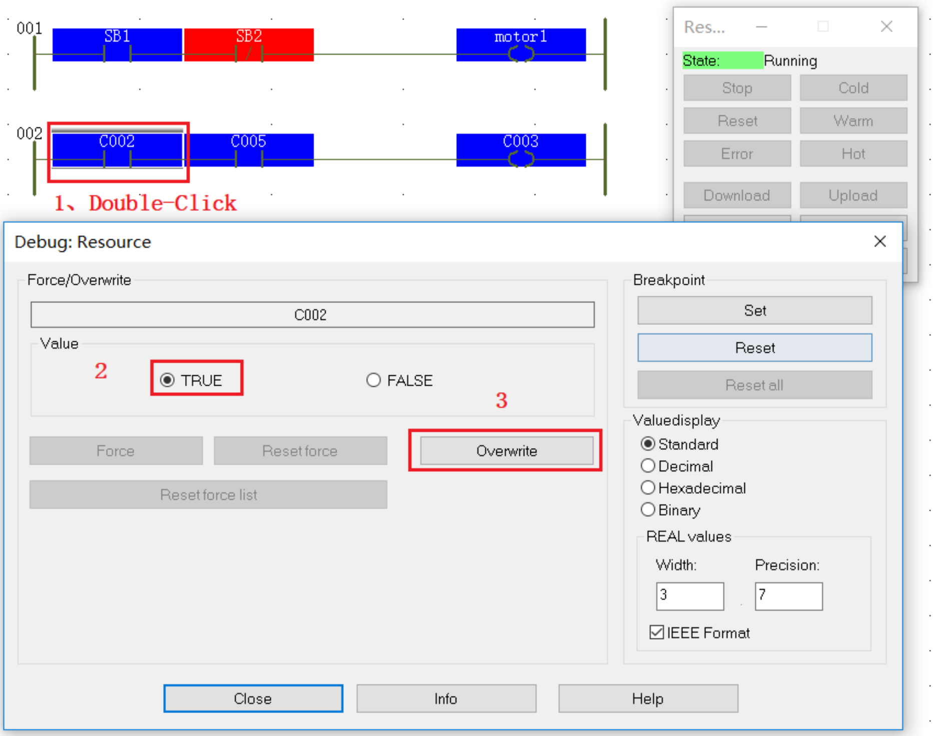
·Force: Assign a value to a variable ( usually a contact or coil ) . This value will remain until the reset is forced.

·Override: A value is temporarily assigned to a variable by the user. This value will remain until the program overwrites the value with the original value in the next program loop. The necessary steps to force and override a variable are almost identical. When the PLC is running, be careful to force or override the variables. Forcing and overriding variables means executing the PLC program with forced or overwritten variable values .

Use "force and override"

1 ➢To ensure that work orders in online mode. Otherwise, press the toolbar " Debug on / off ' icon:

2 Double-click "Variables" in the program to display the " Debug: Resources " dialog box as shown:

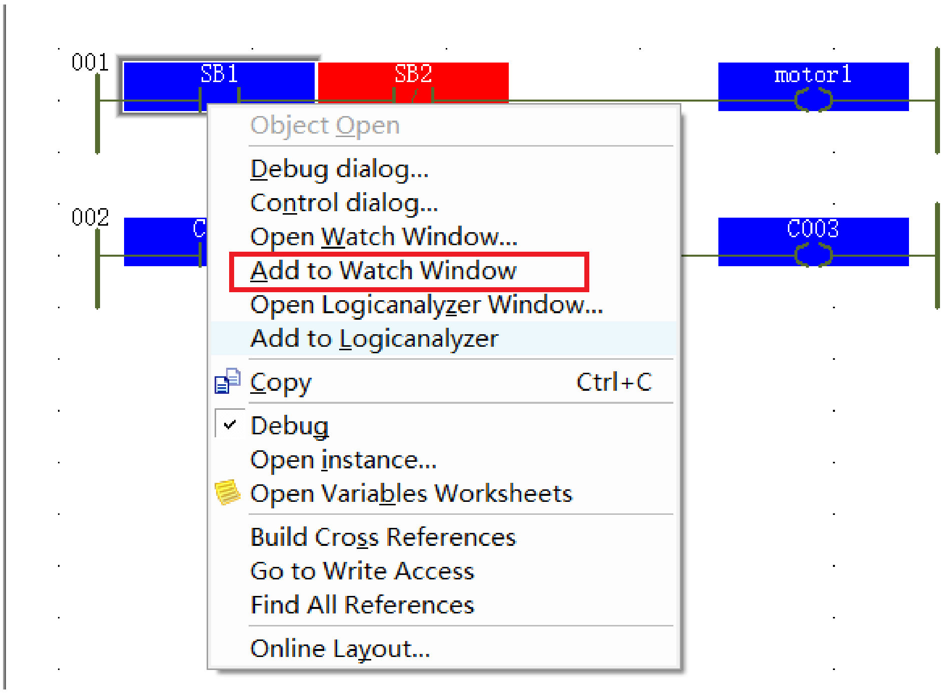


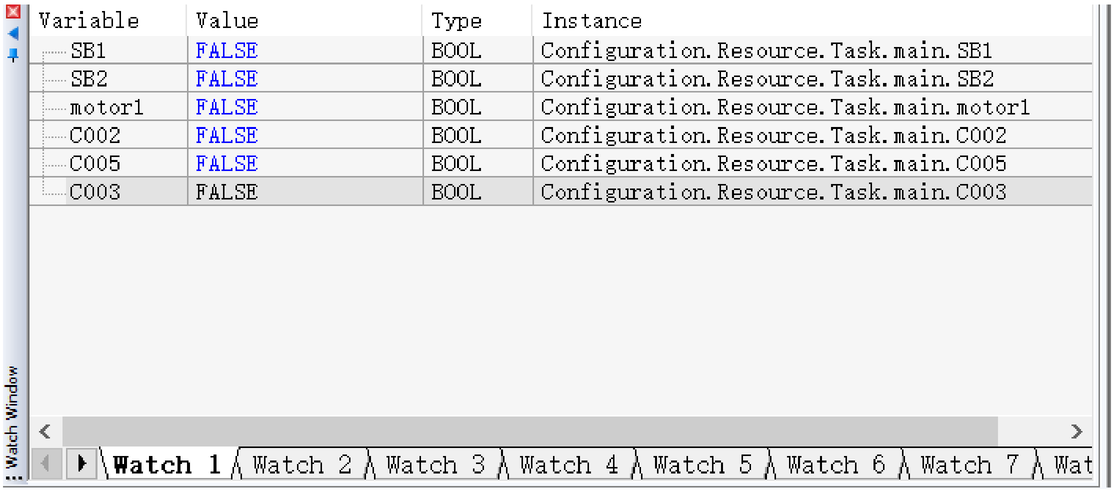

3 Select the radio button " TRUE " and click on the " Force " result, the variable will be forced to " ON " and will be highlighted in red on the online work order .

## 8 .2 variable monitor window

The Variable Watch window is a powerful tool that allows users to easily insert different variables into a list and observe their runtime behavior. Once a variable is added to the watch window, its current value can be monitored without having to open the corresponding work order. Users can focus on debugging and observing the variables that need to change . If this is not the case, please work

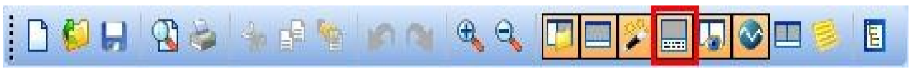
Single switch to the online mode, by pressing it to " Debug on / off " FIG subscript to. Right-click inside the work order and select " Open Watch Window " from the upper and lower menus or click the watch window button on the toolbar , in the ladder diagram, or in the variable worksheet, select the variable to be monitored right click "Add to Watch window" as shown






## 8 .3 Cross reference window

1 ) Cross-reference list contains all the variables used in the current project, power can block, jump, numerals and connectors. This tool for debugging and fault isolation particularly helpful, click on the toolbar " Cross Reference Window " icon to open the cross-reference window as shown below .

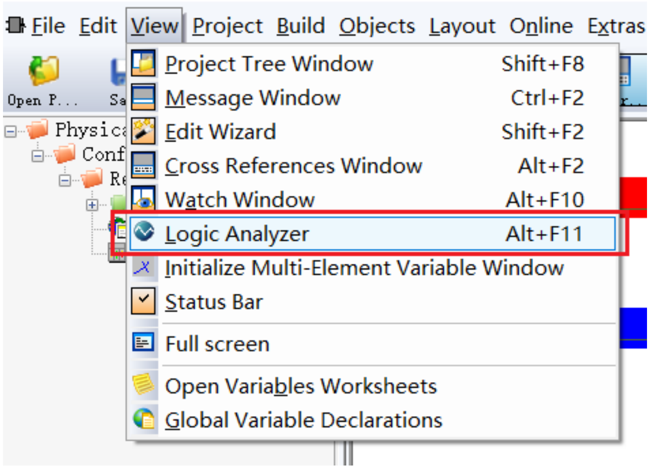


2 ) Place the cursor in the cross-reference window and right- click on the window background to open its context menu. Select the " establishment of cross-reference " menu item will create a cross-reference list , as shown .

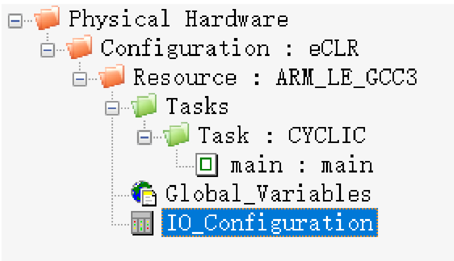

3 ) Double-clicking on a variable in the cross-reference window will open the worksheet that uses this variable and highlight it. Also, if you mark a variable in the work order, the corresponding variable in the cross-reference window will also be marked. The cross-reference list contains all the variables, function blocks, jumps, labels, and connectors used in the current project. This tool is especially helpful for debugging and error isolation.

# 8 .4 logic analyzer

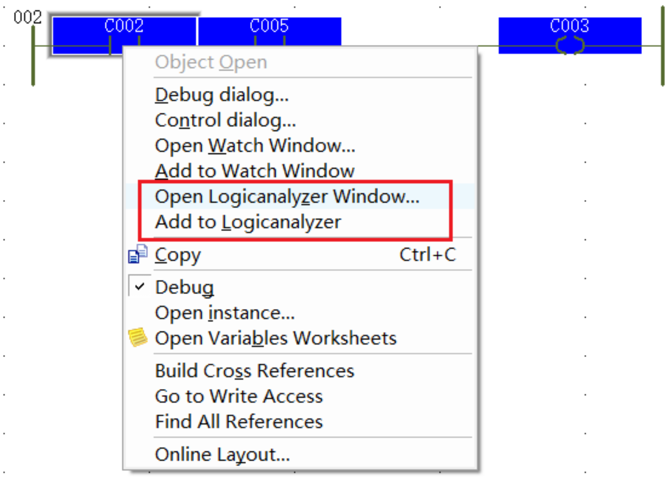
1 ) The logic analyzer can monitor the trend change of multiple analog quantities (such as temperature, liquid level, pressure, etc.), and caNSet the sampling time, which is convenient for users to debug. Select "Logic Analyzer" on the toolbar and pop up under the software. Logic Analyzer" as shown

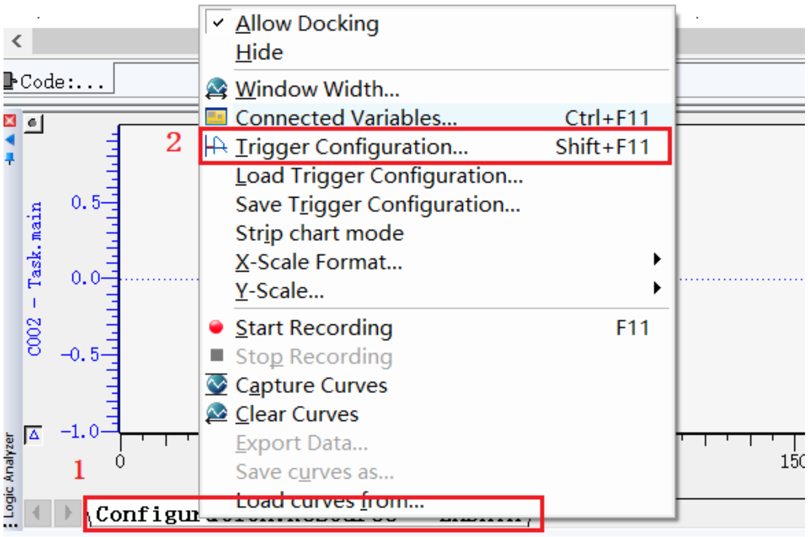



2 ) Before using the logic analyzer, the user needs to change the attribute names of "configuration", "resource" and "task" under the "physical hardware" of the project tree to English name or pinyin. No Chinese characters can be used. The analyzer is set as shown below , as shown

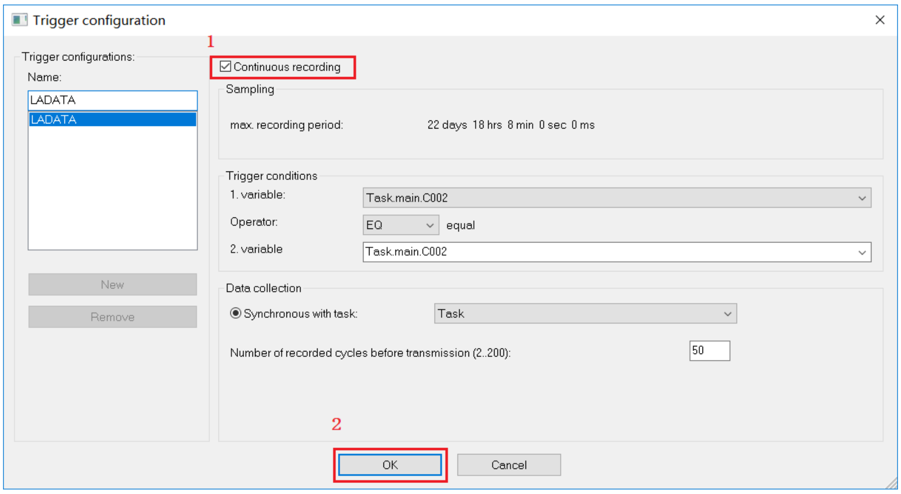



3 ) In the "Debug On / Off" opeNState , right click to add the " variables " to be monitoreDAnd select "Add to Logic Analyzer ". At this time, the "Logic Analyzer" displays tick marks of different colors, indicating that the user has added more different variables, the playing out of window lower left following settings



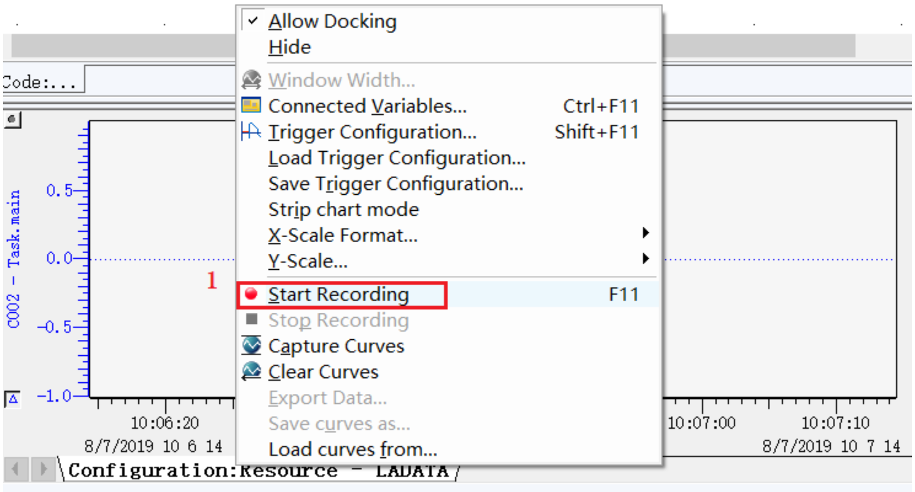




4 ) In the pop up "trigger Configuration" "check" continuous recording , and click " OK " , shown in FIG.

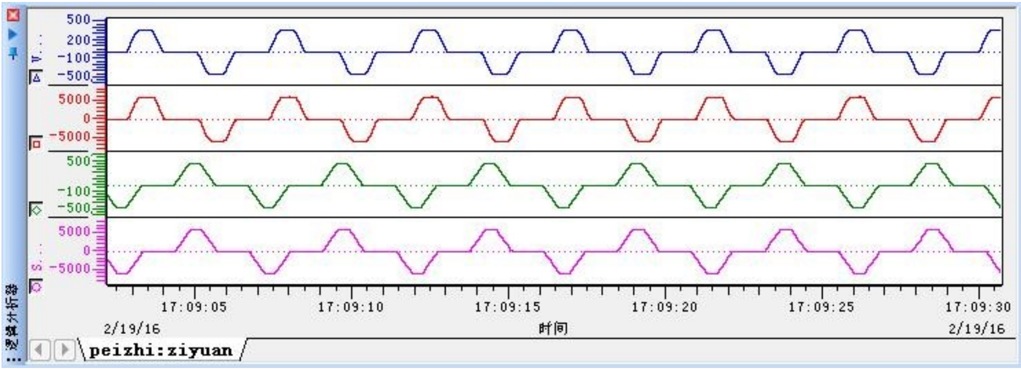



5 ) After completing the above settings, " right click " the logic analyzer name , click " Start

Record " as shown




6 ) Note that when you click " Start Recording " , try not to perform other operations . Otherwise, the communication will not be smooth or crash . If you want to perform other operations, stop the recorder first and then perform other operations. The final monitoring result is as follows: The figure shows . To stop monitoring, repeat the fifth operation and select "Stop Recording" .



## 8 .6 breakpoint debugging

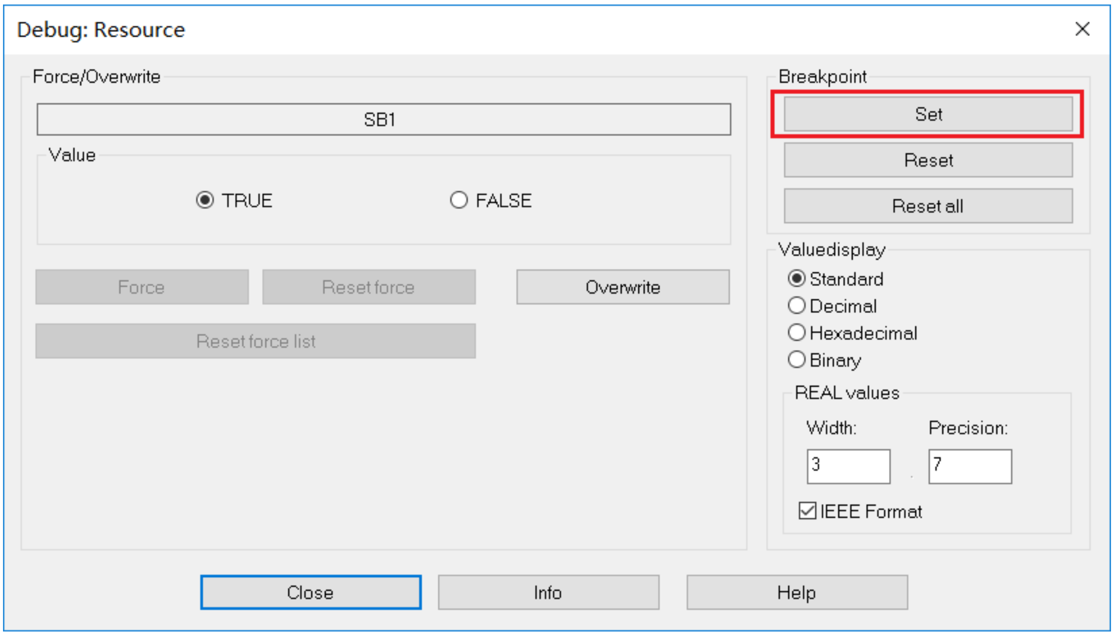
1 ) Like other high-level language development tools, MULTIPROG supports setting breakpoints for PLC programs for debugging programs. After setting a breakpoint, there are two ways to debug the program: single step and trace.

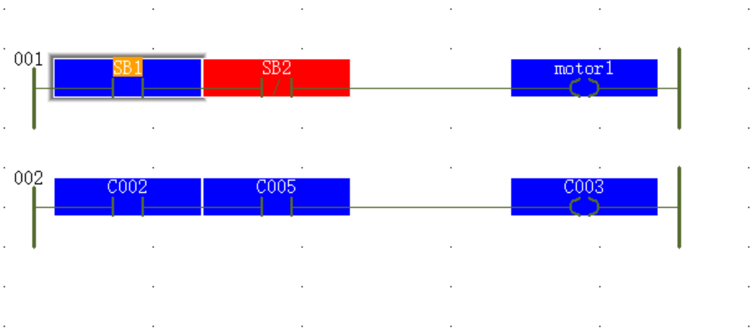
✍·Single step: The PLC executes the next instruction of the current instruction. If it is a function or a function block call, it will execute the complete call process to get the result.

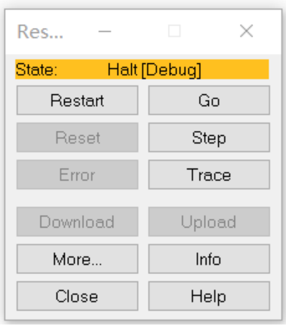
✍·Tracking: The PLC executes the next instruction. If it encounters a user-defined function or a function block call, it will open the corresponding code body. The tracking process executes only one instruction at a time.

1 Turn on the debug mode;

2 Double-click the " SB1 " variable in the code worksheet and click the "Settings" button in the "Debug : Resources" dialog box ; in the online worksheet, " SB1 " is highlighted in orange as shown in the figure. The status of the project control dialog box will also change to "Pause [ Debug ] ", the background is orange, and the button will be programmeDAnd debuggeDAccordingly, as shown in the figure;



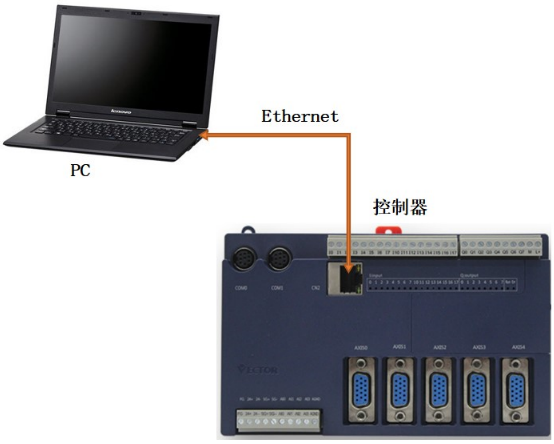



2 ) MULTIPROG supports two standard communication methods, one is serial port and the other is Ethernet based on T CP/IP mode. To facilitate programming and debugging by programmers, MULTIPROG has two built-in analog controllers, "Simulation 1 " and "Simulation 2 " , which are all applications running on the same system as MULTIPROG . Note : If you run the simulation controller operating system is non-real-time, the simulation can only be used for functional verification prograMCan not be used to test real-time "to establish the setting" is used to define the IEC code is compiled downloaded to the controller The set of library definition files of the machine code executed in the machine may have different machine code generated by different versions, so it is generally necessary to select the latest version of the "Create Settings" file. "Online update", "interval" refers to the use of MULTIPROG the Debug mode, the value of the internal variable in the controller MULTIPROG bounDAnd the upper display update cycle.

# Ⅸ Quick Start

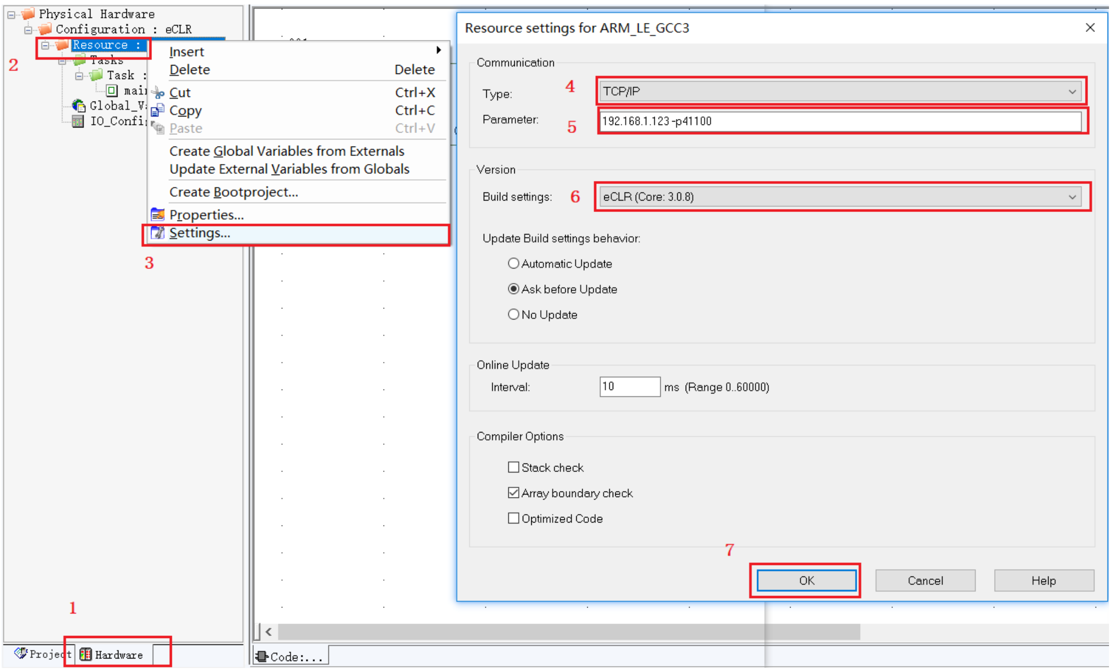
## 9 .1 software and motion controller establishes a connection (Ethernet port communication)

1 : Communication connection diagram




2 : Engineering communication configuration

After the project is created, the communication is quickly configureDAs follows. (Reference: 7 .1 create projects , 7 .3 physical communication parameters , 7 . 4 IO configuration )




Step 1: After the project is created, select “ Hardware ” in the project tree ;

Step 2: Select " Resource : ARM_LEGCC3 " right-click;

Step 3: Settings ;

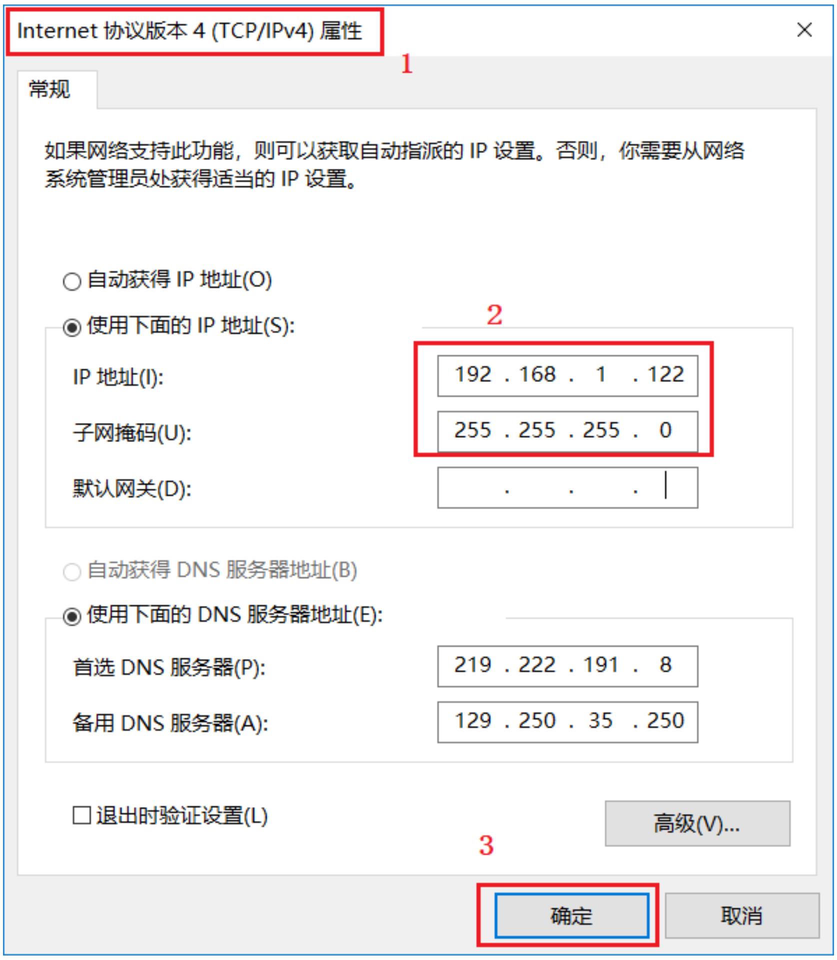
Step 4: communication type: select " TCP / IP " ;

Step 5 : IP address " 192.168.1.123-p41100 " (The IP address on the controller is not allowed to be fixed ) ;

Step 6: Version establishment: Select eCLR ( Core.3.0.8 ) ;

Step7: Click "OK" .

PC computer settings as shown below




Settings on your computer:

Step1: Click -> Start -> ControlPanel -> Network and Internet -> Network - >Connection "Local Area Connection" - > Properties -> Double-click " Internet Protocol Version 4 ( TCP/IPV4 )";

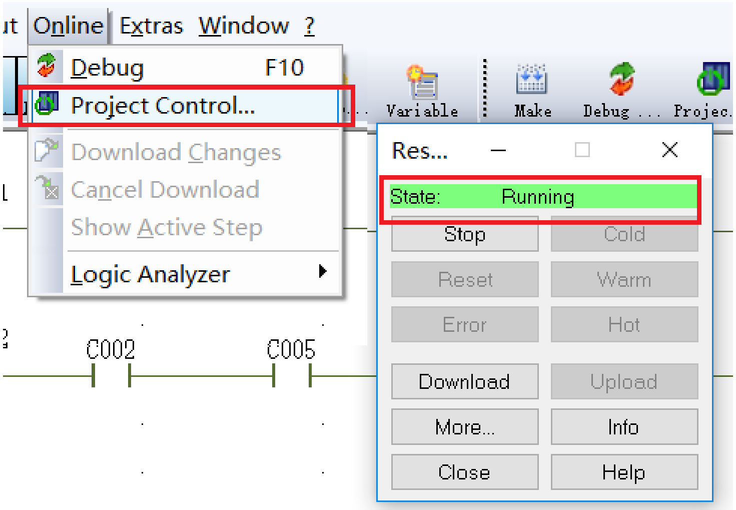
Step 2: Select "Use the following IP address ( S )"， Enter IP be: 192.168.1.122 Subnet Mask: 255.255.255.0 ；

Step3: click "OK" .

3 : Check if the configuration is successful

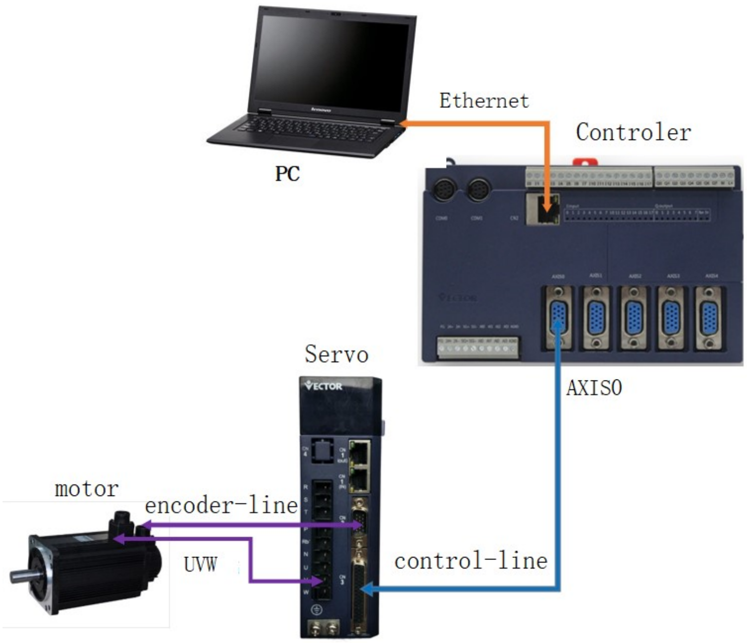
After completing the above steps, click "Project Control Dialog", the "Resources" dialog box will pop up , the status is displayed .

Show "Run", indicating successful communicatioNSettings, whether the status display "Timeout", then check the PC port number is consistent with that software, as shown below .



## 9 .2 control control by sending the analog servo motion (uniaxial start and stop)

**1 : Communication and control connection diagram**




According to the above system , the servo motor is controlled to start, run and stop by the motion controller.

**2 : Set the servo drive parameters**

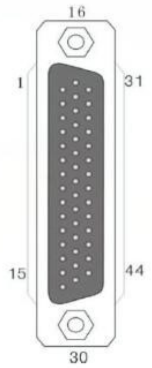
Wikoda VC type servo drive parameter setting ( Please contact our staff for VB servo drive parameter setting method)

| Function number | Numerical value | description |
|---|---|---|
| P0 2 . 01 | 1 | Speed control mode |
| P 04 . 01 | 0 | Speed is derived from the maiNSpeeDA |
| P 04 . 02 | 1 | The maiNSpeeDA is derived from the analog AI 1 |
| P 06 . 01 | 1 | The DI 1 function register function is set to: enable drive |

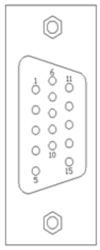
**3 : control wire connection method**

Wei Keda VC type servo drive is the CN . 3 portions pin definitions

| Pin | signal | description | |
|---|---|---|---|
| 37 | OA+ | Encoder signal amplification output | |
| 38 | OA- | | |
| 39 | OB+ | | |
| 40 | OB- | | |

| | | | |
|---|---|---|---|
| 12 | AGND | Analog ground |  |
| 14 | AI1 | Analog input | |
| twenty four | DI1 | Servo DI1 input | |
| 1 0 , 2 6 | + 24 V | External DC24V power supply , for servo DI , DO work use , Remarks: 25 , 26 feet for NPN / PNP Jumper choose to use | |
| 9 , 2 5 | COM | | |

Wilcoda motion controller AXIS port part pin definition

| Pin | signal | description | |
|---|---|---|---|
| 6 | A+ | Encoder signal input |  |
| 7 | A- | | |
| 8 | B+ | | |
| 9 | B- | | |
| 5 | AO+ | Modulus output | |
| 10 | AGND | Analog ground | |

Control line connections are as follows

| Servo (CN3) | | Controler (AXIS0) | |
|---|---|---|---|
| 37 | OA+ | 6 | A+ |
| 38 | OA- | 7 | A- |
| 39 | OB+ | 8 | B+ |
| 40 | OB- | 9 | B- |
| 12 | AGND | 10 | AGND |
| 14 | AI1 | 5 | AO |
| 26 | +24Vjumper | | |
| 27 | SW-DI | | |
| 9 | COM | 0V | External |
| 10 | +24V | +24V | External |
| 24 | DI1 | Q0 | Control external terminal output |

📖·Description :

1 , in order to pass control of the motion controller Q 0 output, to control the servo drive is enabled , need to servo drive external DC24V power to servo DI power (if the servo internal

selection is enabled, there is no need then Q 0 and DI 1 ) ;

2 , DIx signal type ( NPN/PNP ) selection : SW-DI ( 27 feet ) and +24V ( 26 feet) are shorted to NPN ;

3 , in order to reduce interference, the differential signal ( OA + →·A + ) and ( OA- →·A- ), ( the OB + →·B + ) and ( OB- →·B - ) are connected with the twisted pair, the total of the housing.

**4 : PrograMCreation configuration and debugging**

·known

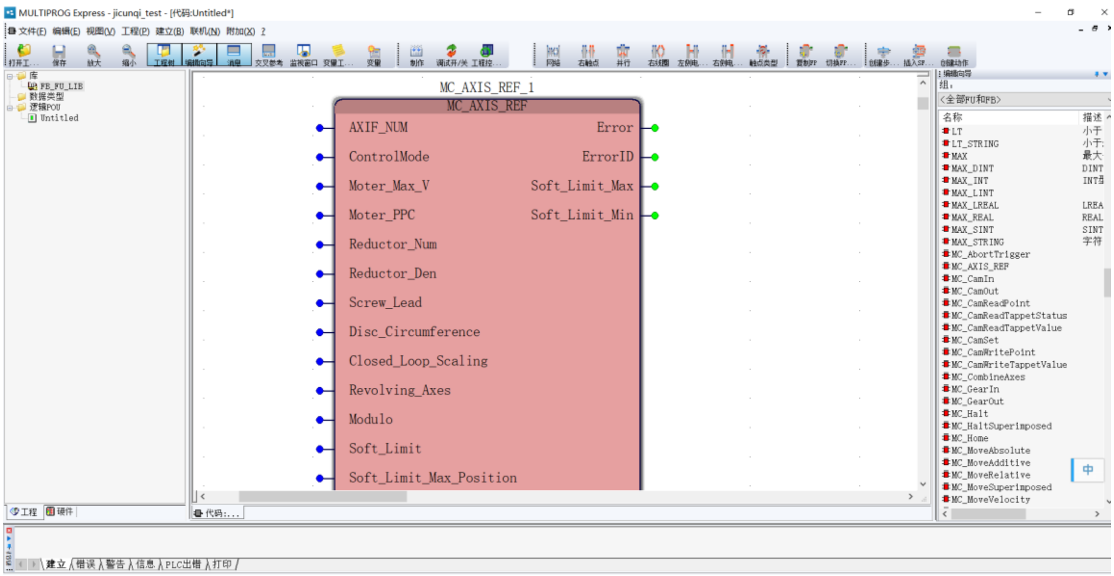
In the PC after a successful communication with the motion controller, set up the system, servo parameter setting is completed, the next start programming control servo motor run and stop; the default user before programming have read " Di Shiyi Zhang motion command " includes: 11.1 insert FB_FU_LIB motion control library , 11. The 2 movement instruction , 11.3 motion instructions basics and . 1 . 1 . . 4 uniaxially instructions.

**( 1 ) Project creation and configuration**

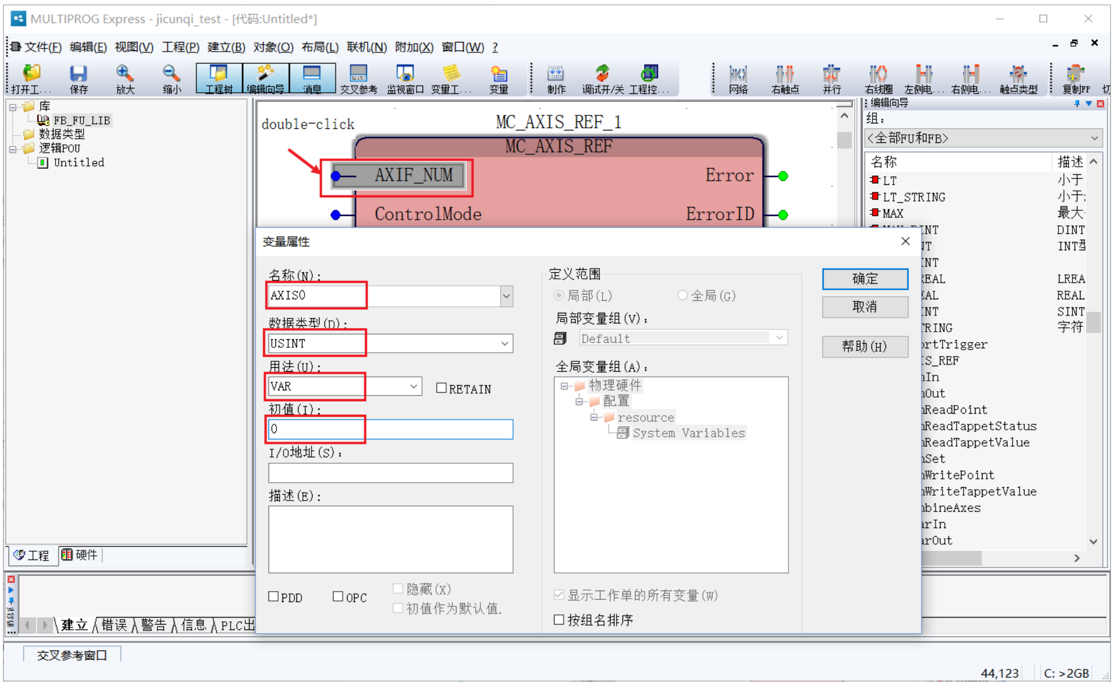
Reference "on Qi Zhang works of creation and configuration." Follow the steps to complete : 7 .1 create projects →·7 .3 physical communication parameters →·7 . 4 IO configuration , which will not be repeated herein.

**( 2 ) Writing a program**

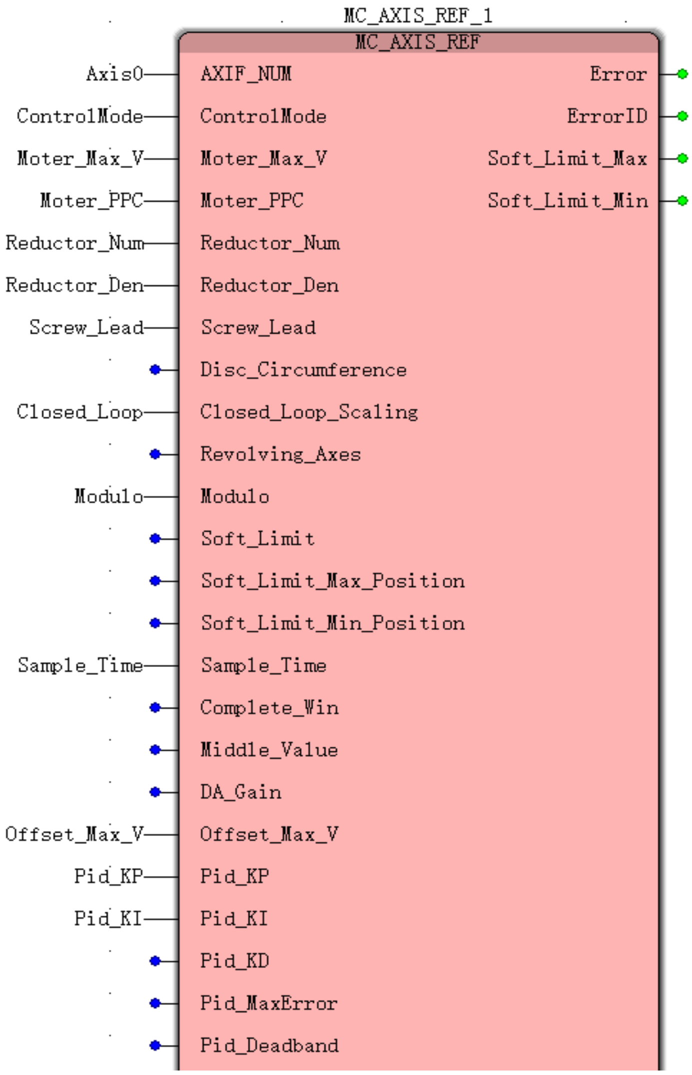
Step 1: Enter the programming interface, select the “ MC _ AXIS_REF ” block in the FB_FU_LI motion control library , hold down the left mouse button and drag to the programming interface, then let go, the module's properties will pop up, you need to name the module. Generally keep the default, click "OK" , as shown below ;




Step 2: Double-click the module input pin (blue dot), and the “Variable Properties” box will pop up to define the variable name, data type, usage, initial value, etc. , as shown in the figure (here the axis AXIS0 is useDAs the control). Axis) ;



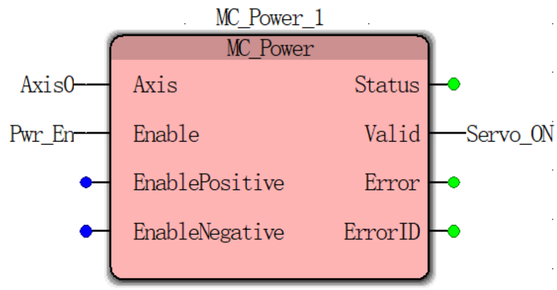

The user caNSimply fill out the default parameters axis parameters can not be set needs to be added, according to the reference function 11.3.3 MC_AXIS_REF ( axis parameter set ) , After completion of the examples herein , as the FIG illustrated ;




Variable name anDAttribute

| variable name | type of data | Initial value |
|---|---|---|
| MC_AXIS_REF_1 | MC_AXIS_REF | |
| Axis0 | USINT | 0 |
| ControlMode | INT | 0 |
| Moter_Max_V | DINT | 3000 |
| Moter_PPC | DINT | 10000 |
| Reductor_Num | LREAL | 1.0 |
| Reductor_Den | LREAL | 1.0 |
| Screw_Lead | LREAL | 60.0 |
| Closed_Loop | LREAL | 1.0 |
| Modulo | LREAL | 3 60.0 |
| Sample_Time | WORD | 20 |
| Offset_Max_V | DINT | 200 |
| Pid_KP | DINT | 80 |
| Pid_KI | DINT | 0 |

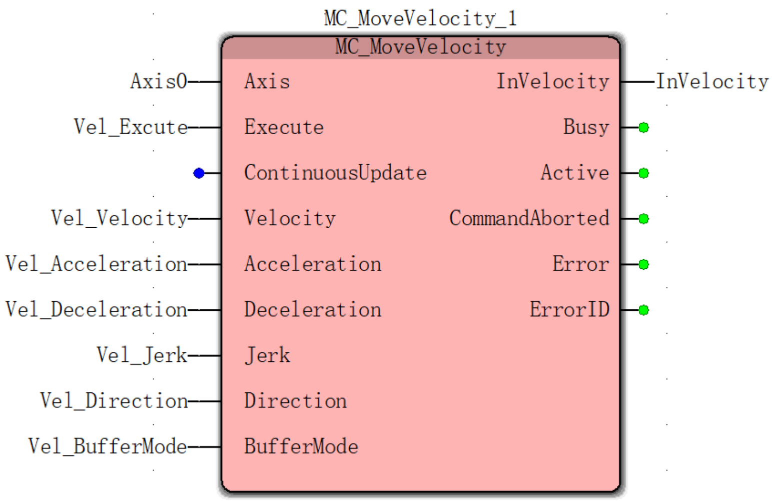
The third step: similarly add the " MC_Power " commanDAs follows , enable the motion controller, and control the enable of the servo drive through Se rv o _ON ( I / O address : %QX0.0 ) output , refer to 1 1.4.1 MC_Power ( enable command ) ;




Variable name anDAttribute

| variable name | type of data | Initial value | address |
|---|---|---|---|
| MC_Power_1 | MC_Power | | |
| Axis0 | USINT | 0 | |
| Pwr_En | BOOL | | |
| Servo_ON | BOOL | | %QX0.0 |

The fourth step: similarly add " MC_MoveVelocity " (speed command module) , used to control the servo motor to run at the set speed , refer to 1 1.4. 2 MC_ MoveVelocity ( speed command ) ;

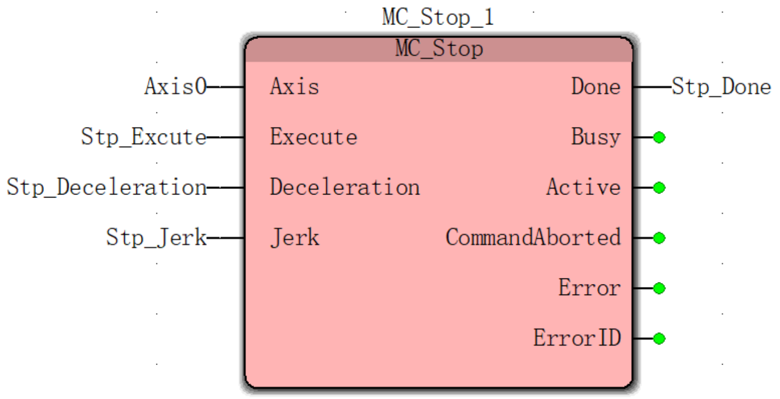



Variable name anDAttribute

| variable name | type of data | Initial value |
|---|---|---|
| MC_MoveVelocity_1 | MC_MoveVelocity | |
| Axis0 | USINT | 0 |
| Vel_Excute | BOOL | |
| Vel_Velocity | LREAL | 5 00.0 |
| Vel_Acceleration | LREAL | 1000.0 |
| Vel_Deceleration | LREAL | 1000.0 |
| Vel_Jerk | LREAL | 1000.0 |

| | | |
|---|---|---|
| Vel_Direction | INT | 1 |
| Vel_BufferMode | INT | 0 |
| InVelocity | BOOL | |

Step 5 : Add " MC_Stop " (stop command) . After the module is executed, the servo motor starts to decelerate and stop . Refer to 1 1.4.10 MC_Stop ( stop command ) ;

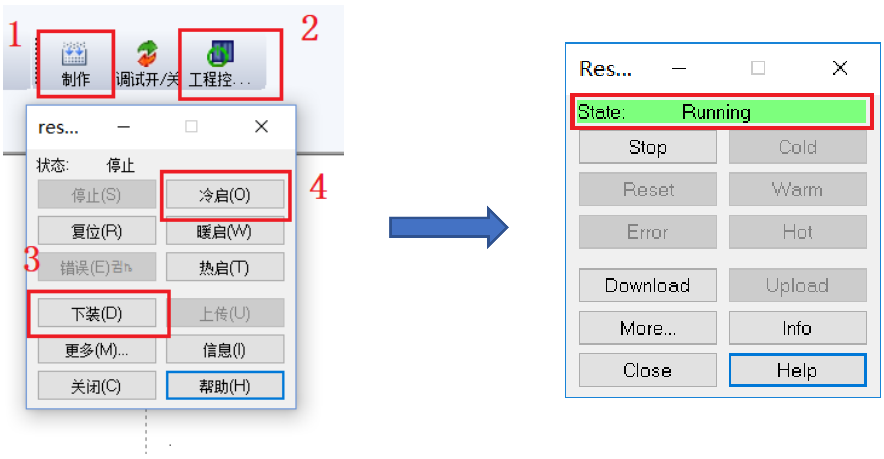



Variable name anDAttribute

| variable name | type of data | Initial value |
|---|---|---|
| MC_Stop_1 | MC_Stop | |
| Stp_Excute | BOOL | |
| Stp_Deceleration | LREAL | 1000.0 |
| Stp_Jerk | LREAL | 1000.0 |
| Stp_Done | BOOL | |

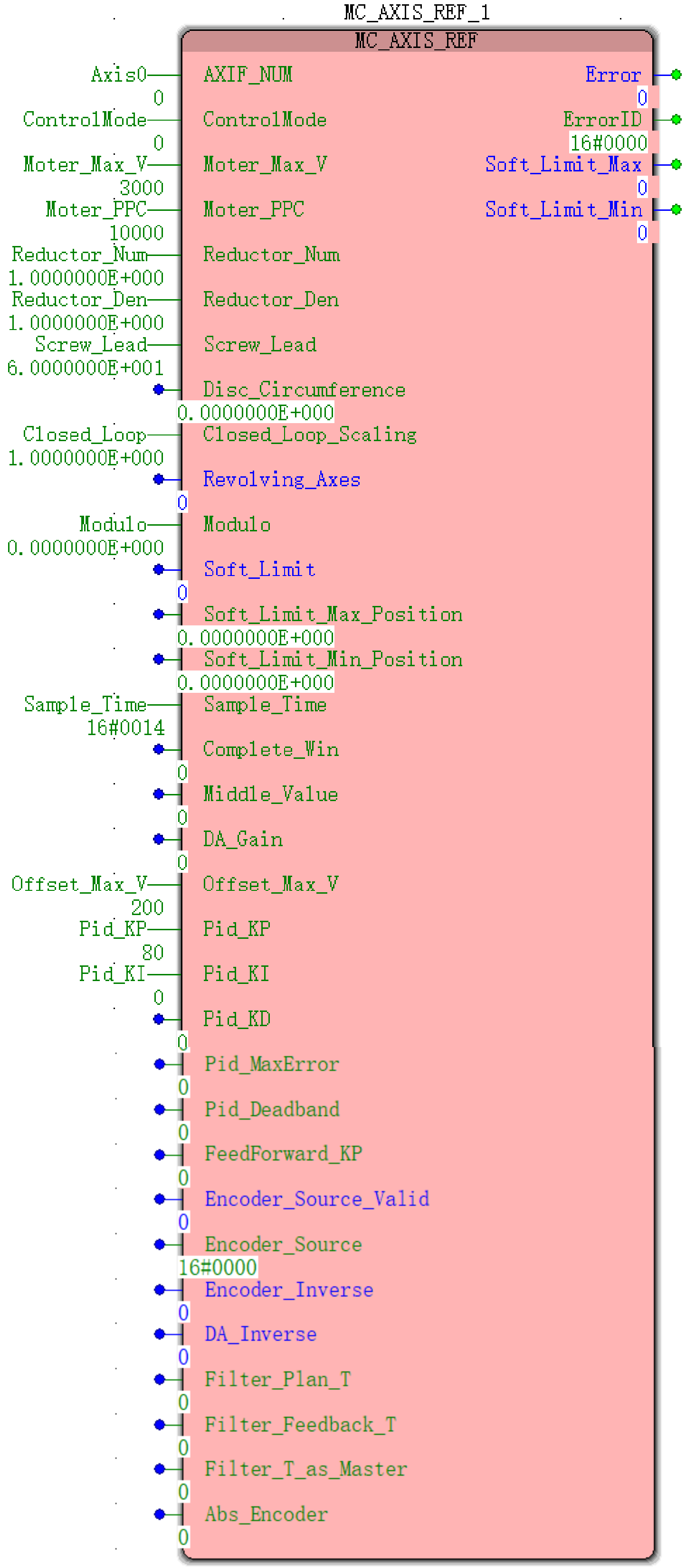
At this point, the programming is complete.

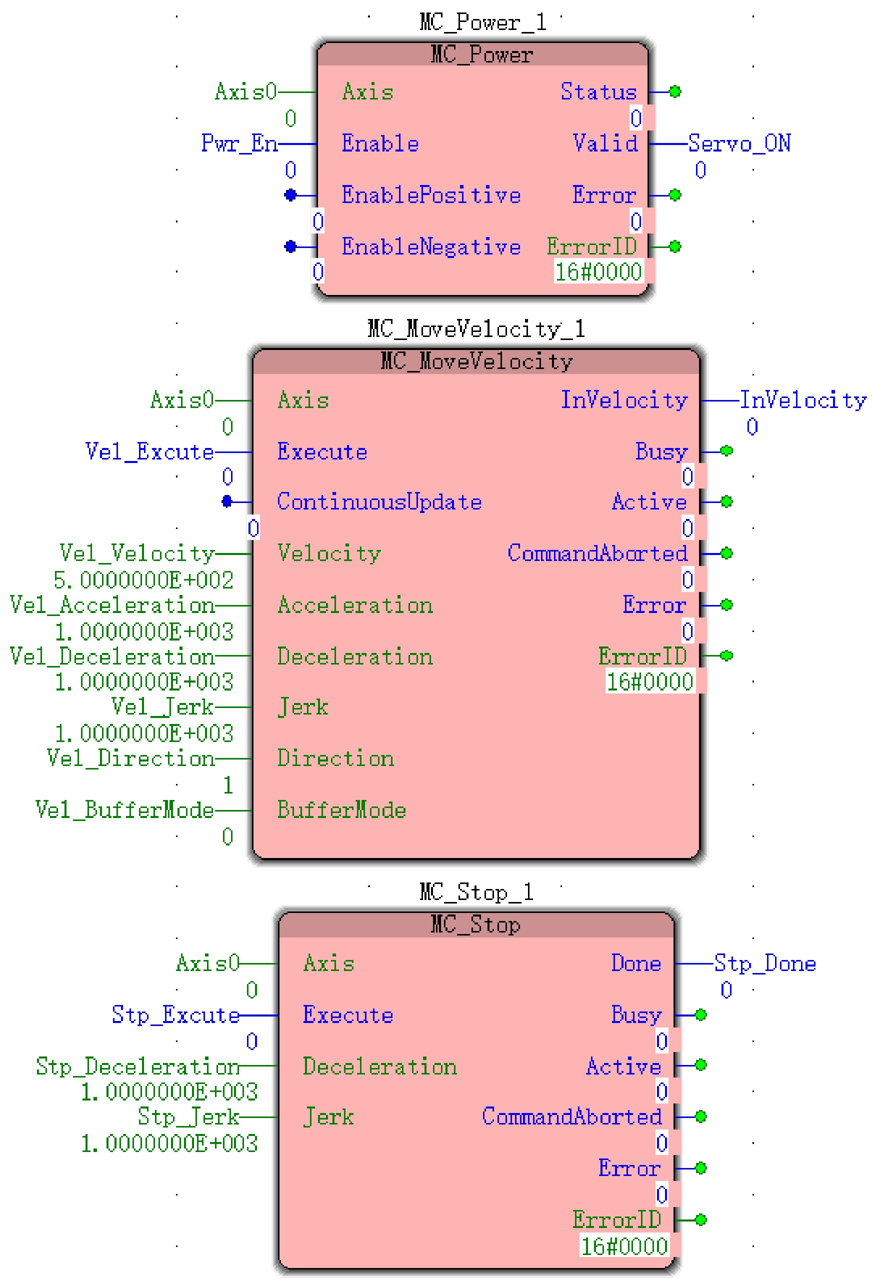
Step 6: Make a bottom-loading project. In the toolbar click on the make , the confirmation process is correct, click download programs, then click on cold start , after a cold start is successful, the status is displayeDAs a running state .



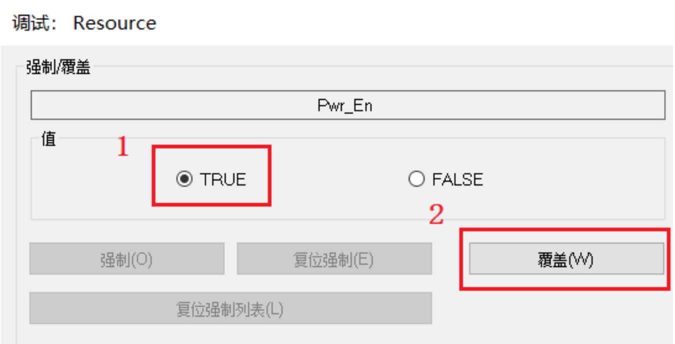
Step 7: Program debugging. Single- click online icon 

on the toolbar can program debugging and monitoring. Online monitoring as shown below

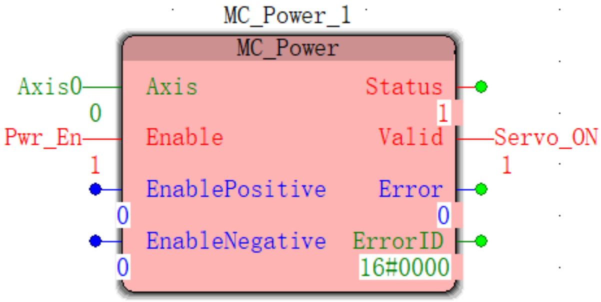




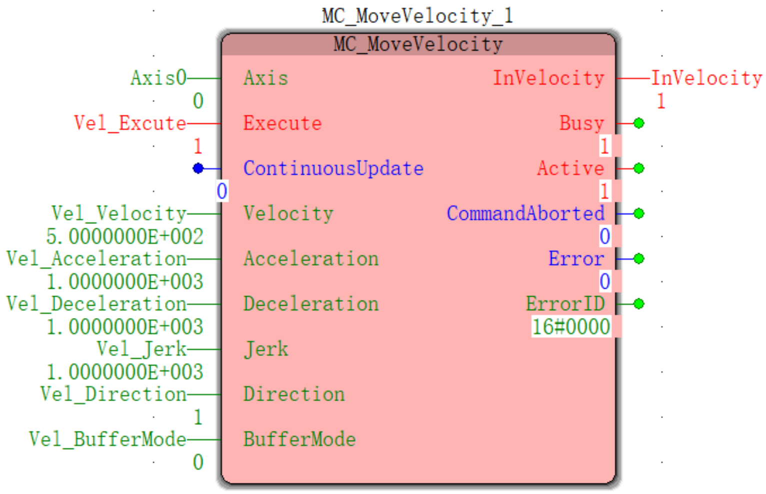

Debug 1： Double-click the input function bit Pwr_ En to pop up the debug: Resource interface, select the value of the variable Ture， then click overwrite, Pwr_ En will change from False to Ture， as shown in the figure；



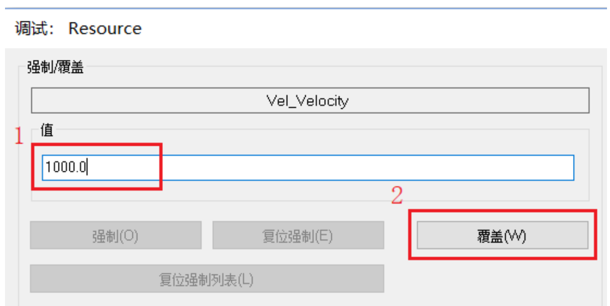

WheNServo _ON changes from False to True， it indicates that axis 0 is enabled successfully, and the servo is enableDAt the same time through the motion controller output.



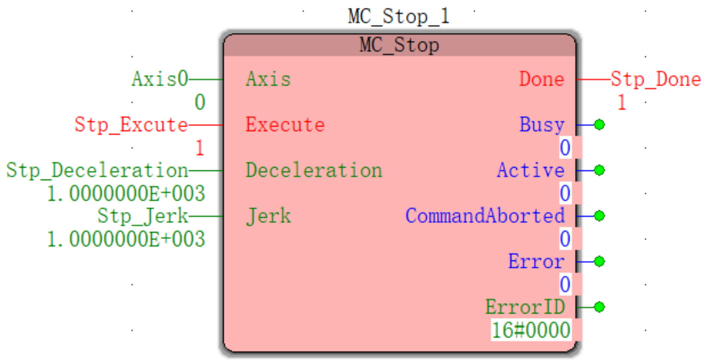

Debug 2 : Similarly, double-click Vel_ Excute to change its value from False to Ture . The controller starts sending analog commands to the servo. The motor starts to accelerate in the positive direction . When InVelocity changes from False to True , the speed reaches the preset. Value 5 00 . 0 ;




Commissioning 3 : Double-click Vel_Velocity , in the pop-up debug window reassigned 1 000 . 0 , click on the cover, and then re-trigger a Vel_Excute the update rate , the motor speed will follow the preset acceleration and deceleration of 5 00 . 0 accelerated to 1 000 . 0 ;

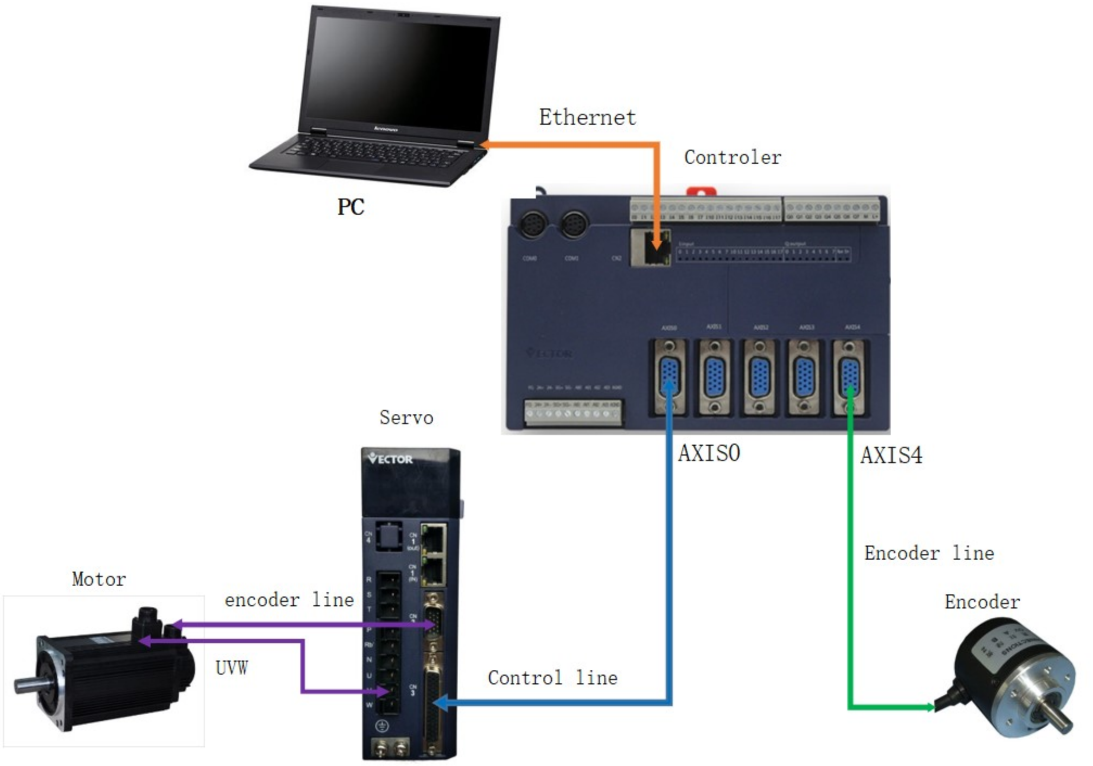



Debug 4 : Double-click Stp _Excute to change its value from False to Ture . The motor will decelerate according to the preset deceleration until it stops. WheNStp _Done changes from False to True , it stops.


MC_Stop_1
MC_Stop
Axis0
0
Stp_Excute
1
Stp_Deceleration
1.0000000E+003
Stp_Jerk
1.0000000E+003
Axis
Execute
Deceleration
Jerk
Done
Busy
0
Active
0
CommandAborted
0
Error
0
ErrorID
16#0000
Stp_Done
1

# 9 .3 controller pulsing motioNServo control ( encoder driveNServo operation)

**1 : Communication and control connection diagram**




According to the above system, the control servo motor runs an electronic gear following the spindle (encoder);

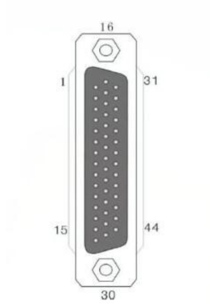
**2 : Set the servo drive parameters**

Wikoda VC type servo drive parameter setting ( Please contact our staff for VB servo drive parameter setting method)

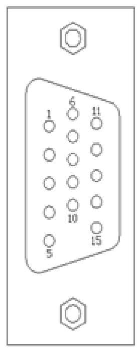
| Function number | Numerical value | description |
|---|---|---|
| P0 2 . 01 | 0 | Position control mode |
| P 03 . 01 | 0 | Position command is derived from external pulse |
| P 03 . 02 | 2 | Command pulse form is AB pulse |
| P 06 . 01 | 1 | The DI 1 function register function is set to: enable drive |

**3 : Control line and encoder line connection**

Wei Keda VC type servo drive is the CN . 3 portions pin definitions

| Pin | signal | description | |
|---|---|---|---|
| 3 1 | X+ | Pulse signal input |  |
| 3 2 | X- | | |
| 3 3 | Y+ | | |
| 3 4 | Y - | | |
| 37 | OA+ | Encoder signal amplification output | |
| 38 | OA- | | |
| 39 | OB+ | | |
| 40 | OB- | | |
| twenty four | DI1 | Servo DI1 input | |
| 1 0 , 2 6 | + 24 V | External DC24V power supply , for DI , DO work use , Remarks: 25 , 26 feet for NPN / PNP Jumper choose to use | |
| 9 , 2 5 | COM | | |

Wilcoda motion controller AXIS port part pin definition

| Pin | signal | description | |
|---|---|---|---|
| 1 | X+ | Pulse signal output |  |
| 2 | X - | | |
| 3 | Y + | | |
| 4 | Y - | | |
| 6 | A+ | Encoder signal input | |
| 7 | A- | | |
| 8 | B+ | | |
| 9 | B- | | |
| 13 | +5 | DC 5V output | |
| 15 | GND | GND | |

Encoder piNSection definition

| Pin | signal | description | |
|---|---|---|---|
| 1 | AO+ | Encoder signal outputs an |  |
| 2 | AO- | | |
| 3 | BO+ | | |
| 4 | BO- | | |
| 5 | +5V | 5V input | |
| 6 | GND | GND | |

Control line connection

| Servo (CN3) | | Controler (AXIS0) | |
|---|---|---|---|
| 31 | X+ | 1 | X+ |
| 32 | X- | 2 | X- |
| 33 | Y+ | 3 | Y+ |
| 34 | Y- | 4 | Y- |
| 37 | OA+ | 6 | A+ |
| 38 | OA- | 7 | A- |
| 39 | OB+ | 8 | B+ |
| 40 | OB- | 9 | B- |
| 26 | +24V jumper | | |
| 27 | SW-DI | | |
| 9 | COM | 0V | External |
| 10 | +24V | +24V | External |
| 24 | DI1 | Q0 | Control external terminal output |

Encoder cable connection

| encoder | | encoder line (AXIS4) | |
|---|---|---|---|
| 1 | OA+ | 6 | A+ |
| 2 | OA- | 7 | A- |
| 3 | OB+ | 8 | B+ |
| 4 | OB- | 9 | B- |
| 5 | 5V+ | 13 | 5V+ |
| 6 | GND | 15 | GND |

📖·Description :

1. In order to control the servo driver enable by controlling the output of the motion controller Q 0 , it is necessary to supply ADC24V power supply to the servo driver to supply power to the servo DI ;

2 , DIx signal type ( NPN/PNP ) selection : SW-DI ( 27 feet ) and +24V ( 26 feet) are shorted to NPN ;

3. In order to reduce the interference, the differential signals ( OA+ →·A+ ) and ( OA- →·A- ), ( OB+ →·B + ) and ( OB- →·B - ) and XY pulse signals are respectively connected by twisted pairs, and the outer casing is grounded. .

**4 : PrograMCreation and debugging**

**Notice**

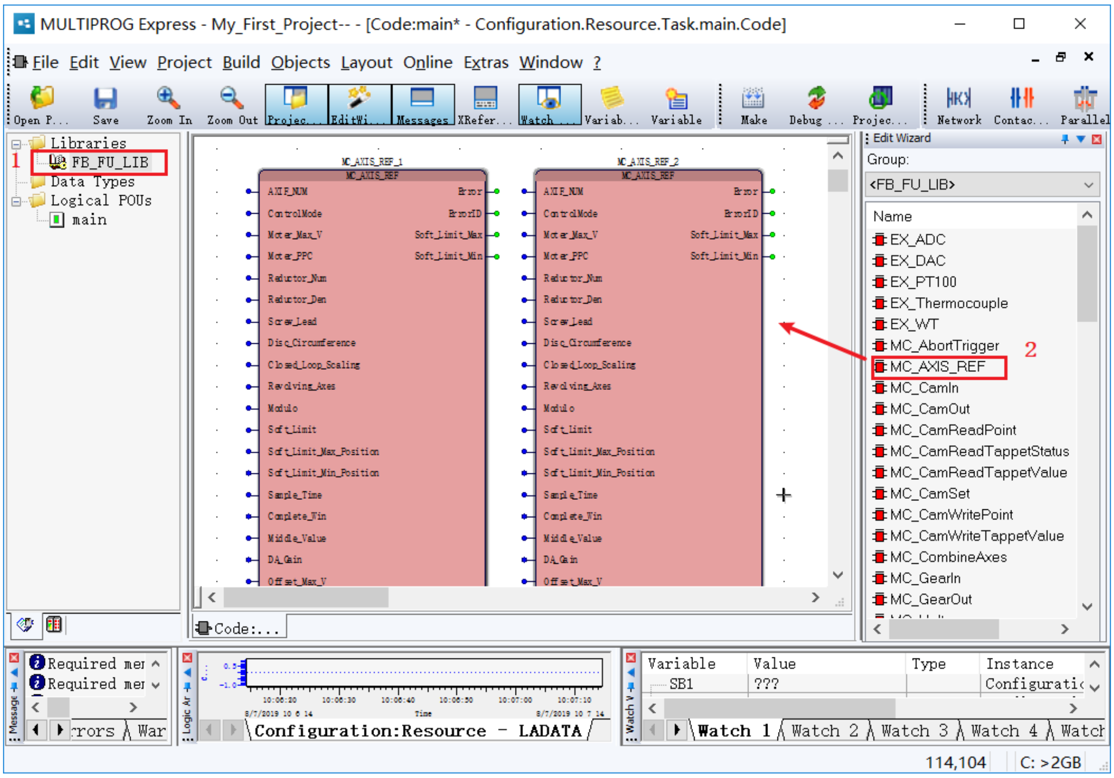
After the above PC and the motion controller communicate successfully, the system is set up, the servo parameter setting is completed, the next step is to start programming to control the servo motor to run and stop; before the programming, the default user has read the "Chapter 11 Motion Command" including: 11.1 insert FB_FU_LIB motion control library , 11. The 2 motion commands , 11.3 motion instructions basics and . 1 . 1 . . 4 uniaxially instructions .

**( 1 ) Project creation and configuration**

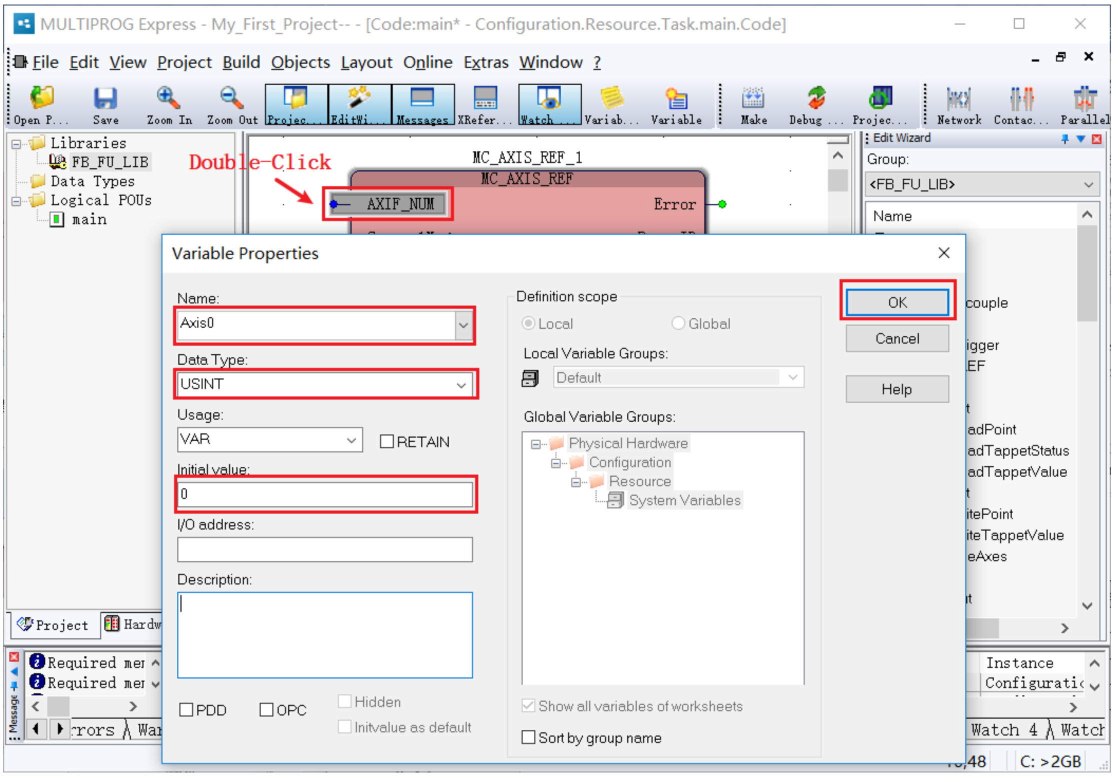
Reference "on Qi Zhang works of creation and configuration." Follow the steps to complete : 7 .1 create projects →·7 .3 physical communication parameters →·7 . 4 IO configuration , which will not be repeated herein.

**( 2 ) Writing a program**

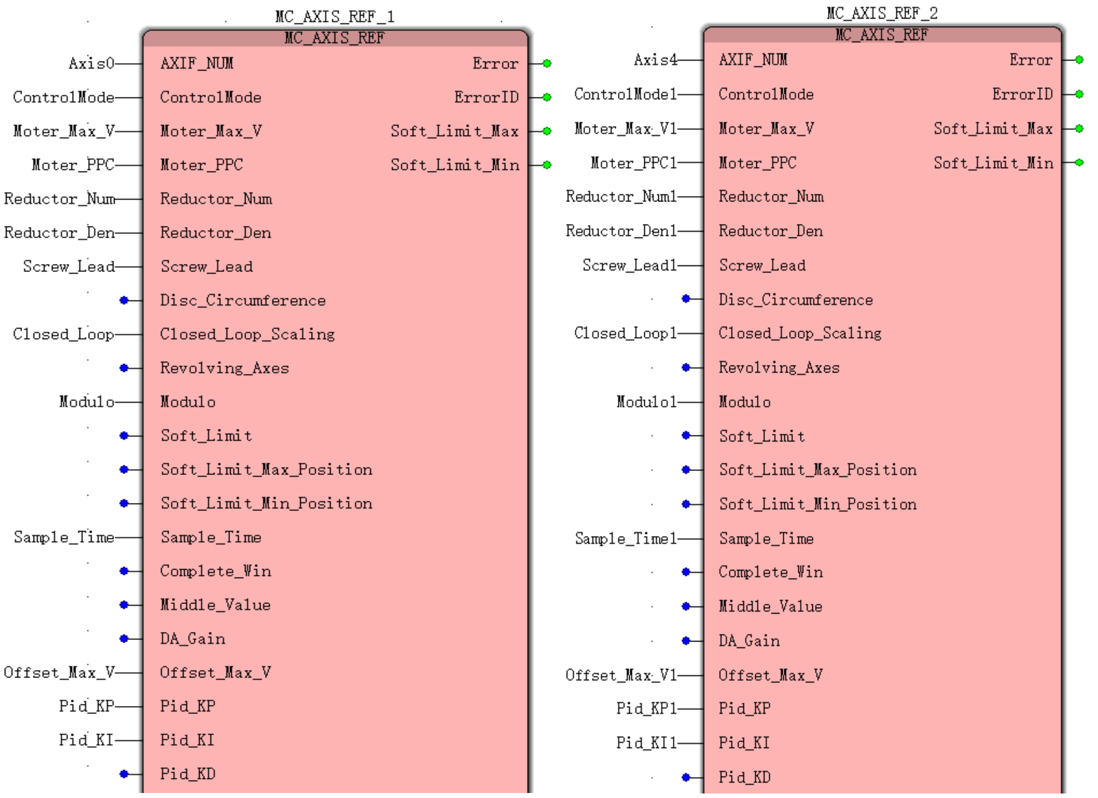
Step 1: Enter the programming interface, select the " MC _ AXIS_REF " block in the FB_FU_LIB motion control library , hold down the left mouse button and drag to the programming interface, then let go, the module's properties will pop up, you need to name the module. in general keep the default, click "OK", as shown, insert two "FIG the MC _ AXIS_REF- 'block ;




Step 2: Double-click the module input pin (blue dot), and the "Variable Properties" box will pop up to define the variable name, data type, usage, initial value, etc., as shown in the figure ;




The user caNSimply fill in the parameters that are not allowed by the axis parameters. You can add settings according to the function requirements. Refer to [11.3.3 MC_AXIS_REF ( Axis Parameter Setting )] . After the example is added , the following figure is shown ;

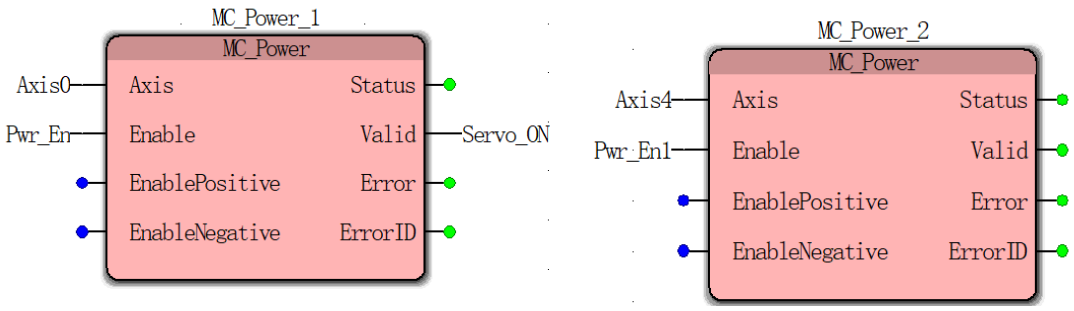



Variable name anDAttribute

| variable name | type of data | Initial value |
|---|---|---|

| | | |
|---|---|---|
| MC_AXIS_REF_1 | MC_AXIS_REF | |
| Axis0 | USINT | 0 |
| ControlMode | INT | 1 |
| Moter_Max_V | DINT | 3000 |
| Moter_PPC | DINT | 10000 |
| Reductor_Num | LREAL | 1.0 |
| Reductor_Den | LREAL | 1.0 |
| Screw_Lead | LREAL | 60.0 |
| Closed_Loop | LREAL | 1.0 |
| Modulo | LREAL | 3 60.0 |
| Sample_Time | WORD | 20 |
| Offset_Max_V | DINT | 200 |
| Pid_KP | DINT | 80 |
| Pid_KI | DINT | 0 |
| MC_AXIS_REF_2 | MC_AXIS_REF | |
| Axis4 | USINT | 4 |
| ControlMode1 | INT | 1 |
| Moter_Max_V1 | DINT | 3000 |
| Moter_PPC1 | DINT | 10000 |
| Reductor_Num1 | LREAL | 1.0 |
| Reductor_Den1 | LREAL | 1.0 |
| Screw_Lead1 | LREAL | 60.0 |
| Closed_Loop1 | LREAL | 1.0 |
| Modulo1 | LREAL | 3 60.0 |
| Sample_Time1 | WORD | 20 |
| Offset_Max_V1 | DINT | 200 |
| Pid_KP1 | DINT | 80 |
| Pid_KI1 | DINT | 0 |

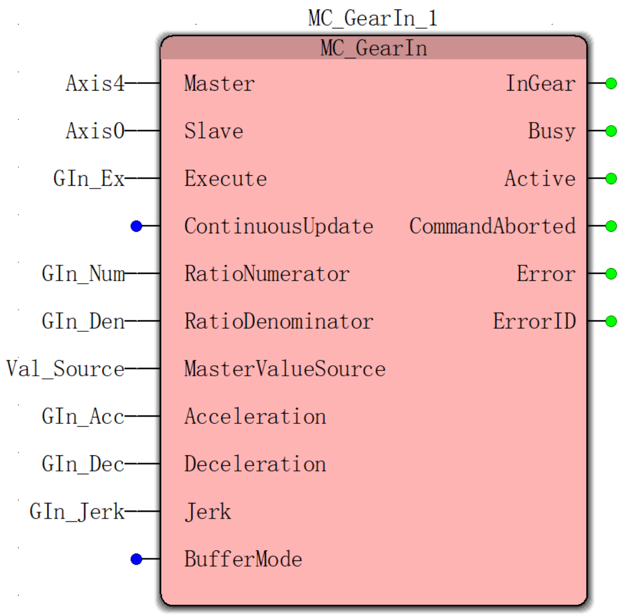
Step3: Add Similarly two " the MC_Power " instruction as to enable the motion controller, while servo axis ( the AXIS 0 ) by Se RV O _ON, the ( the I / O Address: % QX0.0 ) output control causes the servo drive Yes , refer to 1 1.4.1 MC_Power ( Enable Command ) ;




Variable name anDAttribute

| variable name | type of data | Initial value | address |
|---|---|---|---|
| MC_Power_1 | MC_Power | | |
| Axis0 | USINT | 0 | |
| Pwr_En | BOOL | | |
| Servo_ON | BOOL | | %QX0.0 |
| MC_Power_ 2 | MC_Power | | |
| Axis 4 | USINT | 4 | |
| Pwr_En 1 | BOOL | | |

Step4: similarly add " MC_ GearIn " ( electronic gear coupling command ) , used to control the servo follower encoder axis electronic gear movement , reference 1 1.4.2 MC_ GearIn ( electronic gear coupling command ) ;

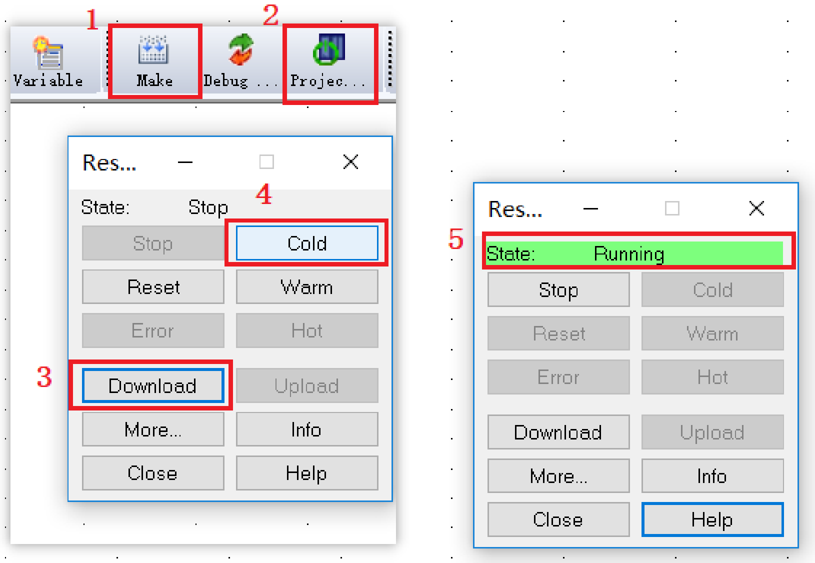



Variable name anDAttribute

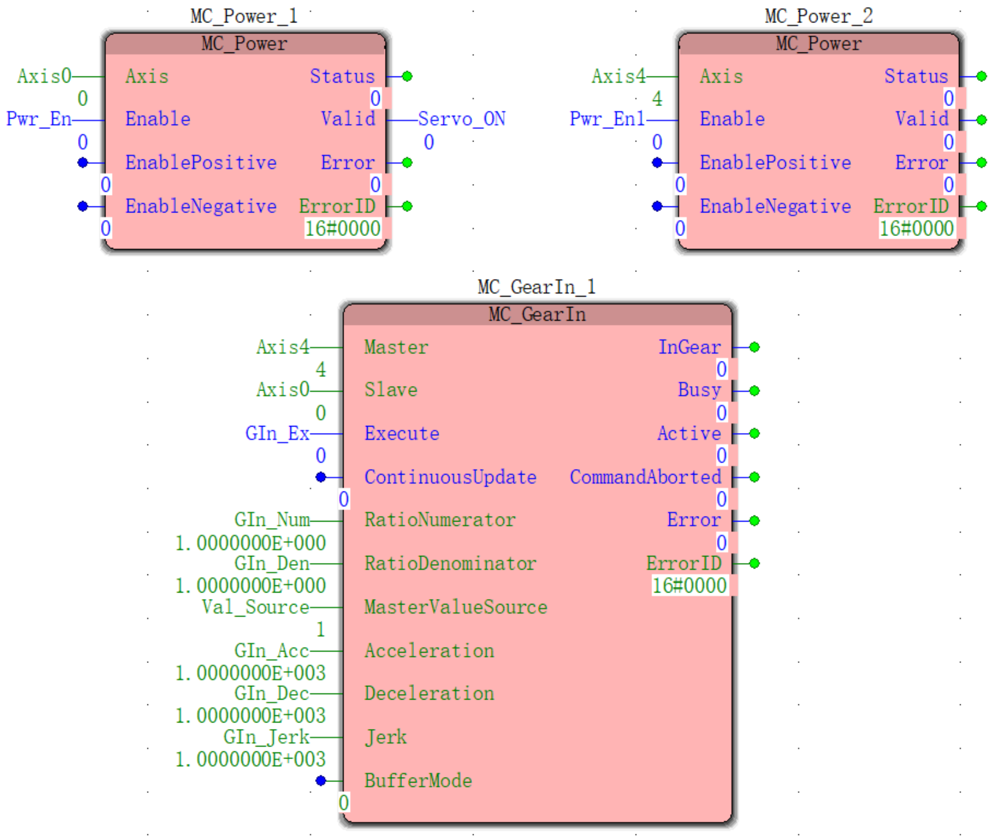
| variable name | type of data | Initial value |
|---|---|---|
| MC_GearIn_1 | MC_GearIn | |
| A xis4 | U SINT | 4 |
| A xis0 | U SINT | 0 |
| GIn_Ex | BOOL | |
| GIn_Num | LREAL | 1.0 |
| GIn_Den | LREAL | 1.0 |
| Val_Source | INT | 1 |
| GIn_Acc | LREAL | 1000.0 |
| GIn_Dec | LREAL | 1000.0 |
| GIn_Jerk | LREAL | 1000.0 |

At this point, the programming is complete.

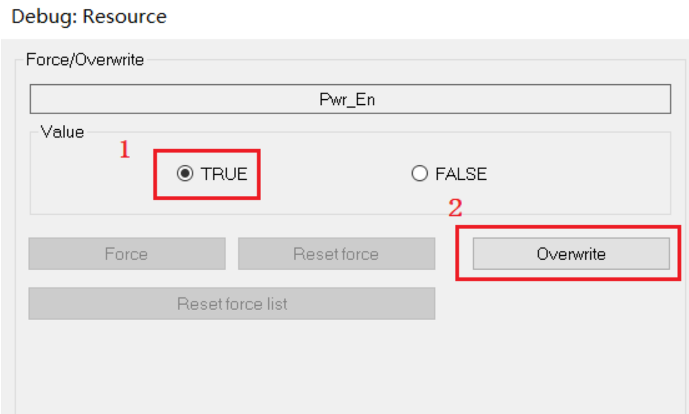
Step5: making Bottoms project. Click on the toolbar to confirm that the program is correct, click on the download program, and then click on the cold start. After the cold start is successful, the status is displayeDAs the running status.



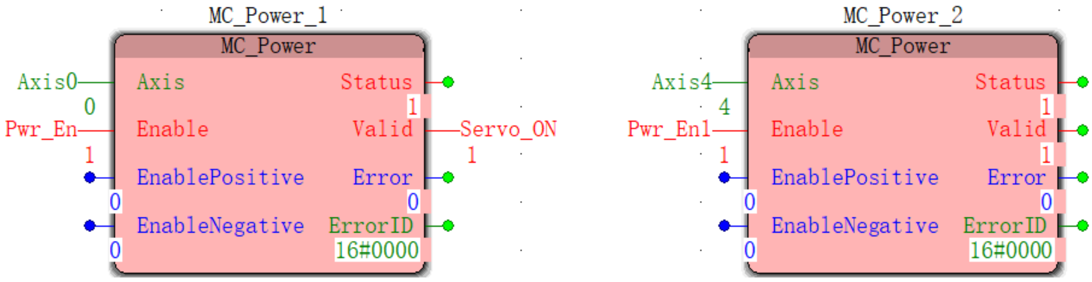

Step6: debugging. Single- click online icon on the toolbar can program debugging and monitoring , monitoring online as shown below ;



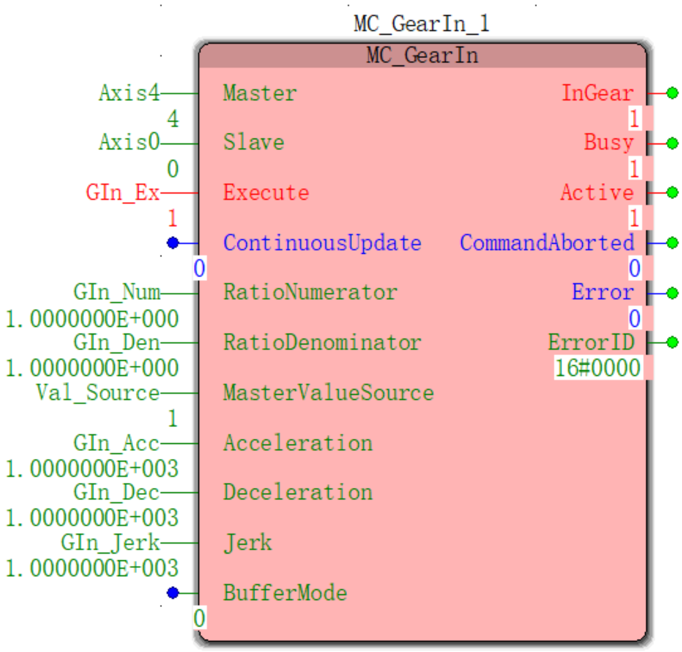

Debugging 1 : Double-click input function bit Pwr_ En , pop-up commissioning: Resource interface, variable values select Ture , then click on the cover, Pwr_ En will False become Ture ; the same token the Pwr_ En1 value becomes Ture ;




WheNServo _ON, the a False becomes Ture , the motion controller describeDAxes is enabled successful, and by Q 0 output while allowing the servo enabled;



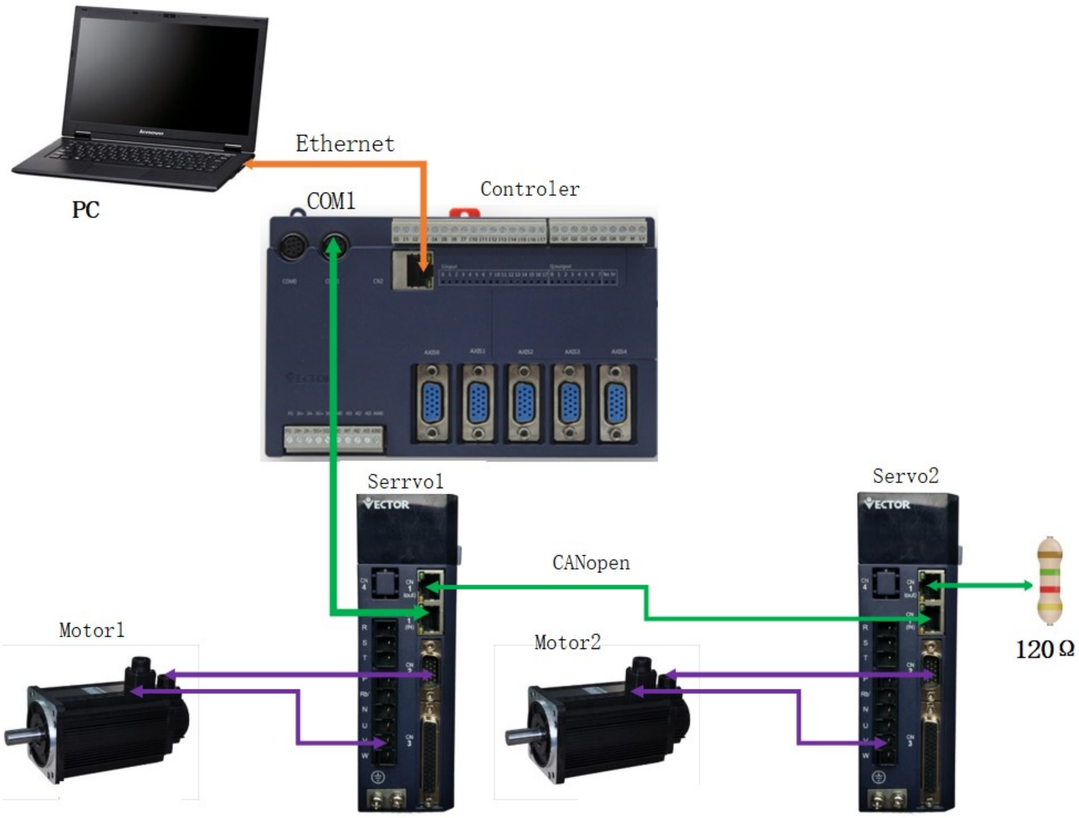

Debug 2 : Similarly, double-click G In_Ex to change its value from False to True , so that controller A XIS0 anDAXIS 4 establish electronic gear relationship, AXIS 4 is the main axis, AXIS 0 is the slave axis, and the gear ratio is 1 :1 ;




Commissioning 3 : At this point, turn the spindle (encoder) and the slave axis (servo axis) will follow the spindle in accordance with the 1: 1 electronic gear ratio.

# 9 .4 controller CANopen through inquiry mode control servo motion ( two -axis motion)

**1 : Communication and control connection diagram**




According to the above system, the motion controller controls the operation of the servo motor through the CANopen communication mode , the motor 1 moves the speed command, and the motor 2 takes the relative displacement command .

**2 : Set the servo drive parameters**

Wikota CANopeNServo drive 1 parameter setting

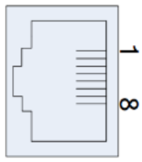
| Function number | Numerical value | description |
|---|---|---|
| P0 8 . 40 | 80 0 | CAN baud rate |
| P 08 . 41 | 1 | CAN node number |

Servo drive 2 parameter setting

| Function number | Numerical value | description |
|---|---|---|
| P0 8 . 40 | 80 0 | CAN baud rate |
| P 08 . 41 | 2 | CAN node number |

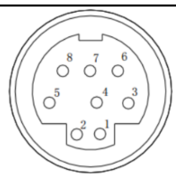
**3 : CANopen network communication line connection**

Wei Keda the CAN Open type servo drive is the CN . 1 part of the pin definitions

| Pin | signal | description | |
|---|---|---|---|
| 1 | CANH | High signal of CAN bus |  |
| 2 | CANL | Low signal of CAN bus | |
| 3 | GND | Power ground | |



Wikoda motion controller COM 1 port part pin definition

| Pin | signal | description | |
|---|---|---|---|
| 6 | CANL | Low signal of CAN bus |  |
| 7 | CANH | High signal of CAN bus | |
| 8 | GND | Power ground | |



CANopen network communication line connection

| Controler (COM1) | | | Servo1 (CN1) | |
|---|---|---|---|---|
| 6 | CANL | ↔ | 1 | CANL |
| 7 | CANH | ↔ | 2 | CANH |
| 8 | GND | ↔ | 3 | GND |

| Servo1 (CN1) | | | Servo2 (CN1) | |
|---|---|---|---|---|
| 1 | CANL | ↔ | 1 | CANL |
| 2 | CANH | ↔ | 2 | CANH |
| 3 | GND | ↔ | 3 | GND |

| Servo2 (CN1) | | | |
|---|---|---|---|
| 1 | CANL | ↔ | 120Ω resistance |
| 2 | CANH | ↔ | |
| 3 | GND | | |

📖·Description :

. 1 , The CANopen next communication mode, the program uses the MC _Power (Enable command ) to make the same time to the module, via the communication will enable the servo driver, so no additional access points to control the servo drive output enable ;

2 , In order to enhance CANopeNStability of communication, CANopen terminal needs to access the network bus 120 [Omega] terminal resistor.

**4 : PrograMCreation and debugging**

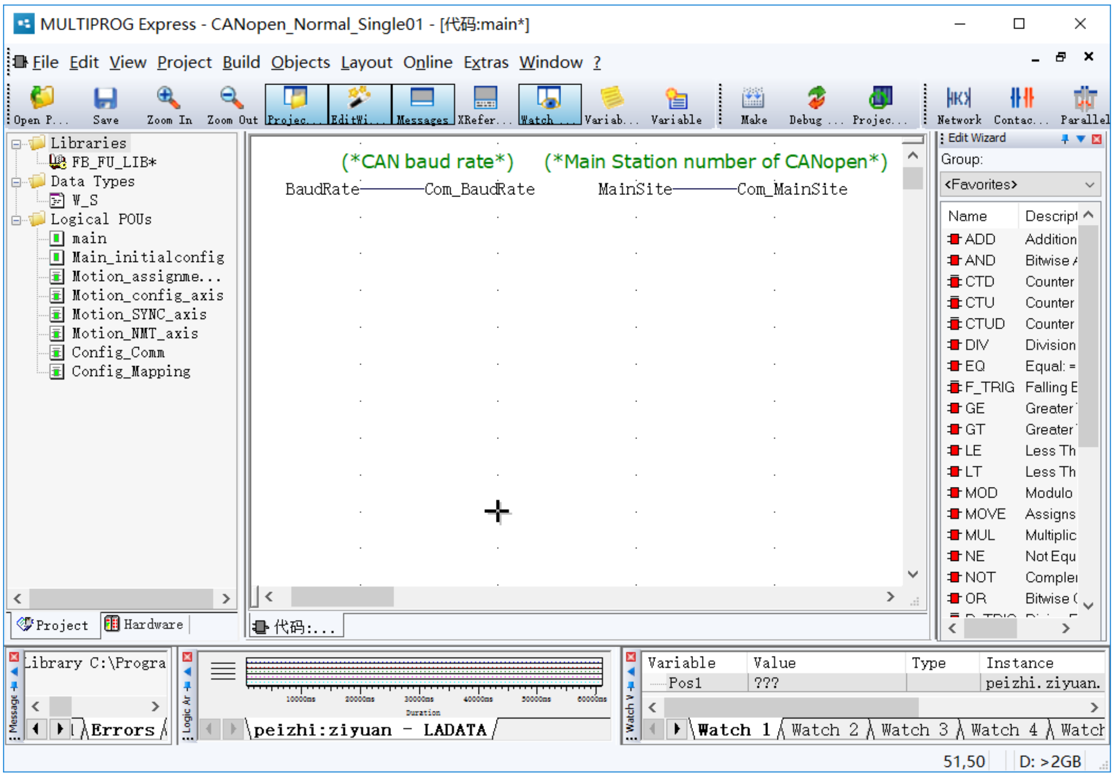
**Notice**

After the above PC and the motion controller communicate successfully, the system is set up, the servo parameter setting is completed, the next step is to start programming to control the servo motor to run and stop; before the programming, the default user has read "Chapter 11 Motion Command" includes: 11.1 insert FB_FU_LIB motion control library , 11. The 2 motion commands , 11.3 motion instructions basics and . 1 . 1 . . 4 uniaxial instruction . "

**( 1 ) Project creation and configuration**

For the convenience of use, our company has equipped the user with a template project for CANopen communication configuration. Users can go to the official website to downloaDAnd directly program on the basis of the template project. ( The template default configuration of a shaft, can be configured up . 1 . 6 axes, may be addeDAs required in the configuration template )

Once you have downloaded the template, extract the open, in the following figure , the reference to Chapter VII of the creation and configuration of the project to complete the PC to communicate with the motion controller, and reference 11.1 insert FB_FU_LIB motion control library complete adding a library, which will not be repeated herein .

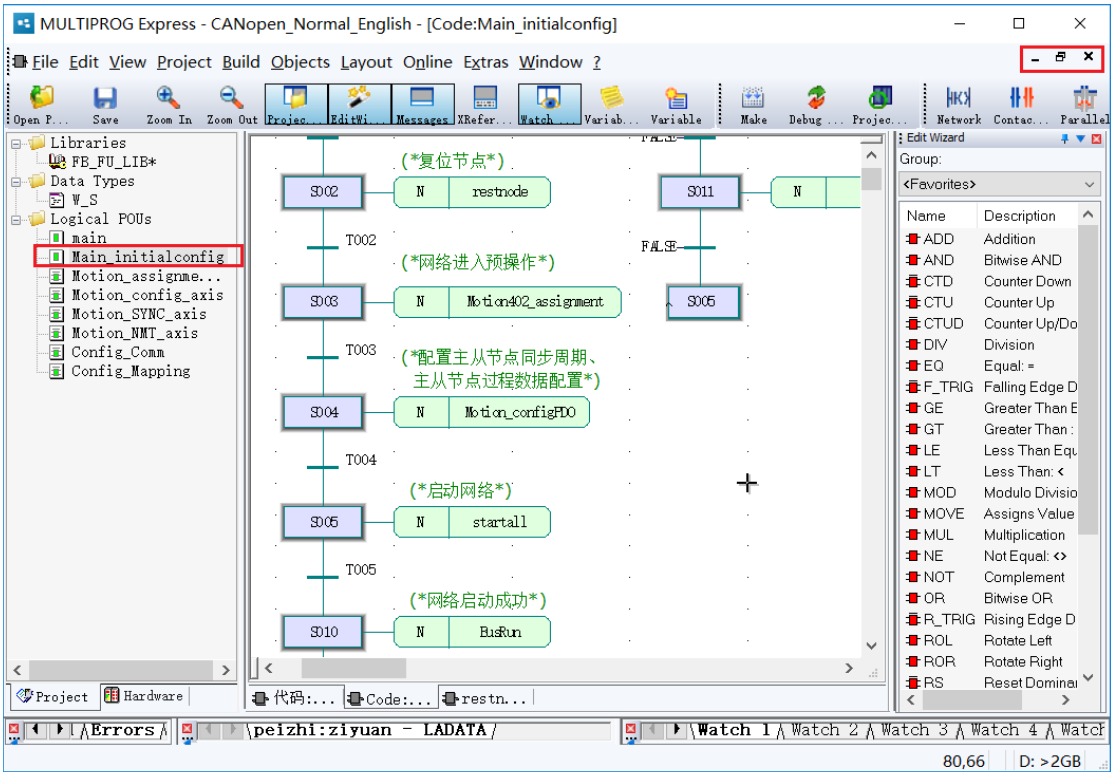



**Note:** The template default CANopen master station number is 1 8 and the CAN bus baud rate is 8 00 (corresponding to the CAN bus baud rate set by the servo driver P 08.40 ), which can be modified by modifying the initial value of the BaudRate .

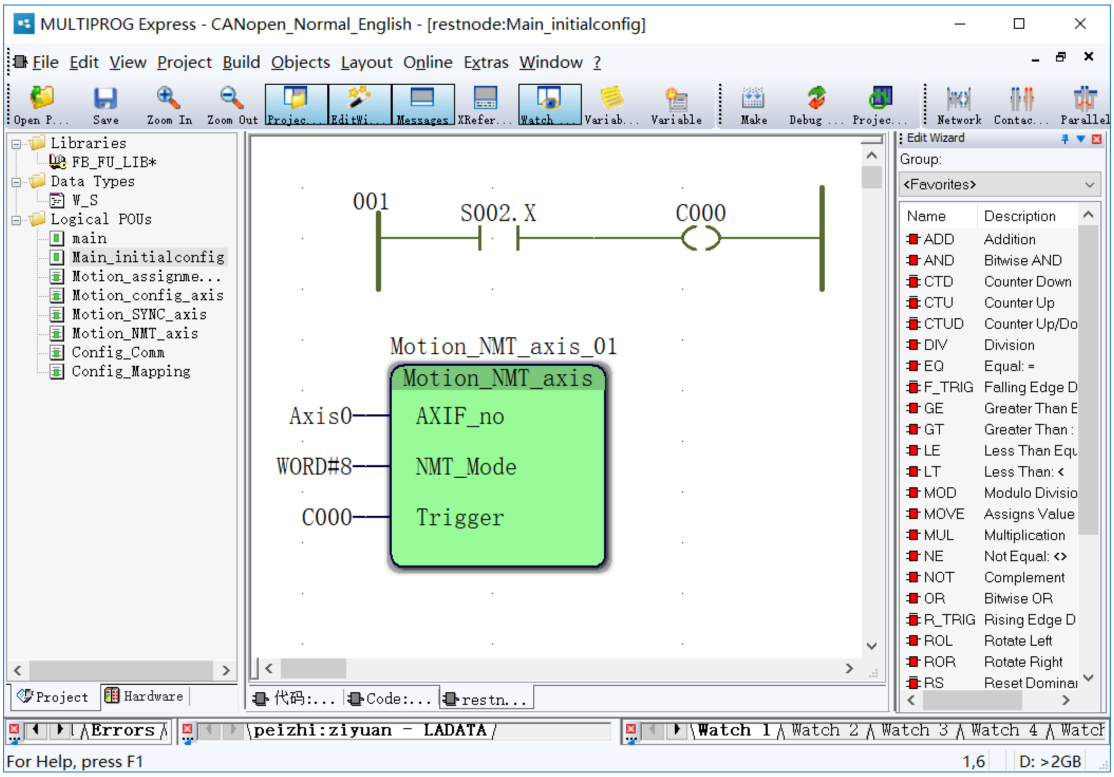
**( 2 ) node configuration**

Step 1: Since the template configures one axis by default (node number is 1 ), and two axes are used in this case, we need to manually adDAnother axis in the configuration template (node number is set to 2 ). Double-click " Main _initialconfig " under "Logical POU " in the project tree to open the configuration flow program of CANopen communication (you can close the window management button in the upper right corner when you need to close), as shown below.

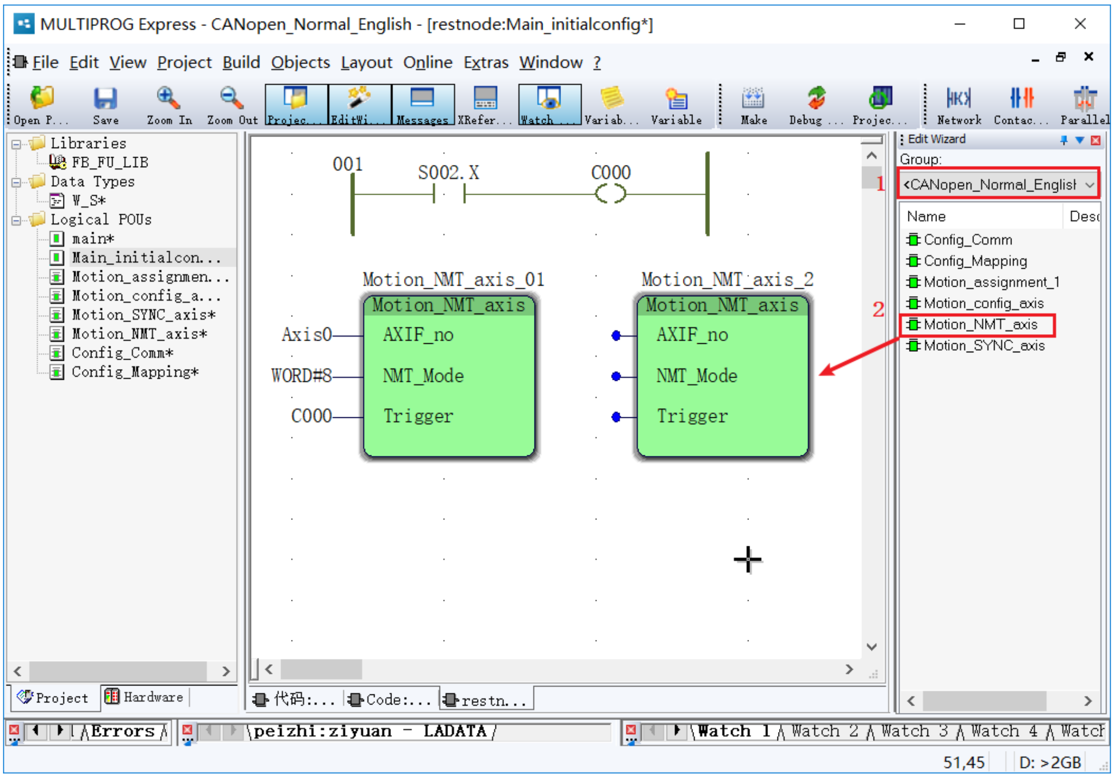
The maiNSteps of the configuration can be seen in the figure: master-slave node reset ( R estnode ), master-slave node enters pre-operation mode ( Motion 402 _ assignment ), configures master-slave node synchronization cycle and master-slave node process data configuration ( M otion_configPDO ), start the bus ( S tartall ) , the detailed steps define the reference CANopen related instructions in Annex IV ;




Step 2: Double-click the " R & lt estnode ", to open the reset node process, there can be seen through the process module after the second package "M otion_NMT_axis " , as shown below, which defines the reference position input function Annex IV . 4 . . 1 . 2 Main Reset from the node , the initial value of axis number Axis 0 is 0, which is used to reset the master station under CAN open network and node 1 (node number = axis number + 1 ). Therefore, we need to adDAnother module. , resetting node 2;



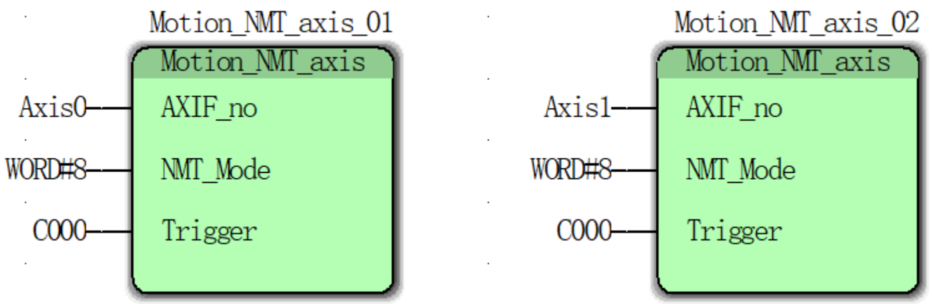
Step 3: In the user-defined library, find the "Motion_NMT_axis" block, hold down the left mouse button and drag it to the programming interface, then let go, then the module's properties will pop up, you need to name the module, generally keep the default. , click "OK" , as shown below;



Step 4: Double-click the module input pin (blue dot), and the "Variable Properties" box

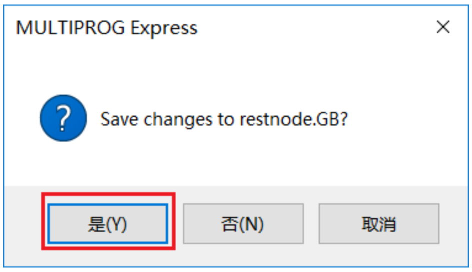
will pop up to define the variable name and data.

Type, usage, the initial value and the like; input pin bit fill in the following FIGS. , Except that ADifferent number axis (axis number of fill at the Axis 1 , the initial value is 1 , the representative node 2 ), the remaining variable fill the axis 0 same ;



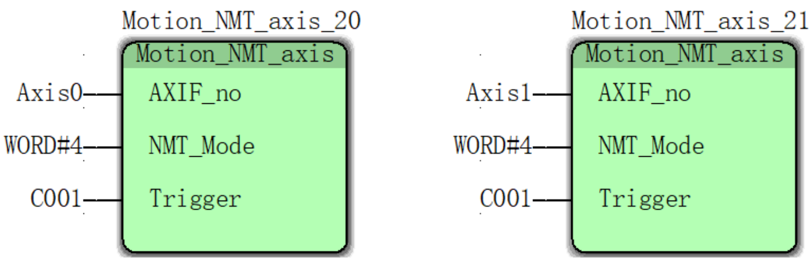

| variable name | type of data | Initial value |
|---|---|---|
| Motion_NMT_axis_02 | Motion_NMT_axis | |
| Axis 1 | USINT | 1 |
| W ORD#8 | | |
| C000 | BOOL | |

After filling in, click to close the window, pop up the dialog box to save it, select "Yes", as shown below , the reset node configuration ( r esetnode ) process of node 2 has been completed ;

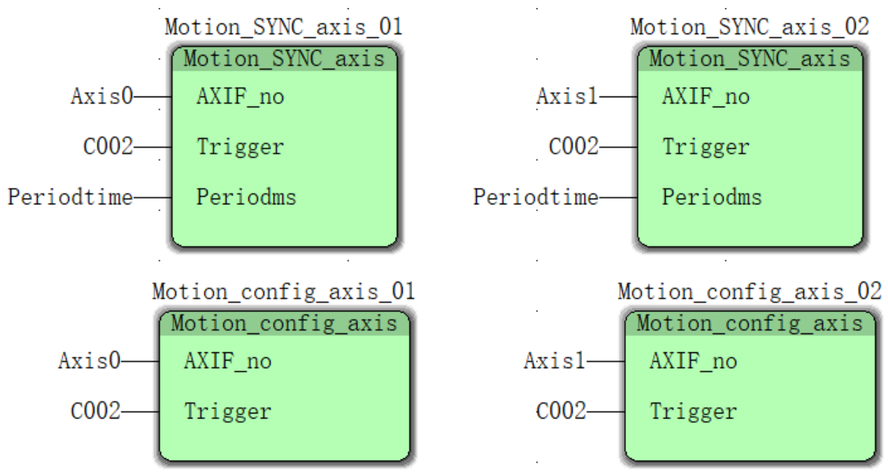



Step 5: After completing the reset node process, return to " Main _initialconfig " and repeat steps 2, 3, and 4 to complete the remaining configuration for node 2 , including : master-slave node enters pre-operation mode ( Motion 402 _ assignment ), configuration Master-slave node synchronization cycle and master-slave process data configuration (( M otion_configPDO )), start bus ( S tartall ). The added modules have the same axis number ( Axis 1 ), and the remaining variables are filled in the same way as axis 0 ;

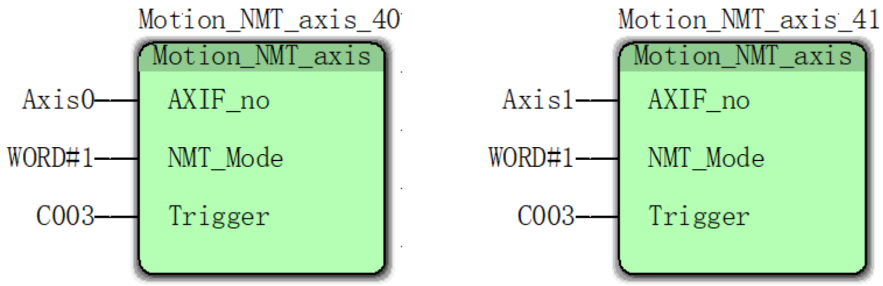
Pre-operation mode ( Motion 402 _ assignment ):




Configure the master-slave node synchronization period ( M otion_configPDO ):




Start bus ( S tartall ):

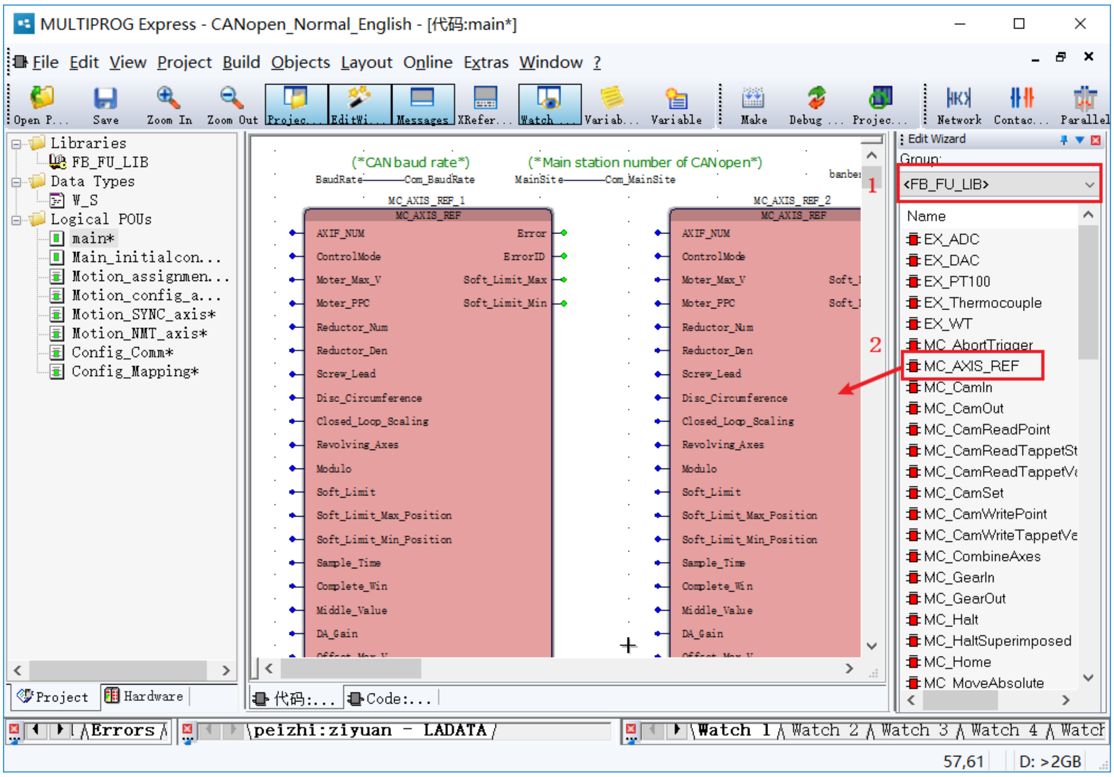



Step 6: After completing all the process configuration, click on the production and confirm that it is correct. At this point, the CANopen communication configuration of node 2 ( Axis 1 ) has been completed ;

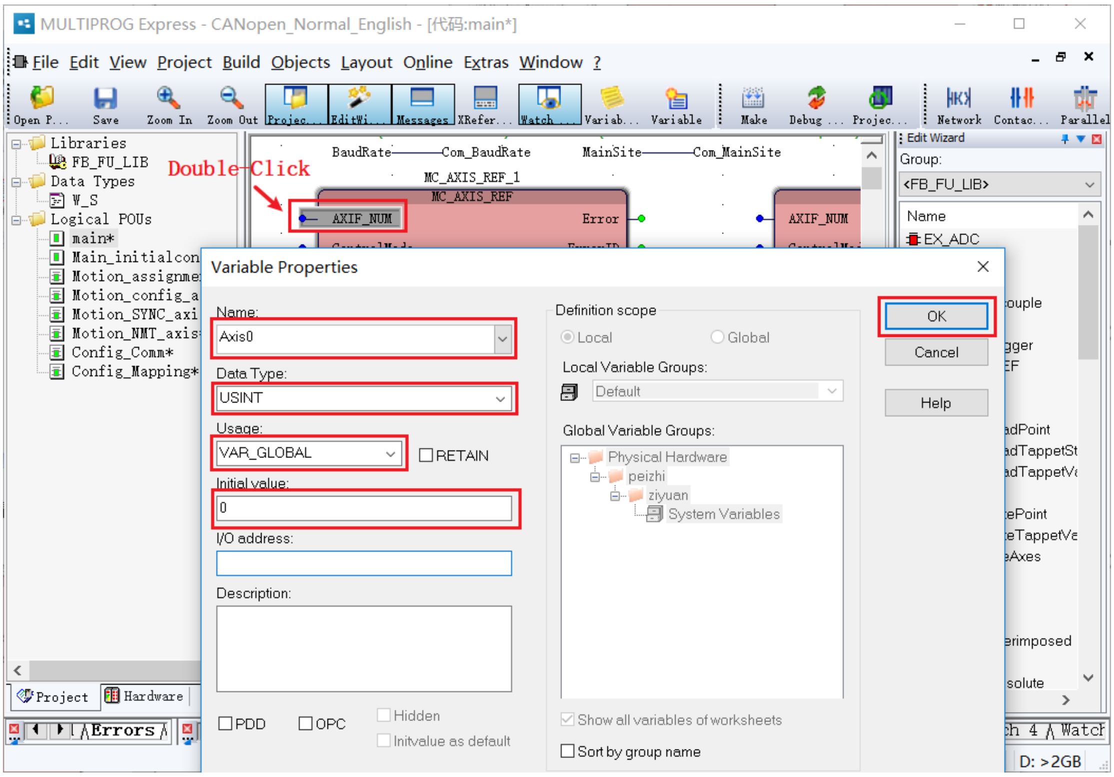
Step 7: Double-click the project tree "logic POU under" "the main ", return to the main program interface, you can write a program to start the movement .

**( 3 ) Writing a program**

Step1: entering the " main " programming interface, in FB_FU_LI selected motion control library " the MC _ AXIS_REF- " die block, hold the left mouse drag to the programming interface, and then let go, the module will pop The properties requires the module name, generally keep the default, click "OK", as shown, insert two " the MC _ AXIS_REF- " die block ;

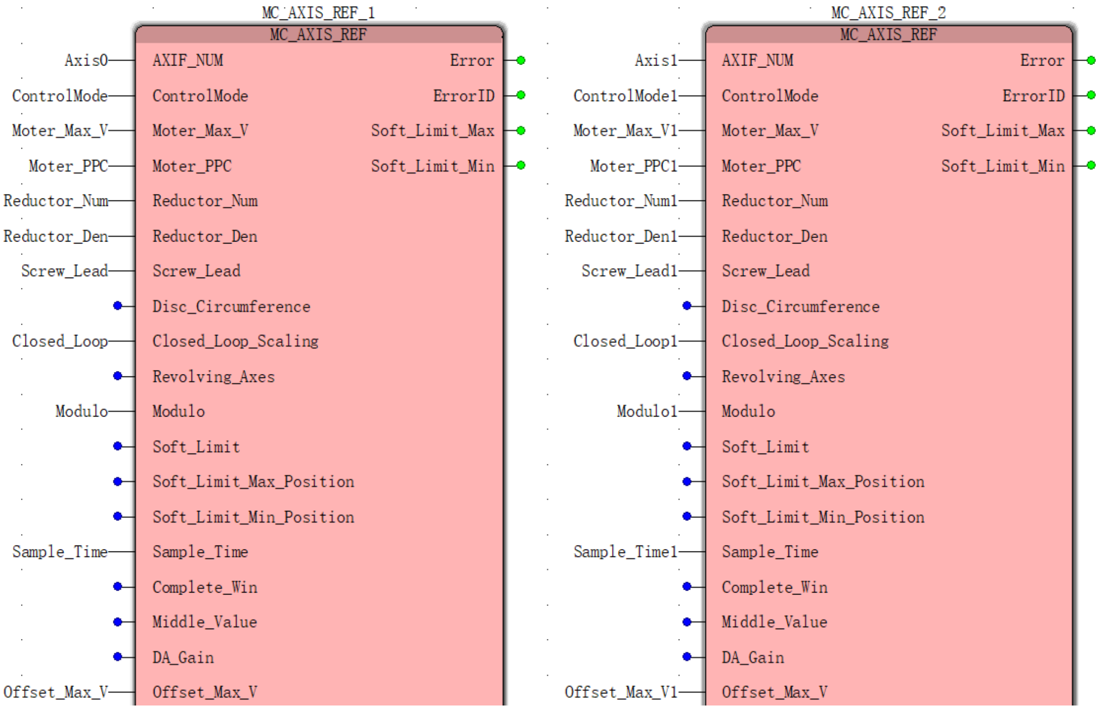



Step 2: Double-click the module input pin (blue dot), then pop up the "Variable Properties" box, define the variable name, data type, usage, initial value, etc. After filling out, click OK , as shown in the figure ;




The user caNSimply fill in the parameters that are not allowed by the axis parameters. You can add settings according to the function requirements. Refer to 11.3.3 MC_AXIS_REF ( Axis

Parameter Setting ) . After the example is added , the following figure is shown ;

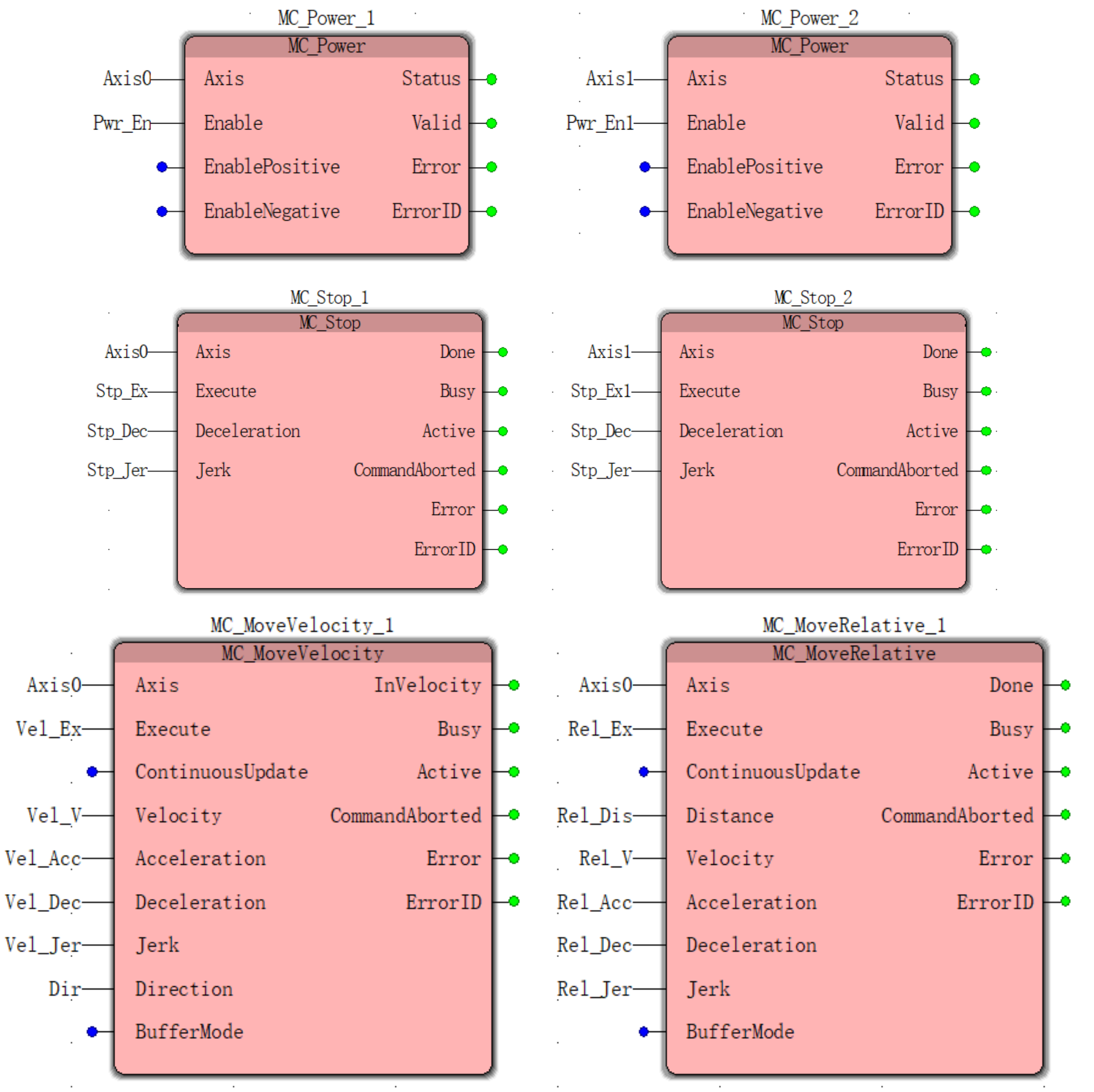



Variable name anDAttribute

| variable name | type of data | Initial value |
|---|---|---|
| MC_AXIS_REF_1 | MC_AXIS_REF | |
| Axis0 | USINT | 0 |
| ControlMode | INT | 2 |
| Moter_Max_V | DINT | 3000 |
| Moter_PPC | DINT | 10000 |
| Reductor_Num | LREAL | 1.0 |
| Reductor_Den | LREAL | 1.0 |
| Screw_Lead | LREAL | 60.0 |
| Closed_Loop | LREAL | 1.0 |
| Modulo | LREAL | 3 60.0 |
| Sample_Time | WORD | 20 |
| Offset_Max_V | DINT | 200 |
| MC_AXIS_REF_2 | MC_AXIS_REF | |
| Axis 1 | USINT | 1 |
| ControlMode 1 | INT | 2 |
| Moter_Max_V 1 | DINT | 3000 |
| Moter_PPC 1 | DINT | 10000 |
| Reductor_Num1 | LREAL | 1.0 |
| Reductor_Den1 | LREAL | 1.0 |
| Screw_Lead 1 | LREAL | 60.0 |
| Closed_Loop 1 | LREAL | 1.0 |

| | | |
|---|---|---|
| Modulo 1 | LREAL | 3 60.0 |
| Sample_Time 1 | WORD | 20 |
| Offset_Max_V 1 | DINT | 200 |

Step3: the manner describeDAbove, were added two " the MC _Power " module, two " MC_ the Stop " module, a " the MC _MoveVelocity " module, a " the MC _MoveRelative " module, variable names and their properties as shown below , the reference 1 1.4.1 MC_Power ( Enable Command ), 1 1 . 4 . 2 MC_ MoveVelocity (speed command) , 1 1.4.3 MC_MoveRelative (relative displacement command) , 1 1 . 4 . 10 MC_Stop (stop command) ;



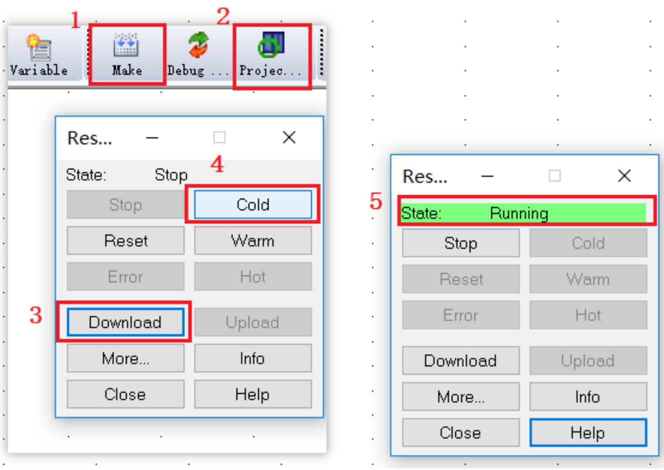

Variable name anDAttribute

| variable name | type of data | Initial value |
|---|---|---|
| MC_Power_1 | MC_Power | |
| Pwr_En | BOOL | |
| MC_Power_2 | MC_Power | |
| Pwr_En1 | BOOL | |
| MC_Stop_1 | MC_Stop | |

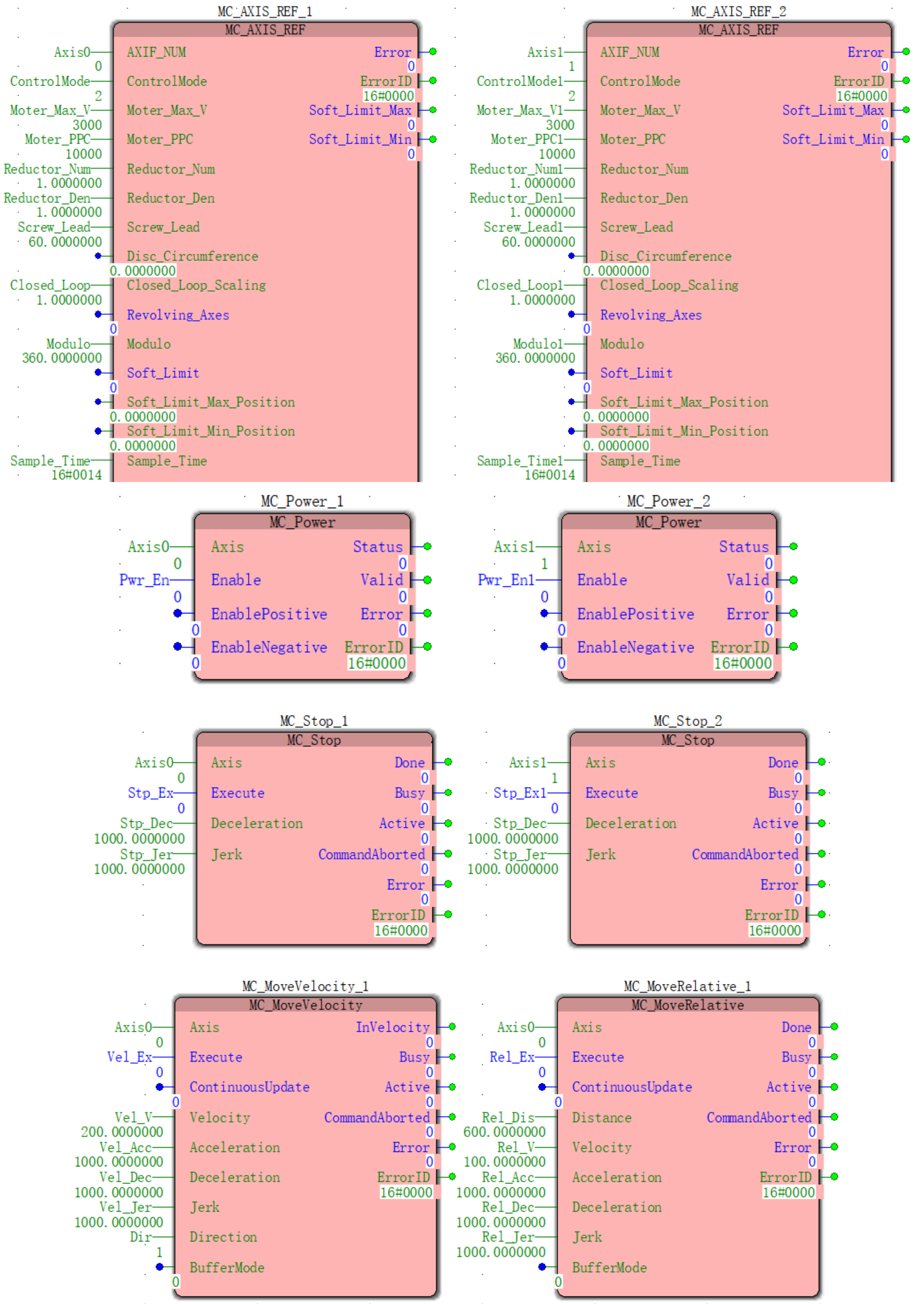
| | | |
|---|---|---|
| Stp_Ex | BOOL | |
| Stp_Dec | LREAL | 1000.0 |
| Stp_Jer | LREAL | 1000.0 |
| MC_Stop_2 | MC_Stop | |
| Stp_Ex1 | BOOL | |
| MC_MoveVelocity_1 | MC_MoveVelocity | |
| Vel_Ex | BOOL | |
| Vel_V | LREAL | 200.0 |
| Vel_Acc | LREAL | 1000.0 |
| Vel_Dec | LREAL | 1000.0 |
| Vel_Jer | LREAL | 1000.0 |
| Dir | INT | 1 |
| MC_MoveRelative_1 | MC_MoveRelative | |
| Rel_Ex | BOOL | |
| Rel_Dis | LREAL | 600.0 |
| Rel_V | LREAL | 100.0 |
| Rel_Acc | LREAL | 1000.0 |
| Rel_Dec | LREAL | 1000.0 |
| Rel_Jer | LREAL | 1000.0 |

At this point, the programming is complete.

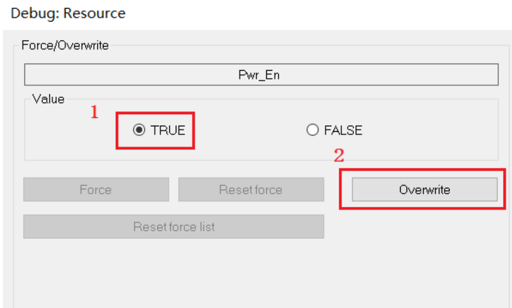
Step4: making the bottom-loading project. Click on the toolbar to make sure the program is correct, click on the download program, and then click on the cold start. After the cold start is successful, the status is displayeDAs the running status .



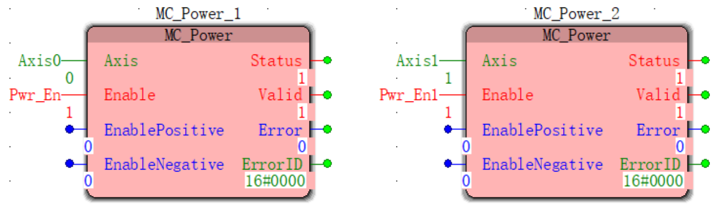

Step5: program debugging. Single- click online icon  on the toolbar can program debugging and monitoring. Online monitoring as shown below



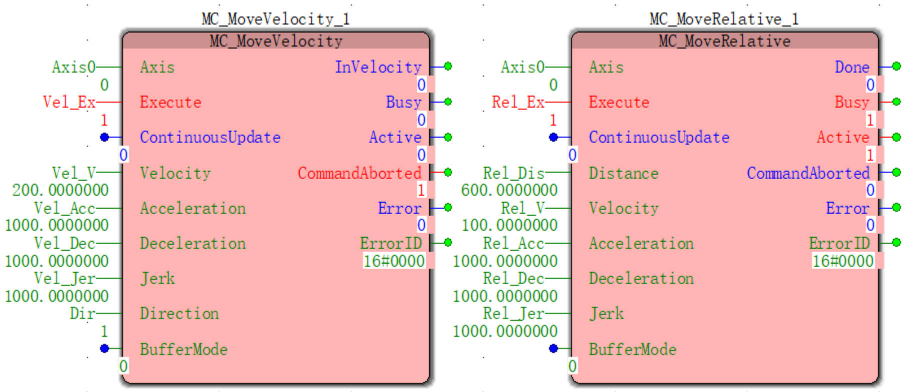

Debugging 1 : Double-click input function bit Pwr_ En , pop-up commissioning: Resource interface, variable values select Ture , then click on the cover, Pwr_ En will False become Ture ; the same token the Pwr_ En1 value becomes Ture ;



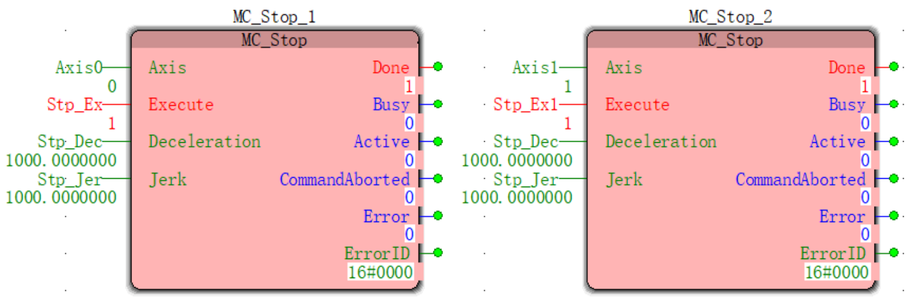

When the Val id is changed from False to Ture , the motion controller axis is successfully enabled, and the servo is enabled simultaneously by communication , as shown in the figure ;




Commissioning 2 : Similarly, each double-click Vel of the _ex , Rel _ex , so that the value False becomes Ture , the controller will control the A XIS 0 at the speed of walking speed mode is set, the Axis . 1 will be set according to the displacement amount and speed of Take the relative displacement mode as shown ;




Debug 3 : At this point, double-click Stp _Ex to change its value from False to True , and the controller will control Axis 0 anDAxis 1 to decelerate according to the set deceleration until it stops , as shown .



# Ⅹ Logic Instructions

The PLC instruction encapsulates the program block, each instruction can complete certain logic and operation operations, and the instructioNSet is a collection of PLC instructions. In the MULTIPROG programming, for programming convenience, these instructions are assigned to several different functional areas ( or libraries ) . These function blocks can be listed separately in the editing wizard in MULTIPROG . This chapter will follow these differences. The division of the functional area ( or library ) , the following instructions are introduced

✎·1 : Function

✎·2 : Function block

✎·3 : Type conversion FU

✎·4 : String FU

✎·5 : Bit manipulation function BIT_UTIL

✎·6 : P roConO S function

Note :

In the instruction description of the IL programming language, the LDAnd ST operators are often used , and their use is as follows :

The LD IN ( * the LD represents the variable IN chargeDAccumulateDAdder * )

The ABS ( * the ABS represents the accumulated value of an absolute value, sending the results accumulateDAdder * )

ST OUT ( * ST represents the accumulated value is assigned to the variable OUT * )

In the instructioNSpecification of the ST programming language, " := " is an assignment operator.

# 1 0.1 function

A function is a program organization unit POU with multiple input parameters and one output parameter . They do not have any internal memory. Calling a function with the same value always returns the same result. The return value is a single variable, or a multi-element variable such as an array or structure. The abbreviation for function is FU .

The following functions can be used during MULTIPROG programming

▲ type conversion function
▲ numerical function
▲ arithmetic operation function
▲ Bit Boolean function
▲ bit string function
▲ Select computing function
▲ Comparative computing function
▲ string function

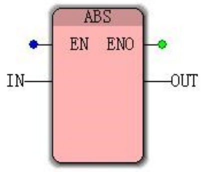
Instructions contained in the function ( in the Edit Wizard, select " Features " from the drop-down list )

| name | name | name | name |
|---|---|---|---|
| ABS | DIV_T_R | MAX | ROL |
| ACOS | EQ | MIN | ROR |
| ADD | EXP | MOD | SEL |
| ADD_T_T | EXPT | MOVE | SHL |
| AND | GE | MUL | SHR |
| ASIN | GT | MUL_T_AI | SIN |
| ATAN | LE | MUL_T_AN | SORT |
| COS | LIMIT | MUL_T_R | SUB |
| DIV | LN | NE | SUB_T_T |
| DIV_T_AI | LOG | NOT | TAN |
| DIV_T_AN | LT | OR | XOR |

In the following LDAnd FBD instruction description, only when the input pin EN is 1 when the command is active , when the instruction is executed successfully, the output pin ENO set to 1 , otherwise the pin ENO set 0

# 10.1.1 ABS (absolute value instruction )

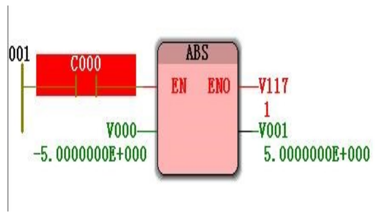
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: ABS instruction is used to find the absolute value of negative number</td></tr>
<tr><td>LD IN</td><td rowspan="2"></td></tr>
<tr><td>ABS</td></tr>
</table>

| | |
|---|---|
| ST OUT |  |
| ST programming language | |
| OUT:=ABS ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

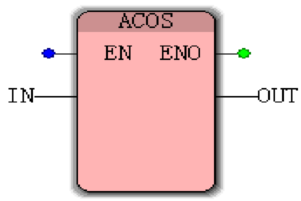
**数据 Data type processed by ABS instruction**

| Input and output | Operand type | description |
|---|---|---|
| IN | ANY_NUM | Input |
| OUT | ANY_NUM | Output |

**program demonstration**

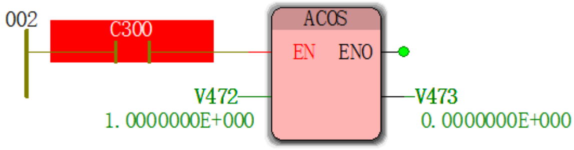
| Find the absolute value of -5.0 : | description |
|---|---|
|  | When the contact C000 bit is ON ,<br>Put the value of V000 in absolute value and install it in V001 . |

## 10.1.2 ACOS ( anti-cosine instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function : ACOS instruction is used to find the inverse cosine of the input value | |
| LD IN |  |
| ACOS | |
| ST OUT | |
| ST programming language | |
| OUT:=ACOS ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢ **ACOS instruction processing data types**

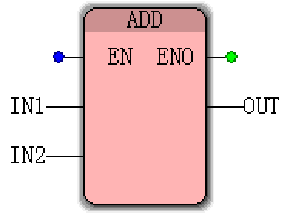
| Find the inverse cosine of 1.0 : | description |
|---|---|

| | |
|---|---|
|   | When the contact C300 is ON , the instruction is executed, and the address is the value of V472 ;<br>Anti After cosine , save the address V473 in ,<br>Execution result: ACOS ( 1.0 ) = 0.0 |

| Input and output | Operand type | description |
|---|---|---|
| IN | REAL | Input |
| OUT | REAL | Output, angle in radians |

**program demonstration**

## 10.1.3 ADD (Additional Instruction )

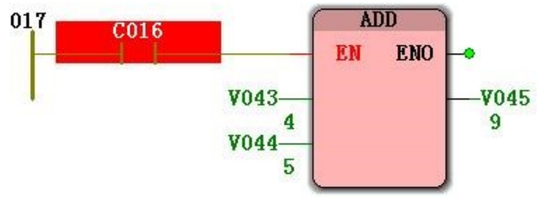
| IL programming language | LD, FBD programming language |
|---|---|
| Function : ADD instruction is used to find the sum of two data | |
| LD IN1 |   |
| ADD IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=IN1+IN2 | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**Data type processed by ADD instruction**

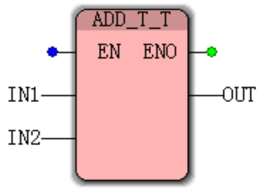
| Input and output | Operand type | description |
|---|---|---|
| IN1 | ANY_NUM | Addend |
| IN2 | ANY_NUM | Addend |
| OUT | ANY_NUM | And: OUT=IN1+IN2 |

**program demonstration**

| Find the value of the integer 4 plus 5 | description |
|---|---|

<table>
<tr><td>
</td><td>When the contact C016 is ON , the instruction is executed, and two 16 -bit integers are added to obtain a 16 -bit integer;<br>Execution result: VO43+V044=V045</td></tr>
</table>

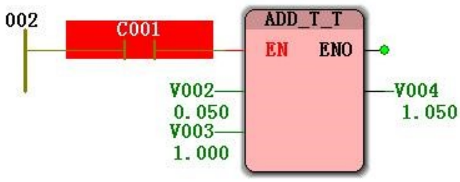
## 10.1.4 ADD_T_T (Time Addition Instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: ADD_T_T instruction is used to find the sum of two time data</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>ADD_T_T IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=ADD_T_T ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

- **ADD_T _T instruction processing data types**

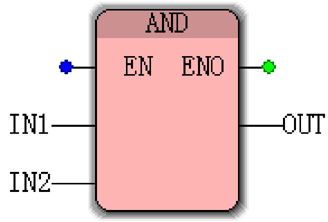
| Input and output | type of data | description |
|---|---|---|
| IN1 | TIME | Addend |
| IN2 | TIME | Addend |
| OUT | TIME | And, OUT=IN1+IN2 |

**program demonstration**

<table>
<tr><th>Find the value of T#1s+T#50ms</th><th>description</th></tr>
<tr><td>
</td><td>When C001 is ON , the instruction is executed, and the time data types are added to be added;<br>Execution result: V002+V003=V004</td></tr>
</table>

## 10.1.5 AND ( Logic and Instruction )

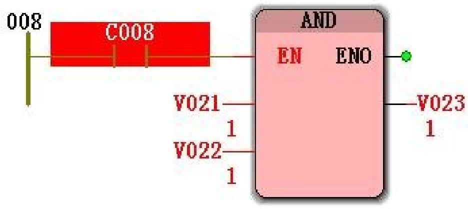
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: AND instruction is used for the logical AND operation of two data</td></tr>
<tr><td>LD IN1</td><td></td></tr>
</table>

| | |
|---|---|
| AND IN2 |  |
| ST OUT | |
| ST programming language | |
| OUT:=IN1&IN2 | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

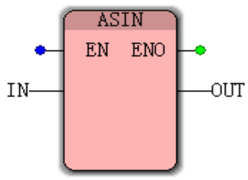
**Data type processed by AND instruction**

| Input and output | Operand type | description |
|---|---|---|
| IN1 | ANY_BIT | Data 1 |
| IN 2 | ANY_BIT | Data 2 |
| OUT | ANY_BIT | Result :<br>IN1=0 , IN2=0, OUT=0;<br>IN1=0 , IN2=1, OUT=0;<br>IN1=1, IN2=0, OUT=0;<br>IN1=1, IN2=1, OUT=1; |

**program demonstration**

| Find the result of the logical AND of two data | description |
|---|---|
|  | When the contact C008 bit is ON , the instruction is executed to phase the two operands;<br>Execution result: V021&V022=V023 |

## 10.1.6 ASIN ( anti-sinusoidal command)

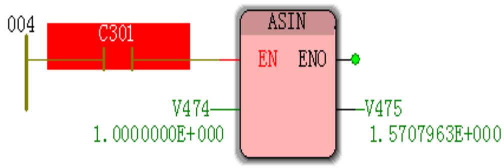
| IL programming language | LD, FBD programming language |
|---|---|
| Function: ASIN instruction for seeking inverse sine | |
| LD IN |  |
| ASIN | |
| ST OUT | |
| ST programming language | |
| OUT:=ASIN ( IN ) | |

| |
|---|
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet |

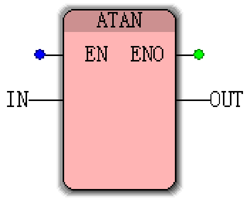
➢ **ASIN instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN | REAL | Input |
| OUT | REAL | Output, angle in radians |

**program demonstration**

| Find the inverse sine of 1.0 | description |
|---|---|
|  | When the contact C301 is ON when this instruction is executed, the address V474 the value of anti after sine , is stored in address V475 in :<br>Execution result: ASIN ( 1.0 ) =1.570 7 9763 … |

## 10.1.7 ATAN (Arc Tangent Command )

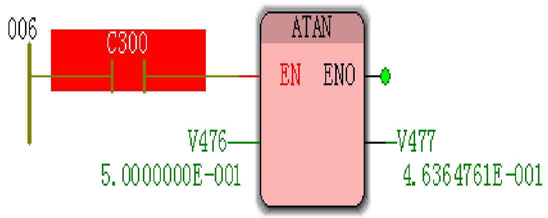
| IL programming language | LD, FBD programming language |
|---|---|
| Function: ATAN instruction is used to find the inverse tangent | |
| LD IN |  |
| ATAN | |
| ST OUT | |
| ST programming language | |
| OUT:=ATAN ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢ ATAN instruction processing data type

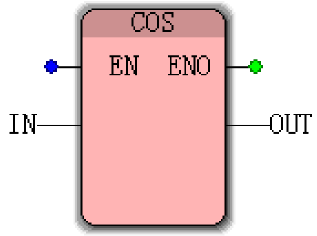
| Input and output | type of data | description |
|---|---|---|
| IN | *REAL* | Input |
| OUT | *REAL* | Output, angle in radians |

➢Funtion and Action examples

| Find the inverse tangent value of 1/2 | description |
|---|---|

| | |
|---|---|
|  | When the contact C300 is ON when the instruction is executed; address V476 the value of the arctangent, stored in the address V477 in :<br>Execution result: ATAN ( 1/2 ) =4.6364761 E-001 |

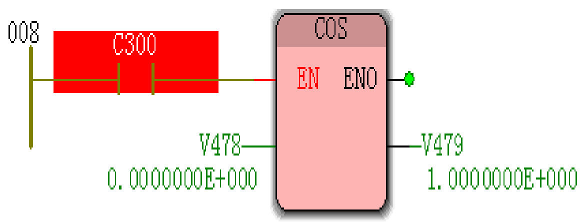
## 10.1.8 COS (cosine command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: OS command is used to find the cosine of the input value | |
| LD IN |  |
| COS | |
| ST OUT | |
| ST programming language | |
| OUT:=COS ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

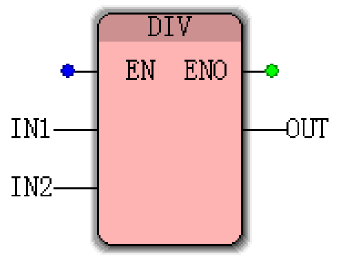
➤Data type processed by COS instruction

| Input and output | type of data | description |
|---|---|---|
| IN | REAL | Input, the angle is expressed in radians |
| OUT | REAL | Output |

➤ Funtion and Action examples

| Find the cosine of 0 ◦ | description |
|---|---|
|  | When the contact C300 is ON time , execution of the instruction; address V478 the value , the cosine deposit to V479 of :<br>Execution result: COS ( 0 ◦ ) =1.00000 |

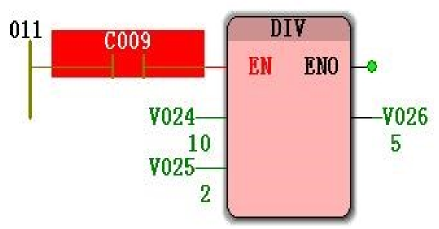
## 10.1.9 DIV (Division Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: DIV instruction for division operation | |
| LD IN1 |  |
| DIV IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=IN1/IN | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**Data type processed by DIV instruction**

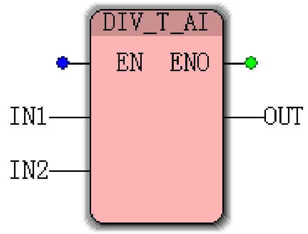
| Input and output | Operand type | description |
|---|---|---|
| IN1 | ANY_NUM | Divisor |
| IN2 | ANY_NUM | divisor |
| OUT | ANY_NUM | Business |

**program demonstration**

| Seeking integer 10 divided by 2 's | description |
|---|---|
|  | When C009 is ON when the instruction is executed, V024 divided by V025 quotient on V026 of:<br><br>Execution result: V024/V025=V026 |

## 10.1.10 DIV_T_AI ( division ( time divided by an integer ) instruction )

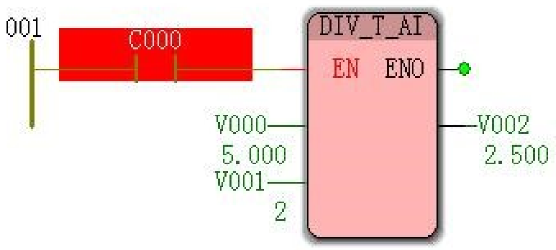
| IL programming language | LD, FBD programming language |
|---|---|
| Function: DIV_T_AI instruction is used to divide time by integer operation | |
| LD IN1 | |

| | |
|---|---|
| DIV_T_AI IN2 |   |
| ST OUT | |
| ST programming language | |
| OUT:=DIV_T_AI ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

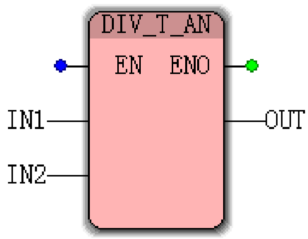
**DIV_T_AI instruction processing data type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | TIME | Divisor |
| IN2 | ANY_INT | divisor |
| OUT | TIME | Business |

**program demonstration**

| Find the quotient of time 5s divided by the integer 2 ; | description |
|---|---|
|   | When the contact C000 bit is ON , the instruction is executed, and the value of the address V000 is divided by the value of the address V001 and stored in V002 .<br><br>Execution result: V000/V001=V002 |

## 10.1.11 DIV_T_AN ( division ( time divided by an integer, a real number ) command )

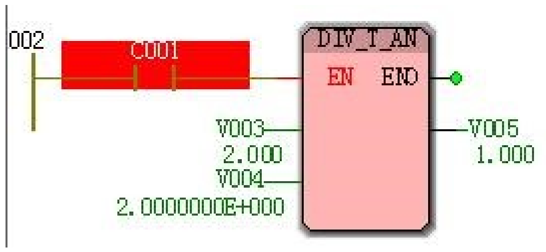
| IL programming language | LD, FBD programming language |
|---|---|
| Function: DIV_T_AN instruction is used to divide time by integer or real operation | |
| LD IN1 |   |
| DIV_T_AN IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=DIV_T_AN ( IN1 , IN2 ) | |

| |
|---|
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet |

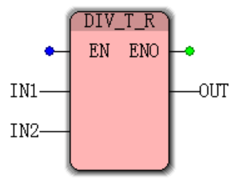
➢ **DIV_T_AN instruction processing data types**

| Input and output | Operand type | description |
|---|---|---|
| IN1 | TIME | Divisor |
| IN2 | ANY_NUM | divisor |
| OUT | TIME | Business |

**program demonstration**

<table>
<tr><th>Find the quotient of time 2s divided by the integer 2 ;</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C001 to the ON , execution of the instruction, the address V003 value by dividing the address V004 provider, stored in the V005 address;<br><br>Execution result: V003/V004=V005</td></tr>
</table>

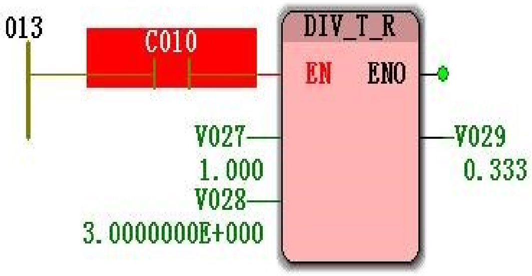
## 10.1.12 DIV_T_ R ( division ( time divided by real number ) instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The DIV_T_R instruction is used to divide the time by the real number operation.</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>DIV TR IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=DIV_T_R ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

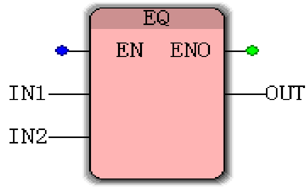
➢ **div_t _R instruction processing data types**

| Input and output | Operand type | description |
|---|---|---|
| IN1 | TIME | Divisor |
| IN2 | REAL | divisor |
| OUT | TIME | Business |

➢**Funtion and action examples**

| Find the time 1s divided by the number of floating point 3.0 : | description |
|---|---|
| <br> | When the contact C010 to ON to execute the instruction, the address V027 value is divided by address V028 quotient value, stored in the V029<br><br>Execution result: V027/V028=V029 |

## 10.1.13 EQ (equal to the instruction )

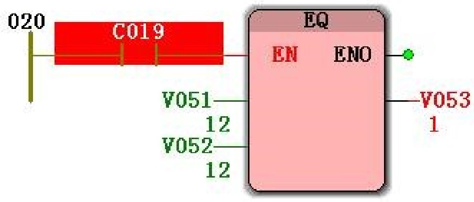
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: EQ command is used to judge whether two numbers are equal</td></tr>
<tr><td>LD IN1</td><td rowspan="5"><br></td></tr>
<tr><td>EO IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=IN1=IN2</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

Data type processed by EQ instruction

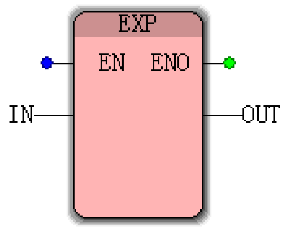
| Input and output | Operand type | description |
|---|---|---|
| IN1 | ELEMENTARY | Data 1 |
| IN2 | ELEMENTARY | Data 2 |
| OUT | BOOL | Output<br>The two numbers are equal and TRUE;<br>The two numbers are not equal |

| | | |
|---|---|---|
| | | anDAre FALSE |

program demonstration

| Compare the values in the two addresses; | description |
|---|---|
|  | When the contact C019 is ON , the instruction is executed; the magnitudes of the values of the two addresses V051 and V052 are compared , and if they are equal, V053 outputs 1 . |

## 10.1.14 EXP ( exponential function instruction of natural number e )

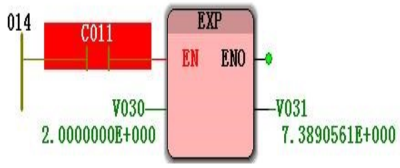
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The instruction is used to calculate the x- th power of the natural constant e , where x is the input, where e ≈ 2.718281828 . | |
| LD IN |  |
| EXP | |
| ST OUT | |
| ST programming language | |
| OUT:=EXP ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**Data type processed by the EXP instruction**

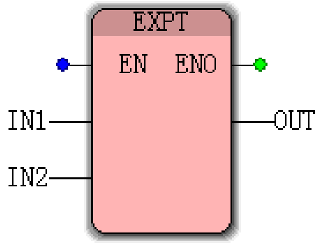
| Input and output | Operand type | description |
|---|---|---|
| IN | REAL | index |
| OUT | REAL | The results, E of IN power |

**➢Funtion and action examples**

| Find the value of $e^2$: | description |
|---|---|

| | |
|---|---|
|   | When the contact C011 is ON , the instruction is executed, wherein the value in the address V030 is the power of X ;<br><br>Execution result: The value of $e^{V030}$ is stored in V031 ; |

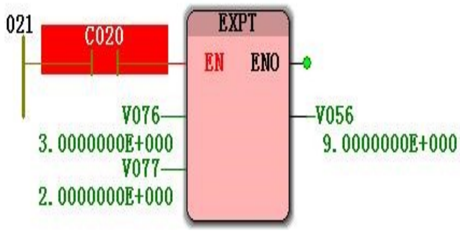
## 10.1.15 EXPT of ( a power of ( X to Y -th power ) instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The EXPT instruction is used to calculate the Y power of X , where X is the first input and Y is the second input.</td></tr>
<tr><td>LD IN1</td><td rowspan="5"></td></tr>
<tr><td>EXP IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=EXPT ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

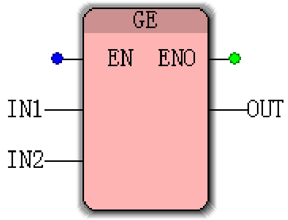
➢ **EXPT of instruction processing data types**

| Input and output | Operand type | description |
|---|---|---|
| IN1 | ANY_REAL | Cardinal number |
| IN2 | ANY_NUM | index |
| OUT | REAL | As a result, the IN2 power of IN1 |

➢ **Funtion and action examples**

| Find the value of the 2nd power of 3 : | description |
|---|---|
|   | When the contact C020 is ON , the instruction is executed, with V076 as the base, V077 is the index, and the result of the operation exists V056;<br><br>Result of execution: $V076^{V077}$=V056 |

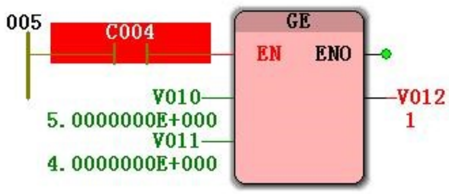
## 10.1.16 GE (greater than or equal to the command )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: GE instruction for comparing two values, when the first input is greater than or equal to the first time two, the output is 1 , the other is 0 .</td></tr>
<tr><td>LD IN1</td><td rowspan="6">
</td></tr>
<tr><td>GE IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=IN1>=IN2;</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

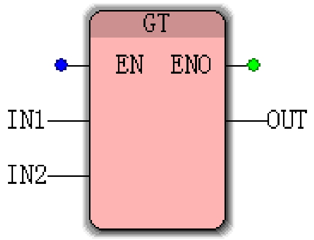
**数据 Data type processed by GE instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY | First input |
| IN2 | ANY | Second input |
| OUT | BOOL | As a result, when IN1 >= IN2 , OUT is 1 |

**program demonstration**

<table>
<tr><th>Compare the size of the two V010 and V011 addresses:</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C004 is ON , the instruction is executed; the value of the V010 value and the V011 value are compared; when the value in the V010 address is greater than or equal to V011 , the output V012 is 1 ;<br>Execution result: V010>V011=V012=1</td></tr>
</table>

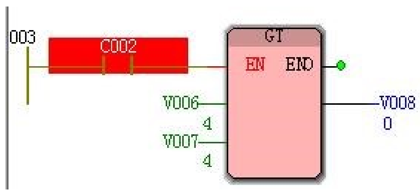
## 10.1.17 GT (greater than instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The GT instruction is used to compare the size of two values. When the first input is greater than the second , the output is 1 and the others are 0 . | |
| LD IN1 |   |
| GT IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=IN1>IN2 | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**Data type processed by GT instruction**

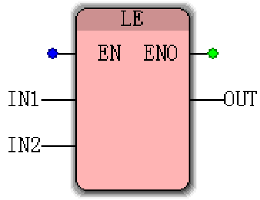
| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY | First input |
| IN2 | ANY | Second input |
| OUT | BOOL | As a result, when IN1 > IN2 , OUT is 1 |

**program demonstration**

| Compare the size of the two V010 and V011 addresses: | description |
|---|---|
|   | When the contact C002 is ON , the instruction is executed. When the value of the address V006 is greater than the value of the address V007 , V008 is V 006. |

## 10.1.18 LE (less than or equal to the instruction )

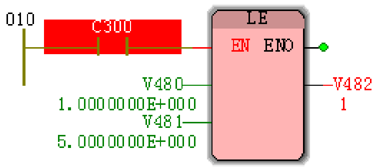
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The LE instruction is used to compare the size of two values. When the first input is less than or equal to the second one, the output is 1 and the others are 0. | |
| LD IN1 | |

| | |
|---|---|
| LE IN2 |  |
| ST OUT | |
| ST programming language | |
| OUT:=IN1<=IN2 | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

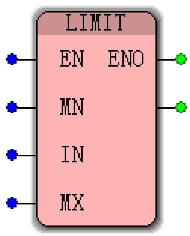
➢**Data type processed by LE instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY | First input |
| IN2 | ANY | Second input |
| OUT | BOOL | result<br>When IN1<=IN2 , OUT is 1<br>When IN1>IN2 , OUT is 0; |

➢**program demonstration**

| Compare V480 and V481 in value | description |
|---|---|
|  | When the contact C300 is ON , the instruction is executed;<br>When the address V480 is smaller than the address V481 the value , V482 output is 1 :<br>When the address V480 is a value greater than equal to the address V481 the value , V482 output is 0 :<br>Execution result: 1.0<5.0 output 1 |

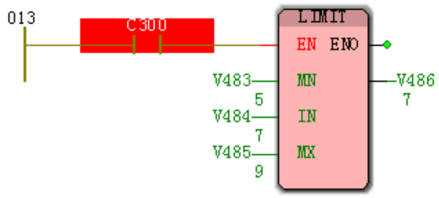
## 10.1.19 LIMIT (limit selection instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The LIMIT instruction is used to limit the input value to the interval determined by the maximum and minimum values. | |
| LD IN1 |  |
| LIMIT IN , IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:= LIMIT ( IN1 , IN, IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN, IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

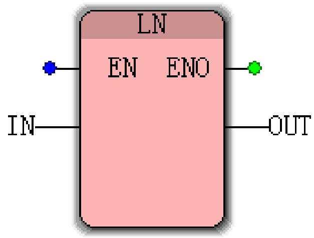
### ➢LIMIT instruction processing data type

| Input and output | Operand type | description |
|---|---|---|
| IN1 | ANY_INT | Minimum value |
| IN | ANY_INT | input value |
| IN2 | ANY_INT | Maximum |
| OUT | ANY | output value<br>When IN1 <= ( the IN ) <= IN2 when , OUT = the IN;<br>When IN<IN1 , OUT=IN1;<br>When IN>IN2 , OUT=IN2; |

### ➢Funtion and Action examples

<table>
<tr><th>Selected from the selection number output 5-9 between integer</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C300 is ON when performing the instru ction ;<br>Input value IN in the MN ~ MX between output IN ,<br>When IN is greater than equal to MX when the output MX;<br>When IN is less than equal to MN output MN:<br><br>Execution result: 5<7 <9 output 1</td></tr>
</table>

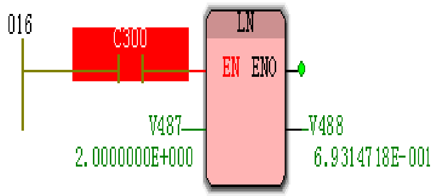
## 10.1.20 LN (Natural Logarithmic Instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The LN instruction is used to calculate the natural logarithm of the input.</td></tr>
<tr><td>LD IN</td><td rowspan="5">
</td></tr>
<tr><td>LN</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=LN ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

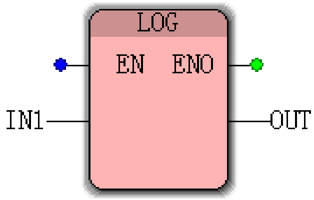
### ➢Data type processed by LN instruction

| Input and output | type of data | description |
|---|---|---|
| IN | REAL | input value |
| OUT | REAL | Result, OUT = LOG e IN |

➢**Funtion and Action examples**

| Seeking to log $_{E}{}^{2}$ is the value of | description |
|---|---|
|  | When the contact C300 is ON when this instruction is executed, where V 487 is the index;<br><br>Execution of Results : log $_{E}{}^{2}$ = 6.9314718E-001 |

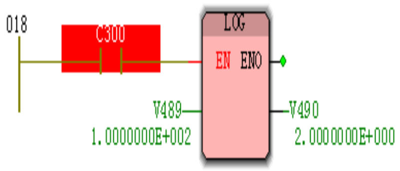
## 10.1.21 LOG (Logarithmic Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The LOG command is used to calculate the base 10 logarithm of the input . | |
| LD IN |  |
| LOG | |
| ST OUT | |
| ST programming language | |
| OUT:=LOG ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

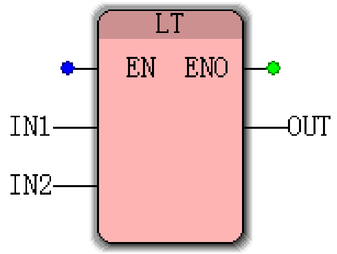
➢**Data type processed by LOG instruction**

| Input and output | Type of data | description |
|---|---|---|
| IN | REAL | input value |
| OUT | REAL | As a result, OUT = LOG10IN=lg ( IN ) |

➢**Funtion and Action examples**

| On the logging log $_{10}{}^{100}$ Value | description |
|---|---|
|  | When the contact C300 is ON when performing the instruction;Where the value of address V489 is an index<br><br>Execution result: log $_{10}{}^{100}$ =2.0 |

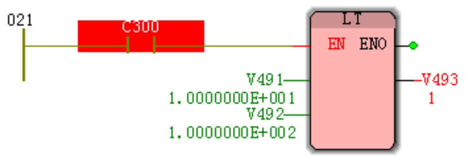
## 10.1.22 LT (less than instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The LT instruction is used to compare the size of two values. When the first input is less than the second, the output is 1 and the others are 0. | |
| LD IN1<br>LT IN2<br>ST OUT<br>ST programming language<br>OUT:=IN1<IN2 |  |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**➢Data type processed by LT instruction**

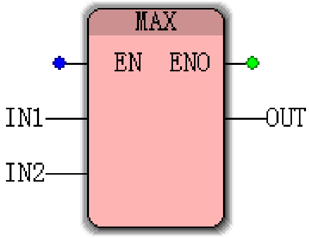
| Input and output | Operand type | description |
|---|---|---|
| IN1 | ANY | First input |
| IN2 | ANY | Second input |
| OU T | BOOL | Result :<br>When IN1 < IN2 , OUT is 1;<br>When IN1 >= IN2 , OUT is 0; |

**➢Funtion and Action examples**

| Comparative V491 ( 10.0 ) and V492 ( 100.0 ) of the value | description |
|---|---|
|  | When the contact is ON when the instruction is executed, when the address V491 is a value smaller than V492 the value , the address V493 the output 1<br><br>Execution result: 10.0< 100.0 = V493 output 1 |

## 10.1.23 MAX (Maximum Instruction )

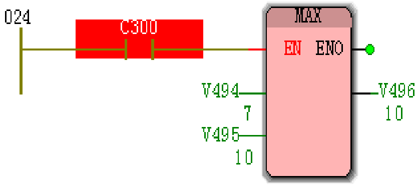
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MAX instruction is used to determine the maximum value of two values. | |

| | |
|---|---|
| LD IN1 |  |
| MAX IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=MAX ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

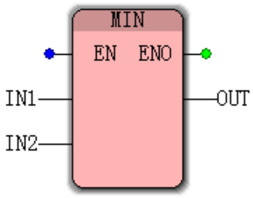
**➢Data type processed by MAX instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_NUM | First input |
| IN2 | ANY_NUM | Second input |
| OUT | ANY | Result :<br>When IN1 <= IN2 , OU T is IN2;<br>When IN1 >= IN2 , OUT is IN1; |

**➢Funtion and Action examples**

| Integer 7 and 10 output the largest value | description |
|---|---|
|  | When the contact is ON , the execution of the instruction, comparing the address V494 and V495 the value, the greater the output there is an address V496 in:<br><br>Execution result : 10>7 output 10 |

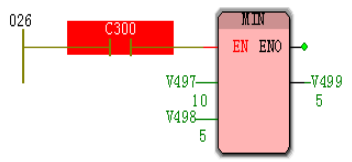
# 10.1.24 MIN (minimum instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The instruction is used to determine the minimum of two values. | |
| LD IN1 |  |
| MIN IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=MIN ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

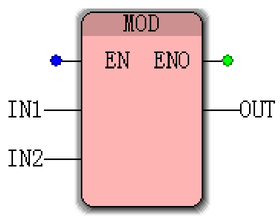
**➢The data type processed by the MIN instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_NUM | First input |
| IN2 | ANY_NUM | Second input |
| OUT | ANY | Result :<br>When IN1 <= IN2 , OUT is IN1;<br>When IN1 >= IN2 , OUT is IN2; |

**➢Funtion and Action examples**

<table>
<tr><th>An integer of 5 to 10 output the most small in value</th><th>description</th></tr>
<tr><td></td><td>When the contact is ON , the instruction is executed to compare the values of the addresses V49 7 and V49 8. The larger output exists in the address V49 9 :<br><br>Execution result : 5<10 output 5</td></tr>
</table>

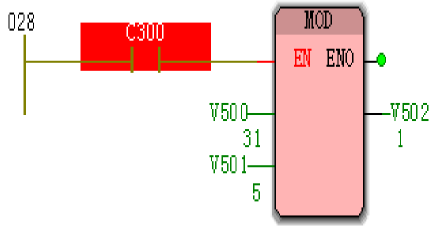
## 10.1.25 MOD (modulo instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The MOD instruction is used to determine the remainder of the division of two values.</td></tr>
<tr><td>LD IN1</td><td rowspan="5"></td></tr>
<tr><td>MOD IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=MOD ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

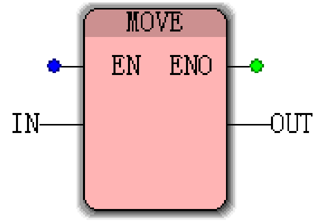
**➢Data type processed by MOD instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_INT | Divisor |
| IN2 | ANY_INT | divisor |
| OUT | ANY_INT | As a result, the remainder of IN1 divided by IN2 |

**➢Funtion and Action examples**

| Find the value of the remainder of the integer 31 divided by 5 . | description |
|---|---|
|  | When the contact C300 is ON time , the instruction is executed;<br>The address of the V500 is divided by the V501 the value of the remainder is stored in the address V502 in<br><br>Execution result: 31/5 remainder is 1 |

## 10.1.26 MOVE (Assignment Command )

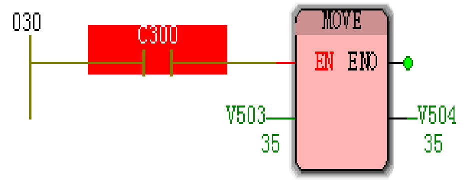
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The MOVE instruction is used to assign an input value to the output value.</td></tr>
<tr><td>LD IN</td><td rowspan="6"></td></tr>
<tr><td>MOVE</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=MOVE ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

➢**Data type processed by MOVE instruction**

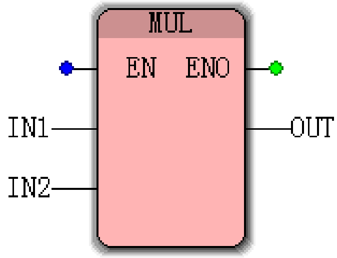
| Input and output | type of data | description |
|---|---|---|
| IN | ANY_NUM | input value |
| OUT | ANY_NUM | Output value, OUT=IN |

➢**Funtion and Action examples**

| Address V503 is the value transmitted to the address V504 | description |
|---|---|

| | |
|---|---|
|   | When the contact C300 is ON when this instruction is executed, the address of the V503 value is transferred to the address V504 in:<br><br>Execution result: V503 = V504 |

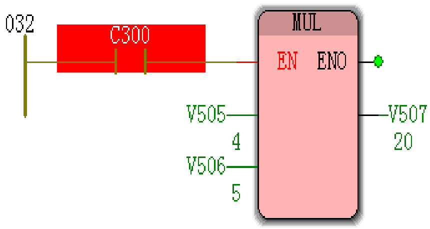
## 10.1.27 MUL (Multiplication Directive )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MUL instruction is used to find the product of two data. | |
| LD IN1<br>MUL IN2<br>ST OUT<br>ST programming language<br>OUT :=IN1*IN2 |   |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

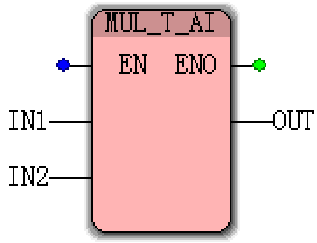
**➢The MUL data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_NUM | First input |
| IN2 | ANY_NUM | Second input |
| OUT | ANY_NUM | Product, OUT=IN1*IN2 |

**➢Funtion and Action examples**

| Find the value of the integer 4*5 | description |
|---|---|
|   | When the contact C300 is ON when this instruction is executed, the address the V505 value by multiplying the address V506 the value accumulated in V507 :<br><br>Execution result: V505 * V506=V507 |

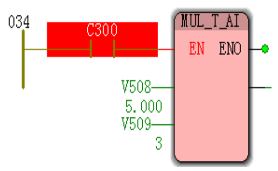
## 10.1.28 MUL_T_AI ( multiplication ( time multiplied by integer ) instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The MUL_T_AI instruction is used to calculate the product of a time type data anDAn integer type data.</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>MUL_T_AI IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:= MUL_T_AI ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

➢ **MUL_T_AI data processing instruction type**

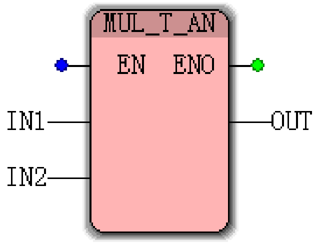
| Input and output | type of data | description |
|---|---|---|
| IN1 | TIME | First input, time |
| IN2 | ANY_INT | Second input, integer |
| OUT | TIME | Product, OUT= IN1*IN2 |

➢**Funtion and Action examples**

| Find the time T #5s multiplied by the value of the integer 3 | description |
|---|---|
|  | When the contact C300 is ON , the instruction is executed, and the value of the address V508 is multiplied by the value product in the address V509 and stored in V510 :<br><br>Execution result: V508 * V50 9 = V510 |

## 10.1.29 MUL_T_AN ( multiplication ( time multiplied by integer, real ) instructions )

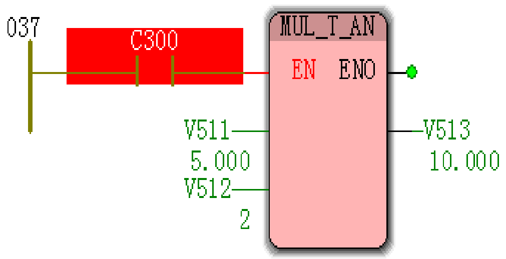
| IL programming language | LD, FBD programming language |
|---|---|

| | |
|---|---|
| Function: The MUL_T_AN instruction is used to calculate the product of a time type data anDAn integer or floating point number. | |
| LD IN1 |  |
| MUL_T_AN IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=MUL_T_AN ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

➢ **MUL_T_AN instruction processing data types**

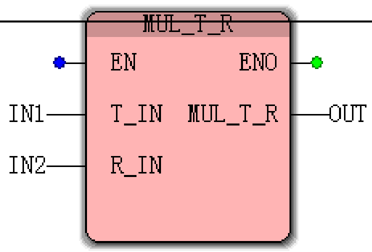
| Input and output | type of data | description |
|---|---|---|
| IN1 | TIME | First input, time |
| IN2 | ANY_NUM | Second input, integer or floating point number |
| OUT | TIME | Product, OU T= IN1*IN2 |

➢**Funtion and Action examples**

| Find the time T #5s multiplied by the value of the integer 2 | description |
|---|---|
|  | When the contact C300 is ON , the instruction is executed, and the value of the address V5 11 is multiplied by the value product in the address V5 12 and stored in V51 3 :<br><br>Execution result: V511 * V5 12 =V5 13 |

## 10.1.30 MUL_T_R ( multiplication ( time multiplied by real number ) instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MUL_T_R instruction is used to calculate the product of a time type data anDA floating point type data. | |
| LD IN1 | |

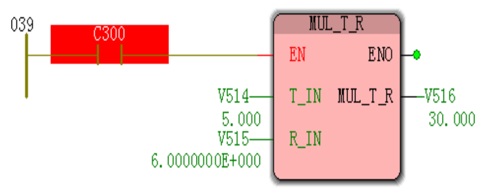


| | |
|---|---|
| MUL_T_R IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=MUL T-R ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

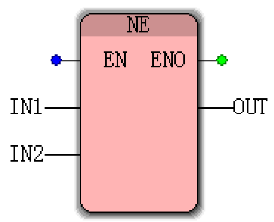
**➤MUL_T_R data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 ( T_IN ) | TIME | First input, time |
| IN2 ( R_IN ) | ANY_NUM | Second input, floating point number |
| OUT ( MUL_T_R ) | TIME | Product, OUT=IN1*IN2 |

**➤Funtion and Action examples**

| Find the time T #5s multiplied by the value of floating point 6.0 | description |
|---|---|
|  | When the contact C300 is ON , the instruction is executed, and the value of the address V5 14 is multiplied by the value product in the address V5 15 and stored in V51 6 :<br><br>Execution result: V514 * V5 15 =V5 16 |

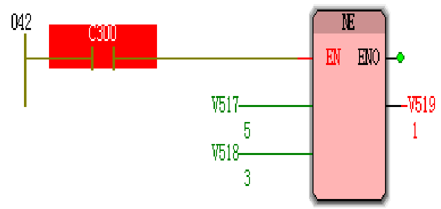
## 10.1.31 NE (not equal to the instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The NE command is used to judge the magnitude relationship between two values. When the first input is not equal to the second one, the output is 1 and the others are 0. | |
| LD IN1 |  |
| NE IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=IN1<>IN2 | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

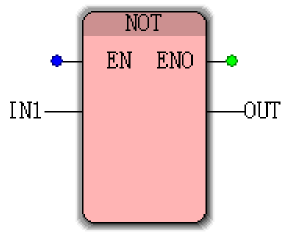
**➤Data type processed by NE instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY | First input |
| IN2 | ANY | Second input |
| OUT | BOOL | Result :<br>When IN1 <> IN2 , OUT is 1;<br>When IN1 = IN2 , OUT is 0; |

➢**Funtion and Action examples**

| Compare the values of addresses V517 and V518 | description |
|---|---|
|  | When the contact C300 is ON when this instruction is executed, the address V5 . 17 values of the address V5 18 is the value of the comparison, the output is not equal to 1<br><br>Execution result: V517 * V5 18 =V5 19 |

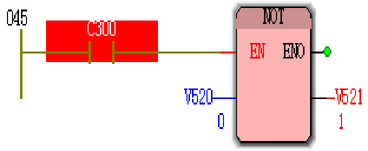
## 10.1.32 NOT ( logical non-instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The NOT instruction is used to invert the input value by bit, such as BYTE#2#11001100 , which is calculated by NOT .<br>BYTE#2#00110011 .</td></tr>
<tr><td>LD IN</td><td rowspan="6"></td></tr>
<tr><td>NOT</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=NOT ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

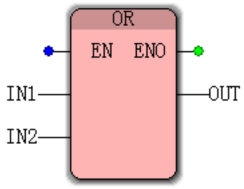
➢**Data type processed by NOT instruction**

| Input and output | type of data | description |
|---|---|---|
| IN | ANY_BIT | Input |
| OUT | ANY_BIT | result |

➢**Funtion and Action examples**

| Invert the value of address V520 ( BOOL ) | description |
|---|---|
|   | When the contact C300 is ON , the instruction is executed to invert the value of the address V5 20 ( BOOL ) :<br><br>Execution result: 0 ( V520 ) inversion output 1 ( V521 ) |

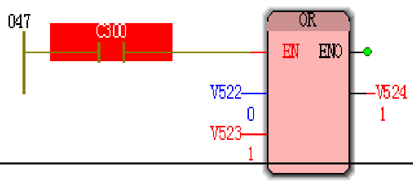
## 10.1.33 OR (Logic or Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The OR instruction is used to logically OR the input value by bit. | |
| LD IN1 |   |
| OR IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:= ( 1N1 ) OR ( 1N2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

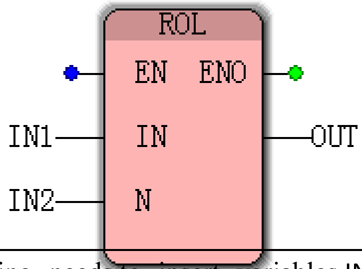
**➢Data type processed by the OR instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_BIT | First input |
| IN2 | ANY_BIT | Second input |
| OUT | ANY_BIT | Result, logical OR operation<br>IN1=0, IN2=0, OUT is 0;<br>IN1=0, IN2=1, OUT is 1;<br>IN1=1 , IN2=0 , OUT is 1;<br>IN1=1 , IN2=1 , OUT is 1; |

**➢Funtion and Action examples**

| Two addresses numerical values or calculation | description |
|---|---|
|   | When the contact C300 is ON , the instruction is executed, and the two addresses are OReDAnd output to the address V524 .<br><br>Execution result: V522^V523=V524 |

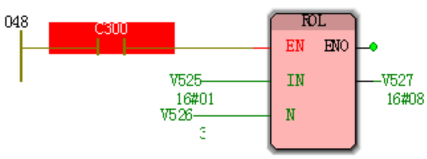
## 10.1.34 ROL (loop left shift instruction )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The ROL instruction is used to rotate the input value to the left by bit.</td></tr>
<tr><td>LD IN1</td><td rowspan="6">
</td></tr>
<tr><td>ROL IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=ROL ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

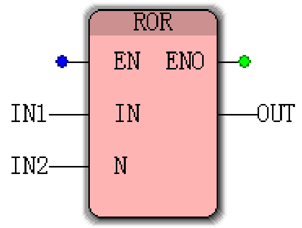
- **The ROL instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_BIT | Input |
| IN2 | ANY_INT | Number of bits shifted left |
| OUT | ANY_BIT | As a result, when the value of the pin IN or N changes, it is shifted left. As follows : shift two digits to the left,<br>0 0 0 0 1 0 1 1<br>Value after moving<br>1 1 0 0 0 0 1 0<br>Value before moving |

- **Funtion and Action examples**

<table>
<tr><th>The value 16 #01 ( BYTE ) is shifted to the left by three digits</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C300 is ON when this instruction is executed,<br>Move all bits of address V525 to the left by 3 bits:<br><br>Execution result: . 1 6 # 01 moves to the left . 3 bit stored to V527 in</td></tr>
</table>

## 10.1.35 ROR (cyclic right shift instruction )

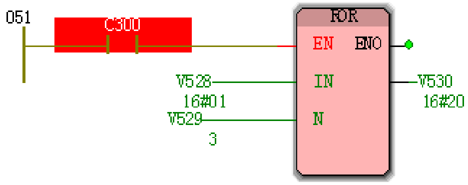
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The ROR instruction is used to cyclically shift the input value to the right. | |
| LD IN1 |  |
| ROR IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=ROR ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

➢**Data type processed by ROR instruction**

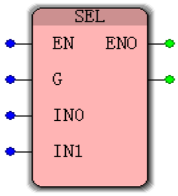
| Input and output | type of data | description |
|---|---|---|
| IN1 ( IN ) | ANY_BIT | Input |
| IN2 ( N ) | ANY_INT | Number of bits shifted right |
| OUT | ANY_BIT | As a result, when the value of the pin IN or N changes, it is shifted to the right. As follows : shift two bits to the right<br>1 1 0 0 0 1 1 1<br>Value after moving right<br>0 0 1 1 0 0 0 1<br>The value before the right movement |

➢**Funtion and Action examples**

| The value 16 # 01 ( BYTE ) rightward shift three | description |
|---|---|

| | |
|---|---|
|  | When the contact C300 is ON when this instruction is executed,<br>The address V528 all the bits to the right 3 bits:<br>Execution results: 1 6#01 Move 3 bits to the right to save to V530 : |

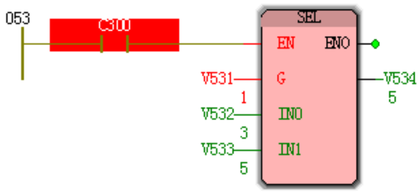
## 10.1.36 SEL (Selection Command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SEL instruction is used to select different input values based on two states of a Boolean quantity. | |
| LD IN |  |
| SEL IN1 , IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=SEL ( IN, IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN, IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

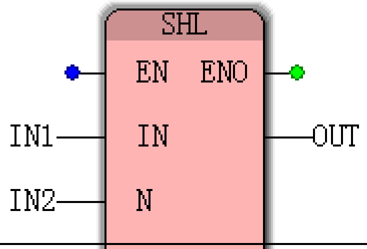
**➢Data type processed by SEL instruction**

| Input and output | type of data | description |
|---|---|---|
| IN ( G ) | BOOL | Select input |
| IN1 ( INO ) | ANY | First input |
| IN2 ( IN1 ) | ANY | Second input |
| OUT | ANY | Result :<br>If IN=0, OUT=IN1;<br>If IN=1, OUT=IN2; |

**➢Funtion and Action examples**

| Choose to output integer 3 or 5 | description |
|---|---|
|  | When the contact C300 is ON when this instruction is executed,<br>When G is . 1 when the output address V533 in value; when G is 0 when the output address V532 the value:<br><br>Execution result: output is 5 |

## 10.1.37 SHL (left shift instruction )

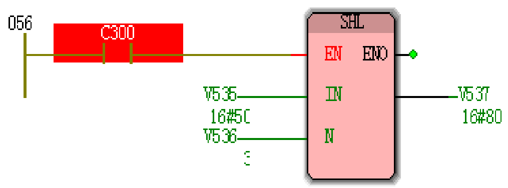
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SHL instruction is used to shift the input value to the left, the left end of the data is shifted out, and the right end is filled with 0 . | |
| LD IN1 |  |
| SHL IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=SHL ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**Data type processed by SHL instruction**

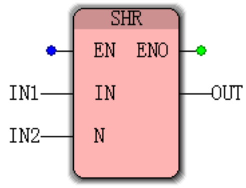
| Input and output | type of data | description |
|---|---|---|
| IN1 ( IN ) | ANY_BIT | Input |
| IN2 ( N ) | ANY_INT | Number of digits shifted to the left |
| OUT | ANY_BIT | As a result, when the value of the pin IN or N changes, it is shifted left. Move left by two, as follows<br>Value before left shift<br>1 1 1 0 0 0 1 1<br>← Value after left shift ←<br>1 0 0 0 1 1 0 0 |

**➢Funtion and Action examples**

| The address V536 ( BYTE ) the value of the left . 3 bit | description |
|---|---|

| | |
|---|---|
|  | When the contact C300 is ON when this instruction is executed, the address V535 all bits to the left . 3 bits: Execution result: 1 6#01 moves left 3 bits output 16 #80 |

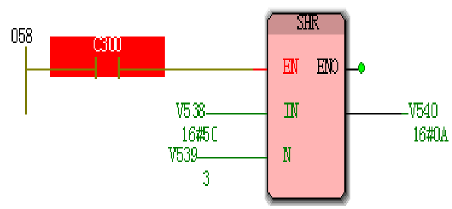
## 10.1.38 SHR (right shift instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The instruction is used to shift the input value to the right, the right end of the data is shifted out, and the left end is filled with 0 . | |
| LD IN1<br>SHR IN2<br>ST OUT |  |
| ST programming language<br>OUT:=SHR ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**Data type processed by SHR instruction**

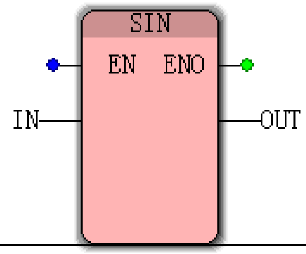
| Input and output | type of data | description |
|---|---|---|
| IN1 ( IN ) | ANY_BIT | Input |
| IN2 ( N ) | ANY_INT | Number of digits shifted to the right |
| OUT | ANY_BIT | As a result, when the value of the pin IN or N changes, it is shifted to the right. Move two digits to the right, as follows<br>The value before the right movement<br>0 0 0 0 0 0 1 1<br>Value after moving right<br>0 0 0 0 1 1 0 0 |

➢**Funtion and Action examples**

| The value 16 # 50 ( BYTE ) right shifted three | description |
|---|---|
|  | When the contact C300 is ON when this instruction is executed, |

| | |
|---|---|
| | The address V538 all the bits to the right 3 bits:<br><br>Execution result: 1 6#50 moves 3 bit output to the right 16 #0A : |

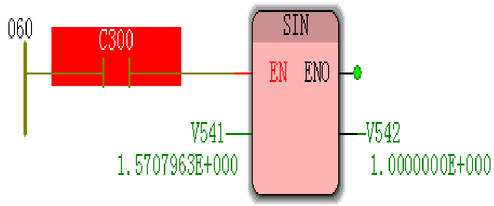
## 10.1.39 SIN (sinusoidal command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SIN instruction is used to find the sine of the input value. | |
| LD IN |  |
| SIN | |
| ST OUT | |
| ST programming language | |
| OUT:=SIN ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**Data type processed by SIN instruction**

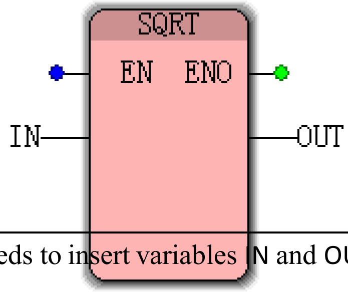
| Input and output | type of data | description |
|---|---|---|
| IN | REAL | Input, the angle is expressed in radians |
| OUT | REAL | Output |

**➢Funtion and Action examples**

| Seeking the SIN ( the value | description |
|---|---|
|  | When the contact C300 is ON when this instruction is executed,<br>Execution result: SIN（π/2）=1 |

## 10.1.40 SQRT (square root instruction )

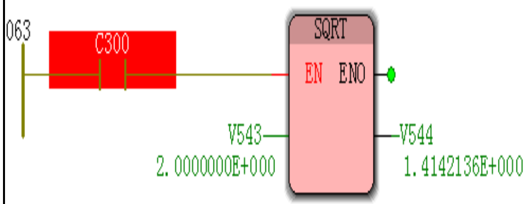
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SQRT instruction is used to find the square root of the input value. | |

| | |
|---|---|
| LD IN |  |
| SORT | |
| ST OUT | |
| ST programming language | |
| OUT:=SORT ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

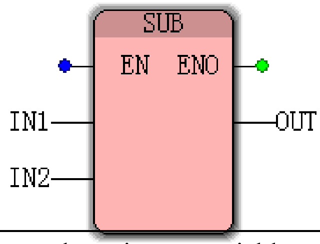
➢**SQRT data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | REAL | Input |
| OUT | REAL | Output |

➢**Funtion and Action examples**

| Find the value | description |
|---|---|
|  | When the contact C300 is ON when this instruction is executed : execution result = 1.4142136E + 000 |

# 10.1.41 SUB (Subtraction Instruction )

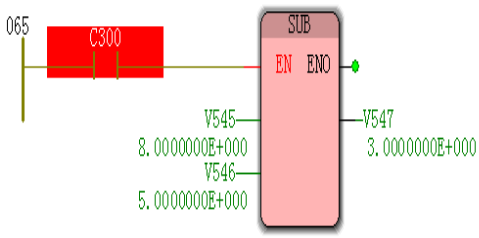
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SUB instruction is used to find the difference between two input values. | |
| LD IN1 |  |
| SUB IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=IN1-IN2 | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**Data type processed by SUB instruction**

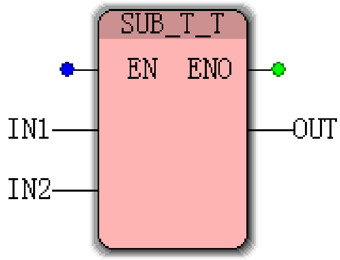
| Input and output | type of data | description |
|---|---|---|

| | | |
|---|---|---|
| IN1 | ANY_NUM | First input |
| IN2 | ANY_NUM | Second input |
| OUT | ANY_NUM | Output, OUT=IN1 - IN2 |

**➢Funtion and Action examples**

| Find the value of floating point number 8.0 minus 5.0 | description |
|---|---|
|   | When the contact C300 is ON , the instruction is executed : the value of the address V545 is subtracted from the value of V546 and stored in the address of V547 :<br><br>Execution result: V545-V546 = V547 |

## 10.1.42 SUB_T_T (Time Subtraction Instruction )

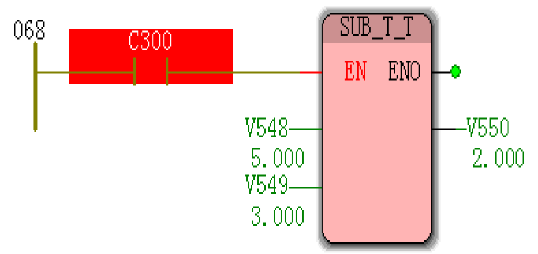
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SUB_T_T instruction is used to find the difference between two time input values. | |
| LD IN1<br>SUB IN2<br>ST OUT<br>ST programming language<br>OUT:=SUB_T_T ( 1N1 , IN2 ) |   |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

**➢SUB_T_ data processing instruction type**

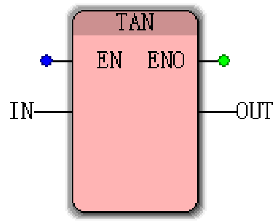
| Input and output | type of data | description |
|---|---|---|
| IN1 | TIME | First input |
| IN2 | TIME | Second input |
| OUT | TIME | Output, OUT=IN1 - IN2 |

**➢Funtion and Action examples**

| Time T # T # 5S-3S 's value | description |
|---|---|

| | |
|---|---|
|   | When the contact C300 is ON , the instruction is executed : the value of the address V548 is subtracted from the value of V549 and stored in the address of the V550 :<br>Execution result: V54 8 -V54 9=V550 |

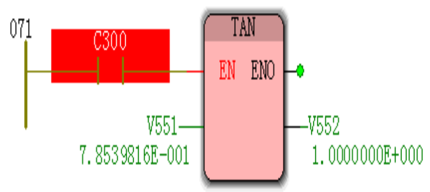
## 10.1.43 TAN (tangential command )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The TAN instruction is used to find the tangent of the input value.</td></tr>
<tr><td>LD IN</td><td rowspan="5"> </td></tr>
<tr><td>TAN</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=TAN ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

**Data type processed by TAN instruction**

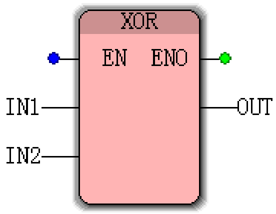
| Input and output | type of data | description |
|---|---|---|
| IN | REAL | Input, the angle is expressed in radians |
| OUT | REAL | Output |

➢**Funtion and Action examples**

| Find the value | description |
|---|---|
|   | When the contact C300 is ON when this instruction is executed perform: result : = 1 |

## 10.1. 44 XO R (Logical XOR instruction )

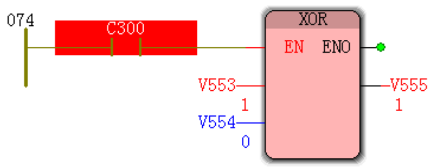
| IL programming language | LD, FBD programming language |
|---|---|

<table>
<tr><td colspan="2">Function: The XOR instruction is used to logically perform an exclusive OR operation on the input value.</td></tr>
<tr><td>LD IN1<br>XOR IN2<br>ST OUT<br>ST programming language<br>OUT:= ( IN1 ) XOR ( IN2 )</td><td>
</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

**Data type processed by XOR instruction**

| Input and output | type of data | description |
|---|---|---|
| IN1 | ANY_BIT | First input |
| IN2 | ANY_BIT | Second input |
| OUT | ANY_BIT | Logical exclusive OR operation<br>IN1=0, IN2=0, OUT is 0;<br>IN1=0, IN2=1, OUT is 1;<br>IN1=1, IN2=0, OUT is 1;<br>IN1=1, IN2=1, OUT is 0; |

**➢Funtion and Action examples**

<table>
<tr><td>Find the value of the exclusive OR of the address V553 ( BOOL ) and V554 ( BOOL )</td><td>description</td></tr>
<tr><td>
</td><td>When the contact C300 is ON when this instruction is executed : When the V553 ( BOOL ) and V554 ( BOOL ) values of different time , the output is 1<br><br>Execution result : V553 and V554 are different output 1</td></tr>
</table>

## 1 0.2 function block

A function block is a program organization unit POU with multiple input and output parameters , which have internal memory, and the return value of the function block depends on the value of its internal storage unit. The abbreviation of the function block is FBD different from the previous function. The function block must be instantiated. The instance name can be the default name or the default name. The instance name must be unique within the POU . In FBDAnd LD programming, this instance name appears in the upper part of the function block.

The following function blocks can be used during MULTIPROG programming

○ bistable function block
○ Pulse edge detection function block
○ counter function block
○ timer function block

Command functions included in the block ( in the editor wizard from the drop-down list " function block ")

| name | description |
|---|---|
| CTD | Down counter |
| CTU | Increment counter |
| CTUD | Up / down counter |
| F_TRIG | Falling edge detection |
| R_TRIG | Rising edge detection |
| RS | Reset priority |
| SR | Set priority |
| TOF | Disconnect delay timer |
| TON | On-delay timer |
| T P | pulse |

## 10.2.1 CTD (Decrement Counter Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The TD instruction is used to count down the input. When the LOAD terminal is FALSE , if there is a rising edge at the CD input, the CV terminal is decremented by 1 . If the count value CV reaches the lower limit value 0 of the counter , a TRUE signal is issueDAt the Q output , and the CTD function block stops counting. When the LOAD terminal is TRUE , the counter stops counting anDAssigns the value of the PV input to the CV terminal. | |

| | |
|---|---|
| LD varl | |
| ST CTD_1.CD | |
| LD var2 | |
| ST CTD_1.L0AD | |
| LD var3 | |
| ST CTD_1.PV | |
| CAL CTD_1 | |
| LD CTD_1.Q | |
| ST var4 | |
| LD CTD_1.CV | |
| ST var5 | |
| ST programming language | |
| CTD_1 ( CD:=var1 , LOAD:=var2, PV:=var3 ) | |
| Var4:=CTD_1.Q | |
| Vase:=CTD_1.CV | |
| Note : IL, ST language programming needs to insert the variable vlar1~var5 constant in the current POU variable worksheet | |

**数据 Data type processed by CTD instruction**

| Input | type of data | description |
|---|---|---|
| CD | BOOL | If the CD has a rising edge , the CV is decremented by 1. |
| LOAD | BOOL | When LOAD is FALSE , the count is started, it is TRUE , the PV is assigned to CV , and the counter is initialized. |
| PV | INT | Count downNStart value |
| Output | type of data | description |
| Q | BOOL | When CV=0 , O=1 |
| CV | INT | Count value |

➢**Funtion and Action examples**

| Q output 1 when contact C000 is | description |
|---|---|

| turned from OFF to ON five times | |
|---|---|
| | This counter function block decrements the count. Assuming a rising edge at the CD input and LOAD = FALSE , the CV is decremented by one. If the final value of the counter ( PV ) is reached , a TRUE signal is sent at the Q output and the function block stops counting.<br>If LOAD=TRUE , the counter is initialized by the value of the PV input. In order to enable the counting process, the LOAD input must be FALSE . Otherwise the counter will be reinitialized. |

## 10.2.2 CTU (Incremental Counter Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The CTU instruction is used to count the input terminal. When the RESET terminal is FLSE , if there is a rising edge at the CU input, the CV terminal is incremented by 1 . If the count value CV reaches the upper limit value PV of the counter , a TRUE signal is issueDAt the Q output , and the CTU function block stops counting. When the RESET terminal is TRUE , the counter stops counting and the CV terminal is cleared. | |
| LD varl | |
| ST CTU_1.CU | |
| LD var2 | |
| ST CTU_1.RESET | |
| LD var3 | |
| ST CTU_1.PV | |
| CAL CTU_1 | |
| LD CTU_1.Q | |
| ST var4 | |
| LD CTU_1.CV | |
| ST var5 | |
| ST language | |
| CTU_1 ( CD:=var1 , LOAD:=var2, PV:=var3 ) | |
| Var4:=CTU_1.Q | |
| Var5:=CTU_1.CV | |

| |
|---|
| Note : IL, ST language programming needs to insert variables var1 ~ var5 or use constants in the current POU variable worksheet |

**Data type processed by CTU instruction**

| Input | type of data | description |
|---|---|---|
| CU | BOOL | If the CU has a rising edge , CV adds 1 |
| RESET | BOOL | When RESET is FALSE , the count is started ;<br>When TRUE , the CV is cleared, the counter is initialized, and the Q terminal is reset. |
| PV | INT | Incremental count upper limit |
| Output | type of data | description |
| Q | BOOL | When CV=PV , Q=1 |
| CV | INT | Count value |

➢**Funtion and Action examples**

| Q output 1 when contact C002 is turned from OFF to ON five times | description |
|---|---|
| | This counter function block counts up. Assuming a rising edge at the CU input and RESET = FALSE , the CV is incremented by one. If the final value of the counter ( PV ) is reached , a TRUE signal is sent at the Q output and the function block stops counting.<br>If RESET = TRUE , the counter is initialized with 0 . In order to enable the counting process, the RESET input must be FALSE . Otherwise the counter will always be reinitialized. |

# 10.2.3 CTUD (increasing or decreasing bidirectional counter command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The CTUD instruction is used to increment or decrement the input. When the RESET terminal and the LOAD terminal are both FALSE , the counting is allowed : if a rising edge occurs at the CU input terminal, the CV terminal is incremented by one; if a rising edge occurs at the CD input terminal, the CV terminal is decremented by one; when CV=PV , then QU=1, the CTUD function block stops counting up ; if CV=0 , then QD=1, the CTUD function block stops counting down. When the RESET terminal is TRUE , the counter stops incrementing and counting down, and the CV terminal is cleared . When the LOAD terminal is TRUE , the counter stops incrementing and counting down, and the PV value is assigned to the CV terminal. | |
| LD varl | |
| ST CTUD_1.CU | |
| LD var2 | |
| ST CTUD_1.CD | |
| LD var3 | |
| ST CTUD_1.RESET | |
| LD var4 | |
| ST CTUD_1.LOAD | |
| LD var5 | |
| ST CTUD_1.PV | |
| CAL CTUD_1 | |
| LD CTUD_1.QU | |
| ST var6 | |
| LD CTUD_1.QD | |
| ST var7 | |
| LD CTUD_1.CV | |
| ST var8 | |
| ST programming language | |
| CTUD_1 ( CU:=var1 , CD:=var2, RESET:= var3, LOAD:=var4, PV:=var5 ) ; | |
| Var6:=CTUD_ 1.QU; | |
| Var7:=CTUD_ 1.QD; | |
| Ar8:=CTUD_ 1.CV; | |
| Note : IL, ST language programming needs to insert the variable val ~ var8 or use constants in the variable worksheet of the current POU | |

**数据 Data type processed by CTU instruction**

| Input | type of data | description |
|---|---|---|
| CU | BOOL | If the CU has a rising edge , CV adds 1 |
| CD | BOOL | If the CD has a rising edge , the CV is decremented by 1. |
| RESET | BOOL | When RESET is FALSE , the count is started. When TRUE , the CV is cleared to zero and the counter is initialized. |
| LOAD | BOOL | When LOAD is FALSE , the count is started. When TRUE , the PV is assigned to CV and the counter is initialized. |
| PV | INT | Count the upper limit or start value |
| Output | type of data | description |
| QU | BOOL | When CV=PV , QU=1 |
| QD | BOOL | When CV=0 , QD=1 |
| CV | INT | Count value |

➤**Funtion and Action examples**

| When the CU has a rising edge, the value is incremented by 1 when the CD has a falling edge current value minus 1. | description |
|---|---|
| | This counter function block increments or decrements the count. Assuming a rising edge at the CU input, the CV is incremented by one. Assuming a rising edge at the CD input, the CV is decremented by one. If CV = PV , the OU is set to TRUE . If CV = PV , the OU is set to TRUE .<br>If RESET = TRUE , the counter is initialized to 0 . If LOAD = TRUE , the counter is initialized to PV . In order to enable the counting process, both the RESET and LOAD inputs must be FALSE . Otherwise the counter will be reinitialized. |

# 10.2.4 F_TRIG (falling edge detection command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: F _TRIG instruction is used to detect the falling edge of the input . If a falling edge is detecteDAt the input CLK , the output Q will change from FALSE to TRUE until the next scan of this command, the Q output will remain is TRUE | |
| LD varl | |
| ST F_TRIG_1.CLK | |
| CAL F_TRIG_1 | |
| LD F_TRIG_1.Q | |
| ST var2 | |
| ST programming language | |
| F_TRIG_ 1 ( CLK:=var1 ) | |
| Var2:=F_TRIG_1.Q | |
| Note : IL, ST language programming needs to insert variables varl1~var2 or use constants in the current POU variable worksheet | |

➢ **the F_TRIG data processing instruction type**

| parameter | type of data | description |
|---|---|---|
| CLK | BOOL | Falling edge is valid |
| Q | BOOL | When CLK has a falling edge, Q= changes from 0 to 1, until the next scan to this instruction |

# 10.2.5 R_TRIG (rising edge detection instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: R _TRIG instruction is used to detect the rising edge of the input . If a rising edge is detecteDAt the input CLK , the output Q will change from FALSE to TRUE until the next scan to this command, the Q output will remain Is TRUE . | |
| LD var1 | |
| ST R_TRIG_1.CLK | |
| CAL R_TRIG_1 | |
| LD R_TRIG_1.Q | |
| ST var2 | |

| | |
|---|---|
| ST programming language | |
| R_TRIG_1 ( CLK:=var1 ) | |
| Var2:=R_TRIG_1.Q | |
| Note : IL, ST language programming needs to insert variables varl1~var2 or use constants in the current POU variable worksheet | |

**数据 Data type processed by CTU instruction**

| parameter | type of data | description |
|---|---|---|
| CLK | BOOL | Valid on rising edge |
| Q | BOOL | When CLK has a rising edge, Q= changes from 0 to 1 until the next scan to this instruction. |

## 10.2.6 RS ( RS Trigger Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The RS instruction is used to implement the function of the RS trigger. If the SET terminal is TRUE and the RESET terminal is FALSE , the output terminal Q1 is set. Even if SET becomes FALSE, Q1 remains set. If RESET1=TRUE , Q1 is reset regardless of whether the SET terminal is TRUE or FALSE . Even if RESET1 changes to FALSE, Q1 remains in the reset state. | |
| LD var1 | |
| ST RS_1.SET | |
| LD var2 | |
| ST RS_1.RESETI | |
| AL RS_1 | |
| LD RS_1.Q1 | |
| ST var3 | |
| ST programming language | |
| RS_1 ( SET:=var1 , RESET:=var2 ) | |
| X ar3:=RS_1.Q1 | |
| Note : IL, ST language programming need to insert variables valll1~var3 or use constants in the current POU variable worksheet | |

**数据 Data type processed by RS instruction**

| parameter | type of data | description |
|---|---|---|
| SET | BOOL | Position |

| | | |
|---|---|---|
| RESET1 | BOOL | Reset |
| Q1 | BOOL | SET=0, RESET1=0, Q1 remains in the last state ;<br>SET=1, RESET1=0, Q1=1;<br>SET=0, RESET1=1, Q1=0; |

➢**Funtion and Action examples**

| When the contacts C008 and C010 are ON at the same time, the reset priority output Q1 0 | description |
|---|---|
| | This bistable function block implements a priority reset at the Q1 output. If the input SET = TRUE , the output Q1 is set. Even if SET is FALSE , Q1 remains set. If RESET1 = TRUE , Q1 is reset. If both inputs are TRUE , the Q1 output is set to FALSE by RESET1 . |

## 10.2.7 SR ( SR Trigger Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The RS instruction is used to implement the function of the RS trigger. If the SET1 terminal is TRUE , the output terminal Q1 is set regardless of whether the RESET terminal is TRUE or FALSE . Even if SET1 becomes FALSE, Q1 remains set. If the RESET terminal is TRUE and the SET1 terminal is FALSE, the Q1 terminal is reset. Even if RESET becomes FALSE, Q1 remains in the reset state. | |
| LD var1 | |
| ST SR_1.SET1 | |
| LD var2 | |
| ST SR_1.RESET | |
| CAL SR_1 | |
| LD SR_1.Q1 | |
| ST var3 | |
| ST programming language | |
| SR_1 ( SET1:=var1 , RESET:=var2 ) | |
| Var3:=SR_1.Q1 | |
| Note : IL, ST language programming need to insert variables valll1~var3 or use constants in the current POU variable worksheet | |

**数据 Data type processed by RS instruction**

| parameter | type of data | description |
|---|---|---|
| SET1 | BOOL | Position |
| RESET | BOOL | Reset |
| Q1 | BOOL | result<br>SET1=0, RESET=0, Q1 remains in the last state ;<br>SET1=1, RESET=0, Q1=1;<br>SET1=0, RESET=1, Q1=0;<br>SET1=1, RESET=1, Q1=1; |

➢**Funtion and Action examples**

| When the contacts C013 and C015 are ON at the same time, the set priority Q1 output 1 | description |
|---|---|
| | This bistable function block implements the priority setting of the Q1 output. If SET1=TRUE is entered , the Q1 output is set. Even if SET is FALSE , Q1 remains set. If RESET = TRUE , the Q1 output is reset. If both inputs are TRUE , the Q1 output is set to TRUE by SET1 . |

## 10.2.8 TOF (Delayed Off Timer Instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The TOF instruction is used to implement the delay disconnect function. If the input terminal IN is TRUE , the output terminal Q immediately becomes TRUE; if the input terminal IN is changed from TRUE to FALSE , the output terminal Q will be delayed by a certain time and then by the TRUE. It becomes FALSE , the delay time is the value of PT , and the ET end records the time between the time when the IN changes to FALSE and the time when the Q end changes to TRUE . | |
| LD var1 | |
| ST TOF_1.IN | |
| LD var2 | |
| ST TOF_1.PT | |
| CAL TOF_1 | |
| LD TOF_1.Q | |
| ST var3 | |
| LD TOF_1.ET | |

| | |
|---|---|
| ST var4 | |
| ST programming language | |
| TOF_1 ( IN:=var1 , PT:=var2 ) | |
| Var3:=TOF_ 1.Q | |
| Var4:=TOF_1.ET | |
| Note : IL, ST language programming needs to insert variables valll1~var4 or use constants in the current POU variable worksheet | |

数据 Data type processed by TOF instruction

| parameter | type of data | description |
|---|---|---|
| IN | BOOL | Enable input |
| PT | TIME | To Q delay disconnecteDAsk end |
| Q | BOOL | Result IN=1, Q=1; IN=0 , after delaying PT , Q changes from 1 to 0. |
| ET | TIME | Timing time from when IN changes to FALSE to when the Q end changes to TRUE |

➤**Funtion and Action examples**

| When the contact C018ON , Q immediately outputs 1 , when the contact is turned from ON to OFF , Q delays PV (set value) and outputs 0. | description |
|---|---|
| | If the input IN changes from TRUE to FALSE , it is turned off after delaying the length of time in the input PT . After the length of the PT value, the Q value is set to FALSE . The process time interval is displayed on the output ET . |

# 1 0.2.9 TON (delay-on timer command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The TON instruction is used to implement the delay on function. If the input terminal IN is FALSE , the output terminal Q will immediately become FALSE; if the input terminal IN changes from FALSE to TRUE , the output terminal Q will be delayed by a certain time and then changed by FALSE. For TRUE , this delay time is the value of PT , and the ET end records the time between the time when the IN changes to FALSE and the time when the Q end changes to TRUE . | |
| LD var1 | |
| ST TON_1.IN | |
| LD var2 | |
| ST TON_1.PT | |
| CALTON_1 | |
| LD TON_1.Q | |
| ST var3 | |
| LD TON_1.ET | |
| ST var4 | |
| ST programming language | |
| TON_1 ( 1N:=var1 , PT:=var2 ) | |
| Var3:=TON_1.Q | |
| Var4:=TON_.ET | |
| Note : IL, ST language programming needs to insert variables valll1~var4 or use constants in the current POU variable worksheet | |

**数据 Data type processed by TON instruction**

| parameter | type of data | description |
|---|---|---|
| IN | BOOL | Enable input |
| PT | TIME | To Q delay turned end Q, such as T # 5S |
| Q | BOOL | Result : If IN=0 , then Q=0; if IN=1 , after delaying PT , Q changes from 0 to 1 |
| ET | TIME | Timing time from when IN changes to TRUE to when the Q end changes to TRUE |

➢**Funtion and Action examples**

| Start when contact C020 is ON , Q delay PV (set value) output 1 | description |
|---|---|

| | |
|---|---|
| | If the input IN changes from TRUE to FALSE , it is turned on after delaying the input of the PT . After the length of the PT value, the Q value is set to TRUE . The process time interval is displayed on the output ET . |

## 10.2.10 TP (pulse command )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The :TP command is used to realize the function of a certain width pulse. If the input terminal IN changes from FALSE to TRUE , the output terminal Q generates a pulse with a time interval of PT . If the input IN becomes FALSE again during the PT time , the output Q still produces a pulse of PT width. The ET end records the time between the time when the IN changes to FALSE and the time when the Q end changes to TRUE .</td></tr>
<tr><td>LD var1</td><td rowspan="13"></td></tr>
<tr><td>ST TP_1.IN</td></tr>
<tr><td>LD var2</td></tr>
<tr><td>ST TP_1.PT</td></tr>
<tr><td>CAL TP_1</td></tr>
<tr><td>LD TP_1.Q</td></tr>
<tr><td>ST var3</td></tr>
<tr><td>LD TP_1.ET</td></tr>
<tr><td>ST var4</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>TP_1 ( IN:=varl , PT:=var2 )</td></tr>
<tr><td>Var3:=TP_1.Q</td></tr>
<tr><td>Var4:=TP_1.ET</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables valll1~var4 or use constants in the current POU variable worksheet</td></tr>
</table>

**Data type processed by TON instruction**

| parameter | type of data | description |
|---|---|---|
| IN | BOOL | The rising edge of IN is valid |
| PT | TIME | Pulse time interval |
| Q | BOOL | As a result, Q produces a pulse of PT width at the rising edge of IN . |

| | | |
|---|---|---|
| ET | TIME | The timing from the time when IN changes to TRUE to the time when the Q end changes to TRUE , the state change of IN does not work for Q. |

# 1 0.3 type conversion FU

The type conversion function, referred to as type conversion FU , converts one type of data into another type of data, so it has an input parameter anDAn output parameter. During the MULTIPROG programming process, the following types can be used to convert FU

- BYTE type BCD data conversion
- WORD type BCD data conversion
- DWORD type BCD data conversion
- BOOL type data conversion
- BYTE type data conversion
- WORD type data conversion
- DWORD type data conversion
- Conversion of SINT data
- Conversion of INT data
- Conversion of DINT type data
- Conversion of USINT type data
- UINT type data conversion
- Conversion of UDINT type data
- Conversion of LREAL type data
- Conversion of REAL type data
- TRUNC decimal rounding

The instructions contained in the function ( in the Edit Wizard, select from the drop-down list " Type Conversion FU ")

| Sort | Funtions | | |
|---|---|---|---|
| BYTE type BCD data conversion | B_BCD_TO_SINT | B_BCD_TO_INT | B_BCD_TO_DINT |
| WORD type BCD data conversion | W_BCD_TO_SINT | W_BCD_TO_INT | W_BCD_TO_DINT |
| DWORD type BCD data conversion | D_BCD_TO_SINT | D_BCD_TO_INT | D_BCD_TO_DINT |
| Conversion of BCD type data | BCD_TO_DINT | | |
| TIME type data conversion | TIME_TO_DINT | | |
| BOOL type data conversion | BOOL_TO_BYTE | BOOL_TO_WORD | BOOL_TO_DWORD |
| | BOOL_TO_SINT | BOOL_TO_INT | BOOL_TO_DINT |
| | BOOL_TO_USINT | BOOL_TO_UINT | BOOL_TO_UDINT |
| | BOOL_TO_REAL | BOOL_TO_LREAL | |
| BYTE type data conversion | BYTE_TO_BOOL<br>BYTE_TO_WORD<br>BYTE_TO_DWORD | BYTE_TO_BOOL<br>BYTE_TO_WORD<br>BYTE_TO_DWORD | BYTE_TO_BOOL<br>BYTE_TO_WORD<br>BYTE_TO_DWORD |

| | | | |
|---|---|---|---|
| WORD type data conversion | WORD_TO_BOOL<br>WORD_TO_BYTE<br>WORD_TO_DWORD | WORD_TO_BOOL<br>WORD_TO_BYTE<br>WORD_TO_DWORD | WORD_TO_BOOL<br>WORD_TO_BYTE<br>WORD_TO_DWORD |
| DWORD type data conversion | DWORD_TO-BOOL<br>DWORD_TO_BYTE<br>DWORD_TO_WORD | DWORD_TO-BOOL<br>DWORD_TO_BYTE<br>DWORD_TO_WORD | DWORD_TO-BOOL<br>DWORD_TO_BYTE<br>DWORD_TO_WORD |
| SINT type data conversion | SINT_TO_B_BCD | SINT_TO_W_BCD | SINT_TO_D_BCD |
| | SINT_TO_BOOL | SINT_TO_BYTE | SINT_TO_WORD |
| | SINT_TO_DWORD | SINT_TO_INT | SINT_TO_DINT |
| | SINT_TO_USINT | SINT_TO_UINT | SINT_TO_UDINT |
| | SINT_TO_REAL | SINT_TO_LREAL | |
| INT type data conversion | INT_TO_B_BCD | INT_TO_W BCD | INT_TO_D_BCD |
| | INT_TO_BOOL | INT_TO_BYTE | INT_TO_WORD |
| | INT_TO_DWORD | INT_TO_SINT | INT_TO_DINT |
| | INT_TO_USINT | INT_TO_UINT | INT_TO_UDINT |
| | INT_TO_REAL | INT_TO_LREAL | |
| DINT type data conversion | DINT_TO_B_BCD | DINT_TO_W_BCD | DINT_TO_D_BCD |
| | DINT_TO_BOOL | DINT_TO_BYTE | DINT_TO_WORD |
| | DINT_TO_DWORD | DINT_TO_SINT | DINT_TO _INT |
| | DINT_TO_USINT | DINT_TO_UINT | DINT_TO_UDINT |
| | DINT_TO_REAL | DINT_TO_LREAL | DINT_TO_BCD |
| | DINT_TO_TIME | | |
| USINT type data conversion | USINT_TO_BOOL | USINT_TO_BYTE | USINT_TO_WORD |
| | USINT_TO_DWORD | USINT_TO_SINT | USINT_TO_INT |
| | USINT_TO_DINT | USINT_TO_UINT | USINT_TO_UDINT |
| | USINT_TO_REAL | USINT_TO_LREAL | |
| UINT type data conversion | UINT_TO_BOOL | UINT_TO_BYTE | UINT_TO_WORD |
| | UINT_TO_DWORD | UINT_TO_SINT | UINT_TO_INT |
| | UINT_TO_DINT | UINT_TO_USINT | UINT_TO_UDINT |
| | UINT_TO_REAL | UINT_TO_LREAL | |
| UDINT type data conversion | UDINT_TO_BOOL | UDINT_TO_BYTE | UDINT_TO_WORD |
| | UDINT_TO_DWORD | UDINT_TO_SINT | UDINT_TO_INT |
| | UDINT_TO _DINT | UDINT_TO_USINT | UDINT_TO_UINT |
| | UDINT_TO_REAL | UDINT_TO_LREAL | |
| LREAL type data | LREAL_TO_BOOL | LREAL_TO_BYTE | LREAL_TO_WORD |

| | | | |
|---|---|---|---|
| conversion | LREAL_TO_DWORD | LREAL_TO_SINT | LREAL_TO_INT |
| | LREAL_TO_DINT | LREAL_TO_USINT | LREAL_TO_UINT |
| | LREAL_TO_UDINT | LREAL_TO_REAL | |
| REAL type data conversion | REAL_TO_BOOL | REAL_TO_BYTE | REAL_TO_WORD |
| | REAL_TO_DWORD | REAL_TO_SINT | REAL_TO_INT |
| | REAL_TO_DINT | REAL_TO_USINT | REAL_TO_UINT |
| | REAL_TO_UDINT | REAL_TO_LREAL | |
| TRUNC type data conversion | TRUNC | TRUNC_SINT | TRUNC_INT |
| | TRUNC_DINT | | |

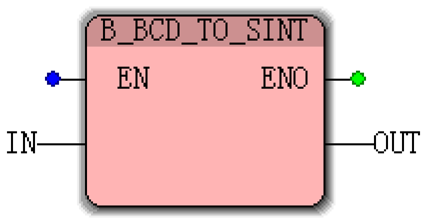
## 10.3.1 Conversion of BYTE type BCD data

| Type conversion FU | Features |
|---|---|
| BYTE type BCD data conversion | The conversion of BYTE type BCD data includes the following three instructions : B_BCD_TO_SINT, B_BCD_TO_INT and B_BCD_TO_DINT . These three instructions convert a BCD (binary encoded decimal) input value of a BYTE data type into an output value of the SINT , INT, and DINT data types, respectively. |

**BYTE type BCD data conversion instruction**

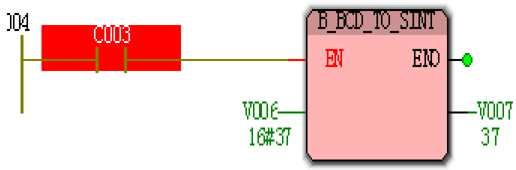
| instruction | input value | output value | description |
|---|---|---|---|
| B_BCD_TO_SINT | BYTE type BCD code | SINT | Input value range BCD code 16#0099; The corresponding output values SINT, INT and DINT are 0~99 . |
| B_BCD_TO_INT | BYTE type BCD code | INT | |
| B_BCD_TO_DINT | BYTE type BCD code | DINT | |

**usage (take B_BCD_TO_SINT as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| B_BCD_TO_SINT | |
| ST OUT | |
| ST programming language | |
| OUT:=B_ BCD_TO_ SINT ( IN ) | |

| |
|---|
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet |

➢**Funtion and Action examples**

| 16#37 changed to short type | description |
|---|---|
|  | When the contact C003 is ON time , the instruction is executed; address as V006 in the 16 hexadecimal into numbers short integer , stored in the V007 in ;<br><br>Execution of results : 16 # 37----37 |

## 10.3.2 Conversion of WORD type BCD data

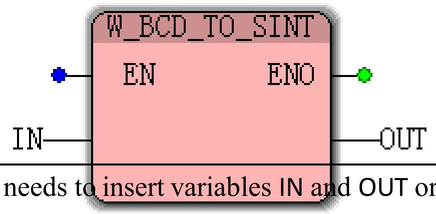
| Type conversion FU | Features |
|---|---|
| WORD type BCD data conversion | The conversion of WORD type BCD data includes the following three instructions : W_BCD_TO_SINT, W_BCD_TO_INT and W_BCD_TO_DINT . These three instructions convert a BCD ( binary encoded decimal ) input value of a WORD data type into an output value of the SINT , INT, and DINT data types, respectively. |

➢ **WORD Type BCD conversion instruction data**

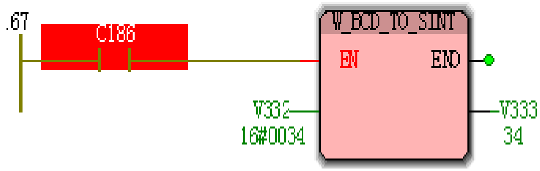
| instruction | input value | output value | description |
|---|---|---|---|
| W_BCD_TO_SINT | WORD type BCD code | SINT | Input value BCD code 16 # 0127 , the output value corresponding to SINT to 127 , and then input the output is increased - . 1; the input values BCD code 16 # 16 # 0000 ~ 9999 , the output value corresponding to INT, DINT are 0 to 9999 . |
| W_BCD_TO_INT | WORD type BCD code | INT | |
| W_BCD _TO_DINT | WORD type BCD code | DINT | |

➢ **usage (take W_BCD_TO_SINT as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN | |

| | |
|---|---|
| W BCD TO SINT |  |
| ST OUT | |
| ST programming language | |
| OUT:=W_BCD_TO_SINT ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**Funtion and Action examples**

| 16 # 0010 turn into a short integer | description |
|---|---|
|  | When the contact C186 is ON time , the instruction is executed; address as V332 in the 16 hexadecimal into numbers short integer , stored in the V333 in ;<br><br>Execution of Results : 16 # 0034 revolutions was 34 |

## 10.3.3 Conversion of DWOR D -type BCD data

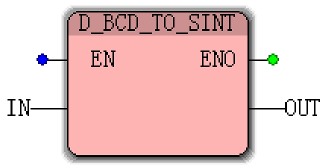
| Type conversion FU | Features |
|---|---|
| DWORD type BCD data conversion | The conversion of DWORD type BCD data includes the following three instructions : D_BCD_TO_SINT, D_BCD_TO_INT and D_BCD_TO_DINT . These three instructions convert a BCD ( binary encoded decimal ) input value of ADWORD data type into an output value of the SINT , INT, and DINT data types, respectively. |

**DWORD type BCD data conversion instruction**

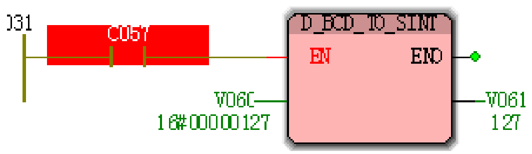
| instruction | input value | output value | description |
|---|---|---|---|
| D_BCD_TO_SINT | DWORD type BCD code | SINT | Enter the value BCD code 16#00000000~16#00000127 , the corresponding output value SINT is 0~127 , the input is increased by -1; the input value BCD code 16#00000000~16# 00032767 , the output value INT is 032,767 , and the output is increased by -1; Input |
| D_BCD_TO_INT | DWORD type BCD code | INT | |
| D_BCD_TO_DINT | DWORD type BCD code | DINT | |

| | | | |
|---|---|---|---|
| | | | value BCD code 16#00000000~16# 99999999 , output value DINT 0~99999999 . |

**usage (take D_BCD_TO_SINT as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| D_BCD_TO_SINT | |
| ST OUT | |
| ST programming language | |
| OUT:=D_ BCD_TO_ SINT ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**➢Funtion and Action examples**

| 16#00000044 turns into a short integer | description |
|---|---|
|  | When the contact C057 is ON time ; execute the instruction ; address as V060 in the 16 hexadecimal into numbers short integer , stored in the V061 in ;<br><br>Execution result: 16 #0000127 converted to short integer 127 |

## 10.3.4 Conversion of BCD type data

| Type conversion FU | Features |
|---|---|
| Conversion of CD type data | The BCD type data conversion instruction BCD_TO _DINT is used to convert a BCD ( binary coded decimal number ) input value into an output value of ADINT data type. This instruction is the same as D_BCD_TO_DINT . See D_BCD_TO_DINT for details . |

## 10.3.5 Conversion of BOOL type data

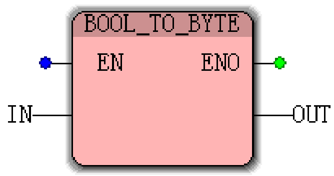
| Type conversion FU | Features |
|---|---|

| | |
|---|---|
| BOOL type data conversion | BOOL type data conversion has 11 instructions, which can convert BOOL type data into BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL and LREAL . |

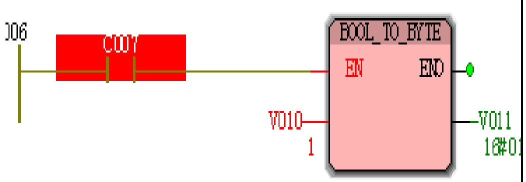
**BOOL type data conversion instruction**

<table>
<tr><th>instruction</th><th>input value</th><th>output value</th><th>description</th></tr>
<tr><td>BOOL_TO_BYTE</td><td>BOOL</td><td>BYTE</td><td rowspan="11">Input value range FALSE or TRUE<br>When the input is FALSE , the output is 0;<br>When the input is TRUE , the output is 1;</td></tr>
<tr><td>BOOL_TO_WORD</td><td>BOOL</td><td>WORD</td></tr>
<tr><td>BOOL_TO_DWORD</td><td>BOOL</td><td>DWORD</td></tr>
<tr><td>BOOL_TO_SINT</td><td>BOOL</td><td>SINT</td></tr>
<tr><td>BOOL_TO_INT</td><td>BOOL</td><td>INT</td></tr>
<tr><td>BOOL_TO_DINT</td><td>BOOL</td><td>DINT</td></tr>
<tr><td>BOOL_TO_USINT</td><td>BOOL</td><td>USINT</td></tr>
<tr><td>BOOL_TO_UINT</td><td>BOOL</td><td>UINT</td></tr>
<tr><td>BOOL_TO_UDINT</td><td>BOOL</td><td>UDINT</td></tr>
<tr><td>BOOL_TO_REAL</td><td>BOOL</td><td>REAL</td></tr>
<tr><td>BOOL_TO_LREAL</td><td>BOOL</td><td>LREAL</td></tr>
</table>

**usage (take BOOL_TO_BYTE as an example)**

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td>LD IN</td><td rowspan="5">
</td></tr>
<tr><td>BOOL_TO_BYTE</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=BOOL_TO_BYTE ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

**➢Funtion and Action examples**

<table>
<tr><th>The BOOL turn bytes</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C007 is ON time , the instruction is executed; address V010 is TRUE when , into the byte type, stored in the V 011 In ;<br><br>Execution result ; V010 ( TRUE ) switch 16 # 01 save to V011 in ;</td></tr>
</table>

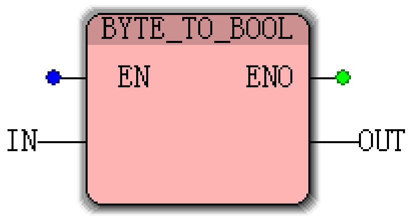
## 10.3.6 Conversion of BYTE type data

| Type conversion FU | Features |
|---|---|
| BYTE type data conversion | BYTE type data conversion has 11 instructions, which can convert BYTE type data into BOOL, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL and LREAL . |

**BYTE type data conversion instruction**

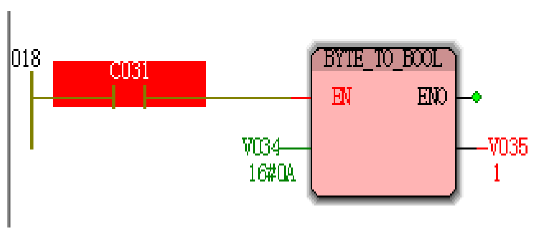
| instruction | input value | output value | description |
|---|---|---|---|
| BYTE_TO_BOOL | BYTE | BOOL | The input value ranges from 0 to 255; the output is BOOL type : only when the input is 0 , the output is FALSE, and in other cases, the output is TRUE; The output is SINT type : input 0~127 corresponds to output 0~127 , input 128~255 corresponds to output -128~-1; when output is WORD, DWORDAnd other types, the output is equal to input. |
| BYTE _O_WORD | BYTE | WORD | |
| BYTE_TO_DWORD | BYTE | DWORD | |
| BYTE_TO_SINT | BYTE | SINT | |
| BYTE_TO_INT | BYTE | INT | |
| BYTE_TO_DINT | BYTE | DINT | |
| BYTE_TO_USINT | BYTE | USINT | |
| BYTE_TO_UINT | BYTE | UINT | |
| BYTE_TO_UDINT | BYTE | UDINT | |
| BYTE_TO_REAL | BYTE | REAL | |
| BYTE_TO_LREAL | BYTE | LREAL | |

**usage (take BYTE_TO_BOOL as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| BYTE_TO_BOOL | |
| ST OUT | |
| ST programming language | |
| OUT:=BYTE_TO_BOOL ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**➢Funtion and Action examples**

| Requirements : 16 # 0A ( 16 decimal number) turn BOOL | description |
|---|---|

| | |
|---|---|
|  | When the contact C031 is ON time ; the instruction is executed; The address V034 in the 16 hex into the number of BOOL , stored in the V035 in ; Execution results: 16#OA to BOOL type Note : As long as the number in V034 is not zero, V035 outputs 1 ( TRUE ) |

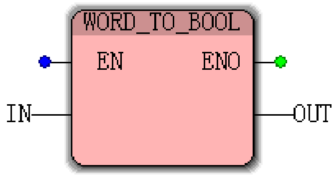
## 10.3.7 Conversion of WORD Data

| Type conversion FU | Features |
|---|---|
| WORD type data conversion | WORD -converted data has . 11 instructions can be WORD respectively converted to data type BOOL, BYTE, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL and LREAL other types. |

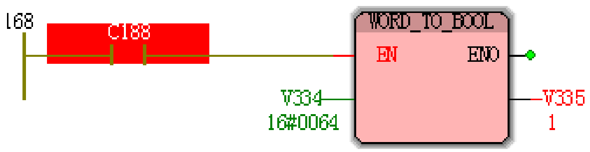
➢ **WORD conversion instruction type data**

| instruction | input value | output value | description |
|---|---|---|---|
| WORD_TO _BOOL | WORD | BOOL | The input value ranges from 0 to 65,535. The output is BOOL type : the output is FALSE only when the input is 0 , and the output is TRUE in other cases ; The output is SINT type : input 0~127 corresponds to output 0~127 , input 128~255 corresponds to output -128~-1 , input increases and output will repeat 0~127 , -128~-1; Output USINT , BYTE Type : Input 0 to 255 corresponding to the output of 0 to 255 , the input further increasing the output will be repeated from 0 to 255; output INT type : Input 0 to 32767 corresponding output of 0 to 32767, input from 32768 to 65535 corresponding to the output -32768~-1; When the output is WORD, DWORD, etc., the output is equal to the input. |
| WORD_TO_BYTE | WORD | BYTE | |
| WORD_TO_DWORD | WORD | DWORD | |
| WORD_TO_SINT | WORD | SINT | |
| WORD_TO_INT | WORD | INT | |
| WORD_TO_DINT | WORD | DINT | |
| WORD_TO_USINT | WORD | USINT | |
| WORD _TO_UINT | WORD | UINT | |
| WORD _TO_UDINT | WORD | UDINT | |
| WORD _TO_REAL | WORD | REAL | |
| WORD_TO_LREAL | WORD | LREAL | |

**Usage (take WORD_TO_BOOL as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| WORD_TO_BOOL | |
| ST OUT | |
| St programming language | |
| OUT:=WORD_ TO_BOOL ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**Funtion and Action examples**

| 16 #0064 转 BOOL type | description |
|---|---|
|  | When the contact C188 is ON when this instruction is executed, the address of V334 in the word into BOOL type , is stored in the V335 in ;<br>Execution results: 16 #0064 转 BOOL<br>Note : When the V334 the value is not zero , V335 output is TRUE ( . 1 ) |

# 10.3.8 Conversion of DWOR D -type data

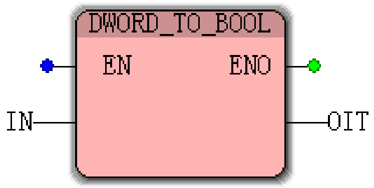
| Type conversion FU | Features |
|---|---|
| DWORD type data conversion | DWORD type data conversion has 11 instructions, which can convert DWORD type data into BOOL, BYTE, WORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL and LREAL . |

**DWORD type data conversion instruction**

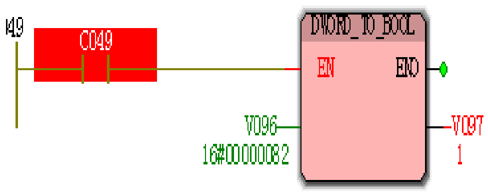
| instruction | input value | output value | description |
|---|---|---|---|
| DWORD_TO_BOOL | DWORD | BOOL | The input value ranges from 0 to 4,294,967,295; the output is BOOL type : only when the input is 0 , the output is FALSE, and in other cases, the output is TRUE;<br>The output is SINT type : input 0~127 corresponds to |
| DWORD_TO_BTTE | DWORD | BYTE | |
| DWORD_TO_WORD | DWORD | WORD | |
| DWORD_TO_SINT | DWORD | SINT | |
| DWORD_TO_INT | DWORD | INT | |
| DWORD_TO_DINT | DWORD | DINT | |

| | | | |
|---|---|---|---|
| DWORD_TO_USINT | DWORD | USINT | output 0~127 , input 128~255 corresponds to output -128~-1 , input increases, output will repeat 0~127 , -128~-1; output is USINT , BYTE Type : Input 0~255 corresponds to output 0~255 . When the input is increased, the output will repeat 0~255; the output is INT type : input 0~32767 corresponds to output 0~32767, input 32768~65535 corresponds to output -32768~-1 , If the input is increased, the output will repeat 0~32767 , -32768~-1;; the output is UINT , WORD type : input 0~65535 corresponds to output 0~65535 , the input will increase and the output will repeat 0~65535; the output is BYTE, When WORD is of type, the output is equal to Enter the lower 8 bits and lower 16 bits of data. |
| DWORD_TO_UINT | DWORD | UINT | |
| DWORD_TO_UDINT | DWORD | UDINT | |
| DWORD_TO_REAL | DWORD | REAL | |
| DWORD_TO_LREAL | DWORD | LREAL | |

**usage (take DWORD_TO_BOOL as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| DWORD_TO_BOOL | |
| ST OUT | |
| ST programming language | |
| OUT:=DWORD_TO_BOOL ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➤**Funtion and Action examples**

| 16#00000082 turn BOOL | description |
|---|---|
|  | When the contact C049 is ON time , the instruction is executed;<br>The address V096 in the DWORD number of revolutions BOOL type , is stored in the V097 in ;<br>Execution result: 16 # 0000082 turn for the BOOL type<br>Note : As long as V096 in value is not to zero when output . 1 ( TRUE ) |

# 10.3.9 Conversion of SINT data

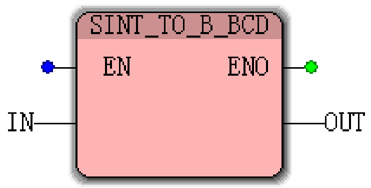
| Type conversion FU | Features |
|---|---|
| SINT type data conversion | SINT type data conversion has 14 instructions, which can convert SINT type data into B_BCD, W_BCD, D_BCD , BOOL, BYTE , WORD , DWORD, INT, DINT, USINT, UINT, UDINT, REAL and LREAL . |

**SINT type data conversion instruction**

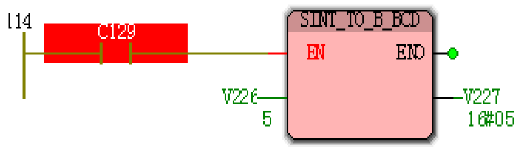
| instruction | input value | output value | description<br>The input value ranges from -128 to 127. |
|---|---|---|---|
| SINT_TO_B_BCD | SINT | BYTE | When inputting 0~99 , output 16#0~99; when inputting other values, output 16#FF |
| SINT_TO_W_BCD | SINT | WORD | When inputting 0~127 , output 16#0~127; when inputting other values, output 16#FFFF |
| SINT_TO_D_BCD | SINT | DWORD | When inputting 0~127 , output 16#0~127 , when inputting other values, output 16#FFFFFFFF |
| SINT_TO_BOOL | SINT | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| SINT_TO_BYTE | SINT | BYTE | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT_TO_WORD | SINT | WORD | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT_TO_DWORD | SINT | DWORD | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT_TO_INT | SINT | INT | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT_TO_DINT | SINT | DINT | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT_TO_USINT | SINT | USINT | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT_TO_UINT | SINT | UINT | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |

| | | | |
|---|---|---|---|
| SINT _TO_UDINT | SINT | UDINT | When inputting 0~127 , output 0~127; input -128~-1 , output 128~255 |
| SINT _TO_REAL | SINT | REAL | When inputting 0~127 , output 0~127; input -128~-1 , output -128~-1 |
| SINT_TO_LREAL | SINT | LREAL | When inputting 0~127 , output 0~127; input -128~-1 , output -128~-1 |

**usage (take SINT_TO_B_BCDAs an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |   |
| SINT_TO_B_BCD | |
| ST OUT | |
| ST programming language | |
| OUT:=SINT_TO_B_BCD ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➤**Funtion and Action examples**

| Short Integer 5 turn to BCD16 # 05 | description |
|---|---|
|   | This type conversion function converts the input value of the data type SINT into the BCD output value ( binary code decimal number ) of the data type BYTE . |

# 10.3.10 Conversion of INT data

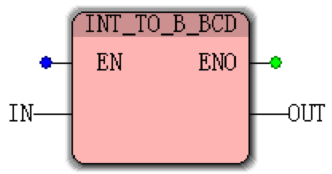
| Type conversion FU | Features |
|---|---|
| INT type data conversion | INT -converted data has 14 instructions can be INT respectively converted data type for the day the BCD, WBCD, D the BCD, BOOL, BYTE , WORD , DWORD , SINT, DINT, USINT, UINT, UDINT, REAL and LREAL other types. |

**INT type data conversion instruction**

| instruction | input value | output value | description The input value ranges from 32768 to 32767; |
|---|---|---|---|
| INT_TO_B_BCD | INT | BYTE | When inputting 0~99 , output 16#0~99; when inputting other values, output 16#FF |
| INT_TO_W_BCD | INT | WORD | When inputting 0~9999 , output 16#0~9999; |

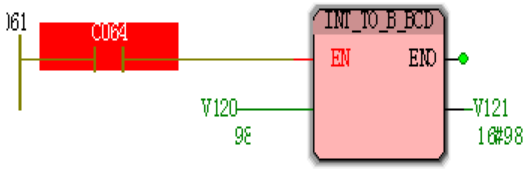
| | | | |
|---|---|---|---|
| | | | when inputting other values, output 16#FFFF |
| INT_TO_D_BCD | INT | DWORD | When inputting 0~32767 , output 16#0 ~32767 , when inputting other values, input 16#FFFFFFFF |
| INT _TO_BOOL | INT | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| INT_TO_BYTE | INT | BYTE | When inputting 0~255 , the output will be 0~255; if the input is increased, the output will repeat 0~255; input -1~255 , output 255~0; if the input is reduced, the output repeats 255~0 |
| INT_TO_WORD | INT | WORD | When inputting 0~32767 , output 0~32767; input -32768~-1 , output 32768~65535 |
| INT_TO_DWORD | INT | DWORD | Input 0 to 32767 , the output 0 to 32767; input -32768- to 1 , the output shown from 32768 to 65535 |
| INT_TO_SINT | INT | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128-1; input and then increase output repeat 0~127, -128~-1; input -1~-128 , output -1~- 128; input -129~-256 , output 127~ input and then decrease the output repeat -1~-128, 127~0 |
| INT_TO_DINT | INT | DINT | When inputting -32768~32767 , output -32768 ~32767 |
| INT_TO_USINT | INT | USINT | When 0~255 is input , the output is 0~255; if the input is increased, the output repeats 0~255. |
| INT_TO_UINT | INT | UINT | When inputting 0~32767 , output 0~32767; input -32768~-1 , output 32768~65535 |
| INT_TO_UDINT | INT | UDINT | Input 0 to 32767 , the output 0 to 32767; input -32768 to -1 , the output 32768 a 65535 |
| INT_TO_REAL | INT | REAL | When inputting 0~32767 , output 0~32767; input -32768~-1 , output -32768~-1 |
| INT_TO_LREAL | INT | LREAL | When inputting 0~32767 , output 0~32767; input -32768~-1 , output -32768~-1 |

**usage (take INT_TO_B_BCD as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| INT_TO_B_BCD | |
| ST OUT | |
| St programming language | |
| OUT:=INT_TO_B_BCD ( IN ) | |

| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet |
|---|

➤**Funtion and Action examples**

| Integer 98 turn as BYTE | description |
|---|---|
|  | When the contact C064 is ON when the instruction is executed, the address of V006 in the integer 98 into BYTE type , is stored in the V121 in ;<br>Execution results: 98 integer to 16 #98 ( BYTE ) |

## 10.3.11 Conversion of DINT type data

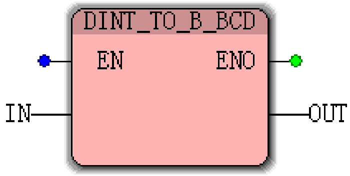
| Type conversion FU | Features |
|---|---|
| DINT type data conversion | DINT type data conversion has 16 instructions, which can convert DINT type data into B_BCD, W_BCD, D_BCD , BOOL, BYTE , WORD , DWORD, SINT , INT, USINT, UINT, UDINT, REAL, LREAL, BCDAnd TIME, etc. Types of. |

**DINT type data conversion instruction**

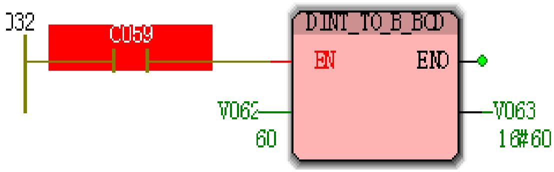
| instruction | input value | output value | Description : The input value ranges from -2,147,483,648...2,147,483,647 |
|---|---|---|---|
| DINT_TO_B_BCD | DINT | BYTE | When inputting 0~99 , output 16#0~99; when inputting other values, output 16#FF |
| DINT_TO_W_BCD | DINT | WORD | When inputting 0~9999 , output 16#0~9999; when inputting other values, output 16#FFFF |
| DINT_TO_D_BCD | DINT | DWORD | When inputting 0~99999999 , output 16#0~99999999 . When other values are input, output 16#FFFFFFF |
| DINT_TO_BOOL | DINT | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| DINT_TO_BYTE | DINT | BYTE | When inputting 0~255 , the output will be 0~255; if the input is increased, the output will repeat 0~255; input -1~-255 , output 255~0; input and then decrease will output repeat 2550 |
| DINT _TO_WORD | DINT | WORD | When inputting 0~65535 , the output is 0~65535; if the input is increased, the output |

| | | | |
|---|---|---|---|
| | | | repeats 0~65535; input -1~-65536 , output 65535~0; if the input is decreased, the output repeats 65535~0 |
| DINT_TO_DWORD | DINT | DWORD | When inputting 0~2,147,483,647 , output 0~2,147,483,647; input- 2,147,483,648~ -1 , output 2,147,483,648~4,294,967,295 |
| DINT_TO_SINT | DINT | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1; input and then increase output repeat 0~127, -128~-1; input -1 ~-128 , output -1~ -128; input -129~-256 , output 127~0; input and then decrease, output repeat -1~-128, 127~0 |
| DINT _TO-INT | DINT | INT | When input -32768 ~32767 , the output is -32768 ~32767; when the input is greater than 32767 , the output repeats -32768~32767; the input is less than -32768 , the output repeats 32767-32768 |
| DINT_TO_USINT | DINT | USINT | When 0~255 is input , the output is 0~255; if the input is increased, the output repeats 0~255. |
| DINT_TO_UINT | DINT | UINT | When inputting 0~65535 , the output is 0~65535; if the input is increased, the output repeats 0~65535; input -65536~-1 , output 0~65535; if the input is reduced, the output repeats 65535~0 |
| DINT-TO_UDINT | DINT | UDINT | When inputting 0~2,147,483,647 , output 0~2,147,483,647; input- 2,147,483,648~ -1 , output 2,147,483,648~4.294.967.295 |
| DINT_TO_REAL | DINT | REAL | When inputting 0~2,147,483,647 , the output is 0~2,147,483,647; input -2,147,483,648~-1 , output- 2,147,483,648~-1 , the precision will be reduced |
| DINT_TO_LREAL | DINT | LREAL | When inputting 0~2,147,483,647 , the output is 0~2,147,483,647; input -2,147,483,648~-1 , output- 2,147,483,648~-1 , the precision will be reduced |
| DINT_TO_BCD | DINT | BCD | Input value DINT 0~99999999 , output value BCD code 16#00000000~16#99999999 . |
| DINT_TO_TIME | DINT | TIME | The output value is iNSeconds ; the input value is 0~2,147,483,647 , the output value is 0~2147483.647 seconds ; the input value is -2147483648~-1 seconds, and the output value is 2147483.648~4294967.295 seconds. |

**usage (take DINT_TO_B_BCDAs an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |   |
| DINT_TO_B_BCD | |
| ST OUT | |
| ST programming language | |
| OUT:=DINT_TO_B_BCD ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**Funtion and Action examples**

| Double integer 60 rpm BYTE | description |
|---|---|
|   | When the contact C059 is ON when this instruction is executed, the address of V062 in the double integer of 60 revolutions of BYTE stored in V063 in ;<br>Execution of Results : 60 ( DINT ) switch 16 # 60 ( BYTE ) |

# 10.3.12 Conversion of USINT type data

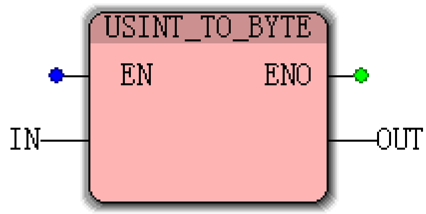
| Type conversion FU | Features |
|---|---|
| USINT type data conversion | USINT conversion type data there . 11 instructions can be USINT respectively converted to data type BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, UINT, UDINT, REAL and LREAL other types. |

**USINT type data conversion instruction**

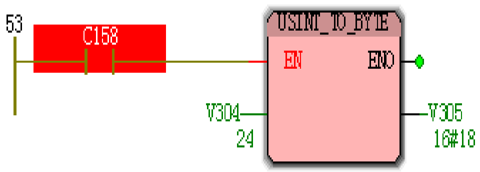
| instruction | input value | output value | Description The input value ranges from 0 to 255. |
|---|---|---|---|
| USINT_TO_BOOL | USINT | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| USINT_TO_BYTE | USINT | BYTE | When 0~255 is input , the output is 0~255. |

| | | | |
|---|---|---|---|
| USINT_TO_WORD | USINT | WORD | When 0~255 is input , the output is 0~255. |
| USINT_TO_DWORD | USINT | DWORD | When 0~255 is input , the output is 0~255. |
| USINT_TO_SINT | USINT | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1 |
| USINT_TO_INT | USINT | INT | When 0~255 is input , the output is 0~255. |
| USINT_TO_DINT | USINT | DINT | When 0~255 is input , the output is 0~255. |
| USINT_TO_UINT | USINT | UINT | When 0~255 is input , the output is 0~255. |
| USINT_TO_UDINT | USINT | UDINT | When 0~255 is input , the output is 0~255. |
| USINT_TO_REAL | USINT | REAL | When 0~255 is input , the output is 0~255. |
| USINT_TO_LREAL | USINT | LREAL | When 0~255 is input , the output is 0~255. |

**usage (using USINT_TO_BYTE as an example) IL programming language**

| LD IN | LD, FBD programming language |
|---|---|
| USINT_TO_BYTE |  |
| ST OUT | |
| ST programming language | |
| OUT:=USINT_TO_BYTE ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**Funtion and Action examples**

| 24 ( USINT ) to BYTE | description |
|---|---|
|  | When the contact C158 is ON when this instruction is executed,<br>The address V304 in the unsigned short integer of 24 rpm to BYTE , stored in the V 305 In ;<br>Execution of Results : 24 ( DINT ) switch 16 # 18 is ( BYTE ) |

## 10.3.13 Conversion of UINT type data

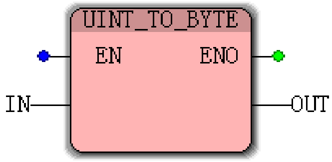
| Type conversion FU | Features |
|---|---|
| UINT type data conversion | UINT type data conversion has 11 instructions, which can convert UINT type data into BOOL, BYTE, WORD, DWORD , SINT, INT, DINT, USINT, UDINT, REAL and LREAL . |

**UINT type data conversion instruction**

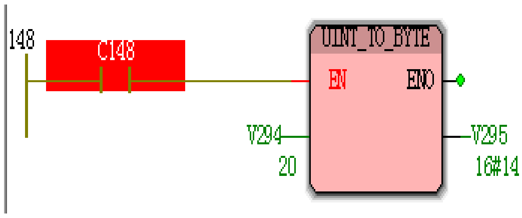
| instruction | input value | output value | Description Input value range 0 65535 |
|---|---|---|---|
| UINT _TO_BOOL | UINT | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| UINT _TO_BYTE | UINT | BYTE | When 0~255 is input, the output will be 0~255; if the input is increased, the output will repeat 0~255. |
| UINT _TO_WORD | UINT | WORD | When 0~65535 is input , the output is 0~65535. |
| UINT _TO_DWORD | UINT | DWORD | When 0~65535 is input , the output is 0~65535. |
| UINT _TO_SINT | UINT | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1; input and then increase output repeat 0~127, -128~-1 |
| UINT _TO_INT | UINT | INT | When inputting 0~32767 , output 0~32767; input 32768~65535, output -32768~-1 |
| UINT _TO_DINT | UINT | DINT | When 0~65535 is input , the output is 0~65535. |
| UINT _TO_USINT | UINT | USINT | When 0~255 is input , the output is 0~255; if the input is increased, the output repeats 0~255. |
| UINT _TO_UDINT | UINT | UDINT | When 0~65535 is input , the output is 0~65535. |
| UINT_TO_REAL | UINT | REAL | When 0~65535 is input , the output is 0~65535. |
| UINT _TO_LREAL | UINT | LREAL | When 0~65535 is input , the output is 0~65535. |

**usage (take UINT_TO_BYTE as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN | |
| UINT TO BYTE | |
| ST OUT | |

| St programming language | |
|---|---|
| OUT:=UINT_TO_BYTE ( IN ) |  |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**Funtion and Action examples**

| Unsigned integer 20- turn BYTE | description |
|---|---|
|  | When the contact C148 is ON when the instruction is executed, the address of V294 in the unsigned integer of 20 revolutions of BYTE stored in V295 in ; Execution of Results : 20 is ( the U- the INT ) switch 16 # 14 ( BYTE ) |

# 10.3.14 Conversion of UDINT type data

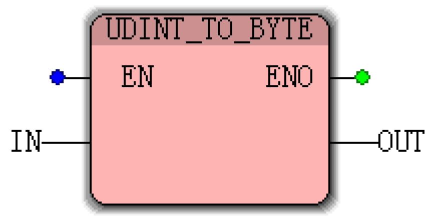
| Type conversion FU | Features |
|---|---|
| UDINT type data conversion | UDINT type data conversion has 11 instructions, which can convert UDINT type data into BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, REAL and LREAL . |

**UDINT type data conversion instruction**

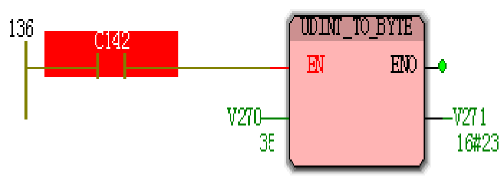
| instruction | input value | output value | Description Input value range 04,294,967,295 |
|---|---|---|---|
| UDINT_TO_BOOL | UDINT | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| UDINT _TO_BYTE | UDINT | BYTE | When 0~255 is input, the output will be 0~255; if the input is increased, the output will repeat 0~255. |
| UDINT _TO_WORD | UDINT | WORD | When inputting 0~65535 , the output will be 0~65535 . If the input is increased, the output will repeat 0~65535. |
| UDINT _TO_DWORD | UDINT | DWORD | When 0~4,294,967,295 is input , the output is 0~4,294,967,295 |
| UDINT_TO_SINT | UDINT | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1; input and then increase output repeat 0~127, -128~-1 |
| UDINT_TO_INT | UDINT | INT | When inputting 0~32767 , the output is |

| | | | |
|---|---|---|---|
| | | | 0~32767; input 32768~65535 output is -32768~-1 , if the input is increased, the output repeats 0~32767, -32768~-1 |
| UDINT_TO_DINT | UDINT | DINT | When inputting 0~2,147,483,647 , the output is 0~2,147,483,647; into 2,147,483,648~4,294,967,295 , lose- 2,147,483,648~-1 |
| UDINT_TO_USINT | UDINT | USINT | When 0~255 is input , the output is 0~255; if the input is increased, the output repeats 0~255. |
| UDINT_TO_UINT | UDINT | UDINT | When inputting 0~65535 , the output is 0~65535; if the input is increased, the output repeats 0~65535 |
| UDINT_TO_REAL | UDINT | REAL | When 0~4,294,967,295 is input , the output is 0~4,294,967,295 , the accuracy will be reduced. |
| UDINT _TO_LREAL | UDINT | LREAL | When 0~4,294,967,295 is input , the output is 0~4,294,967,295 , the accuracy will be reduced. |

**usage (take UDINT_TO_BYTE as an example)**

| **IL programming language** | **LD, FBD programming language** |
|---|---|
| LD IN |  |
| UDINT_TO_BYTE | |
| ST OUT | |
| ST programming language | |
| OUT:=UDINT_TO_BYTE ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➤**Funtion and Action examples**

| No match double full 35 turn BYTE | description |
|---|---|
|  | When the contact C142 is ON when this instruction is executed, the address of V270 in the unsigned integer bis 35 turn as BYTE , stored in the V271 in ;<br>Execution of Results : 35 ( UDINT ) switch 16 # 23 is ( BYTE ) |

## 10.3.15 Conversion of REAL type data

| Type conversion FU | Features |
|---|---|

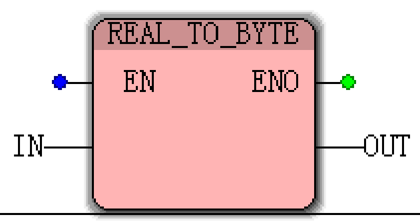
| | |
|---|---|
| Conversion of REAL type data | REAL type data conversion has 11 instructions, which can convert REAL type data into BOOL, BYTE, WORD, DWORD , SINT, INT, DINT, USINT, UINT, UDINT and LREAL . |

**REAL type data conversion instruction**

| instruction | input value | output value | description<br>REAL type input value whose fractional part is discardeDAt the time of conversion |
|---|---|---|---|
| REAL_TO_BOOL | REAL | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| REAL_TO_BYTE | REAL | BYTE | When inputting 0~255 , the output will be 0~255; if the input is increased, the output will repeat 0~255; input -1~-255 , output 255~0; input and then decrease will repeat 255~0 |
| REAL_TO_WORD | REAL | WORD | When inputting 0~65535 , the output is 0~65535 . If the input is increased, the output repeats 0~65535; input -1~-65536 , output 65535~0 , input and then decrease, the output repeats 65535~0 |
| REAL_TO_DWORD | REAL | DWORD | When inputting 0~2,147,483,647 , output 0~2,147,483,647 , input and increase output is 2,147,483,647; input -1~-2,147,483,648 , output 4,294,967,295~2,147,483,648 , input and then reduce output is still 2,147,483,648; accuracy will decrease |
| REAL_TO_SINT | REAL | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1; input and then increase output repeat 0~127, -128~-1; input -1 ~-128 , output -1~ -128; input -129~-256 , output 127~0; input and then decrease, the output repeats -1~128, 127~0 |
| REAL _TO_INT | REAL | DINT | When inputting 0~32767 , the output is 0~32767; input 32768~65535} output -32768~-1; if the input is increased, the |

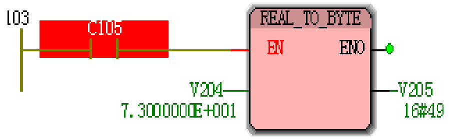
| | | | |
|---|---|---|---|
| | | | output repeats 0~32767, -32768~-1; input -1~-32768 , output -1~ -32768; input -32769~-65536 , output 32767~0; input and then decrease, output repeat -1~-32768, 32767~0 |
| REAL _TO_DINT | REAL | DINT | When inputting 0~2,147,483,647 , the output is 0~2,147,483 and the output is still 2,147,483,647-1 ~-2,147,483,648 , the output is -1~-2,147,483,648, the input is reduceDAnd the output is still -2,147,483,648 |
| REAL_TO_USINT | REAL | USINT | When inputting 0~255 , the output is 0~255 . If the input is increased, the output repeats 0~255; input -1~-256 , output 255~0 , output decreases and the output repeats 255~0 |
| REAL_TO_UINT | REAL | UINT | When inputting 0~65535 , the output is 0~65535 . If the input is increased, the output repeats 0~65535; input -1~-65536 , output 65535~0 , input and then decrease, the output repeats 65535~0 |
| REAL_TO_UDINT | REAL | UDINT | When inputting 0~2,147,483,647 , the output is 0~2,147,483,647 , the input is increased to 2,147,483,647; the input is -1~-2,147,483,648 , the output is 4,294,967,295~2,147,483,648 , the input is reduceDAnd the output is still 2,147,483,648; the precision will be reduced |
| REAL_TO_LREAL | REAL | LREAL | Input equal to output |

**usage (take REAL_TO_BYTE as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| REAL_TO_BYTE | |
| ST OUT | |
| ST programming language | |
| OUT:=REAL_TO_BYTE ( IN ) | |

| |
|---|
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet |

➤**Funtion and Action examples**

| Float 73 is .0 turn BYTE | description |
|---|---|
|  | When the contact C105 is ON , the execution of the instruction , the address of V204 in the floating 73.0 revolutions of BYTE stored in the V 205 in ;<br>Execution of results : 73.0 ( REAL ) switch 16 # 49 ( BYTE ) |

## 10.3.16 Conversion of LREAL type data

| Type conversion FU | Description |
|---|---|
| LREAL type data conversion | LREAL conversion type data there . 11 instructions can be LREAL respectively converted to data type BOOL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT and REAL types. |

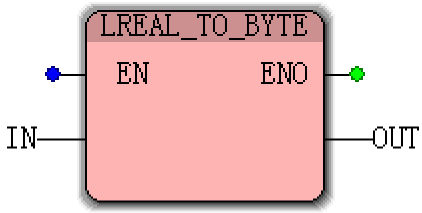
**LREAL type data conversion instruction**

| instruction | input value | output value | Describe the REAL type of input value whose fractional part is discardeDAt the time of conversion |
|---|---|---|---|
| LREAL_TO_BOOL | LREAL | BOOL | When input 0 , output FALSE; when other values are input, output TRUE |
| LREAL_TO_BYTE | LREAL | BYTE | When 0~255 is input , the output is 0~255; if the input is increased, the output will be repeated.<br>0~255; input -1~-255 , output 255~0; input and then decrease, output repeats 255~0 |
| LREAL_TO_WORD | LREAL | WORD | When inputting 0~65535 , the output is 0~65535 . If the input is increased, the output repeats 0~65535; input -1~-65536 , output 65535~0 , input and then decrease, the output repeats 65535~0 |
| LREAL_TO_DWORD | LREAL | DWORD | When inputting 0~2,147,483,647 , output 0~2,147,483,647 , input and |

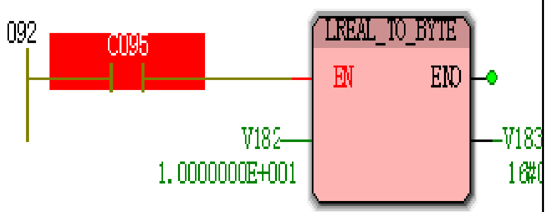
| | | | |
|---|---|---|---|
| | | | increase output is 2,147,483,647; input -1~-2,147,483,648 , output 4,294,967,295~2,147,483,648 , input and then reduce output is still 2,147,483,648; accuracy will decrease |
| LREAL_TO_SINT | LREAL | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1; input and then increase output repeat 0~127, -128-1; input -1~-128 , output -1~- 128; input -129~-256 , output 127~0; input and then decrease, the output repeats -1~-128, 127~0 |
| LREAL_TO_INT | LREAL | DINT | When inputting 0~32767 , the output is 0~32767; input 32768~65535 output is -32768~-1; if the input is increased, the output repeats 0~32767 -32768~-1; input -1~-32768 , output -1~-32768 ; input -32769 -65536 ~ , outputs 32767 ~ 0; input the output decreases again repeated -1 to -32768, 32767 ~ 0 |
| LREAL_TO_DINT | LREAL | DINT | When inputting 0~2,147,483,647 , the output is 0~2,147,483647 and the output is still 2,147,483,647; input -1~-2,147,483,648 , output -1~-2,147,483,648<br>Input and then reduce the output is still -2,147,483,648 |
| LREAL_TO_USINT | LREAL | USINT | When inputting 0~255 , the output is 0~255 . If the input is increased, the output repeats 0~255; input -1~-256 , output 255~0 , output decreases and the output repeats 255~0 |
| LREAL_TO_UINT | LREAL | UINT | When inputting 0~65535 , the output is 0~65535 . If the input is increased, the output repeats 0~65535; input -1~-65536 , output 65535~0 , input and then decrease, the output repeats 65535~0 |
| LREAL_TO_UDINT | LREAL | UDINT | When inputting 0~2,147,483,647 , |

| | | | |
|---|---|---|---|
| | | | output 0~2,147,483,647 , input and increase output is 2,147,483,647; input -1~-2,147,483,648 , output 4,294,967,295~2,147,483,648 , input and then reduce output is still 2,147,483,648; accuracy will decrease |
| LREAL_TO_LREAL | LREAL | REAL | Input equals output, accuracy is reduced |

**usage (take LREAL_TO_BYTE as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| LREAL_TO_BYTE | |
| ST OUT | |
| ST programming language | |
| OUT:=LREAL_TO_BYTE ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**Funtion and Action examples**

| Long floating point number 10.0 rpm BYTE | description |
|---|---|
|  | When the contact C095 is ON , the execution of the instruction, the address of V182 in the long floating-point number 10.0 revolutions of BYTE , stored in the V183 in ;<br>Execution of results : 10.0 ( the LREAL ) switch 16 # 0A ( BYTE ) |

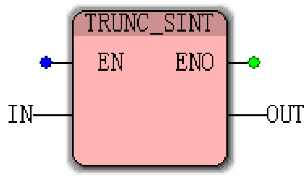
## 10.3.17 TRUNC decimal rounding

| Type conversion FU | Features |
|---|---|
| TRUNC decimal rounding | The TRUNC fraction has four instructions, which can convert floating-point data to SINT , INT , DINT, and so on. |

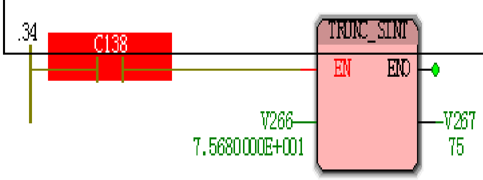
**RUNC decimal rounding instruction**

| instruction | input value | output value | description |
|---|---|---|---|
| TRUNC | REAL | Unsigned integer | Truncating the fractional part of the input value to get an integer value |
| TRUNC_SINT | REAL | SINT | When inputting 0~127 , output 0~127; input 128~255 , output -128~-1; input and then increase output repeat 0~127, -128~-1; input 1~-128 , output -1~ -128; input -129~-256 , output 127~0; input and then decrease, output repeat -1~-128, 127~0 |
| TRUNC_INT | REAL | INT | Input 0 to 32767 when , outputs 0 to 32767; input from 32768 to 65535 , the output from -32768 to -1; input to add the output Repeat 0~32767 , -32768~-1; input -1~-32768 , output -1~-32768; input -32769~-65536 , output 32767~0; input and then decrease, the output repeats -1~-32768, 32767~0 |
| TRUNC _DINT | REAL | DINT | When 0~2,147,483,647 is input , the output is 0~2,147,483,647. Input and then increase the output is still 2,147,483,647; input -1~-2,147,483,648 , output -1~-2,147,483,648 , input and then reduce the output is still -2,147,483,648 |

**usage (take TRUNC_SINT as an example)**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| TRUNC_SINT | |
| ST OUT | |
| ST programming language | |
| OUT:=TRUNC_ SINT ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**➢Funtion and Action examples**

| Floating point number 75.68 rounding | description |
|---|---|
|  | When the contact C138 is ON when this instruction is executed, |

| | |
|---|---|
| | The address of V266 in the floating-point number 75.68 rounding stored in the V 267 in ;<br>The implementation of the results : 75.69 turn to 75<br>(Note: This function is rounded up , the number after the decimal point is rounded off) |

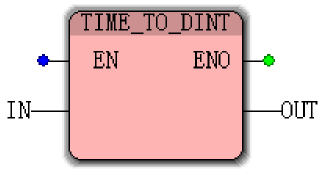
## 10.3.18 Conversion of TIME type data

| Type conversion FU | Features |
|---|---|
| TIME_TO _DINT type conversion | The TIME_TO _DINT type conversion function converts a TIME type input value into ADINT type output value ( any time value is converted to a millisecond value and then the millisecond value is converted to DINT ) . The TIME type data must be an unsigned number starting with T# . The time value greater than #2147483647 will be negative because the DINT type is a signed number and its maximum value is 2,147,483,647 . For example, the input value T#4294967295<br>A millisecond will result in an output value of -1 . |

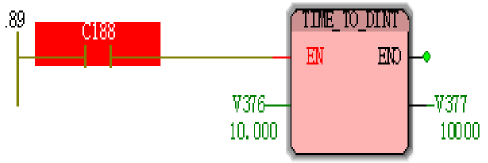
➢ **TIME_TO_DINT data type conversion instruction processing**

| instruction | input value | output value | description |
|---|---|---|---|
| TIME_TO_DINT | TIME | DINT | Input value range T#0~2147483647MS ( equal to T#0~2147483.647S , converted to DINT output value<br>0~2147483647; input value range T#2147483648~4294967295 , converted to DINT output value -2147483648~-1 . |

**usage**

| IL programming language | LD, FBD programming language |
|---|---|
| LD IN |  |
| TIME_TO_DINT | |
| ST OUT | |
| ST programming language | |
| OUT:=TIME_TO_DINT ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➤**Funtion and Action examples**

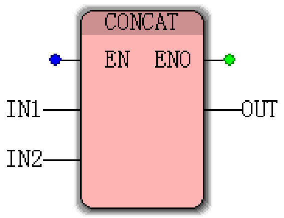
| The T # 10 S converted into bis integer | description |
|---|---|
|  | When the contact C188 is ON when the instruction is executed, the address of V376 in the time value ( T # 10s ) turn to DINT , stored in the V277 in ;<br>Execution of Results : T # 10S ( the TIME ) converted to 10000 ( DINT )<br>Note that ( time units into DINT between the turn into 1000 ;<br>(Example lS --- 1000 ) |

## 1 0.4 String FU

·string function, referred to as the string FU , is a comparison, conversion, lookup, connection, etc. operation for strings, with multiple input parameters and one output parameter. During the MULTIPROG programming process, you caNSelect " String FU " from the drop-down list in the Edit WizarDAnd use the following command.

| classification | Features | | |
|---|---|---|---|
| Merge Insert Delete Replacement Length Limit | CONCAT | INSERT | DELEtE |
| | REPLACE | LEN | LI MIT_STRING |
| | FIN D | | |
| Size and location | MAX_STRING | M IN_STRING | LEFT |
| | MID | RIGHT | SEL_STRING |
| Comparison | GT_STRING | GE_STRING | EQ_STRING |
| | NE_STRING | LE STRING | LT_STRING |
| Convert string to other | STRING_TO_BYTE | STRING_TO_WORD | STRING_TO_DWORD |
| | STRING_TO_SINT | STRING_TO_INT | STRING_TO_DINT |
| | STRING_TO_USINT | STRING_TO_UINT | STRING_TO_UDINT |
| | STRING_TO_LREAL | STRING_TO_REAL | STRING_TO_TIME |
| Other conversion to string | BYTE_TO_STRING | WORD_TO_STRING | DWORD_TO_STRING |
| | SINT_TO_STRING | INT_TO_STRING | DINT_TO_STRING |
| | USINT_TO_STRING | UINT_TO_STRING | UDINT_TO_STRING |
| | REAL_TO_STRING | LREAL_TO_STRING | TIME_TO_STRING |

### 10.4.1 CONCAT (Merge String)

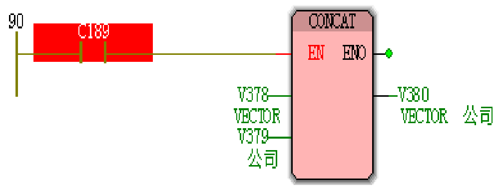
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The CONCAT merge string instruction is used to merge two strings.</td></tr>
<tr><td>LD IN</td><td rowspan="5">
</td></tr>
<tr><td>CONCAT</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=CONCAT ( IN1 , IN2 )</td></tr>
</table>

| |
|---|
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet |

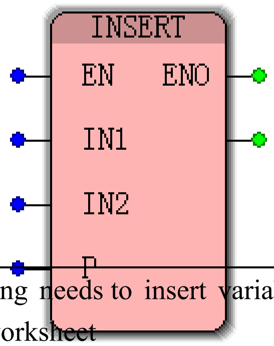
➢ **CONCAT data processing instruction type**

| Input | type of data | description |
|---|---|---|
| IN1 | STRING | First input |
| IN2 | STRING | Second input |
| Output | type of data | description |
| OUT | STRING | Output, OUT=IN1 +IN2; the second input is addeDAfter the first input, the output is not allowed to have the same name as the input |

➢**Funtion and Action examples**

<table>
<tr><th>“ VECTOR Company”</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C189 is ON when the instruction is executed , the address of V378 in the string + address V379 in the string , stored in the V 380 In ;<br>Execution of the results : the VECTOR + company = VECTOR Company</td></tr>
</table>

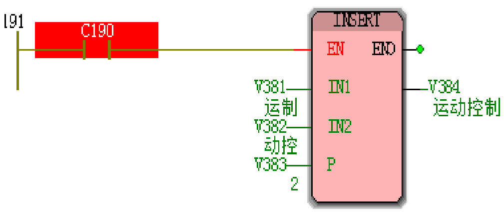
## 10.4.2 INSERT (insert string)

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The INSERT Insert String instruction is used to insert a string or character into another string.<br>usage</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>INSERT IN2, P</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=INSERT ( IN1 , IN2, P )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2, P and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

**数据 Data type processed by INSERT instruction**

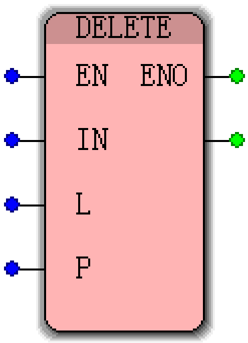
| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | The first input string, when using a variable, the variable must be set to a non-null initial value, or a string constant |
| IN2 | STRING | EleveNSecond input string to be inserted into the first input, using the variable, variable amount of non-null initial value must be set, the string constant may be used |
| P | ANY_INT | At the insertion point in the first input, P must be an integer greater than 0 ; the first character position in the string is 1 and the following characters are 2, 3..., |
| OUT | STRING | The output second input is inserteDAfter the P character of the first input, and the output is not allowed to have the same name as the input. |

➢**Funtion and Action examples**

| The string " operation control " into the "string" Win "System run " syllable | description |
|---|---|
|  | When the contact C190 is ON when this instruction is executed, the address of V381 in the insertioNString address V382 in the first string after stored in V 384 In , P represents the insertion of a few strings after<br>Execution results: transportation + motion control = motion control |

## 10.4.3 DELETE ( copies except string)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The DELETE delete string instruction is used to delete several connected characters determined in a string. | |
| LD IN | |
| DELETE L, P | |
| ST OUT | |
| ST programming language | |

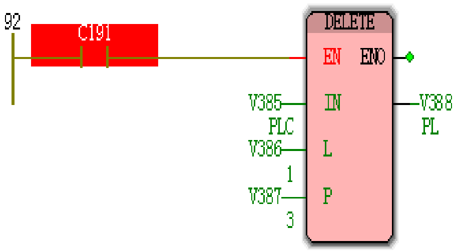


| OUT:=DELETE ( IN, L, P ) | |
|---|---|
| Note : IL, ST language programming needs to insert variables IN, L, P and OUT or use constants in the variable worksheet of the current POU | |

**数据 Data type processed by DELETE instruction**

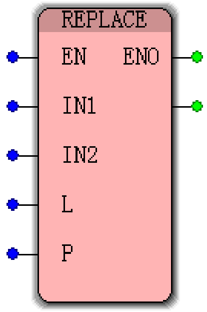
| Input and output | type of data | description |
|---|---|---|
| IN | STRING | Enter a string or use a constant string |
| L | ANY_INT | The number of characters to be deleted, L can be 0, 1, 2... |
| P | ANY_INT | In the input string, the position of the first character to be deleted, P must be an integer greater than 0 ; the first character position in the string is 1 , and the following characters are 2, 3... |
| OUT | STRING | The output input string is not affected by the deletion. In fact, this instructioNSelects several strings in the input string to form a new string. The output does not allow the same name as the input. |

**➢Funtion and Action examples**

| Delete the third character "C" in the string "PLC" | description |
|---|---|
|  | When the contact C 191 contact is O N , the execution of the instruction; delete address V 385 string in the third character, in an amount of . 1 a; where L is the number of characters to be deleted, P is the first to be deleted Several characters<br><br>Execution result: P LC----PL |

## 10.4.4 REPLACE (replace string)

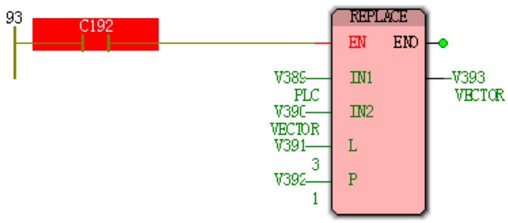
| IL programming | LD, FBD programming language |
|---|---|

| language | |
|---|---|
| Function: The REPLACE Replace String command replaces several connected characters in a string with a string. | |
| LD IN |  |
| REPLACE L, P | |
| ST OUT | |
| ST programming language | |
| OUT:=REPLACE ( IN, L, P ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2, L, P and OUT or use constants in the current POU variable worksheet | |

**数据 Data type processed by the REPLACE instruction**

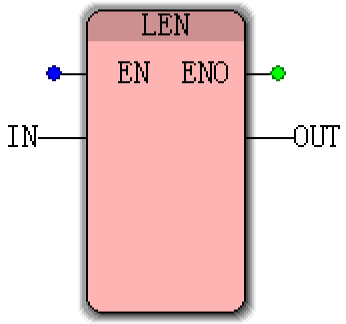
| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | First input string |
| IN2 | STRING | The second input string will replace some characters in the first input string |
| L | ANY_NT | The number of characters to be replaced, L can be 0, 1, 2... |
| P | ANY_INT | In the first input string, the position of the first character to be replaced, P must be an integer greater than 0 ; the first character position in the string is 1 , and the following characters are 2, 3... |
| OUT | STRING | Output the first input string is not affected by the substitution, in fact, this instruction is to select the first two characters in the first input string and the second string to form a new string, the output is not alloweDAnd input Same name |

**REPLACE function anDAction example**

| Replace the string "PLC" with the string "VECTOR" | description |
|---|---|
|  | When the contact C 192 is O N , the execution of the instruction; address V 389 string, replace address V 390 string, where P is the number of strings is replaced by the first few<br>Start being replaced, L is the number of characters to be replaced.<br>Execution result: PLC-----VECTOR |

| | |
|---|---|
| | |

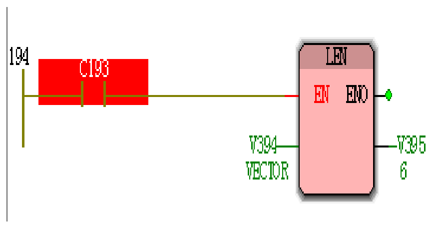
## 10.4.5 LEN ( string length )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: LEN determines the length of a string. | |
| LD IN |  |
| LEN | |
| ST OUT | |
| ST programming language | |
| OUT:=LEN ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

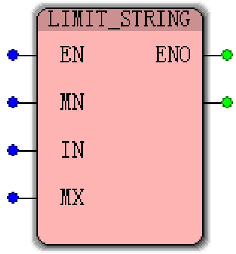
**数据 Data type processed by LEN instruction**

| Input and output | type of data | description |
|---|---|---|
| IN | STRING | Input string |
| OUT | INT | Output, the length of one character is 1 |

**➢Funtion and Action examples**

| Determine the length of a string | description |
|---|---|
|  | When the contact C 193 is O N when the instruction is executed , the address V 394 length of the string is displayeDAs an integer, there is an address V 395 in<br><br>Execution result: V ECTOR-------6 |

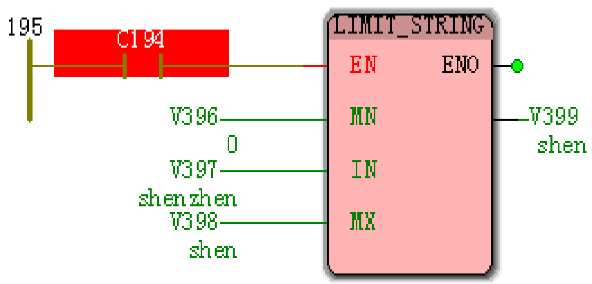
## 10.4.6 LIMIT_STRING (set string limit)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The LIMIT_STRING instruction is used to limit the range of characters or strings entered by the upper and lower limits. | |
| LD IN1 |  |
| LIMIT_ STRING IN, IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=LIMIT_ STRING ( IN1 , IN, IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN, IN2 and OUT or use constants in the current POU variable worksheet | |

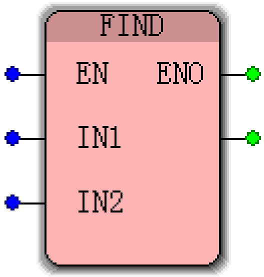
➢ **LIMIT_STRING instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN1 ( MN ) | STRING | Lower limit of characters |
| IN | STRING | Enter a character or string |
| IN2 ( MX ) | STRING | Upper limit of characters |
| OUT | STRING | Output, the upper and lower limits of the input value will be equal to the upper and lower limits respectively ; the size of the characters will be judgeDAccording to the size of the ASCII code value ; when the string is input, only the first character participates in the comparison, that is, if When a character is in the upper and lower limits, the string is all output, otherwise only the upper or lower limit is output. |

➢**Funtion and Action examples**

| Qualified string output | description |
|---|---|
|  | When the contact C 194 is O N , executing the instruction, when the address V 397 string length in the address V 396 , V 398 is between the output itself, whether those output upper and lower limits. Execution result: shenzhen-shen |

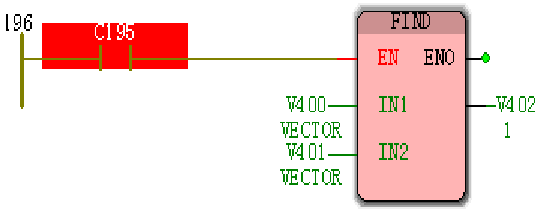
### 10.4.7 FIND (FinDA character that appears in a string)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The FIND instruction is used to determine the position of the second string ( whichever is the first character ) that appears in the first string . | |
| LD IN1 |  |
| FIND IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=FIND ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

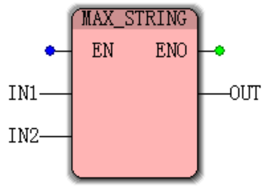
➢**the FIND data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | The first input character or string, the position of the first character from the left in the string is 1 |
| IN2 | STRING | The second input character or string, the position of the first character from the left in the string is 1 |
| OUT | STRING | The output of the second string of the first character string appears in the first position, if the character is not included in the first string, output O . |

➢**Funtion and Action examples**

| Find the same string | description |
|---|---|
|  | When the contact is ON , the instruction is executed. When the character string in the address V 400 is equal to the character string of the address 4 01A , the address V 402 outputs 1 ;<br><br>The result of the execution: V ECTOR = VECTOR output 1 ; |

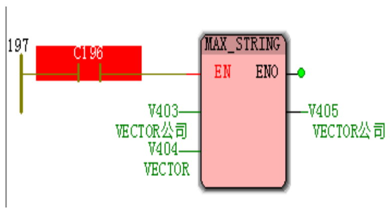
## 10.4.8 MAX_STRING ( take a larger string )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The MAX_STRING instruction is used to determine the larger of the two characters.<br>Usage</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>MAX_STRING IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=MAX_<br>STRING ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

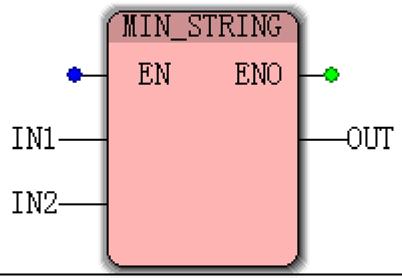
➤ **max_string data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | The first input character or string |
| IN2 | STRING | The second input character or string |
| OUT | STRING | Output, the larger of two characters or a string ; the size of the character is calculateDAccording to its ASCII code ; when the string is input, only the first character participates in the comparison, that is, if the first character of a string is larger , then all the output of this string |

➤**Funtion and Action examples**

<table>
<tr><th>Compare the largest string</th><th>description</th></tr>
<tr><td>
</td><td>When the contact is O N when the instruction is executed, the address comparator V 403 string address V 404 length of the string, there is a string of the maximum output V 405 of:<br>Implementation of the results: V Ector company - --- VECTOR<br>Output : V Ector company</td></tr>
</table>

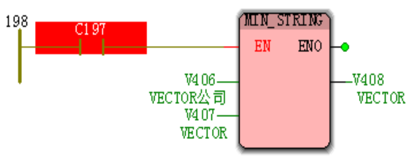
## 10.4.9 MIN_STRING (take a smaller string)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MIN_STRING instruction is used to determine the smaller of the two characters. | |
| LD IN1<br>MIN _ STRING IN2<br>ST OUT<br>ST programming language<br>OUT:=MIN_ STRING ( IN1 , IN2 ) |  |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

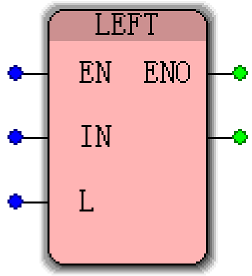
➢ **MIN_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | The first input character or string |
| IN2 | STRING | The second input character or string |
| OUT | STRING | Output, the smaller of two characters or a string ; the size of the character is calculateDAccording to its ASCII code ; when the string is input, only the first character participates in the comparison, that is, if the first character of a string is smaller , then all the output of this string |

**➢Funtion and Action examples**

| Compare the smallest string | description |
|---|---|
|  | When the contact is O N when the instruction is executed, the address comparator V 406 string address V 407 length of the string, there is a minimum string output V 408 of<br>Execution results: V ECTOR - VECTOR<br>Output : V ECTOR |

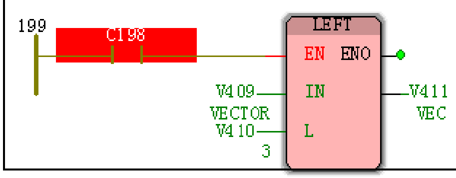
### 10.4.10 LEFT (Remove the last few characters of the string)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The LEFT instruction is used to retrieve the last few consecutive characters in the input string. | |
| LD IN<br>LEFT L<br>ST OUT |  |
| ST programming language | |
| OUT:=LEFT ( IN, L ) | |
| Note : IL, ST language programming needs to insert variables IN, L and OUT or use constants in the variable worksheet of the current POU | |

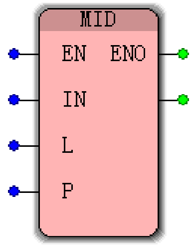
**数据 Data type processed by LEFT instruction**

| Input and output | type of data | description |
|---|---|---|
| IN | STRING | Input string |
| L | ANY_INT | The number of characters to be fetched |
| OUT | STRING | Output, take out L consecutive characters from the leftmost side of the input string ; L must be greater than 0 , and less than or equal to the number of characters of the input string IN |

**➢Funtion and Action examples**

| Extract the first three characters of the string "VECTOR" | description |
|---|---|
|  | When the contact is O N when the instruction is executed, the address V 409 string extracted first three characters<br>Execution result: V ECTOR - ---VEC |

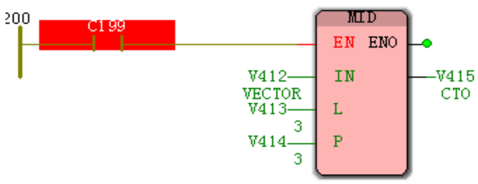
## 10.4.11 MID (Remove several characters in a string)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MID instruction is used to retrieve the last few consecutive characters in the input string. | |
| LD IN |  |
| MID L, P | |
| ST OUT | |
| ST programming language | |
| OUT:=MID ( IN, L, P ) | |
| Note : IL, ST language programming needs to insert variables IN, L, P and OUT or use constants in the variable worksheet of the current POU | |

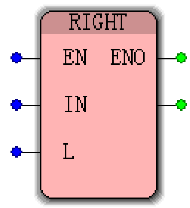
**数据 Data type processed by MID instruction**

| Input and output | type of data | description |
|---|---|---|
| IN | STRING | Input string |
| L | ANY_INT | The number of characters to be fetched |
| P | ANY_INT | The starting position of the character to be fetched, that is, the character is taken from the Pth character of the input string. |
| OUT | STRING | Output, take out the P from the Pth to P+L consecutive characters in the input string ;<br>Note : L must be greater than 0; the first P+L characters must be in the input string ; P must be greater than 0 and less than or equal to the maximum number of characters in the input string ; IN must be a non-empty string |

**➤Funtion and Action examples**

| Extract any character in a string | description |
|---|---|
|  | When the contact C 199 is ON , the instruction is executed to extract three characters in the string in the address V 412 , where L is the number of symbols to be extracted, and P is the starting position of the extracted character.<br>Execution result: V ECTOR---CTO |

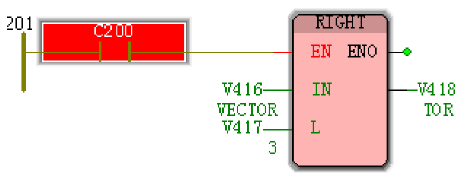
## 10.4.12 RIGHT (remove the rightmost characters of the string)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The RIGHT command is used to retrieve the rightmost few consecutive characters in the input string. | |
| LD IN |  |
| RIGHT L | |
| ST OUT | |
| ST programming language | |
| OUT:=RIGHT ( IN, L ) | |
| Note : IL, ST language programming needs to insert variables IN, L and OUT or use constants in the variable worksheet of the current POU | |

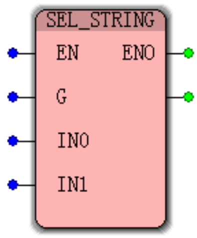
**数据 RIGHT command processing data type**

| Input and output | type of data | description |
|---|---|---|
| IN | STRING | Input string |
| L | ANY_INT | The number of characters to be fetched |
| OUT | STRING | Output, take out L consecutive characters from the rightmost side of the input string ; L must be greater than 0 , and less than or equal to the number of characters of the input string IN |

➢**Funtion and Action examples**

| Extract the rightmost 3 characters of the string V ECTOR | description |
|---|---|
|  | When the contact C 200 is of O N , the execution of the instruction, address V416 three rightmost character is extracted, the output V 418<br><br>Execution result: V ECTOR---TOR |

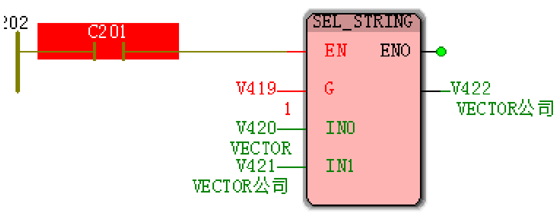
# 10.4.13 SEL_STRING (binary selection of strings)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SEL_STRING instruction is used to retrieve the last few consecutive characters in the input string. | |
| LD IN |   |
| SEL_STRING IN1 , IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=SEL_STRING ( IN, IN1 , IN2 ) | |
| Note : IL, ST language programming need to insert variables IN, IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

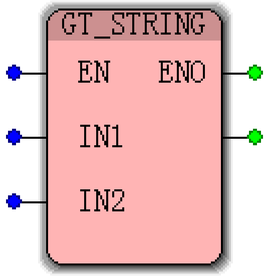
➤**SEL_STRING instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN ( G ) | BOOL | Select input |
| IN1 ( INO ) | STRING | First input |
| IN2 ( IN1 ) | STRING | Second input |
| OUT | STRING | Output<br>If IN=0, OUT=IN1; if IN=1, OUT=IN2 |

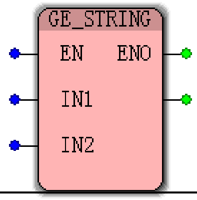
➤**Funtion and Action examples**

| String for selecting output by the value of variable V 419 | description |
|---|---|
|   | When the contact C 201 is O N time , the instruction execution, when the address V 419 is 0 when the output of the address V420 in a string; when the address V419 is . 1 , the output address V421 in the string : |

## 10.4.14 GT_STRIN (string is greater than)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The GT_STRING directive is used to determine if the first string is greater than the second string. | |
| LD IN1 |  |
| GT_STRING IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=GT_ STRING ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

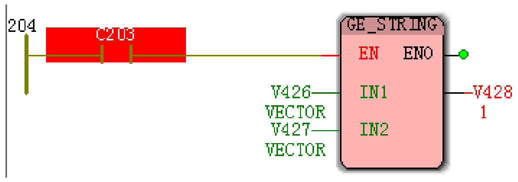
## 10.4.15 GE_STRING (string is greater than or equal to)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The GE_STRING directive is used to determine if the first string is greater than or equal to the second string. | |
| LD IN1 |  |
| GE_STRING IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=GE_STRING ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

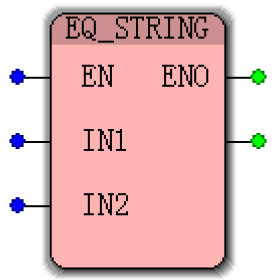
➢ **GE_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | First input |
| IN2 | STRING | Second input |
| OUT | BOOL | Output if the first character of the first input string is greater than or equal to the first<br>The first character of the two input strings is considered to be IN1>=IN2, OUT=1; otherwise OUT=0; the size of the characters is calculateDAccording to its ASCII code. |

➢**Funtion and Action examples**

| Comparison V42 . 6 and V427 two character strings | description |
|---|---|
|  | When the contact C 203 is O N time , the instruction execution, when the address of V42 . 6 is a string length greater than equal to the address V42 . 7 of string length when; address V42 . 8 outputs 1 ; No's output 0<br>Execution of results ; V42 . 6 > = V42 . 6 Output 1 |

## 10.4.16 EQ_STRING (string equals)

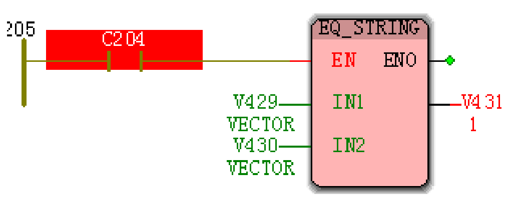
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The EQ_STRING instruction is used to determine if the first string is equal to the second string. | |
| LD IN1 |  |
| EO_STRING IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=EQ_STRING ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

➢ **EQ_STRING instruction processing data types**

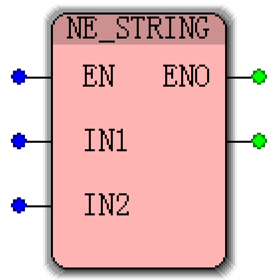
| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | First input |
| IN2 | STRING | Second input |
| OUT | BOOL | Output, if the first input string is exactly the same as the second input string, then IN1=IN2, OUT=1; otherwise OUT=0 |

➢**Funtion and Action examples**

| Comparison V42 . 9 and V430 two character strings | description |
|---|---|

| | |
|---|---|
|  | When the contact C 204 is O N time , the instruction execution, when the address of V42 . 9 is the string length is equal to the address V4 30 the string length of time; address V4 31 is the output 1 ; No's output 0<br>Execution of results ; V42 . 9 = V4 30 Output 1 |

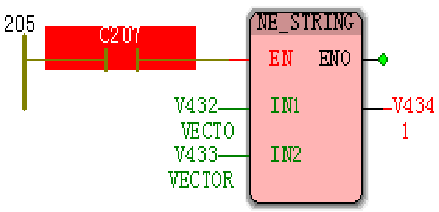
## 10.4.17 NE_STRING (string is not equal)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The NE_ STRING instruction is used to determine if the first string is not equal to the second string. | |
| LD IN1<br>NE_STRING IN2<br>ST OUT |  |
| ST programming language | |
| OUT:=NE_STRING ( IN1 , IN2 ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

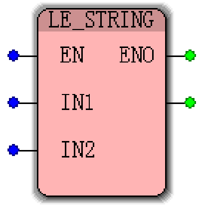
➤ **NE_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | First input |
| IN2 | STRING | Second input |
| OUT | BOOL | Output, if the first input string is not the same as the second input string, then OUT=1; otherwise OUT=0; this instruction is the opposite of EQ_STRING |

➤**Funtion and Action examples**

| Comparison V432 and V433 two character strings | description |
|---|---|
|  | When the contact C 207 is O N time , the instruction is executed, when the address V4 32 the string length is not equal to the address V4 33 is the string length of time; address V4 34 is the output 1 ; No's output 0<br>Execution of results ; V42 . 9 < > V4 30 Output 1 |

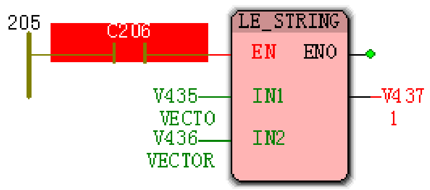
## 10.4.18 LE_STRING (string is less than or equal to)

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The LE_STRING directive is used to determine if the first string is less than or equal to the second string.</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>LE_STRING IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=LE_STRING ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

### ➢ LE_STRING instruction processing data types

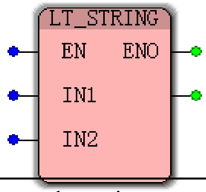
| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | First input |
| IN2 | STRING | Second input |
| OUT | BOOL | Output, if the first character of the first input string is less than or equal to the first character of the second input string, then IN1<=IN2, OUT=1; otherwise OUT=0; the size of the characters are pressed Its ASCII code calculation |

### ➢Funtion and Action examples

<table>
<tr><th>Comparison V4 35 and V436 two character strings</th><th>description</th></tr>
<tr><td>
</td><td>When the contact C 206 is O N time , the instruction is executed, when the address V4 35 of the string address length less than or equal V4 36 the string length of time; address V4 37 [ output 1 ; No's output 0<br>Execution of results ;<br>V4 35 < = V4 36 Output 1</td></tr>
</table>

## 10.4.19 LT_STRING (string is less than)

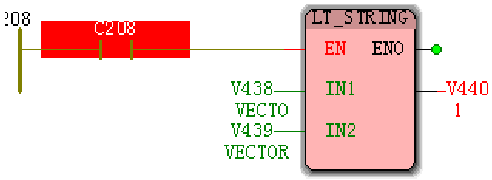
| IL programming language | LD, FBD programming language |
|---|---|

<table>
<tr><td colspan="2">Function: The LT_STRING instruction is used to determine if the first string is smaller than the second string.</td></tr>
<tr><td>LD IN1</td><td rowspan="5">
</td></tr>
<tr><td>LT_STRING IN2</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=LT_STRING ( IN1 , IN2 )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

➢ **LT_STRING instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN1 | STRING | First input |
| IN2 | STRING | Second input |
| OUT | BOOL | Output, if the first character of the first input string is less than the first character of the second input string, then IN1<IN2, OUT=1; otherwise OUT=0; the size of the character is in accordance with its ASCII code Calculation |

➢**Funtion and Action examples**

| Comparison V4 38 is the V439 two character strings | description |
|---|---|
|   | When the contact C 206 is O N time , the instruction is executed, when the address V4 38 is a string of length less than the address V4 39 the string length of time; address V4 40 outputs 1 ; No's output 0<br>Execution of results ;<br>V4 35 <V4 36 Output 1 |

# 10.4.20 STRING_TO_* (converts strings to other types)

| String FU | Features |
|---|---|
| STRING_TO_* directive | The STRING_TO_* instruction is used to convert a STRING type data into a valiDANY value type. There are 12 instructions to convert the STRING type data into BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL. , LREAL and TIME and other types. |

**STRING_TO_* conversion instructions**

| instruction | input value | output value | description |
|---|---|---|---|
| STRING_TO_BYTE | STRING | BYTE | Input format 0~255* , output is BYTE type |
| STRING_TO_WORD | STRING | WORD | Input format 0~65535* ” , the output is WORD type |
| STRING_TO_DWORD | STRING | DWORD | Input format 0~4,294,967,295* , output as DWORD type |
| STRING_TO_SINT | STRING | SINT | Input format -128~127* , output is SINT type |
| STRING_TO_INT | STRING | INT | Input format -32768~-32767* , output is INT type |
| STRING_TO_DINT | STRING | DINT | Input format -2,147,483,648~ 2,147,483,647 " , output DINT |
| STRING_TO_USINT | STRING | USINT | Input format 0~255 ” , output is USINT type |
| STRING_TO_UINT | STRING | UINT | Input format 0~65535 " , the output is UINT type |
| STRING_TO_UDINT | STRING | UDINT | Input format 0~4,294,967,295 ” , the output is UDINT type |
| STRING_TO_REAL | STRING | REAL | Input format decimal * , output is REAL type |
| STRING_TO_LREAL | STRING | LREAL | Input format decimal * , output is LREAL type |
| STRING_TO_TIME | STRING | TIME | Input format T#0~T#4,294,967,295 " milliseconds, output seconds |
| *: In additioNSTRING_TO_TIME external command, can bring the corresponding BYTE #, WORD # other type prefix, and " 16 # " prefix, the prefix may not ; input character valid values sign and decimal numerals 0 to 9; the numbers appear The other characters and the recurring numbers are invalid. | | | |

**usage (take STRING_TO_BYTE as an example)**

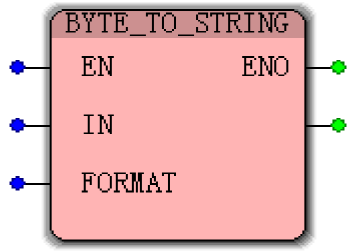
| Input and output | type of data | description |
|---|---|---|
| IN | STRING | Enter a character or string, input format 0~255 |
| OUT | BYTE | Output, convert input to BYTE type |

During the numerical conversion process, the following error may occur :

☞ input has an illegal prefix, such as INT##123, INT# is an illegal prefix ;

☞ input value exceeds the range of the output data type, such as the input value of 1024 of STRING_TO_BYTE instruction , 1024 is greater than the range of 0~255 of BYTE type .

## 10.4.21 *_TO_STRING (other types are converted to strings)

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: *_TO_STRING is divided into 12 instructions, which can convert BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, REAL, LREAL and TIME to STRING type.<br>Usage ( take BYTE_TO_STRING as an example )</td></tr>
<tr><td>LD IN</td><td rowspan="5">
</td></tr>
<tr><td>BYTE_TO_STRING FORMAT</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=BYTE_TO_ STRING ( IN, IN1 )</td></tr>
</table>

**BYTE_TO_STRING instruction processing data type**

| Input and output | type of data | description |
|---|---|---|
| IN | BYTE | Input range 0~255 |
| FORMAT | STRING | Valid format : %c, %x, %u, default %x |
| OUT | STRING | Output, when the format is %c , the output is the input ASCII code ; when the format is %x , the output is the input hexadecimal number ; when the format is %u , the output is the input unsigned decimal number. |

➢ **WORD_TO_STRING instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN | WORD | Input range 0~65535 |
| IN1 ( FORMAT ) | String | Valid format : %x, %u, default %x |
| OUT | STRING | Output, when the format is %X , the output is the input hexadecimal number ; when the format is %U , the output is the input unsigned decimal number |

➢ **DWORD_TO_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|

| | | |
|---|---|---|
| IN | DWORD | Input range 0~4,294,967,295 |
| IN1 ( FORMAT ) | String | Valid format : %x, %u, default %x |
| OUT | STRING | Output, when the format is %X , the output is the input hexadecimal number ; when the format is %U , the output is the input unsigned decimal number |

- **SINT_TO_STRING instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN | SINT | Input range -128~127 |
| IN1 ( FORMAT ) | String | Valid format : %d, default %d |
| OUT | STRING | Output, output as input signed decimal number |

- **INT_TO_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | INT | Input range is 32768~32767 |
| IN1 ( FORMAT ) | String | Valid format : %d, default %d |
| OUT | STRING | Output, output as input signed decimal number |

- **DINT_TO_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | DINT | Input range -2,147,483,648~2,147,483,647 |
| IN1 ( FORMAT ) | String | Valid format : %d, default %d |
| OUT | STRING | Output, output as input signed decimal number |

- **USINT_TO_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | USINT | Input range 0~255 |
| FORMAT | String | Valid format : %u, default %u |
| OUT | STRING | Output, output as the input symbol decimal number |

- **UINT_TO_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | UINT | Input range 0~65535 |
| IN1 ( FORMAT ) | String | Valid format : %U, default %U |
| OUT | STRING | Output, output as input unsigned decimal number |

- **DINT_TO_STRING data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | UDINT | Input range 0~4,294,967,295 |
| IN1 ( FORMAT ) | String | Valid format : %u, default %u |
| OUT | STRING | Output, output as input unsigned decimal number |

- **REAL_TO_STRING instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN | REAL | Input range - 3.402823466 E+38... -1.175494351 E-38 and +1.175494351 E-38... +3.402823466 E+38 |
| IN1 ( FORMAT ) | String | Valid format : %e, %f, default %e |
| OUT | STRING | Output, when the format is %e , the output is the floating point number represented by the scientific notation of the input ; when the format is %f , the output is the floating point number of the input. |

The data type processed by the LREAL_T0_STRING instruction is the same as REAL_TO_STRING except that the input range is different.

- **TIME_TO_STRING data processing instruction type**

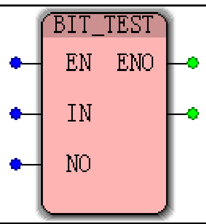
| Input and output | type of data | description |
|---|---|---|
| IN | TIME | Enter the TIME type, range T#O~T#4,294,967,295 " milliseconds, such as T#1 MS |
| IN1 ( FORMAT ) | String | Valid format : %u, default %u |
| OUT | STRING | Output, when the format is %u , the output is the TIME type that represents the input as unprefixed , in milliseconds, such as input T#1 S , output 1000; if no format is specified, the output represents the input as a prefixed TIME type If you input T#1 S , the output is T#1000MS |

## 1 0.5 Bit operation function BIT_UTIL

The bit manipulation function is a program organization unit POU with multiple input parameters and one output parameter . It can reaDAnd write the input bit string. The bit operation function is abbreviateDAs BIT_UTIL . The bit manipulation function needs to be inserted separately into the firmware library. The instruction contained in the bit manipulation function ( in the editing wizard, select from the drop-down list " BIT_UTIL " )

| BIT_UTIL function block | | | |
|---|---|---|---|
| BIT_TEST | GET_CHAR | GET_LSB | GET_MSB |
| I_BIT_IN_BYTE | I_BIT_IN_WORD | I_BIT_IN_DWORD | PARITY_BYTE |
| PARITY_WORD | PARITY_DWORD | R_BIT_IN_BYTE | R_BIT_IN_WORD |
| R_BIT_IN_DWORD | S_BIT_IN_BYTE | S_BIT_IN_WORD | S_BIT_IN_DWORD |
| SET_LSB | SET_MSB | STRING_TO_BUFFER | SWAP |

## 10.5.1 BIT_TEST (Read bit value instruction in bit string )

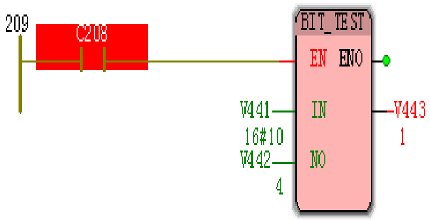
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The BIT_TEST instruction is used to read the value of a single bit in the input bit string. | |
| LD IN |   |
| BIT_TEST IN1 | |
| ST OUT | |
| ST programming language | |
| OUT:=BIT_T0_EST ( IN, IN1 ) | |
| Note : IL, ST language programming needs to insert variables IN, IN1 and OUT or use constants in the variable worksheet of the current POU | |

➢ **BIT_TEST data processing instruction type**

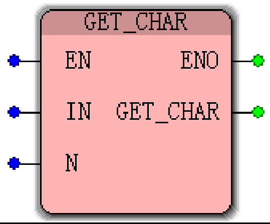
| Input and output | type of data | description |
|---|---|---|
| IN ( IN ) | ANY_BIT | Input bit string |
| IN1 ( NO ) | SINT | Bit string to be read in the first NO bit range : BYTE 0 ~. 7, WORD0 ~ 15, ~ 31 is DWORD0 |
| OUT | BOOL | Output, enter the value of |

| | | |
|---|---|---|
| | | the NOth bit in the bit string |

➢**Funtion and Action examples**

| Read the value of the 4th bit in the address V441 byte ; | description |
|---|---|
|   | When the contact C 208 is O N time , the instruction is executed , the reaDAddress V441 in the first 4 bit value , V442 address output of the corresponding bit status : Note (the first of several from 0 to start calculation) |

## 10.5.2 GET_ CHAR (Remove the character instruction in the string )

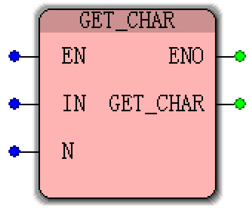
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The GET_CHAR instruction is used to get a character from the input string and the ASCII code represents the character. | |
| LD IN<br>GET_CHAR IN1<br>ST OUT<br>ST programming language<br>OUT:=GET_CHAR ( IN, IN1 ) |   |
| Note : IL, ST language programming needs to insert variables IN, IN1 and OUT or use constants in the variable worksheet of the current POU | |

➢ **to GET_CHAR data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN ( IN ) | ANY_BIT | Input string |
| IN1 ( N ) | INT | The Nth character in the string to be fetched , ranging from 0 to 32767 |
| OUT ( GET_CHAR ) | INT | Output, input the ASCII value of the Nth character in the string |

Note : This command is not supported by the current software version.

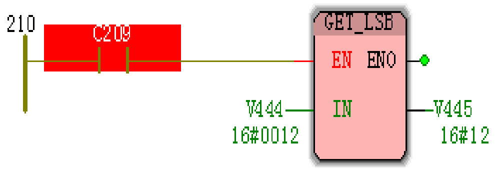
## 10.5.3 GET_LSB ( Remove the lower 8 -bit instruction in the bit string )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The GET_LSB instruction is used to read the value of the lower byte ( the Less Significant BYTE ) in the input bit string . | |
| LD IN |  |
| GET_LSB | |
| ST OUT | |
| ST programming language | |
| OUT:=GET_LSB ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN , and OUT or use constants in the variable worksheet of the current POU | |

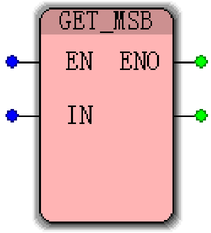
➢ **GET_LSB instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN | WORD | Input bit string |
| OUT | BYTE | Output, lower byte ( lower 8 bits ) value |

➢**Funtion and Action examples**

| A reaDAddress V444 low . 8 -bit value | description |
|---|---|
|  | When the contact C 209 is O N time , the instruction is executed , the address of V444 is low . 8 bit of the value of the state taken out , stored in the address V445 in ;<br>Execution result: 16 #0012 ( word ) -- 16 #12 ( byte ) |

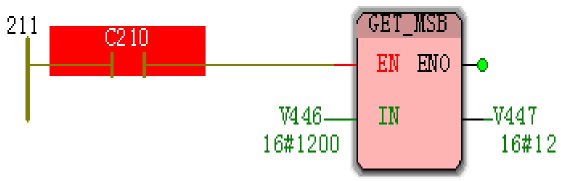
## 10.5.4 GET_MSB ( Remove the high 8 -bit instruction in the bit string )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: GET_MSB instruction for reading the most significant byte of the input bit string ( The Most Significant BYTE ) values. | |
| LD IN<br>GET_MSB<br>ST OUT<br>ST programming language<br>OUT:=GET_MSB ( IN ) ; |  |
| Note : IL, ST language programming needs to insert variables IN , and OUT or use constants in the variable worksheet of the current POU | |

➢ **GET_MSB instruction processing data types**

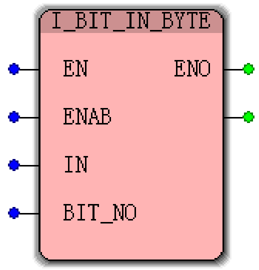
| Input and output | type of data | description |
|---|---|---|
| IN | WORD | Input bit string |
| OUT | BYTE | Output, the value of the highest byte ( higher 8 bits ) |

➢**Funtion and Action examples**

| Read the value of the upper 8 bits of the address V446 | description |
|---|---|
|  | When the contact C 210 is O N time , the instruction is executed , the address of V44 . 6 high . 8 bit of the value of the state taken out , stored in the address V44 . 7 in ;<br>Execution result: 16 #1200 ( word ) -- 16 #12 ( byte ) |

## 10.5.5 I_BIT_IN* ( Invert the single bit in the bit string)

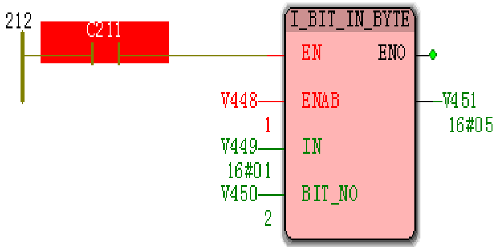
| IL programming language | LD, FBD programming language |
|---|---|

<table>
<tr><td colspan="2">Function: The I_BIT_IN* instruction is used to invert a single bit in the input bit string, including the I_BIT_IN_WORDAnd I_BIT_IN_DWORD instructions, and can handle input bit strings of BYTE, WORDAnd DWORD types.</td></tr>
<tr><td>LD ENAB</td><td rowspan="5">
</td></tr>
<tr><td>I_ BIT_IN_BYTE IN , BIT_ NO</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=I_ BIT_IN_ BYTE ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN1, IN, IN2 and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

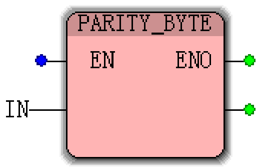
## ➢ I_BIT_IN_BYTE data processing instruction type

| Input and output | type of data | description |
|---|---|---|
| IN1 ( ENAB ) | BOOL | Enable |
| IN | BYTE<br>WORD<br>DWORD | Input bit string |
| IN2<br>( BIT_N O ) | SINT | The NO bit in the bit string to be operated , the value range is 0~7 for BYTE , 0~15 for WORD , 0~31 for DWORD ( other values are invalid ) |
| OUT | BYTE | Output, when IN1 is FALSE , the output OUT is equal to the input IN; when IN1 is TRUE , the output OUT is the input IN of NO value of the bit negated |

## ➢Funtion and Action examples

| NegateDAddress V449 in the second of the two values of the bits | description |
|---|---|
|   | When the contact C 211 is O N time , the instruction is executed , and when the ENAB is ON , the inverted V449 in the second 2 bit , outputs; if ENAB is FALSE when , without inverted output self itself;<br>Execution result: 16#01 ( byte ) --- 16#05 ( byte ) |

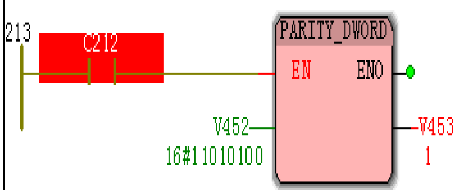
## 10.5.6 PARITY_* (parity instruction for bit string )

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: PARITY_ * instruction to check the input bit string is . 1 the number of bits is odd or even number, an odd number if the output is FALSE , is even if the output is TRUE, PARITY_ * instructions comprise team RITY_BYTE, PARITY_WORDAnd PARITY _DWORD directive, capable of handling input bit strings of BYTE, WORDAnd DWORD types.</td></tr>
<tr><td>LD IN</td><td rowspan="5">
</td></tr>
<tr><td>PARITY_BYTE</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=PARITY_BYTE ( IN )</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet</td></tr>
</table>

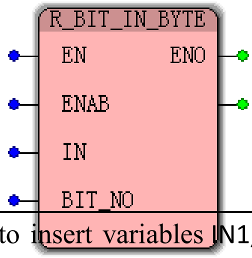
**➢PARITY_BYTE data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN | BYTE<br>WORD<br>DWORD | Input bit string |
| OUT | BOOL | When the input bit string as a number of bits is an odd number, the output is FALSE; is a number of bits of an even number of time ( including 0 th to 1 -bit ) , output is TRUE |

**➢Funtion and Action examples**

| Check the number of 1 in the address V452 for output | description |
|---|---|
|   | When the contact C 212 is O N time , the instruction is executed , the address V4 52 is in is one of the number removed , when is 1 is an odd number , the address V453 output is 0 ; when a is 1 the number is an even number when ( or all 0 time ) address V453 output 1<br>Execution result: 16#11010100 ( D WORD ) --- - 1 ( BOOL ) |

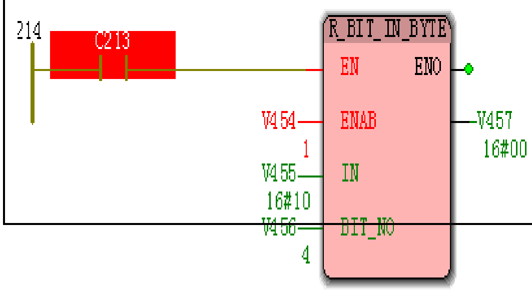
# 10.5.7 R _BIT_IN_* ( instruction of a single position 0 in a bit string )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The R _BIT_IN_* instruction is used to input a single bit position 0 in the input bit string , including the R_BIT_IN_BYTE and R_BIT_IN_DWORD instructions, to handle input bit strings of BYTE, WORDAnd DWORD types. | |
| LD ENAB |  |
| R_BIT_IN_ BYTE IN , BIT_NO | |
| ST OUT | |
| ST programming language | |
| OUT:=R_BIT_IN_ BYTE ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN, IN2 and OUT or use constants in the current POU variable worksheet | |

➢ **R_BIT_IN_BYTE instruction processing data types**

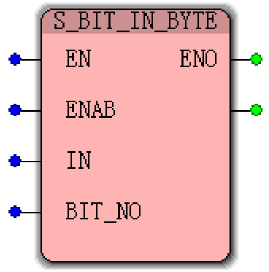
| Input and output | type of data | description |
|---|---|---|
| IN1 ( ENAB ) | BOOL | Enable |
| IN | BYTE<br>WORD<br>DWORD | Input bit string |
| IN2 ( BIT_NO ) | SINT | A first bit string to be operated NO bit range : BYTE when 0 ~ 7, WORD when 0 ~ 15 DWORD when 0 to 31 ( the other value is invalid ) |
| OUT | BYTE | Output, when IN1 is FALSE , the output OUT is equal to the input IN; when IN1 is TRUE , the output OUT is the input IN of the NO position 0 value of |

➢**Funtion and Action examples**

| Reset the status of the 4th bit in address V455 | description |
|---|---|
|  | When the contact C 213 is O N time , the instruction is executed , when the ENAB is ON time , reset address V455 in the first of four of the state; when ENAB is FALSE when the output address V455 current value; |

| | |
|---|---|
| | Execution result: 16#10 ( BYTE ) ---<br>16#00 ( BYTE ) |

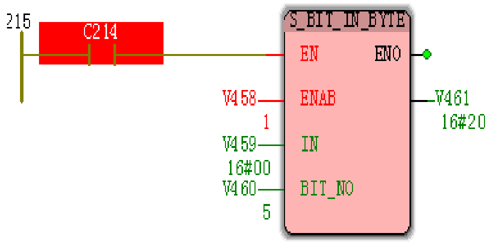
## 10.5.8 S_BIT_IN_* ( 1 instruction in a single bit in the bit string )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: SBIT_IN_ * instruction is used to position a single input bit string 1 , comprising S_BIT_IN_BYTE, S_BIT_IN_WORDAnd S_BIT _IN_DWORD instructions, capable of handling BYTE, WORD , and DWORD types of input bit sequence. | |
| LD ENAB<br>S_BIT_IN_BYTE IN, BIT_NO<br>ST OUT<br>ST programming language<br>OUT:=S_BIT_IN_BYTE ( IN ) |  |
| Note : IL, ST language programming needs to insert variables IN1, IN, IN2 and OUT or use constants in the current POU variable worksheet | |

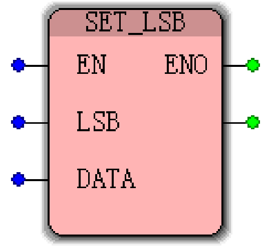
**➢S_BIT_IN_BYTE instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN1 ( ENAB ) | BOOL | Enable |
| IN | BYTE<br>WORD<br>DWORD | Input bit string |
| IN2 ( BIT_NO ) | SINT | The NO position in the bit string to be operated , the value range is 0~7 for BYTE , 0~15 for WORD , 0~31 for DWORD ( other values are invalid ) |
| OUT | BYTE | Output, when IN1 is FALSE , the output OUT is equal to the input IN; when IN1 is TRUE , the output OUT is the input IN of NO position 1 value of |

**➢Funtion and Action examples**

| Set the status of the 5th bit in address V458 | description |
|---|---|
|   | When the contact C 214 is O N time , the instruction is executed , when the ENAB is ON time , set address V45 . 8 in the first of five of the state; when ENAB is FALSE when the output address V455 current value;<br>Execution result: 16#00 ( BYTE ) --- 16#20 ( BYTE ) |

## 10.5.9 SET_LSB ( Write instructions to the lower 8 bits in the bit string )

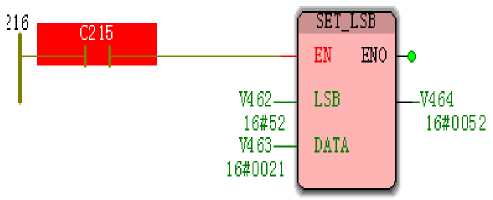
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SET_LSB instruction is used to write a value to the lower byte ( the Less Significant BYTE ) in the input bit string . | |
| LD IN1<br>SET_LSB IN<br>ST OUT<br>ST programming language<br>OUT:=SET_LSB ( IN1 , IN ) |   |
| Note : IL, ST language programming needs to insert variables IN1, IN and OUT or use constants in the variable worksheet of the current POU | |

➢**SET_LSB instruction processing data types**

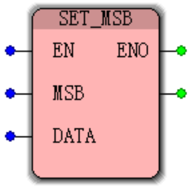
| Input and output | type of data | description |
|---|---|---|
| IN1 ( LSB ) | BYTE | The value to be written, range 0~255 |
| IN ( DATA ) | WORD | Input bit string, 0~65535 |
| OUT | WORD | Output, change the lower byte ( lower 8 bits ) of the input bit string to IN1 |

➢**Funtion and Action examples**

| Change the value of the lower 8 bits of address V463 to 16 #52 | description |
|---|---|

| | |
|---|---|
|  | When the contact C 215 is O N time , the instruction is executed , the address of V463 is low . 8 bit values to 16 # 52 , and then output :<br><br>Results of execution : 16 #0021-----16#0052 |

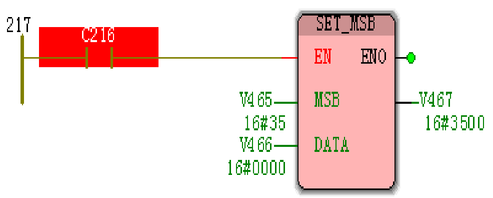
## 10.5.10 SET_MSB ( the high bit string 8 write digit command )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: SET_MSB instruction being for the most significant byte string input bit ( The Most Significant BYTE ) Write value. | |
| LD IN1 |  |
| SET_MSB IN | |
| ST OUT | |
| ST programming language | |
| OUT:=SET_MSB ( IN1 , IN ) | |
| Note : IL, ST language programming needs to insert variables IN1, IN and OUT or use constants in the variable worksheet of the current POU | |

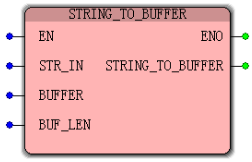
➢ **SET_MSB instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN1 ( MSB ) | BYTE | The value to be written, range 0~255 |
| IN ( DATA ) | WORD | Input bit string, 0~65535 |
| OUT | WORD | Output, change the highest byte ( high 8 bits ) of the input bit string to IN1 |

➢**Funtion and Action examples**

| Change the value of the upper 8 bits of address V466 to 16 #35 | description |
|---|---|
|  | When the contact C 216 is O N time , the instruction is executed , the address of V466 high . 8 bit values to 16 # 35 , and then output :<br><br>The result of the execution : 16 #0000-----16#3500 |

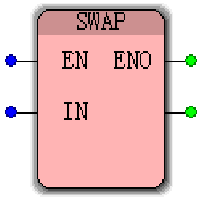
## 10.5.11 STRING_TO_BUFFER (copy string to buffer instruction )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The STRING_TO_BUFFER instruction is used to copy a string into a buffer, which is a byte array . | |
| LD IN |  |
| STRING_TO_BUFFER IN1 , IN2 | |
| ST OUT | |
| ST programming language | |
| OUT:=STRING_TO_ BUFFER ( IN1,IN ) | |
| Note : IL, ST language programming needs to insert variables IN, IN1, IN2 and OUT or use constants in the current POU variable worksheet | |

➢ **STRING_TO_BUFFER instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN ( STR_IN ) | STRING | Input string |
| IN1 [0] ( BUFFER ) | BYTE | An element of a buffer ( byte array ) , such as IN1 [0] or IN1[1] |
| IN2 ( BUF_LEN ) | INT | Number of characters copied into the buffer |
| OUT ( STRING_TO_BUFFER ) | INT | The output is not defined yet. In fact, the character is copied into the buffer IN1 . An element ( byte ) in IN1 stores the ASCII code of the copied character . |

## 10.5.12 SWAP (swapping high byte and low byte instructions )

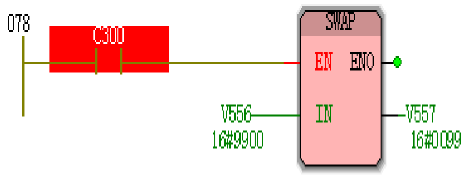
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The SWAP instruction is used to swap the position of the highest byte ( MSB ) and lower byte ( LSB ) of the input bit string . | |
| LD IN |  |
| SWAP | |
| ST OUT | |
| ST programming language | |
| OUT:=SWAP ( IN ) | |

| | |
|---|---|
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**数据 Data type processed by SWAP instruction**

| Input and output | type of data | description |
|---|---|---|
| IN ( IN ) | WORD | Input bit string |
| OUT ( STRING_TO_BUFFER ) | WORD | Output bit string, exchange the high and low bytes of the input bit string and output |

➢**Funtion and Action examples**

| The address V556 ( 16 # 9900 ) is high . 8 bits of the low . 8 bit swap | description |
|---|---|
|  | When the contact C300 is ON , the instruction is executed to exchange the upper 8 bits of the address V556 with the lower 8 bits ; Execution result: 16 # 9900 exchange into a 16 # 0099 |

## 10.6 ProConOS Features

FILE_OPEN, FILE_CLOSE , FILE_SEEK , FILE_TELL , FILE_READ , FILE_WRITE, FILE_REMOVE instructions in ProConOS are not available.

### 10.6 .1 BUF type conversion to other types

BUF type data can be divided into 12 function blocks, which can copy basic data types from byte stream to variables, arrays or elements of user-defined structure, that is, convert to BYTE, WORD, DWORD, SINT, INT, DINT, respectively. USINT, UINT , UDINT , REAL, STRING, TIME and other data, it is mainly used for data transfer or to perforMCommunication in the application on different hardware platforms.

The BUF type data conversion instruction in ProConOS needs to be selected from the drop-down list " ProConOS " in the editing wizard . Source data must be an array of bytes ( data type BYTE ) , but also of ANY_BIT ( BOOL excluding ) or ANY_INT type ( number of bytes is converted to data type stored can not exceed the target ) .

**BUF type data conversion instruction**

| instruction | source data | Converted data | description | |
|---|---|---|---|---|
| BUF _ TO _BYTE | BUF | BYTE | The data format of the buffer is MOTOROLA or I NTEL . The source data can also be ANY_BIT ( BOOL ).<br>Except ) or A NY_INT | The number of bytes converted is 1 |
| BUF _ TO _WORD | BUF | WORD | | The number of bytes converted is 2 |
| BUF _ TO _DWORD | BUF | DWORD | | The number of bytes converted is 4 |
| BUF _ TO _SINT | BUF | SINT | | The number of bytes converted is 1 |
| BUF _ TO _INT | BUF | INT | | The number of bytes converted is 2 |
| BUF _ TO _DINT | BUF | DINT | | The number of bytes converted is 4 |
| BUF _ TO _USINT | BUF | USINT | | The number of bytes converted is 1 |
| BUF _ TO _UINT | BUF | UINT | | The number of bytes converted is 2 |
| BUF _ TO _UDINT | BUF | UDINT | | The number of bytes converted is 4 |
| BUF _ TO _RAEL | BUF | REAL | The data format of the buffer is MOTOROLA | Enter the hexadecimal code of IEEE 754 floating point number. The number of bytes converted can be 4 |
| BUF _ TO _ STRING | BUF | STRING | The data format of the buffer is MOTOROLA or I NTEL | Enter the ASCI I code as a character. The number of bytes to be converted can be a positive integer such as 1, 2... |
| BUF _ TO _ TIME | BUF | TIME | The data format of the buffer is INTEL | The input is a hexadecimal code in milliseconds. The number of bytes converted is 4 and the |

| | | | | |
|---|---|---|---|---|
| | | | | output unit is seconds. |

➢ **BUF_TO_ * data processing instruction type**

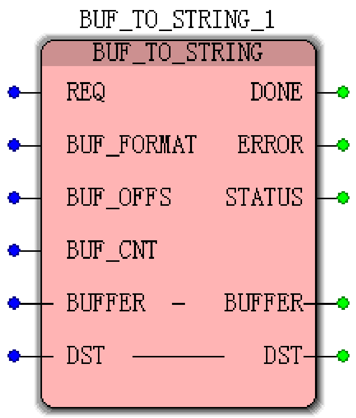
| Pin | Features | type of data | description |
|---|---|---|---|
| REQ | Input | BOOL | Valid on rising edge |
| BUF_FORMAT | Input | BOOL | TRUE indicates that the buffer data is in MOTOROLA format ; FALSE indicates INTEL format |
| BUF_OFFS | Input | DINT | The starting byte number of the buffer being converted, 0 is the first byte of the buffer |
| BUF_CNT | Input | DINT | The number of bytes converted in the buffer |
| BUFFER | Input - output | ARRAY | Buffer, a byte array |
| DST | Input - output | Array or ANT | The storage area, the content of the converted byte is placed here, the type of DST should be consistent with the specific BUF_TO_* type, such as the DST of the BUF_TO_BYTE instruction must be BYTE type |
| DONE | Output | BOOL | After the conversion is complete, set to 1 until REQ is 0. |
| ERROR | Output | BOOL | The conversion is normal, 0 , otherwise set to 1 |
| STATUS | Output | INT | If the conversion is not normal, an error code is given, as shown in the table below. |

Note : The data storage order of MOTOROLA and INTEL microprocessor is different. The INTEL format is high and low byte array, and the MOTOROLA format is low and high byte array.

| error code | description |
|---|---|
| 0 | The conversion process is completed normally |
| 1 | BUFFER and DST output-output type error |
| 2 | Exceeding the length of the buffer, the number of bytes to be copied BUF_CNT is larger than the number of available bytes of the buffer BUFFER |
| 3 | Exceeding the length of the storage area, the number of bytes to be copied BUF CN T exceeds the length of the storage area |
| 4 | This data type is not supported |
| 5 | The length of the byte to be converted does not correspond to the byte length of the storage area. The former number of bytes must be divisible by the latter number of bytes. |

| | |
|---|---|
| 6 | Conversion of INTEL/MOTOROLA failed |
| 7 | The length of the string is not appropriate. For the data type string, it is necessary to do additional checks. |
| 8 | Storage areAData type error |
| 9 | BUF_OFFS value is incorrect |
| 10 | BUF_CNT value is incorrect |
| 11 | The buffer is the same as the storage area address |

Note : The number of bytes available for buffer BUFFER - starting from the BUF_OFFS bytes in the buffer to the last word

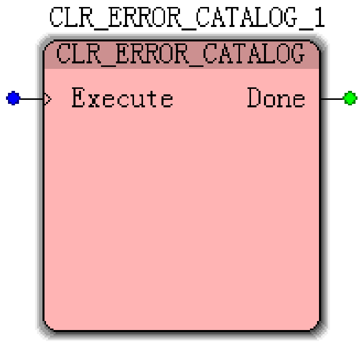
| IL programming language | LD, FBD programming language |
|---|---|
| See help file |  |
| ST programming language | |
| See help file | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

## 10.6.2 Other types are converted to BUF type

Other types can be converted to BUF type data, and the basic data types in the variables, arrays or elements of the user-defined structure can be copied into the byte stream. There are 12 instructions, respectively BYTE, WORD, DWORD, SINT , INT , Data such as DINT, USINT, UINT, UDINT, REAL, STRING, TIME are converted into BUF type data, which is mainly used for data transfer or communication in applications on different hardware platforms.

Converting other types in ProConOS to BUF type instructions requires selecting " ProConOS " from the drop-down list in the Edit Wizard . The storage area must be a byte array ( data type is BYTE ) or ANY_BIT ( except BOOL ) or ANY_INT . The usage of these instructions is similar to the above, " BUF type conversion to other types " , and will not be described here.

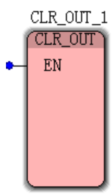
## 10.6.3 CLR_ERROR_CATALOG ( except for the complete error directory )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The CLR_ERROR_CATALOG instruction is used to delete the complete error directory. | |
| LD IN |  |
| CLR_ERROR_CATALOG | |
| ST OUT | |
| ST programming language | |
| OUT:=CLR_ERROR_CATALOG ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢ **CLR_ERROR_CATALOG data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN ( Execute ) | BOOL | Valid on rising edge |
| OUT ( Done ) | BOOL | 0: The error directory cannot be deleted.<br>1: The error directory was successfully deleted. |

## 10.6.4 CLR_OUT ( Set the output of the I/O image to 0 pointer )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The CLR_OUT instruction is used to set the output of the I/O image area to 0. | |
| LD IN |  |
| ST INCLR_OUT_1.EN | |
| CAL CLR_OUT_1 | |
| ST programming language | |
| CLR_OUT_1 ( EN:= ( IN )) | |
| Note : IL, ST language programming needs to insert variable IN or use constant in the | |

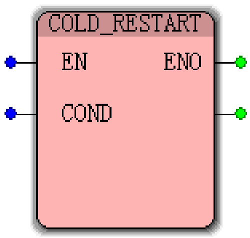
| |
|---|
| current POU variable worksheet |

➢ **CLR_OUT data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN ( Execute ) | BOOL | If TRUE , set all outputs of the I/O image to zero |

Note : The CLR_OUT instruction is temporarily unavailable.

## 10.6.5 COLD_RESTART ( PLC cold start )

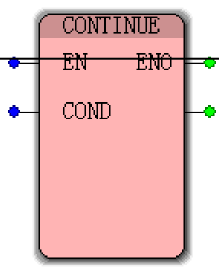
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The COLD_RESTART instruction is used to cold start the PLC . Initialize all dataADuring a cold restart. If the program has a stack overflow, a string error, or ADivide by 0 , you can call this command to automatically restart the execution of the program. | |
| LD IN |   |
| COLD_RESTART | |
| ST OUT | |
| ST programming language | |
| COLD_RESTART_1 ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

➢**COLD_RESTART data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| IN ( COND ) | BOOL | If TRUE , perform a cold restart |
| OUT | BOOL | OUT is TRUE if COND=TRUE and can be cold started |

Note : The COLD_RESTART instruction is temporarily unavailable.

## 10.6.6 CONTINUE ( continue running the program )

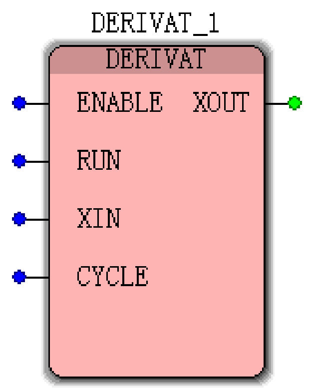
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The CONTINUE instruction is used to run other programs when there is a temporary error in the program ( such as a timer error ) . This feature should not be used for exception events such as divide by 0 , stack overflow, bus errors, and boundary errors. | |
| LD IN |   |
| CONTINUE | |

| | |
|---|---|
| ST OUT | |
| ST programming language | |
| CONTINUE ( IN ) | |
| Note : IL, ST language programming needs to insert variables IN and OUT or use constants in the current POU variable worksheet | |

**数据 Data type processed by CONTINUE instruction**

| Input and output | type of data | description |
|---|---|---|
| IN ( COND ) | BOOL | If TRUE , the execution of the program is continued. |
| OUT | BOOL | If COND = TRUE , and you can execute the program, compared with TRUE |

# 10.6.7 DERIVAT ( differential )

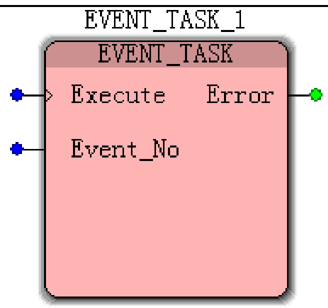
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The DERIVAT instruction is used to perform time differential calculation on data. When using the differential instruction, the task type of the POU needs to be set to periodic scan ( CYCLIC ) , and the scan period, task type setting and scan perioDAre set according to their own needs. See the programming model in this manual ◊ Hardware ◊ Tasks.</td></tr>
<tr><td>LD ENABLE</td><td rowspan="14">
</td></tr>
<tr><td>ST DERIVAT_1.ENABLE</td></tr>
<tr><td>LD RUN</td></tr>
<tr><td>ST DERIVAT_1.RUN</td></tr>
<tr><td>LD XIN</td></tr>
<tr><td>ST DERIVAT_1.XIN</td></tr>
<tr><td>LD CYCLE</td></tr>
<tr><td>ST DERIVAT_1.CYCLE</td></tr>
<tr><td>CAL DERIVAT_1</td></tr>
<tr><td>LD DERIVAT_1.XOUT</td></tr>
<tr><td>ST XOUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>DERIVAT_1 ( ENABLE:= IN, RUN:=IN1, XIN:=IN2, CYCLE:=T ) ;</td></tr>
</table>

| | |
|---|---|
| OUT:=DERIVAT_1.XOUT; | |
| Note : IL, ST language programming need to insert variables ENABLE, RUN, etc. in the current POU variable worksheet or use constants | |

➢ **DERIVAT data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| ENABLE | BOOL | Execution function block when TRUE |
| RUN | BOOL | FALSE , the function block is pauseDAnd the output is set to 0 |
| XIN | REAL | input value |
| CYCLEX | TIME | When Q constant, iNSeconds, this instruction is actually [the XIN ( n- ) -XIN ( n--. 1 ) ] / CYCLE differential |
| OUT | REAL | Output value, differential result |

## 10.6.8 EVENT_TASK ( trigger event )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The EVENT_TASK instruction is used to trigger an event task. The event number can be defined. If the event number exists in an event task of the user, it is activated, that is, the program is assigned to the event task. | |
| LD START |  |
| ST EVENT_TASK_1.Execute | |
| LD IN | |
| ST EVENT_TASK_1.Event No | |
| CAL EVENT_TASK_1 | |
| LD EVENT_TASK_1.Erroe | |
| ST OUT | |
| ST programming language | |
| EVENT_TASK_1 ( Execute:=START, Event_No:=IN ) ;<br>OUT:=EVENT_TASK_1.Error; | |
| Note : IL, ST language programming need to insert variables START , IN and OUT or use constants in the current POU variable worksheet | |

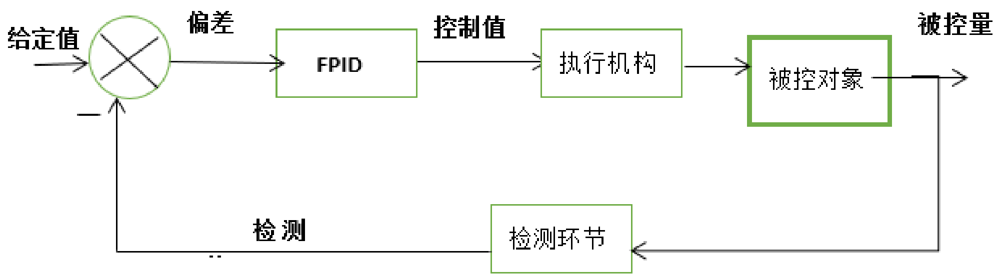
➢ **EVENT_TASK data processing instruction type**

| Input | type of data | description |
|---|---|---|
| START ( Execute ) | BOOL | Valid on rising edge |
| IN ( Event_No ) | UINT | Event number |
| OUT ( Error ) | BOOL | Error code : |

| | | |
|---|---|---|
| | | 0:1 no error occurred ; The event number is not within the valid range of the value |

## 10.6.9 FPID

| ProConOS function | Features |
|---|---|
| FPID | The FPID commanDAutomatically calculates the control value according to the deviation between the set value and the detected value, so that the detected value can track the set value, and the set value is a value that is expected to be maintained by the controlled device, and the detected value is The value detected by the behavior of the control device, and the control value is a value that controls the behavior or indirect behavior of the controlled device, thus forming a closed-loop control loop. As shown in the figure below, the FPID is the core part. |




Deviation = set value - detected value.

When the FPID instruction is in the automatic working mode, its output value is the result of the calculation after the PID operation, as follows

Output = = kp $\left(e(t) + \frac{1}{Ti}\int_0^t e(\tau)_{d\tau} + T_d \frac{de(t)}{dt}\right)$ FPID instruction in manual mode, the output value is equal to the manual output value

➢ **FPID data processing instruction type**

| Input and output | Types of | description |
|---|---|---|
| REMOTE | BOOL | Remote setting for TRUE and local setting for FALSE |
| AUTO | BOOL | TRUE when FPID to self- move mode of operation, FALSE when the manual mode of operation |
| DIRECTN | BOOL | TRUE is positive ( detection value is higher than the set value ◊ control value rises ) , FALSE is the reaction ( detection value is higher than the set |

| | | |
|---|---|---|
| | | value ◊ control value decreases ) |
| INTLCK | BOOL | TRUE when FPID output Yout is forced, by a force values FPID input INTLCKV set ; FALSE output does not force ( output Yout via a proportional integral derivative Calcd value or manually ) |
| Tscan | REAL | Time constant, iNSeconds, can generally be set to REAL#0.1 o The larger the Tscan value, the stronger the control effect |
| Yman | REAL | FPID output value in manual mode |
| SPR | REAL | Remote set value |
| SPL | REAL | Local setting |
| X | REAL | Detected value |
| KP | REAL | Proportion, the user caNSet an initial value, such as 6.5 |
| TI | REAL | Integral, iNSeconds, the user caNSet an initial value, such as 60 |
| TD | REAL | Differential, iNSeconds, the user caNSet an initial value, such as 0 |
| HIGH | REAL | Output the upper limit of Yout |
| LOW | REAL | Output the lower limit of Yout |
| INTLCKV | REAL | Mandatory value, valid when INTLCK is TRUE |
| Output | Types of | description |
| OUT | REAL | output value |

### 10.6.10 GET_ERROR ( details of errors obtained in the error directory )

Note : The GET_ERROR instruction is temporarily unavailable.

### 10.6.11 GET_ERROR_CATALOG ( information about the current content obtained in the error directory )

Note : The GET_ERROR_CATALOG instruction is temporarily unavailable.

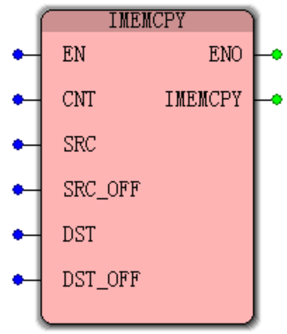
## 10.6.12 GET_SYM ( search for the symbolic name of the PDD variable )

Note : The GET_SYM instruction is temporarily unavailable.

## 10.6.13 HOT_RESTART ( PLC Hot Start )

Note : The HOT_RESTART instruction is temporarily unavailable.

## 10.6.14 IMEMCPY ( data replication )

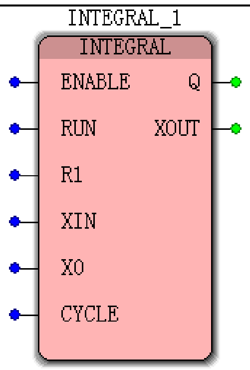
<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: The IMEMCPY instruction is used to copy data from the source data area to the target data area.</td></tr>
<tr><td>LD CNT</td><td rowspan="5">
</td></tr>
<tr><td>IMEMCPY SRC, SRC_OFF, DST, DST_OFF</td></tr>
<tr><td>ST OUT</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>OUT:=IMEMCPY ( CNT, SRC, SRC_OFF, DST, DST_OFF ) ;</td></tr>
<tr><td colspan="2">Note : IL, ST language programming needs to insert variables CNT , SRC, etc. in the current POU variable worksheet or use constants</td></tr>
</table>

➢ **IMEMCPY data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| CNT ( CNT ) | INT | The number of bytes to be copied |
| SRC[0] ( SRC ) | BYTE | The first byte of the source data area , written as SRC[0] when the data type is ARRAY |
| SRC_OFF ( SRC OFF ) | INT | The starting byte number of the source data area, the sequence number of the first byte is 0. |
| DST[0] ( DST ) | BYTE | The first byte of the target data area , written as DST[0] when the data type is ARRAY |
| DST_OFF ( DST OFF ) | INT | The starting byte serial number of the target data |

| | | |
|---|---|---|
| | | area, the sequence number of the first byte is 0. |
| OUT （IMEMCPY） | INT | Error code：<br>0 copied data, no error occurred<br>14 buffers exceed the data segment<br>15 target area is an input group |

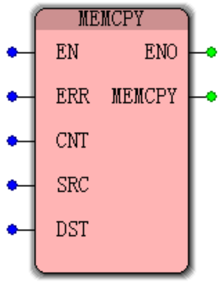
## 10.6.15 INTEGRAL（integration）

<table>
<tr><th>IL programming language</th><th>LD, FBD programming language</th></tr>
<tr><td colspan="2">Function: IN T EG RAL instruction is used to calculate the integral time of data. When using integral instruction, it is necessary to set the task type of POU to cycle scan（CYCLIC）, and set the scan cycle according to your own needs. And the scan cycle can be found in this manual programming model ◊ Hardware ◊ Tasks.</td></tr>
<tr><td>See help file</td><td rowspan="3">
</td></tr>
<tr><td>ST programming language</td></tr>
<tr><td>INTEGRAL_1（ENABLE:=ENABLE, RUN:=RUN, R1:=R1, XIN:=XIN, XO:= X0, CYCLE:=CYCLE）; Q:=INTEGRAL_1.0;XOUT:=INTEGRAL_1.XOUT;</td></tr>
<tr><td colspan="2">Note：IL, ST language programming need to insert variable E, RUN, etc. or use constants in the current POU variable worksheet</td></tr>
</table>

➢ **the INTEGRAL data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| ENABLE | BOOL | Execution function block when TRUE |
| RUN | BOOL | When TRUE, the integratioNStarts, when FALSE, the integration is paused, and the output keeps the last integral value. |
| R1 | BOOL | Reset when TRUE, reset value is XO |
| XIN | REAL | input value |
| XO | REAL | Reset value |
| CYCLE | TIME | Time constant, iNSeconds, this instruction actually integrates XIN CYCLE |
| Q | BOOL | Q is equal to R1 inversion |
| XOUT | REAL | Output value |

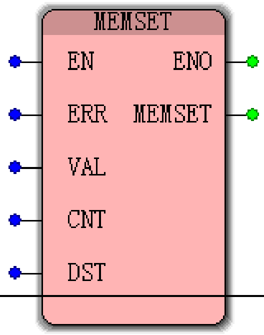
## 10.6.16 MEMCPY ( Data Copy Instruction)

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MEMCPY instruction is used to copy data from the source data area to the target data area. | |
| LD ERR |  |
| MEMCPY CNT, SRC[0], DST[0] | |
| ST OUT | |
| ST programming language | |
| OUT:= MEMCPY ( ERR, CNT, SRC[0], DST[0] ) | |
| Note : IL, ST language programming need to insert variables ERR, CNT , SRC[0], etc. or use constants in the current POU variable worksheet | |

➢**the MEMCPY data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| ERR ( ERR ) | INT | Error code : 0- one correctly copied ; 14- one buffer exceeds data segment ; 15 - target |
| CNT ( CNT ) | INT | The area is an input array. Note : This is the output parameter placed on the left ! |
| SRC[0] ( SRC ) | BYTE | The number of bytes to copy |
| DST[0] ( DST ) | BYTE | The first byte of the source data area , written as SRC[0] when the data type is ARRAY |
| OUT ( IMEMCPY ) | WORD | The first byte of the target data area , written as SRC[0] when the data type is ARRAY |

## 10.6.17 MEMSET ( DatADistribution )

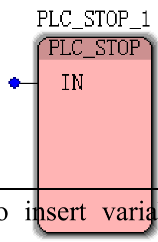
| IL programming language | LD, FBD programming language |
|---|---|
| Function: The MEMSE T instruction is used to distribute source data to the target data area. | |
| LD ERR |  |
| MEMSET VAL, CN 工 DST[0] | |
| ST OUT | |
| ST programming language | |
| OUT:=MEMSET ( ERR, VAL, CNT, | |

| | |
|---|---|
| DST[0] ) | |
| Note : IL, ST language programming needs to insert variables ERR, VAL, CNT, etc. in the current POU variable worksheet or use constants | |

➢ **MEMSET data processing instruction type**

| Input and output | type of data | description |
|---|---|---|
| ERR ( ERR ) | INT | Error code : 0 to 1 correctly copied ; The 14- one buffer exceeds the data segment ; 15 - the target area is an input array. Note : This is the output parameter placed on the left ! |
| VAL ( VAL ) | BYTE | Source data |
| CNT ( CNT ) | DINT | The number of copies to be distributed, one source data can be distributed to N bytes in the target data area , one byte per copy |
| DST[0] ( DST ) | BYTE | The first byte of the target data area , written as SRC[0] when the data type is ARRAY |
| OUT ( IMEMCPY ) | WORD | Output, the duty is not defined, the characters actually being copied to the target data areADST the |

## 10.6.18 PLC_STOP ( PLC stop )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: PLC_STOP instruction for stopping the PLC , cold restart process, initializes all data. If the program has a stack overflow, a string error, or ADivide by 0 , you can call this command to automatically restart the execution of the program. | |
| LD IN |  |
| ST PLC_STOP_1.IN | |
| CAL PLC_STOP_1 | |
| ST programming language | |
| PLC_ STOP_ 1 ( IN ) | |
| Note : IL, ST language programming needs to insert variable IN or use constant in the current POU variable worksheet | |

**P LC_STOP instruction processed data type**

| Input and output | type of data | description |
|---|---|---|
| IN | BOOL | The rising edge is valid, |

| | | |
|---|---|---|
| | | the PLC stops |

## 10.6.19 RD_*_BY_SYM ( Read the value of the PDD variable )

📖·Instruction function overview: RD_*_BY_SYM includes the following instructions for reading variables of different data types of PDD

| | | | |
|---|---|---|---|
| RD_BOOL_BY_SYM | RD_BYTE _BY_SYM | RD_WORD_BY_SYM | RD_DWORD_BY_SYM |
| RD_SINT_BY_SYM | RD_INT_BY_SYM | RD_DINT_BY_SYM | RD_USINT_BY_SYM |
| RD_UINT_BY_SYM | RD_UDINT_BY_SYM | RD_REAL_BY_SYM | RD_STRING_BY_SYM |
| RD_TIME_BY_SYM | RD_INPUT_GROUP | | |

PDD is a method that can access the value of a variable by the name of the variable. It is a method used when the controller kernel layer accesses the value of the PLC variable defined by MULTIPROG . The RD_*_BY_SYM instruction is temporarily unavailable, and the general user directly reads and writes. The variables are fine.

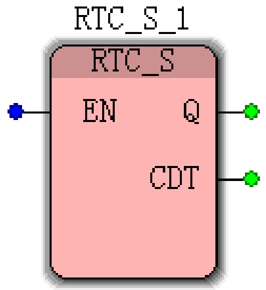
## 10.6.20 WR_*_BY_SYM (write the value of the PDD variable)

📖·Instruction function overview: WR_*_BY_SYM includes the following instructions for writing variables of different data types of PDD

| | | | |
|---|---|---|---|
| WR_BOOL_BY_SYM | WR_BYTE_BY_SYM | WR_WORD_BY_SYM | WR_DWORD_BY_SYM |
| WR_SINT_BY_SYM | WR_INT_BY_SYM | WR_DINT_BY_SYM | WR_USINT_BY_SYM |
| WR_UINT_BY_SYM | WR_UDINT_BY_SYM | WR_REAL_BY_SYM | WR_STRING_BY_SYM |
| WR_TIME_BY_SYM | WR_OUTPUT_GROUP | | |

PDD is a method that can access the value of a variable by the name of the variable. It is a method used when the controller kernel layer accesses the value of the PLC variable defined by MULTIPROG . The WR_*_BY_SYM instruction is temporarily unavailable, and the general user directly reads and writes. The variables are fine.

## 10.6.21 RTC_S ( Read PLC Clock )

| IL programming language | LD, FBD programming language |
|---|---|
| Function: The RTC_S instruction is used to read the PLC clock in a string variable, and the read clock is in GMT format. The format of the oral and time output string specified in IEC61131-3 is : DT#1998-11-21-15:27:56.46 . | |
| LD IN |  |
| ST RTC_S_1.EN | |
| CAL RTC_S_1 | |
| LD RTC_S_1.Q | |
| ST Q | |
| LD RTC_S_1.CDT | |
| ST OUT | |
| ST programming language | |
| RTC_S_1 ( EN:= ( IN )) ; Q:=RTC_S_1.Q; OUT:=RTC_S_1.CDT; | |
| Note : IL, ST language programming needs to insert variables IN, O, OUT or use constants in the variable worksheet of the current POU | |

➢ **RTC_S instruction processing data types**

| Input and output | type of data | description |
|---|---|---|
| IN ( EN ) | BOOL | If TRUE , the actual date and time is written to the linked output string. |
| Q ( Q ) | BOOL | If EN is TRUE, Q is TRUE , otherwise Q is FALSE |
| OUT ( CTD ) | STRING | Date and time of the acquisition, such as DT#2011-08-15-10:08:55.19 |

## 10.6.22 WARM_RESTART ( PLC Warm Start )

Note : The WARM_RESTART instruction is temporarily unavailable.

V A motion controller programming manual

# XI Motion Commands

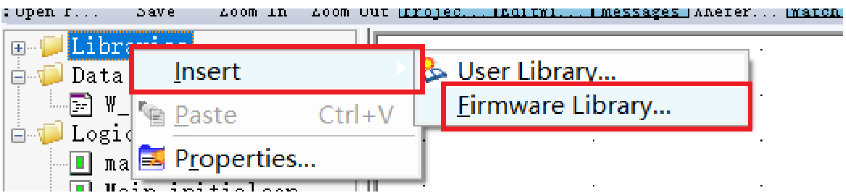
## 11.1 Insert FB_FU_LIB (motion control firmware library)

### 11.1.1 Features

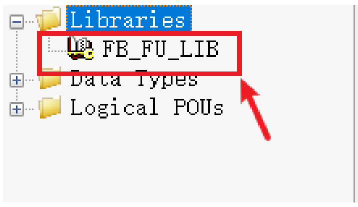
FB_FU_LIB firmware library is provided for motion control and is used as a library function for the user. The user does not need complicated programming, just call and set some simple parameters to use. The motion control library contains a wealth of motion commands such as absolute point for single axis motion, relative point function, electronic cam for multi-axis motion, electronic gear, and overlay function.

### 11.1.2 Adding firmware library

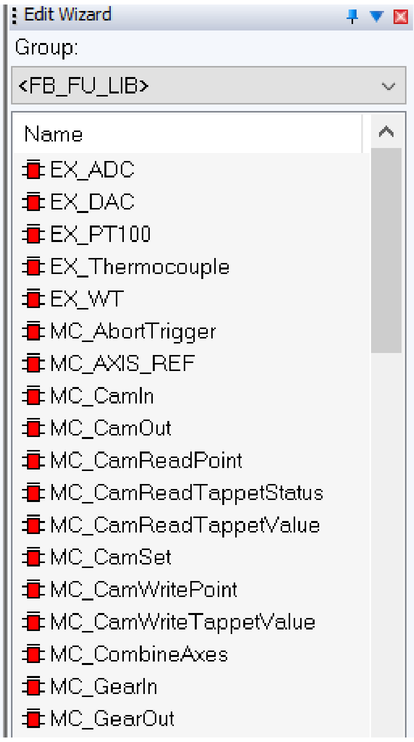
(1) Right click "Library" in the project tree window, select "Insert" and select "Firmware Library" as shown.




(2) Find the location of the stored "FB_FU_LIB" file, find the file, click the "Include" button, and when finished, as shown.

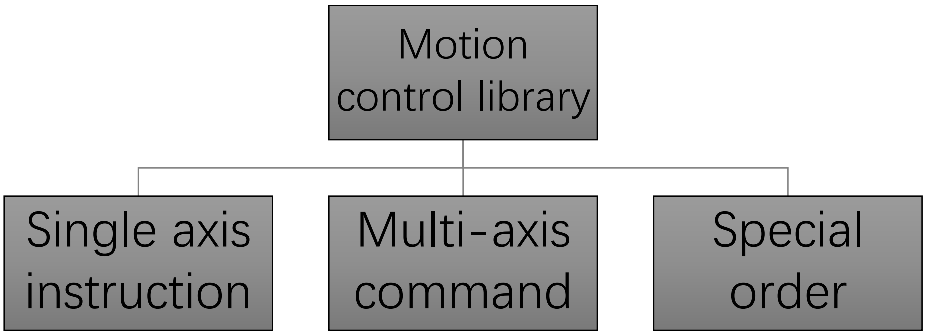



(3) Click "FB_FU_LIB" under the "Edit Wizard" group to view the newly inserted motion control library as shown below.


Edit Wizard
Group:
<FB_FU_LIB>
Name
EX_ADC
EX_DAC
EX_PT100
EX_Thermocouple
EX_WT
MC_AbortTrigger
MC_AXIS_REF
MC_CamIn
MC_CamOut
MC_CamReadPoint
MC_CamReadTappetStatus
MC_CamReadTappetValue
MC_CamSet
MC_CamWritePoint
MC_CamWriteTappetValue
MC_CombineAxes
MC_GearIn
MC_GearOut

# 11.2 motion commands

## 11.2.1 Motion Control Library Classification




## 11.2.2 Movement instruction list

| Instruction set | Instruction code | Features | page number |
|---|---|---|---|
| Uniaxial instruction | MC_Power | Enable command | 11.4.1 MC_Power (ENABLE command) |
| | MC_MoveVelocity | Speed command | 11.4.2 MC_MoveVelocity (speed command) |
| | MC_MoveRelative | Relative displacement instruction | 11.4.3 MC_MoveRelative (relative displacement instruction) |
| | MC_MoveAdditive | Additional displacement instruction | 11.4.4 MC_MoveAdditive (additional displacement instruction) |
| | MC_MoveAbsolute | Absolute displacement instructions | 11.4.5 MC_MoveAbsolute (absolute displacement instructions) |
| | MC_MoveSuperimposed | Additional Motion commands | 11.4.6 MC_MoveSuperimposed (additional displacement instruction) |
| | MC_HaltSuperimposed | Pause additional displacement command | 11.4.7 MC_HaltSuperimposed (Pause additional displacement) |
| | MC_Home | Homing instruction | 11.4.8 MC_Home (zero return instruction) |

| | | | |
|---|---|---|---|
| | MC_SetOverride | Speed overshoot command | 11.4.9 MC_SetOverride (overshoot speed command) |
| | MC_Stop | Stop command | 11.4.10 MC_Stop (stop command) |
| | MC_Halt | Pause command | 11.4.11 MC_Halt (pause command) |
| | MC_SpecialMoveAbsolute | Special instructions absolute displacement | 11.4.11 MC_SpecialMoveAbsolute (special absolute displacement instructions) |
| | MC_ReadActualPosition | Read live position command | 11.4.12 MC_ReadActualPosition (real position instruction read) |
| | MC_ReadActualVelocity | Real-time speed reading instruction | 11.4.13 MC_ReadActualVelocity (read real-time speed) |
| | MC_ReadMotionState | Read axis motion command | 11.4.14 MC_ReadMotionState (read axis motion command) |
| | MC_ReadStatus | Read status command axis | 11.4.15 MC_ReadStatus (Read axis state) |
| | MC_SetPosition | Position setting command | 11.4.16 MC_SetPosition (position setting instruction) |
| | MC_Phasing | Spindle command offset | 11.4.17 MC_Phasing (shift spindle command) |
| | MC_TouchProbe | Position capture command | 11.4.18 MC_TouchProbe (position capture command) |
| | MC_AbortTrigger | Position capture interrupt instruction | 11.4.19 MC_AbortTrigger (position capture interrupt instruction) |
| | NS_MC_Jog | Jog command | 11.4.20 NS_MC_Jog (jog command) |
| | NS_MC_StopByPos | Mode Specifies the phase stop command | 11.4.21 NS_MC_StopByPos (position designated mode stop command) |

| | | | |
|---|---|---|---|
| | NS_MC_ReadParameter | A read parameter command | 11.4.22 NS_MC_ReadParameter (read command parameter) |
| Multiaxial instruction | MC_GearIn | Electronic gear coupling instruction | 11.5.1 MC_GearIn (electronic gear coupling instructions) |
| | MC_GearOut | Instruction from the electronic gear | 11.5.2 MC_GearOut (electronic gear disengaged instruction) |
| | MC_CombineAxes | Combined dual-spindle gear command | 11.5.3 MC_CombineAxes (double spindle gears combined instruction) |
| | NS_MC_RotaryCutIn | Peeling instructions | 11.5.9 NS_MC_RotaryCutIn (peeling instruction) |
| | NS_MC_SpecialCamIn | Special instructions cam | 11.5.10 NS_MC_SpecialCamin (special cam instruction) |
| | NS_MC_SpecialCombineAxes | Special double joint gear shaft instruction | 11.5.11 NS_MC_SpecialCombineAxes (special double joint spindle gear command) |
| | MC_CamIn | Electronic cam associated instruction | 11.5.12 MC_CamIn (electronic cam associated instruction) |
| | MC_CamOut | Instruction from the electronic cam | 11.5.13 MC_CamOut (electronic cam departing instruction) |
| | MC_CamWritePoint | The cam point information write command | 11.5.14 MC_CamWritePoint (cam point information write command) |
| | MC_CamReadPoint | The cam point information read command | 11.5.15 MC_CamReadPoint (cam point information reading instruction) |
| | MC_CamSet | Change the | 11.5.16 MC_CamSet |

| | | | |
|---|---|---|---|
| | | entry into force of the cam point instruction | (changes to take effect cam point instructions) |
| | MC_ReadTappetStatus | A plurality of read status command tappet points | 11.5.17 MC_ReadTappetStatus (read status command plurality of lifters points) |
| | MC_ReadTappetValue | A plurality of read instructions tappet point information | 11.5.18 MC_ReadTappetValue (single read command tappet point information) |
| | MC_WriteTappetValue | Edit instruction information tappet point | 11.5.19 MC_WriteTappetValue (edit point information tappet instruction) |
| Special instructions | NS_CC_ADC | AD instruction | 11.6.1 NS_CC_ADC (AD instruction) |
| | NS_CC_DAC | DA nstruction | 11.6.2 NS_CC_DAC (DA instruction) |
| | EX_ADC | Extended instruction AD | 11.6.3 EX_ADC (AD extended instruction) |
| | EX_DAC | DA extended instruction | 11.6.4 EX_DAC (DA expansion module) |
| | NS_CC_NOoutput | Output instruction is prohibited QXX | 11.6.5 NS_CC_NOoutput (prohibition command output QXX) |
| | NS_CC_Counter | High-speed count instruction | 11.6.6 NS_CC_Counter (High-Speed Counter) |
| | NS_CC_CNTI | High-speed counting interrupt instruction | 11.6.7 NS_CC_CNTI (high-speed counter interrupt instruction) |
| | NS_CC_CNT_Out | Interval comparison output instruction | 11.6.8 NS_CC_CNT_Out (comparison output instruction section) |
| | NS_CC_DI_Counter | High-speed count | 11.6.9 NS_CC_DI_Counter (DI- |

| | | | |
|---|---|---|---|
| | | instruction DI | speed count instruction) |
| | NS_CC_EXTI | DI interrupt instruction | 11.6.10 NS_CC_EXTI (DI interrupt instruction) |
| | NS_CC_ReadPulseVelocity | Read pulse rate command | 11.6.11 NS_CC_ReadPulseVelocity (read-axis pulse rate controlled) |
| | RTC_S | Clock special register | 11.6.12 RTC_S (special register clock) |
| G commands | NC_GroupEnable | Group enable command shaft | 11.7.1 NC_GroupEnable (ENABLE command axis group) |
| | NC_MoveLiner | Linear interpolation command | 11.7.2 NC_MoveLiner (linear interpolation) |
| | NC_MoveCircula | Circular interpolation command | 11.7.3 NC_MoveCircula (circular interpolation) |
| | NC_CartesianCoordinate | Cartesian coordinate system command | 11.7.4 NC_CartesianCoordinate (Cartesian robot command) |
| CANopen code instructions | NS_CC_CANopen_NMT_Read | Network status read command | 3.1.1 NS_CC_CANopen_NMT_Read (network status read instruction) |
| | NS_CC_CANopen_NMT_Write | Network status write instruction | 3.1.2 NS_CC_CANopen_NMT_Write (network state write command) |
| | NS_CC_CANopen_PDO_Comm | PDO process data communication configuration parameters | 3.1.3 NS_CC_CANopen_PDO_Comm (PDO process data communication configuration parameters) |
| | NS_CC_CANopen_PDO_Map | PDO process data mapping configuration parameters | 3.1.4 NS_CC_CANopen_PDO_Map (PDO process data mapping configuration parameters) |
| | NS_CC_CANopen_RPDO | PDO data mapping area | 3.1.5 NS_CC_CANopen_RPDO |

| | | | |
|---|---|---|---|
| | | read command | (PDO data mapping area read command) |
| | NS_CC_CANopen_TPDO | PDO data mapping area assignment instruction | 3.1.6 NS_CC_CANopen_TPDO (PDO data mapping area assignment instruction) |
| | NS_CC_CANopen_SDO_Read | Service data reading instruction | 3.1.7 NS_CC_CANopen_SDO_Read (service data reading instruction) |
| | NS_CC_CANopen_SDO_Write | Service Data assignment instruction | 3.1.8 NS_CC_CANopen_SDO_Write (service data assignment instruction) |

# 11.3 Basics of motion control instructions

## 11.3.1 Command modes of motion controller

**Digital pulse:** This method is similar to the control method of the stepping motor. The motion controller sends the pulse signal of the pulse/direction or CW/CCW type to the servo driver. Our company only supports the AB pulse; the servo driver works in the position control mode. The position loop is completed by the servo drive. Japanese servos and domestic servo products mostly use this mode. The advantage is that the system debugging is simple and not easy to cause interference, but the disadvantage is that the servo system responds slightly slower.

**Analog signals:** In this way, the motion controller sends a ±10V analog voltage command to the servo driver, and receives position feedback signals from position detectors such as motor encoders or linear encoders; the servo driver operates in speed control mode, and the position closed loop is controlled by motion. The device is completed. Most of the servo products in Europe and America use this mode of operation. The advantage is that the servo response is fast, but the disadvantage is that it is sensitive to on-site interference, and the debugging is slightly complicated.

**CANopen communication:** This method is to control the operation of the servo driver through the communication protocol. See Appendix IV for details.

**The following describes the general debugging steps for the motion controller to control the servo axis with analog signals:**

(1) initialization parameters

After confirming that the servo driver wiring is correct, first initialize the parameters of the servo drive (restore the factory settings). After the servo drive completes the factory setting: set the control mode; set the enable by external control; the gear ratio of the encoder signal output; set the proportional relationship between the control signal and the motor speed (the analog output voltage corresponds to the servo shaft speed).

(2) Wiring

Connect the signal line between the motion controller and the servo. The following wiring is required: the analog output line of the motion controller, the servo enable control signal line, and the encoder signal line of the servo output. (For details, please refer to 11.4.2 Wiring method)

(3) Test direction

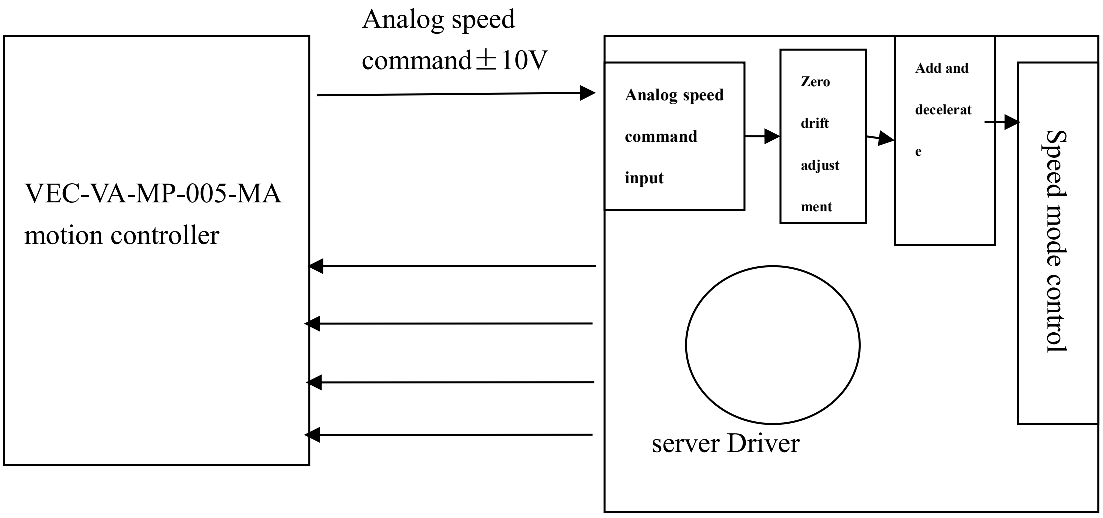
For a closed-loop control system, if the direction of the feedback signal is incorrect, the consequences must be catastrophic. The servo driver's enable signal is controlled by the motion controller programming output Q0. At this point the servo axis should rotate at a lower speed, which is called "zero drift". Execute the motion controller command module (DA module). Use this command module to see if the motor speed and direction can be controlled by this command (parameter) and monitor whether the encoder feedback signal is consistent (ie, ensure that the encoder is federically incremented when the analog voltage is given as a positive voltage). When the analog voltage is given to a negative voltage, the encoder feedback is decremented). If it cannot be controlled, or the encoder feedback is incorrect, check the parameter settings of the analog wiring, encoder feedback line and control mode.

(4) inhibition zero drift

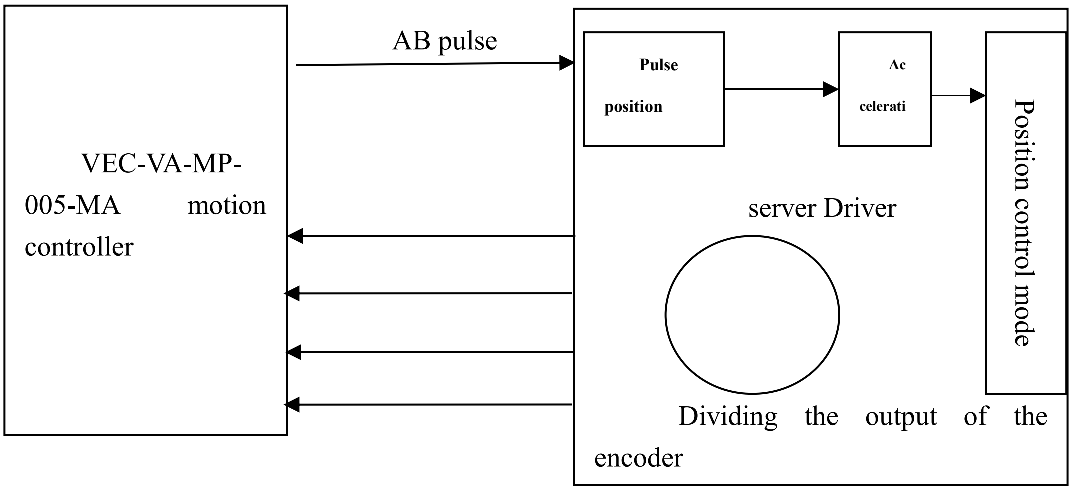
In the closed-loop control process, the existence of zero drift will have a certain influence on the control effect, and it is best to suppress it. With the command module on the motion controller or adjusting the zero drift value on the servo to suppress the zero drift phenomenon, it can be realized (refer to the analog zero drift adjustment), so that the motor speed approaches zero. Since the zero drift itself also has a certain randomness, it is not necessary to require the motor speed to be absolutely zero.

## 11.3.2 movement control.

VEC-VA-MP-005-MA analog motion controller controlling the servo drive ± 10V operating velocity mode, the servo drive encoder signal divided output fed back to the motion controller, the position loop is completed by a motion controller, as shown below shown;

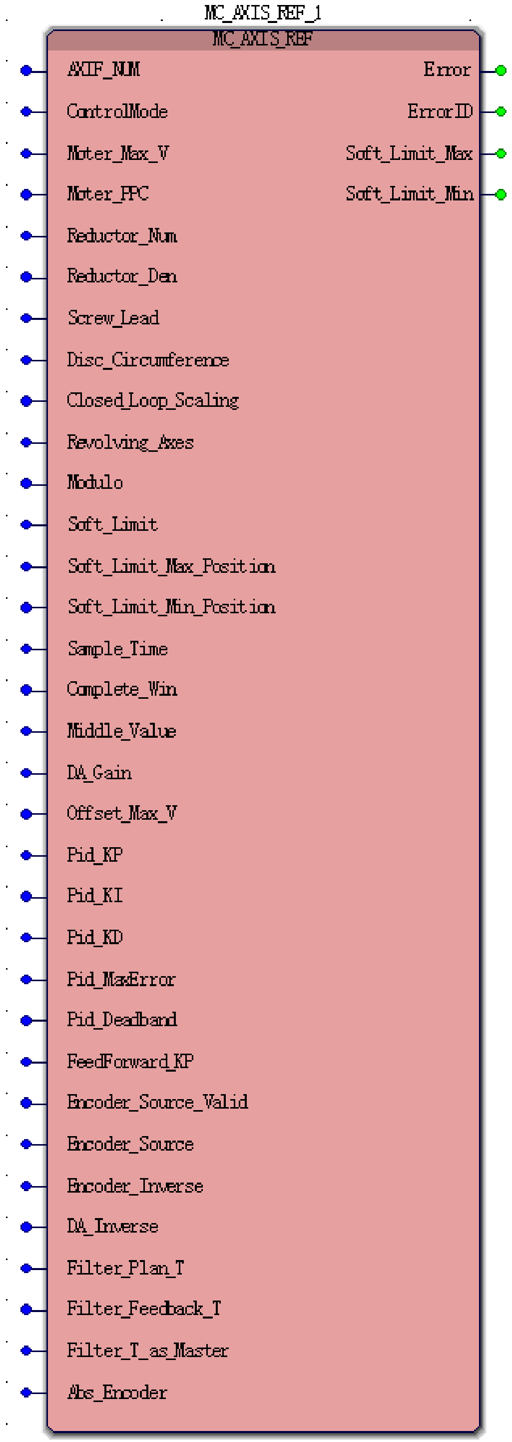



The VEC-VA-MP-005-MA motion controller uses the pulse amount control servo driver to operate in the position mode. The servo driver divides the output encoder signal and feeds it back to the motion controller. The position closed loop is completed by the servo driver, as shown in the figure below.



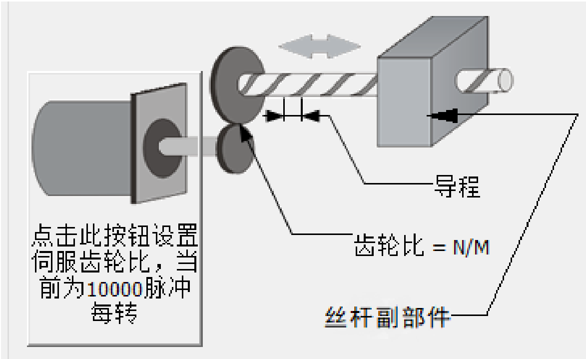
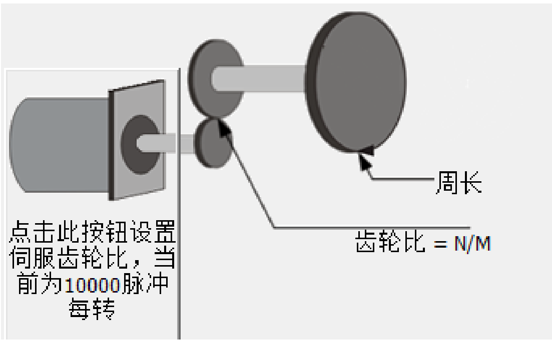
## 11.3.3 MC_AXIS_REF (axis parameter setting)

| FB / FU | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to configure the parameters of the controlled servo axis AXISXXX | VEC-VA-MP-005-MA |



➢ **Input parameters**

| name | Features | type of data | Range setting (default value) |
|---|---|---|---|
| Axis_Num (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) |
| Axis number description: Axis_Num is the axis number of the controlled axis. Under the control of analog or pulse mode, the axis numbers 0~4 are the real axes, and the 5~11 is the virtual axis. Compared with the real axis, the virtual axis has no actual control effect; In CANopen mode, the axis numbers 0~15 can be used as real or virtual axes. The real axis of the network is configured, and the virtual axis is not configured. In the template given by CANopen, the node number of the control = axis number + 1. | | | |
| The ControlMode (control mode) | Select the output mode of the motion control commands 0: analog control 1: Pulse Control 2: CANopen control | INT | 0: analog control 1: Pulse Control 2: CANOPEN control (0) |
| | | | |
| Motor_Max_V (maximum speed) | Servo drive allows maximum motor speed Unit: r / min | DINT | A positive number |
| The motion controller outputs ±10V analog, it corresponds to the maximum forward and reverse speed of the servo drive; that is, the servo speed corresponding to each volt. For example, if the servo driver analog gain is set to 300 (r/min)/v, the parameter is 10*300=3000 r/min. | | | |
| Motor_PPC (Pulses per revolution) | Pulses per revolution of the motor | DINT | A positive number |
| The number of pulses of the divided servo output per revolution of the servo drive is multiplied by 4 times, not necessarily the resolution of the motor encoder *4 times the frequency (Some servos can support encoder crossover output settings) | | | |
| Reductor_Num (Molecular reduction gear ratio) | Reduction ratio of the motor shaft to the execution terminal | LREAL | A positive number |
| + | | | |
| Reductor_Den (Reduction gear | Deceleration ratio denominator from motor | LREAL | A positive number |

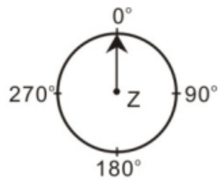
<table>
<tr><td>denominator)</td><td>shaft to actuator</td><td></td><td></td></tr>
<tr><td colspan="4"></td></tr>
<tr><td>Screw_Lead<br>(Screw lead)</td><td>Lead lead, the distance traveled by the lead screw<br>Unit: per unit</td><td>LREAL</td><td>A positive number</td></tr>
<tr><td colspan="4">When Disc_Circumference is 0, the default is the transmission terminal screw.<br>
<br>Set the lead of the terminal drive screw. When the reduction ratio numerator/reduction ratio denominator = 1/2, it indicates the distance that the servo motor rotates two turns of the screw sub-assembly.</td></tr>
<tr><td>Disc_Circumference<br>(Disc perimeter)</td><td>Disk perimeter terminal<br>Unit: unit</td><td>LREAL</td><td>A positive number</td></tr>
<tr><td colspan="4">When Disc_Circumference not zero, Screw_Lead invalid, a disk drive default terminal<br>
<br>Set the circumference of the terminal transmission disc. When the deceleration ratio numerator/reduction ratio denominator = 1/2, it indicates that the arc length of the servo motor rotating two turns is its circumference.</td></tr>
<tr><td>Closed_Loop_Scaling<br>(Double-loop coefficient)</td><td>Double closed loop coefficient, number of motor pulses per meter of terminal mechanism / number of pulses per meter of external encoder, if there is no external encoder, this parameter is set to 1.0.</td><td>LREAL</td><td>A positive number</td></tr>
<tr><td colspan="4"></td></tr>
</table>

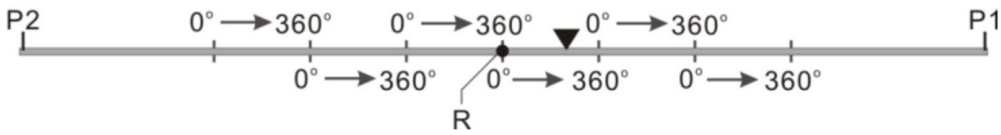
<table>
<tr><td>Revolving_Axes<br>(Controlled axis type)</td><td>Setting the controlled axis type<br>0: Linear axis<br>1: rotation shaft</td><td>BOOL</td><td>0: Linear axis<br>1: rotation shaft</td></tr>
<tr><td colspan="4">

**Linear axis:**


Linear axis model Note:<br>P1: positive limit<br>P2: Anti limit<br>Origin: Origin

**The rotary shaft:**




Notes axis of rotation model:<br>P1: positive limit<br>P2: Anti limit

**Linear and rotary axes' differences:**

The difference between the linear axis and the rotary axis is mainly that the rotary axis is cycled by the die. If the position of the linear axis terminal actuator is 500, the position of the corresponding rotary axis is 140 (when the mode is 360), and the calculation method is the remainder obtained by dividing 500 by the die.

</td></tr>
<tr><td>Modulo<br>(mold)</td><td>Mode setting<br>rotational shaft<br>Unit: unit</td><td>LREAL</td><td>A positive number</td></tr>
<tr><td colspan="4">The actual position of the terminal to bisect actuator cycle</td></tr>
<tr><td>Soft_Limit<br>(Soft limit)</td><td>When Soft_Limit = 1,<br>turn soft limit function</td><td>BOOL</td><td>TRUE or FALSE</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>Soft_Limit_Max_Position</td><td>The maximum value</td><td>LREAL</td><td>Positive,</td></tr>
</table>

| | | | |
|---|---|---|---|
| (Soft limit maximum value) | of the software limit. If the maximum position exceeds this value, Soft_Limit_Max outputs TRUE. | | negative, zero (0) |
| | | | |
| Soft_Limit_Min_Position (Soft limit the minimum value) | The soft limit is the minimum value. If the minimum position is lower than this value, Soft_Limit_Min outputs TRUE. | LREAL | Positive, negative, zero (0) |
| | | | |
| Sample_Time (sampling time) | Setting the sampling pulse encoder feedback time (unit: ms) | WORD | A positive number |
| Set time sampling pulse encoder feedback. When Abs_Encoder = 2, Sample_Time must range [1,3] | | | |
| Complete_Win (Points to complete the window) | Retention | DINT | Retention |
| | | | |
| Middle_Value (Analog value zero drift) | Analog value zero drift | DINT | Positive, negative, 0 (0) |
| When the analog zero drift value is 0, the analog output is 0V. When ControlMode=0 is selected, this value needs to be the same as the value of DAC_Value in 11.3.5 Analog Zero Drift Adjustment. | | | |
| DA_Gain (Analog gain) | Analog gain setting | DINT | Positive, negative, 0 (0) |
| Set the analog gain, which is normally set to zero. The analog offset is adjusted to adjust the linearity of the output analog to ensure that the analog output of the controller is consistent with the servo analog gain to improve the control accuracy. (1+DA_Gain/10000)* Original analog output. | | | |
| Offset_Max_V (Maximum compensation rate) | The maximum compensation rate Unit: r / min | DINT | A positive number, 0 (0) |
| The maximum compensation speed of the servo motor, when there is an error in the analog closed-loop control, the servo motor will get a compensation speed to achieve precise control of the position. The maximum value of this compensation speed is Offset_Max_V. | | | |
| Pid_KP | Proportional gain | DINT | A positive number, 0 (0) |

| | | | |
|---|---|---|---|
| This parameter is valid when the control mode is selected as analog control, ie ControlMode=0. The proportional gain is similar to the position loop proportional gain of the servo drive. When the position closed loop control is completed on the motion controller, increasing the value shortens the positioning time; when the value is 0, the controller will not adjust the position closed loop; This value can be increased when the motor is not shaken) | | | |
| Pid_KI | Integral gain | DINT | A positive number, 0 (0) |
| This parameter is valid when the control mode is selected as analog control, ie ControlMode=0. The integral gain is similar to the integral gain of the position loop of the servo. When the value is appropriately increased, the cumulative error can be reduced. When the value is set too large, the motor will be shaken. | | | |
| Pid_KD | Differential gain | DINT | A positive number, 0 (0) |
| This parameter is valid when the control mode is selected as analog control, ie ControlMode=0. The integral gain is similar to the differential gain of the position loop of the servo. Generally, the differential is not enabled. | | | |
| Pid_MaxError | PID maximum error | DINT | A positive number, 0 (0) |
| When the number of error pulses exceeds this value, the integral gain is useless and is generally set to zero. | | | |
| Pid_Deadband | PID dead zone | DINT | A positive number, 0 (0) |
| PID error deadband follower means (the number of error pulses) is within this value, no adjustment of the PID. | | | |
| FeedForward_KP | Feedforward gain | DINT | A positive number, 0 (0) |
| Traditional P control requires a tracking error (setpoint - actual value) causes this error profile shaft, poor dynamic response, increased during execution of the contour, feed forward gain to appropriately increase the following error can be reduced during operation. | | | |
| Encoder_Source_Valid (Source encoder significant bit) | Source encoder valid bit | BOOL | TRUE / FALSE |
| When the value is TRUE, set Encorder_Source port select shaft encoder signals as source port. When this is FALSE, the function block parameter select shaft axis Axis_Num port set as an encoder signal source port. | | | |
| Encoder_Source (Source encoder) | Setting encoder signal source | WORD | 0-4 |
| With Encorder_Source_Valid, setting the encoder signal source. | | | |
| Encoder_Inverse (Reverse significant bit encoder) | Reverse significant bit encoder | BOOL | TRUE / FALSE |
| When the value is TRUE, the shaft opening position of the received pulses counted negated | | | |

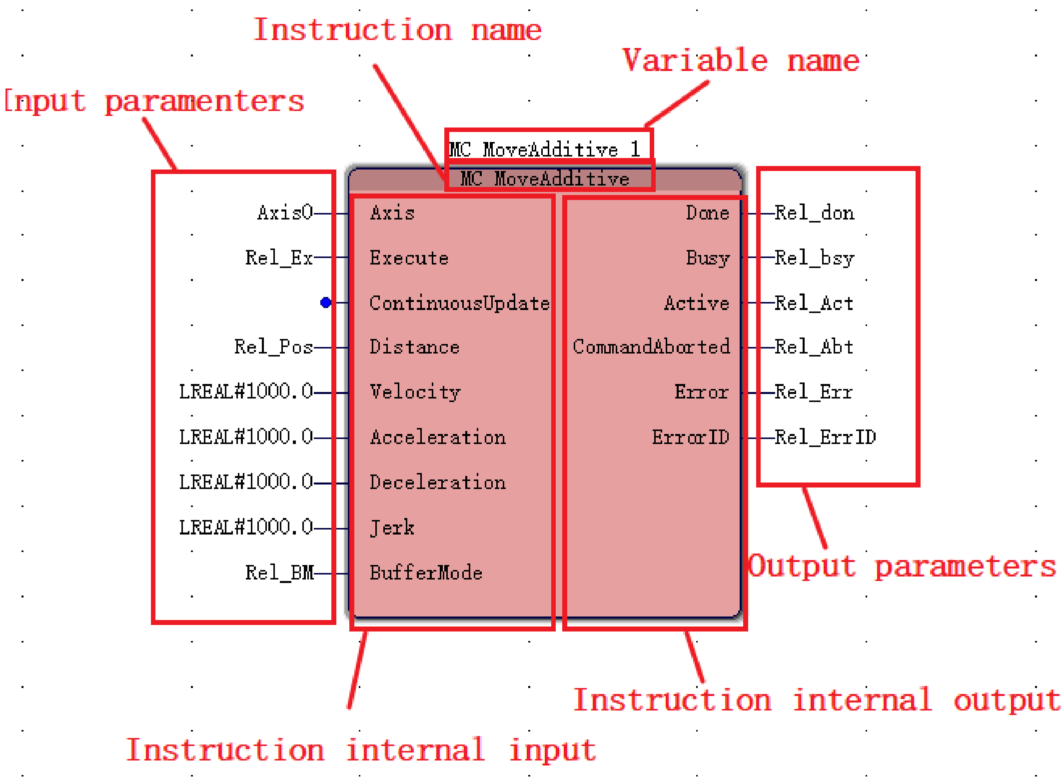
<table>
<tr><td>DA_Inverse<br>(Analog inversion)</td><td>DA reverse</td><td>BOOL</td><td>TRUE / FALSE</td></tr>
<tr><td colspan="4">DA_Inverse has two modes 0 and 1<br>1) When DA_Inverse=0, the controller controls the servo motor counterclockwise, that is, the analog output is positive voltage;<br>2) When DA_Inverse=1, the controller controls the servo motor clockwise, that is, the analog output is a negative voltage;<br>Special Note: The two modes of DA_Inverse need to be matched with the encoder direction, otherwise the closed loop control cannot be formed.<br>For example, in the example of the following instruction, when MC_Power is executed, if the servo axis can be positioned, the value does not need to be more<br>If the motor is running at a set compensation speed Offset_Max_V, change the value from the original 0 to 1 or the original 1 to 0, or change the encoder A/B line to any one. By Encoder_Inverse<br>Or the register MB3.9654 is set to modify the encoder direction; the special register address (%MB3.9654) corresponds to the modified axis AXISXX (XX represents 0~4) as follows:
<table>
<tr><th>Special register address</th><th>Numerical (binary)</th><th>Modified shaft</th></tr>
<tr><td>% MB3.9654</td><td>00000001 (decimal 1)</td><td>AXIS0</td></tr>
<tr><td>% MB3.9654</td><td>0000 0010 (decimal 2)</td><td>AXIS1</td></tr>
<tr><td>% MB3.9654</td><td>0000 0100 (decimal 4)</td><td>AXIS2</td></tr>
<tr><td>% MB3.9654</td><td>00001000 (decimal 8)</td><td>AXIS3</td></tr>
<tr><td>% MB3.9654</td><td>0001 0000 (decimal 16)</td><td>AXIS4</td></tr>
</table>
For example: modify the axis AXIS0 feedback encoder counting direction; just fill in the special register %MB3.9654, you can change from the original increment to the decrement, or from decrement to increment. If you need to modify multiple axes, write 1 to the corresponding bit.</td></tr>
<tr><td>Filter_Plan_T<br>(Given filtering)</td><td>For a given position<br>and a given speed filtering</td><td>DINT</td><td></td></tr>
<tr><td colspan="4">Filter_Plan_T units of underlying period, a period of 2ms. Enable change invalid.</td></tr>
<tr><td>Filter_Feedback_T<br>(Feedback filter)</td><td>Speed feedback filter</td><td>DINT</td><td></td></tr>
<tr><td colspan="4">Filter_Feedback_T units of underlying period, a period of 2ms. Enable change invalid.</td></tr>
<tr><td>Filter_T_as_Master</td><td>Spindle speed filtering</td><td>DINT</td><td></td></tr>
<tr><td colspan="4">Expressed as provided Filter_T_as_Master spindle, it outputs it to the filtering speed of the shaft from the real axis.</td></tr>
<tr><td>Abs_Encoder</td><td>Setting an absolute<br>encoder</td><td>USINT</td><td>0-2<br>(0)</td></tr>
<tr><td colspan="4">Set the absolute encoder type, only the spindle port 4 can be connected to the absolute encoder<br>0: not enabled<br>1: Enable 23-bit absolute encoder<br>2: Enable 24 is an absolute encoder (this mode is only supported when the absolute encoder function is enabled, and the 24-bit encoder must be a Nikon encoder)</td></tr>
</table>

**Module Description:**

- When using the servo axis to control the servo motor, the AXIS_REF module of the controlled servo axis must be correctly configured according to the mechanical parameters. Otherwise, the servo axis cannot correctly control the servo motor operation;
- In the following single-axis instruction and multi-axis instruction program examples, the AXIS_REF module needs to be called and the relevant information is correctly configured (the following program demonstration will not be repeated);
- Use up to one module per axis for the entire project.

## 11.3.4 sports instruction constitutes

**Motion command configuration shown in FIG.**



## 11.3.5 Analog offset adjustment

**Zero drift Definition:**Zero drift analog amplifier means when the input signal is zero, the output is not zero is called zero drift phenomenon. That is: When the input of the amplifier short circuit, at the output there is an irregular phenomenon generated voltage changes slowly.

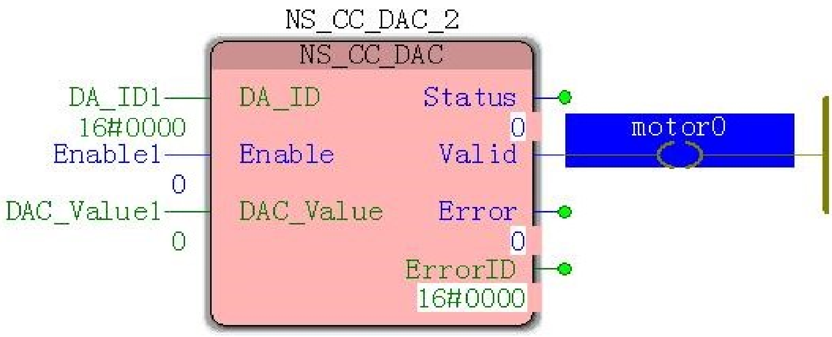
**The motion controller instruction module zero drift adjustment step; (here, respectively to servo drive Vector VB and VC will be described servo drive)**

➢ **Vector VB servo drives:**

(1) Reference11.6.2 NS_CC_DAC (DA instruction)Instructions and digital to analog conversion module DA relation, after the completion of programming.

(2) in the Edit Wizard, the callout "NS_CC_DAC" module, where 0 is the adjusting shaft (Axis0) zero drift, DA_ID initial value fill other axis "0" and so on (0-3), can control multi-axis simultaneously tune a plurality of "NS_CC_DAC" module.

(3) Fill in the data input module type variable, the variable name can be named their own, but to ensure that the user name is not duplicated without having to complete the actual physical address, the software automatically assigned an address. Be downloaded after clicking "Create" no error when finished.




(4) the online debugging mode, when the Enable becomes FALSE TRUE by the (already in ensuring the servo enabled state), if the motor is running at the speed of zero drift, DAC_Value adjustment value at this time to ensure that the servo drive in a stationary state, Enable the TRUE to FALSE, offset adjustment is completed, and then completing the axis parameters as the initial value of the value DAC_Value MC_AXIS_REF (axis parameter) of the module Middle_Value.

➢ **Vector VC servo drives:**

Steps (1), (2), and (3) are the same as the VB servo driver. In step (4), in the online debugging mode, when Enable is changed from FALSE to TRUE (ensure that the servo is already enabled), if the motor is at zero speed During operation, the servo P06.68 (AI1 zero drift mV) / P06.73 (AI2 zero drift mV) / P06.78 (AI3 zero drift mV) is adjusted through the BOP panel to ensure that the servo drive is at rest, zero drift The adjustment is complete. When filling the axis parameter, use 0 as the initial value of the MC_AXIS_REF (axis parameter) module Middle_Value.

**note:**

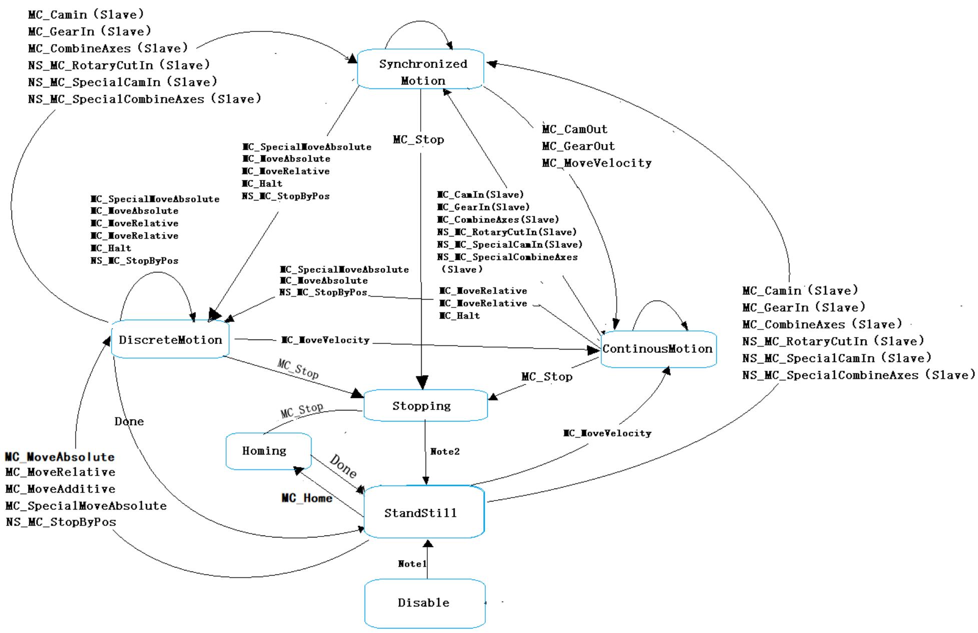
1) After the zero drift adjustment is completed, it is determined that the Enable state on the

module is False, and the state of the parameter Enable is not allowed to be "TRUE" during the execution of the program by the "NS_CC_DAC module" and the "MC_Power module", otherwise the function module is run. The motor will run on a meal.

2) The examples described below will all be the default example after the zero drift has been adjusted. The following program demonstration will not repeat the description.

## 11.3.6 state machine

When the VEC-VA-MP-005-MA motion controller controls each axis using motion control commands, each axis has an internal operating state. The state switching of the controlled axes follows the state machine shown in the figure below. The state machine defines the motion commands that can be executed in each state and the state after the motion command is executed. When the motion command is used by the user, the state machine can determine whether a motion command can be used in the current state. The state machine of the VEC-VA-MP-005-MA motion controller is shown in the figure below, and the arrow indicates that the part is the state of the axis.




Note1: Enable with MC_Power command and MC_Power. Status is True.

Note2: MC_Stop.Done is True and MC_Stop.Execute is False.

| No. | Axis Status | Explanation |
|---|---|---|
| 1 | StandStill | Ready to execute state |
| 2 | Disable | The state is not performed |
| 3 | Stopping | Stop state |
| 4 | Homing | Homing state |
| 5 | Discrete Motion | Discrete motion |
| 6 | Continuous Motion | Continuous motion |
| 7 | Synchronized Motion | Synchronous Movement |

The state of the shaft can be determined based MC_ReadStatus (read status command axis) output pin, refer to the specific use instructions "11.4.15 MC_ReadStatus (Read axis state). "

## 11.3.7 BufferMode Features

For the same axis, when there is a motion command control axis during the motion, other motion commands can be started. When the two motion commands are handed over, there are two options for the handover mode. The handover mode can be based on the BufferMode of the latter motion command. Pin parameter settings to choose from. The meaning of the BufferMode related terms is as follows:

1. Current command: motion command of the current control axis
2. Handover instructions: instructions waiting to be executed
3. Handover speed: the speed at which the current command switches to the handover command
4. Target speed: Velocity pin parameters in the instruction
5. Target position: Position or Distance pin parameters in the displacement related instruction.

- **Two kinds of transfer mode**

| Transfer mode | Action Description |
|---|---|
| 0: mcAborting (interrupt) | Immediate action to interrupt the current instruction execution and delivery instructions |
| 1: mcBuffered (wait) | Wait for the current instruction execution after the normal action, and execute the handover command immediate action |

- **note**

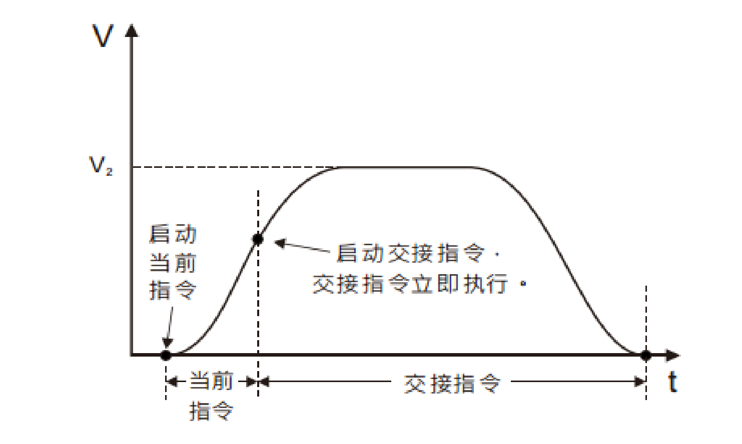
The same axis only supports the first-level BufferMode buffer: For the motion instruction with BufferMode, if the motion instruction 2 with BufferMode=1 is used to transfer the motion instruction 1, and the motion instruction 2 has not been executed, the motion instruction of BufferMode=1 is executed at this time. Invalid and error, but does not affect the execution of instruction 1 and instruction 2.

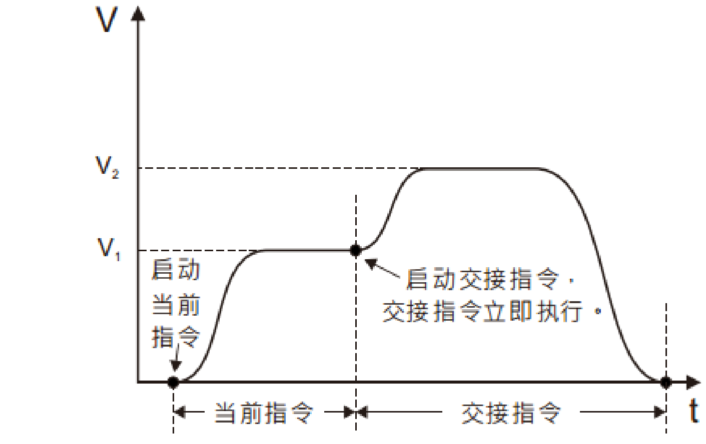
**Program example**:

Brief description of BufferMode with two relative displacement instructions.The first relative displacement command speed is v1, the displacement is S1, the second relative displacement command speed is v2, and the displacement is S2. Changing the BufferMode of the second displacement instruction makes the two instructions have different handover procedures, as explained below:

■ When BufferMode=0, the following four situations are explained:

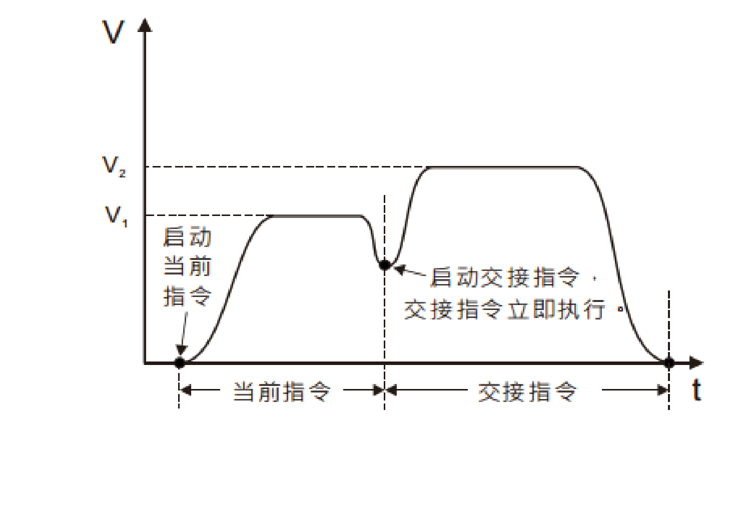
| 1, when the current interrupt command acceleration | 2, the current instruction uniform stage interrupte transport |
|---|---|

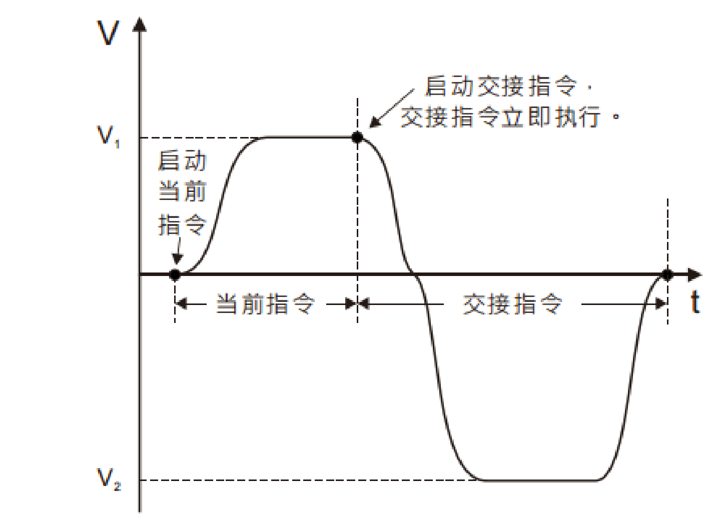



**3, the deceleration phase of the current instruction interrupt**

**4, the instruction transfer direction opposite to th current instruction**

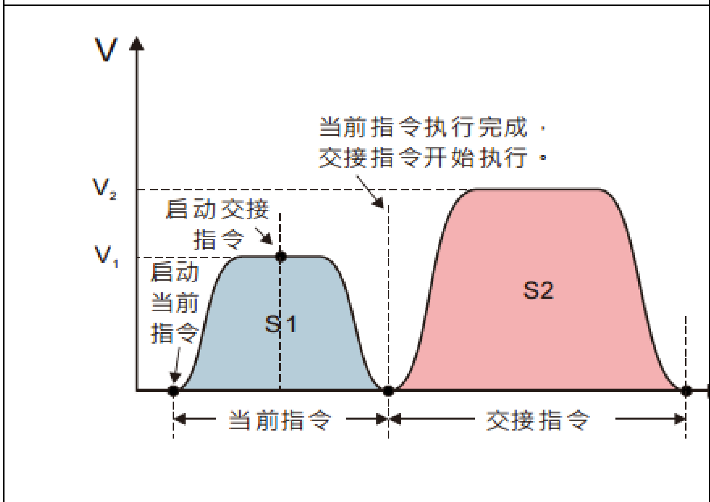


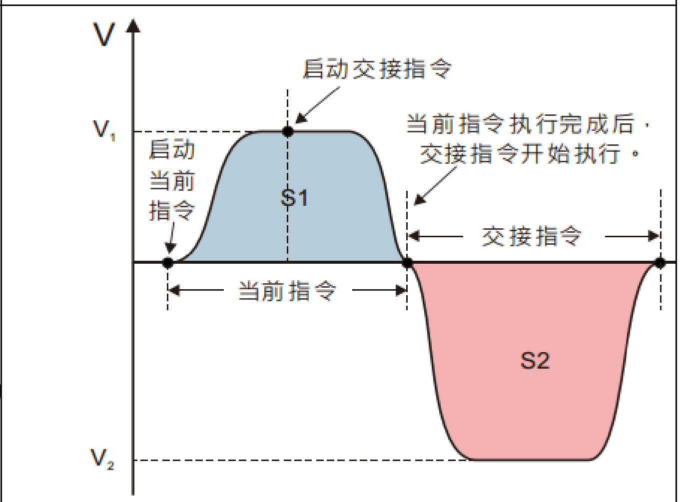

**Note: When the controlled axis current instruction and the transfer instruction transfer acceleration / deceleration handover command acceleration / deceleration**

■ When BufferMode = 1, the following will be described for two cases:

**1, the transfer instruction to the current instruction in the same direction**




**2, the instruction transfer direction opposite to the current instruction**



## 11.4 Uniaxial Instruction

### Precautions:

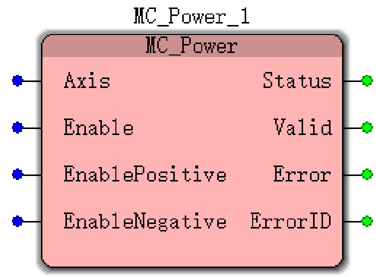
- For non-moving command, MC_ReadActualVelocity, MC_ReadActualPosition, MC_SetOverride, MC_ReadMotionState, MC_ReadStatus, NS_MC_ReadParameter, MC_SetPosition, can be used in any state of the shaft.
- Each instruction in the same number of works on the same axis using the following:

| | |
|---|---|
| MC_AXIS_REF | Up to 1 module per axis for the entire project |
| MC_Power | Up to 1 module per axis for the entire project |
| MC_CamIn | Up to 3 module per axis for the entire project |
| MC_CamOut | Up to 3 module per axis for the entire project |
| MC_CombineAxes | Up to 3 module per axis for the entire project |
| MC_GearIn | Up to 3 module per axis for the entire project |
| MC_GearOut | Up to 3 module per axis for the entire project |
| MC_Halt | Up to 3 module per axis for the entire project |
| MC_Home | Up to 3 module per axis for the entire project |
| MC_MoveAbsolute | Up to 3 module per axis for the entire project |
| MC_MoveAdditive | Up to 3 module per axis for the entire project |
| MC_MoveRelative | Up to 3 module per axis for the entire project |
| MC_MoveVelocity | Up to 3 module per axis for the entire project |
| MC_Stop | Up to 3 module per axis for the entire project |
| NS_MC_StopByPos | Up to 3 module per axis for the entire project |
| MC_SpecialMoveAbsolute | Up to 3 module per axis for the entire project |
| NS_MC_RotaryCutIn | Up to 1 module per axis for the entire project |
| NS_MC_SpecialCamIn | Up to 1 module per axis for the entire project |
| NS_MC_SpecialCombineAxes | Up to 3 module per axis for the entire project |
| MC_HaltSuperimposed | Up to 1 module per axis for the entire project |
| MC_MoveSuperimposed | Up to 1 module per axis for the entire project |
| MC_Phasing | Up to 3 module per axis for the entire project |
| NS_MC_Jog | Up to 1 module per axis for the entire project |
| MC_SetOverride | Up to 1 module per axis for the entire project |
| MC_SetPosition | Up to 1 module per axis for the entire project |
| MC_TouchProbe | Up to 1 module per axis for the entire project |
| MC_AbortTrigger | Up to 1 module per axis for the entire project |
| NS_MC_CamReadPoint | Up to 1 module per axis for the entire project |
| NS_MC_CamReadTappetStatus | Up to 3 module per axis for the entire project |
| NS_MC_CamReadTappetValue | Up to 1 module per axis for the entire project |

| | |
|---|---|
| NS_MC_CamSet | Up to 1 module per axis for the entire project |
| NS_MC_CamWritePoint | Up to 1 module per axis for the entire project |
| NS_MC_CamWriteTappetValue | Up to 1 module per axis for the entire project |
| MC_ReadActualPosition | Any number |
| MC_ReadActualVelocity | Any number |
| MC_ReadMotionState | Any number |
| MC_ReadStatus | Any number |
| NS_MC_ReadParameter | Any number |

## 11.4.1 MC_Power (ENABLE command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to enable the respective servo axis or enable release | VEC-VA-MP-005-MA |




➢ **Input parameters**

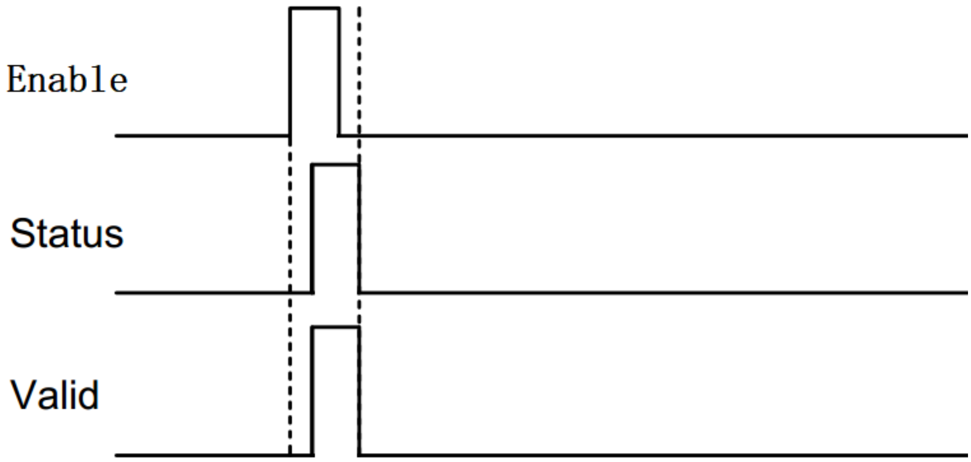
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| The Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 -11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable is TRUE |
| Enable (execute bit) | When Enable is True, the instruction is executed. | BOOL | TRUE or FALSE (FALSE) | Enable is TRUE |
| EnablePositive (forward rotation) | Retention | BOOL | Retention | Retention |
| EnableNegative (allowing inversion) | Retention | BOOL | Retention | Retention |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Status | This parameter indicates when the instruction is TRUE control shaft | BOOL | TRUE or FALSE |
| Valid | The output parameter | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| | represents the effective output command is TRUE | | |
| Error | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| ErrorID | Instruction execution error code error | WORD | - |

- **FIG timing variation output parameter**

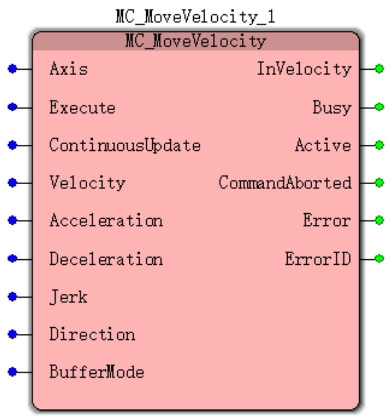



- **Function Description**

- When the Enable FALSE to TRUE, a delay period, Status, Valid simultaneously TRUE;
- When the Enable TRUE to FALSE, a delay period, Status, Valid while is FALSE;
- This instruction is for causing the controlled release servo axis or enabled;
- When the analog or pulsed mode, it just the motion controller corresponding to the control servo axes enabled, not servo drive enable itself can, servo drive enable it needs to be set depending on the servo manufacturer; down CANopen mode directly enabled servo itself;
- Only one the MC_Power a shaft (Enable command)
- Uniaxial and multiaxial instruction before executing the instruction, the instruction must be executed MC_Power executed or not executed in reverse order
  When the motion control function will not be executed.

## 11.4.2 MC_MoveVelocity (speed command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to set the control shaft in accordance with the deceleration to the movement at a uniform speed and the set speed | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 -11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Execute from FALSE to TRUE |
| Execute (execution position) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | - |
| ContinuousUpdata | Retention | - | - | - |
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | Positive (non-default) | Execute from FALSE to TRUE |
| Accleration (Acceleration) | The set target acceleration (unit: unit / S2) | LREAL | Positive (non-default) | Execute from FALSE to TRUE |
| Decleration (decrease speed) | Set target deceleration (unit: unit / S2) | LREAL | Positive (non-default) | Execute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| Jerk (The rate of change of acceleration) | The rate of change of the set target acceleration / deceleration (Unit: unit / S3) | LREAL | Positive (non-default) | Execute from FALSE to TRUE |
| Direction (direction) | Set the operation direction<br>1: positive direction<br>3: Negative direction<br>4: Continuation of the current direction | INT | 1:positive direction<br>3:Negative direction<br>4: Continuation of the current direction<br>(Non-default) | Execute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions<br>0: immediately interrupted<br>1: Wait | INT | 0: immediately interrupted<br>1: Wait<br>(0) | Execute from FALSE to TRUE |

**Description:**

1. This instruction starts when Execute changes from FALSE to TRUE. This instruction is being executed. When Execute is changed from TRUE to FALSE, there is no effect on the execution of this instruction.

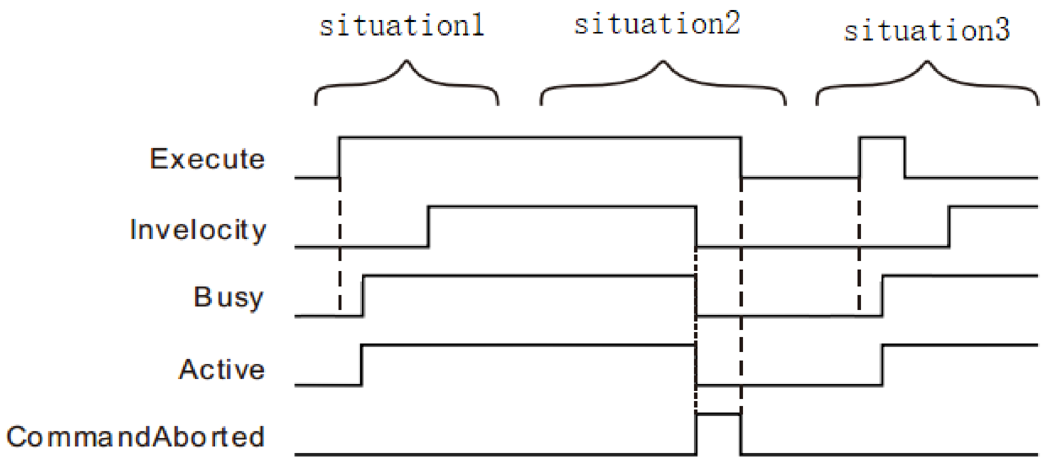
2. When the instruction is being executed and Execute is changed from FALSE to TRUE again, the instruction can be re-executed. The pin parameters that can be re-validated include Velocity, Acceleration, Deceleration, Jerk, Direction, and BufferMode.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| InVelocity (arrival rate) | This parameter represents the speed output reaches to TRUE | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| | output instruction is TRUE | | |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




Case 1: When the Execute FALSE to TRUE, after a period, Busy, Active becomes TRUE. When the speed of arrival, Invelocity becomes TRUE, while still Busy and Active remains TRUE state.

Case 2: When Execut is TRUE, the instruction is interrupted when the other instruction, CommandAborted becomes TRUE, while Invelocity, Busy and Active becomes FALSE, when a TRUE to FALSE Execute, CommandAborted becomes FALSE.

Case 3: In the course of instruction execution, when the Execute TRUE to FALSE, after reaching the speed, InVelocity becomes TRUE, the Busy remains to TRUE and the Active state.

➢ **Function Description**

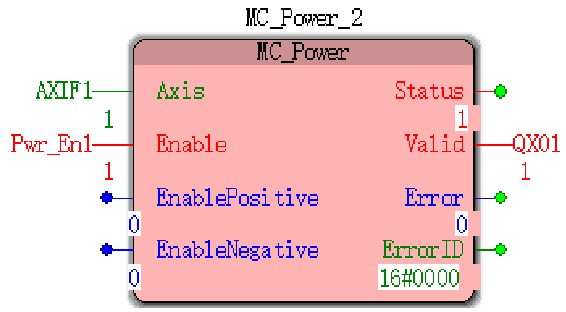
- The instruction to execute upon the Execute FALSE to TRUE. If the instruction is no transfer of command, regardless of whether the instruction is executed, the Execute again when the FALSE to TRUE, the command can be executed again, this time to re-pin the parameters in force include Velocity, Acceleration, Deceleration, Jerk, Direction, BufferMode ;
- When you modify Velocity speed value of the controlled axes, you need to re-trigger Execute, speed can be changed;
  - When the instruction is executed after the completion, i.e. the Invelocity FALSE to TRUE, even by changing the target speed command MC_SetOverride, Invelocity this time remains to TRUE. When MC_MoveVelocity not completed, i.e. InVelocity to FALSE, by changing the target speed command MC_SetOverride Upon reaching the new target speed, only the InVelocity FALSE to TRUE.

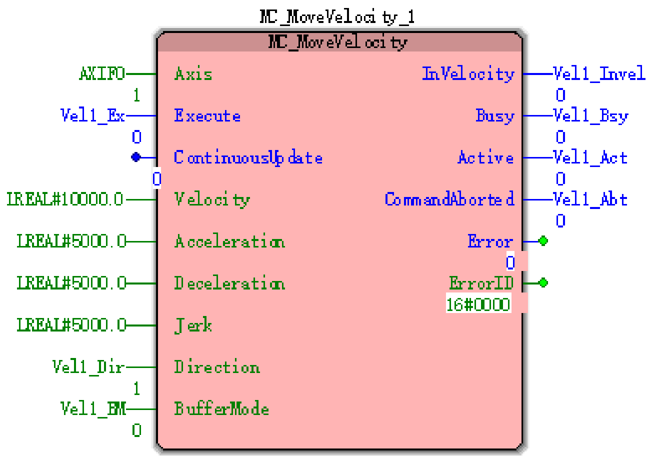
## Examples of a program

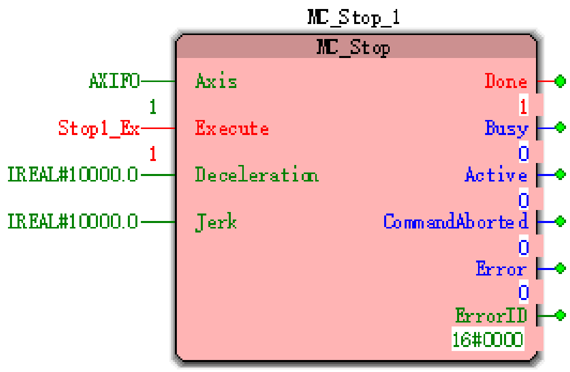
In the example below MC_MoveVelocity instruction execution time of the individual.

**1、Variables and procedures**

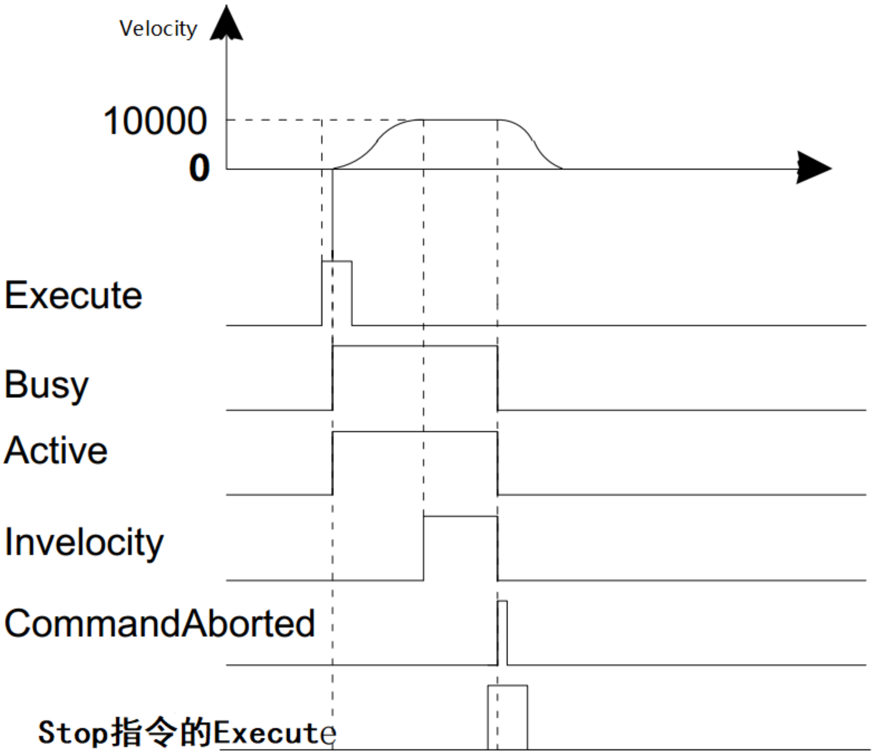
| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity1 | MC_MoveVelocity | |
| AXIF0 | USINT | 1 |
| Vel1_Ex | BOOL | FALSE |
| Vel1_Dir | INT | 1 |
| Vel1_BM | INT | 0 |
| Vel1_Invel | BOOL | |
| Vel1_Bsy | BOOL | |
| Vel1_Act | BOOL | |
| Vel_Abt | BOOL | |
| MC_Stop1 | MC_Stop | |
| Stop_Ex | BOOL | FALSE |





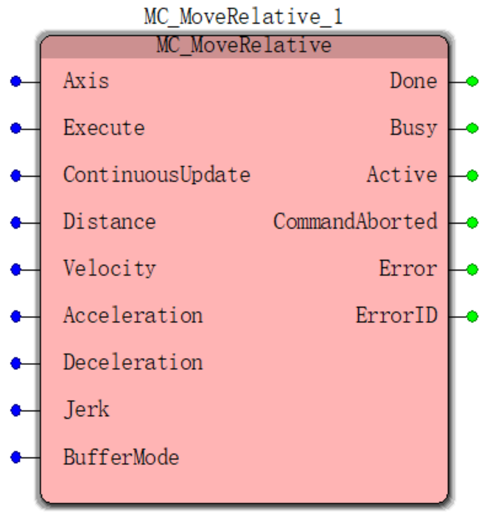

**2、Timing and motion profiles of FIG.**




- When Vel1_Ex a FALSE to TRUE, Vel1_Bsy, Vel_Act simultaneously become TRUE, starts instruction execution speed; when the speed reaches, Vel1_Invel becomes TRUE, and at the same time Vel1_Bsy Vel1_Act remains to TRUE.
- When Stop_Ex1 a FALSE to TRUE, after a period, CommandAborted a FALSE to TRUE (in this case, if Vel1_Ex FALSE, then after a period CommandAborted becomes FALSE), while, Busy and Active becomes FALSE.

## 11.4.3 MC_MoveRelative (relative displacement instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control axis current position as a starting point, according to the set speed, acceleration and deceleration, rate of change of acceleration of the moving distance setting | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute | When the Exexcute FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE (FALSE) | - |
| ContinuousUpdata | Retention | - | - | - |
| Distance (distance) | Goal Setting distance (Unit: unit) | LREAL | Positive, negative, zero (0) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0: immediately interrupted 1: Wait | INT | 0: immediately interrupted 1: Wait (0) | Exexcute from FALSE to TRUE |

**Description:**

1, the instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed when the Execute TRUE to FALSE, no effect on the implementation of the directive.

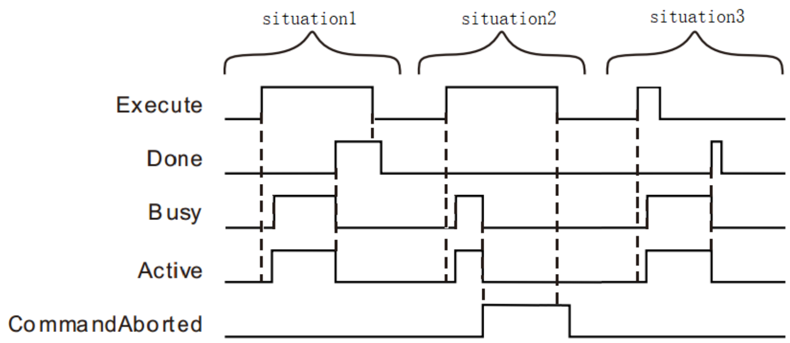
2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be re-executed, the parameters can be revalidated The pin comprises a Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of | BOOL | TRUE or |

| | | | |
|---|---|---|---|
| | the faulting instruction when the output instruction is TRUE | | FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, Busy and Active simultaneously become TRUE. When the positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, if after the completion of the positioning, Execute a TRUE to FALSE, after a period, Done becomes FALSE.

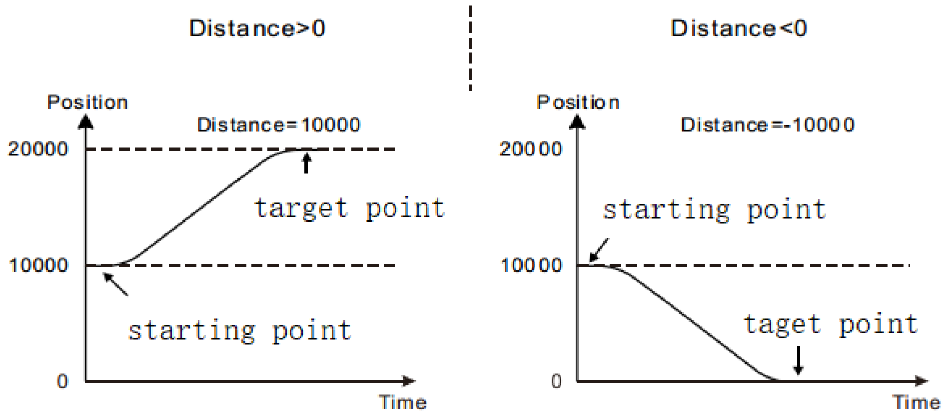
**Case 2:**When the Execute is TRUE, the instruction is interrupted other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

**Case 3:**During instruction execution, after the Execute TRUE to FALSE, when the instructions are executed, the Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, the Done becomes FALSE.

➢ **Function Description**

- This instruction is used to set speed control shaft, deceleration and acceleration rate of change moving distance setting, with reference to the starting point of the distance of the axis position when the instruction to start execution.
- Distance starting point with the reference axis together determine a target position under control of the instruction, i.e. the reference target position start position = + Distance.
- 0 Distance When completed, the movement target position for the current position of the axis, i.e., instruction execution in the next cycle is started, the Done becomes TRUE.

As shown below, the reference position of the starting point is 10,000, when Distance> 0 (10,000), the axis of the forward movement, the target position is 20,000 (10,000 + 10,000), in the lower left diagram; when Distance <0 (-10000) , the reverse shaft, the target position is 0 (10000-10000), as the lower right in FIG.

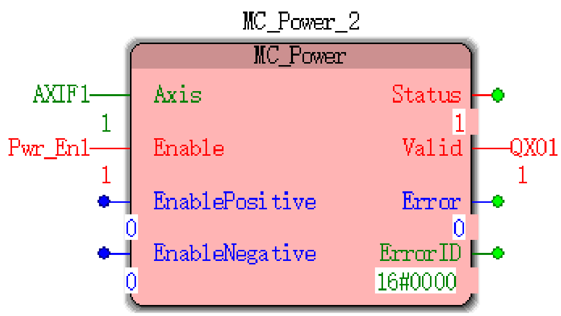


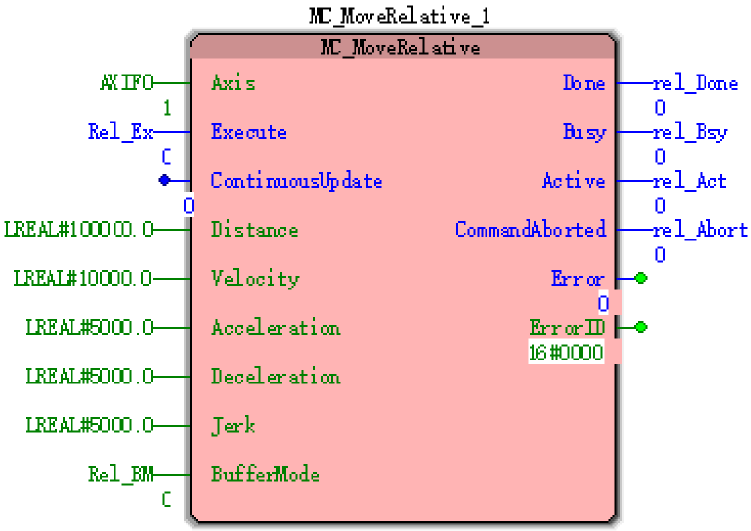

**Examples of a program**

In the example below MC_MoveRelative instruction execution time of the individual.

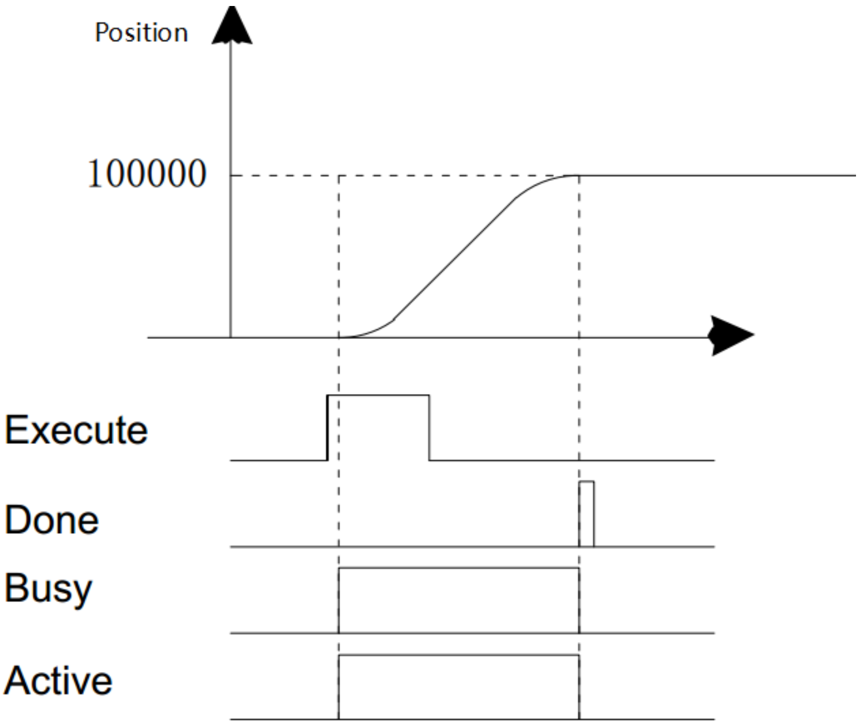
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveRelativey1 | MC_MoveRelative | |
| AXIF0 | USINT | 1 |
| Rel_Ex | BOOL | FALSE |
| Rel_BM | BOOL | 0 |
| Rel_Done | BOOL | |
| Rel_Bsy | BOOL | |
| Rel_Act | BOOL | |
| Rel_Abort | BOOL | |






**2, Motion curve and timing diagram**




- When the Rel_Ex FALSE to TRUE, a period, while the Busy and Active is TRUE, starts executing the instruction in accordance with the relative displacement of the parameter setting, the current position of the axis at this time is 0, the target location is 100,000.
- When the shaft reaches the position 100 000, the instruction execution is completed, the output Done to TRUE.

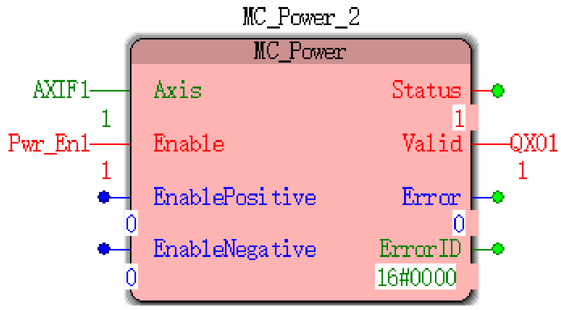
**Program Example Two**

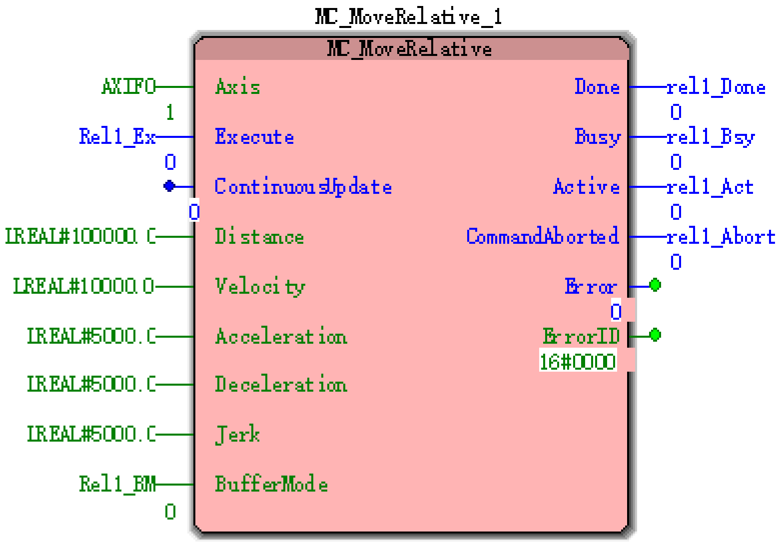
MC_MoveRelative interrupted MC_MoveRelative example is shown below.

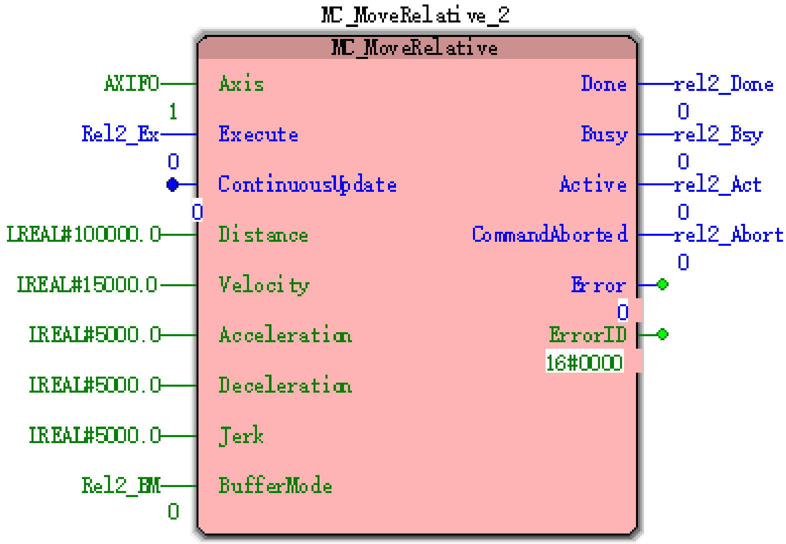
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|

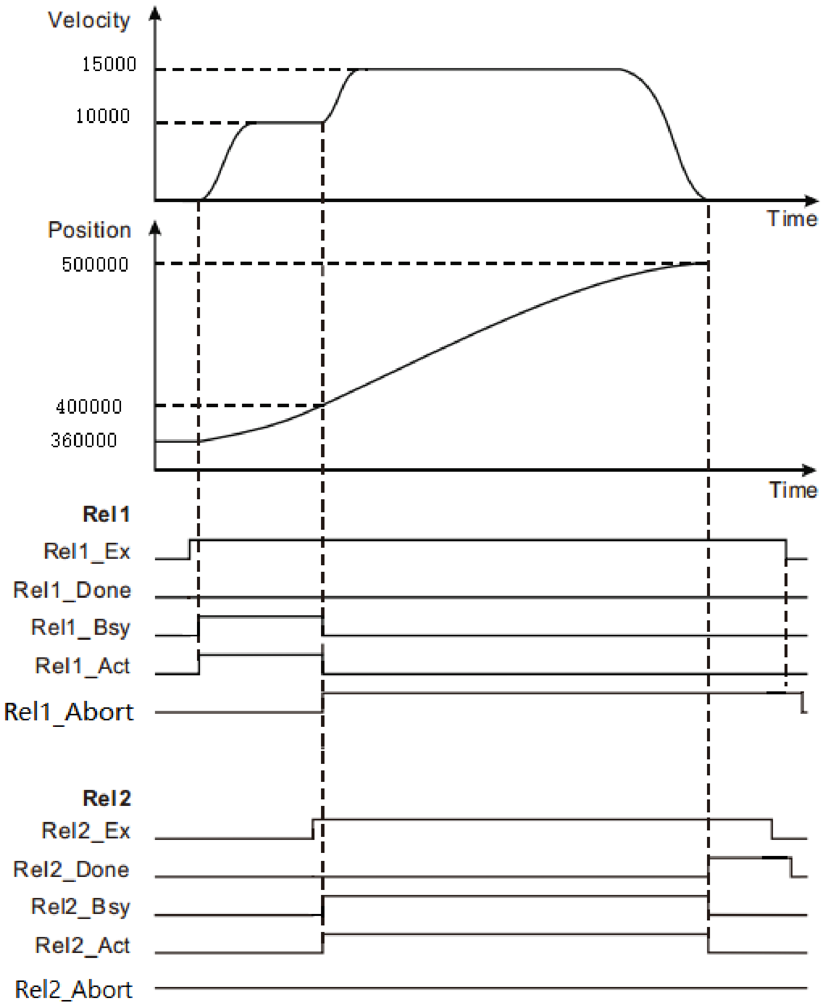
| | | |
|---|---|---|
| MC_MoveRelativey1 | MC_MoveRelative | |
| AXIF0 | USINT | 1 |
| Rel_Ex | BOOL | FALSE |
| Rel_BM | BOOL | 0 |
| Rel_Done | BOOL | |
| Rel_Bsy | BOOL | |
| Rel_Act | BOOL | |
| Rel_Abort | BOOL | |
| MC_MoveRelativey2 | MC_MoveRelative | |
| AXIF0 | USINT | 1 |
| Rel2_Ex | BOOL | FALSE |
| Rel2_BM | BOOL | 0 |
| Rel2_Done | BOOL | |
| Rel2_Bsy | BOOL | |
| Rel2_Act | BOOL | |
| Rel2_Abort | BOOL | |






**2, Motion curve and timing diagram**



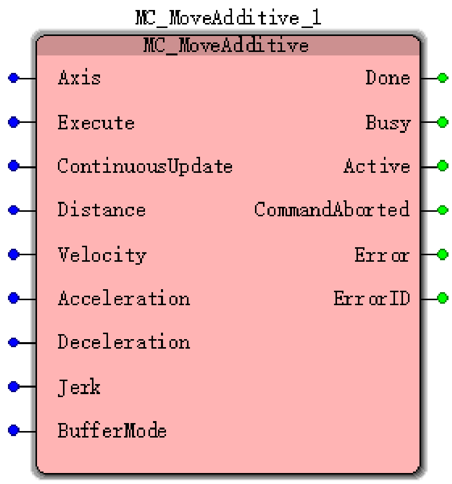

- When Rel1_Ex a FALSE to TRUE, the first MC_MoveRelative instruction starts execution, the current position of the axis at this time is 360 000, the position of the target

(46,000 + 10,000 = 36,000).

- When the shaft position is reached 40000, Rel2_Ex a FALSE to TRUE, MC_MoveRelative second instruction begins execution, and the execution of the first instruction MC_MoveRelative is interrupted, the output parameter Rel1_Abt becomes TRUE.
- Position when the shaft reaches 50,000 (50,000 + 10,000 = 40,000), the second MC_MoveRelative execution is completed, the output parameter Rel2_Done becomes TRUE.

## 11.4.4 MC_MoveAdditive (additional displacement instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction from the control shaft in accordance with the set speed, acceleration and deceleration moved a additional | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute | When the Exexcute FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE (FALSE) | - |
| Continuous Updata | Retention | - | - | - |
| Distance (distance) | Goal Setting distance | LREAL | Positive, negative, zero | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | (Unit: unit) | | (0) | |
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0: immediately interrupted 1: Wait | INT | 0: immediately interrupted 1: Wait (0) | Exexcute from FALSE to TRUE |

**Description:**

1, the instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed when the Execute TRUE to FALSE, no effect on the implementation of the directive.

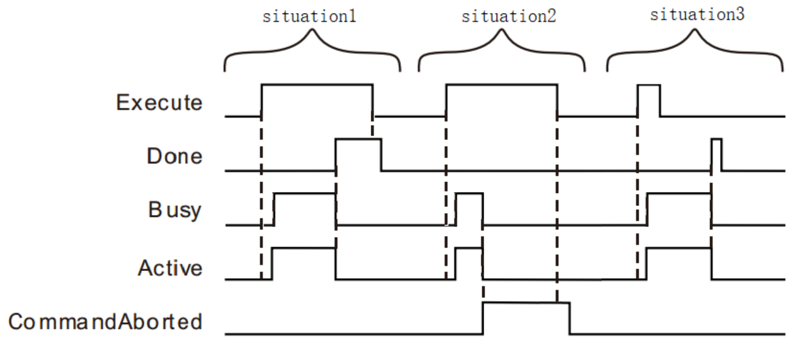
2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be re-executed, the parameters can be revalidated The pin comprises a Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode.

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;
When the positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

**Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

**Case 3**: After during instruction execution, Execute a TRUE to FALSE, when the instructions are executed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

➢ **Function Description**

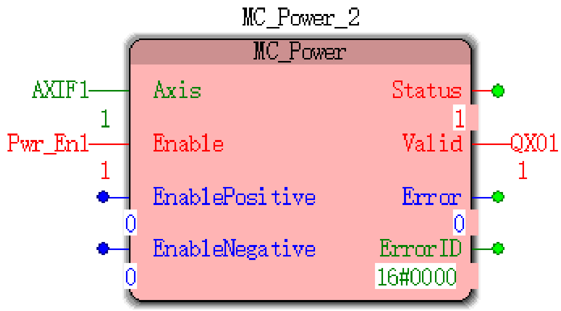
- This instruction means for controlling the terminal to perform the set rate of acceleration of the mobile some additional distance.
- The current command is a command related to the displacement, if not complete, then the instruction is executed in the mobile terminal from the remaining actuator to a command from the front and from the sum of this instruction set, when this instruction is complete, the terminal actuator after the current command is a speed command, this command will terminate the instruction execution speed, when executed, according to a set speed, acceleration and deceleration of the moving distance setting; the final position of the former and the sum of a travel command from the instruction set stop
- When this instruction is executed alone, the effect same as the MC_MoveRelative

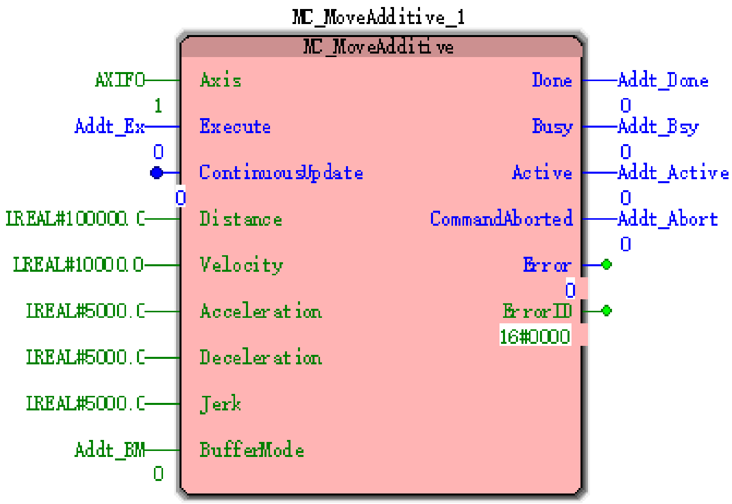
**Examples of a program**

Examples when a separate instruction is executed as follows MC_MoveAdditive

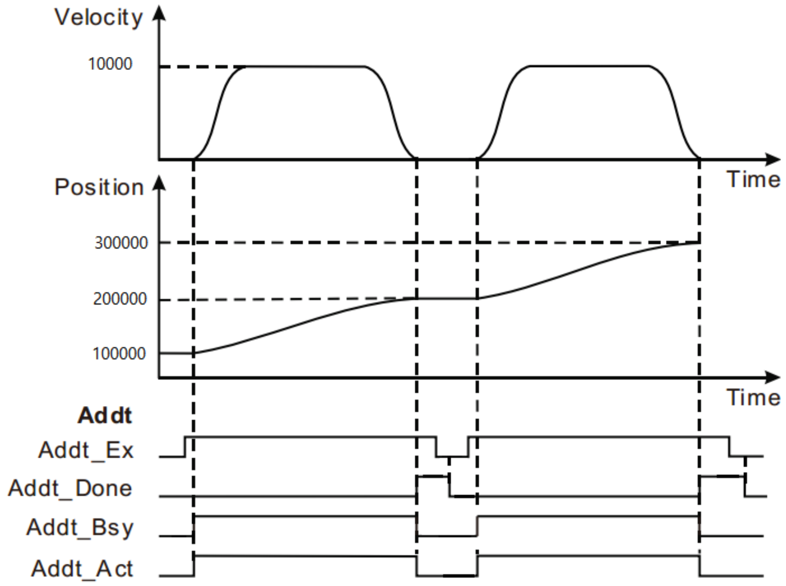
**1、Variables and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveAdditive1 | MC_MoveAdditive | |
| AXIF0 | USINT | 1 |
| Addt_Ex | BOOL | FALSE |
| Addt_BM | INT | 0 |
| Addt_Done | BOOL | |
| Addt_Bsy | BOOL | |
| Addt_Active | BOOL | |
| Addt_Abort | BOOL | |




**2、Timing and motion profiles of FIG.**




- When Addt_Ex a FALSE to TRUE, the motion controller controls the operation of the servo motor to the current position as a reference point, after a period Addt_Bsy, Addt_Act becomes TRUE. After the completion of the servo motor set distance, a done bit Addt_Done FALSE to TRUE, and at the same time Addt_Bsy Addt_Act a TRUE to FALSE.
- When Addt_Ex a TRUE to FALSE, a bit Addt_Done reset cycle is complete.
- After completion of the servo motor set distance, Addt_Ex again by the FALSE to TRUE, the motion controller controls the operation of the servo motor, the servo motor after completion of the set distance, again by the complete bit Addt_Done FALSE to TRUE.

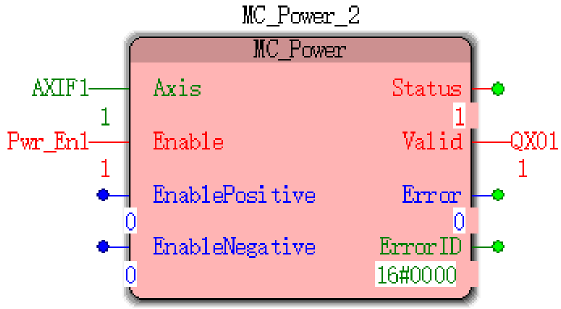
**Program Example Two**

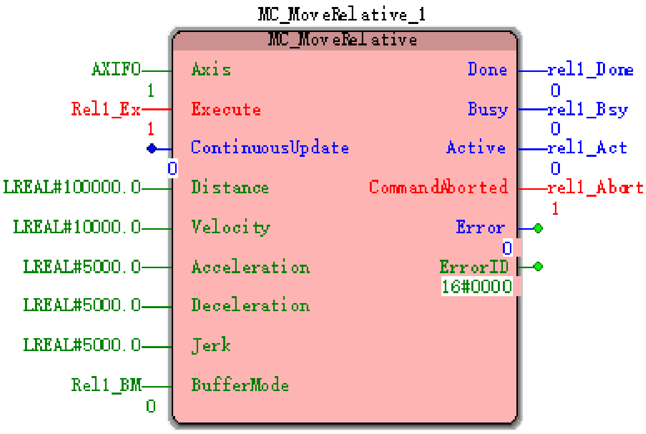
MC_MoveAdditive interrupt instruction examples below MC_MoveRelative

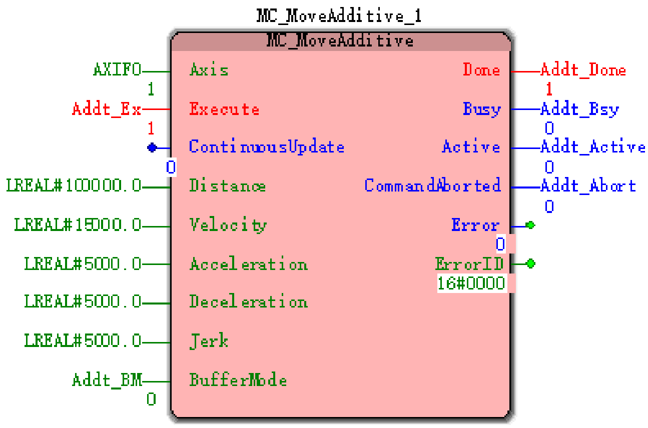
**1、Variables and procedures**

| Variable name | type of data | The initial value |
|---|---|---|
| MC_MoveRelative_1 | MC_MoveRelative | |
| AXIF0 | USINT | 1 |
| Rel1_Ex | BOOL | FALSE |
| Rel1_BM | INT | 0 |
| Rel1_Done | BOOL | |
| Rel1_Bsy | BOOL | |
| Rel1_Act | BOOL | |
| Rel1_Abort | BOOL | |
| MC_MoveAdditive1 | MC_MoveAdditive | |
| Addt_Ex | BOOL | FALSE |
| Addt_BM | INT | 0 |
| Addt_Done | BOOL | |
| Addt_Bsy | BOOL | |

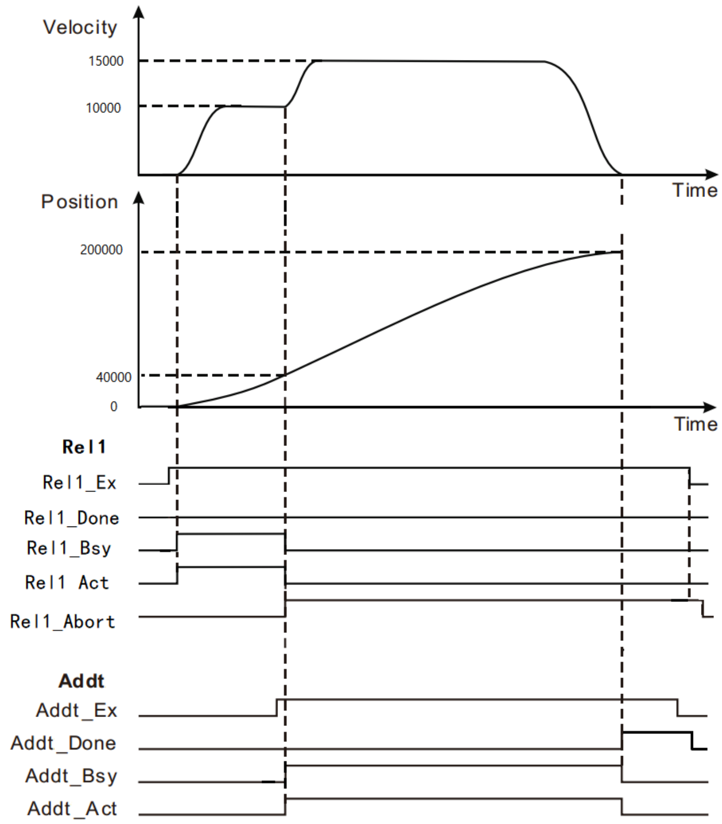
| | | |
|---|---|---|
| Addt_Active | BOOL | |
| Addt_Abort | BOOL | |



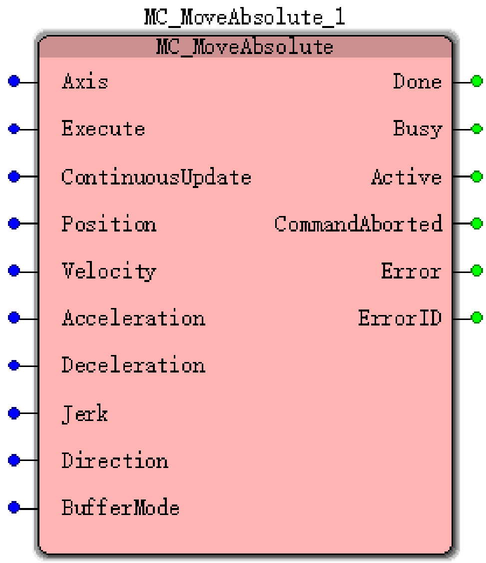

**2, Motion curve and timing diagram**




- When Rel1_Ex a FALSE to TRUE, the motion controller controls the operation of the servo motor to the current position as a reference point, the position of the shaft at Position = 40,000, additional displacement instruction execution, Addt_Ex a FALSE to TRUE, after one cycle, interrupt bit Rel1_Abt from FALSE becomes TRUE. Meanwhile, the servo motor movement to position the second additional command parameters. When the servo motor reaches a set distance (set this time from the sum of the distances to two instructions), completed by a bit Addt_Done FALSE to TRUE.
- When Addt_Ex a TRUE to FALSE, a bit Addt_Done reset cycle is complete.

## 11.4.5 MC_MoveAbsolute (absolute displacement instructions)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to set speed control axes, acceleration and deceleration to move relative to the target zero position | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute | When the Exexcute FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE (FALSE) | - |

| | | | | |
|---|---|---|---|---|
| Continuous Updata | Retention | - | - | - |
| Position (position) | Set the target position Rotary shaft: 0≤ Position <mold Linear axis: Unlimited (Unit: unit) | LREAL | Positive, negative, zero (0) | Exexcute from FALSE to TRUE |
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |
| Direction (direction) | Running direction (the rotation axis only when the parameter is effective) 1: Forward 2: The shortest distance 3: Reverse 4: Current direction | INT | 1: Forward 2:The shortest distance 3: Reverse 4:Current direction (Non-default) | Exexcute from FALSE to TRUE and the axis as a rotation axis mode |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0: immediately interrupted 1: Wait | INT | 0:immediately interrupted 1: Wait (0) | Exexcute from FALSE to TRUE |

**Description:**

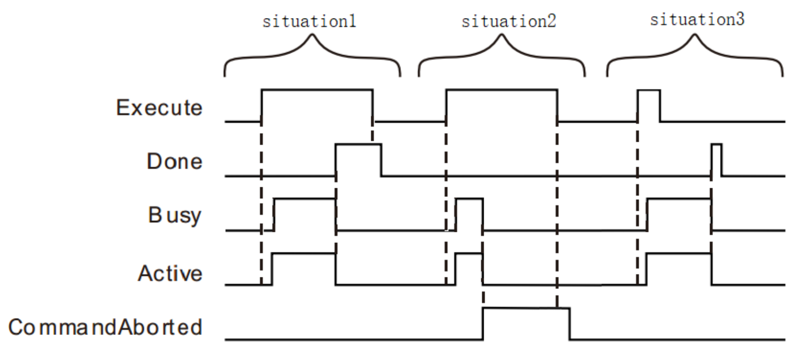
1. This instruction starts execution when the Execute FALSE to TRUE. The instruction is being executed when the Execute TRUE to FALSE, no effect on the implementation of the directive.

2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be re-executed, the parameters can be revalidated The pin comprises a Distance, Velocity, Acceleration, Deceleration, Jerk, Direction, BufferMode. 3. When the shaft as a rotation axis, Position parameters may be within 0 to die but not including the value of the mold, if the absolute value is greater than or equal to the parameter Position mode, this instruction execution error; when the shaft linear axis, parameter Position and regardless of the size of the mold can be set to an arbitrary constant. 4. Direction parameter is valid only when the shaft rotation axis, a detailed description of the parameters, refer to the instruction described in the function section Direction.

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;

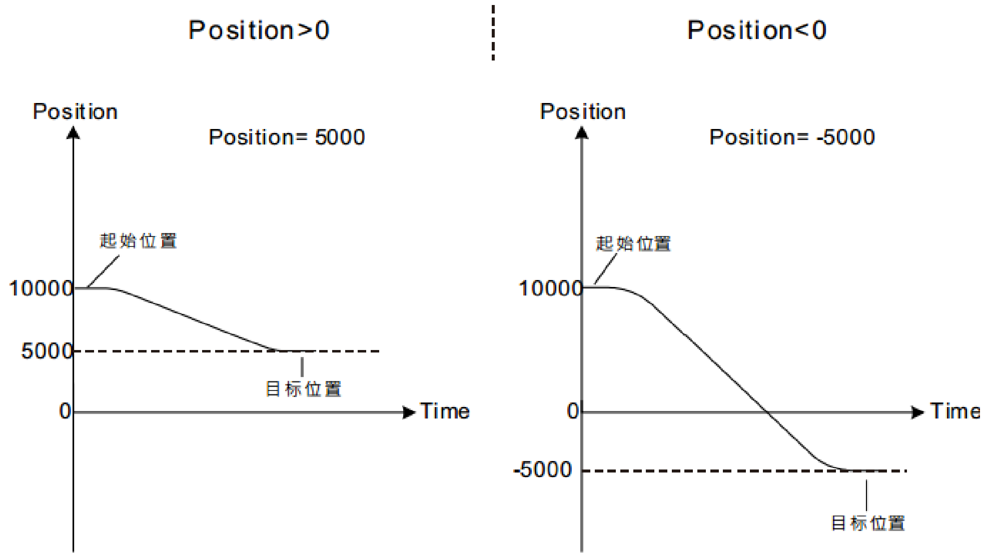
When the positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

**Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

**Case 3**: After during instruction execution, Execute a TRUE to FALSE, when the instructions are executed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

- **Function Description**

- This instruction is used to set speed control shaft, and a deceleration jerk to move relative to the target zero position.
- Axis position command execution starting absolute displacement 10,000, when Position> 0 (5000), the axis will reverse movement, the target position 5000, shown below as left; when Position <0 (-5000), the reaction shaft turn, the target position -5000, as shown in the lower right in FIG.

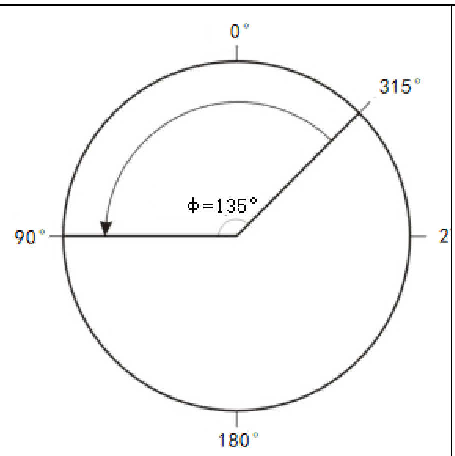
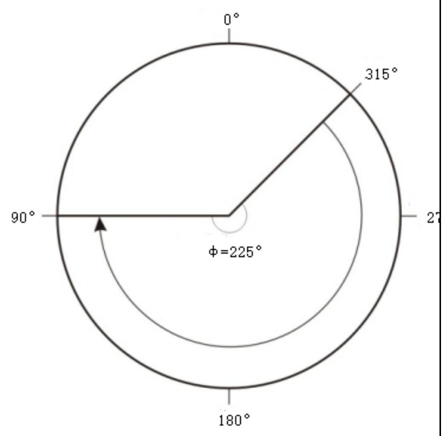
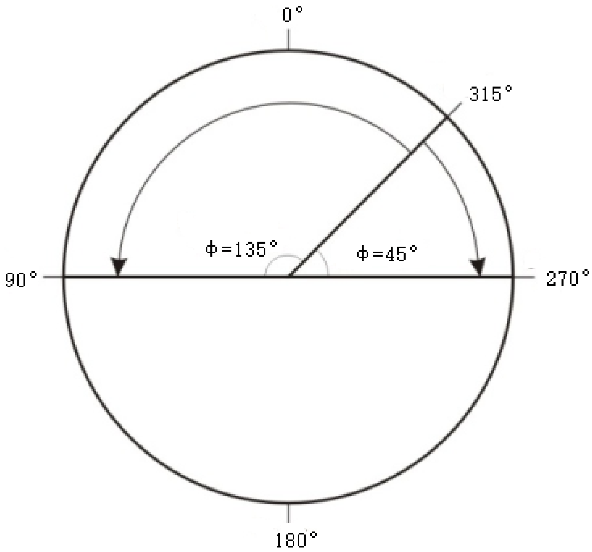


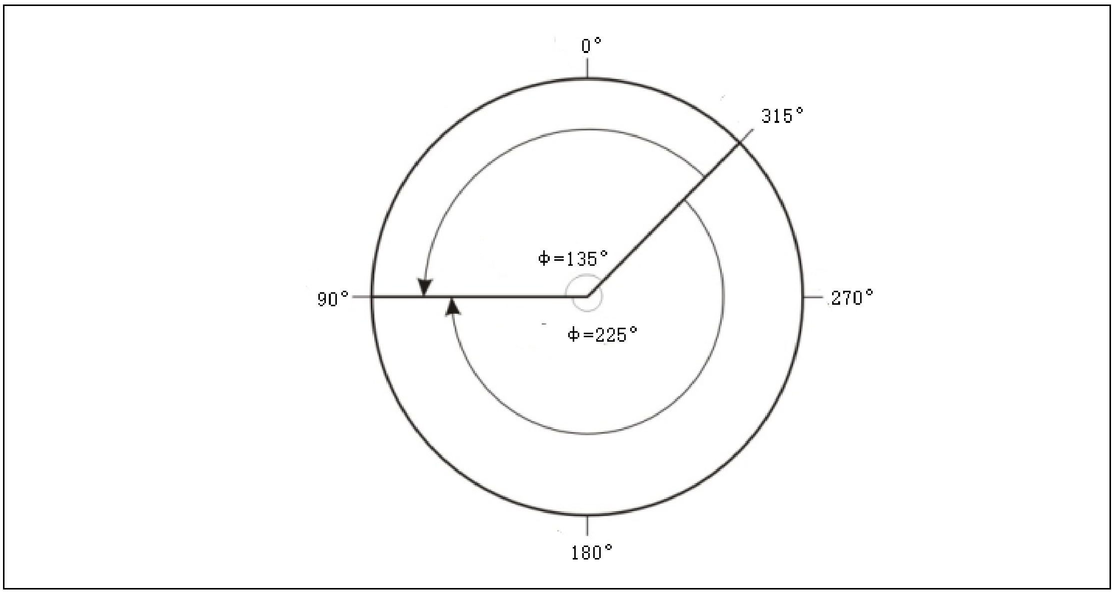

*note:Once this instruction is terminated other instructions during operation, the remaining distance is not completed will be discarded, the new instruction execution functions.*

- Direction

Direction parameters take effect only when the shaft is a rotating shaft, its different values, the direction of movement of the axis shown in the following table examples (modulo 360):

| | |
|---|---|
| Direction: 1 (positive direction)<br>Location: 315<br>Destination: 90<br>Movement angle: 135 ° | Direction: 3 (reverse orientation)<br>Location: 315<br>Destination: 90<br>Movement angle: 225 ° |

<table>
<tr>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>Direction: 2 (minimum)<br>Location: 315<br>Destination: 90<br>Movement angle: 135 °</td>
<td>Direction: 2 (minimum)<br>Location: 315<br>Destination: 270<br>Movement angle: 45 °</td>
</tr>
<tr>
<td colspan="2">
</td>
</tr>
<tr>
<td>Direction: 4 (continuation of the direction of the current)<br>Before the execution, the state of the function block rotating shaft; being inverted<br>Location: 315<br>Destination: 90<br>Movement angle: 225 °</td>
<td>Direction: 4 (continuation of the direction of the current)<br>Before the function block execution state of the shaft; still being forward<br>Location: 315<br>Destination: 90<br>Movement angle: 135 °</td>
</tr>
</table>

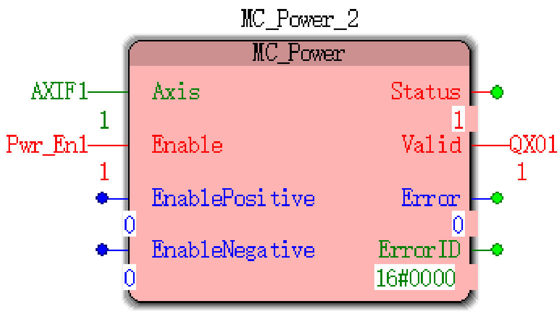


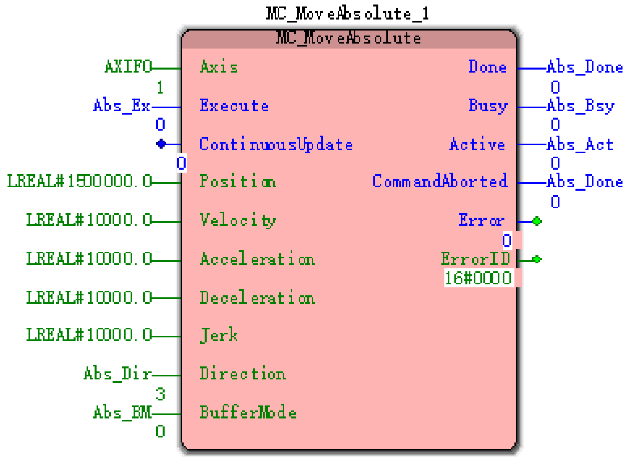

## Examples of a program

The controlled axis is a linear axis, while examples of single instruction is executed as follows MC_MoveAbsolute

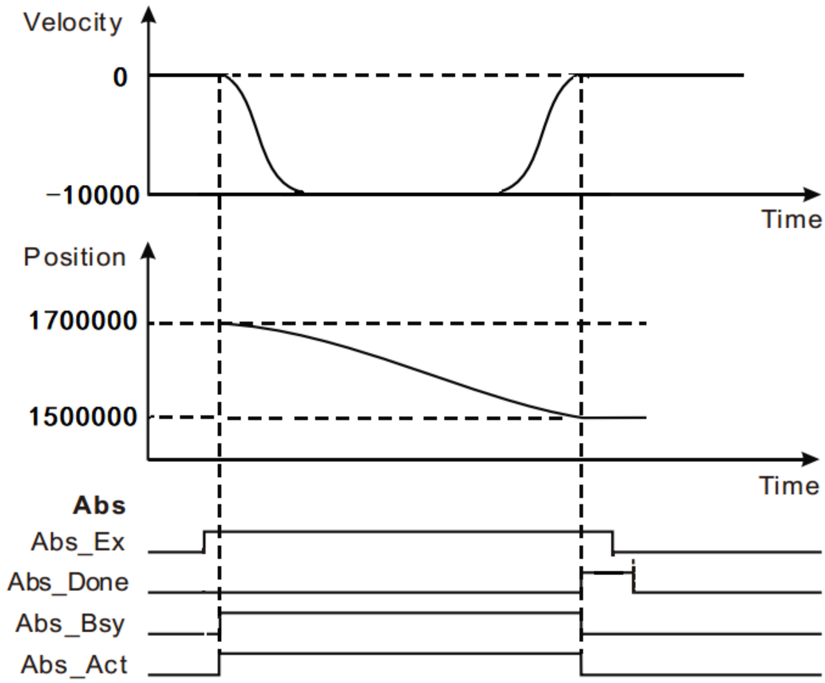
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveAbsolue_1 | MC_MoveAbsolute | |
| AXIF0 | USINT | 1 |
| Abs_Ex | BOOL | FALSE |
| Abs_Dir | INT | 1 |
| Abs_BM | INT | 0 |
| Abs_Done | BOOL | |
| Abs_Bsy | BOOL | |
| Abs_Act | BOOL | |
| Abs_Done | BOOL | |






**2, Motion curve and timing diagram**



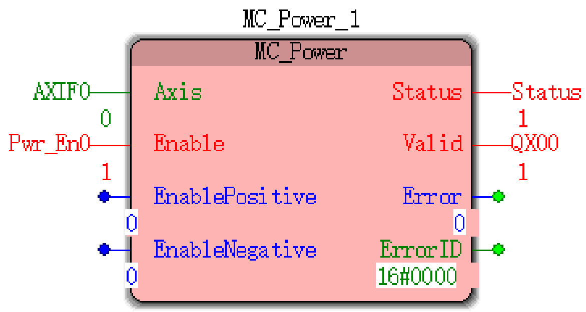

- When the Abs_Ex MC_MoveAbsolute FALSE to TRUE to start the instruction execution, the current position of the axis at this time is 1.7 million, 1.5 million target position.
- When the shaft moves to the 1500000, the instruction execution is completed.

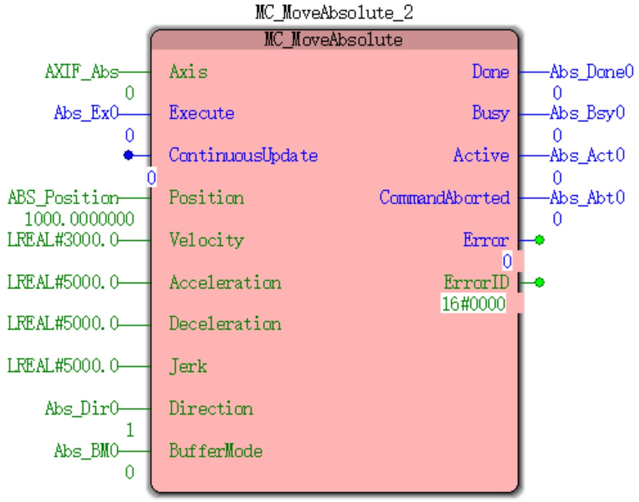
## Program Example Two

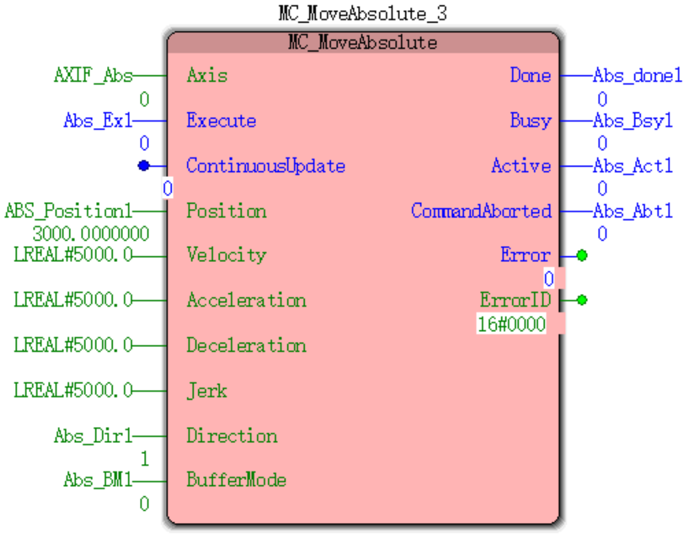
The controlled axis is a linear axis, MC_MoveAbsolute instruction interrupt instruction MC_MoveAbsolute examples are as follows:

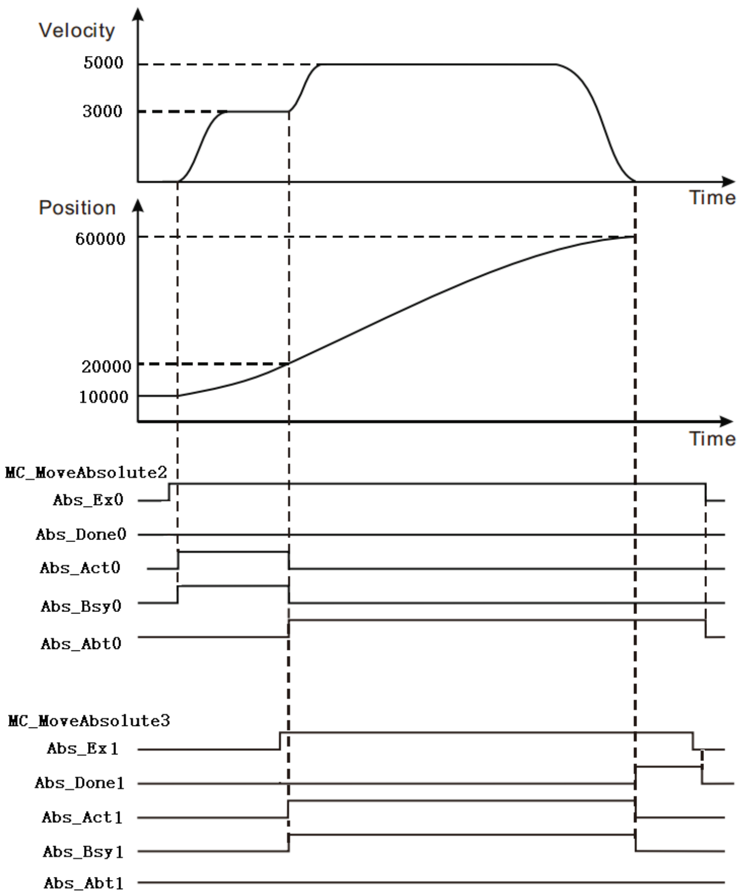
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveAbsolue_2 | MC_MoveAbsolute | - |
| Abs_Ex0 | BOOL | FALSE |
| ABS_Position | LREAL | 30000.0 |
| Abs_Dir0 | INT | - |
| Abs_BM0 | INT | 0 |
| Abs_Done0 | BOOL | FALSE |
| Abs_Bsy0 | BOOL | FALSE |
| Abs_Act0 | BOOL | FALSE |
| Abs_Abt0 | BOOL | FALSE |
| MC_MoveAbsolue_3 | MC_MoveAbsolute | - |
| Abs_Ex1 | BOOL | FALSE |
| ABS_Position1 | LREAL | 60000.0 |
| Abs_Dir1 | INT | - |
| Abs_BM1 | INT | 0 |
| Abs_Done0 | BOOL | FALSE |
| Abs_Bsy0 | BOOL | FALSE |
| Abs_Act0 | BOOL | FALSE |
| Abs_Abt0 | BOOL | FALSE |




MC_MoveAbsolute_2
MC_MoveAbsolute
AXIF_Abs
Axis
Done
Abs_Done0
Abs_Ex0
Execute
Busy
Abs_Bsy0
ContinuousUpdate
Active
Abs_Act0
ABS_Position
1000.0000000
Position
CommandAborted
Abs_Abt0
LREAL#3000.0
Velocity
Error
LREAL#5000.0
Acceleration
ErrorID
16#0000
LREAL#5000.0
Deceleration
LREAL#5000.0
Jerk
Abs_Dir0
Direction
Abs_BM0
BufferMode

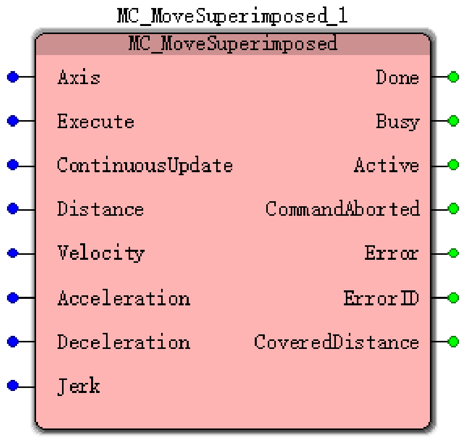

MC_MoveAbsolute_3
MC_MoveAbsolute
AXIF_Abs
Axis
Done
Abs_done1
Abs_Ex1
Execute
Busy
Abs_Bsy1
ContinuousUpdate
Active
Abs_Act1
ABS_Position1
3000.0000000
Position
CommandAborted
Abs_Abt1
LREAL#5000.0
Velocity
Error
LREAL#5000.0
Acceleration
ErrorID
16#0000
LREAL#5000.0
Deceleration
LREAL#5000.0
Jerk
Abs_Dir1
Direction
Abs_BM1
BufferMode




- When Abs_Ex0 a FALSE to TRUE, the first MC_MoveAbsolute instruction starts execution, the current position of the axis at this time is 1000, the target location 3000.
- When the shaft position is reached 20000, Abs_Ex1 a FALSE to TRUE, MC_MoveAbsolute second instruction begins execution, and the execution of the first instruction MC_MoveAbsolute is interrupted, the output parameter Abs1_Abt becomes TRUE.
- Position when the shaft reaches 60,000, the second MC_MoveAbsolute instructions are executed, the output parameter Abs2_Done becomes TRUE.

## 11.4.6 MC_MoveSuperimposed (additional displacement instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control movement of the shaft in the current state of the set speed, acceleration and deceleration some additional independent set distance. | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Exexcute FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE (FALSE) | - |
| Continuous Updata | Retention | - | - | - |
| Distance (distance) | Goal Setting distance (Unit: unit) | LREAL | Positive, negative, zero (0) | Exexcute from FALSE to TRUE |
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |

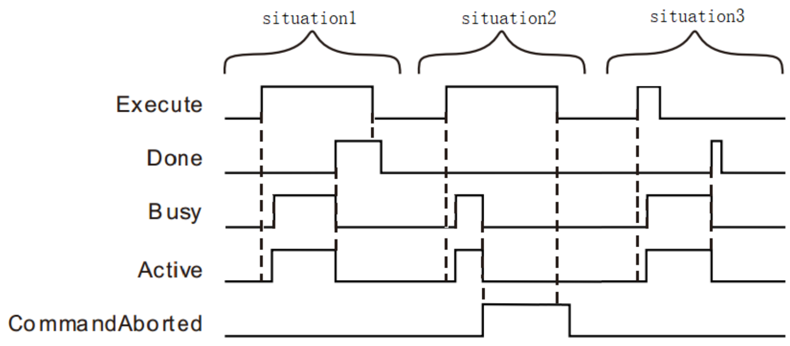
**Description:**

1, the instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed when the Execute TRUE to FALSE, no effect on the implementation of the directive.
2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be executed again, this time can be revalidated pin parameters include Distance, Velocity, Acceleration, Deceleration, Jerk.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| CoveredDistance (Additional cumulative distance) | The instruction to start the accumulated since the added distance | LREAL | Negative, positive, 0 |

➢ **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;
When the completion of the additional displacement, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

**Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

**Case 3**: After during instruction execution, Execute a TRUE to FALSE, when the instructions are executed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

➢ **Function Description**

● MC_MoveSuperimposed instruction is executed, the previous instruction does not terminate (not including MC_MoveSuperimposed MC_HaltSuperimposed and instructions) is executed, two instructions are simultaneously executed, distance, speed, acceleration and deceleration in real time superimposed (when an instruction reaches the set speed, its acceleration 0). A current instruction execution is completed, will not be superimposed on its speed, acceleration and deceleration, MC_MoveSuperImposed instruction still operate independently.

● State when the shaft is Standstill, MC_MoveSuperImposed instruction execution, MC_MoveSuperimposed MC_MoveRelative instruction and is equivalent to the instruction.

● When MC_MoveSuperimposed common command and control axis motion command, then execute other instruction motion (not including MC_MoveSuperimposed and MC_HaltSuperimposed instructions). If Buffermode move command after execution = 0, then the first movement MC_MoveSuperimposed instruction execution and instruction will be interrupted; if Buffermode instructions executed after the movement to other values, and the motion command MC_MoveSuperimposed instruction will not be executed first, interrupted.

● When MC_MoveSuperimposed command individually controlled axes, and then perform another MC_MoveSuperimposed instruction, the previous instruction was interrupted MC_MoveSuperimposed.

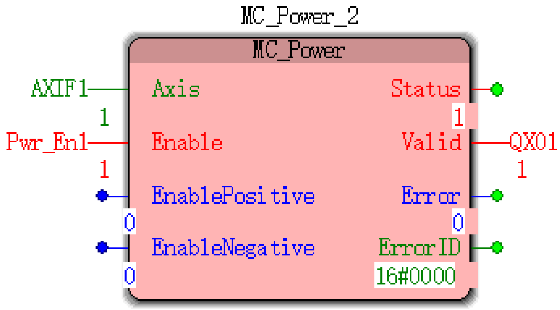
- MC_MoveSuperimposed instruction is executed, and then execution MC_HaltSuperimposed instruction, MC_MoveSuperimposed instruction was interrupted.
- This instruction does not affect the current state of the machine

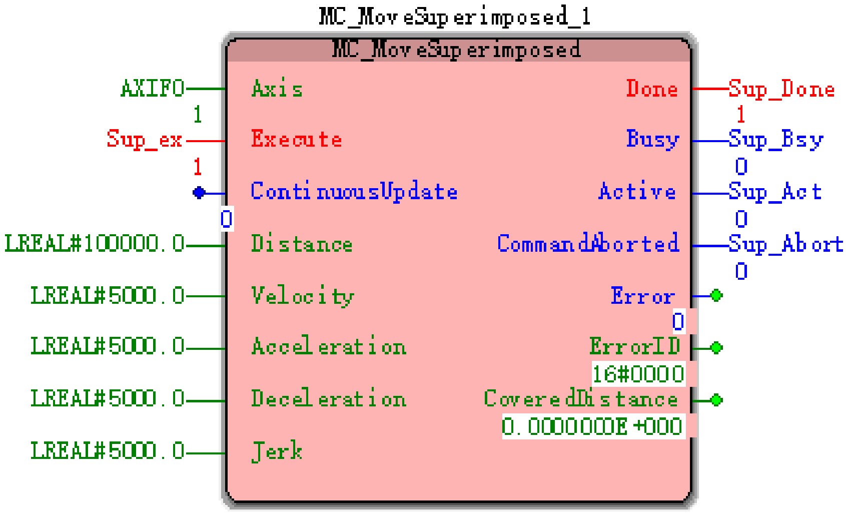
## Examples of a program

Examples when a separate instruction is executed as follows MC_MoveSuperimposed

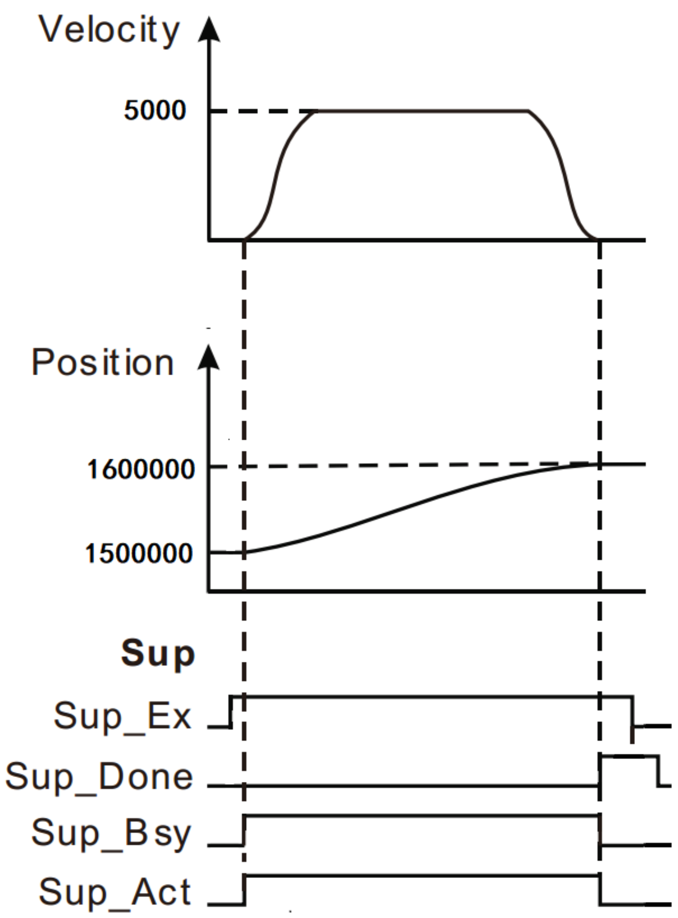
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveSuperimposed_1 | MC_MoveSuperimposed | |
| AXIF0 | USINT | 1 |
| Sup_Ex | BOOL | FALSE |
| Sup_Done | BOOL | |
| Sup_Bsy | BOOL | |
| Sup_Act | BOOL | |
| Sup_Abort | BOOL | |




**2, Motion curve and timing diagram**




- When Sup_Ex becomes TRUE, after a period, Sup_Bsy, Sup_Act becomes TRUE, the motion controller controls the operation of the servo motor to the current position as a reference point.
- After the completion of the servo motor set distance, Sup_Done becomes TRUE, and at the same time Sup_Bsy Sup_Act becomes FALSE.
- When Sup_Ex becomes FALSE, Sup_Done becomes FALSE.

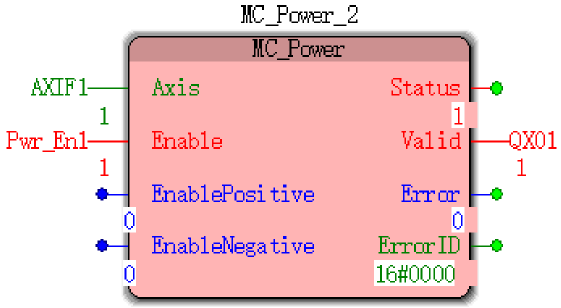
**Program Example Two**

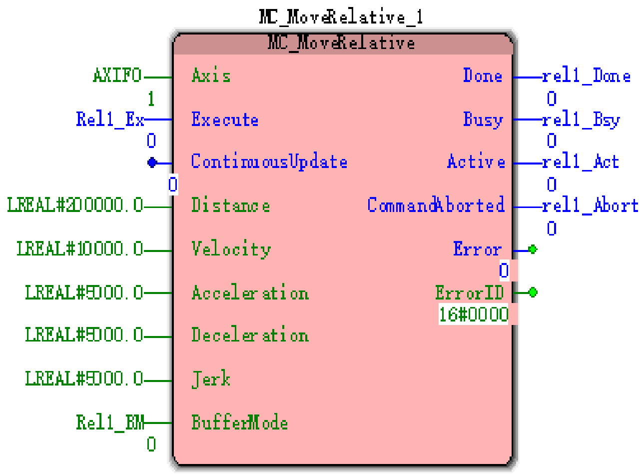
MC_MoveSuperimposed MC_MoveRelative and instructions with the example below:

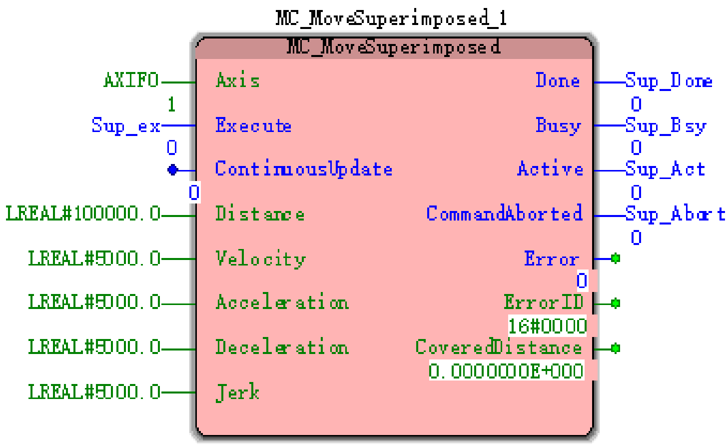
**1、Variables and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveRelative_1 | MC_MoveRelative | |
| AXIF0 | USINT | 1 |
| Rel1_Ex | BOOL | FALSE |
| Rel1_BM | INT | 0 |
| Rel1_Done | BOOL | |
| Rel1_Bsy | BOOL | |
| Rel1_Act | BOOL | |
| Rel1_Abort | BOOL | |

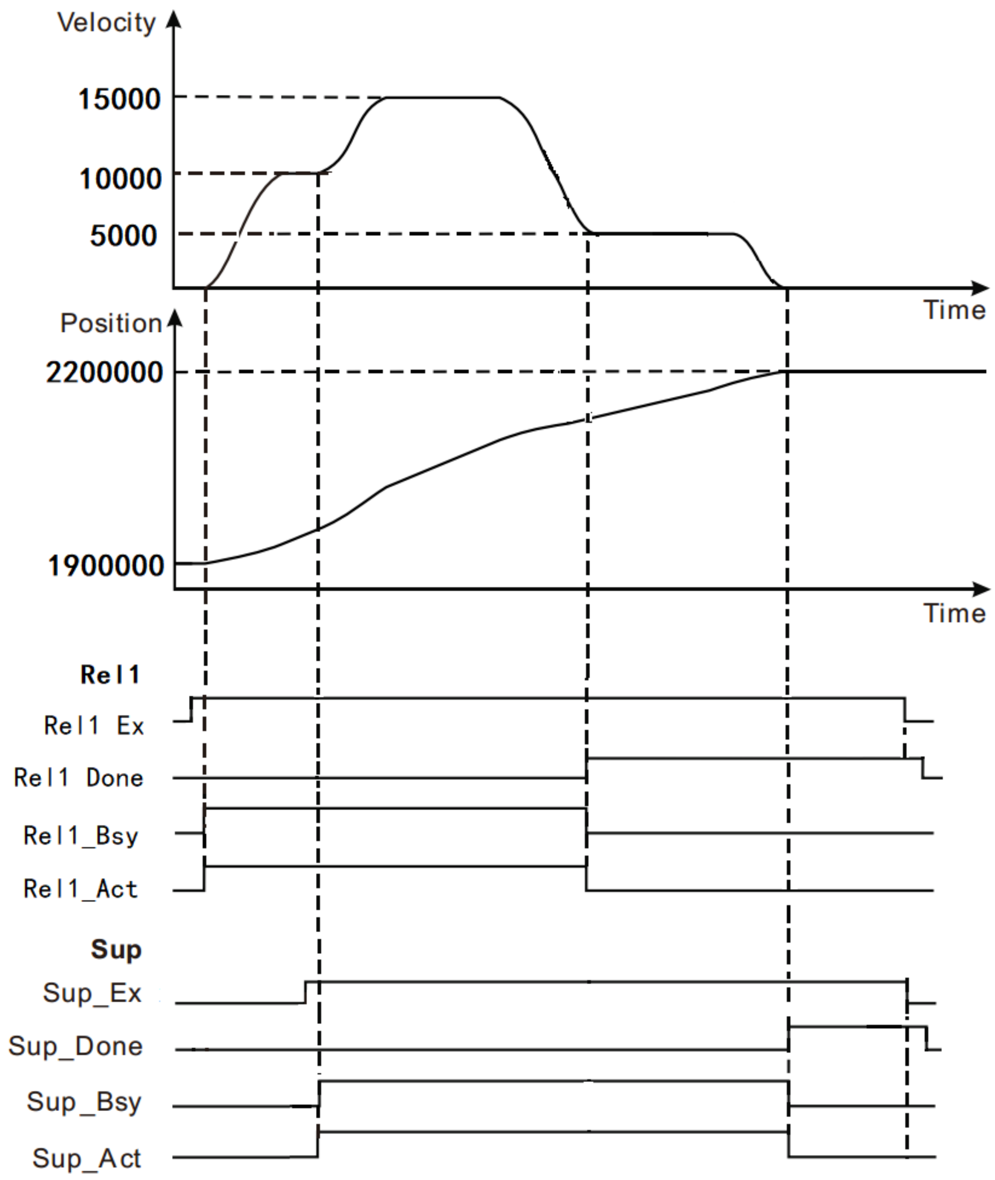
| MC_MoveSuperimposed_1 | MC_MoveSuperimposed | |
|---|---|---|
| Sup_Ex | BOOL | FALSE |
| Sup_Done | BOOL | |
| Sup_Bsy | BOOL | |
| Sup_Act | BOOL | |
| Sup_Abort | BOOL | |




**2、Timing and motion profiles of FIG.**




● When Rel_Ex becomes TRUE, after a period Rel_Act, Rel_BsyBecomes TRUE, the motion controller controls the operation of the servo motor to the current position as a reference point.

● When Sup_Ex becomes TRUE, after a period, Sup_Act, Sup_BsyBecomes TRUE, MC_MoveSuperimposed instruction starts execution, and an acceleration of the servo motor speed will (in this case acceleration is 0) is superimposed.

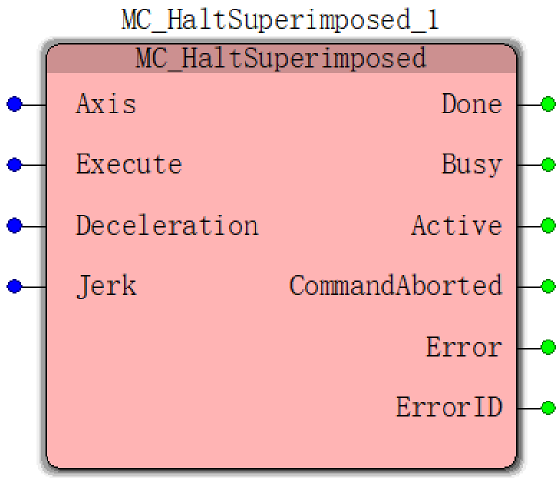
● When a position command is completed MC_MoveRelative, Rel_Done becomes TRUE, Rel_Bsy and Rel_Act becomes FALSE. The final position of the shaft of the total two command and processing initial position set position.

● When the additional distance MC_MoveSuperimposed instruction completion, Sup_Done becomes TRUE, Sup_Bsy and Sup_Act becomes FALSE. The final position of the shaft of the total two command and processing initial position set position.

● When Rel_Ex becomes FALSE, Rel_Done becomes FALSE. When Sup_Ex becomes FALSE, Sup_Done
Becomes FALSE.

## 11.4.7 MC_HaltSuperimposed (Pause additional displacement)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to suspend the additional displacement command | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | |
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |

**Description:**

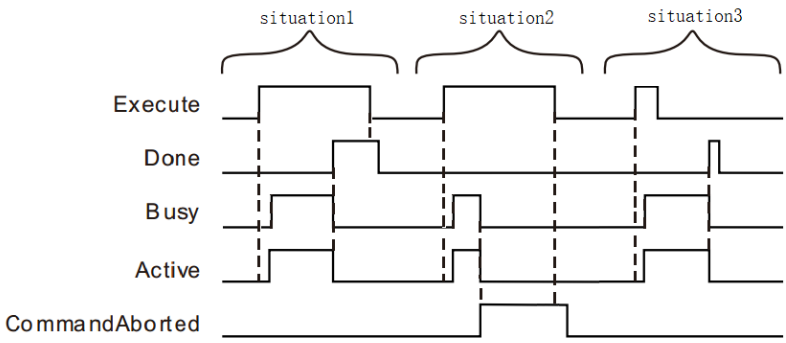
1,The instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed, ExecuteWhen the TRUE to FALSE, this instruction performs no effect.

2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be re-executed, the parameters can be revalidated The pin comprises Deceleration, Jerk.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output paramete**




**Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;

When the suspension is added to complete the displacement, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

**Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

**Case 3**: After during instruction execution, Execute a TRUE to FALSE, when the instructions are executed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

➢ **Function Description**

- MC_HaltSuperimposed instruction execution can only be interrupted for MC_MoveSuperimposed instructions.
- When MC_MoveSuperimposed and other motion control instructions common axis, and then execute MC_HaltSuperimposed instructions, instruction MC_HaltSuperimposed MC_MoveSuperimposed interrupt instruction, but the execution is not affected other motion instructions.

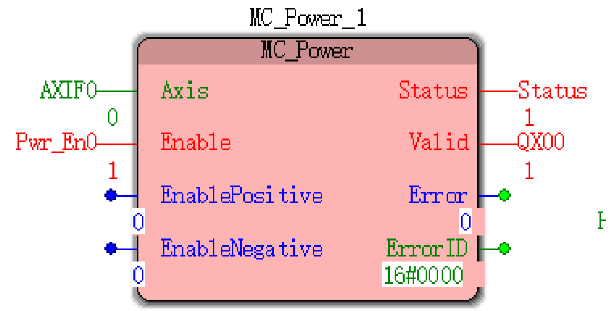
**Examples of a program**

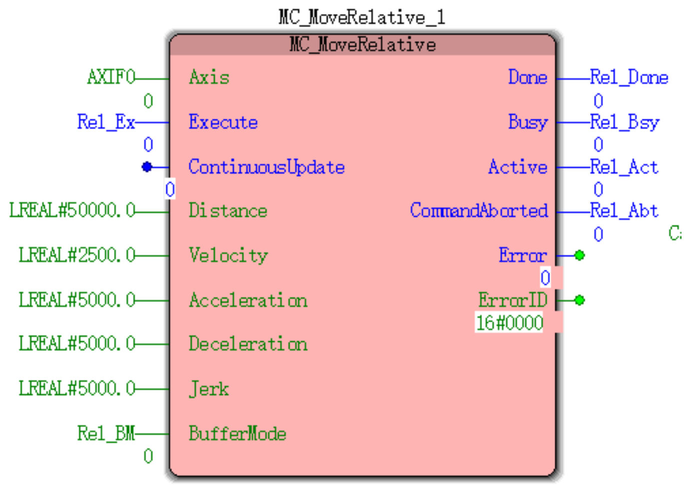
MC_HaltSuperimposed suspend instruction command is added to MC_MoveSuperimposed MC_MoveRelative instruction of the example below:

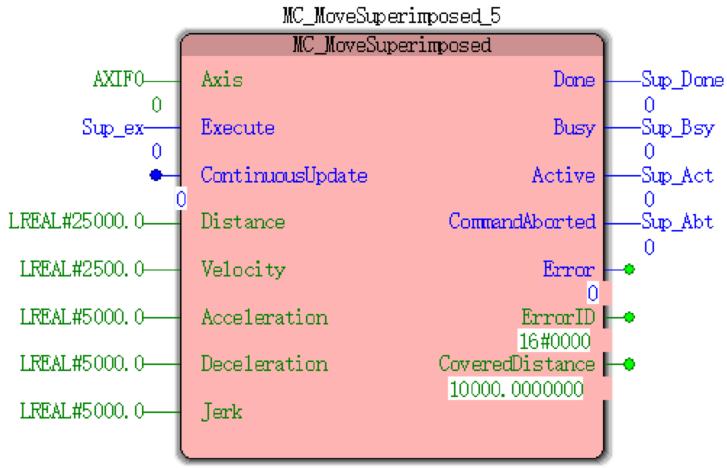
**1, variables, and procedures**

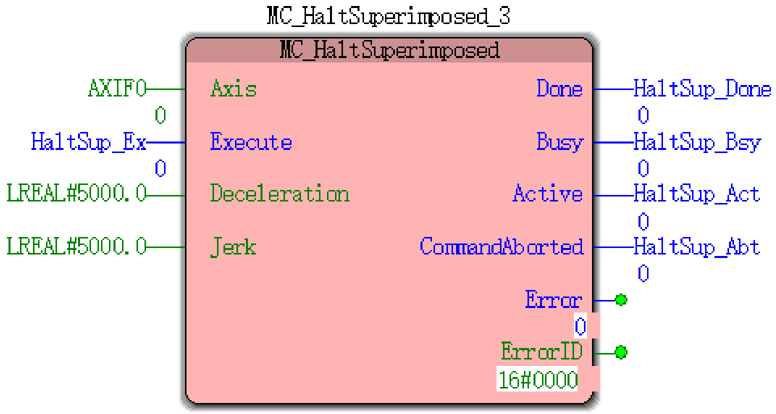
| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveRelative_1 | MC_MoveRelative | - |
| AXIF0 | USINT | 1 |
| Rel_Ex | BOOL | FALSE |
| Rel_BM | INT | 0 |
| Rel_Done | BOOL | FASLE |

| | | |
|---|---|---|
| Rel_Bsy | BOOL | FALSE |
| Rel_Act | BOOL | FALSE |
| Rel_Abort | BOOL | FALSE |
| MC_MoveSuperimposed_5 | MC_MoveSuperimposed | - |
| Sup_Ex | BOOL | FALSE |
| Sup_Done | BOOL | FALSE |
| Sup_Bsy | BOOL | FALSE |
| Sup_Act | BOOL | FALSE |
| Sup_Abort | BOOL | FALSE |
| MC_HaltSuperimposed_3 | MC_HaltSuperimposed | - |
| HaltSup_Ex | BOOL | FALSE |
| HaltSup_Done | BOOL | FALSE |
| HaltSup_Bsy | BOOL | FALSE |
| HaltSup_Act | BOOL | FALSE |
| HaltSup_Abt | BOOL | FALSE |

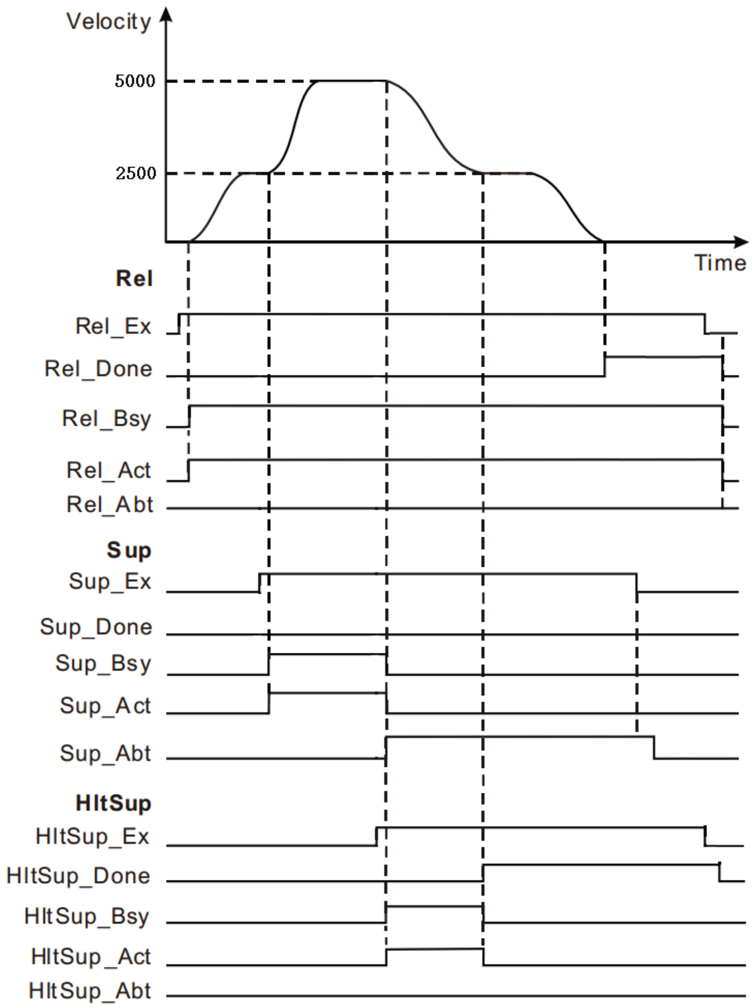




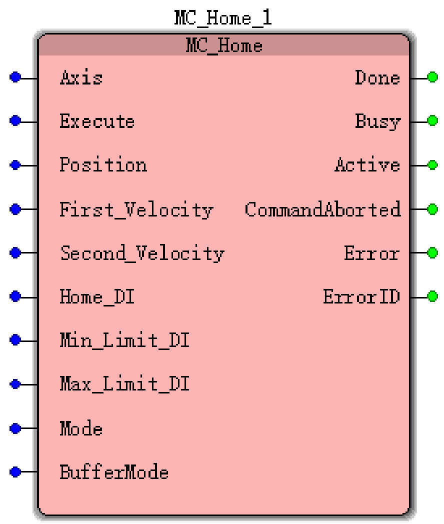

**2, Motion curve and timing diagram**




- When Rel_Ex becomes TRUE, after a period, Rel_Bsy and Rel_Act becomes TRUE, the motion controller controls the operation of the servo motor to the current position as a reference point. Sup_Ex becomes TRUE when, after a period, Sup_Bsy and Sup_Act becomes TRUE, MC_MoveSuperimposed instruction starts execution, and an acceleration of the servo motor speed will (in this case acceleration is 0 axis) is superimposed.
- When Hltsup_Ex becomes TRUE, after a period, Hltsup_Bsy and Hltsup_Act becomes TRUE, MC_HaltSuperimposed instruction starts execution, MC_MoveSuperimposed instruction was interrupted, Sup_Bsy, Sup_Act becomes FALSE, while Sup_Abt becomes TRUE. MC_HaltSuperimposed instruction interrupt MC_MoveSuperimposed instruction execution.
- When Hltsup_Done becomes TRUE, Hltsup_Bsy and Hltsup_Act become FALSE.
- MC_HaltSuperimposed instruction execution does not affect the execution of MC_MoveRelative instructions.

# 11.4.8 MC_Home (zero return instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used according to the mode and the control shaft speed parameter setting operation is performed homing | VEC-VA-MP-005-MA |




➢ **Input parameters**

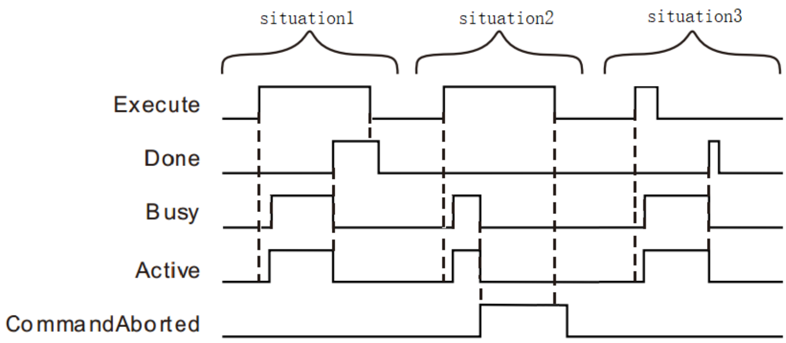
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE | - |
| Position (position) | The controlled axis origin offset, unit: unit | LREAL | Negative, positive, 0 | Exexcute from FALSE to TRUE |
| First_Velocity (First speed) | First speed shaft charged OPR, Unit: r / min | LREAL | Positive (non-default) | Determined by the Mode |

| | | | | |
|---|---|---|---|---|
| Second_Velocity (2nd speed) | 2nd speed shaft charged OPR, Unit: r / min | LREAL | Positive (non-default) | Determined by the Mode |
| Home_DI (Origin switch) | DI designated as a home switch | INT | 0 to 63 | Exexcute from FALSE to TRUE |
| Min_Limit_DI (Reverse limit) | Specify a limit switch as an inverted DI | INT | 0 to 63 | Exexcute from FALSE to TRUE |
| Max_Limit_DI (Forward limit) | DI designated as a forward limit switch | INT | 0 to 63 | Exexcute from FALSE to TRUE |
| Mode (mode) | Set homing mode | INT | 17 to 30, 35 | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Retention | - | - | - |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG timing variation output parameter**




- **Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;
  When the positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.
- **Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.
- **Case 3**: In the course of instruction execution, after the Execute TRUE to FALSE, when the positioning is completed, the Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, the Done becomes FALSE.

- **Function Description**

- This instruction according to the selected homing mode, and the home switch to forward or reverse limit switchMotion ControllerofDigitalThe entry point to achieve homing function.
- It provided a two-stage real axis speed mode and OPR OPR axis parameters in the software section. Describe homing mode, see Appendix A.
- This instruction only when the shaft is in a state StandStill may perform, when executed in other states, this command being given.
- Position parameter defines the return to the origin position offset relative to the servo zero position.

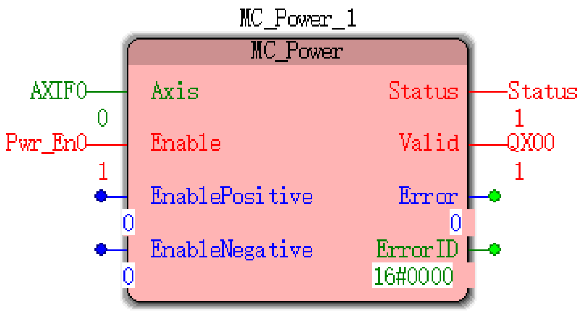
**Program Example**

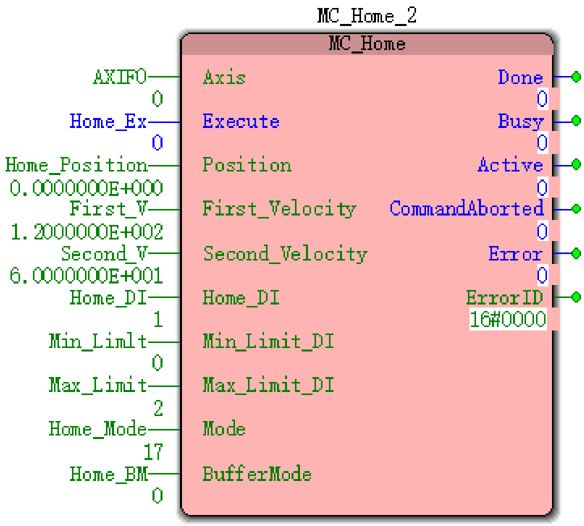
Homing select the appropriate mode and the photoelectric switch mechanism positions when Home_Ex from FALSE becomes TRUE, the motion controller controls the operation of the servo motor, the mechanical drive mechanism back to the origin position A.

1、Variables and procedures

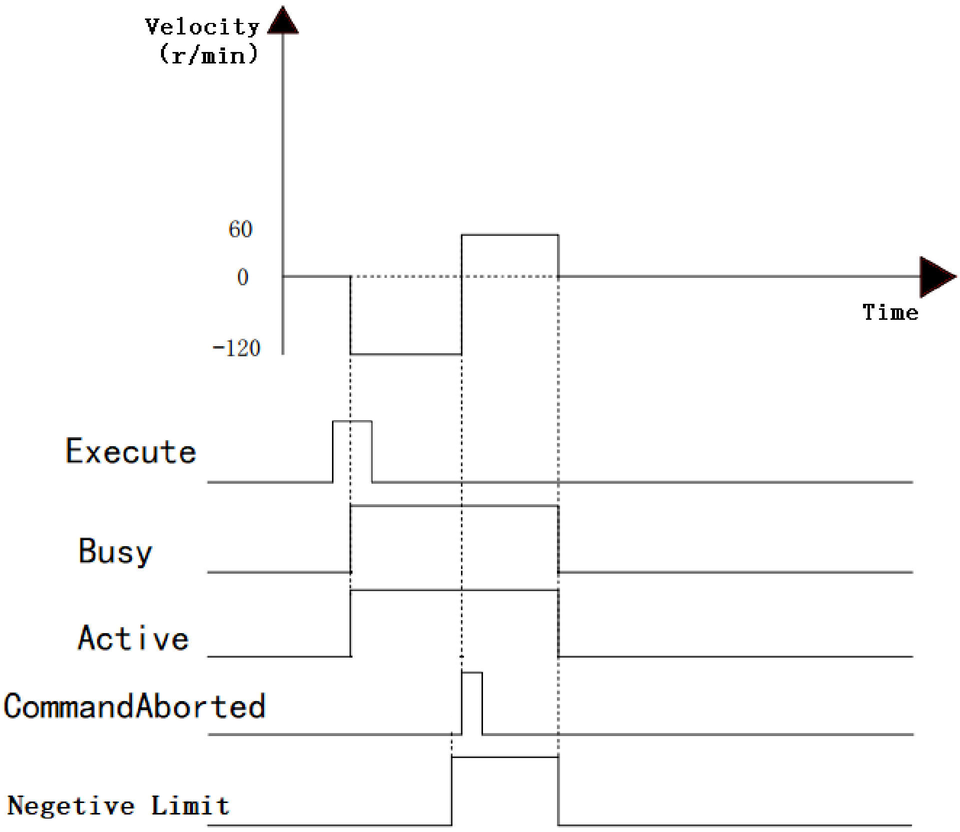
| variable name | type of data | The initial value |
|---|---|---|
| MC_Home_2 | MC_Home | |
| Home_Ex | BOOL | FALSE |
| Home_Position | LREAL | 0.0 |

| | | |
|---|---|---|
| First_V | LREAL | 120.0 |
| Second_V | LREAL | 60.0 |
| Home_DI | INT | 1 |
| Min_Limit | INT | 0 |
| Max_Limit | INT | 2 |
| Home_Mode | INT | 17 |
| Home_BM | INT | 0 |



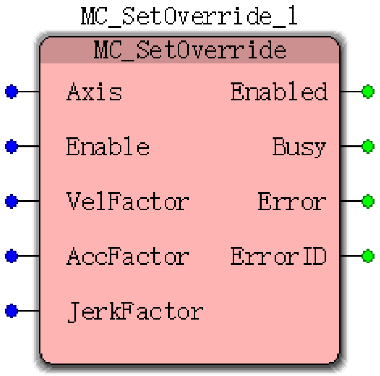

2、 Timing and motion profiles of FIG.




- Mode = 17, when the Home_Ex becomes FALSE TRUE, the motion controller controls the operation of the servo motor, means starts reverse rotation, after reaching the reverse limit switch is turned forward, and then leaving the stop position of the reverse limit switch, so as to drive institutions back to the mechanical origin position A.

## 11.4.9 MC_SetOverride (overshoot speed command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to change the current as a percentage of the controlled target speed shaft | VEC-VA-MP-005-MA |




➢ **Input parameters**

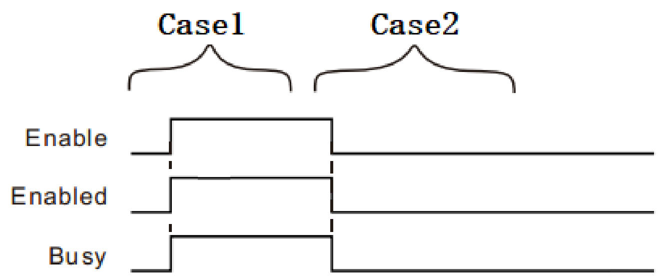
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable is TRUE |
| Enable | When Enable is TRUE, this AD | BOOL | TRUE or FALSE | - |
| VelFactor | Speed overshoot value (unit:%) | LREAL | 0 to 500 (0) | Enable is TRUE |
| AccFactor | Retention | LREAL | Retention | - |
| JerkFactor | Retention | LREAL | Retention | - |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Enabled (control) | When this parameter is | BOOL | TRUE or |

| | | | |
|---|---|---|---|
| | TRUE indicates output command under the control shaft | | FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing output parameter**




Case 1:When Enable the FALSE to TRUE, Busy,Enabled To TRUE

Case 2:When Enable a TRUE to FALSE, Enabled simultaneously become FALSE and Busy

➢ Function Description

This instruction is used as a percentage of the target speed change shaft

- The instructions may change the target speed as follows:

| | |
|---|---|
| MC_MoveAbsolute(Absolute displacement instructions) | MC_MoveAdditive(Additional displacement instruction) |
| MC_MoveRelative(Relative displacement instruction) | MC_SpecialMoveAbsolute(Special absolute displacement instructions) |
| MC_MoveVelocity(Speed command) | |

The new target speed as follows:

The target speed after the change in the target speed instruction execution = current *VelFactor

- VelFactor unit is%. "100", "100%." VelFactor valid range from 0 to 500, the instruction execution MC_SetOverride beyond the effective range, the instruction being given
- For MC_MoveVelocity command, the target relative speed change, the axis acceleration (or deceleration) to the target speed after the change according to Acceleration (or with Deceleration) the currently executing instruction.
- ForMC_MoveAbsolute,MC_MoveAdditive,MC_MoveRelative,MC_SpecialMoveAbsoluteCommand, the target speed after the change of the relative shaft is accelerated (or decelerated) according to Acceleration (or with Deceleration) the currently executing instruction *VelFactor2To the target speed after the change.

- When VelFactor set to "0", the target speed becomes "0", the performance of the operation shaft is decelerated at a rate of "0" operation.
- He wants to keep state action, but want to temporarily stop the axis movement, the VelFactor set to "0." At this time, the shaft does not change state.
- When motion or motion instruction execution transfer instruction may be changed VelFactor to set a new target speed

Enable is TRUE, modify VelFactor, VelFactor take effect immediately without restarting MC_SetOverride instruction.

- Enable is TRUE, modify VelFactor, VelFactor beyond the effective range, MC_SetOverride given instruction, it returns the target speed 100%. When Enable becomes FALSE, in order to accelerate VelFactor = 100 or reduced to a target.
- When MC_MoveVelocity instruction execution using MC_SetOverride instruction, when the instruction becomes TRUE MC_MoveVelocity InVelocity is, even when changing the target speed, the TRUE state InVelocity is maintained.
- You can use a same axis of the module.

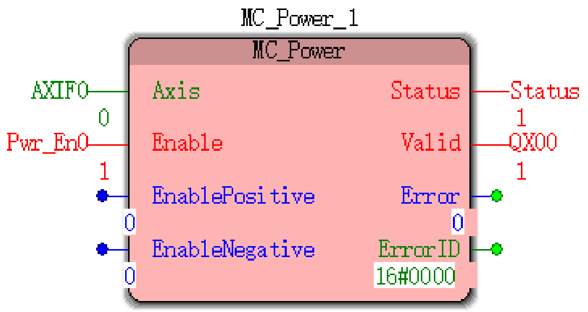
Program Example

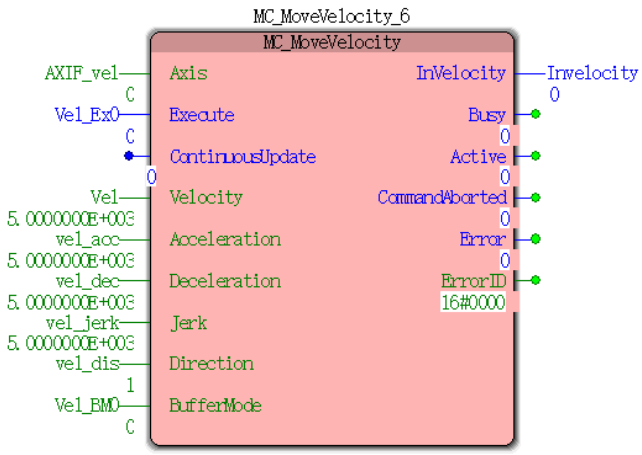
In the example below when MC_SetOverride instruction execution:

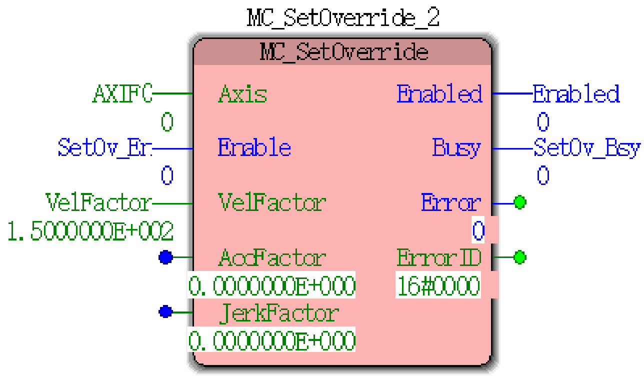
Impact on the results MC_MoveVelocity instruction paradigm about MC_SetOverride instruction execution.

1 variables, and the program name

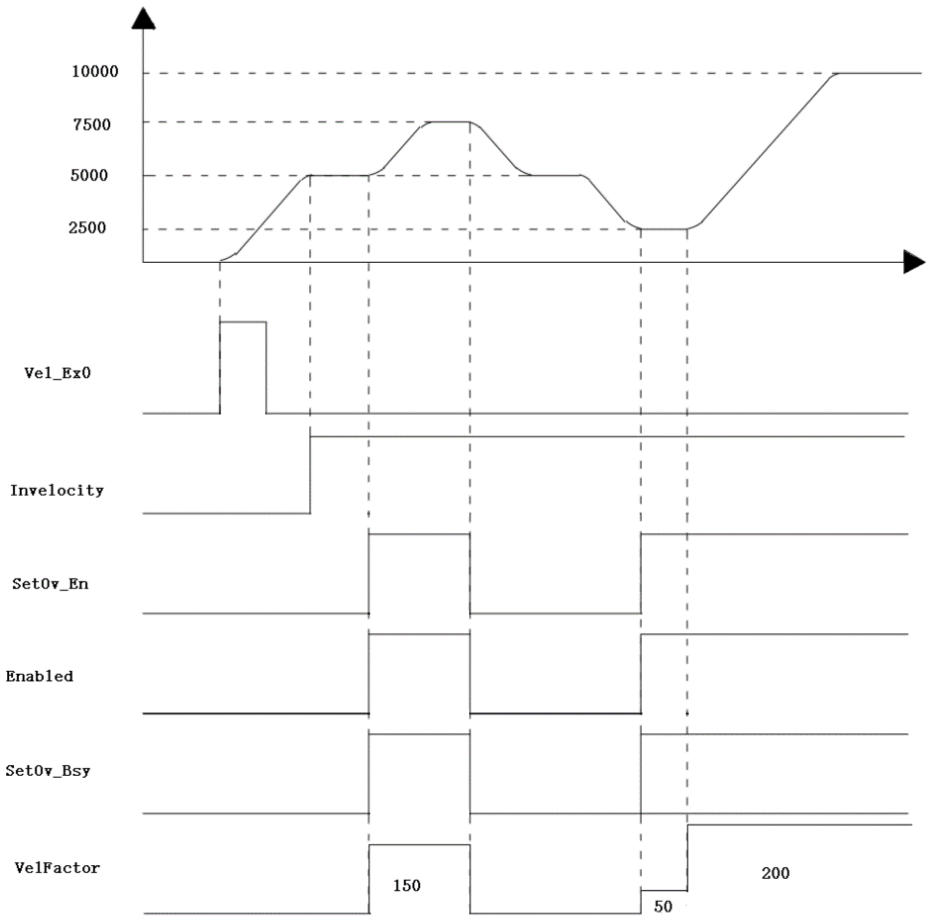
| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVeleocity_6 | MC_MoveVelocity | |
| AXIF0 | USINT | 0 |
| Vel_EX0 | BOOL | FALSE |
| Vel | LREAL | 5000.0 |
| Vel_acc | LREAL | 5000.0 |
| Vel_dec | LREAL | 5000.0 |
| Vel_Jerk | LREAL | 5000.0 |
| Vel_dis | INT | 1 |
| Vel_BM0 | INT | 0 |
| Invelocity | BOOL | - |
| MC_SetOverride_2 | MC_SetOverride | |
| SetOv_En | BOOL | FALSE |
| VelFactor | LREAL | 150.0 |
| Enabled | BOOL | |
| SetOv_Bsy | BOOL | |



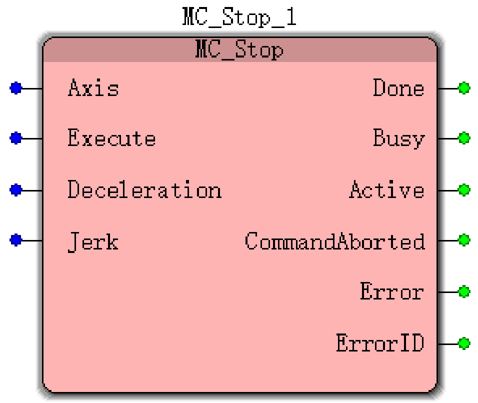

2 Motion curve and timing diagram




- When Vel_Ex0Becomes TRUEAfter time, a period Vel_BsyAnd Vel_Act becomes TRUEAfter axis starts forward rotation, the output reaches the set speed Invelocity TRUEAnd then SetOv_EnSet TRUEWhile Enbaled, SetOv_BsyBecomes TRUE, MC_SetOverride effective date of the controlled axis according VelFactorThe value of re-generate the target speed.
- When SetOv_EnBecomes FALSEWhen, VelFactor = 100 corresponds to the deceleration target speed.
- MC_SetOverride modified during execution of instructions VelFactorValues, VelFactorValue takes effect immediately, the target speed command MC_MoveVelocity will change accordingly.

## 11.4.10 MC_Stop (stop command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control deceleration of the axes of the set deceleration until it stops. Stopping state machine enters the shaft | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Execute TRUE when the FALSE to |
| Execute | When the Execute FALSE to TRUE, the instruction is executed | BOOL | TRUE or FALSE | - |
| Deceleration | decrease speed (Unit: unit / S2): | LREAL | Positive (non-default) | |
| Jerk | Deceleration change rate (Unit: unit / S3) | LREAL | Positive (non-default) | |

**Description:**

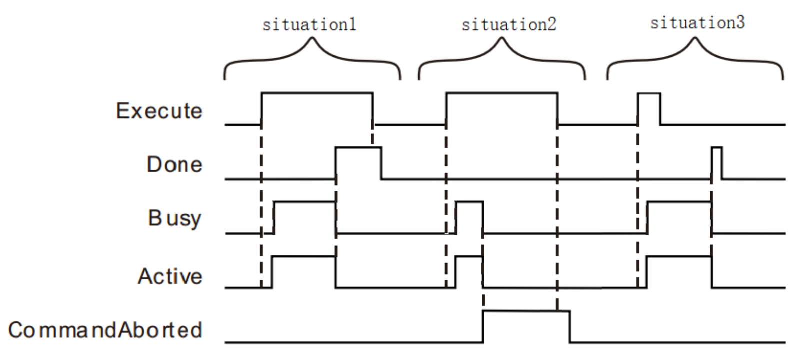
1,The instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed, ExecuteWhen the TRUE to FALSE, this instruction performs no effect.

2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be re-executed, the parameters can be revalidated The pin comprises Deceleration, Jerk.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




➢ **Case 1:**When the Execute FALSE to TRUE, after a period at the same time becomes TRUE and the Active Buys; when the position is reached, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

- **Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.
- **Case 3**: After during instruction execution, Execute a TRUE to FALSE, when positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

- **Function Description**

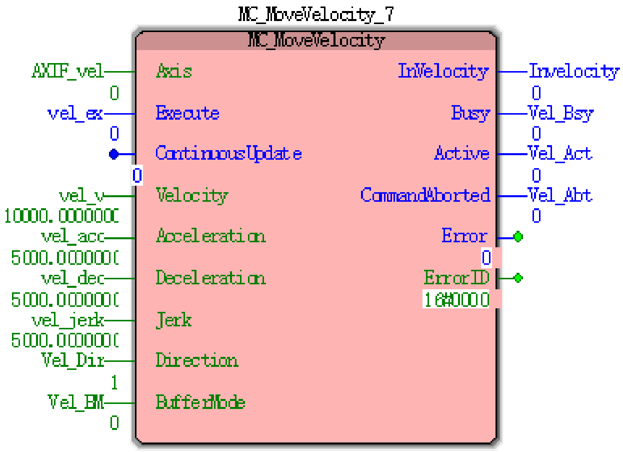
- MC_Stop instruction execution is completed, the shaft speed falls to zero as long as Excute is TRUE, it has been Stopping axis status, other motion instructions can not be executed at this time.
- And comparing MC_Halt instructions, instruction MC_Stop lock shaft, the controller can not perform another movement instruction (instruction not included MC_Stop) MC_Stop during execution of instructions. After MC_Stop instruction has completed, the shaft has stopped, the controller can not execute other instruction motion, only when the MC_Stop Excute by TRUE to FALSE in order to execute other instruction motion

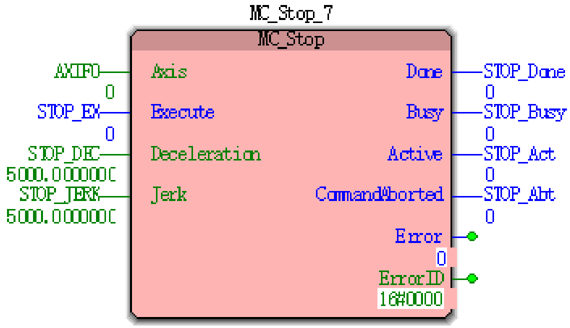
**Program Example**

MC_Stop example below when instruction execution:

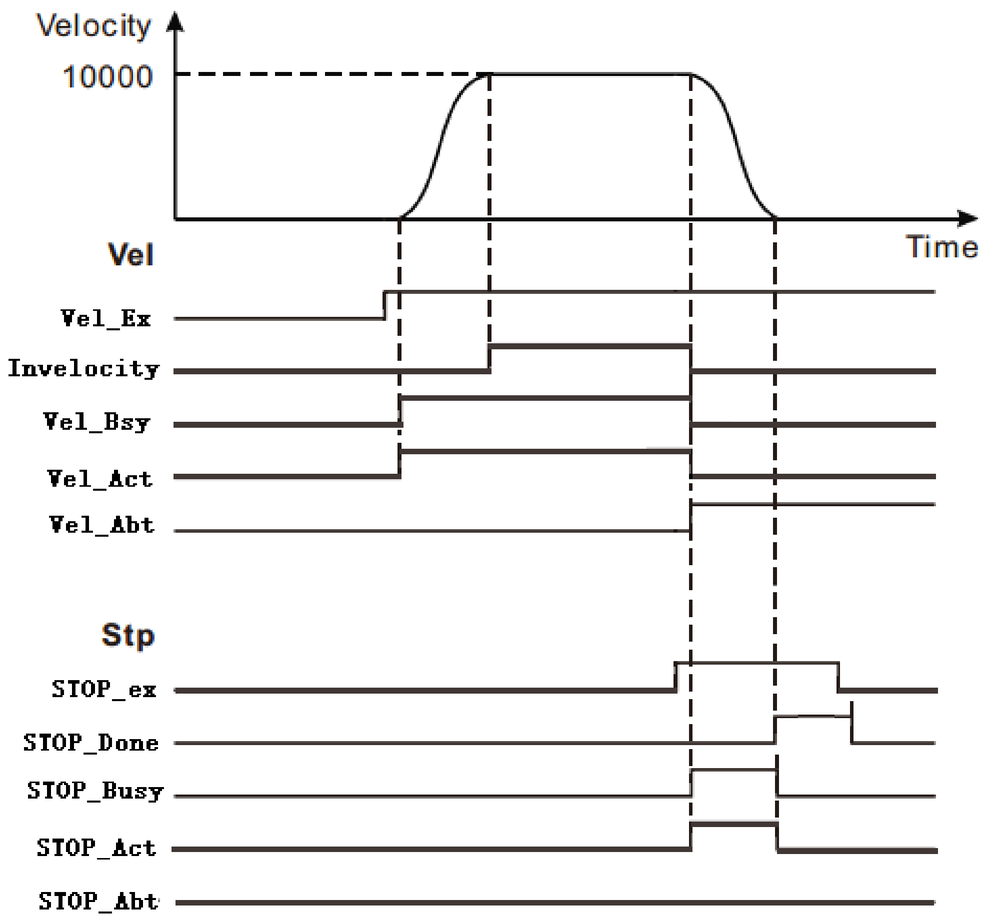
1, Variables, and the program name

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVeleocity_7 | MC_MoveVelocity | - |
| AXIF_vel | USINT | 0 |
| vel_ex | BOOL | FALSE |
| Vel_v | LREAL | 10000.0 |
| Vel_acc | LREAL | 5000.0 |
| Vel_dec | LREAL | 5000.0 |
| Vel_jerk | LREAL | 5000.0 |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| Invelocity | BOOL | |
| Vel_Bsy | BOOL | |
| Vel_Act | BOOL | |
| Vel_Abt | BOOL | |
| MC_Stop_7 | MC_Stop | |
| AXIF0 | USINT | 0 |
| STOP_EX | BOOL | FALSE |
| STOP_DEC | LREAL | 5000.0 |
| STOP_JERK | LREAL | 5000.0 |
| STOP_Done | BOOL | |
| STOP_Busy | BOOL | |
| STOP_Act | BOOL | |
| STOP_Abt | BOOL | |



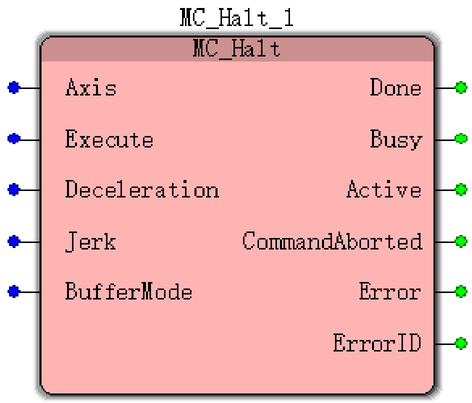

2, Motion curve and timing diagram




- When Vel_Ex becomes TRUE, after a period Vel_Bsy, Vel_Act becomes TRUE, the servo motor begins to move forward. When the servo motor reaches the target speed, Invelocity becomes TRUE.
- When STOP_ex becomes TRUE, after a period STOP_Busy, STOP_Act becomes TRUE, while Invelocity becomes FALSE, Vel_Abt becomes TRUE, the servo motor starts to decelerate.
- When the shaft speed is reduced to zero, STOP_Done becomes TRUE, while STOP_Busy, STOP_Act becomes FALSE.
- When STOP_ex becomes FALSE, after a period STOP_Done becomes FALSE.

## 11.4.11 MC_Halt (pause command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control deceleration of the axes of the set deceleration until it stops. | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Execute TRUE when the FALSE to |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction is executed | BOOL | TRUE or FALSE | - |
| Deceleration (decrease speed) | decrease speed (Unit: unit / S2): | LREAL | Positive (non-default) | Execute TRUE when the FALSE to |
| Jerk (Rate of change of deceleration) | Deceleration change rate (Unit: unit / S3) | LREAL | Positive (non-default) | Execute TRUE when the FALSE to |

| | | | | |
|---|---|---|---|---|
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0:immediatelyint errupted 1: Wait | INT | 0: immediately interrupted 1: Wait (0) | Execute TRUE when the FALSE to |

**Description:**

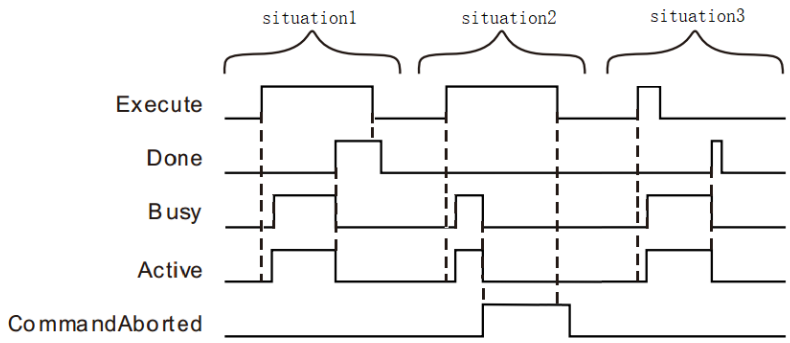
1,The instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed, ExecuteWhen the TRUE to FALSE, this instruction performs no effect.

2, when the instruction is being executed, the Execute again by the FALSE to TRUE, the instructions may be re-executed, the parameters can be revalidated The pin comprises Deceleration, Jerk, BufferMode.

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborte d (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG output timing parameters**



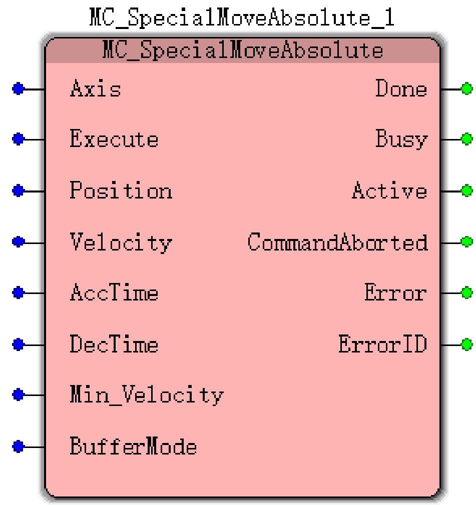

- **Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;
  When the position is reached, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.
- **Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.
- **Case 3**: After during instruction execution, Execute a TRUE to FALSE, when positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

- **Function Description**

  - MC_Halt instruction starts execution, the state machine enters DiscreteMotion, when the shaft speed is reduced to 0, Done becomes TRUE, while the state machine changes to Standstill.
  - MC_Stop instructions and comparison, the MC_Halt instruction does not lock shaft, the movement of the controller may perform other instructions. MC_Halt during instruction execution, the shaft during deceleration, may perform other instruction motion MC_Halt interrupt instruction; MC_Halt finish executing the instruction, after the shaft has been stopped, the controller may perform other motion command to restart the shaft.

## 11.4.11 MC_SpecialMoveAbsolute (special absolute displacement instructions)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to set speed control axes, acceleration and deceleration to move relative to the target zero position | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Pulse mode**: 0-4 (real axis) 5 to 11 (imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Exexcute FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE (FALSE) | - |
| Position (position) | Set the target position (Unit: unit) | LREAL | Positive, negative, zero (0) | Exexcute from FALSE to TRUE |
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| AccTime (acceleration time) | Target set acceleration time (Unit: S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| DecTime (deceleration time) | The set target deceleration time (Unit: S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| Min_Velocity | The set minimum target rotational speed (Unit: unit / S) | LREAL | Positive (non-default) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0:immediately interrupted 1: Wait | INT | 0: immediately interrupted 1: Wait (0) | Exexcute from FALSE to TRUE |

**Description:**

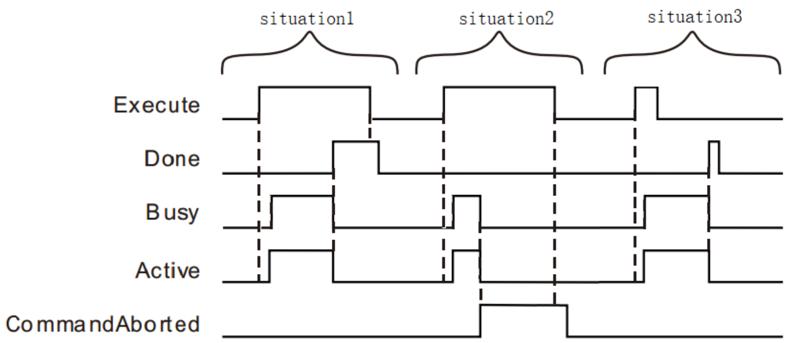
1, the instruction is executed, the controlled axis Min_Velocity actuated to stop Min_Velocity.

2, the next instruction to be executed = 1 ControlModel, the amount of which is a semi-closed loop control pulse, before the instruction has not been executed, according to the input parameters on the module, a programming corresponding to a given position and a given speed table runtime, the current feedback pulse, determined at a given speed. Thus the present non-reciprocal adjustment module configured to control relatively large inertia would be more stable.

➢ **Output parameters**

| **name** | **Features** | **type of data** | **Output range** |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**



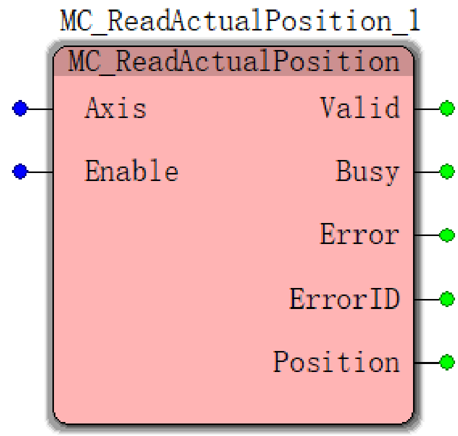

➢ **Case 1:**When the Execute FALSE to TRUE, after a period at the same time becomes TRUE and the Active Buys; When the positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

➢ **Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

➢ **Case 3**: After during instruction execution, Execute a TRUE to FALSE, when the instructions are executed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

## 11.4.12 MC_ReadActualPosition (real position instruction read)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the actual position of the axis | VEC-VA-MP-005-MA |




➢ **Input parameters**

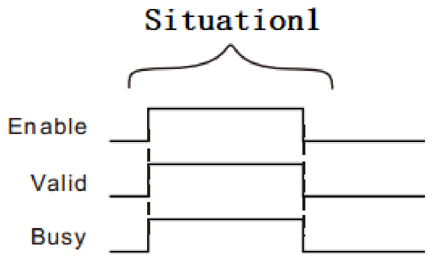
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable made to TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction execution | BOOL | TRUE or FALSE | - |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Valid (output valid) | The parameter output outputs a valid instruction is TRUE | BOOL | TRUE |
| Busy (execution) | This parameter indicates to | BOOL | TRUE or |

| | | | |
|---|---|---|---|
| | TRUE output instruction is executed | | FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| Position (actual position) | The actual position of the axis | LREAL | Real |

➢ **FIG output change timing**




**Case 1:**When Enable the FALSE to TRUE, Valid and Busy simultaneously become TRUE ,. When Enable becomes FALSE, Valid, Busy all become FALSE.

➢ **Function Description**

This instruction is used to read the actual position of the shaft (including the real axis and the imaginary axis encoder shaft)

- The actual position

Units of the actual position of this instruction is read as a unit, and the unit of the servo actuator feedback to the controller the position of a pulse, therefore the actual position obtained by the feedback pulses position servo drive after conversion, use the axis parameter in the conversion of Motor_PPC, Reductor_Num, Reductor_Den, means lead / perimeter (Screw_Lead / Disc_Circumference). Conversion relation shown in the following formula:

$$\text{ActualPosition} = \frac{institutional\ lead\ /\ perimeter}{(motor_{PPC}) * \frac{Reductor\_Num}{Reductor\_Den}} * Servo\ position\ feedback\ pulse$$

Position output performed when a linear axis, the command axis = ActualPosition;

If the axis of the rotary shaft, the output of the instruction execution Position = ActualPosition% Modulo (modulo result Position ActualPosition press axis parameters do modulo operation), the value of Position changes between 0 ~ Modulo.

- **The actual location update timing**

Because of this actual position from the position feedback servo drive pulses, the refresh timing of the actual position provided by MC_AXIS_REF Sample_Time sampling time of the pulse encoder feedback decisions. In a sampling period, the number of servo position feedback pulses to the controller action occurs only once. Thus, the real axis command read

Real-time position is less than the actual position capture, real-time location For obtaining higher, use the position capture function.

- **The actual position of influence MC_SetPosition**

After MC_SetPosition instruction executed, the actual position of the read command MC_ReadActualPosition shall be added is

MC_SetPosition position shift amount caused by the instruction, as shown in the following formula in terms of the relationship:

$$\text{ActualPosition} = \text{MC}\quad \text{SetPosition} + \frac{institutional\ lead{-}perimeter}{(motor_{PPC}) * \frac{Reductor_{Num}}{Reductor_{Den}}} *$$

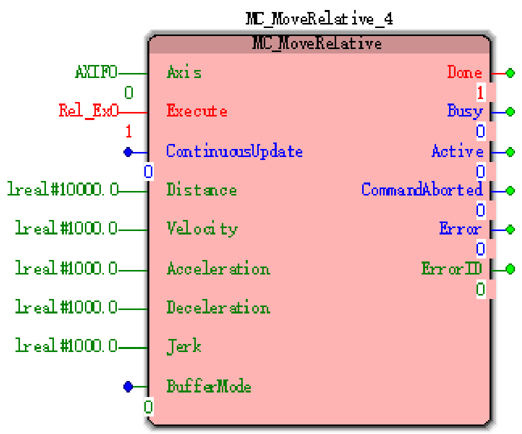
$Servo\ position\ feedback\ pulse$

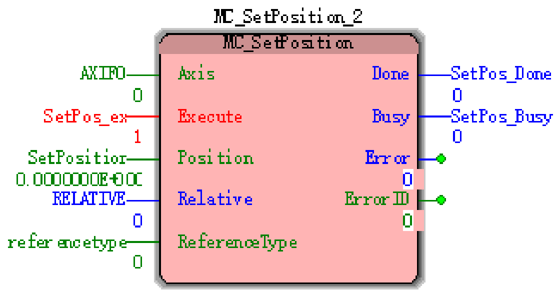
**Program Example**

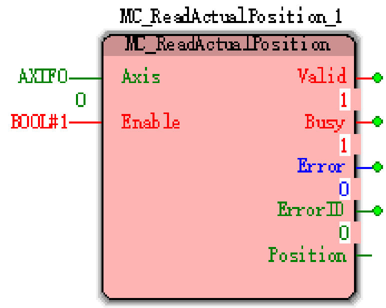
Effects of the present embodiment described MC_SetPosition MC_ReadActualPosition instruction command, the example procedure is as follows:

**1, variables, and procedures**

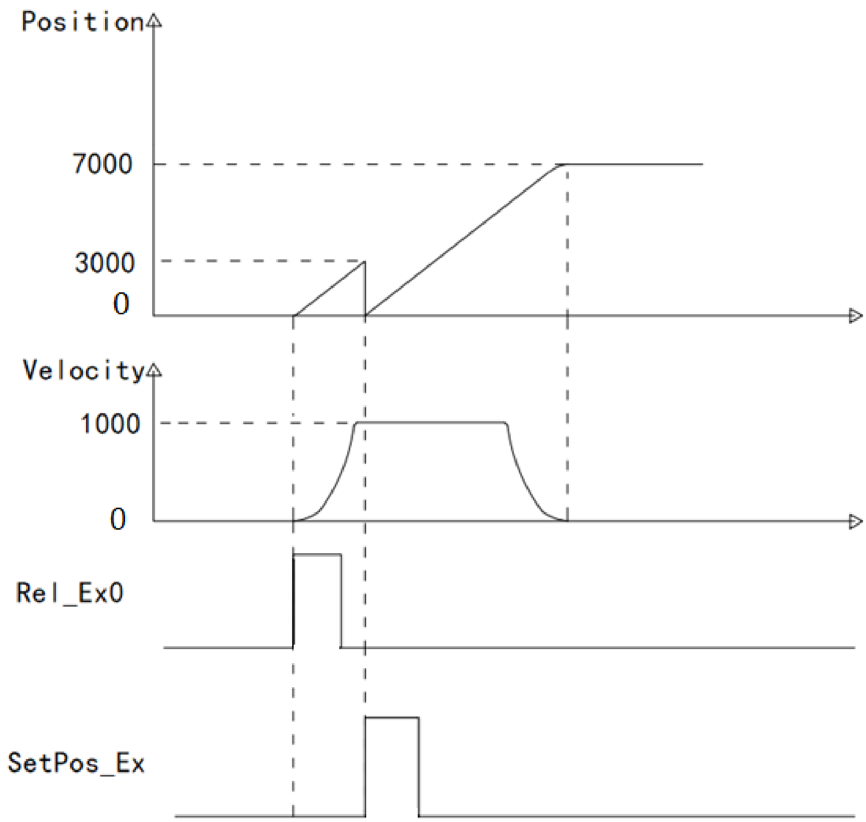
| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveRelative_4 | MC_MoveRelative | - |
| AXIF0 | USINT | 0 |
| Rel_Ex0 | BOOL | - |
| MC_SetPosition_2 | MC_SetPosition | - |
| SetPos_ex | BOOL | - |
| SetPosition | LREAL | 0.0 |
| RELATIVE | BOOL | FALSE |
| referencetype | BOOL | FALSE |
| Read_P0 | LREAL | - |






## 2. Motion curve and timing diagram

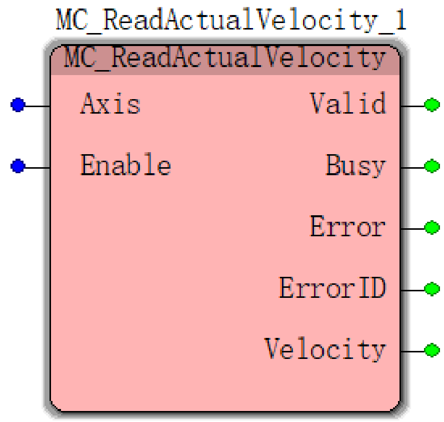



- MC_ReadActualPosition execution instruction fetch real-time position, the MC_MoveRelative performed, charged at a set speed to the target shaft position 10000 movement.

- ActualPosition is 3000, MC_SetPosition instruction execution, since instructions MC_SetPosition selected as the absolute position, therefore, performed after the completion of ActualPosition = 0, ActualPosition MC_MoveRelativel instruction is complete when the 7,000.
- As can be seen by the speed command MC_SetPosition curve image above does not affect the movement is performed, but it reflects MC_RealActualPosition ActualPosition curve value read ActualPosition affected MC_SetPosition instructions.

## 11.4.13 MC_ReadActualVelocity (read real-time speed)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the actual speed of the shaft | VEC-VA-MP-005-MA |




➢ **Input parameters**

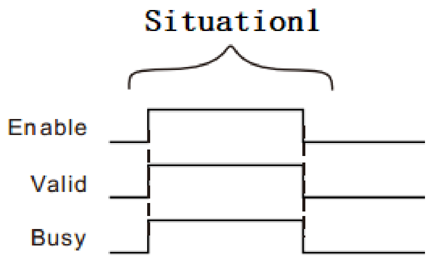
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable made to TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction execution | BOOL | TRUE or FALSE | - |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Valid (output valid) | The parameter output outputs a valid instruction is TRUE | BOOL | TRUE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| | the faulting instruction when the output instruction is TRUE | | |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| Velocity (actual speed) | Axis actual speed Unit: unit / S | REAL | Real |

➢ **FIG output change timing**




**Case 1:**When Enable the FALSE to TRUE, Valid and Busy simultaneously become TRUE ,. When Enable becomes FALSE, Valid, Busy all become FALSE.

➢ **Function Description**

Speed of the moving speed of the read command Velocity actuator terminal in units of cells / S, converted to revolutions per minute of the motor:

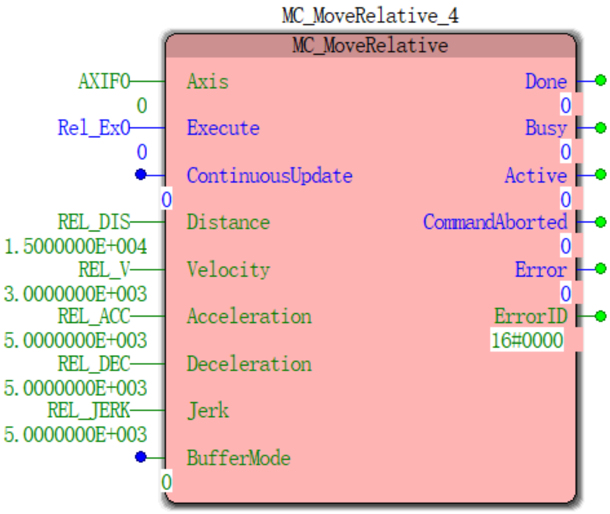
$$r / min = \frac{\text{Velocity}}{(\text{Screw\_Lead 或 Disc\_Circumference}) * \frac{Reductor\_Num}{Reductor\_Den}} * 60$$

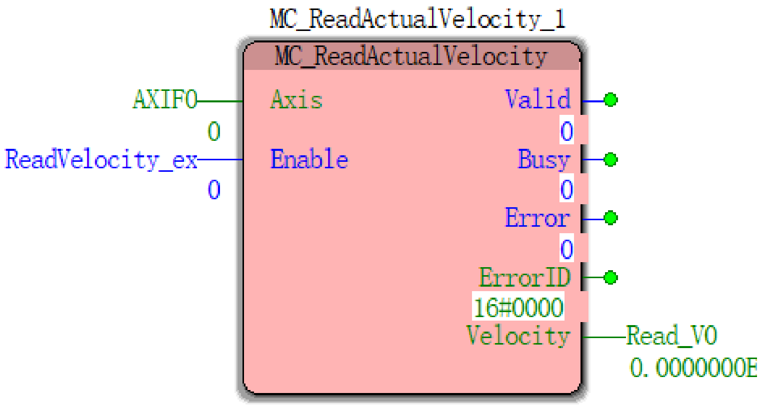
**Program Example**

Performing MC_MoveRelative, MC_ReadActualVelocity controlled axis realtime speed is read.

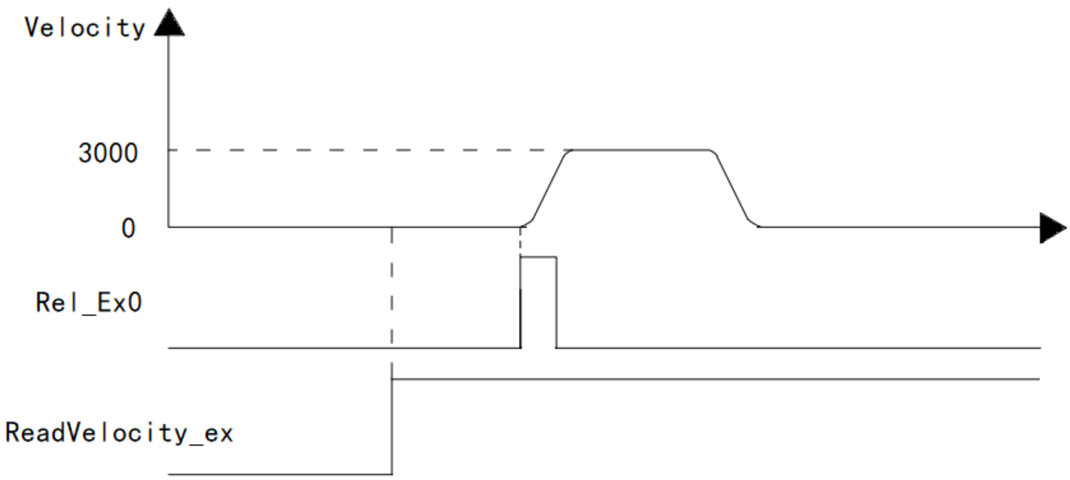
**1、 Variables and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveRelative_4 | MC_MoveRelative | - |
| AXIF0 | USINT | 0 |
| Rel_Ex0 | BOOL | FALSE |
| REL_DIS | LREAL | 15000.0 |
| REL_V | LREAL | 3000.0 |
| REL_ACC | LREAL | 5000.0 |
| REL_DEC | LREAL | 5000.0 |
| REL_JERK | LREAL | 5000.0 |
| MC_ReadActualVelocity_1 | MC_ReadActualVelocity | - |
| ReadVelocity_ex | BOOL | FALSE |
| Read_V0 | LREAL | - |



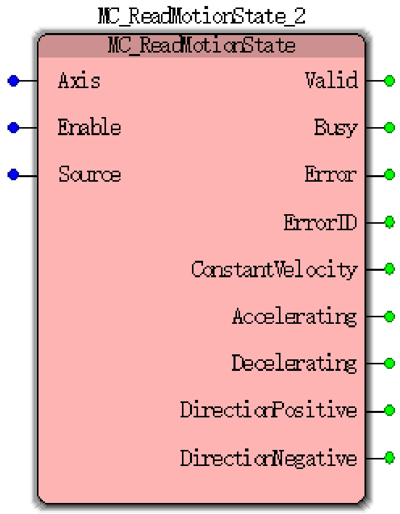




**2、 Timing and motion profiles of FIG.**



## 11.4.14 MC_ReadMotionState (read axis motion command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the state of motion controlled axes | VEC-VA-MP-005-MA |




- **Input parameters**

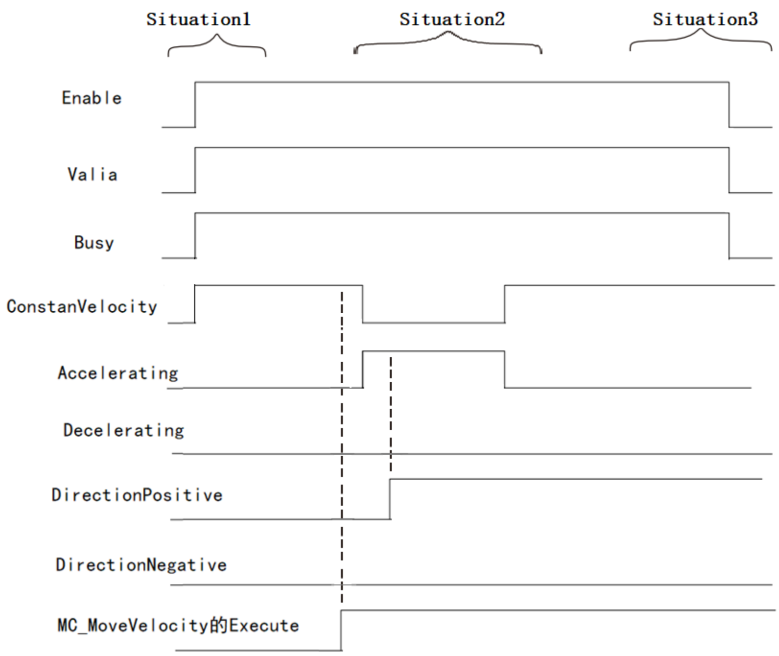
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction execution | BOOL | TRUE or FALSE | - |
| Source (Reserved) | Retention | INT | - | - |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Valid (output valid) | The parameter output outputs a valid instruction is TRUE | BOOL | TRUE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| ConstanVelocity (Uniform state) | The output parameter is TRUE axis represents doing uniform motion | BOOL | TRUE or FALSE |
| Accelerating (Acceleration state) | This parameter indicates the output shaft speed increases the absolute value is TRUE | BOOL | TRUE or FALSE |
| Declerating (Decelerating state) | This parameter indicates the output shaft speed to reduce the absolute value is TRUE | BOOL | TRUE or FALSE |
| DirectionPositive (Forward axis) | This parameter indicates the output shaft is increased when the current position is TRUE | BOOL | TRUE or FALSE |
| DirectionNegative (Axis inversion) | The output parameter is TRUE indicates the current position of the axis is reduced | BOOL | TRUE or FALSE |

- **FIG timing variation output parameter**




Case 1: In the controlled axis is Standstill state, when the Enable from FALSE to TRUE, and Busy vaild simultaneously become TRUE, ConstantVelocity, Accelerating, Decelerating, DirectionPositive, according to the output pin axis DirectionNegative state to TRUE or FALSE.

Case 2: the instruction execution speed, the controlled-axis accelerometer, Accelerating output is TRUE, ConstantVelocity output becomes FALSE, when the current position of the controlled axis is increased DirectionPositive output to TRUE.

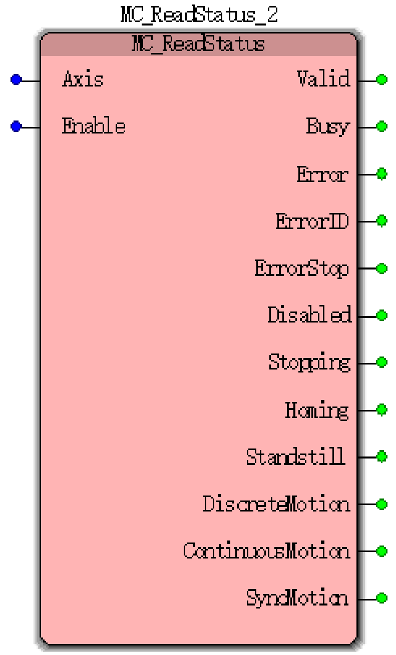
Case 3: When the Enable TRUE to FALSE, Vaild Busy and simultaneously become FALSE, ConstantVelocity, Accelerating, Decelerating, DirectionPositive, DirectionNegative output pin state remains unchanged when the Enable TRUE.

- **Function Description**

This instruction is used to read the current state of motion of the servo axis. Servo axis motion comprises: uniform motion, acceleration or deceleration motion, and the forward or reverse.

## 11.4.15 MC_ReadStatus (Read axis state)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the state information of the controlled axes | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction execution | BOOL | TRUE or FALSE | - |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|

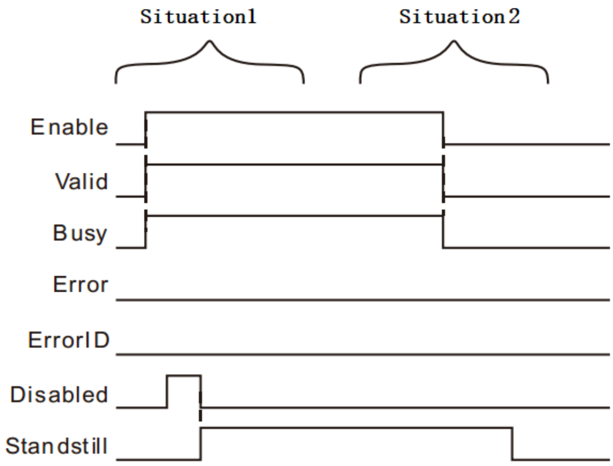
| | | | |
|---|---|---|---|
| Valid (output valid) | The parameter output outputs a valid instruction is TRUE | BOOL | TRUE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| ErrorStop (Abnormal stop) | Refer to the state machine described11.3.6 state machine | BOOL | TRUE or FALSE |
| Disabled (Not performed) | | BOOL | TRUE or FALSE |
| Stopping (Normal stop) | | BOOL | TRUE or FALSE |
| Homing (OPR) | | BOOL | TRUE or FALSE |
| Standstill (Ready to execute) | | BOOL | TRUE or FALSE |
| DiscreteMotion (Discrete motion) | | BOOL | TRUE or FALSE |
| ContinousMotion (Continuous motion) | | BOOL | TRUE or FALSE |
| SyncMotion (Synchronous motion) | | BOOL | TRUE or FALSE |

**Description:**

1、 The instruction is executed when the Enable is TRUE, status read axes.

2, when the instruction by the Enable TRUE to FALSE, Vaild, Busy becomes FALSE, the output ErrorStop, Disabled, Stopping, Homing, Standstill, DiscreteMotion, ContinuousMotion Enable and SyncMotion remains unchanged state of TRUE.

- **FIG output timing parameters**




Case 1: When the Enable FALSE to TRUE, Vaild Busy and simultaneously become TRUE, ErrorStop, Disabled, Stopping, Homing, Standstill, DiscreteMotion, ContinuousMotion The shaft and SyncMotion state to TRUE or FALSE.

Case 2: When the Enable TRUE to FALSE, Vaild Busy and simultaneously become FALSE, the output Disabled, Stopping, Homing, Standstill, DiscreteMotion, ContinuousMotion and SyncMotion Enable pin remains unchanged state of TRUE.

➢ **Function Description**

This instruction is used to read the state information of the controlled axis of the shaft, a detailed description about an axis refer to the status of the state machine described

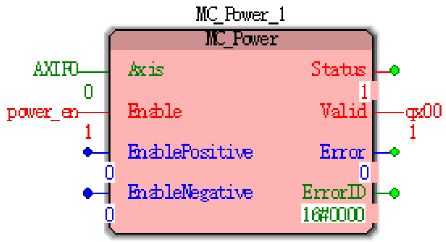
**Program Example**

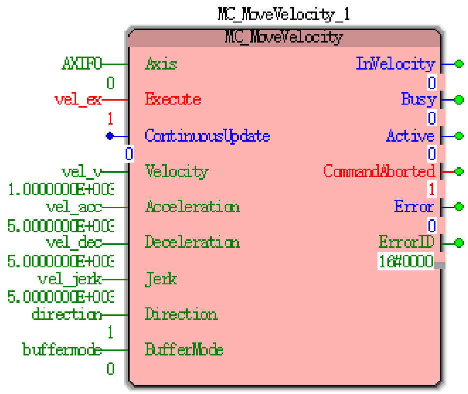
MC_ReadStatus instruction execution example as follows:

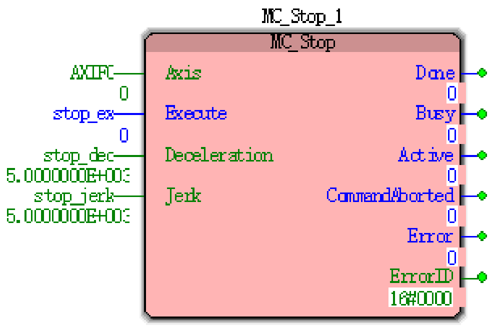
**1、 Variables and procedures**

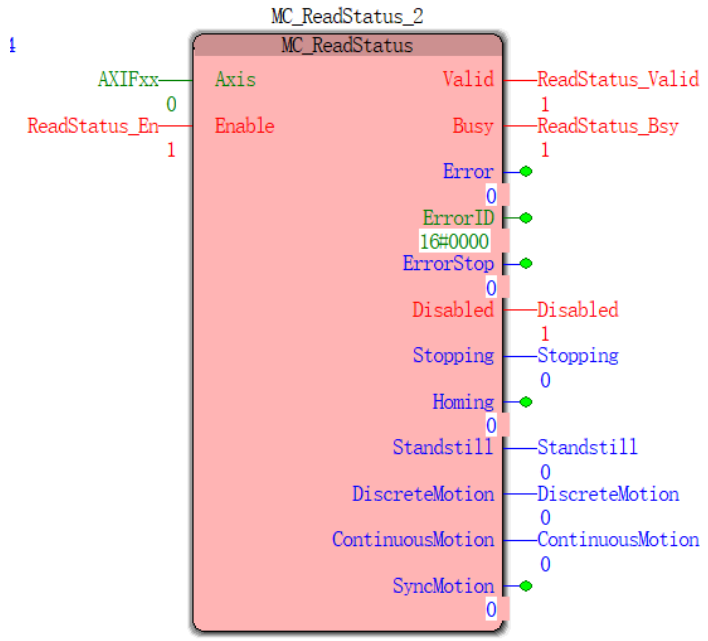
| variable name | type of data | The initial value |
|---|---|---|
| MC_Power_1 | MC_Power | - |
| AXIF0 | USINT | 0 |
| Power_en | BOOL | FALSE |
| MC_MoveVelocity_1 | MC_MoveVelocity | - |
| Vel_ex | BOOL | FALSE |
| Vel_v | LREAL | 1000.0 |
| Vel_acc | LREAL | 5000.0 |
| Vel_dec | LREAL | 5000.0 |
| Vel_jerk | LREAL | 5000.0 |
| direction | INT | 1 |
| buffermode | INT | 0 |
| MC_Stop_1 | MC_Stop | - |
| stop_ex | BOOL | FALSE |
| stop_dec | LREAL | 5000.0 |
| stop_jerk | LREAL | 5000.0 |

| | | |
|---|---|---|
| MC_ReadStatus_2 | MC_ReadStatus | - |
| AXIFxx | USINT | 0 |
| ReadStatus_En | BOOL | FALSE |
| ReadStatus_Valid | BOOL | FALSE |
| ReadStatus_Bsy | BOOL | FALSE |
| Disabled | BOOL | FALSE |
| Stopping | BOOL | FALSE |
| Standstill | BOOL | FALSE |
| DiscreteMotion | BOOL | FALSE |
| ContinuousMotion | BOOL | FALSE |

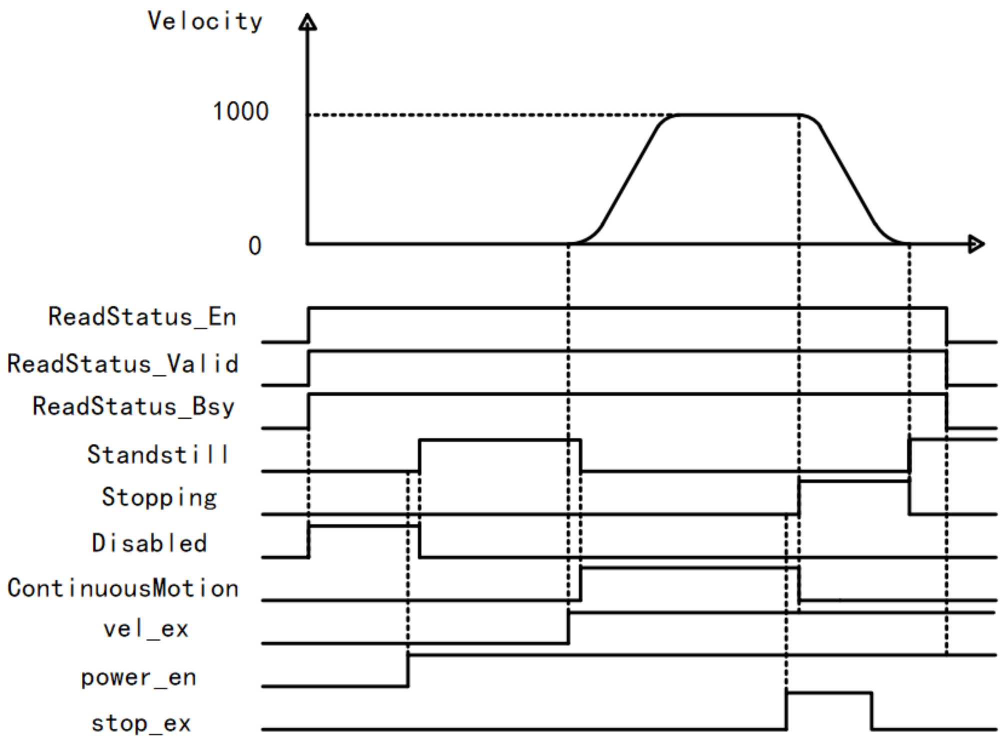





**2, the motion profile and timing**

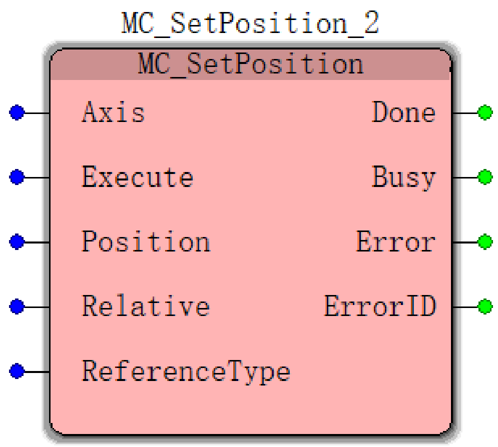



- Enable MC_Power from FALSE to TRUE instruction, the latter instruction cycle Disabled MC_ReadStatus by TRUE to FALSE, while Standstill MC_ReadStatus instruction from FALSE to TRUE (i.e., the state machine changes from the Disabled Standstill)
- Execute the speed command from FALSE to TRUE, after a period MC_ReadStatus Standstill instruction from TRUE to FALSE, ContinuousMotion while MC_ReadStatus instruction from FALSE to TRUE (i.e., the state machine changes from the Standstill ContinuousMotion)

- Execute the instruction MC_Stop FALSE to TRUE, ContinuousMotion after a period MC_ReadStatus instruction from TRUE to FALSE, Stopping while MC_ReadStatus instruction from FALSE to TRUE (i.e., the state machine changes from the ContinuousMotion Stopping)

## 11.4.16 MC_SetPosition (position setting instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to set the position value of the axis to a predetermined value, and does not cause the shaft to produce the actual motion. | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Execute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| Position (position) | Set the target position (Unit: unit) | LREAL | Positive, negative, zero (0) | Execute from FALSE to TRUE |
| Relative (Relative mode) | Set a target position and the current position of relative mode or | BOOL TRUE: Relative Mode FALSE: Absolute | TRUE or FALSE (FALSE) | Execute from FALSE to TRUE |

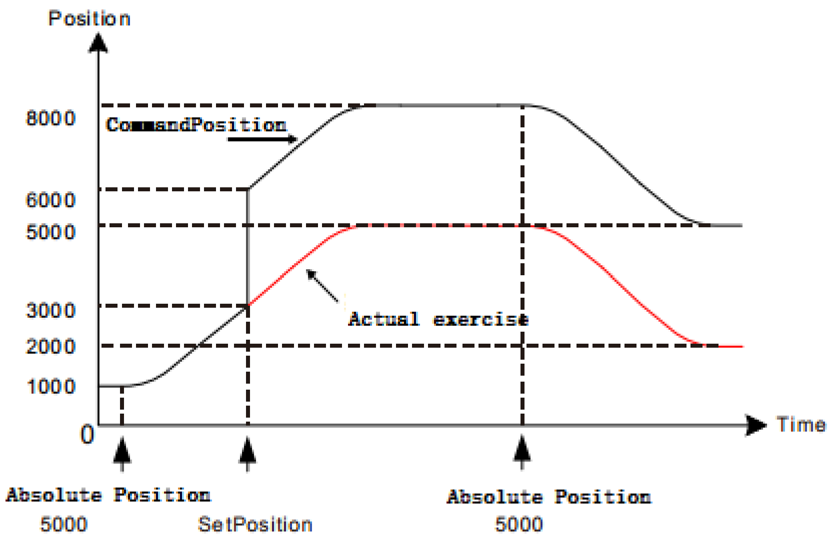
| | | | | |
|---|---|---|---|---|
| | absolute mode | Mode | | |
| ReferenceType (Type reference position) | Reference position set type | INT 0: command position 1: The actual position | 0 (0) | Execute from FALSE to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

● **Function Description**

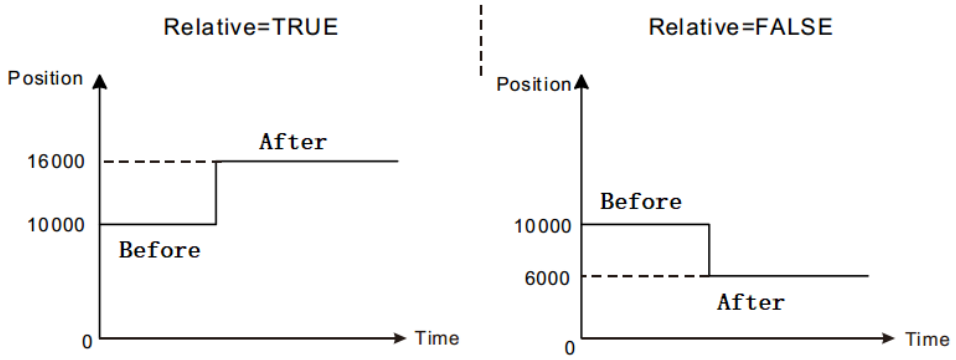
This instruction is used to set the position value of the axis to a predetermined value, and does not cause the shaft to produce the actual motion. The implementation of this directive will not have a real impact on the movement in progress, but this instruction is executed instruction to complete before you begin the actual implementation of the results of an impact, as shown below:




● **Position and Relative**

Input parameters Position, Relative axial position of the instruction execution start time (herein used to mean "reference position") together determine the position of the value to be set.

Relative to the input parameters define the relationship between the input parameter Position reference position. When Relative = TRUE, Position relative relationship with the reference position, the reference position setting position = + Position; when Relative = FALSE, Position absolute relationship with the reference position, the position setting value = Position. As shown below, the reference position instruction execution 10000, Position input parameter value 6000, when the input parameters Relative to a different value, corresponding to the implementation of the results were lower left and bottom right.




- **ReferenceType**

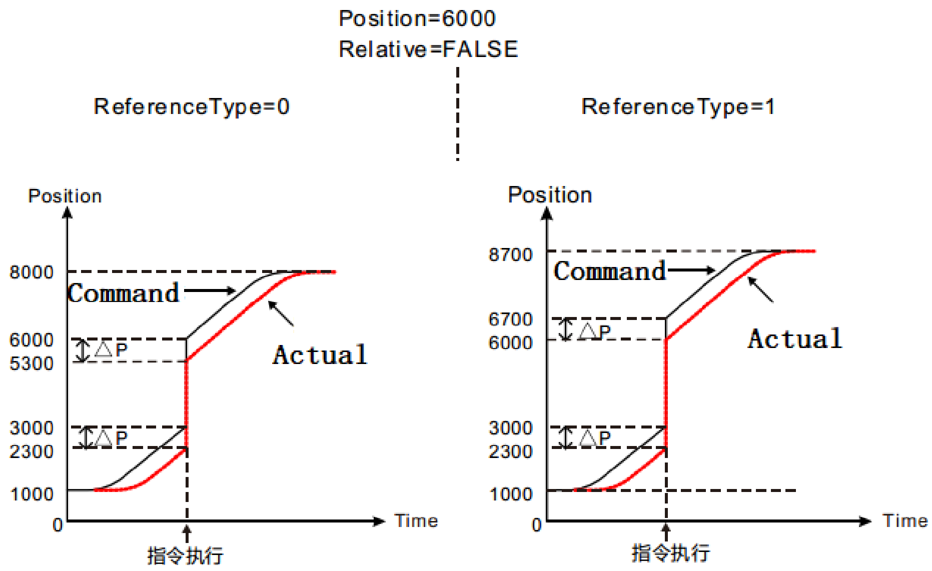
ReferenceType input parameters for the command for selecting the reference position or actual position. When ReferenceType = 0, the reference position for the axis command position; when ReferenceType = 1, the reference position for the actual position of the shaft.

When selected as the reference position of the position command, the instruction command is calculated based on the current position and the Position of the target position command, the position and the modification value for the target value of the position command; while the actual position of the shaft will also change , variation is: change in the amount equal to the change in position command and the actual position, that is, the difference between the command position and the actual position in the instruction execution time and the start time of the instruction execution is completed unchanged.

Mode selecting process when the reference position is the actual position and the reference position when the mode selection process for the same reason commanded position.

If the instruction execution MC_SetPosition, the axes are stationary, the reference position are performed to select the actual effect of the command position and the actual position is no difference, because the shaft is stationary, there is no difference (difference between the command position and the actual position 0);

If the instruction execution MC_SetPosition, the axis is in motion, there is a difference between the command position and the actual position (the difference is not 0, the response time caused by the command), the reference position are executed when the actual selection command position and the actual position MC_SetPosition instruction execution (absolute mode, position = 6000) when the curve as shown, for positioning the shaft is moving (target position 5000), then the command position and the actual position of the shaft: differences in effect, as the example shown in FIG. 3000 and 2300, respectively (difference △ P = 700). If the reference position selection command position, after executing the instructions, commands the shaft position becomes 6000, the actual position becomes 5300 (5300 = 6000- △ P), shown at bottom left; if the actual position of the reference position selection, the command after the execution, the actual position of the shaft 6000 is changed, the command position becomes 6700 (6700 = 6000 + △ P), as shown below right:

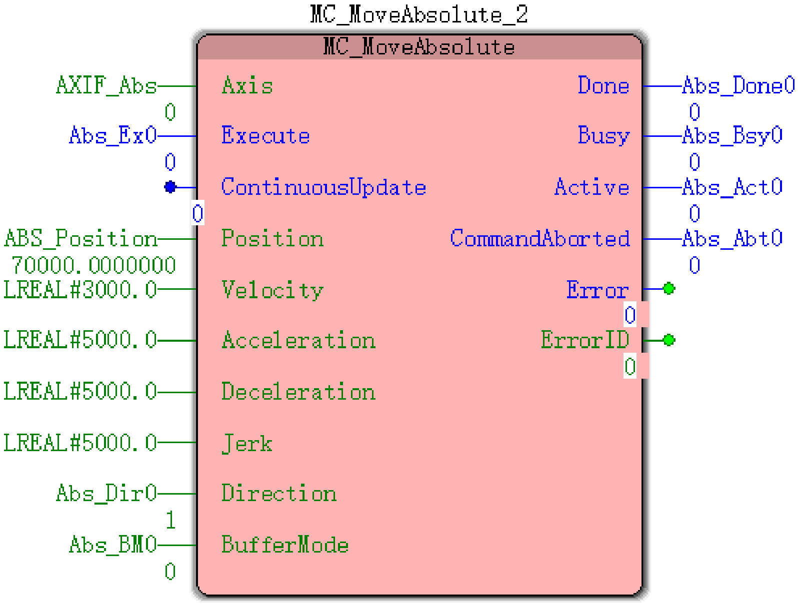


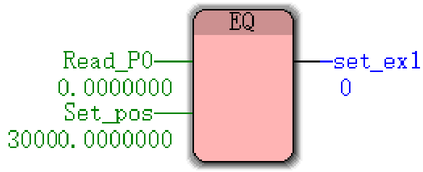

**Program Example**

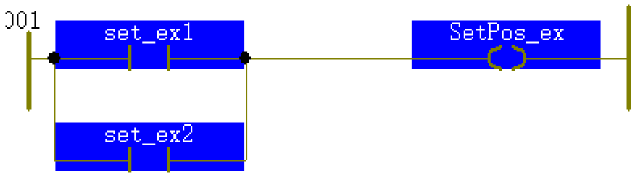
Effects This example describes the effect of execution of the instruction MC_SetPosition MC_MoveAbsolute executed being executed: no effect on the actual implementation of the results MC_SetPosition MC_MoveAbsolute instruction being executed.

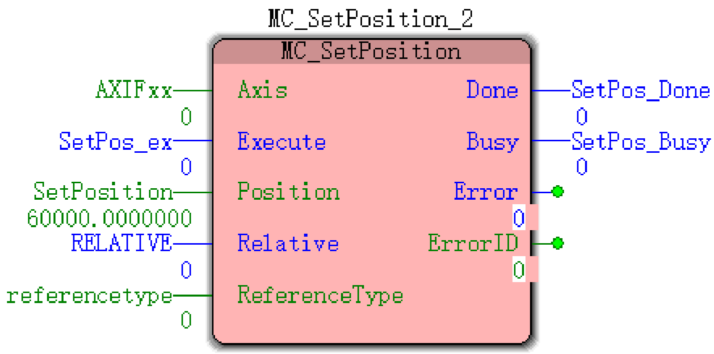
**2、 Variables and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveAbsolute_2 | MC_MoveAbsolute | - |
| AXIF_abs | USINT | 0 |
| Abs_Ex0 | BOOL | FALSE |
| Abs_Position | LREAL | 70,000 |
| Abs_BM0 | INT | |
| Abs_Done0 | BOOL | |
| Abs_Bsy0 | BOOL | |
| Abs_Act0 | BOOL | |
| Abs_Abt0 | BOOL | |
| MC_SetPosition_2 | MC_SetPosition | |
| AXIFxx | USINT | 0 |
| SetPos_ex | BOOL | FALSE |
| SetPosition | LREAL | 60,000 |
| RELATIVE | BOOL | 0 |
| referencetype | INT | |
| SetPos_Done | BOOL | |
| SetPos_Busy | BOOL | |

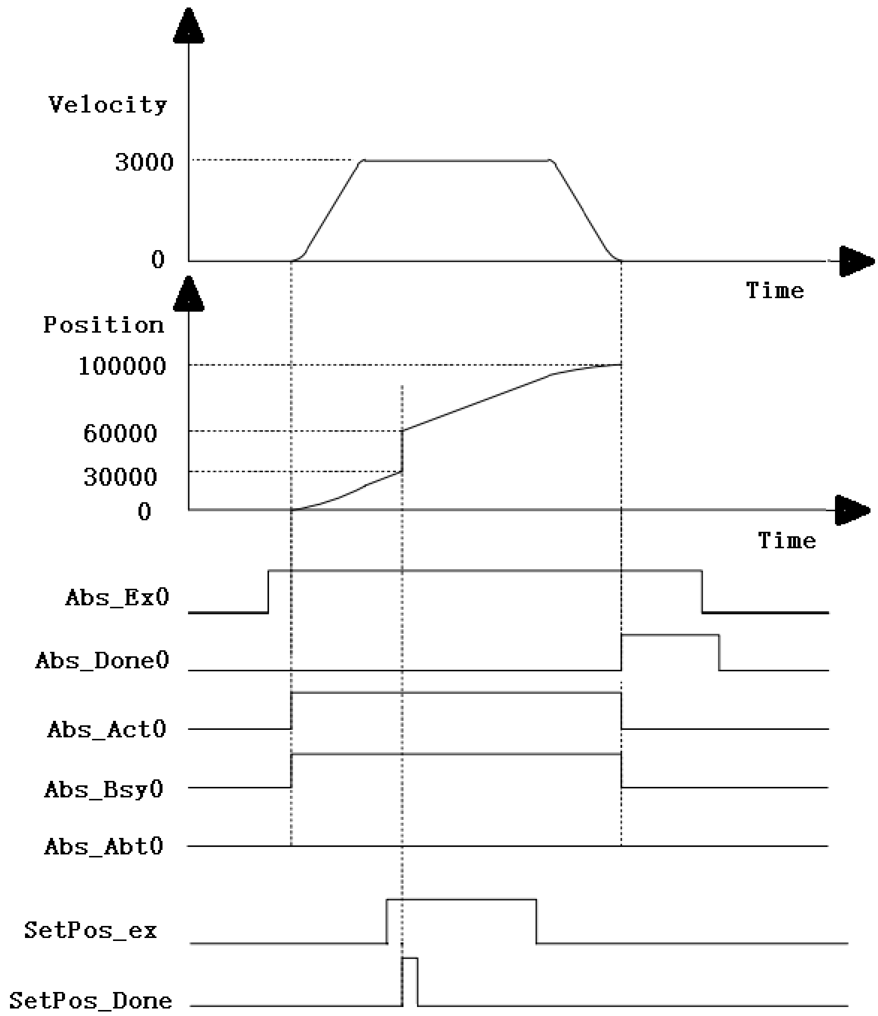


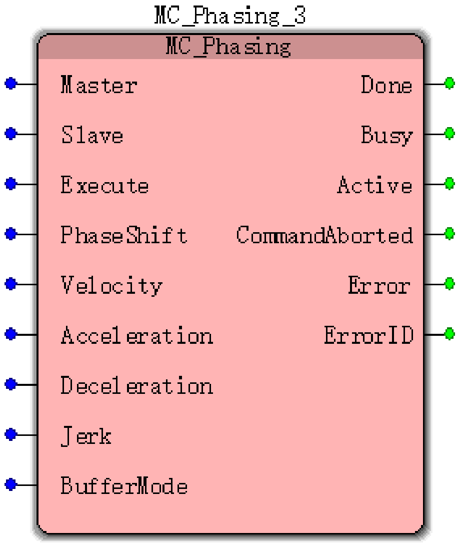

**2, the motion profile and timing**




- When Abs_Ex0 a FALSE to TRUE, MC_MoveAbsolute instruction starts execution, MC_SetPosition instruction execution when the current position is greater than 30,000.
- MC_SetPosition command position when the instruction to start execution of 30,000, a command execution completion position which is 60,000, when the position of the instruction execution is completed MC_MoveAbsolute 100,000.
- Of speed variations can be seen in the figure above: MC_SetPosition instruction does not affect the practical implementation of the results of executing MC_MoveAbsolute

## 11.4.17 MC_Phasing (shift spindle command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to position the additional main section shift does not affect the movement of the spindle | VEC-VA-MP-005-MA |




➢ **Input parameters**

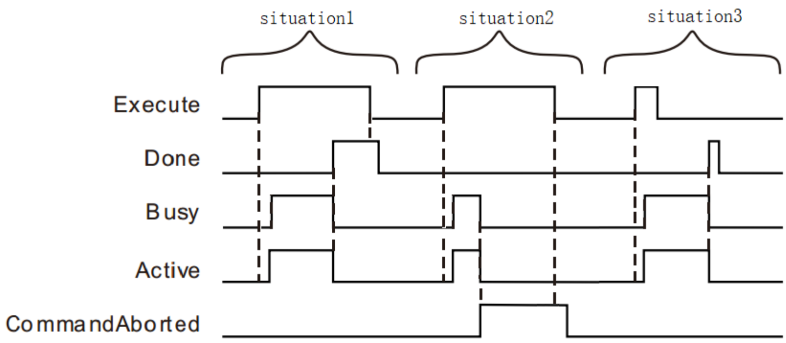
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Master (spindle) | Setting instruction to be controlled spindle | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Slave (Slave axis) | Setting instruction from the shaft to be controlled | USINT | 0-4 the real axis 5 to 11 virtual axis (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | is executed | | | |
| PhaseShift (Offset) | Setting spindle position offset (Unit: unit) | LREAL | Positive, negative, zero (0) | Exexcute from FALSE to TRUE |
| Velocity (speed) | Spindle speed offset is set | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Acceleration (Acceleration) | Spindle offset is set acceleration | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Decleration (decrease speed) | Spindle offset is set deceleration | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Jerk (Plus / deceleration rate of change) | Spindle offset is set the rate of change of acceleration / deceleration | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Retention | - | - | - |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;
After completion of the offset spindle, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.

**Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

**Case 3**: After during instruction execution, Execute a TRUE to FALSE, when the instructions are executed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

➢ **Function Description**

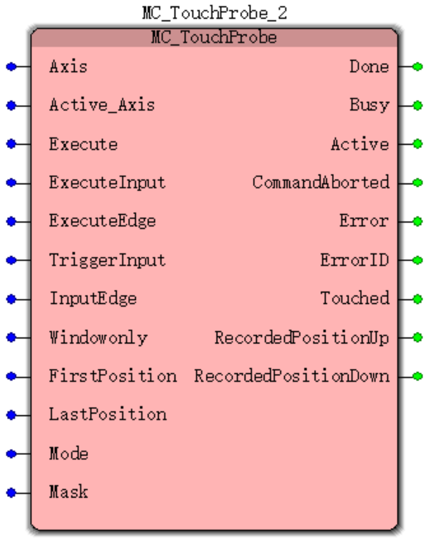
- This command is used to overlay a virtual displacement of the spindle motion by some set distance, velocity, acceleration / deceleration of the rate of change of deceleration, it does not affect the actual movement of the spindle, the spindle acquired from the shaft to the physical location will based on the offset, resulting in a position offset Slave axis of the slave follower.

- MC_PhasingPolyaxial instructions may act as follows:

| | |
|---|---|
| MC_GearIn (electronic gear) | NS_MC_RotaryCutIn (peeling instruction) |
| MC_CamIn (electronic cam) | NS_MC_SpecialCAmin (special cam) |
| NS_MC_SpecialCombineAxes (Special two-spindle coupling) | |

## 11.4.18 MC_TouchProbe (position capture command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This position of the capture command for the shaft | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (Axis No.) | In mode 0, 1 and high-speed counter for pairing, 3,4 mode as the count number of the shaft | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Execute from FALSE to TRUE |
| Active_Axis (Hardware axis number) | Set position to capture the source hardware axis number | USINT | 0-4 the real axis | Execute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FASLE becomes TRUE, the instruction is executed. | BOOL | TRUE or FALSE | |
| ExecuteInput (Trigger enable bit) | Inputs I0 ~ I7, I10 ~ I17 of a capture trigger bit position, the pin corresponding to the input | WORD | 0 ~ 15 | Execute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | value of 0 to 7 inputs I0 ~ I7,8 ~ 15 corresponding to the input point I10 ~ I17. The pin mode (Mode) is equal to 2 active. | | | |
| ExecuteEdge (Signal edge) | FALSE, the selection signal is a falling edge input DI, TRUE, the selection of a rising edge of the input signal DI; | BOOL | TRUE or FALSE | Execute from FALSE to TRUE |
| TriggerInput (Trigger bit) | Inputs I0 ~ I7, I10 ~ I17 of a capture trigger bit position, the pin corresponding to the input value of 0 to 7 inputs I0 ~ I7,8 ~ 15 corresponding to the input point I10 ~ I17. | WORD | 0 ~ 15 (0) | Execute from FALSE to TRUE |
| InputEdge (Signal edge) | Setting signal trigger edge<br>0: Falling<br>1: Rising | BOOL | TRUE or FALSE | Execute from FALSE to TRUE |
| Windowly | Retention | - | - | - |
| FirstPosition | Retention | - | - | - |
| LastPosition | Retention | - | - | - |
| Mode (mode) | Position capture mode selection, see below Mode Description | INT | 0-6 (0) | Execute from FALSE to TRUE |
| Mask | Retention | - | - | - |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed, the parameter is only valid in 0,3 Mode =. | BOOL | TRUE or FALSE |
| Busy (execution) | The output parameter is TRUE representing instructions is being executed. | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| Touched (Continuous mode loop count) | When the selected mode is latched position 2,4, every time the latch position, Touched plus 1, Toucheded0 ~ 3 cyclically. | INT | 0-3 |
| RecordedPositionUp (Rising latch position) | When InputEdge = TRUE, store the latched position thereto. Unit: pulse | LREAL | Positive, negative, 0 |
| RecordedPositionDown (Falling edge latches position) | When InputEdge = FALSE, store the position of the latch thereto. Unit: pulse | LREAL | Positive, negative, 0 |

- **Mode Description**

| mode | Position capture trigger |
|---|---|
| Mode = 0 | The single mode is a high-speed counter is latched position capture instruction executed in this mode:<br>①TriggerInput only effective in the first trigger.<br>②NS_CC_Counter needs a high count number of instructions used in conjunction. Active_Axis ineffective, requires input parameter Axis AXIF_no same input parameters NS_CC_Counter instruction. |
| Mode = 1 | The continuous mode is a high-speed counter is latched position capture instruction executed in this mode:<br>①Each trigger TriggerInput updated once a latched position.<br>②NS_CC_Counter needs a high count number of instructions used in conjunction. Active_Axis ineffective, requires input parameter Axis AXIF_no same input parameters NS_CC_Counter instruction. |
| Mode = 2 | The special mode is a high-speed counter latch continuously, perform location capturing order in this mode:<br>①Before each position latch trigger TriggerInput update, you need to trigger a ExecuteInput.<br>②NS_CC_Counter needs a high count number of instructions used in conjunction. In this mode, Active_Axis ineffective, requires input parameter Axis AXIF_no same input parameters NS_CC_Counter instruction. |
| Mode = 3 | This mode is a single encoder latch. In this mode, performs location capturing |

| | |
|---|---|
| | order:<br>①TriggerInput only effective in the first trigger.<br>②Axis number register latch axis position, Active_Axis axis actual pulse source encoder. |
| Mode = 4 | This mode is a continuous encoder latch. In this mode, performs location capturing order:<br>①Each trigger TriggerInput updated once a latched position.<br>②Axis number register latch axis position, Active_Axis axis actual pulse source encoder. |
| Mode = 5 | This mode is dedicated CANopen single latch mode, the DI signal with the servo external latch is used, the position of the capture instruction executed in this mode:<br>①TriggerInput only effective in the first trigger.<br>②Axis number register latch axis position, Active_Axis ineffective. |
| Mode = 6 | The continuous mode is CANopen dedicated latch mode, the DI signal with the servo external latch is used, the position of the capture instruction executed in this mode:<br>①Each trigger TriggerInput updated once a latched position.<br>②Axis number register latch axis position, Active_Axis ineffective. |

➢ **Function Description**

- Position capture command to capture a position (RecordedPositionUp / RecordedPositionDown) Is a servo encoder from shaft / spindleofFeedback pulses;
- This high-speed position capture command belonging to the instruction, the count of the underlying hardware, the scan cycle is not affected.
- In mode 0, the instruction requires complex NS_CC_Counter (High-Speed Counter) used, the position is captured servo encoder feedback value A / B pulse alone does not make sense.
- When you have finished using continuous latch mode, the need to replace re-trigger input conditions, (position capture interrupt instruction), exit the current continuous latch mode, can then trigger must first MC_AbortTrigger;
- In CANopen control mode, mode 5 and 6, the function of the special need to download the template probe CANopen Division I, in conjunction with the VEC CANopen servo drives used. Servo function comes probe, an external latch signal DI position information (encoder unit) when the changes, and then transmitted to the master station outputs.

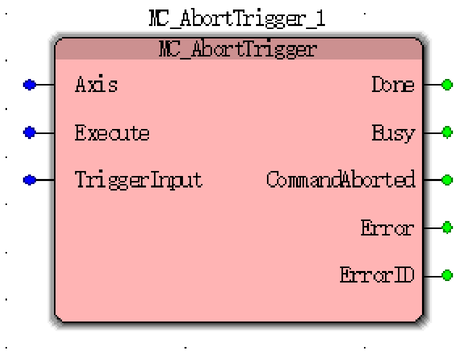
  ① VEC supports two probes simultaneously enabled, the position information can be recorded simultaneously rising and falling edges of the signal corresponding to each probe, while the latch 4 to the position information (rising and falling by switching InputEdge);

  ② 1 as a probe Probe Select signal DI8 probe 2 as a probe selected DI9 signal, and DI8 DI9 herein refers to the DI servo;

  ③ When selecting DI8 probe signal, the servo axis number Axis = node number 1; when DI9 signal probes, axis servo node number number Axis = +31;

### 11.4.19 MC_AbortTrigger (position capture interrupt instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to interrupt the position of the capture shaft | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (Axis No.) | The value is set to the same value MC_TouchProbe Axis command. Directed shaft for counting numbers need to be interrupted | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Execute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute from FALSE to TRUE, the instruction execution starts. | BOOL | TRUE or FALSE | - |
| Triggerinput | Retention | - | - | - |

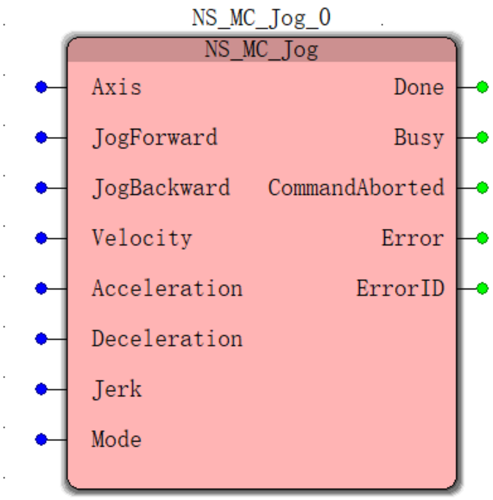
➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| | TRUE indicates instructions are executed | | |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

## 11.4.20 NS_MC_Jog (jog command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction can be used to move the function, also can be superimposed on the speed. | VEC-VA-MP-005-MA |




➢ **Input parameters**

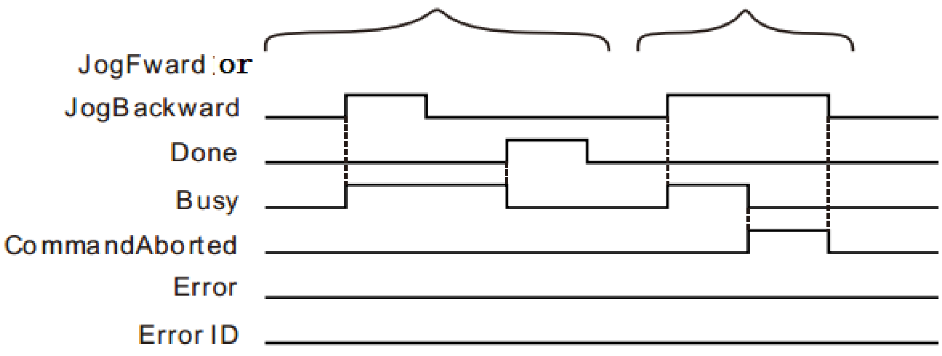
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| The Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | JogForward or JogBackward to TRUE |
| JogFoward (JOG) | When JogFoward changed from FALSE TRUE, the instruction is executed | BOOL | TRUE or FALSE | - |
| JogBackward (Jog Reverse) | When JogBackward changed from FALSE TRUE, the instruction is executed | BOOL | TRUE or FALSE | - |

| | | | | |
|---|---|---|---|---|
| Velocity (speed) | Set target speed (Unit: unit / S) | LREAL | A positive number (Non-default) | JogForward or JogBackward to TRUE |
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | A positive number (Non-default) | JogForward or JogBackward to TRUE |
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | A positive number (Non-default) | JogForward or JogBackward to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | A positive number (Non-default) | JogForward or JogBackward to TRUE |
| Mode (Mode) | Jog mode selection: 0: Jog jog speed the process of change, need to re-trigger Excute take effect; 1: Jog jog speed the process of change takes effect immediately | INT | 0 or 1 | JogForward or JogBackward to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1:**When JogForward or JogBackward a FALSE to TRUE, Busy becomes TRUE. When the movable stop point, the shaft speed is reduced to 0, Busy becomes FALSE, and at the same time maintaining a Done period becomes TRUE.

**Case 2:**When a JogBackward JogForward or FALSE to TRUE, the instruction is interrupted by other instruction,

CommandAborted becomes TRUE, the Busy becomes FALSE; or when JogForward JogBackward changed by TRUE FALSE, CommandAborted becomes FALSE.

➢ **Function Description**

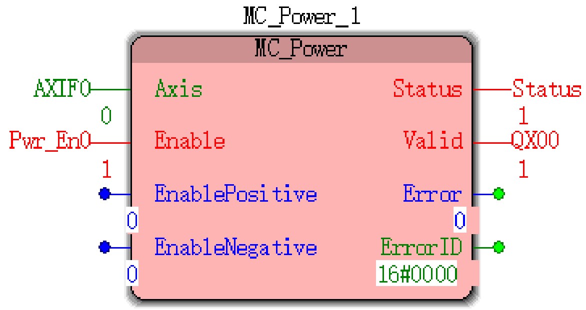
- This instruction is used to specify a given axis overlay jog speed, JogForward is TRUE controlled axis overlay a forward jog speed, JogBackward is TRUE controlled axis overlay a reverse jogging speed. When superposed jog speed reduced speed 0, Done after a period TRUE to FALSE
- This instruction does not affect the current state machine.

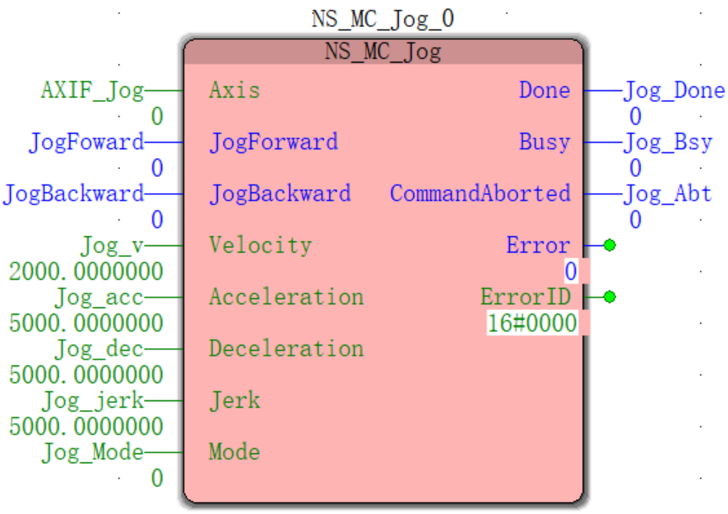
**Program Example**

When the instruction following examples NS_MC_Jog performed separately:

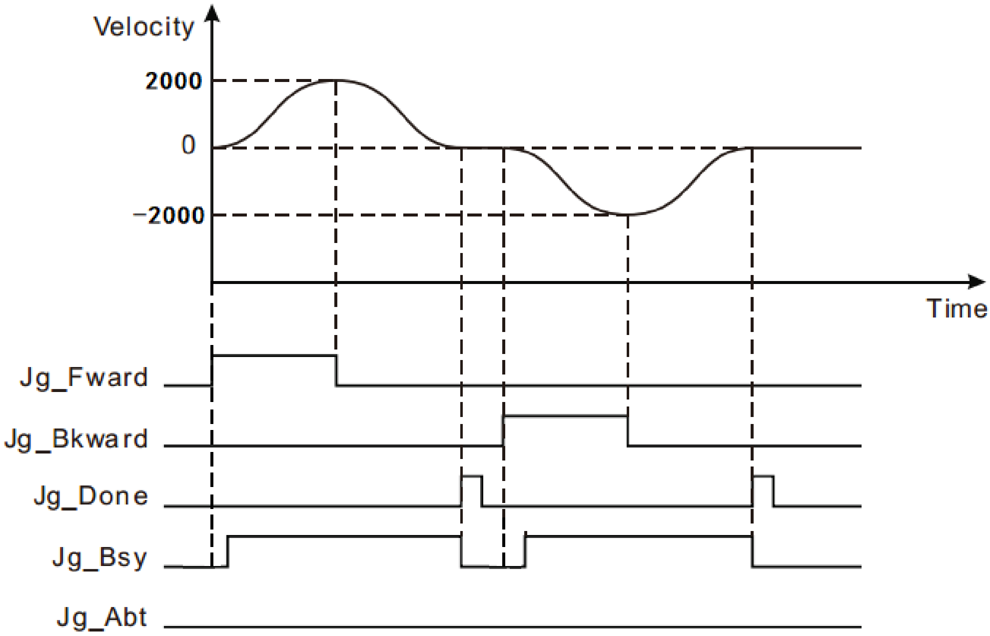
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_MC_Jog_2 | NS_MC_Jog | - |
| JogForward | BOOL | FALSE |
| JogBackward | BOOL | FALSE |
| Jog_v | LREAL | 2000.0 |
| Jog_acc | LREAL | 5000.0 |
| Jog_dec | LREAL | 5000.0 |
| Jog_jerk | LREAL | 5000.0 |
| Jog_Done | BOOL | FALSE |
| Jog_Bsy | BOOL | FALSE |
| Jog_Abt | BOOL | FALSE |




**2 . The motion profile and timing diagrams**

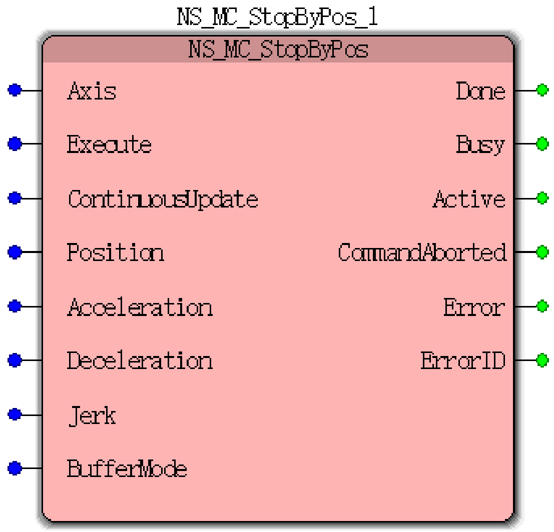



■ When the Jog_Forward NS_MC_Jog FALSE to TRUE to start execution instruction, after a period of Jog_Bsy FALSE to TRUE, the axis movement in positive direction; FALSE to TRUE to the Jog_Forward, the shaft begins to decelerate, when the deceleration is 0, Jog_Bsy becomes It is FALSE, while the rear Jog_Done a period from FALSE to TRUE becomes FALSE.

- When Jog_Backward a FALSE to TRUE, NS_MC_Jog instruction starts execution, after a period of Jog_Bsy FALSE to TRUE, start the reverse operation of the shaft; and a TRUE to FALSE Jog_Backward, the shaft begins to slow when the velocity is reduced to zero, Jog_Bsy after the bit is set to FALSE, FALSE to TRUE while the Jog_Done one cycle becomes FALSE.

# 11.4.21 NS_MC_StopByPos (position designated mode stop command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | It stopped at the specified position specified axis mode command for this operation | VEC-VA-MP-005-MA |




➢ **Input parameters**

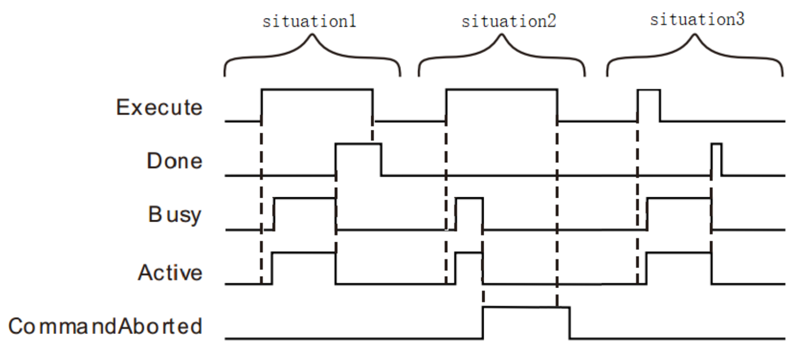
| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (execution position) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | - |
| Continous Update | Retention | | Retention | - |

| | | | | |
|---|---|---|---|---|
| Position (position) | Die set location | LREAL | 0≤Position <mold | Exexcute from FALSE to TRUE |
| Acceleration (Acceleration) | Goal setting acceleration (Unit: unit / S2) | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions<br>0: immediately interrupted<br>1: Wait | INT | A positive number (Non-default) | Exexcute from FALSE to TRUE |

➢ **Output parameters**

| **name** | **Features** | **type of data** | **Output range** |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG timing variation output parameter**

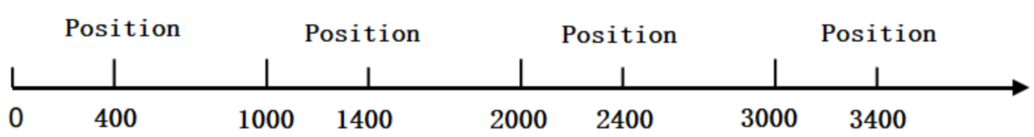



- **Case 1:** When the Execute FALSE to TRUE, after a period Buys Active and simultaneously become TRUE;
  When the position is reached, Done becomes TRUE, and the Busy Active becomes FALSE, it is the Execute TRUE to FALSE after a period, Done becomes FALSE.
- **Case 2**: When the Execute is TRUE, the instruction is interrupted after the other instructions, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.
- **Case 3**: After during instruction execution, Execute a TRUE to FALSE, when positioning is completed, Done becomes TRUE, and the Busy Active becomes FALSE, and after a period, Done becomes FALSE.

- **Function Description**

  - This instruction is used to specify the axes of the addition, the set deceleration, jerk is stopped at a specified position on the mold. Position mode is specified in a position, which value is less than the value of the parameter setting Modulo MC_AXIS_REF axis of the Execute instruction from FALSE to TRUE, the controlled axis according to the set acceleration / deceleration, the acceleration / rate of change of the position of the control shaft deceleration stop mode position position setting, the shaft finally stops is a whole multiple of + position of Moulo

As shown in, the following figure is a Modulo 1000, 400 Position, Position can be stopped at a specified position control mode by the command terminal of the actuator, the actuator may stop in the terminal unit 400, unit 1400, unit 2400, unit 3400 .......

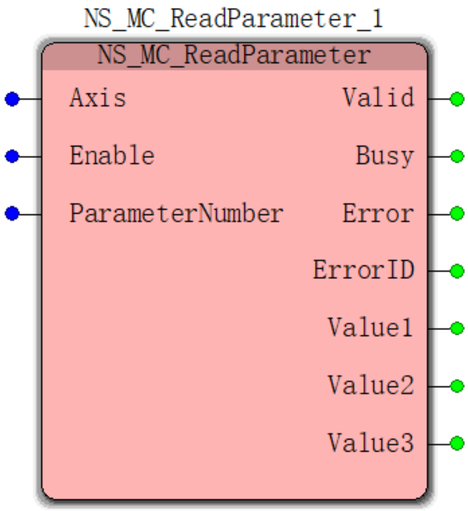



On the motion controller calculates the position of the real-time mode explained as follows:

Die = real-time position location of the terminal actuator% Module

## 11.4.22 NS_MC_ReadParameter (read command parameter)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the relevant parameters of the controlled axis | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| Axis (axis number) | Setting instruction to be controlled axes | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction execution | BOOL | TRUE or FALSE | |
| ParameterNumber (Monitoring function number) | Monitor the corresponding function of the control shaft No. value | INT | A positive number | Enable is TRUE |

➢ **Output parameters**

| name | Features | type of | Output range |
|---|---|---|---|

| | | data | |
|---|---|---|---|
| Valid | The parameter outputs a TRUE output instruction is valid | BOOL | TRUE or FALSE |
| Busy | The output parameter is TRUE Table Illustrates the instruction being executed | BOOL | TRUE or FALSE |
| Error | This parameter indicates the output command execution error to TRUE | BOOL | TRUE or FALSE |
| ErrorID | Instruction execution error code error. | WORD | - |
| Value1 | Function number corresponding to the monitored parameter | DINT | - |
| Value2 | | REAL | - |
| Value3 | | LREAL | - |

**Monitoring function parameter list**

| Function No. | Applicable function blocks | Monitoring description | The timing of the entry into force |
|---|---|---|---|
| 3 | Uniaxial / Multiaxial instruction | Feedback encoder position (unit: pulse) | Run command execution |
| 4 | Uniaxial / Multiaxial instruction | A given position (unit: pulse) | Run command execution |
| 5 | Uniaxial / Multiaxial instruction | The encoder position feedback setposition | Run command execution |
| 6 | Uniaxial / Multiaxial instruction | After a given position setposition | Run command execution |
| 7 | Uniaxial / Multiaxial instruction | Given the speed of planning theory (Unit: r / min) | Run command execution |
| 8 | Uniaxial / Multiaxial instruction | Real-time speed after the position compensation loop PI | Run command execution |
| 9 | Uniaxial / Multiaxial instruction | Real-time at a given speed (unit: r / min) | Run command execution |
| 10 | Uniaxial / Multiaxial instruction | Each incremental position underlying given period | Run command |

| | | | |
|---|---|---|---|
| | | (Unit: Pulse) | execution |
| 11 | Uniaxial / Multiaxial instruction | Real-time error (Unit: Pulse) | Run command execution |
| 12 | Uniaxial / Multiaxial instruction | Real-time analog output | Run command execution |
| 16 | Uniaxial / Multiaxial instruction | Given the current position of the mold | Run command execution |
| 17 | Uniaxial / Multiaxial instruction | Given the current pulses corresponding to the position of the mold | Run command execution |
| 18 | Uniaxial / Multiaxial instruction | The current position of the feedback mode | Run command execution |
| 19 | Uniaxial / Multiaxial instruction | Feedback current pulses corresponding to the position of the mold | Run command execution |
| 20 | Uniaxial / Multiaxial instruction | Sampling time to the number of pulses collected Sample_Time | Run command execution |
| 24 | Multiaxial instruction | Absolute encoder reads the absolute position of the lap | Run command execution |
| 40 | Multiaxial instruction | After filtering the speed of the spindle | Run command execution |
| 41 | Multiaxial instruction | The position of the spindle | Run command execution |
| 42 | Multiaxial instruction | After filtering speed of the second spindle | Run command execution |
| 43 | Multiaxial instruction | Position of the second spindle | Run command execution |
| 44 | Uniaxial / Multiaxial instruction | Directly read values of the encoder, the encoder for checking whether a wrong or reverse | Run command execution |
| 45 | Uniaxial / Multiaxial instruction | Direct imaginary axis encoder reading value | Run command execution |

| | | | |
|---|---|---|---|
| 46 | Uniaxial instruction | Reading module that Active_Axis MC_TouchProbe actual real time axis of the latch pulse control error (unit: pulse) | Run command execution |
| 47 | Uniaxial / Multiaxial instruction | Latching the shaft when the instant error trigger DI0 (Unit: Pulse) | Run command execution |
| 48 | Uniaxial / Multiaxial instruction | Latching the shaft when the instant error trigger DI1 (Unit: Pulse) | Run command execution |
| 49 | Uniaxial instruction | Reading module that Active_Axis MC_TouchProbe actual axle latch timing pulse in real time speed (unit: r / min) | Run command execution |
| 70 | Multiaxial instruction | Read MC_CamIn spindle position, the unit is a subscriber unit | Run command execution |

## 11.5 Multiaxial instruction

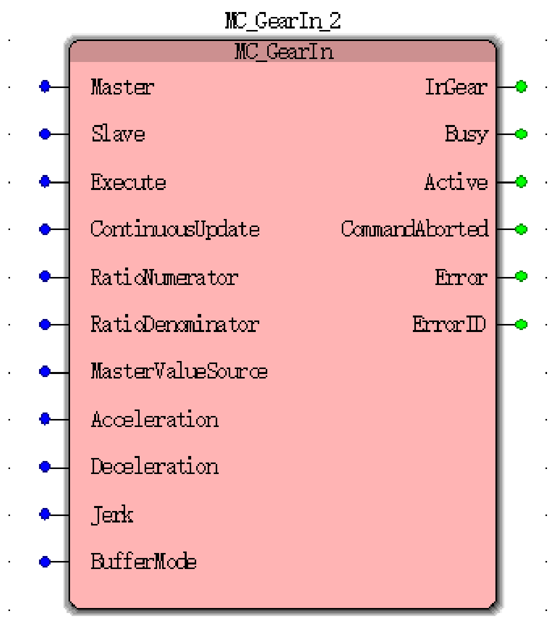
### MasterValueSource Description:

When MasterValueSource = 0, the following spindle axis from a position command.

When MasterValueSource = 1, follows from the actual position of the spindle axis, the actual position of a number of pulses detected by the hardware to the axis of the opening decision.

### 11.5.1 MC_GearIn (electronic gear coupling instructions)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This relationship established instructions for electronic gear shaft between two | VEC-VA-MP-005-MA |




➢ **Input parameters**

| Name | Features | Type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Master (Spindle) | Setting instruction to be controlled spindle | USINT | **Analog / Pulse**: 0-4 (real axis) | Exexcute from FALSE to |

| | | | | |
|---|---|---|---|---|
| | | | 5 to 11 (imaginary axis)<br>**CANopen mode**: 0-15 (real axis / imaginary axis)<br>(0) | TRUE |
| Slave<br>(Slave axis) | Setting instruction from the shaft to be controlled | USINT | **Analog / Pulse**:<br>0-4 (real axis)<br>5 to 11 (imaginary axis)<br>**CANopen mode**: 0-15 (real axis / imaginary axis)<br>(0) | Exexcute from FALSE to TRUE |
| Execute<br>(Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | - |
| ContinousUpdate | Retention | - | - | - |
| RatioNumerator<br>(Electronic gear molecule) | Molecular electronic gear | LREAL | Positive, negative,<br>(Non-default) | Exexcute from FALSE to TRUE |
| RatioDenominator<br>(Electronic gear denominator) | The denominator of electronic gear | LREAL | A positive number<br>(Non-default) | Exexcute from FALSE to TRUE |
| MasterValueSource<br>(Select location source) | Source selection command from the shaft<br>0: Follow the spindle axis from a position command<br>1: the actual position of the shaft from the spindle to follow | INT | 0 or 1 | Exexcute from FALSE to TRUE |
| Acceleration<br>(Acceleration) | Goal setting acceleration<br>(Unit: unit / S2) | LREAL | A positive number<br>(Non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| Deceleration (decrease speed) | Set target deceleration (Unit: unit / S2) | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | The rate of change of the target acceleration or deceleration setting (Unit: unit / S3) | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0: immediately interrupted 1: Wait | INT | 0: immediately interrupted 1: Wait (0) | Exexcute from FALSE to TRUE |

**Description:**

1. This instruction starts execution when the Execute FALSE to TRUE. Regardless of whether the instruction is executed, when the Execute FALSE to TRUE again, the instructions may be re-executed, the parameters can be revalidated The pin comprises RatioNumerator, RatioDenominator, MasterValueSource, Acceleration, Deceleration, Jerk, BufferMode, and outputs TRUE CommandAborted .

2. When the instruction is executed, execution of the instruction from the shaft as the other motion instructions can be interrupted MC_MoveVelocity this instruction, the spindle and gear will be released from the relationship between the axes. MC_Halt may be performed or stopped from MC_Stop axis.

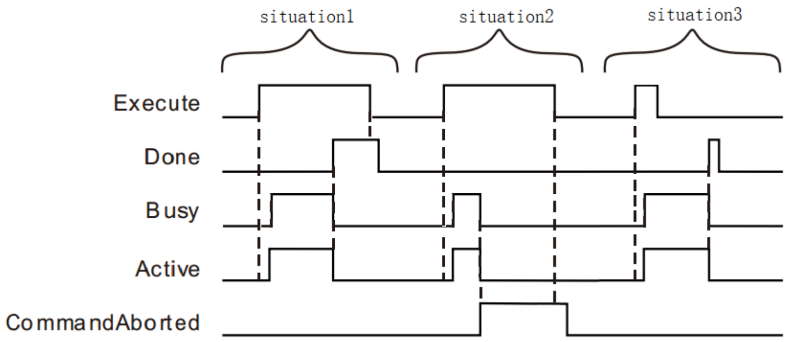
3, the instruction is followed by the pulse change spindle.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| InGear (synchronized state) | This parameter is TRUE output shaft from the synchronized state represents | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, and after a period, Busy, Active becomes TRUE. When the synchronous state has been reached, InGear becomes TRUE, while Busy Active and remains to TRUE.

**Case 2:** When the Execute is TRUE, the shaft is controlled from other instructions, the instruction is interrupted by another instruction, CommandAborted becomes TRUE, the Busy and Active to FALSE; Execute when a TRUE to FALSE, FALSE becomes a cycle after CommandAborted .

**Case 3:**During instruction execution, when the Execute TRUE to FALSE, InGear becomes TRUE, the Busy Active and remains to TRUE.

- **Function Description**

- This instruction is used to establish a relationship between two electronic gear shaft. After this instruction is executed, according to the electronic gear shaft molecules, electronic gear denominator, the source of the command, acceleration, deceleration, jerk, the transfer mode of operation and the spindle gear. Spindle axis may be real, imaginary axis or shaft encoder, the shaft may be a real axis or an imaginary axis.
- When this instruction is executed, the shaft need enabled state, the spindle enable or are the lower energy state.
- When two electronic gear shaft is not established relationship (i.e. InGear FALSE when the instruction), execute the instruction, designated according to the command from electronic gear shaft molecules, electronic gear denominator, deceleration, acceleration jerk reaches the target speed (real-time speed spindle).

①When the spindle real constant acceleration, deceleration Slave axis of the instruction set, the acceleration change rate reaches the target speed

②When the real-time change of the acceleration of the spindle, towards the target speed change shaft speed from the following equation.

$$\mathbf{Slave\ Acc(or\ Dec)} = \mathbf{Master\ Acc(or\ Dec)} * \frac{\boldsymbol{Electronic\ gear\ Numerator}}{\boldsymbol{Electronic\ gear\ Denominator}}$$

After establishing two-axis electronic gear relationship (InGear the instruction is TRUE), from the relationship with the electronic gear shaft speed molecule, the denominator of the electronic gear and spindle speed as follows:

$$\mathbf{Slave\ taget\ speed} = \mathbf{Master\ taget\ speed} * \frac{\boldsymbol{Electronic\ gear\ Numerator}}{\boldsymbol{Electronic\ gear\ Denominator}}$$

- Electronic gear ratio

$$\mathbf{Electronic\ gear\ ratio} = \frac{\boldsymbol{Electronic\ gear\ Numerator}}{\boldsymbol{Electronic\ gear\ Denominator}}$$

When the ratio is positive, the same direction of movement of electrons from the gear shaft and the spindle.

When the ratio is negative, and from the opposite direction of movement of the spindle shaft.

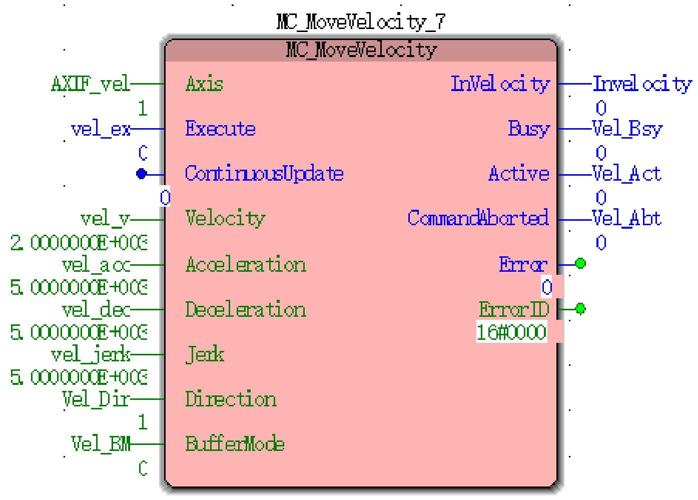
**Program Example**

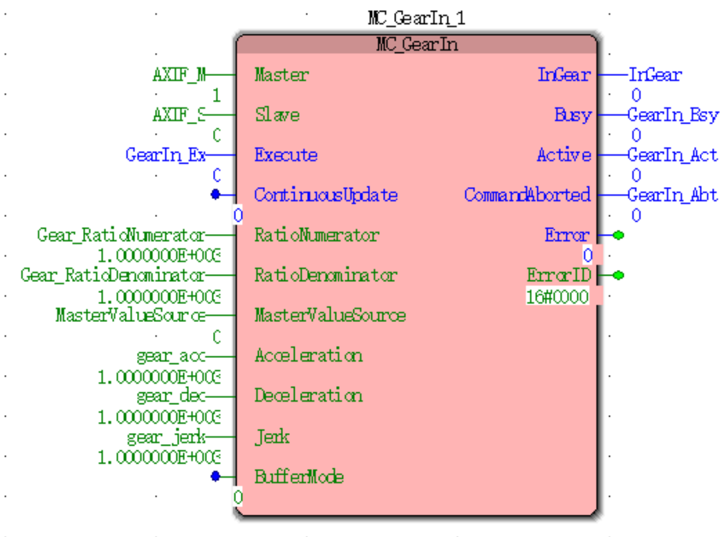
MC_GearIn program of instructions in the example below:

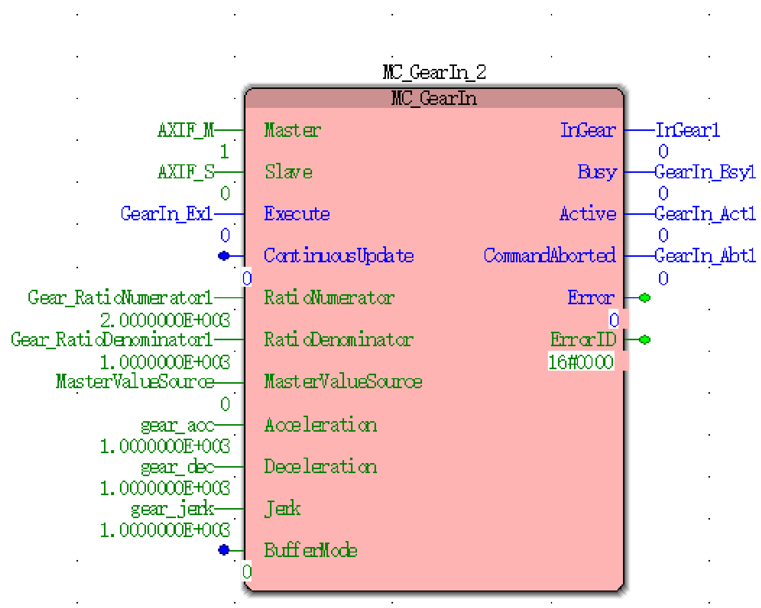
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity_7 | MC_MoveVelocity | |
| AXIF_vel | USINT | 1 |
| Vel_ex | BOOL | FALSE |
| Vel_v | LREAL | 2000.0 |
| Vel_acc | LREAL | 5000.0 |
| Vel_dec | LREAL | 5000.0 |
| Vel_jerk | LREAL | 5000.0 |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| Invelocity | BOOL | |
| Vel_Bsy | BOOL | |
| Vel_Act | BOOL | |
| Vel_Abt | BOOL | |
| MC_GearIn_1 | MC_GearIn | |
| AXIF_M | USINT | 1 |
| AXIF_S | USINT | 0 |
| GearIn_Ex | BOOL | FALSE |
| Gear_RatioNumerator | LREAL | 1000.0 |
| Gear_RatioDenominator | LREAL | 1000.0 |
| MasterValueSource | INT | 0 |
| Gear_acc | LREAL | 1000.0 |
| Gear_dec | LREAL | 1000.0 |

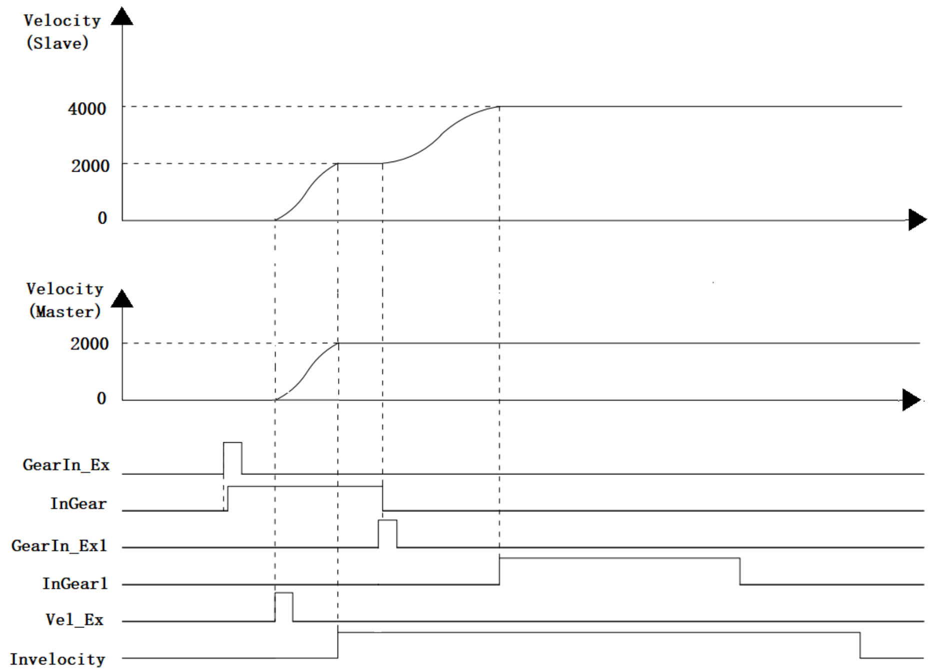
| | | |
|---|---|---|
| Gear_Jerk | LREAL | 1000.0 |
| InGear | BOOL | |
| GearIn_Bsy | BOOL | |
| GearIn_Act | BOOL | |
| GearIn_Abt | BOOL | |
| MC_GearIn_2 | MC_GearIn | |
| GearIn_Ex1 | BOOL | FALSE |
| Gear_RatioNumerator1 | LREAL | 2000.0 |
| Gear_RatioDenominator1 | LREAL | 1000.0 |
| InGear1 | BOOL | |
| GearIn_Bsy1 | BOOL | |
| GearIn_Act1 | BOOL | |
| GearIn_Abt1 | BOOL | |






**2, Motion curve and timing diagram**




- MC_GearIn1 electronic gear ratio of the numerator and denominator are both 1, GearIn_Ex a FALSE to TRUE, after a period, Gear_Bsy, Gear_Act, InGear becomes TRUE, the spindle and the establishment of the gear shaft from the relationship.
- Electronic gear spindle and the establishment of the shaft from the relationship, Vel_Ex changes from FALSE to TRUE, and after a period, Vel_Bsy, Vel_Act becomes TRUE, the instruction execution speed of the spindle, the spindle operation shaft follows.
- MC_GearIn2 electronic gear ratio of the numerator and denominator are 2 and 1, GearIn_Ex1 a FALSE to TRUE, and after a period, GearIn_Bsy1, GearIn_Act1 and GearIn_Abt1 becomes TRUE, the electronic gear set Slave axis than in accordance with the instruction MC_GearIn2, source of the command, the acceleration the rate of change of acceleration, reaching the target speed transfer mode. Because GearIn2 numerator and denominator of the

electronic gear ratio of 2 and 1, respectively, so that the target speed of the shaft is twice the speed of the spindle. When InGear1 becomes TRUE, the speed of the shaft is twice the speed of the spindle.

## 11.5.2 MC_GearOut (electronic gear disengaged instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction for releasing the electronic gear relationship between the two axes established | VEC-VA-MP-005-MA |

➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Slave (Slave axis) | Setting instruction from the shaft to be controlled | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | - |

**Description:**

After establishing the relationship between electronic gear (MC_GearIn) 1. two axes, from electronic gear from the shaft by the relationship If MC_GearOut command, the speed will remain disengaged from the shaft to continue to run.

2. The instructions are executed, the instruction can be executed from the other motion axes.

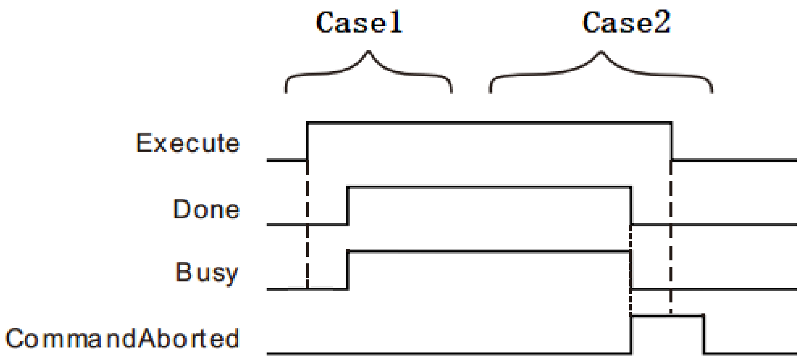
3. After the two axes from the electronic gear relationship (MC_GearOut), if you want to stop the shaft, can be used MC_Halt, MC_Stop or NS_MC_StopByPos instructions cause the slave axis is stopped.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to | BOOL | TRUE or |

| | | | |
|---|---|---|---|
| | TRUE output instruction is executed | | FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1**: When the Execute FALSE to TRUE, after a period, Done becomes TRUE. After Execute a TRUE to FALSE, Busy and Done remains to TRUE.

**Case 2**: When the Execute is TRUE, if the instruction is interrupted by another instruction, CommandAborted becomes TRUE, the Busy and Done becomes FALSE; Execute when a TRUE to FALSE, after a period, CommandAborted becomes FALSE.

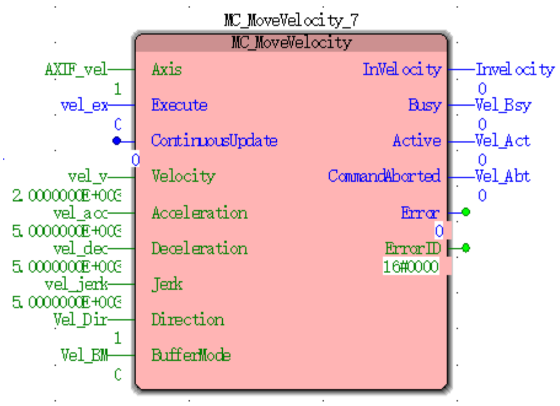
**Program Example**

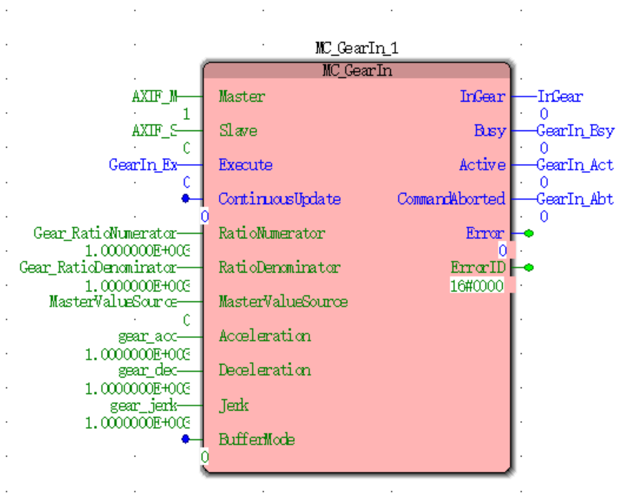
MC_GearOut program of instructions in the example below:

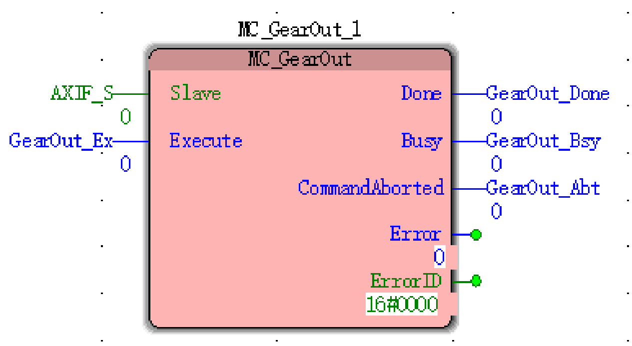
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity_7 | MC_MoveVelocity | |
| AXIF_vel | USINT | 1 |
| Vel_ex | BOOL | FALSE |
| Vel_v | LREAL | 1000.0 |
| Vel_acc | LREAL | 5000.0 |
| Vel_dec | LREAL | 5000.0 |
| Vel_jerk | LREAL | 5000.0 |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| Invelocity | BOOL | |

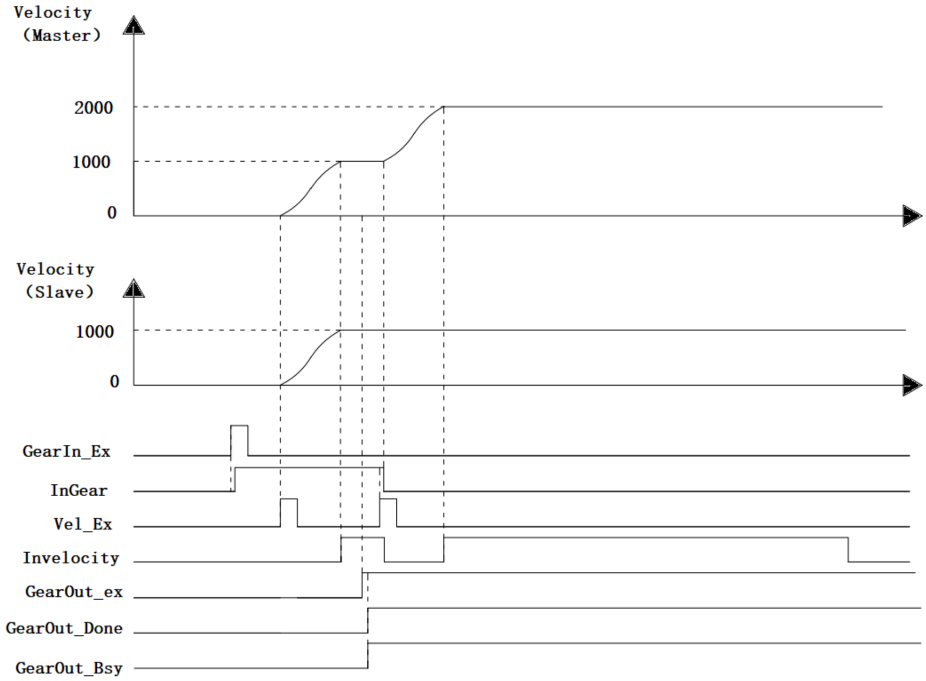
| | | |
|---|---|---|
| Vel_Bsy | BOOL | |
| Vel_Act | BOOL | |
| Vel_Abt | BOOL | |
| MC_GearIn_1 | MC_GearIn | |
| AXIF_M | USINT | 1 |
| AXIF_S | USINT | 0 |
| GearIn_Ex | BOOL | FALSE |
| Gear_RatioNumerator | LREAL | 1000.0 |
| Gear_RatioDenominator | LREAL | 1000.0 |
| MasterValueSource | INT | 0 |
| Gear_acc | LREAL | 1000.0 |
| Gear_dec | LREAL | 1000.0 |
| Gear_Jerk | LREAL | 1000.0 |
| InGear | BOOL | |
| GearIn_Bsy | BOOL | |
| GearIn_Act | BOOL | |
| GearIn_Abt | BOOL | |
| MC_GearOut_1 | MC_GearOut | |
| GearOut_Ex | BOOL | FALSE |
| GearOut_Done | BOOL | |
| GearOut_Bsy | BOOL | |
| GearOut_Abt | BOOL | |





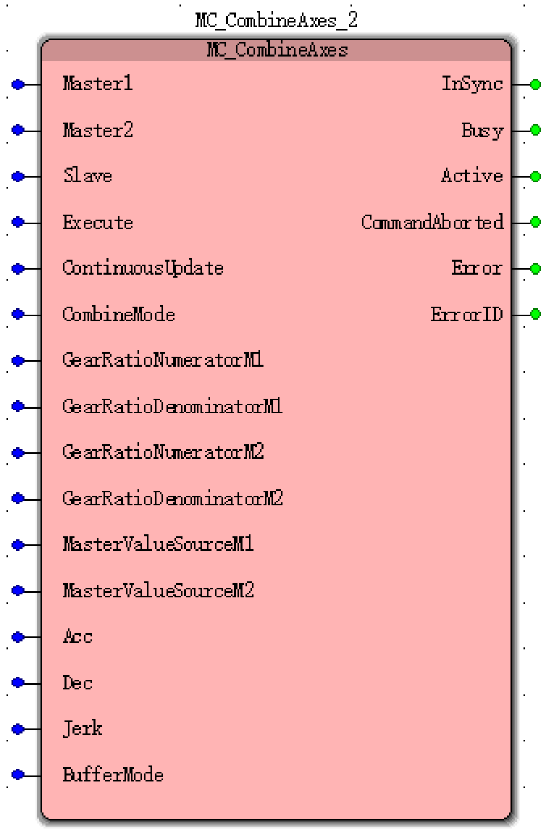

## 2, Motion curve and timing diagram



- MC_GearIn1 electronic gear ratio of the numerator and denominator are both 1, GearIn_Ex a FALSE to TRUE, after a period, Gear_Bsy, Gear_Act, InGear becomes TRUE, the spindle and the establishment of the gear shaft from the relationship.
- Electronic gear spindle and the establishment of the shaft from the relationship, Vel_Ex changes from FALSE to TRUE, and after a period, Vel_Bsy, Vel_Act becomes TRUE, the instruction execution speed of the spindle, the spindle operation shaft follows.
- When the spindle speed command is executed, GearOut_Ex a FALSE to TRUE, after a period, GearOut_Bsy, GearOut_Done and GearIn_Abt becomes TRUE, the current from the shaft speed continues to operate.
- Velocity spindle speed command to modify the parameters of 2000.0, is performed again, increasing the spindle speed 2000.0, no longer subject to the influence from the spindle axis.

## 11.5.3 MC_CombineAxes (double spindle gears combined instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | The positions of the two spindles adding or subtracting a value from the output shaft position | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Master1<br>(Spindle 1) | When controlling the first shaft Location sources. | USINT | **Analog / Pulse**:<br>0-4 (real axis)<br>5 to 11 (imaginary axis)<br>**CANopen mode**: 0-15 (real axis / imaginary axis)<br>(0) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| Master2<br>(Spindle 2) | When controlling the second shaft Location sources. | USINT | 0-4 the real axis<br>5 to 11 virtual axis<br>(CANopen Mode: 0 ~ 15, can be real or imaginary axis)<br>(0) | Exexcute from FALSE to TRUE |
| Slave<br>(Slave axis) | Accused of shaft | USINT | 0-4 the real axis<br>5 to 11 virtual axis<br>(CANopen Mode: 0 ~ 15, can be real or imaginary axis)<br>(0) | Exexcute from FALSE to TRUE |
| Execute<br>(Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | - |
| ContinuousUpdate | Retention | - | - | - |
| CombineMode<br>(Synthesis Mode) | Select the synthesis mode<br>0: adding the change in the position of each of the two spindles<br>1: change in the position of each of the two spindles subtraction | INT | 0 or 1 | Exexcute from FALSE to TRUE |
| GearRatioNumerator M1<br>(Spindle 1 Gear Ratio) | Setting spindle 1 Gear Ratio | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |
| GearRatioDenominator M1<br>(Spindle gear denominator) | Setting spindle gear denominator | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |
| GearRatioNumerator M2<br>(2 spindle Gear Ratio) | Setting spindle 2 Gear Ratio | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| GearRatioDenominator M2 (2 spindle gear denominator) | Setting spindle gear denominator | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |
| MasterValueSourceM1 (Spindle synchronization source 1) | Setting spindle synchronization source 1<br>0: command position<br>1: The actual position | INT | 0 or 1 | Exexcute from FALSE to TRUE |
| MasterValueSourceM2 (Spindle synchronization source 2) | Setting spindle synchronization source 2<br>0: command position<br>1: The actual position | INT | 0 or 1 | Exexcute from FALSE to TRUE |
| Acc (Acceleration) | Setting acceleration from the shaft<br>Unit: unit / S2 | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Dec (decrease speed) | Setting the deceleration from the shaft<br>Unit: unit / S2 | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Jerk (The rate of change of acceleration) | Setting a rate of change of acceleration Slave axis<br>Unit: unit / S3 | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions.<br>0: interrupted<br>1: Wait | INT | 0 or 1 | Exexcute from FALSE to TRUE |

**Description:**

1. This instruction starts execution when the Execute FALSE to TRUE. Whether the command has been executed is completed, the Execute a FALSE to TRUE again, the instructions may be re-executed, the parameters can be revalidated The pin comprises CombineMode, RatioNumeratorM1, RatioDenominatorM1, RatioNumeratorM2, RatioDenominatorM2, MasterValueSourceM1, MasterValueSourceM2, Acceleration, Deceleration, Jerk, BufferMode.

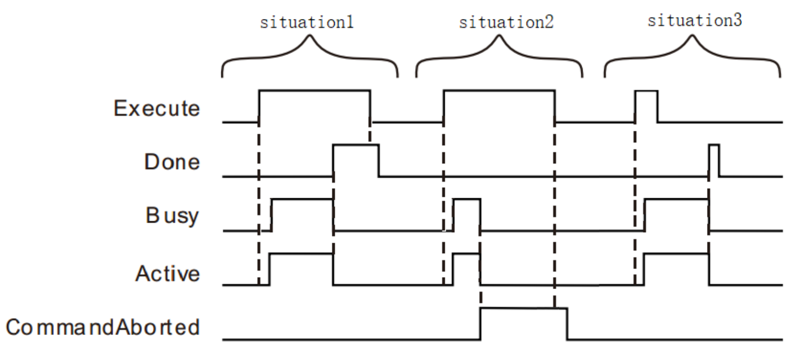
2. When the instruction is executed, execution of the instruction from the shaft as the other motion instructions can be interrupted MC_MoveVelocity this instruction, the spindle will be released from the positional relationship between the axes. MC_Halt may be performed or stopped from MC_Stop axis.

➢ **Output parameters**

| name | Features | type of | Output range |
|---|---|---|---|

| | | data | |
|---|---|---|---|
| InSync (synchronized state) | This parameter is TRUE output shaft from the synchronized state represents | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period, Busy, Active becomes TRUE. When you have two spindles and synchronous slave axis, InSync becomes TRUE, while Busy Active and remains to TRUE.

**Case 2**: When the Execute is TRUE, the Busy is TRUE, Active is TRUE, and when the two spindle synchronization has been Slave axis, InSync is TRUE, this time interrupted by another instruction of this command, CommandAborted becomes TRUE, while Invelocity, Busy and Active becomes FALSE, TRUE when the Execute becomes FALSE, after a period, CommandAborted becomes FALSE.

**Case 3:**In the process of implementation, when the Execute TRUE to FALSE, the instruction is still being executed, Busy and Active status will not change. When you have two spindles and synchronous slave axis, InSync becomes TRUE, while Busy Active and remains to TRUE.

- **Function Description**

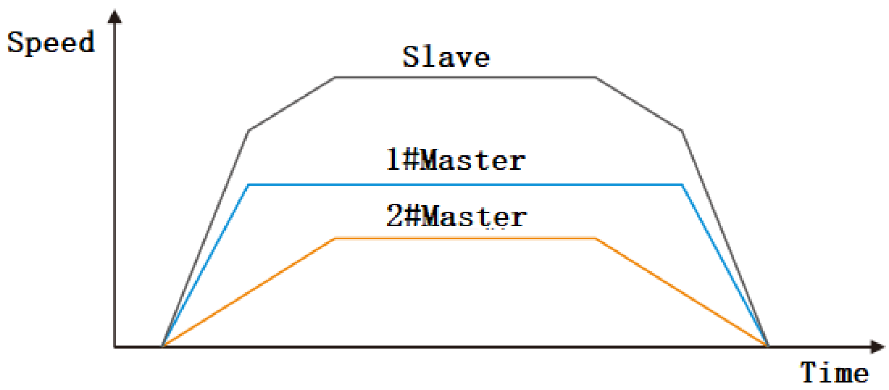
This instruction is used to position two spindles adding or subtracting a value as the output from the shaft position.Here is the location of the unit pulse.

- **This instruction synthetically divided into two: the addition or subtraction**

Position change amount of the change amount and II One main spindle is added or subtracted, the calculated value as an output shaft from a position change amount.

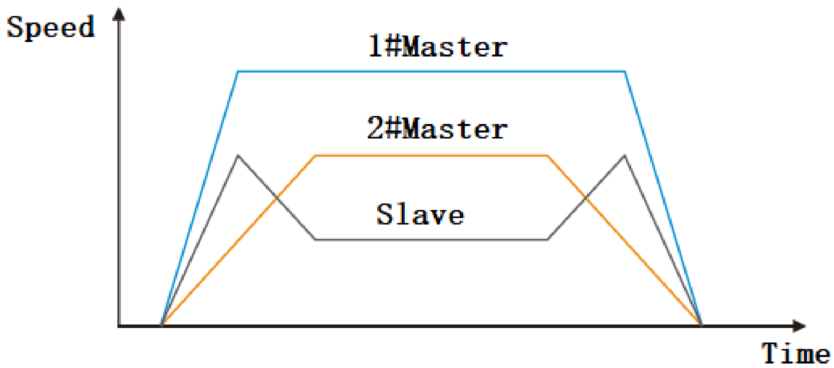
- When the value is 0 CombineMode

$$\text{SlavePositionChange} = \text{1\#MasterPositionChange} * \frac{\text{GearRatioNumerator M1}}{\text{GearRatioDenominator M1}} + \text{2\#MasterPositionChange} * \frac{\text{GearRatioNumerator M2}}{\text{GearRatioDenominator M2}}$$




- When the value is 1 CombineMode

$$\text{SlavePositionChange} = \text{1\#MasterPositionChange} * \frac{\text{GearRatioNumerator M1}}{\text{GearRatioDenominator M1}} - \text{2\#MasterPositionChange} * \frac{\text{GearRatioNumerator M2}}{\text{GearRatioDenominator M2}}$$




- Mainshaft gear ratio numerator and denominator are adjusted to set the amount of change of the two spindle positions factor using Equation supra.

- Spindle synchronization source may be set to 0: position command; 1: the actual position, a position to confirm the source of the amount of change. Is set to 0 for the amount of change of the spindle position command addition or subtraction, it is set to 1 when the amount of change in the actual position of the spindle addition or subtraction.
- Acceleration, deceleration and acceleration of the rate of change represented before executing this instruction, the motion of the spindle has, at this time if this instruction is executed, will be accelerated or decelerated Slave axis of the acceleration, deceleration, and jerk, in order to achieve and synchronous spindle position change. After synchronization InSync is TRUE, the instructions are executed.
- To this end of the main shaft from the instruction relationship, use a motion command from the control shaft (e.g. MC_Stop), BufferMode 0 input pin fill to interrupt this instruction is released from the relationship between the master axis.

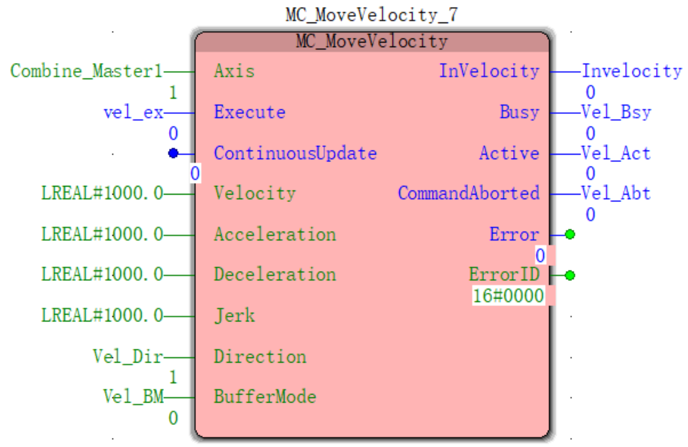
**Program Example**

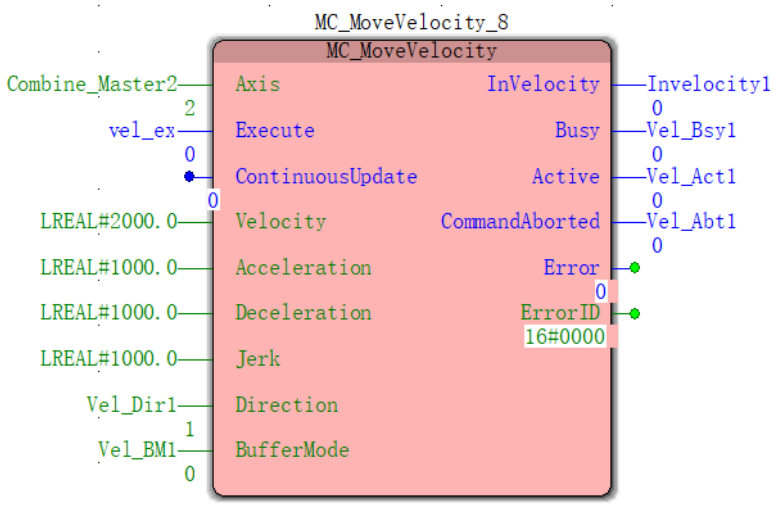
Example MC_CombineAxes instructions of the program are as follows:

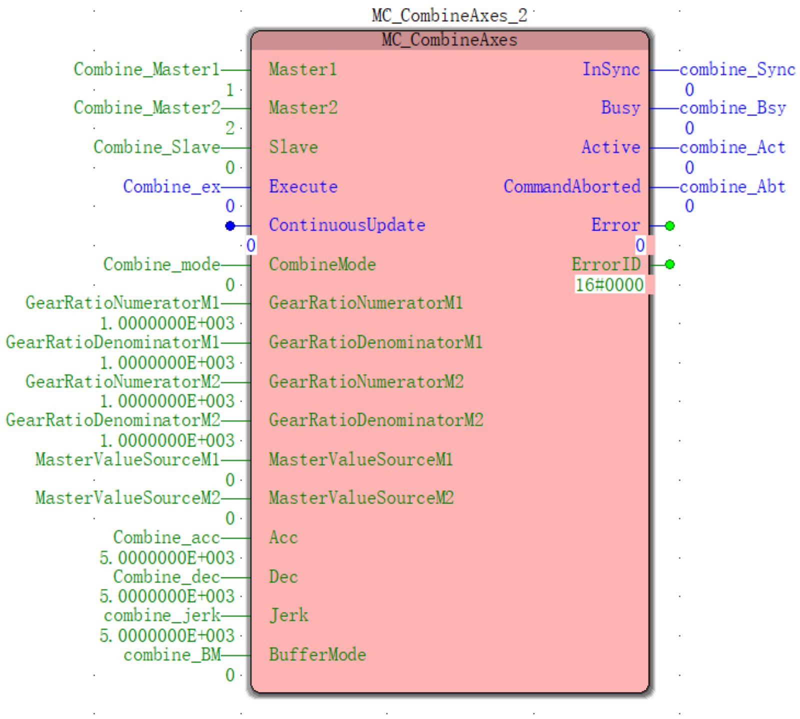
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity_7 | MC_MoveVelocity | |
| Combine_Master1 | USINT | 1 |
| Vel_ex | BOOL | FASLE |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| Invelocity | BOOL | |
| Vel_Bsy | BOOL | |
| Vel_Act | BOOL | |
| Vel_Abt | BOOL | |
| MC_MoveVelocity_8 | MC_MoveVelocity | |
| Combine_Master2 | USINT | 2 |
| Vel_ex1 | BOOL | FASLE |
| Vel_Dir1 | INT | 1 |
| Vel_BM1 | INT | 0 |
| Invelocity1 | BOOL | |
| Vel_Bsy1 | BOOL | |
| Vel_Act1 | BOOL | |
| Vel_Abt1 | BOOL | |
| MC_CombineAxes_2 | MC_CombineAxes | |
| Combine_Slave | USINT | 1 |
| Combine_ex | BOOL | FALSE |
| Combine_mode | INT | 0 |
| GearRatioNumeratorM1 | LREAL | 1000.0 |

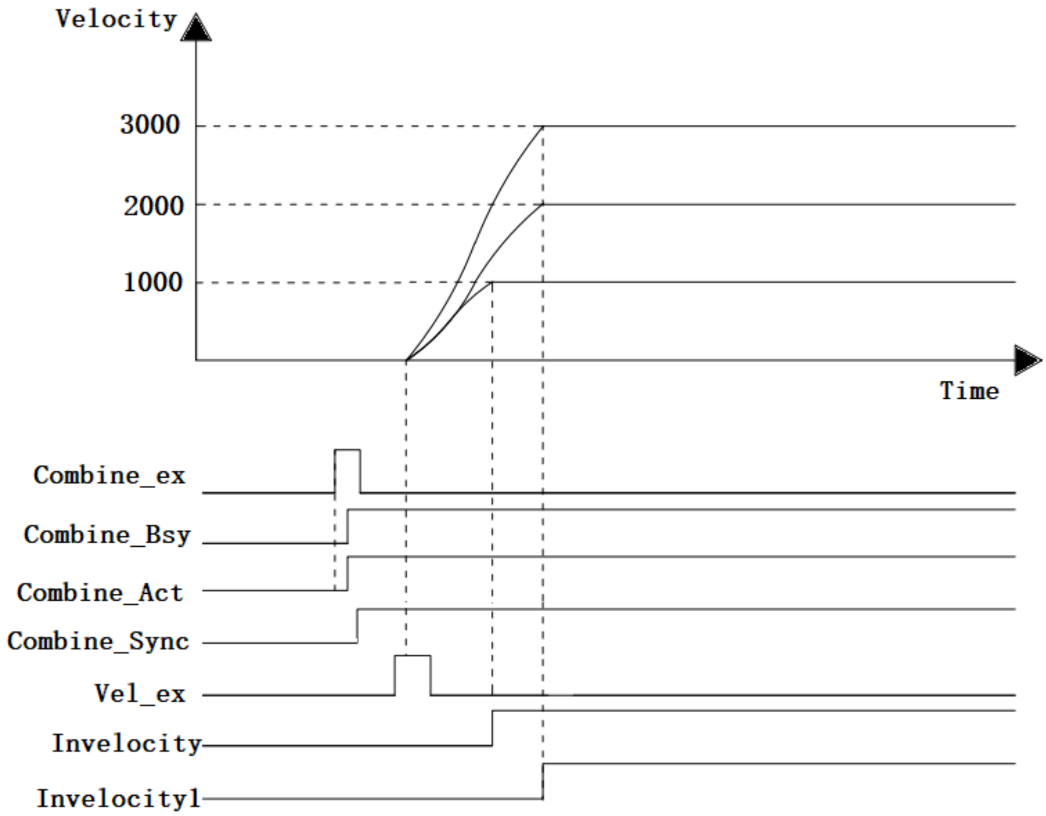
| | | |
|---|---|---|
| GearRatioDenominatorM1 | LREAL | 1000.0 |
| GearRatioNumeratorM2 | LREAL | 1000.0 |
| GearRatioDenominatorM2 | LREAL | 1000.0 |
| MasterValueSourceM1 | INT | 0 |
| MasterValueSourceM2 | INT | 0 |
| Combine_acc | LREAL | 5000.0 |
| Combine_dec | LREAL | 5000.0 |
| Combine_jerk | LREAL | 5000.0 |
| Combine_BM | INT | 0 |
| Combine_Sync | BOOL | |
| Combine_Bsy | BOOL | |
| Combine_Act | BOOL | |
| Combine_Abt | BOOL | |






**2, Motion curve and timing diagram**




- When the Combine_ex FALSE to TRUE, MC_CombineAxes instruction starts execution, after some time, the instruction is executed successfully, Combine_InSync becomes TRUE, the three axes in accordance with the instruction required to achieve synchronous movement state. At this time, the two spindles Excute MC_MoveVelocity instruction becomes TRUE, two spindle starts moving, when the movement of the shaft also starts according to a change amount of a position and two spindles, the position change amount per unit time Slave axis is two spindle position and the amount of change. When the instructions are executed spindle, three axes remain synchronized. To interrupt the synchronization state of the three axes, are available Slave axis, to lift the synchronization state using the corresponding interrupt instruction converter according to the state machine of FIG.

## 11.5.4 peeling electronic cam Profile

**Brief introduction**

Electronic cam is fundamentally a function of the cam computer-implemented, the entire system generally consists of two parts, hardware and software. The system hardware includes a microprocessor, a memory device, D / A converter, a controller and actuators. The encoder signals the microprocessor to obtain from the storage device corresponding to the displacement signal calculated from the formula or a displacement value, then the input of the cam displacement value D / A converter, the converted signal processing to achieve the corresponding actuator is driven by the controller exercise.

**Applications**

Electronic cam having a wide range of applications in the mechanical industry material field cut to length, for example cut to length steel, wood

Material for coil, aluminum strip for cutting fixed-length, the corrugated crop, laterally sealing the packaging bags slitting, punching,

Embossing. Electronic cam application flying shear, peeling, cut recovery, and stop discharging collectively called shear cut the wheel, called transverse.

Relative to the longitudinal cross terms, refers to vertically cut the material in the material transport direction, cutting length is generally fixed.

(1) Peeling

In this mechanism, mounted on a roller (or the number of the shearing blades), driven by movement of the shearing blades rotating roll, the roll motion of the severing one week once (or several times).

(2) Flying Shear

And peeling in the same definitions section, but in some special shear, for example shear plate, the eccentric shaft

Mounting the blade holder about the fixed shearing blade rotary movement, to achieve the purpose of the cut to length [5].

(3) to recover shears

Cut recovery is characterized by: the sync area set pulling speed of the shearing member and feeding same feed rate, shear motion in the sync area is completed, and different lengths of the cut by adjusting the speed to accommodate non-synchronous zone. Chase cut and peeling, the biggest difference is flying shear: Shear chase reciprocating motion, and peeling, flying shear is a movement in the same direction. Another application of shear to catch flying saw, flying saw cutting means that when synchronization feed mechanism, the material for cutting.

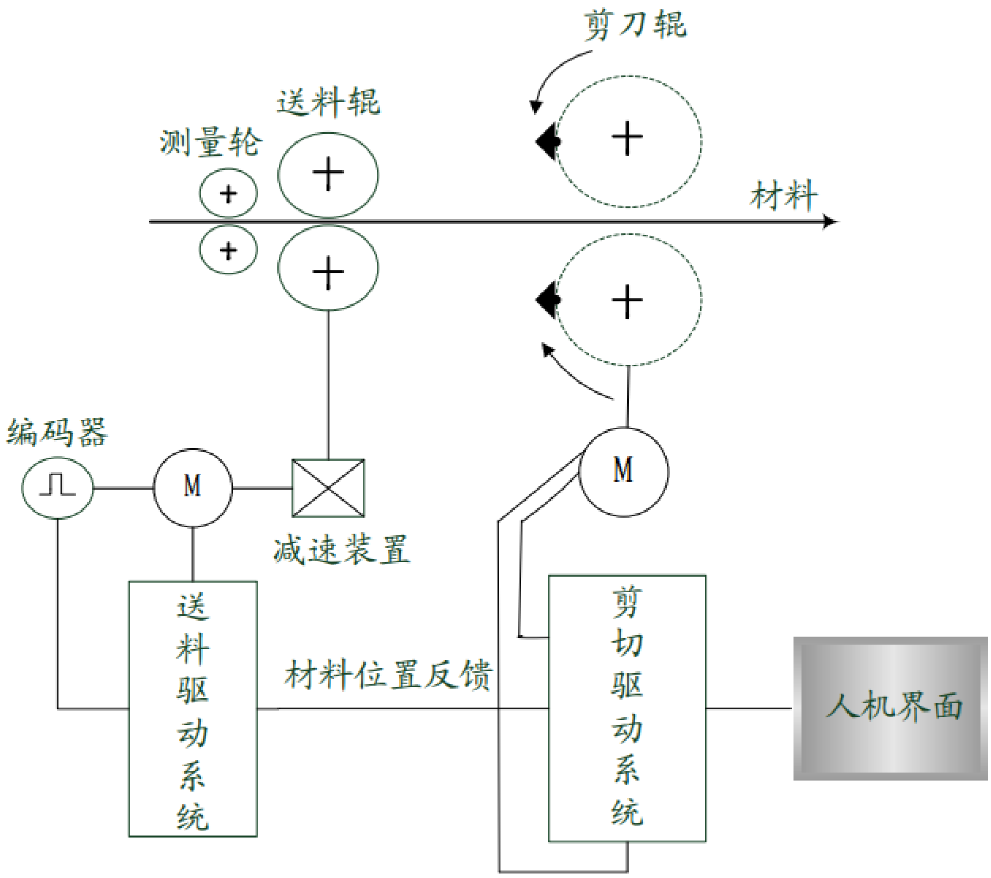
(4) stop cutting

And said cutting different ways, stop feeding the control shaft shear, rapid feed in the shearing blade is lifted off time and cut to stop. Several different from the previous, stop feeding the cut feed gap, so that the shear stop control simple, low processing efficiency.

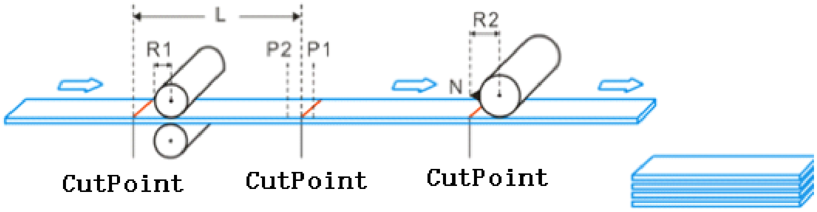
## 11.5.5 peeling function of the system configuration

An essential part of functions including peeling knife roll shaft and the axis, detecting element, and a motor control unit. Other

Depending on the processed products you may also need color. As shown, each blade roll rotation of the complete system configuration in FIG 11.5.9 first shearing, measuring the length measuring roll material, according to pre-designed control unit controlling the movement of the knife roller cam curve, so that blade into engagement with the material by the length of the material it is exactly desired length, thus completing precise cut to length.



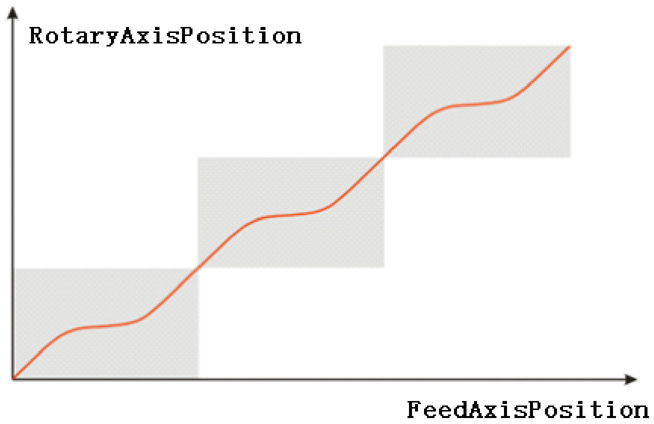
## 11.5.6 Peeling process parameters




| FIG parameters | The actual meaning | Command name |
|---|---|---|
| L | Setting the cut length (unit: unit (means)) | Cut_Length (cutting length) |
| R1 | Material feed roller radius (unit: unit (means)) | FeedAxisRadius (feed shaft radius) |
| R2 | Peeling axis radius; | Cutter_Cir (cutter perimeter) = 2 * R2 * 3.1415 |
| P1 | Sync area | Sync_Angle (synchronous zone) |
| P2 | | |
| N | Bit number peeling axis | RotaryAxisKnifeNum |
| Special Note: Please refer to the name of instruction 11.5.9 NS_MC_RotaryCutIn (peeling instruction) | | |

## 11.5.7 peeling function control characteristics

Rotary Cut function is a special electronic cam function. Continuous cutting, peeling schematic curve which follows:




**Features**

1, The user can freely set the cutting length according to process requirements, can be less than or greater than the cut length equal to the circumference of the cutter.

2. In the synchronization area, the peeling axis feed shaft according to a certain operation speed ratio (speed generally equal), and the cut material occurs in the synchronization area.

3, The peeling function is activated, the feed peeling axis to follow the phase of the operation shaft, the shaft can therefore feed a constant speed, acceleration, deceleration, irregular movement.

4, the roller peeling peeling function supports multiple tip.

5, after the peeling function, rotary cutter stop-zero, i.e., entry point.

## 11.5.8 peeling Features

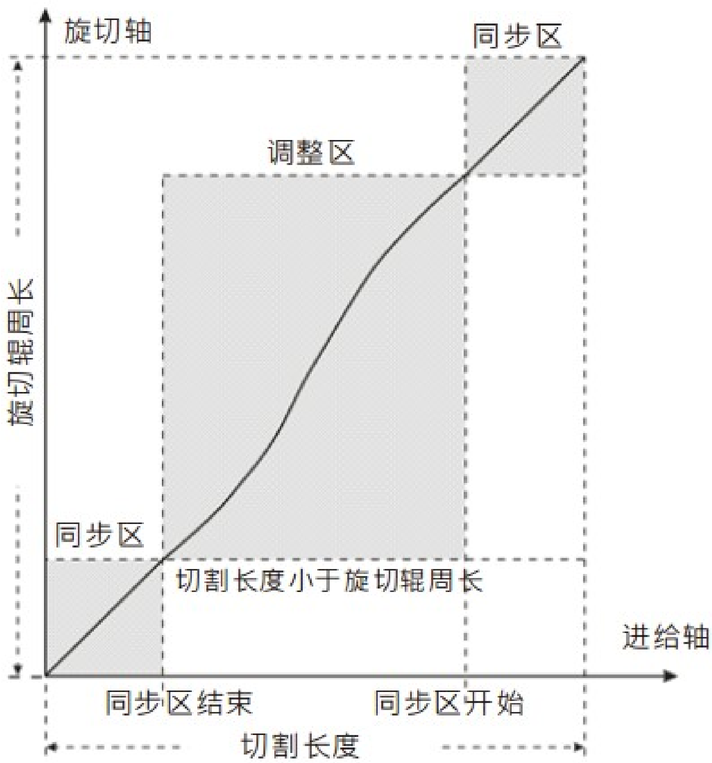
Peeling area and curve into synchronous adjustment zone.

***Sync area*:**At this time, the peeling axis feed shaft at a fixed speed ratio operation (linear velocity of the tip of the cutting face generally equal), and the cut material occurs in the synchronization area.

***Adjustment zone*:**Due to the different cutting lengths, corresponding displacement adjustment needs to be done. The cutting length adjustment region can be divided into the following three cases.

**1: Short material cutting**

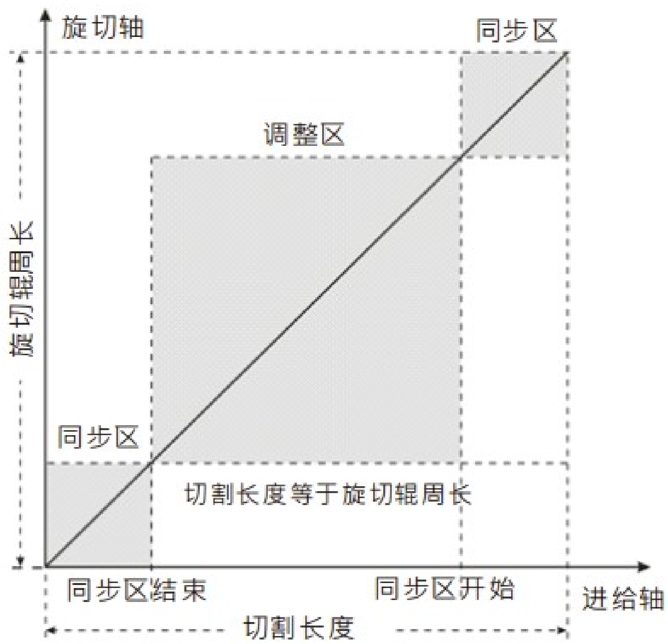
When the cut length is less than the circumference of the roller peeling, peeling cycle follows a curve of any




When feeding short cut, peeling axis adjustment must be accelerated in the region, and then decelerate to synchronous speed.

**2: Isometric cutting**

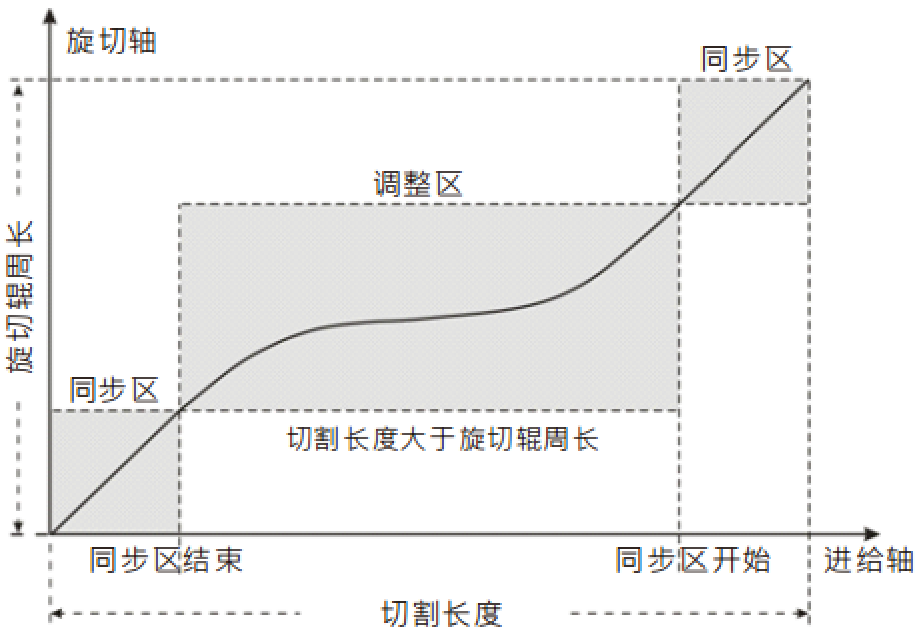
When peeling a length equal to the circumference of the knife roll, peeling curve of any one of the following cycle:




In this case, sync area with the non-synchronous zone peeling axis and the feed axis speed synchronization has been maintained, no adjustment peeling axis.

**3: Long cutting material**

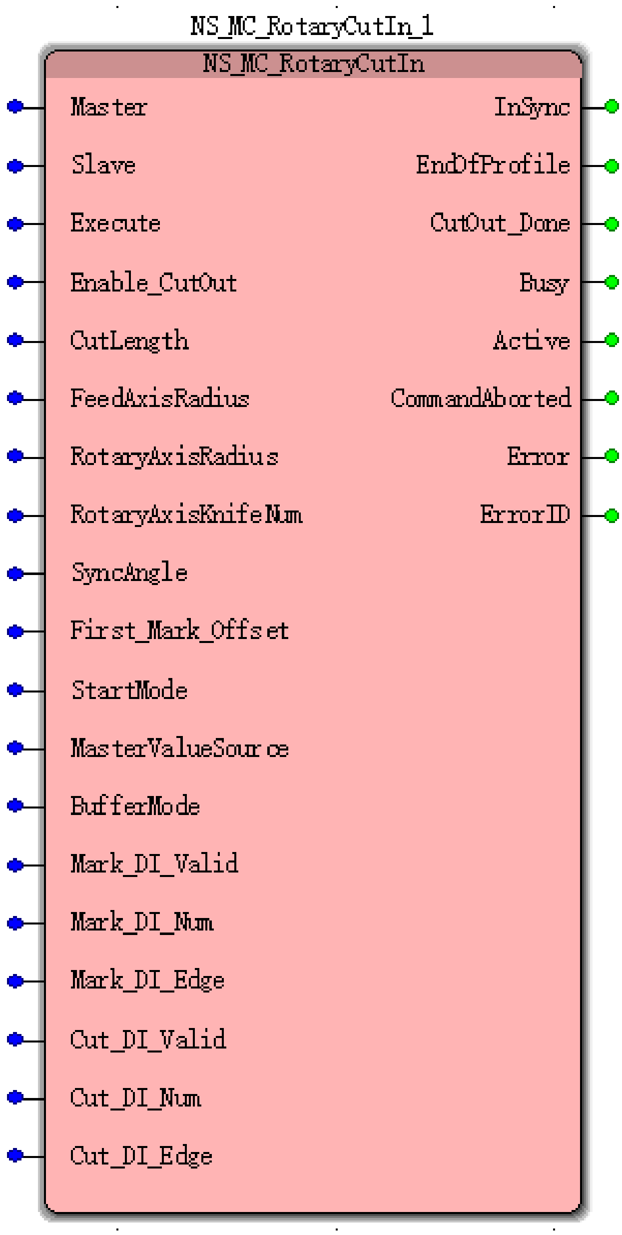
When the peeling length is greater than the circumference of the roller cutter, a peeling profile according to any of the following cycle:




In this case, peeling axis adjustment should first deceleration zone, and then accelerated to synchronous speed. If the length is much greater than peeling knife roll circumference, the cutter roll may have decelerated to zero, stay for some time, and then accelerated to synchronous speed. The longer the cutting length, the longer the time spent.

## 11.5.9 NS_MC_RotaryCutIn (peeling instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to establish a relationship between two axes peeling | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Master (Spindle) | Setting instruction to be controlled spindle | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Slave (Slave axis) | Setting instruction from the shaft to be controlled | USINT | 0-4 the real axis 5 to 11 virtual axis (CANopen Mode: 0 ~ 15, can be real or imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction is executed. | BOOL | TRUE or FALSE | - |
| Enable_CutOut (Lifting peeling bit) | When Enable_CutOut is TRUE and the peripheral end of the cam point, the main shaft is released from the peeling relation, from the shaft stops at the point of tangency. | BOOL | TRUE or FALSE | - |
| CutLength (Cut length) | Setting the cut length (refer to the feed axis length) Unit: unit | LREAL | A positive number | Exexcute from FALSE to TRUE |
| FeedAxisRadius (Feed shaft radius) | Setting the feed axis spindle radius Unit: unit | LREAL | A positive number | Exexcute from FALSE to TRUE |
| RotaryAxisRadius (Peeling axis radius) | Peeling setting spindle radius, i.e. the distance to the center of the roller peeling the tip (Unit: unit) | LREAL | A positive number | Exexcute from FALSE to TRUE |
| RotaryAxisKnifeNum (Peeling head axes) | Peeling axis setting bit number, i.e. the number of tip peeling roller installed | USINT | Positive integer (1 to 16) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | when the bit number is greater than 1, the knife needs to try to ensure consistent distance between the two knives. | | | |
| SyncAngle (Synchronous angle) | Range sync area (peeling angle refers to the axis) Unit: degrees | LREAL | 0`360 | Exexcute from FALSE to TRUE |
| First_Mark_Offset (First color distance) | Punctuation color distance to the cutting point, this value can be provided to achieve the first knife to cut the color punctuation. Unit: unit | LREAL | A positive number, 0 | Exexcute from FALSE to TRUE |
| StartMode (Startup mode) | Setting the startup mode of the instruction, as detailed in the mode described. | INT | 0-4 | Exexcute from FALSE to TRUE |
| MasterValueSource (Spindle synchronization source) | Source selection command from the shaft, when the selection time StartMode = 2,3,4, MasterValueSource must be 1. 0: Follow the spindle axis from a position command 1: the actual position of the shaft from the spindle to follow | INT | 0,1 | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0: immediately interrupted 1: Wait | INT | 0,1 | Exexcute from FALSE to TRUE |
| Mark_DI_Valid (Color code signal valid bit) | Status is TRUE valid color code signal; FALSE invalid color code signal; | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| Mark_DI_Num (Punctuation color signal) | Punctuation terminal specified color number, input | INT | 0 ~ 15 | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | Value of 0 to 7 corresponding to the input point I0.0 ~ I0.7,8 ~ 15 corresponding to the input I1.0 ~ I1.7 | | | |
| Mark_DI_Edge (Color signal edge punctuation) | Punctuation set color signal trigger edge along 0: Falling 1: Rising | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| Cut_DI_Valid (Cut point signal valid bit) | Status is TRUE effective cut point signal; FALSE cutting point signal is invalid; | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| Cut_DI_Num (Cut point signal) | Cutting bit number designated terminal, the input value of 0 to 7 corresponding to the input point I0.0 ~ I0.7,8 ~ 15 corresponding to the input I1.0 ~ I1.7 | INT | 0 ~ 15 | Exexcute from FALSE to TRUE |
| Cut_DI_Edge (Cutting edge point signal) | Cutting edge set-point signal trigger along 0: Falling 1: Rising | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |

**Description:**

- For Disc_Circumference peeling axis and the feed axis, set FeedAxisRadius (feed shaft radius), RotaryAxisRadius (peeling axis radius) before, should their respective axis parameter module MC_AXIS_REF, set their respective radii match (circle circumference of the disc) parameters for use in MC_ReadActualPosition, MC_ReadActualVelocity module.
- When the instruction is being executed may be modified CutLength, after re-trigger value Execute SyncAngle, FeedAxisRadius, RotaryAxisRaidus, RotaryAxisKnifeNum, First_Mark_Offset, Mark_DI_Valid, Cut_DI_Valid, the modified parameters to take effect in the next cycle after the trigger cam Execute. Execute and do not re-trigger CommandAborted set to TRUE;
- The CANopen mode, or the tangent point with color function is not available.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| InSync (synchronized state) | This parameter is TRUE output shaft from the | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| | synchronized state represents | | |
| EndOfProfile (peeling end of the cam execution flag) | This parameter indicates the output end of the cam is TRUE is performed | BOOL | TRUE or FALSE |
| CutOut_Done (complete lifting peeling) | This parameter indicates the output shaft is released from the main completion peeling relationship is TRUE | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

**Mode Description**

| mode | Explanation |
|---|---|
| 0 | After the instruction is executed, the direct current to the synchronous speed point tangent point follows the start of the spindle. |
| 1 | After the instruction is executed, the current point to point half of the circumference of the knife (i.e., the tangent point opposite points) follow the master boot zero speed |
| 2 | After the instruction is executed, the trigger signal color to the current tool point of half the circumferential point (i.e. the point opposite the point of tangency) follow the master boot zero speed. The real axis of the spindle must be in this mode, it is set Mark_DI_Valid Ture, and is False Cut_DI_Valid |
| 3 | After the instruction is executed, the current point of the knife periphery half of the points (i.e., opposing points tangent point) zero speed start to follow the spindle, to reach synchronous speed in the sync area, waiting for the cut point trigger, the tangential point trigger, executing the next cam cycle. The real axis of the spindle must be in this mode, it is set Cut_DI_Valid Ture, and is False Mark_DI_Valid |
| 4 | After the instruction is executed, the trigger color signal, the current point of the knife periphery half of the points (i.e., opposing points tangent point) zero speed start follow the master waits cut point trigger, while the chromatic scale, if the color patch triggered later than the point of tangency trigger immediate early cut, then color when |

| | |
|---|---|
| | triggered from a rotary axis will be compensated, can ensure that the next sets of standard. The real axis of the spindle must be in this mode, Cut_DI_Valid Mark_DI_Valid and are set to Ture |

➢ **Function Description**

1,The camshaft is a kind of follow the same movement, the camshaft position itselfYesBy the parameters we setThe spindle positionAutomatic planning out

2,Cutting wheel module is continuous cam curve. As long as proper planning parameters without continuousExecuting instructions, May remain fixed length mode. (Because the cutting wheel and the position of zero clearance profile planning point is tangent point, and is typically applied at the tangent point from the shaft)

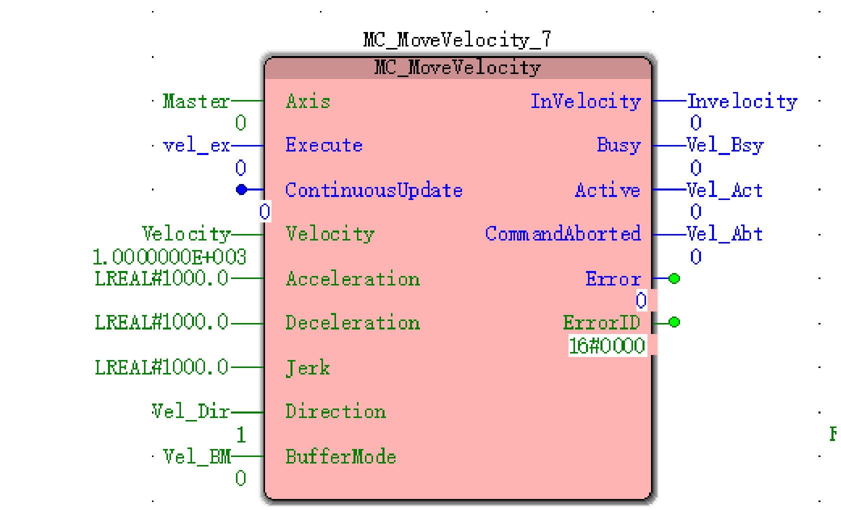
3, in mode 3, 4, the signal contact point indispensable, otherwise it is impossible to realize the function of electronic cam wheel cut from the shaft, both tangent point from the start position is the end position of the axis cleared zero, no cut point Slave axis has been running at synchronous speed, can not be executed next cam cycle;

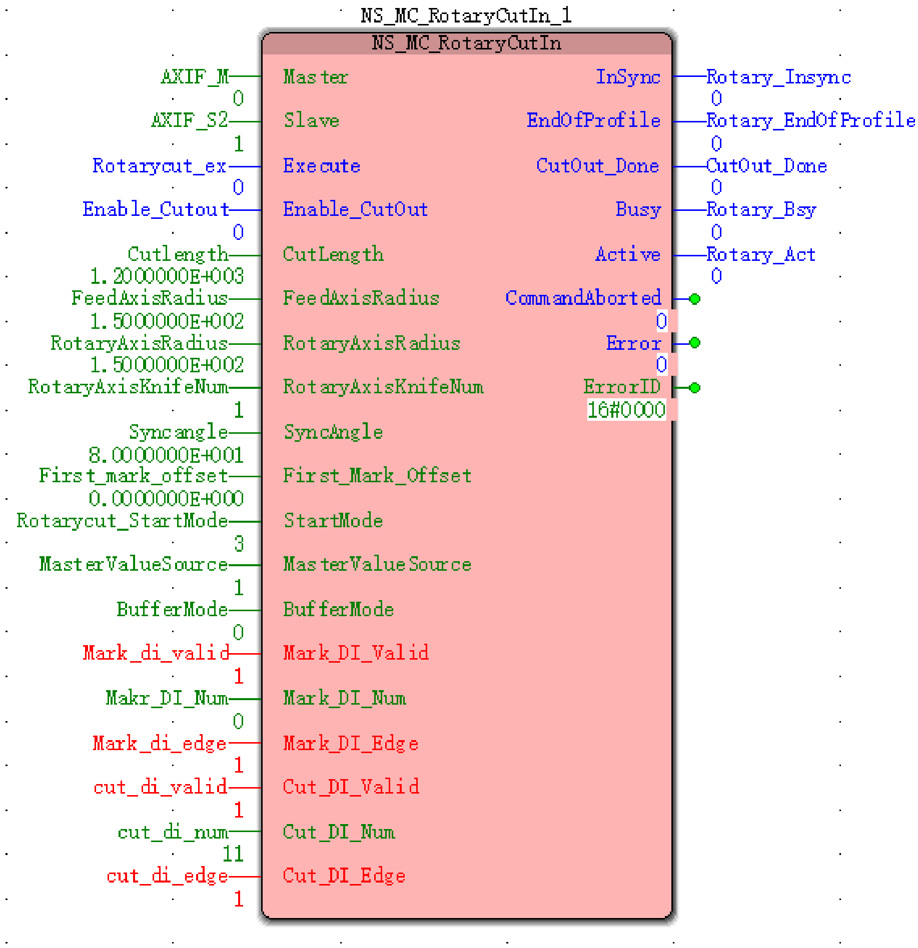
4, under CANopen mode, only mode 0,1.

Program Example

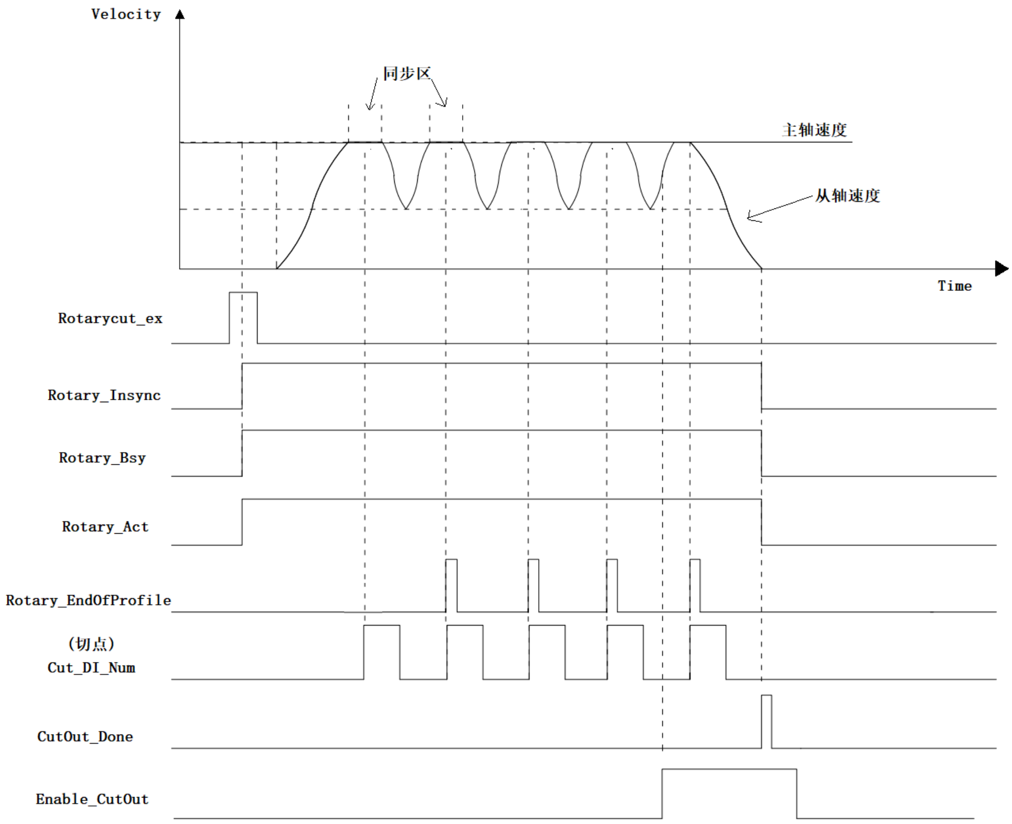
1, Variables, and procedures

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity_7 | MC_MoveVelocity | |
| Master | USINT | 0 |
| Velocity | LREAL | 1000.0 |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| NS_MC_RotaryCutIn_1 | NS_MC_RotaryCutIn | |
| AXIF_M | USINT | 0 |
| AXIF_S2 | USINT | 1 |
| Rotarycut_ex | BOOL | FALSE |
| Enable_Cutout | BOOL | FASLE |
| StartMode | INT | 3 |
| MasterValueSource | INT | 1 |
| BufferMode | INT | 0 |
| Mark_di_valid | BOOL | TRUE |
| Mark_DI_Num | INT | 11 |
| Mark_DI_Edge | BOOL | TRUE |
| Rotary_Insync | BOOL | |
| Rotary_EndOfProfile | BOOL | |
| CutOut_Done | BOOL | |
| Rotary_Bsy | BOOL | |
| Rotary_Act | BOOL | |


MC_MoveVelocity_7
MC_MoveVelocity
Master
0
vel_ex
0
0
Velocity
1.0000000E+003
LREAL#1000.0
LREAL#1000.0
LREAL#1000.0
Vel_Dir
1
Vel_BM
0
Axis
Execute
ContinuousUpdate
Velocity
Acceleration
Deceleration
Jerk
Direction
BufferMode
InVelocity
Busy
Active
CommandAborted
Error
0
ErrorID
16#0000
Invelocity
0
Vel_Bsy
0
Vel_Act
0
Vel_Abt
0


NS_MC_RotaryCutIn_1
NS_MC_RotaryCutIn
AXIF_M
0
AXIF_S2
1
Rotarycut_ex
0
Enable_Cutout
0
Cutlength
1.2000000E+003
FeedAxisRadius
1.5000000E+002
RotaryAxisRadius
1.5000000E+002
RotaryAxisKnifeNum
1
Syncangle
8.0000000E+001
First_mark_offset
0.0000000E+000
Rotarycut_StartMode
3
MasterValueSource
1
BufferMode
0
Mark_di_valid
1
Makr_DI_Num
0
Mark_di_edge
1
cut_di_valid
1
cut_di_num
11
cut_di_edge
1
Master
Slave
Execute
Enable_CutOut
CutLength
FeedAxisRadius
RotaryAxisRadius
RotaryAxisKnifeNum
SyncAngle
First_Mark_Offset
StartMode
MasterValueSource
BufferMode
Mark_DI_Valid
Mark_DI_Num
Mark_DI_Edge
Cut_DI_Valid
Cut_DI_Num
Cut_DI_Edge
InSync
EndOfProfile
CutOut_Done
Busy
Active
CommandAborted
0
Error
0
ErrorID
16#0000
Rotary_Insync
0
Rotary_EndOfProfile
0
CutOut_Done
0
Rotary_Bsy
0
Rotary_Act
0

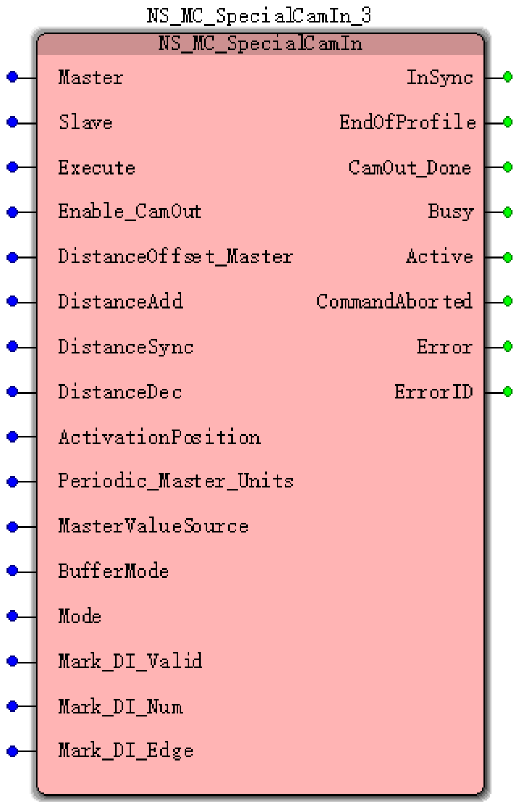
2, Motion curve and timing diagram




- To StartMode =3Executing instructions, Execute becomes TRUEAfter a period, InSync, Busy, Active becomes TRUE,From the shaft reaches the soft-start sync area, a trigger signal is tangent point, the cam next cycle. Then cut each time the trigger point signal, output EndOfProfile signal TRUEA cycle, if the cut point signal has not been triggered, the slave axis has been running at a line speed synchronization.
- Enable_CutOut becomes TRUE, Cut the trigger point signal from the shaft to lift peeling relationship and stop at the tangent point of the opposition point.

## 11.5.10 NS_MC_SpecialCamin (special cam instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | Establishing a special instruction is used for two cam shafts relationship between | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Master (Spindle) | Setting instruction to be controlled spindle | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Slave (Slave axis) | Setting instruction from the shaft to be controlled | USINT | 0-4 the real axis 5 to 11 virtual | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | | | axis (CANopen Mode: 0 ~ 15, can be real or imaginary axis) (0) | |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | - |
| Enable_CamOut (Special relationship lift cam) | When the master is released soon Enable_CamOut is TRUE, the next cam cycle comes from the special relationship between the cam shaft | BOOL | TRUE or FALSE | |
| DistanceOffset_Master (Spindle position offset) | Setting spindle cam offset distance from the starting point | LREAL | A positive number | Exexcute from FALSE to TRUE |
| DistanceAdd (Acceleration distance) | Acceleration distance from the shaft sync area reached by a stationary | LREAL | A positive number | Exexcute from FALSE to TRUE |
| DistanceSync (Synchronous distance) | Synchronous operation to the distance Slave axis of the sync area | LREAL | A positive number,0 | Exexcute from FALSE to TRUE |
| DistanceDec (Deceleration distance) | Slave axis reaches the deceleration distance from the stationary sync area | LREAL | A positive number | Exexcute from FALSE to TRUE |
| ActivationPosition (Engagement start position) | Setting the engagement process begins spindle position, i.e., when the spindle passes through this position, the engaging operation started from the shaft. 0,1 effective mode. (Engagement start position relative position) | LREAL | A positive number, 0 | Exexcute from FALSE to TRUE |
| Periodic_Master_Units (Spindle unit cam cycle) | Periodic_Master_Units = DistanceOffset_Master + DistanceAdd * | LREAL | Positive, zero | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | 30/16 + DistanceSync + DistanceDec * 30/16 Effective Mode 0 | | | |
| MasterValueSource (Select location source) | Source selection command from the shaft, when selected when Mode = 1,2, MasterValueSource must be 1. 0: Follow the spindle axis from a position command 1: the actual position of the shaft from the spindle to follow | INT | 0,1 | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions 0: immediately interrupted 1: Wait | INT | 0,1 | Exexcute from FALSE to TRUE |
| Mode (Instruction execution mode) | Setting the startup mode of the instruction, as detailed in the mode described. | INT | 0-2 | Exexcute from FALSE to TRUE |
| Mark_DI_Valid (Color code signal valid bit) | Status is TRUE valid color code signal; FALSE invalid color code signal; | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| Mark_DI_Num (Color code signal) | Punctuation terminal specified color number, input Value of 0 to 7 corresponding to the input point I0.0 ~ I0.7,8 ~ 15 corresponding to the input I1.0 ~ I1.7 | INT | 0 ~ 15 | Exexcute from FALSE to TRUE |
| Mark_DI_Edge (Color code signal edge) | Punctuation set color signal trigger edge along 0: Falling 1: Rising | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
| --- | --- | --- | --- |
| InSync (synchronized state) | This parameter is TRUE output shaft from the synchronized state represents | BOOL | TRUE or FALSE |
| EndOfProfile (end of the cam execution flag) | This parameter indicates the output end of the cam is TRUE is performed | BOOL | TRUE or FALSE |
| CutOut_Done (complete lifting peeling) | This parameter indicates the output shaft is released from the main completion peeling relationship is TRUE | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

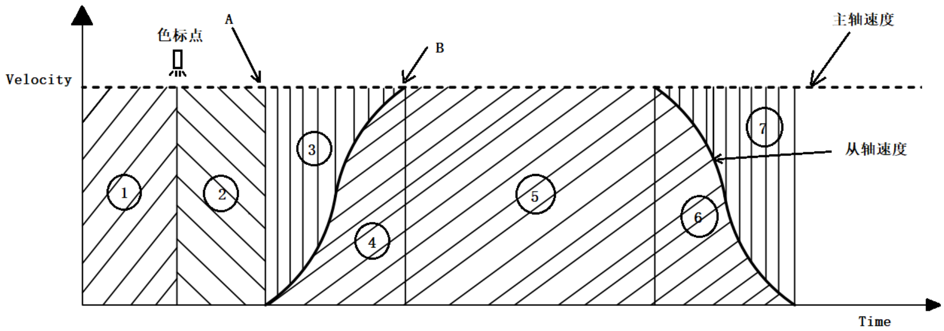
➢ **Mode Description**

| Mode | Explanation |
| --- | --- |
| 0 | When the instruction is executed, when the spindle reaches ActivationPosition starts engagement, the axis of Periodic_Master_Units motion cycles, the need to Mark_DI_Valid this mode is set to False |
| 1 | When the instruction is executed, when the spindle reaches the start ActivationPosition engaged to perform a Periodic_Master_Units cam cycle, then a trigger for each color, according to the axisDistanceOffset_Master, DistanceAdd, DistanceSync, DistanceDec a complete set cam cycle, In this mode it needs to be set to Ture Mark_DI_Valid |
| 2 | When the instruction is executed, a trigger for each color, according to the axisDistanceOffset_Master, DistanceAdd, DistanceSync, DistanceDec a complete set cam cycle, In this mode it needs to be set to Ture Mark_DI_Valid |

➢ **Function Description**

- When the instruction is being executed may be modified DistanceOffset_Master, DistanceAdd, DistanceSync

  DistanceDec, the value of the retrigger Periodic_Master_Units Execute, the modified parameters to take effect in the next cycle after the trigger cam Execute. After modifying Mark_DI_Valid, Mark_DI_Num, Mark_DI_Edge value of the modified parameter re-trigger Execute effective immediately. Execute and do not re-trigger CommandAborted set to TRUE;
- Mode 2, if not performed on a complete cam cycle, the trigger signal and the color does not immediately begin the next cycle again a cam;
- Under CANopen mode, colored punctuation function can not be used;
- MC_SpecialCamIn implemented instructions for controlling a synchronous movement of the cam shaft from the cam in accordance with the relationship with the spindle preplanned;
- Source electronic cam shaft selectively to the real axis (AXIS0 ~ AXIS3) or the imaginary axis or spindle (AXIS4);
- The electronic cam instruction parameters DistanceOffset_Master, AvtivationPosition, DistanceAdd, DistanceSync, DistanceDec, electronic cam curve generated automatically; cam position relationship between the main shaft as shown;




(1) The upper panel shows the electronic cam and the spindle speed timing positional relationship:

(2) ① He expressed ActivationPosition, ② ShowDistanceOffset_Master, ④ An electron acceleration distance setting cam, ⑤ It represents a constant speed from the electronic cam (electronic cam shaft from the main axis equal speed), ⑥ An electron deceleration distance setting cam. among them, ③ versus ④ , ⑥ versus ⑦ The area ratio are all 14:16. This ratio can be solved by a period from the spindle axis to recover the desired speed acceleration distance.

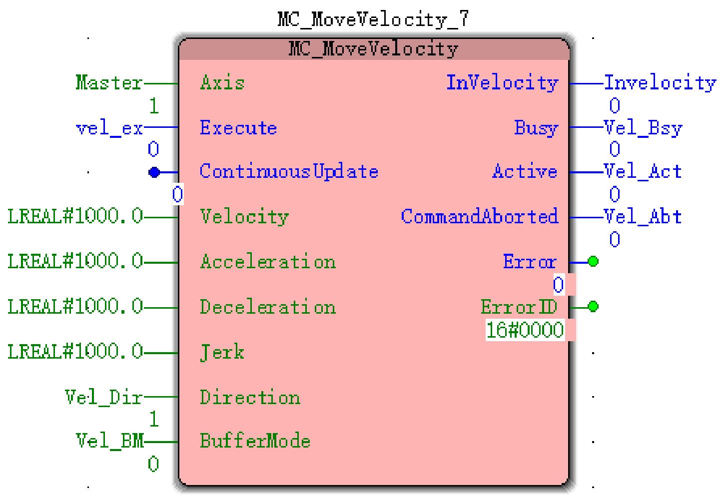
Example: Suppose claim spindleA motion to BAfter position (distancefor 100 Units),Electronic camFrom the shaftSynchronization with the main axis of the speed, the acceleration distance of the electronic cam should be set to the number? Solution: Set Electronic cam acceleration distance X,then X = 100 * 16/30;

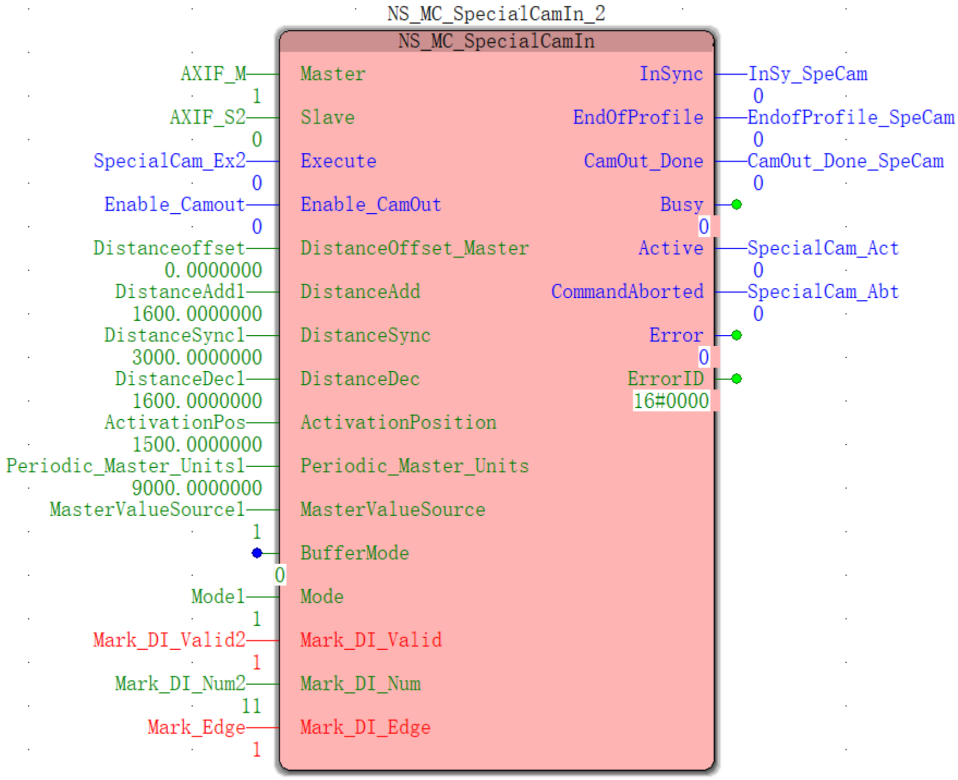
**Program Example**

In a mode instruction execution NS_MC_SpecialCamin examples shown below.

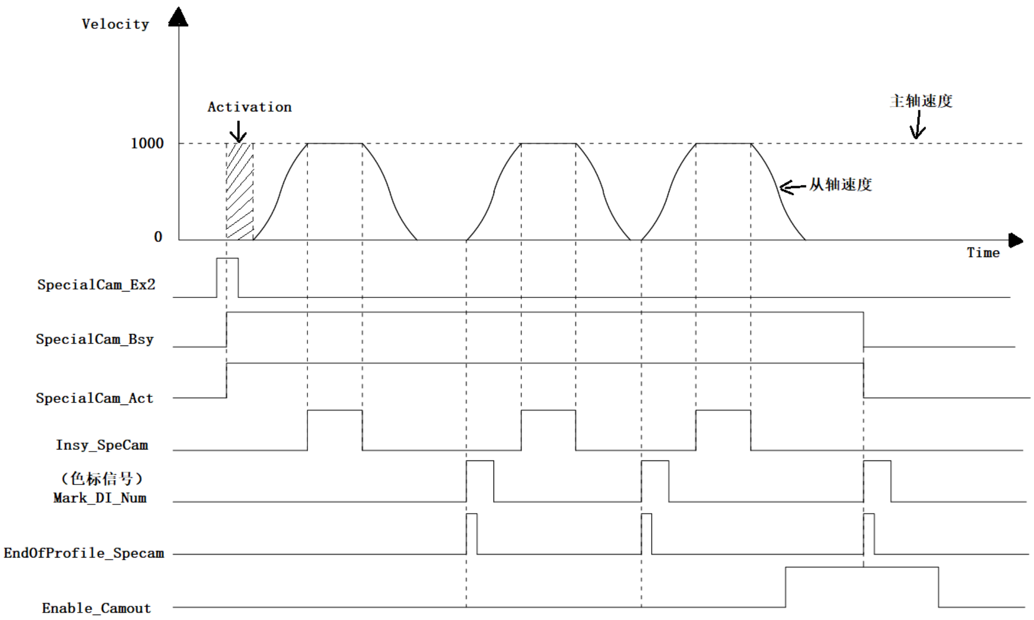
**1, procedures, and variables**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity_7 | MC_MoveVelocity | |
| Master | USINT | 1 |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| NS_MC_SpecialCamIn_3 | NS_MC_SpecialCamIn | |
| AXIF_M | USINT | 1 |
| AXIF_S2 | USINT | 0 |
| SpecialCam_Ex2 | BOOL | FALSE |
| Enable_Camout | BOOL | FALSE |
| Distanceoffset | LREAL | 0.0 |
| DistanceAdd1 | LREAL | 1600.0 |
| DistanceSync1 | LREAL | 3000.0 |
| DistanceDec1 | LREAL | 1600.0 |
| ActivationPos | LREAL | 1500.0 |
| Periodic_Master_Units1 | LREAL | 9000.0 |
| MasterValueSource1 | INT | 1 |
| Mode1 | INT | 1 |
| Mark_DI_Valid2 | BOOL | TRUE |
| Mark_DI_Num2 | INT | 11 |
| Mark_DI_Edge2 | BOOL | TRUE |
| Insy_SpeCam | BOOL | |
| endOfProfile_specam | BOOL | |
| CamOut_done_SpeCam | BOOL | |
| SpecialCam_Bsy | BOOL | |
| SpecialCam_Act | BOOL | |
| SpecialCam_Abt | BOOL | |






**2, the motion profile and timing diagrams**

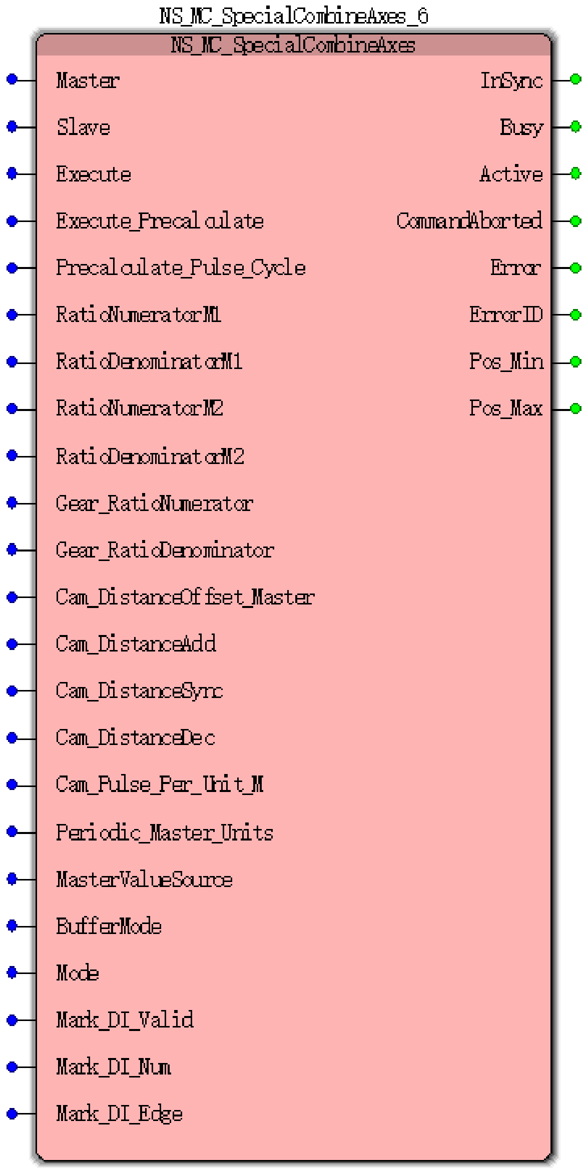



- Mode = 1 NS_MC_SpecialCamIn to execute instructions, the Execute becomes TRUE, after a period, Busy, Active becomes TRUE, in this case as a starting point, when the spindle reaches the Activation, the distance Slave axis in accordance with the acceleration set to sync area in the sync area inner, InSync output TRUE, and then completes the synchronization deceleration distance from the end of the first synchronization period.

- After the completion of the first cam cycle, the trigger signal color, synchronized motion from a cam shaft in accordance with the table set immediately generated parameters.
- Enable_Camout set to TRUE, the time of arrival of a next color signal, released from the relationship between the main cam shaft.

## 11.5.11 NS_MC_SpecialCombineAxes (special double joint spindle gear command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to establish a special relationship between the gear joint between two two-spindle axes | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Master (Spindle) | Setting instruction to be controlled spindle | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Slave (Slave axis) | Setting instruction from the shaft to be controlled | USINT | 0-4 the real axis 5 to 11 virtual axis (CANopen Mode: 0 ~ 15, can be real or imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution | BOOL | TRUE or FASLE | |
| Execute_Precaculate (Precomputed execute bit) | When Execute_Precaculate is TRUE, the calculated value Pos_Min and Pos_Max | BOOL | TRUE or FASLE | |
| Precaculate_Pulse_ Cycle (Pre-calculated number of spindles cycle pulse) | This parameter is used to calculate and Pos_Max Pos_Min Precalculate_Pulse_Cycle = Periodic_Master_Units * Cam_Pulse_Per_Unit_M Unit: Pulse | LREAL | A positive number | When Execute_Precaculate changed from FALSE to TRUE |
| RatioNumerator M1 (Spindle 1 Gear Ratio) | Setting spindle 1 Gear Ratio | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |
| RatioDenominator M1 (Spindle gear denominator) | Setting spindle gear denominator | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| RatioNumerator M2 (2 spindle Gear Ratio) | Setting spindle 2 Gear Ratio | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |
| RatioDenominator M2 (2 spindle gear denominator) | Setting spindle gear denominator | LREAL | Positive or negative (Non-default) | Exexcute from FALSE to TRUE |
| Gear_RatioNumerator (Electronic gear molecule) | Molecular electronic gear ratio MC_GearIn | LREAL | Positive, negative, (Non-default) | Exexcute from FALSE to TRUE |
| Gear_RatioDenominator (Electronic gear denominator) | Electronic gear denominator MC_GearIn | LREAL | Positive, negative, (Non-default) | Exexcute from FALSE to TRUE |
| Cam_DistanceOffset_Master (Spindle position deviation) | Unit: unit | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |
| Cam_DistanceAdd (Acceleration distance) | NS_MC_SpecialCamIn acceleration distance sync area reached by a stationary | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Cam_DistanceSync (Synchronous distance) | NS_MC_SpecialCamIn distance synchronous operation of the sync field | LREAL | Positive, zero (0) | Exexcute from FALSE to TRUE |
| Cam_DistanceDec (Deceleration distance) | NS_MC_SpecialCamIn deceleration distance from the sync area reached stationary | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Cam_Pulse_Per_Unit_M (Spindle number of pulses per unit) | This value is determined according to MC_SpecialCamIn cam curve planning, setting this value such as 5, when the spindle take 10,000 pulses at this time that the mobile terminal of the actuator spindle position 2000 units. | LREAL | A positive number | Exexcute from FALSE to TRUE |
| Periodic_Master_Units (Spindle cam means cycles) | Periodic_Master_Units = DistanceOffset_Master + DistanceAdd * | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | 30/16 + DistanceSync + DistanceDec * 30/16 | | | |
| MasterValueSource (Select location source) | Source selection command from the shaft<br>0: Follow the spindle axis from a position command<br>1: the actual position of the shaft from the spindle to follow<br>(Mode 1 this value must be 1) | INT | 0,1 | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions<br>0: immediately interrupted<br>1: Wait | INT | 0,1 | Exexcute from FALSE to TRUE |
| Mode (mode) | Instruction execution mode<br>Mode 0: superimposed on the tracking position changes of the spindle 1 and the spindle 2 from the shaft<br>Mode 1: consult our technical staff | INT | 0,1 | Exexcute from FALSE to TRUE |
| Mark_DI_Valid (Valid bit color) | Status is TRUE valid color code signal; FALSE invalid color code signal; | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| Mark_DI_Num (Color code signal) | Punctuation terminal specified color number, input<br>Value of 0 to 7 corresponding to the input point I0.0 ~ I0.7,8 ~ 15 corresponding to the input I1.0 ~ I1.7 | INT | 0 ~ 15 | Exexcute from FALSE to TRUE |
| Mark_DI_Edge (Color code signal edge) | Punctuation set color signal trigger edge along<br>0: Falling<br>1: Rising | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |

➢ **Function Description:**

- When the instruction is being executed may be modified RatioNumeratorM1, RatioDenominatorM1, RatioNumeratorM2, RatioDenominatorM2, Gear_RatioNumerator, Gear_RatioDenominator, Cam_DistanceOffset_Master, Cam_DistanceAdd, Cam_DistanceSync, Cam_DistanceDec, Cam_Pulse_Per_Unit_M, Periodic_Master_Units
  Execute the retrigger value modified parameter effect at the next cycle after the trigger cam Execute. Execute and do not re-trigger CommandAborted set to TRUE;
- Under CANopen mode, colored punctuation function can not be used;
- NS_MC_SpecialCombineAxes instruction can be regarded as the superposition of two parts:
  ①By following the movement of the spindle axis from the instruction MC_GearIn, herein referred to as the spindle 1
  ②By following the movement of the spindle axis from the instruction NS_MC_SpecialCamIn, herein referred to as the spindle 2
  therefore,

$$\mathrm{SlavePositionChange} = 1\#\mathrm{MasterPositionChange} * \frac{\mathrm{GearRatioNumerator\ M1}}{\mathrm{GearRatioDenominator\ M1}} + 2\#\mathrm{MasterPositionChange} * \frac{\mathrm{GearRatioNumerator\ M2}}{\mathrm{GearRatioDenominator\ M2}}$$

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| InSync (synchronized state) | This parameter is TRUE output shaft from the synchronized state represents | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| Pos_Min (minimum position) | Within one execution cycle, the position of minimum distance Slave axis of the cycles | LREAL | |

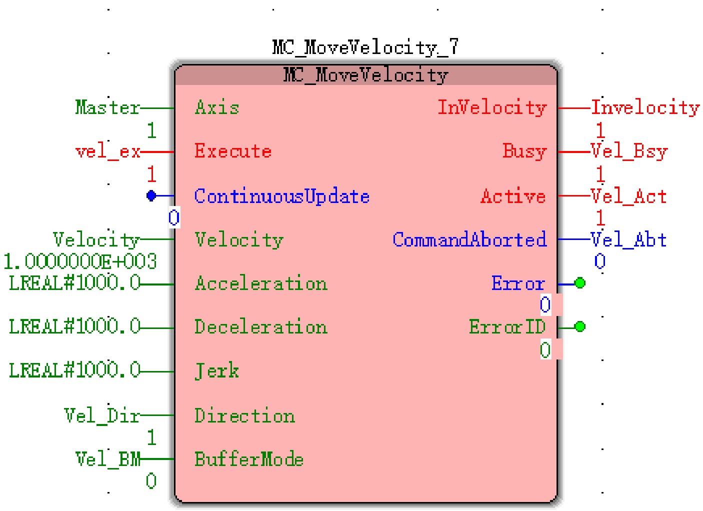
| | | | |
|---|---|---|---|
| | performed starting from<br>Unit: Unit | | |
| Pos_Max (maximum position) | Within one execution cycle, the execution cycle from the starting point Slave axis of the maximum position<br>Unit: Unit | LREAL | |

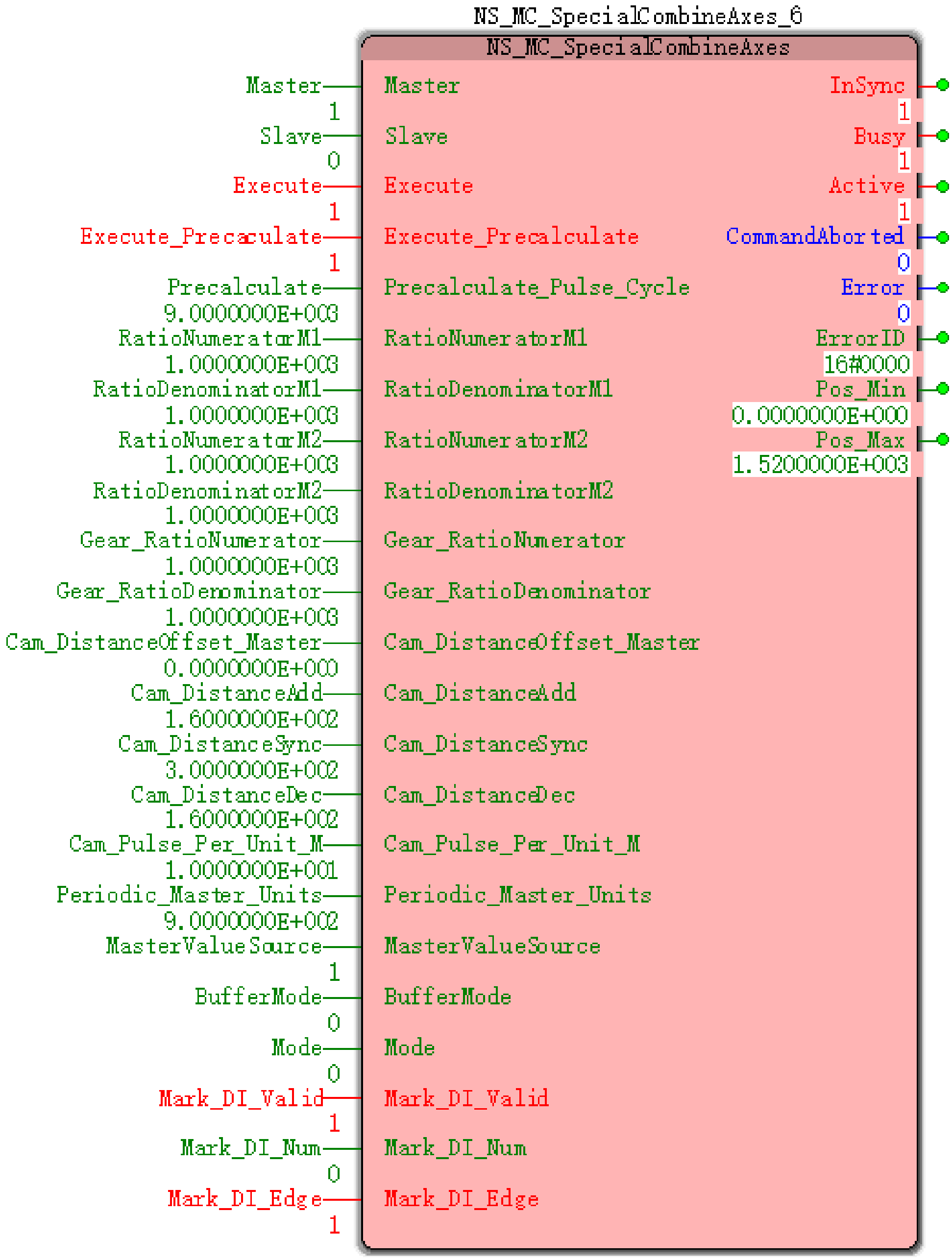
**Program Example**

In mode 0 instructions execute NS_MC_SpecialCamIn

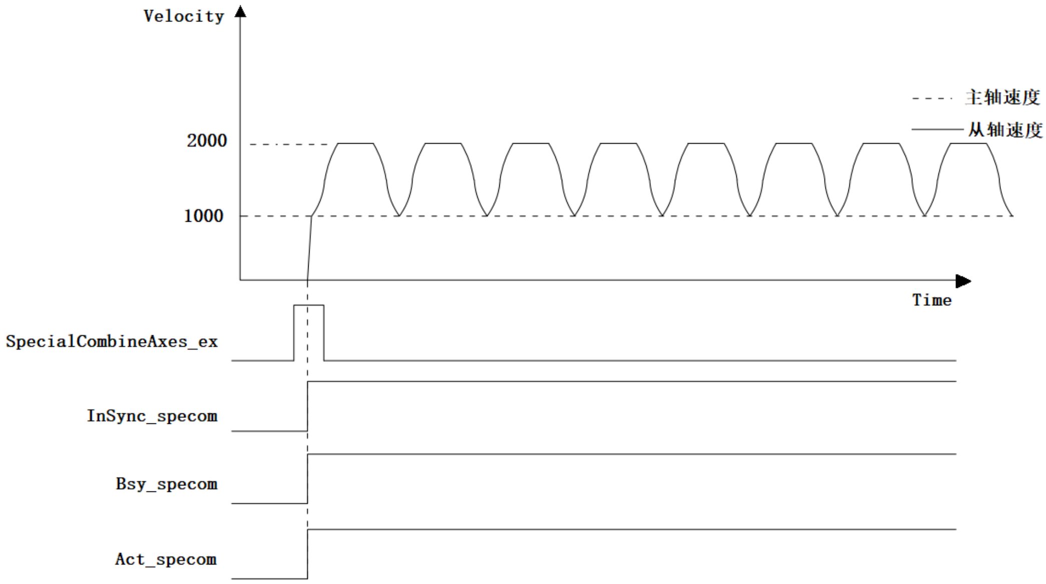
**1, procedures, and variables**

| variable name | type of data | The initial value |
|---|---|---|
| MC_MoveVelocity_7 | MC_MoveVelocity | |
| Master | USINT | 1 |
| Vel_Dir | INT | 1 |
| Vel_BM | INT | 0 |
| Master | USINT | 1 |
| Slave | USINT | 0 |
| Execute | BOOL | FASLE |
| Execute_Precalculate | BOOL | FALSE |
| Precalculate | LREAL | 9000.0 |
| RatioNumeratorM1 | LREAL | 1000.0 |
| RatioDenominatorM1 | LREAL | 1000.0 |
| RatioNumeratorM2 | LREAL | 1000.0 |
| RatioDenominatorM2 | LREAL | 1000.0 |
| Gear_RatioNumerator | LREAL | 1000.0 |
| Gear_RatioDenominator | LREAL | 1000.0 |
| Cam_DistanceOffset_Master | LREAL | 0.0 |
| Cam_DistanceAdd | LREAL | 160.0 |
| Cam_DistanceSync | LREAL | 300.0 |
| Cam_DistanceDec | LREAL | 160.0 |
| Cam_Pulse_Per_Unit_M | LREAL | 10.0 |
| Periodic_Master_Units | LREAL | 900.0 |
| MasterValueSource | INT | 0 |
| Mode_specom | INT | 0 |
| InSync_specom | BOOL | |
| Bsy_specom | BOOL | |
| Act_specom | BOOL | |
| Abt_specom | BOOL | |


MC_MoveVelocity_7
MC_MoveVelocity
Master
Axis
InVelocity
Invelocity
1
vel_ex
Execute
Busy
Vel_Bsy
1
ContinuousUpdate
Active
Vel_Act
0
Velocity
Velocity
CommandAborted
Vel_Abt
1.0000000E+003
LREAL#1000.0
Acceleration
Error
LREAL#1000.0
Deceleration
ErrorID
LREAL#1000.0
Jerk
Vel_Dir
Direction
Vel_BM
BufferMode



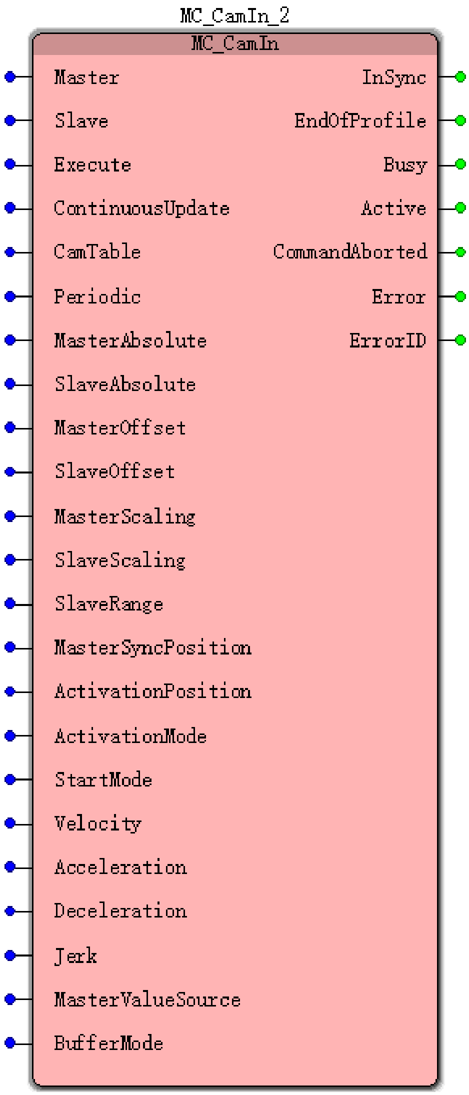

**2, a timing graph and**




- Spindle speed 1000, execution NS_MC_SpecialCombineAxes, made in accordance with the motion from the shaft of the cam follower shaft position change table.

## 11.5.12 MC_CamIn (electronic cam associated instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This relationship established instructions for electronic cam shaft between two | VEC-VA-MP-005-MA |




➢ **Input parameters**

| **Name** | **Features** | **type of data** | **Predetermined area (Default value)** | **The timing of the entry into force** |
|---|---|---|---|---|
| Master (Spindle) | Set electronic cam shaft | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen** | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | | | **mode**: 0-15 (real axis / imaginary axis) (0) | |
| Slvae (Slave axis) | Set from the electronic cam shaft | USINT | 0-4 the real axis 5 to 11 virtual axis (CANopen Mode: 0 ~ 15, can be real or imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | |
| ContinousUpdate (Reserved) | . | | | |
| CamTable (Electronic cam table number) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |
| Periodic (Cycle Sport) | Setting electronic cam operating cycle of a cycle or run only | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| MasterAbsolute (Spindle absolute) | Setting spindle position mode: When TRUE, the absolute position of the spindle mode; to FALSE, the relative position of the spindle mode | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| SlaveAbsolute (Absolute slave axis) | Mode is set from the position of the axis: When TRUE, the main mode is the absolute mode position; to FALSE, the spindle opposite position mode to mode | BOOL | TRUE or FALSE | Exexcute from FALSE to TRUE |
| MasterOffset (Spindle position offset) | Setting spindle position offset (Unit: unit) | LREAL | Positive, negative, 0 (0) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| SlaveOffset (Offset Slave axis position) | Setting a position offset Slave axis (Unit: unit) | LREAL | Positive, negative, 0 (0) | Exexcute from FALSE to TRUE |
| MasterScaling (Spindle position zoom ratio) | Set the scale of the spindle position | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| SlaveScaling (Scaling ratio Slave axis position) | Axis scale is set from the position | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| SlaveRange (Slave axis of the cam phase range) | Setting range from the phase of the cam shaft | LREAL | | |
| MasterSyncPosition (Reserved) | Retention | | | |
| ActivationPosition (Engagement start position) | Setting the engagement process begins when the main shaft position, i.e. the position when the spindle passes, from the shaft engagement operation started (Unit: unit) | LREAL | Positive, negative, 0 (0) | Exexcute from FALSE to TRUE |
| ActivationMode (Mode boot mode) | Engagement start position setting mode | INT | 0: the relative position of the axis<br>1: Absolute shaft position<br>2: Absolute phase axis<br>3: Absolute cam phase<br>(0) | Exexcute from FALSE to TRUE |
| StartMode (Engaging mode) | Performing engagement operation mode is set from the shaft | INT | 0: The shortest distance<br>1: Forward<br>2: Reverse | Exexcute from FALSE to TRUE |
| Velocity (speed) | Maximum meshing operation setting execution process allows stacking velocities from the shaft (Unit: unit / S) | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Acceleration (Acceleration) | Maximum engagement setting operation performed during | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | the acceleration from the shaft to allow superimposition (Unit: unit / S2) | | | |
| Deceleration (decrease speed) | Setting execution during the engagement operation of the maximum allowed deceleration is superimposed from the shaft (Unit: unit / S2) | LREAL | A positive number (Non-default) | Exexcute from FALSE to TRUE |
| Jerk (Plus / deceleration rate of change) | Retention | | | |
| MasterValueSource (Spindle position source) | Electronic cam type setting spindle position calculation process | INT | 0: Follow the spindle axis from a position command 1: the actual position of the shaft from the spindle to follow | Exexcute from FALSE to TRUE |
| BufferMode (Transfer mode) | Setting the transfer mode between the two instructions | INT | 0: immediately interrupted 1: Wait (0) | Exexcute from FALSE to TRUE |

**Description:**

1,The instruction to execute upon the Execute FALSE to TRUE. The instruction is being executed, ExecuteBy the TRUEeWhen going to FALSE, no effect on the implementation of the Directive.

2, when the instruction is being executed, BufferMode = at 0, Execute FALSE to TRUE when the re, the instructions may immediately interrupt their own.

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| InSync (synchronized state) | This parameter is TRUE output shaft from the synchronized state represents | BOOL | TRUE or FALSE |
| EndOfProfile (end of the cam execution flag) | This parameter indicates the output end of the cam is TRUE is performed | BOOL | TRUE or FALSE |

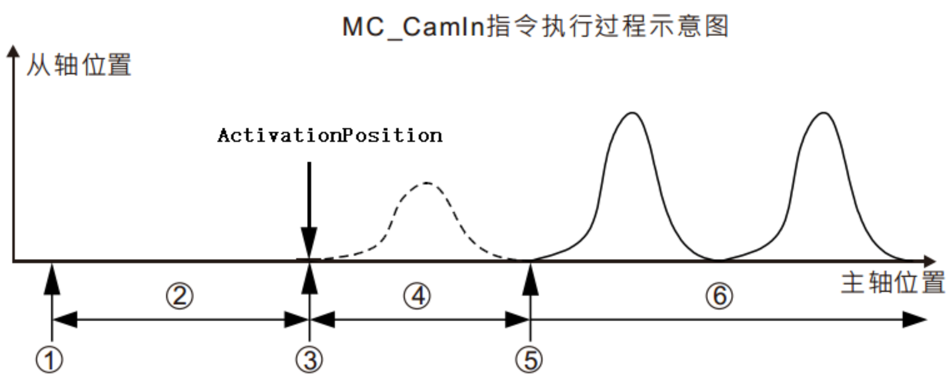
| | | | |
|---|---|---|---|
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Active | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted | This parameter indicates the output instruction execution is interrupted to TRUE | BOOL | TRUE or FALSE |
| Error | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| ErrorID | Command execution error is the error code | WORD | |

- **Function Description:**
- MC_CamIn implemented instructions for controlling a synchronous movement of the cam from the cam shaft in accordance with a preplanned relationship with the spindle;
- MC_CamOut instructions for the release cam relationship.

### Brief instruction MC_CamIn

#### ✧ Instruction execution flow MC_CamIn

MC_CamIn instruction execution flow as shown below:




stage①: Trigger MC_CamIn instruction execution

stage②: Wait engagement start

stage③: Engaging the spindle reaches the start position, the shaft engagement operation started

stage④: During engagement

stage⑤: Complete engagement, the main shaft from the synchronization

stage⑥: Synchronized movement from the main shaft

**stage①:**Trigger MC_CamIn instruction execution

MC_CamIn instruction execution at the moment, will immediately enter the shaft from the engagement start wait state.

**note:**If ActivationPosition ActivationMode is 0 and is 0 (location relative to the shaft), the current speed from the shaft will be moving toward the synchronous speed, in addition, from immediately moving axis stops! The MC_CamIn instruction input parameters at the moment is read and locked instruction, the instruction for use during execution.

**stage②:**Wait for the engagement start

From the shaft in a stationary state, waiting to begin execution timing of the arrival of the engaging operation, i.e., after waiting for the spindle ActivationPosition location specified parameters. From the waiting time axis will be different in different circumstances, if the instruction to start execution MC_CamIn spindle ActivationPosition i.e. at the location specified parameters, the slave axis starts executing the engaging operation; never have a chance if the spindle reaches the specified parameter ActivationPosition position, the slave axis will never be able to begin engagement never be achieved cam synchronization. Parameters ActivationPosition, ActivationMode role at this stage.

**stage③:**Engaging the spindle reaches the start position, the shaft engagement operation started

When the spindle passes ActivationPosition location specified parameters, started from the shaft engagement operation. Parameters MasterAbsolute, SlaveAbsolute, MasterOffset, SlaveOffset, MasterScaling, SlaveScaling onset of action in the moment, for the correspondence relationship between the master axis position of the shaft from the cam phase of its determination.

**stage④:**During engagement

StartMode engagement operation performed by the parameter Slave axis in the manner specified. In addition to the StartMode parameters, parameters Velocity, Acceleration, Deceleration is also applied at this stage, they will determine the meshing process, from the shaft speed, acceleration / deceleration motion characteristics of these items.

**stage⑤:**Complete engagement, the main shaft from the synchronization

After the engagement operation started from the shaft, the main shaft if the corresponding cam phase satisfy the relationship of the cam planning, completion of the engagement, to achieve synchronization of the cam shaft from the spindle.

**Description:**The figure shows only the case when the engagement start position of the spindle is greater than MC_CamIn instruction starts execution timing of the main shaft position, and the case is equal to less than the same way may be derived.

- ActivationPosition

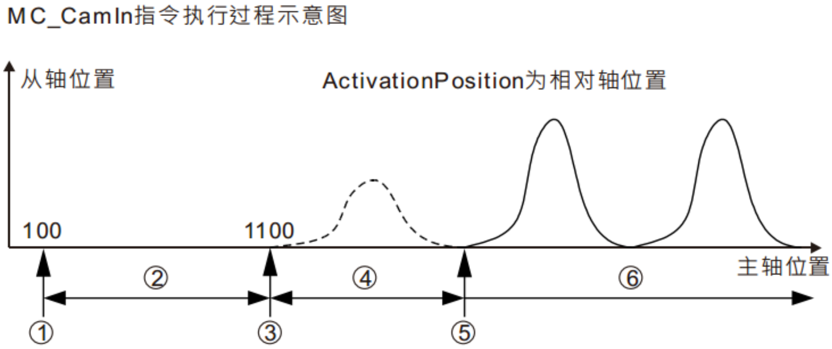
Start position parameter ActivationPosition cam engaged (the position of the spindle "position"), i.e., the right trigger MC_CamIn instruction execution and the main shaft reaches ActivationPosition, engagement operation started from the shaft.

ActivationPosition may be: the position of the spindle, the phase of the spindle, the spindle cam phase parameter selected by ActivationMode.

- **ActivationPosition relative axial position**

When time parameters ActivationMode = 0, ActivationPosition axis position, and the position of the spindle MC_CamIn instruction starts execution time relative relationship, i.e. the actual position of engagement of the spindle at the start position of the spindle MC_CamIn instruction starts execution time plus ActivationPosition.

For example: MC_CamIn instruction starts execution time of the position of the spindle 100, ActivationPosition 1000, the actual position of engagement of the spindle at the start of 1100 (1100 + 100 = 1000).




stage①: Trigger MC_CamIn instruction is executed, this time to the absolute position of the spindle 100

stage②: Wait engagement start

stage③: Engaging the spindle reaches the start position (1100), engagement operation started from the shaft

stage④: During engagement

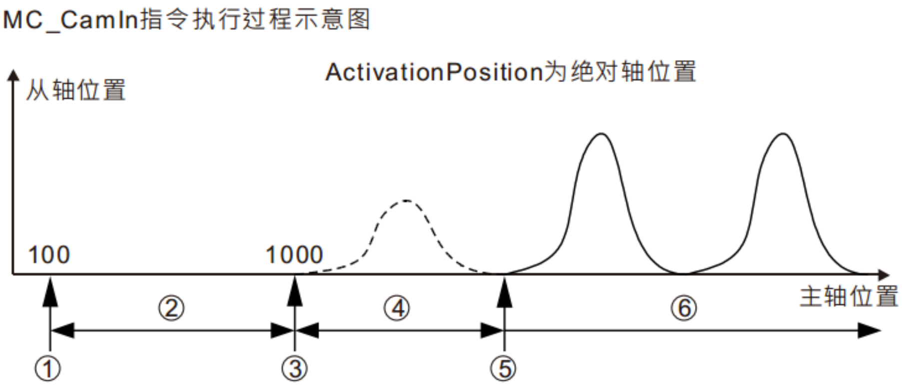
stage⑤: Complete engagement, the main shaft from the synchronization

stage⑥: Synchronized movement from the main shaft

- **ActivationPosition the absolute axis position**

When the parameter ActivationMode = 1, ActivationPosition axis position, and the position of the spindle MC_CamIn instruction starts execution time absolute relationship, i.e. the actual position of the spindle engaging at the start of ActivationPosition.

For example: MC_CamIn instruction starts execution time of the position of the spindle 100, ActivationPosition 1000, the actual position of engagement of the spindle at the start of 1000 (1000 = ActivationPosition).



stage①: Trigger MC_CamIn instruction is executed, this time to the absolute position of the spindle 100
stage②: Wait engagement start
stage③: Engaging the spindle reaches the start position (1000), engagement operation started from the shaft
stage④: During engagement
stage⑤: Complete engagement, the main shaft from the synchronization
stage⑥: Synchronized movement from the main shaft

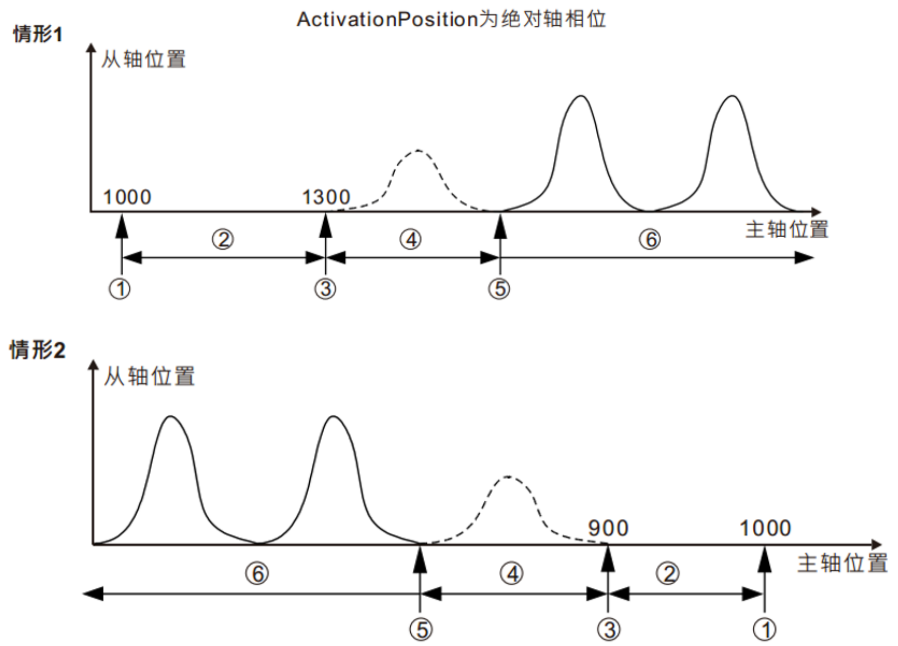
- **Absolute phase axis ActivationPosition**

When the parameter ActivationMode = 2, ActivationPosition absolute phase axis (axis absolute position of the absolute phase axis done modulo result of modulo arithmetic). The absolute phase when the spindle shaft is ActivationPosition, started from the shaft engagement operation.

The shaft having absolute phase characteristics cycles, during operation of the spindle, which is equal to the absolute-axis phase ActivationPosition case may appear several times, but only after MC_CamIn instruction starts execution, the absolute phase of the spindle axis is equal to the first ActivationPosition, beginning from the shaft perform engagement operation.

For example: mold of the main shaft 400, ActivationPosition = 100, MC_CamIn instruction starts execution timing of the main shaft position 1000, due to the timing of the spindle MC_CamIn instruction starts execution of the absolute phase of the shaft 200 (200 = 400 1000%), from the shaft does not execute the engagement action. Thereafter, when the position of the main shaft 1300 (the absolute phase of the shaft 100 1300 = 400%) or 900 (100 is the absolute phase axis 400 = 900%), from the shaft engaging operation started (% denotes a remainder operation)

MC_CamIn指令执行过程示意图



stage①: Trigger MC_CamIn instruction execution, the absolute position of the spindle at this time is 1000 (the absolute phase of the shaft 200)

stage②: Wait engagement start

stage③: Engaging the spindle reaches the start position (1300 case 1, case 2 to 900), the shaft engagement operation started

stage④: During engagement

stage⑤: Complete engagement, the main shaft from the synchronization

stage⑥: Synchronized movement from the main shaft

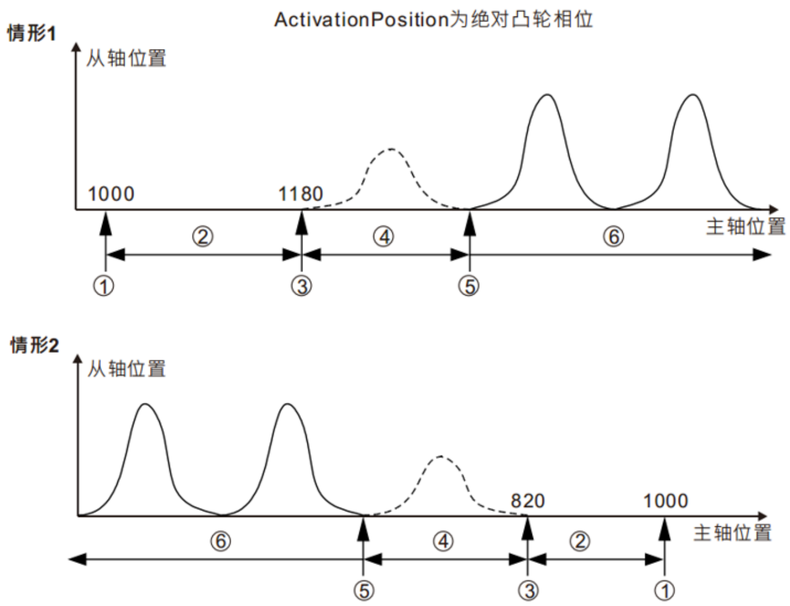
**note:**When ActivationPosition absolute phase axis, ActivationPosition effective range parameter is: 0 ~ mold (not including mold). If values are not within the valid range of the parameter ActivationPosition when MC_CamIn instruction execution, and the error fails!

- **ActivationPosition absolute cam phase**

When the parameter when ActivationMode = 3, ActivationPosition absolute cam phase (absolute phase of the cam axis absolute position its cam cycle do result of modulo operation). When the phase of the cam shaft ActivationPosition, started from the shaft engagement operation.

Cycles having cam phase characteristic, the main shaft during operation, it may appear more than equal to the phase of the cam

When ActivationPosition case, but only after MC_CamIn instruction execution starts, the spindle is equal to the first cam phase ActivationPosition, engagement operation started from the shaft. For example: the maximum range of the cam shaft value table 360, ActivationPosition = 100, MC_CamIn instruction starts execution timing of the main shaft position 1000, due to the spindle MC_CamIn instruction starts execution time absolute phase of the cam 280 (280 = 360 1000%), from shaft engagement operation is not performed. Thereafter, when the spindle is 1180 when the position (the absolute phase of the cam 1180% 100 = 360) or 820 (100 is the absolute phase of the cam 360 = 820%), the engagement operation started from the shaft.



stage①: Trigger MC_CamIn instruction execution, the absolute position of the spindle at this time is 1000 (the absolute phase of the cam 280)

stage②: Wait engagement start

stage③: Engaging the spindle reaches the start position (position 1180 of the spindle case 1, case 2 spindle position 820), the shaft engagement operation started

stage④: During engagement

stage⑤: Complete engagement, the main shaft from the synchronization

stage⑥: Synchronized movement from the main shaft

**note:** When the absolute phase ActivationPosition cam ActivationPosition effective range parameter is: 0 to cam cycle (period not including the value). If values are not within the valid range of the parameter ActivationPosition when MC_CamIn instruction execution, and the error fails!

- **From the relationship between the master axis position**

Relationship between pre-planning software cam master-slave relationship between the position of the axis, "position" herein from the main phase of the cam shaft, rather than the actual axis position. If the relationship between the cam as a function of pre-planned CAM, CAM input is a function of the phase of the cam shaft, the output shaft from the cam phase, as follows:

y = CAM (x)

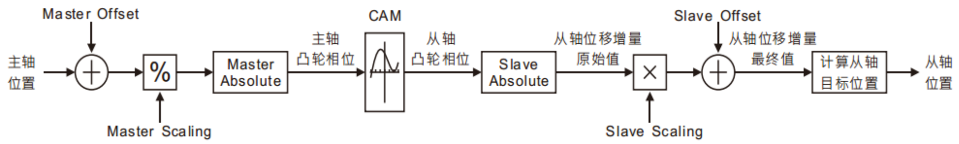
x: spindle cam phase

y: the phase of the cam shaft from

From the phase position of the cam shaft, there is a conversion between them. MasterAbsolute parameter conversion relationship between the shaft position and the cam phase, SlaveAbsolute, MasterOffset, SlaveOffset, MasterScaling, SlaveScaling, please refer to the relevant details.

Cam follower shaft from the spindle do MC_CamIn synchronized movement under the action of the instruction. Synchronous movement of the cam, Slave axis position of the main shaft of the cam to establish correspondence between pre-planning relationship (cam curve or cam table) based on the calculated position of the spindle axis from the position of the shaft

Process is shown below:




- **MasterAbsolute and SlaveAbsolute**

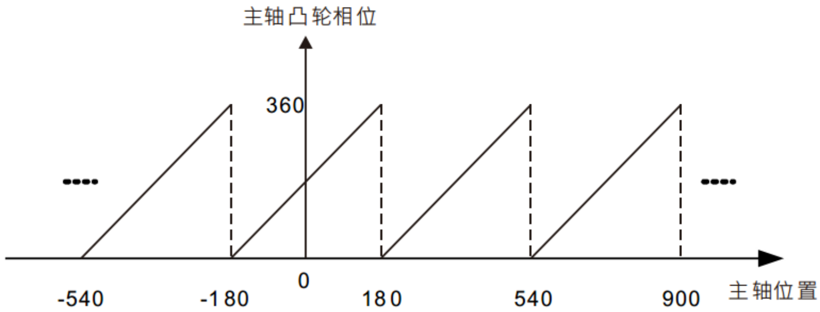
MasterAbsolute correspondence relationship between the parameters used to specify the position of the spindle axis and its cam phase: When the parameter is TRUE, the absolute relationship; when the parameter is FALSE, the relative relationship. SlaveAbsolute MasterAbsolute parameters and empathy.

MasterAbsolute SlaveAbsolute parameters and acting on the engagement start timing, that is, a correspondence relationship between the shaft position and the cam phase to establish engagement start timing (note: the corresponding relationship is established by engaging actions of the start time, rather than MC_CamIn instruction starts execution time).

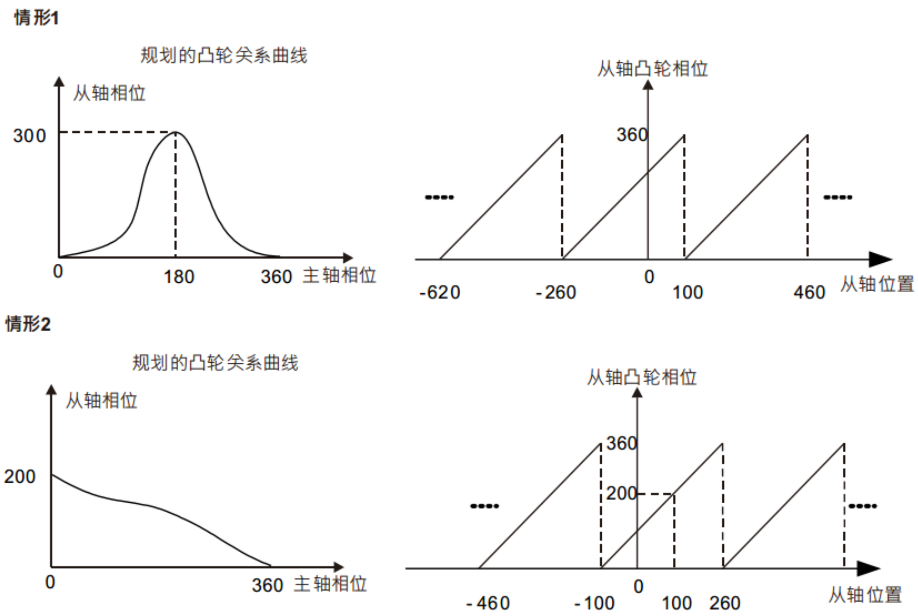
After this, the cam phase calculation, in accordance with the correspondence relation.

- **Relative mode**

When MasterAbsolute axis position parameter is FALSE, between the axis position of the spindle relative to its cam phase relationship, i.e., start time of the spindle engaging a corresponding cam phase thereof is 0, and thereafter, the spindle cam phase is calculated, in accordance with the correspondence relation will be. For example: spindle relative mode, the relationship between the cam spindle 360 is the maximum range, the engagement start time axis position of the main shaft 180, the axis position of the spindle 180 which corresponds to the cam phase is 0, the position of the shaft 200 which corresponds to a phase of the cam 20 ( 20 = (200-180) 360%), and so on. In this case, the relationship between the cam and its phase axis position of the spindle (spindle position) as shown below:

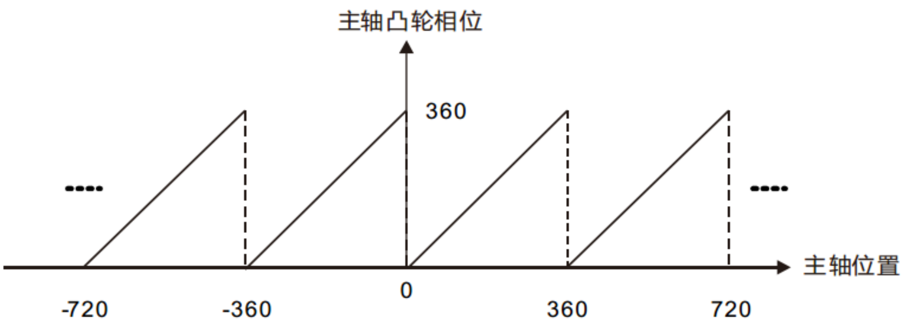



When SlaveAbsolute parameter is FALSE, the position of the shaft between the shaft and its opposing relationship to the cam phase, i.e. the phase of the engagement start timing of the cam shaft at the moment planning phase satisfy the relationship of the cam from the cam shaft. When the axis is in relative mode, the method determines Slave axis of the spindle cam phase different from the phase axis is determined when the cam must satisfy the condition: the engagement start timing phase from the cam shaft and the cam at the moment the phase of the cam shaft satisfies planning. For example: From axis relative mode, the cam shaft from the relationship between the maximum value of the range 360, from the start time of the engagement position of the shaft axis 100, at the moment when the spindle cam phase is 0 (according to the requirements from the phase relationship of the cam shaft of the cam is 0), shaft 100 from a position corresponding to which the cam phase is 0, as shown. 1 case; if by a cam shaft from the cam phase relation as in claim 200, 100 from the shaft position which corresponds to a phase of the cam 200, as the case of FIG. 2 shows.




- **Absolute Mode**

  When the cam shaft position relationship MasterAbsolute parameter is TRUE, the position of the spindle shaft between the cam phase and its absolute relationship, at any time, the cam phase of the spindle is equal to the time the main shaft spindle maximum range value The result of the modulo operation. For example: spindle absolute mode, the cam spindle relationship is the maximum range 360, the axis position of the main shaft 100, a cam phase which is 100 (= 100% 100 360); the position of the shaft 500 of the main shaft, which cam a phase of 140 (= 500% 140 360), and so the relationship between the spindle axis position with its cam phase as shown.



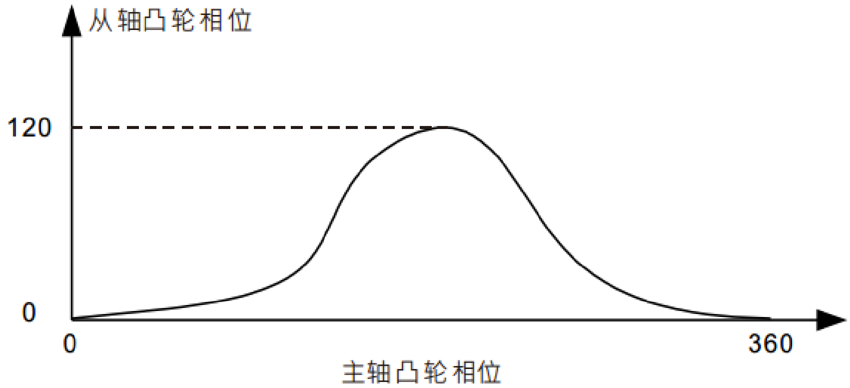

When SlaveAbsolute parameter is TRUE, the position of the shaft between the shaft and its absolute cam phase relationship, at any time, from the phase of the cam shaft is equal to the time to make the position of the cam shaft from a relationship between the shaft axis from a maximum range value modulo result of the operation. When the absolute mode Slave axis, consistent with the correspondence relationship between the phase of its spindle axis position and its cam.

- **Scaling and position misalignment (Offset and Scaling)**

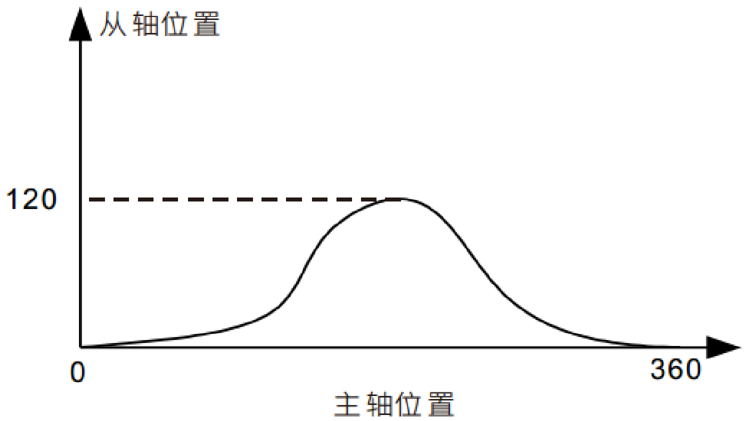
  From the relationship between the main axis of the cam advance planning, but the implementation of the cam, the position shift may be pre-planned based on a cam relationship with the parameter "Offset" and "Scaling" position or scale, for example: the same processing products are several different sizes, the only one kind of cam planning relation, and by changing the parameters "Offset" and "Scaling" to accommodate the processing of switching between different sized products.

MasterOffset absolute mode parameters are valid for the relative or spindle; SlaveOffset parameter is valid only when the axis is absolute mode (SlaveAbsolute = TRUE), the invalid axis relative mode (SlaveAbsolute = FALSE).

Together determine the relationship between the main cam and the actual execution of positional deviation from the scaling ratio of the shaft, which effects will be described by the following examples. Cam advance planning relationship as shown below:

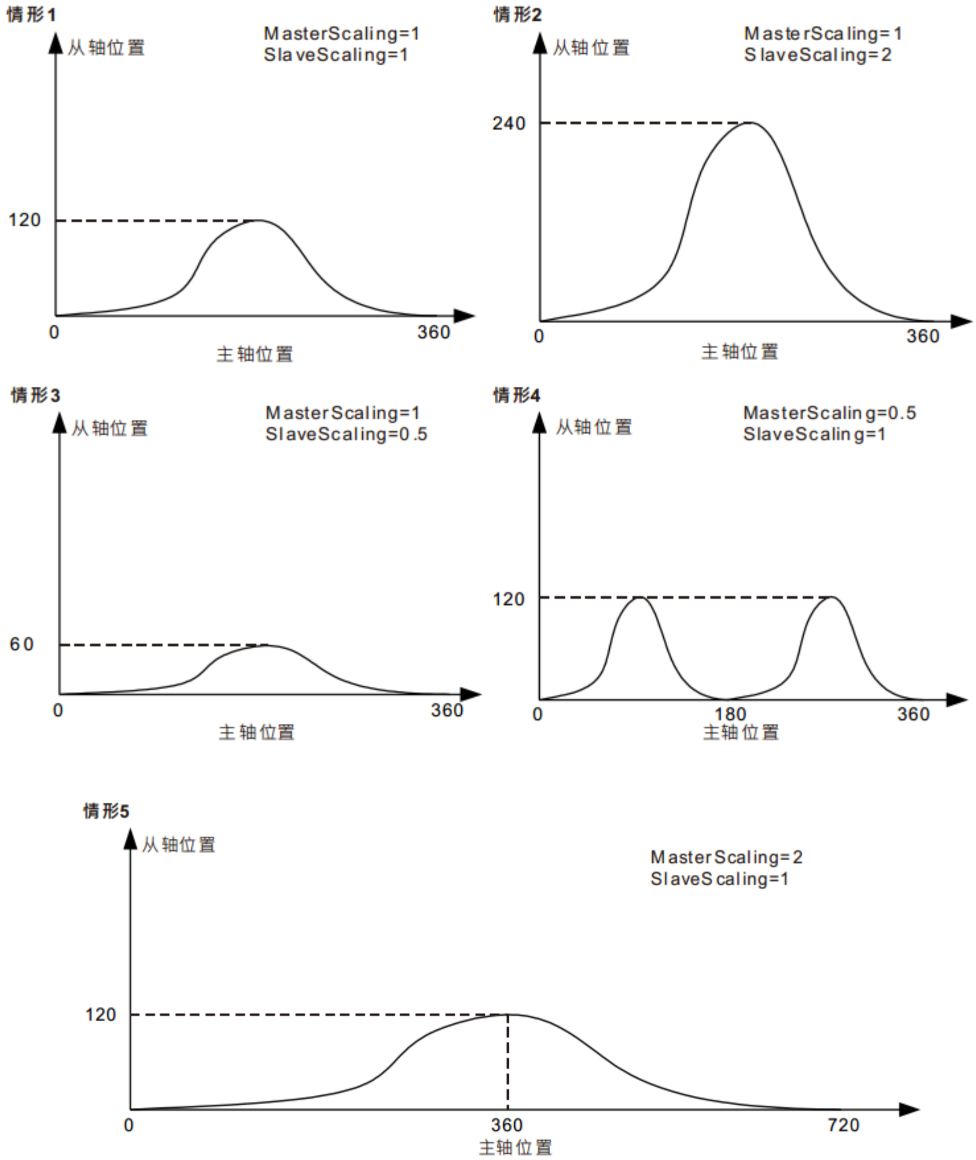



When the main shaft when both the absolute mode, and performs the engagement operation, the main are 0, and scaling without using the offset from the position of the shaft axis (the default), the main execution of the real position of the cam shaft corresponding relationship As shown below:




When the position of the offset or scale is not the default value, as the corresponding impact relations from the actual position of the main cam execution:

- **From the main axis offset0Effect of the main axis of the zoom ratio from the relation of the actual implementation of the cam**




**Case 1:**When the main shaft from the zoom ratio is 1, the offset is zero, consistent with the actual relationship between the cam and the pre-planned.

**Case 2:**When the spindle is a scale, zoom ratio from the shaft 2, from the primary offset of 0, corresponding to the position of the spindle axis from the position to the pre-planned times.

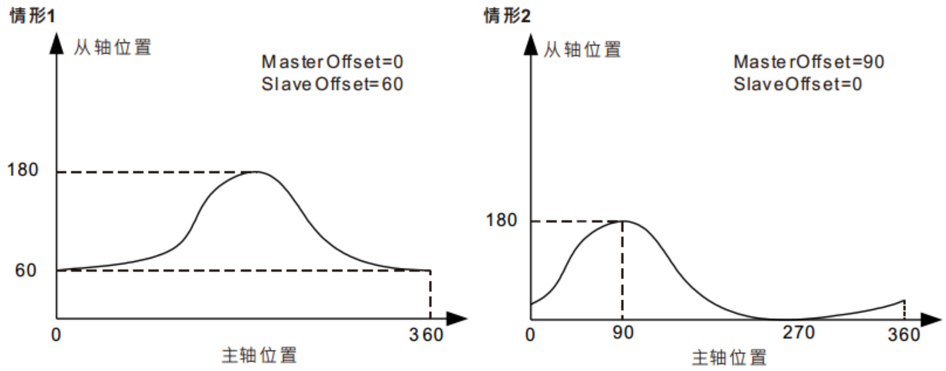
**Case 3:**When the spindle is a scale, the scale axis ratio of 0.5, a main axis offset from 0, corresponding to the position of the spindle becomes a pre-planned position of 1/2 Slave axis.

**Scenario 4:**When the spindle zoom ratio of 0.5, the zoom ratio from the shaft 1, a main shaft cam offset from 0, the corresponding position of the spindle axis from the pre-planned position to 1/2. If from the perspective of the cam phase, the cam phase of the spindle is preplanned 1/2, i.e., the cam shaft 360 goes from 180 cycles (360 * 180 = 0.5), unchanged from the phase of the cam shaft.

**Scenario 5:**When the scale of the spindle 2, the zoom ratio from the shaft 1, a main axis offset from 0, the corresponding position of the spindle axis from the position to the pre-planned times. If from the perspective of the cam phase, the cam is twice the original phase of the spindle, i.e., the cam shaft 360 goes from 720 cycles (720 * 2 = 360), Slave axis of the cam phase constant.

- **Scaling from affecting the main axis ratio of 1, the relationship between the main cam axis offset from the actual implementation of the**

Spindle offset curve corresponds to the actual axis position to be performed when the cam moves laterally; axis offset Slave axis position corresponding to the execution of longitudinal movement of the cam curve.

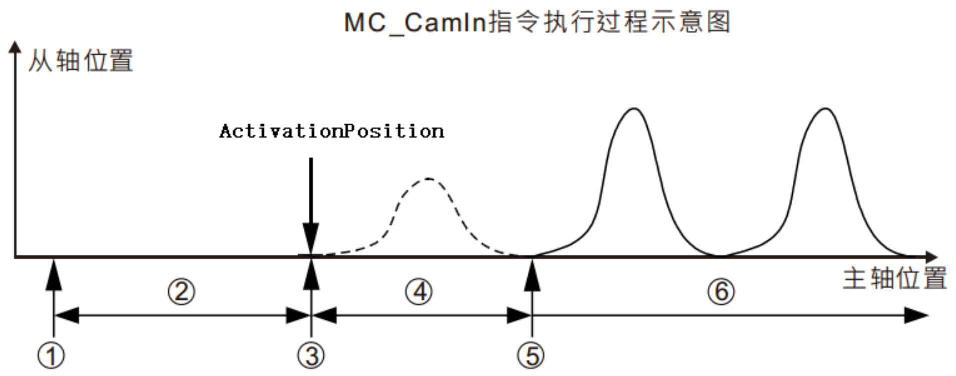



**Case 1:**When the main shaft from the zoom ratio is 1, the spindle offset 0, offset from the shaft 60, the spindle 60 from the position corresponding to the coupled position are in the shaft based on the pre-planned. For example: a cam network plan, the position of the spindle shaft 180 from the shaft corresponding to position 180, the actual implementation, the corresponding position Slave axis of the shaft 240 (240 + 60 = 180).

**Case 2:**When the main shaft from the zoom ratio is 1, the spindle 90 is offset Slave axis offset of 0, with the axis of the shaft from the main shaft axis position corresponding to a position shifted by 90 (plus the offset amount) on the basis of pre-planned . For example: a cam network plan, the spindle axis position 180 corresponding to the shaft position is 180, the actual implementation, a spindle axis position 90 corresponding to the shaft position 180, cam relationship i.e., pre-planned spindle axis position 180 ( 180 = 90 + 90) corresponding to the shaft position.

## ■ StartMode

Meshing process, the parameters can be specified by the operation mode StartMode axis, i.e. StartMode acting on the instruction stage MC_CamIn④,As shown below:




stage①: Trigger MC_CamIn instruction execution

stage②: Wait engagement start

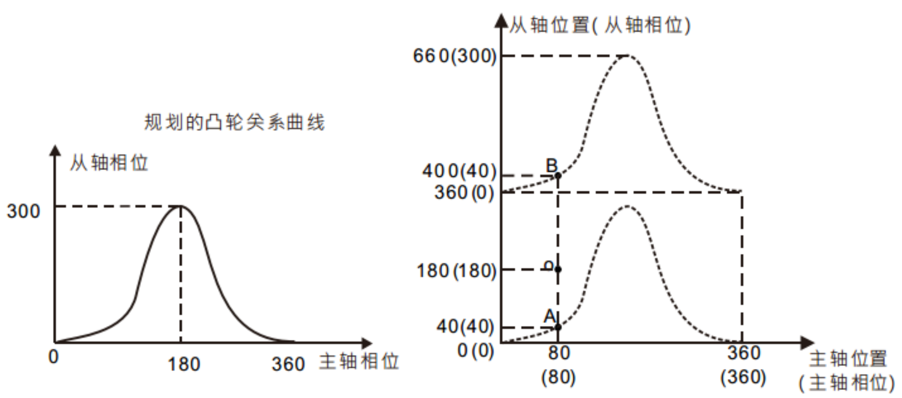
stage③: Engaging the spindle reaches the start position, the shaft engagement operation started

stage④: During engagement

stage⑤: Complete engagement, the main shaft from the synchronization

stage⑥: Synchronized movement from the main shaft

Sync request from the master cam of the cam shaft of the cam phase satisfy the relationship defined by the engagement process was synchronized phase process from moving toward the axis of the spindle and the cam phase synchronous phase satisfy the relationship defined by the cam. Since the phase of the cam shaft having a cyclical characteristic, i.e., each cam has a plurality of phase-axis position corresponding thereto, when engaged, there are alternative desired synchronization position, there is such a wide selection of engaging manner. For example: the start of execution engagement cam from the phase of the main axis are 80 and 180 (e.g., O lower right in the drawing), but the requirements defined by the phase relationship of the cam from the cam shaft 40, the shaft from a desired moment the synchronization position 40 or 400 (e.g., the bottom right point a and point B), can be from O to a or B. O to the meshing process parameter selection StartMode




StartMode There are three modes available, namely: the shortest distance gradient (StartMode = 0), a positive gradient (StartMode = 1) and the reverse gradient (= StartMode -1), the user can select a different mode according to the actual needs of engagement.

- **StartMode = 0 (the shortest distance gradient)**

  If the parameter StartMode = 0, the engagement operation is executed, and the synchronization position Slave axis in the direction of the shortest distance, this time from the motion axis parameters Velocity, Acceleration, Deceleration affected by.

- **StartMode = 1 (positive gradient)**

  If the parameter StartMode = 1, the engaging operation is performed, the forward toward the shaft from the synchronous position, this time from the motion axis parameters Velocity, Acceleration, Deceleration affected by.

- **StartMode = -1 (inverse gradient)**

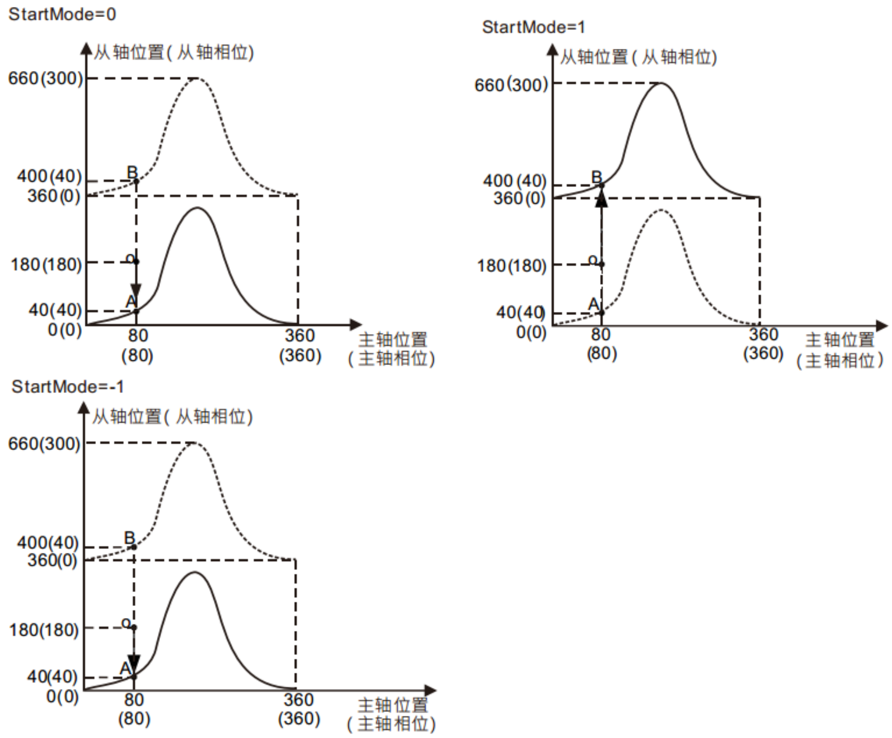
  If the parameter StartMode = -1, the engagement operation is performed, the reverse position Slave axis toward synchronization,, Acceleration, Deceleration Effect this time from the movement of the shaft receiving parameters Velocity.

**E.g:**Starts performing engagement operation, the master from the cam phase axis, respectively 80 and 180 (below point O), The cam relation defined in claim spindle cam phase is 80, the phase Slave axis of the cam 40 (as in FIG. when point a or point B), then select a different mode StartMode engagement process, the operation mode from the shaft as shown.

**StartMode = 0:**Now gradient Slave axis O from the point A to the point A at the synchronization point, because the distance from point O to the point A is smaller than the distance between the point B is O;

**StartMode = 1:**Now the axis positive gradient from point O to point B;

**StartMode = -1:**Now the reverse gradient Slave axis from the point A to the point O;

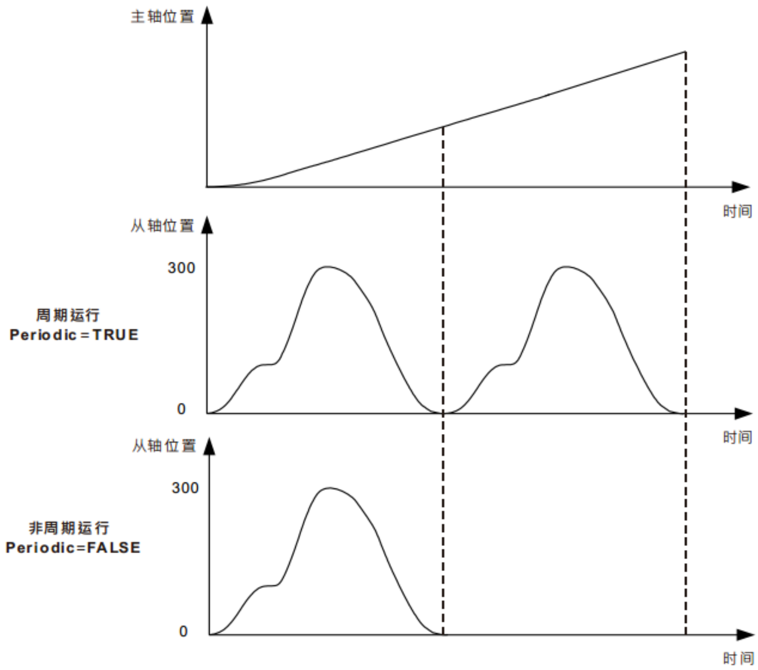



- **Acyclic / cyclic execution cam (with the Periodic)**

  In practical applications, the electronic cam, and some may require circulation operation in cycles, while others may only need to run a cycle, i.e. for these two parameters Periodic selected situations.

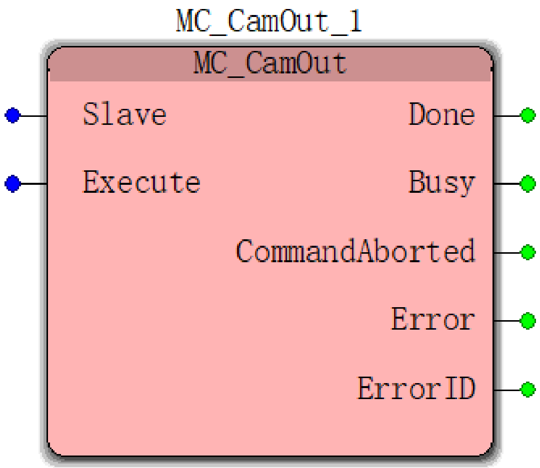
  When the parameter Periodic = TRUE, follow the master axis according to the execution cycles of the cam until the cam releasing relationship;

When the parameter Periodic = FALSE, from the cam shaft and the spindle synchronization, performed when the end point of the cam cycle, the relationship between the cam shaft and is released from the spindle, and immediately stop the movement from the shaft.


主轴位置
时间
从轴位置
300
周期运行
Periodic=TRUE
0
时间
从轴位置
300
非周期运行
Periodic=FALSE
0
时间

## 11.5.13 MC_CamOut (electronic cam departing instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to release a relationship between two electronic cam shafts established | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Slvae (Slave axis) | Set from the electronic cam shaft | USINT | **Analog / Pulse**: 0-4 (real axis) 5 to 11 (imaginary axis) **CANopen mode**: 0-15 (real axis / imaginary axis) (0) | Exexcute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |

➢ **Output parameters**

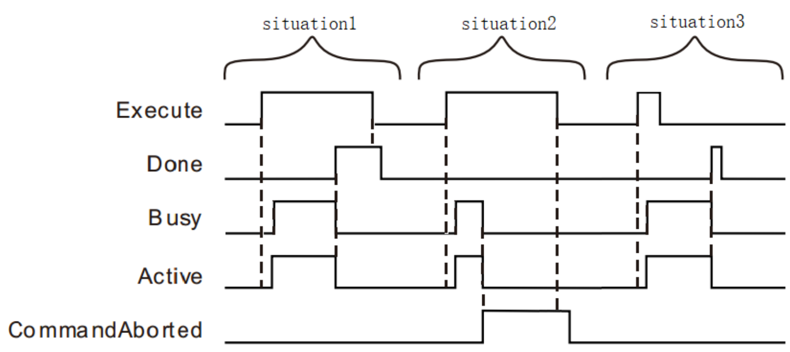
| name | Features | type of | Output range |
|---|---|---|---|

| | | data | |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

**Function Description**

MC_CamOut electronic cam instructions for releasing an established relationship. Directives continues to run at speed from the cam axis, and disengaged from the retaining shaft. For the control cam stop motion from the shaft, or may use MC_Halt MC_Stop instruction from the shaft. After completion of the instruction execution MC_Halt or MC_Stop Slave axis stops and the cam releasing relationship.

- **FIG output timing parameters**



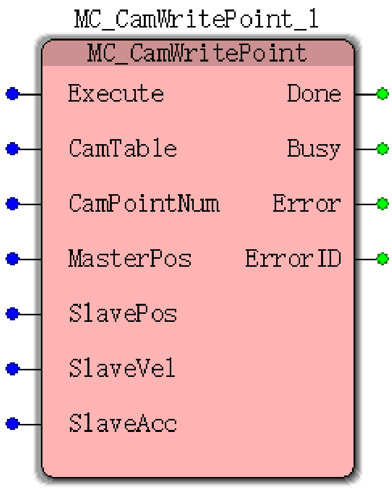

Case 1: When the Execute FALSE to TRUE, after a period, Busy, Done becomes TRUE. After Execute a TRUE to FALSE, Busy and Done remains to TRUE.

Case 2: When the Execute is TRUE, if the instruction is interrupted by another instruction, CommandAborted becomes TRUE, the Busy and Done becomes FALSE; Execute when a TRUE to FALSE, after a period CommandAborted becomes FALSE.

Case 3: In the course of the instruction execution, when less than one cycle, the Execute a TRUE to FALSE, when reaching a cycle, the Done becomes TRUE, and Busy remains to TRUE.

## 11.5.14 MC_CamWritePoint (cam point information write command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to write the cam point information | VEC-VA-MP-005-MA |




➢ **Input parameters**

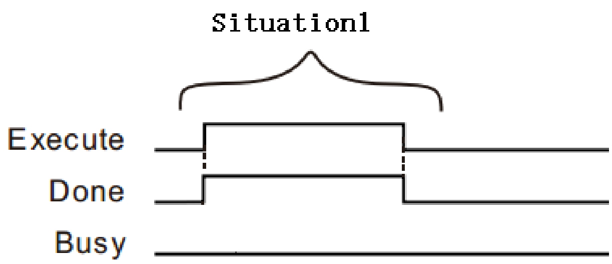
| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| CamTable (Electronic cam No.) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |
| CamPointNum (Cam point number) | Select the read point of the cam | UINT | | Exexcute from FALSE to TRUE |
| MasterPos | Set read command spindle position of the electronic cam point | LREAL | A positive number, 0 | MasterPos |
| SlavePos | Reading instruction is provided from an electronic point of the cam shaft position | LREAL | Positive, negative, 0 | SlavePos |

| | | | | |
|---|---|---|---|---|
| SlaveVel | Set read command from the electronic cam shaft speed point | LREAL | Positive, negative, 0 | SlaveVel |
| SlaveAcc | Set read command from the electronic cam shaft acceleration point | LREAL | Positive, negative, 0 | SlaveAcc |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG output timing parameters**



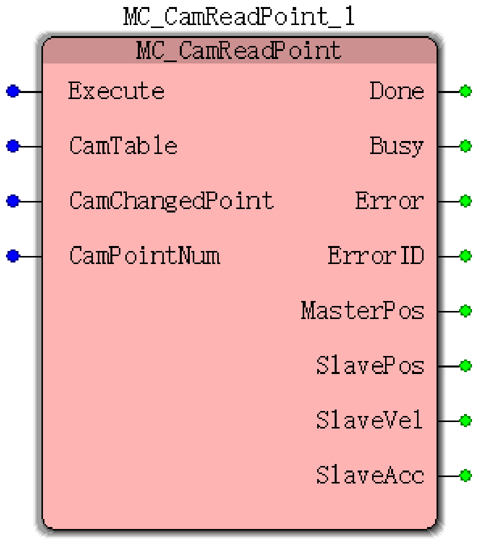

- Case 1: When the Execute FALSE to TRUE, while Done becomes TRUE, and if the time Execute to FALSE, FALSE becomes Done

**Function Description**

This instruction is used to write information in the electronic table cam of the cam point. After a successful write electronic cam point, the curve after the change does not take effect immediately, you need to change the instruction execution after MC_CamSet cam curve to take effect.

## 11.5.15 MC_CamReadPoint (cam point information reading instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the cam point information | VEC-VA-MP-005-MA |




➢ **Input parameters**

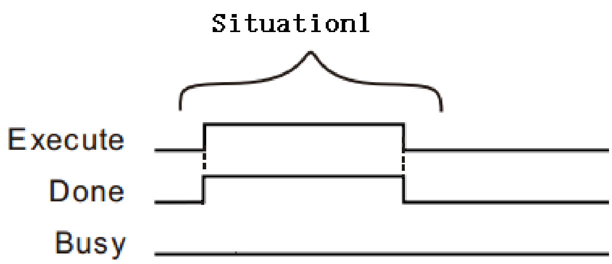
| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| CamTable (Electronic cam No.) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |
| CamChangePoint (Cam point before or after the information selection change) | When is FALSE, the command to read information before the change point of the cam; When TRUE, the command reads the information after the change point of the cam. | BOOL | TRUE / FALSE | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| CamPointNum (Cam point number) | Select the read point of the cam | UINT | | Exexcute from FALSE to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| MasterPos | Set read command spindle position of the electronic cam point | LREAL | A positive number, 0 |
| SlavePos | Reading instruction is provided from an electronic point of the cam shaft position | LREAL | Positive, negative, 0 |
| SlaveVel | Set read command from the electronic cam shaft speed point | LREAL | Positive, negative, 0 |
| SlaveAcc | Set read command from the electronic cam shaft acceleration point | LREAL | Positive, negative, 0 |

➢ **FIG output timing parameters**




➢ Case 1: When the Execute FALSE to TRUE, while Done becomes TRUE, and if the time Execute to FALSE, FALSE becomes Done

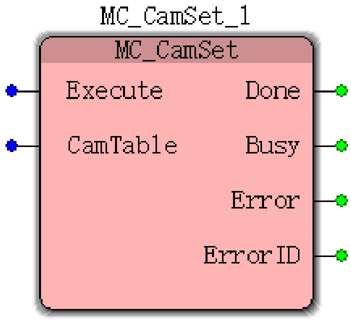
■ **Function Description**

This instruction is used to read information in an electronic cam CAM table points. When

CamChangedPoint is FALSE, the read information before the cam point instruction MC_CamSet changes, when CamChangedPoint is TRUE, the read point of the cam changes MC_CamSet instruction information.

## 11.5.16 MC_CamSet (changes to take effect cam point instructions)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to change the point of entry into force of the cam | VEC-VA-MP-005-MA |



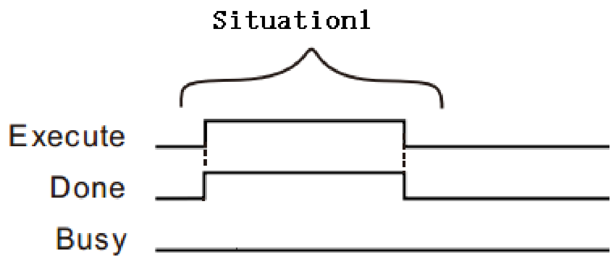

➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| CamTable (Electronic cam No.) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done (done bit) | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG output timing parameters**




Case 1: When the Execute FALSE to TRUE, while Done becomes TRUE, and if the time Execute to FALSE, FALSE becomes Done

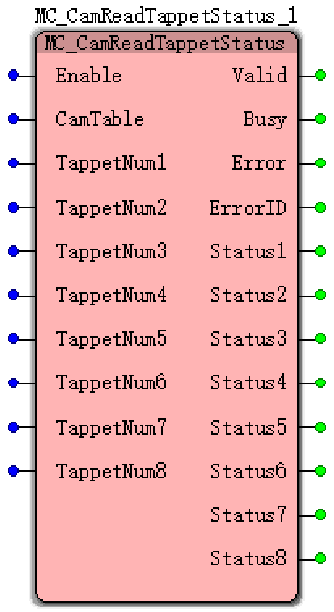
■ **Function Description:**

This instruction is used to change the point of entry into force of the cam. First instruction using MC_CamWritePoint electronic cam cam point information table corresponding write, and execute instructions MC_CamSet, the change takes effect cam point information.

MC_CamSet instruction execution, the cam curve after the changes take effect immediately.

## 11.5.17 MC_ReadTappetStatus (read status command plurality of lifters points)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the state of a plurality of points tappets | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| CamTable (Electronic cam No.) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |
| TappetNum1 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| TappetNum2 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| TappetNum3 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| TappetNum4 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| TappetNum5 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| TappetNum6 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| TappetNum7 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| TappetNum8 (Tappet point number) | Tappet set point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |

➢ **Output parameters**

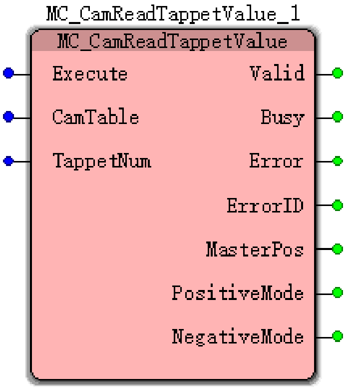
| name | Features | type of data | Output range |
|---|---|---|---|
| Valid | The parameter is TRUE outputs a command valid when executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error code when execution instruction | WORD | - |
| Status1 (1 point tappet state) | State TappetNum1 specified number of points tappet | BOOL | TRUE or FALSE |
| Status2 (2 tappet point state) | State TappetNum2 specified number of points tappet | BOOL | TRUE or FALSE |
| Status3 (3 tappet point state) | State TappetNum3 specified number of points tappet | BOOL | TRUE or FALSE |
| Status4 (4 tappets point state) | State TappetNum4 specified number of points tappet | BOOL | TRUE or FALSE |
| Status5 (Ram state point 5) | State TappetNum5 specified number of points tappet | BOOL | TRUE or FALSE |
| Status6 (6 points tappet state) | State TappetNum6 specified number of points tappet | BOOL | TRUE or FALSE |
| Status7 (Tappet state point 7) | State TappetNum7 specified number of points tappet | BOOL | TRUE or FALSE |
| Status8 (8 tappet point state) | State TappetNum8 specified number of points tappet | BOOL | TRUE or FALSE |

■ **Function Description**

This instruction is used to read the state of eight points of the tappet. Each tappet point state spindle through the forward or reverse state point of the tappet, the tappet of each state point is determined by the setting of each lifter point. The status of each point in the tappet end of the cam shaft through the forward or backward through the cam start point to FALSE.

## 11.5.18 MC_ReadTappetValue (single read command tappet point information)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is a single instruction for reading information tappet point | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| CamTable (Electronic cam No.) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |
| TappetNum (Tappet point number) | Read tappet point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Valid | The parameter is TRUE outputs a command valid when executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| MasterPos (Spindle position) | Display spindle position | LREAL | |
| PositiveMode (Forward through mode) | When the tappet axis positive rotation through point selection mode | INT | 0: PositiveDisable<br>1: PositiveOn<br>2: PositiveOff<br>3: PositiveInvert |
| NegativeMode (After reverse mode) | After the tappet axis inversion point when the selected mode | INT | 0: NegativeDisable<br>1: NegativeOn<br>2: NegativeOff<br>3: NegativeInvert |

■ **Function Description**

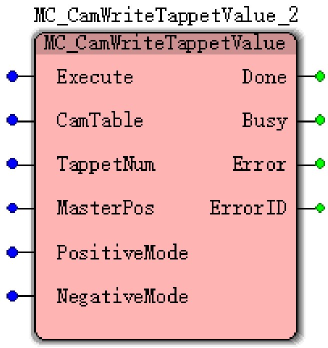
Tappet point information includes the location of the main point of the tappet, the forward and reverse passes through the pattern mode. When the shaft forward tappet point may select modes are PositiveDisable, PositiveOn, PositiveOff or PositiveIvert; mode when the tappet axis inversion point may select are NegativeDisable, NegativeOn, NegativeOff or NegativeIvert. The meaning of each pattern represents the following table:

| mode | Features | meaning |
|---|---|---|
| PositiveDisable | shut down | Spindle forward passes that point, to read the state of the tappet point unchanged |
| PositiveOn | Position | Spindle forward passes this point, the read state when the tappet point set state. |
| PositiveOff | Reset | Spindle forward passes this point, the read state when the tappet point reset state. |
| PositiveInvert | Negate | Spindle forward passes this point, forward after the state before the set point, then the read state when the reset state of the tappet point; forward state before the point after a reset, the read state when the tappet point set state. |
| NegativeDisable | shut down | Spindle reverse passes this point, the state of the read point tappet unchanged |
| NegativeOn | Position | The spindle back up through this point, the read state when the tappet point set state. |
| NegativeOff | Reset | The spindle back up through this point, the read state when the tappet point reset state. |

| | | |
|---|---|---|
| NegativeInvert | Negate | The spindle back up through the point, after the reverse state before the set point, then the read state when the tappet point reset state; the state before the reverse point after a reset, the read state when the tappet point set state. |

## 11.5.19 MC_WriteTappetValue (edit point information tappet instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to write, add, delete information tappet point instruction | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE (FALSE) | |
| CamTable (Electronic cam No.) | Establishing a main cam for setting table based on the relationship of the cam shaft from | USINT | 0 to 31 | Exexcute from FALSE to TRUE |
| TappetNum (Tappet point number) | Read tappet point number | UINT | (Non-default) | Exexcute from FALSE to TRUE |
| MasterPos (Spindle position) | Tappet spindle position set point | LREAL | | Exexcute from FALSE to TRUE |
| PositiveMode (Forward through mode) | Forward tappet axis point mode setting<br>0:Close<br>1: Set<br>2: Reset<br>3: Inversion | INT | 0-3<br>(0) | Exexcute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| NegativeMode (After reverse mode) | Mode setting tappet axis inversion point<br>0:Close<br>1: Set<br>2: Reset<br>3: Inversion | INT | 0-3<br>(0) | Exexcute from FALSE to TRUE |

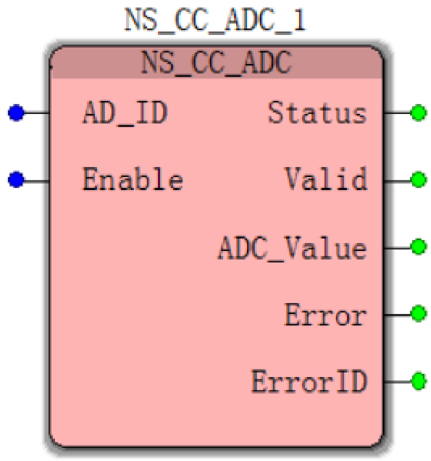
➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done (done bit) | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

## 11.6 special instructions

### 11.6.1 NS_CC_ADC (AD instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to convert analog to digital and outputs | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| AD_ID (Analog input number) | Select the required analog rpm The amount of opening AI0～AI3, wherein Numerical Simulation of 0 to 3 corresponding to the number of Input port AI0～AI3 | WORD | 0-3 (Non-default) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction is executed | BOOL | TRUE or FALSE | |

**Description:**

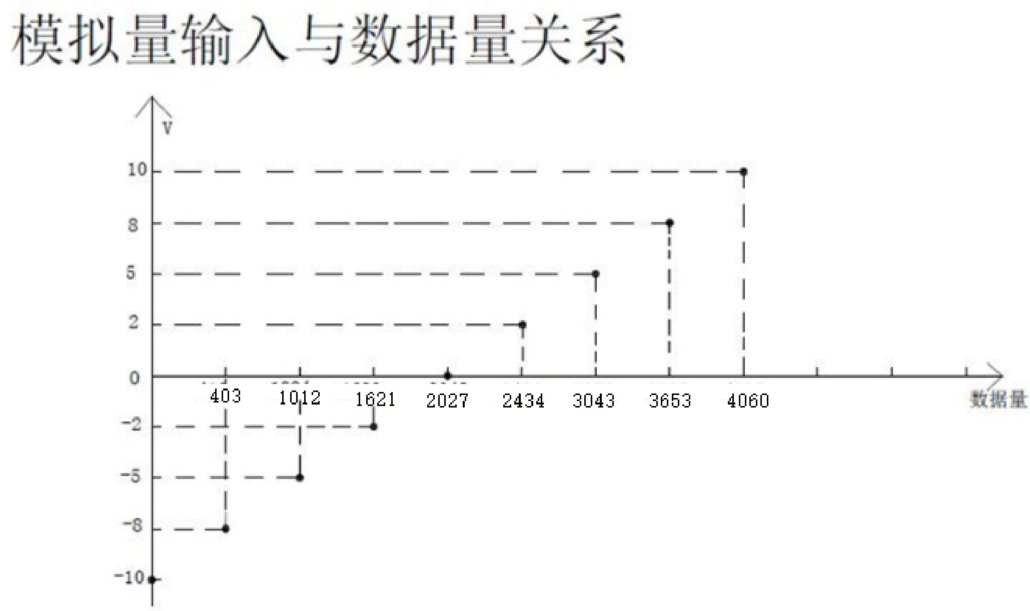
(1)The controller has four analog inputs (±10V) Shown relation between the analog and digital conversion of the following amounts:

| name | Numerical |
|---|---|
| Analog input voltage (V) | 0 ~ -10 ~ + 10 |
| Digital | 0 ~ 2048 ~ 4096 |

(2)The experimental results (due to the presence of an error, there will be fluctuations in the corresponding digital value for reference)

| Input voltage(V) | Digital |
|---|---|
| 0 | 2028 |
| 2 | 2434 |
| 5 | 3043 |
| 8 | 3653 |
| 10 | 4060 |
| -10 | 0 |
| -8 | 403 |
| -5 | 1012 |
| -2 | 1621 |
| 0 | 2027 |

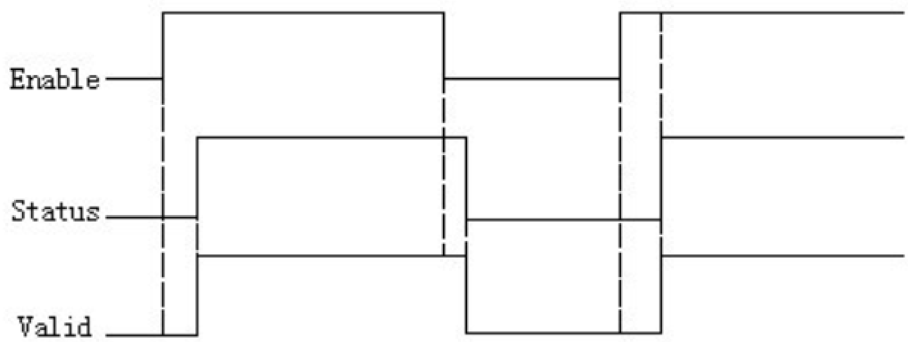
(3)Test table obtained by the analog input and the amount of data relationship diagram



➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Status<br>(Status bit) | The output parameter is TRUE Representing instructions being executed in | BOOL | TRUE or FALSE |
| Valid<br>(Significant Bit Enable) | When the output parameter is TRUE. Indicates that the instruction is controlling the axis | BOOL | TRUE or FALSE |
| ADC_Value<br>(Analog current value of the amount of data transferred) | The current output is converted to analog. As the amount of data values | DINT | 0 ~ 4096 |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID<br>(Error code ID) | Error Error code when execution instruction | WORD | - |

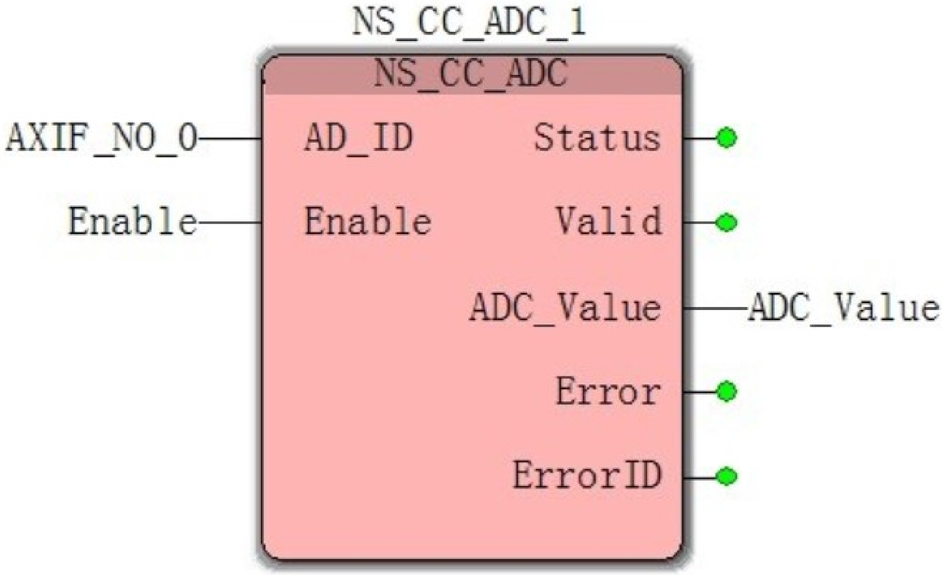
- **FIG timing variation output parameter**




**Program Example**

AI0 input 5V, instructions into the data amount value ADC_Value

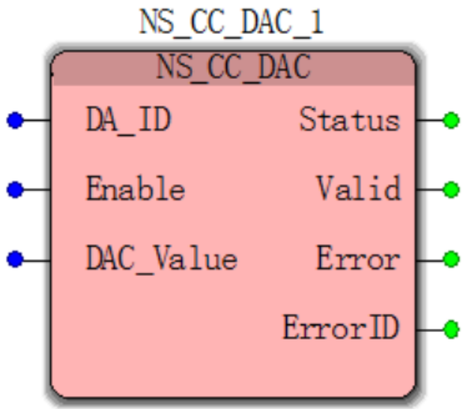
| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_ADC_1 | AXIF_NO_0 | |
| AXIF_NO_0 | WORD | 0 |
| Enable | BOOL | |




Case 1: When the Enable FALSE to TRUE, the external input AI0 DC5V, the controller converts the 5V voltage into a digital output displays ADC_Vable 3070 (there may be errors to the actual subject)

## 11.6.2 NS_CC_DAC (DA instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | The instructions for converting the data amount into an analog output<br>(Output voltage range of ± 10V) | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| DA_ID (Analog input number) | Select the analog port as an output, The corresponding number of 0 to 3 AXIS0～AXIS3 shaft | WORD | 0-3 (Non-default) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction is executed | BOOL | TRUE or FALSE | |
| DAC_Value (The quantity of data is provided) | Setting data value | DINT | -2048 to 2047 | Enable is TRUE |

**Description:**

(1) Digital-to-analog quantity corresponding to the following relationship:

Digital input : -2048 ~0 0~ 2047

Analog output :-10V ~ 0V 0V ~ 10V

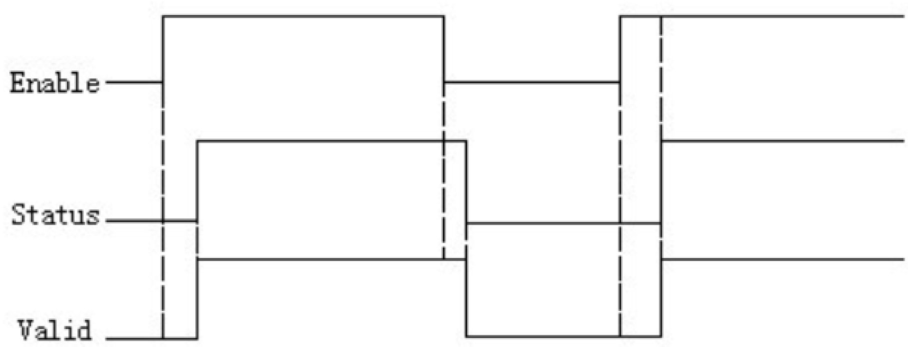
(2) when this instruction is executed, the variable value is 0 DAC_Value, standard analog output 0V.

(3) This is an open loop control instruction, the execution module, the input voltage according to the corresponding value DAC_Value.

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Status (Status bit) | The output parameter is TRUE Representing instructions being executed in | BOOL | TRUE or FALSE |
| Valid (Significant Bit Enable) | The output parameter is TRUE Is a command indicating the control shaft | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (Error code ID) | Error Error code when execution instruction | WORD | - |

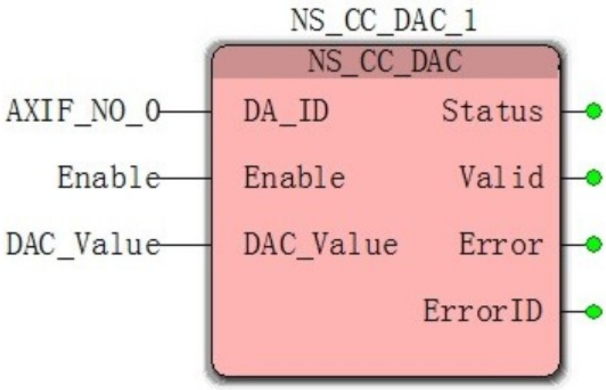
- **FIG timing variation output parameter**




**Program Example**

Meter testing the first channel DA output 5V

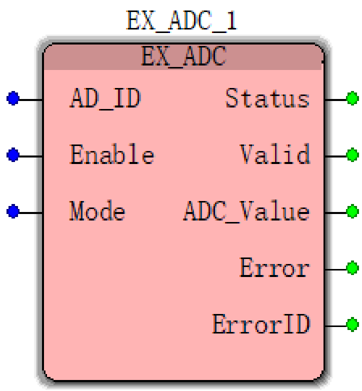
| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_DAC_1 | NS_CC_DAC | |
| AXIF_NO_0 | WORD | 0 |
| Enable | BOOL | |
| DAC_Value | DINT | 0 |




situation**1**: When Enable changes from FALSE to TRUE, Valid changes from FALSE to TRUE, and the next cycle, the analog quantity output 0V, the modified variable DAC_Value is 1024, the analog output 5V.

## 11.6.3 EX_ADC (AD extended instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to convert the analog input expansion module into AD and outputs the data amount | VEC-VA-MP-005-MA |




➢ **Input parameters**

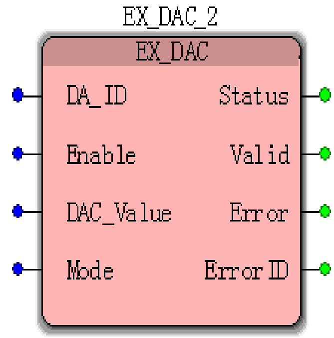
| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| AD_ID (Analog input number) | Select the required analog rpm.The amount of opening AI4 ~ AI36, value of 4-7 corresponds to the first extension module of AD V0 ~ V4 | WORD | 4 to 36 (Non-default) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction is executed | BOOL | TRUE or FALSE | |
| Mode (Mode) | Under different modes EX_ADC<br>0, corresponding to the input voltage of 0 to 5 0 to 4095<br>1, 0 ~ 10V input voltage corresponding to 0 to 4095<br>2, the input voltage corresponding to 0 to 4095 ± 10V<br>3, 4 ~ 20mA input current corresponding to 819 to 4095 | WORD | 0-3 (Non-default) | |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Status (Status bit) | The output parameter is TRUE. Representing instructions being executed in | BOOL | TRUE or FALSE |
| Valid (Significant Bit Enable) | The output parameter is TRUE. Is a command indicating the control shaft | BOOL | TRUE or FALSE |
| ADC_Value (Analog current value of the amount of data transferred) | The current output is converted to analog. As the amount of data values | DINT | 0 ~ 4096 |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (Error code ID) | Error Error code when execution instruction | WORD | - |

## 11.6.4 EX_DAC (DA expansion module)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | The instructions for converting the data amount DA expansion module into analog outputs | VEC-VA-MP-005-MA |



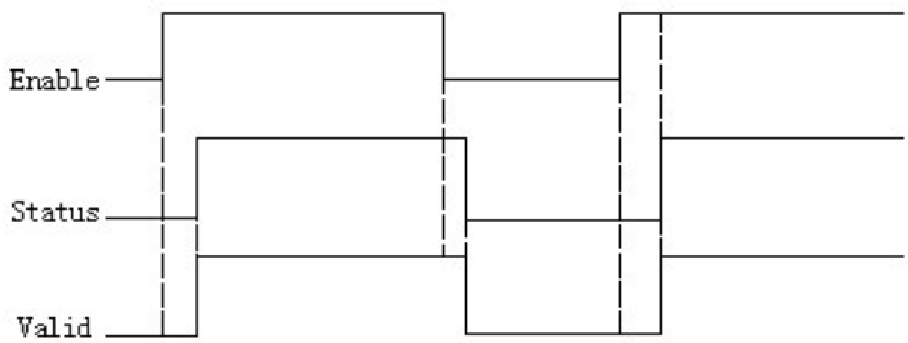

➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| AD_ID (Analog input number) | Select the required analog rpm. The amount of opening AI4 ~ AI36.Value of 4-7 corresponds to the first extension module of AD V0 ~ V4 | WORD | 4 to 36 (Non-default) | Enable is TRUE |
| Enable (Execute bit) | When Enable is TRUE, the instruction is executed | BOOL | TRUE or FALSE | |
| DAC_Value (The quantity of data is provided) | Setting data value | DINT | 0 to 4095 | Enable is TRUE |
| Mode (Mode) | Under different modes EX_DAC<br>0,0 to 4095 corresponding to the output voltage --10V ~ + 10V<br>1,0 to 4095 corresponding to the output voltage -0V ~ + 5V<br>2,0 to 4095 corresponding to the output voltage -0V ~ + 10V | WORD | 0-2 (Non-default) | |

➢ **Output parameters**

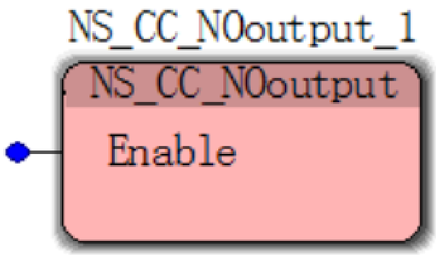
| name | Features | type of data | Output range |
|---|---|---|---|
| Status<br>(Status bit) | The output parameter is TRUE Representing instructions being executed in | BOOL | TRUE or FALSE |
| Valid<br>(Significant Bit Enable) | The output parameter is TRUE Is a command indicating the control shaft | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID<br>(Error code ID) | Error Error code when execution instruction | WORD | - |

- **Timing variation output parameter**



## 11.6.5 NS_CC_NOoutput (prohibition command output QXX)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command prohibits all QX output (X: represents 0 to 128) | VEC-VA-MP-005-MA |




- **Input parameters**

| name | Features | type of data | Setting range (Non-default) | The timing of the entry into force |
|---|---|---|---|---|
| Enable (Prohibition significant bit) | TRUE prohibit all QX output, FALSE is allowed QX output. | BOOL | TRUE or FALSE | |

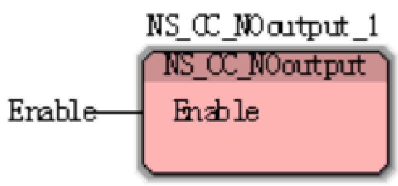
**Function Description:**

(1) This command disables all QXX output user selected according to site requirements.

(2) When Enable is TRUE when all of the DO, the output will be disabled when Enable is FALSE, the DO restored to the state before the execution of the module (all the QX is prohibited, can not specify a single QX.X);

**Program Example**

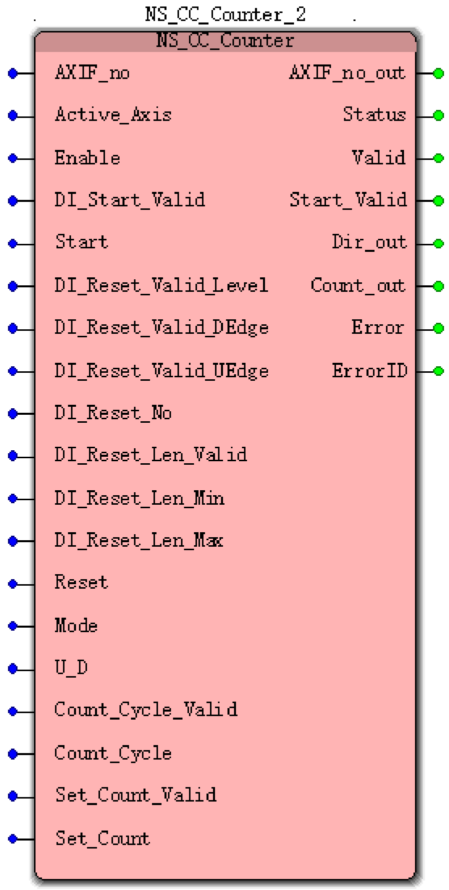
| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_NOoutput | NS_CC_NOoutput | |
| Enable | BOOL | |




situation1:whenEnablebyFALSEChanges toTRUEWhen allDigitalOutput will be banned output, whenEnable byTRUEChanges toFALSETime,QXXTo restore the state before.

## 11.6.6 NS_CC_Counter (High-Speed Counter)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | For counting the number of pulses, the counting method is not affected by a hardware counter scanning period, the maximum count frequency is 1MHz | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (Count axis number) | Set number counter shaft | WORD | 0-6 (0) | Enable TRUE when the FALSE to |
| Active_Axis (Current count axis) | Pulse count setting Source:<br>0: AXIS0;<br>1: AXIS1;<br>2: AXIS2;<br>3: AXIS3<br>4: AXIS4 | WORD | 0-4 | Enable TRUE when the FALSE to |

| | | | | |
|---|---|---|---|---|
| Enable (Execute bit) | When Enable by FALSE Changes toTRUE When the instruction is executed. | BOOL | TRUE or FALSE | - |
| DI_Start_Valid (External DI start significant bit) | When the DI_Start_Valid When becomes FALSE TRUE, it allows you to specify the external DI active, high-speed counter starts (DI has been specified, be described in detail down) | BOOL | TRUE or FALSE | - |
| Start (Start bit) | When Start is TRUE, start high-speed counter | BOOL | TRUE or FALSE | - |
| DI_Rest_Valid _Level (External active-high bit is cleared DI) | When DI_Reset_Valid_ Level is TRUE, DI is cleared to clear high | BOOL | TRUE or FALSE | - |
| DI_Rest_Valid _DEdge (External DI cleared falling Significant bit) | When DI_Reset_Valid_ Dedge is TRUE, DI cleared the falling edge clear | BOOL | TRUE or FALSE | - |
| DI_Rest_Valid _UEdge (Rising external DI cleared Significant bit) | When DI_Reset_Valid_ Uedge is TRUE, DI cleared to rising cleared | BOOL | TRUE or FALSE | - |
| DI_Reset_No (External input DI cleared number) | Specifies the external clear signal terminal, the input value of 0 to 7 corresponding to the input point I0 ~ I7,8 ~ 15 corresponding to the input point I10 ~ I17 | WORD | TRUE or FALSE | - |
| DI_Rest_Len_ Valid (External DI clearing interval significant bit) | When DI_Rest_Len_Valid When TRUE, DI clear signal only effective in clearing the zone | BOOL | TRUE or FALSE | - |
| DI_Reset_Len | Setting a valid range | WORD | | - |

| | | | | |
|---|---|---|---|---|
| _Min (External DI cleared range floor) | limit signal DI is cleared | | | |
| DI_Reset_Len_Max (External DI cleared range ceiling) | The upper limit is set valid interval signal DI is cleared | WORD | | - |
| Reset (Clear bit) | When Reset is TRUE, clears the counter current value | BOOL | TRUE or FALSE | - |
| Mode (Mode selection) | Retention | DINT | Retention | - |
| U_D (Counting direction) | Retention | DINT | Retention | - |
| Count_Cycle_Valid (Cycle Count Valid bits) | When is Count_Cycle_ Valid is TRUE, start the counting cycles when the counter value reaches the counter Count_Cycle set value, the new count is automatically cleared, constant cycle. Count_Cycle_Valid to FALSE when not enable this feature. | BOOL | TRUE or FALSE | - |
| Count_Cycle (Set pulse loop counter value) | When is Count_Cycle_ Valid is TRUE, start the counting cycles when the counter value reaches the counter Count_Cycle set value, the new count is automatically cleared, constant cycle.Count_Cycle_Valid to FALSE when not enable this feature. | DINT | Positive, negative, | - |
| Set_Count_Valid (Pulse current count value set valid bit) | When the Set_Count_ Valid is TRUE, the High-speed counter current value is set to Set_Count value | BOOL | TRUE or FALSE | - |

| | | | | |
|---|---|---|---|---|
| Set_Count (Set the current count of pulses) | High-speed counter current value setting | DINT | Positive, negative, 0 | - |

**Description:**

(1) 5-way motion controller integrated with hardware high speed counter respectively port AXIS0 ~ AXIS4 (currently only supports the AB phase signal), the counting is not affected by a pure hardware scan cycle, the maximum count frequency is 1MHz. 5-way counter port 7 can be called high-speed counter module AXIF_no (counter number) from 0 to 6, respectively, are opposite to each other counting methods, without disturbing each other.

(2) This command integrates feature-rich, containing DI external start, the counter is cleared, and clearing the counter variable start, cycle counting function, set the number of functions,

(3) sources by pulse counter Active_Axis (axis current count) is selected, any given, but also cross-axis counting function.

(4) high-speed counter-rich functional integration, can be selected according to requirements, function of the input parameters to an unused blank.

(5) Start the external terminals DI internal high-speed counter has been specified, when a high number counter, when DI_Start_Valid is TRUE, the terminal being used as the external DI start high-speed counter input terminals, when DI_Start_Valid is FALSE, but when an ordinary I / O using the following specific distribution

| AXIF_no (axis number) | Specifies the external terminals DI as the start counter signal high | DI_Reset_No |
|---|---|---|
| 0 | DI0.3 | Optional (not allowed to repeat with DI start) |
| 1 | DI0.5 | Optional (not allowed to repeat with DI start) |
| 2 | DI0.7 | Optional (not allowed to repeat with DI start) |
| 3 | DI1.1 | Optional (not allowed to repeat with DI start) |
| 4 | DI1.3 | Optional (not allowed to repeat with DI start) |
| 5 | DI1.5 | Optional (not allowed to repeat with DI start) |
| 6 | DI1.7 | Optional (not allowed to repeat with DI start) |

- **Output parameters**

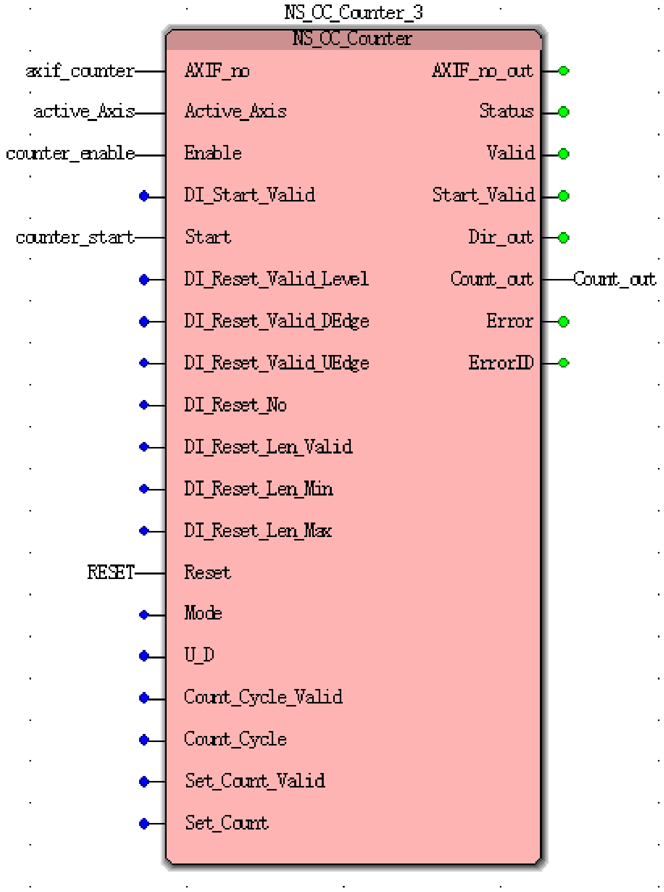
| name | Features | type of data | Output range |
|---|---|---|---|
| AXIF_no_out (Output axis number) | No current count output shaft | WORD | 0-6 |
| Status (Status bit) | The output parameter is TRUE. Representing instructions being executed in | BOOL | TRUE or FALSE |
| Valid (Significant Bit Enable) | The output parameter is TRUE. Is a command indicating the control shaft | BOOL | TRUE or FALSE |
| Start_Valid | Retention | BOOL | - |
| Dir_out | Retention | BOOL | - |
| Count_out | Current count of pulses | DINT | Positive, negative, 0 |
| Error | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID | Error Error code when execution instruction | WORD | - |

**Demonstration program** Ⅰ

Example: Variable start pulse and clears the count shaft axif_counter

**1. Variables, and procedures**

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_Counter_3 | NS_CC_Counter | |
| axif_counter | WORD | 0 |
| active_Axis | WORD | 0 |
| counter_enable | BOOL | 1 |
| counter_start | BOOL | 1 |
| RESET | BOOL | |



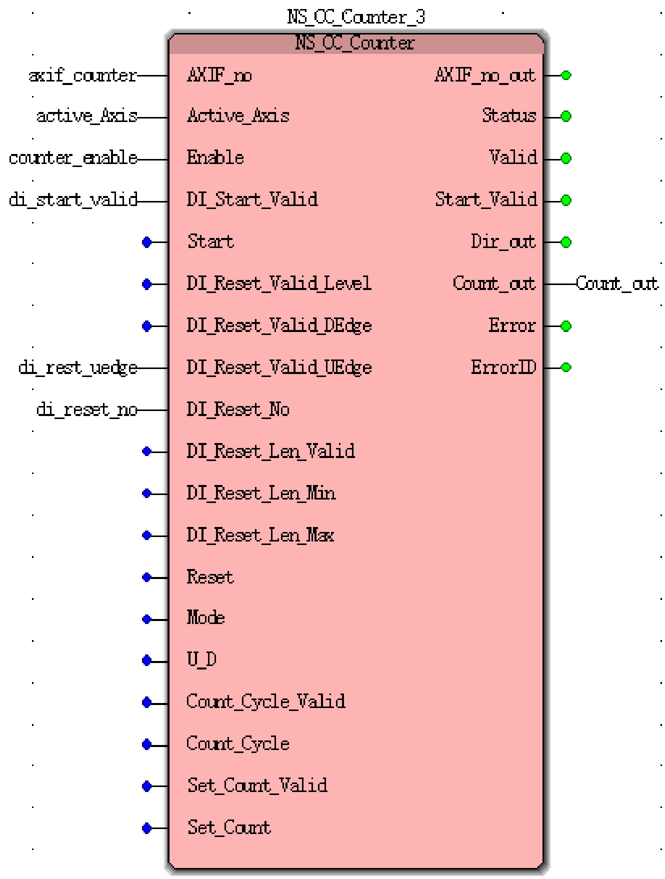

**Case 1:**When counter_enable from FALSE to TRUE, and held, counter_start from FALSE to TRUE starts allows high-speed counter, Count_out display the current count of pulses. When the RESET FALSE to TRUE, Count_out display the current pulse count is 0, then the number of the input pulse, Count_out display the current pulse count remains at 0 when the RESET TRUE to FALSE, allowed to continue to start counting.

**Demonstration Program II**

Example: external terminal DI3 counter is cleared and the start pulse number axif_counter outer shaft DI0

1. Variables, and procedures

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_Counter_3 | NS_CC_Counter | |
| axif_counter | WORD | 0 |
| active_Axis | WORD | 0 |
| counter_enable | BOOL | 1 |
| di_start_Valid | BOOL | 1 |
| di_reset_uedge | BOOL | 1 |
| di_reset_no | BOOL | 0 |




**Case 1:**When counter_enable FALSE to TRUE grounds, and maintained. di_start_Valid becomes TRUE, and if the external signal DI3 signal is valid, the counter 3 starts counting the number of high, Count_out display the current pulse when the rising edge DI0 di_reset_no specified, the current value of the high-speed counter is cleared.

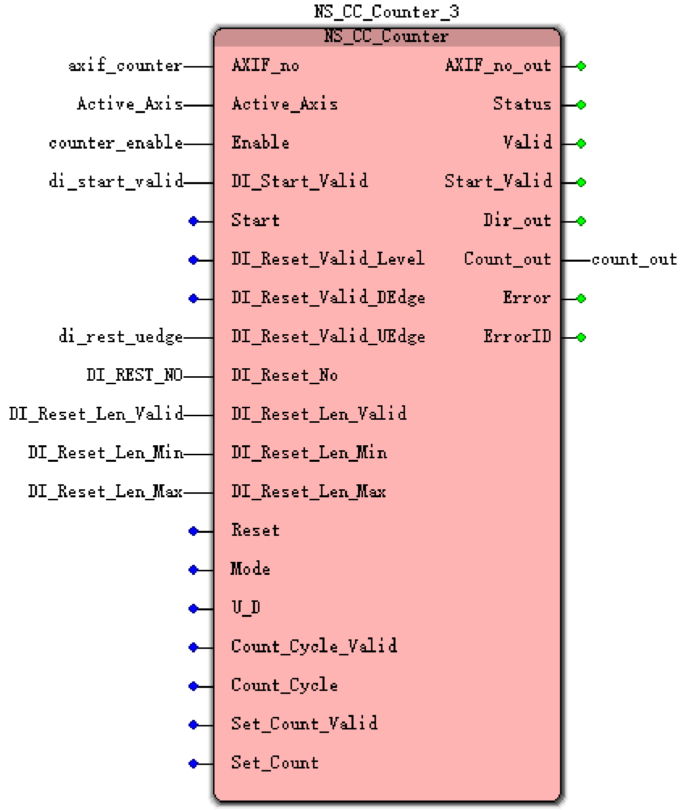
**Demonstration program Ⅲ**

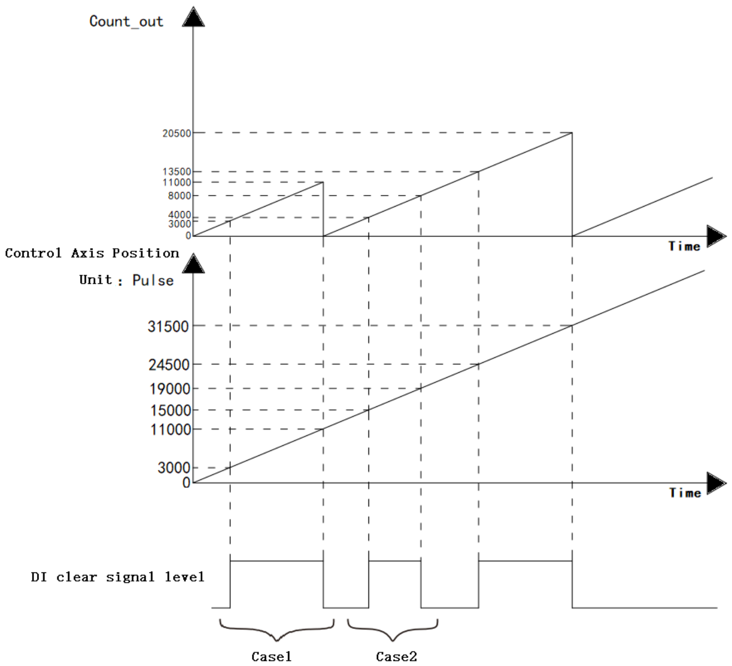
Example: outer boot and the external terminals DI0 DI3 is cleared, and sets the number of pulses axif_counter external shaft section DI valid counter is cleared

1. variables, and procedures

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_Counter_3 | NS_CC_Counter | |
| axif_counter | WORD | 0 |
| active_Axis | WORD | 0 |
| counter_enable | BOOL | 1 |
| di_start_Valid | BOOL | 1 |
| di_reset_uedge | BOOL | 1 |
| di_reset_no | BOOL | 0 |
| DI_Rest_Len_Valid | BOOL | 1 |

| | | |
|---|---|---|
| DI_Rest_Len_Min | WORD | 5000 |
| DI_Rest_Len_Max | WORD | 10000 |




**Case 1:**When counter_enable FALSE to TRUE grounds, and maintained. di_start_Valid becomes TRUE when the external signal DI3 signal is valid, the counter 3 starts counting the number of high, Count_out display the current pulse, when the designated DI0 di_reset_no from FALSE to TRUE again until DI0 becomes FALSE, the controlled axis through this period

displacement (pulse) 8000 (11000-3000), DI_Rest_Len_Min ~ DI_Rest_Len_Max within the set range, so again when DI0 to FALSE, the operation of Count_out cleared.

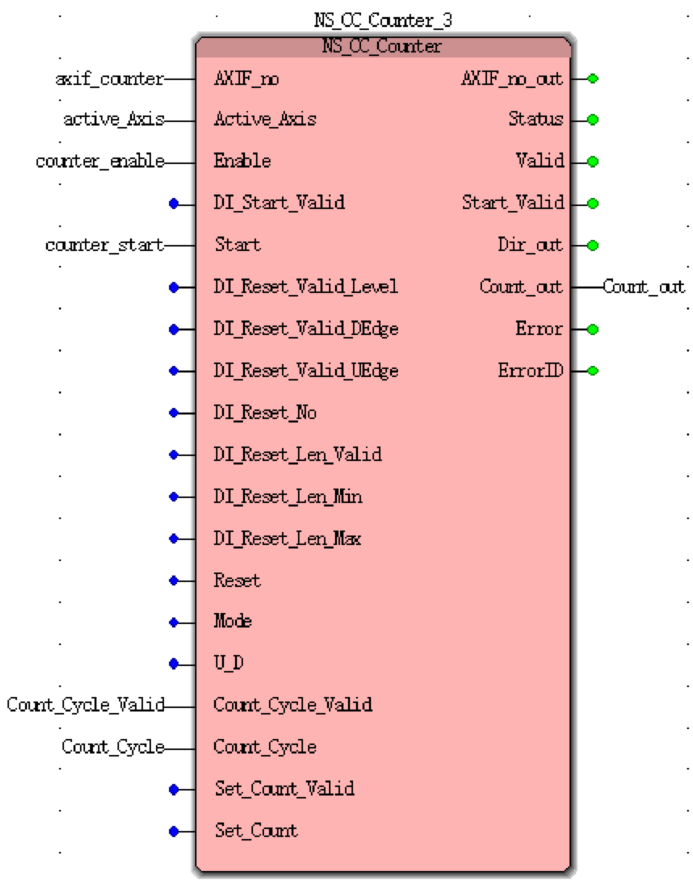
**Case 2:**When counter_enable FALSE to TRUE grounds, and maintained. di_start_Valid becomes TRUE when the external signal DI3 signal is valid, the counter 3 starts counting the number of high, Count_out display the current pulse, when the designated DI0 di_reset_no from FALSE to TRUE again until DI0 becomes FALSE, the controlled axis through this period displacement (pulse) 4000 (19000-15000), is not within DI_Rest_Len_Min ~ DI_Rest_Len_Max set interval, and therefore becomes FALSE when DI0 again, will not be cleared Count_out operation.

### Demonstration program Ⅳ

Example: loop count mode pulse number counter shaft axif_counter

1, variables, and procedures

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_Counter_3 | NS_CC_Counter | |
| axif_counter | WORD | 0 |
| active_Axis | WORD | 0 |
| counter_enable | BOOL | 1 |
| counter_start | BOOL | 1 |
| Count_Cycle_Valid | BOOL | 1 |
| Count_Cycle | DINT | 5000 |



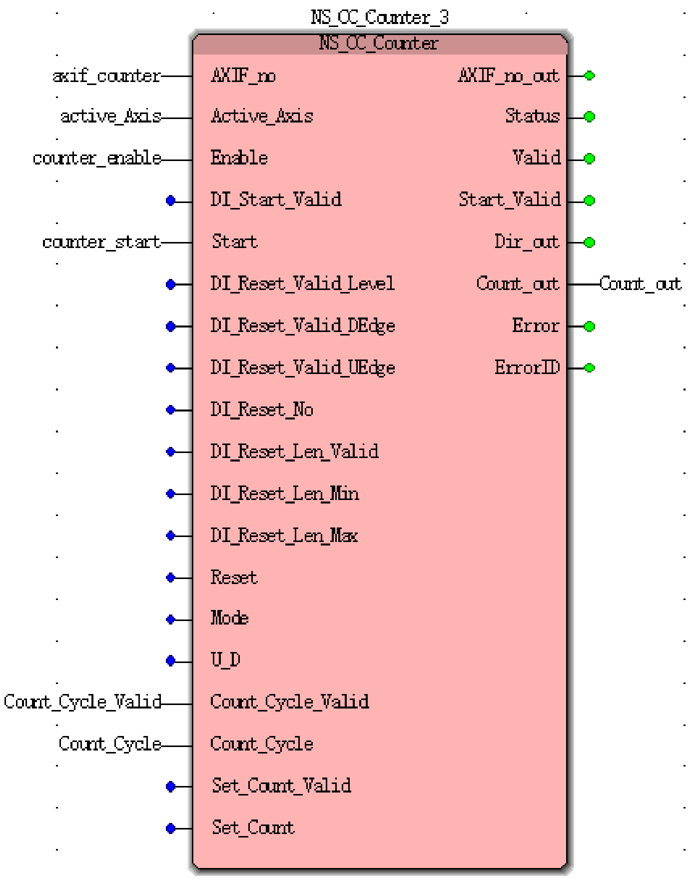
**Case 1:**When counter_enable from FALSE to TRUE, and held, counter_start from FALSE to TRUE starts allows high-speed counter, Count_out display the current count of pulses. When the high-speed counter current value of 5000, the counter is automatically cleared recount, constant cycle.

**Demonstration program** V

Modes Number of pulses counter shaft axif_counter

1, variables, and procedures

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_Counter_3 | NS_CC_Counter | |
| Axif_counter | WORD | 0 |
| active_Axis | WORD | 0 |
| counter_enable | BOOL | 1 |
| Star | BOOL | 1 |
| Set_Count_Valid | BOOL | |
| Set_Count | DINT | 10000 |

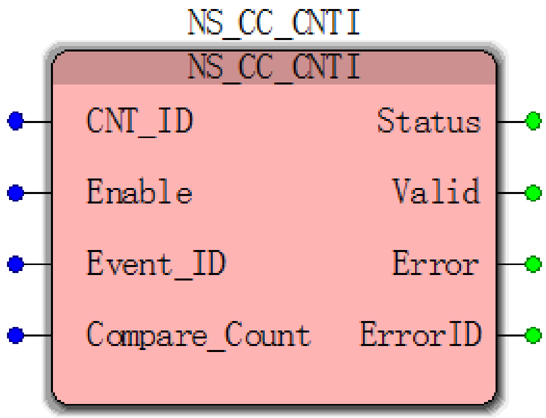



**Case 1:**When counter_enable a FALSE to TRUE, counter_start from FALSE to TRUE starts allows high-speed counter, Count_out display the current count of pulses. When

Set_Count_Valid variable from FALSE becomes TRUE, high-speed counter Count_out current value is modified 10,000, and kept 10,000, when Set_Count_Valid variable by TRUE becomes FALSE, when a pulse comes in, counts the number of pulses will continue to increment the basis of 10,000 or on decremented.

## 11.6.7 NS_CC_CNTI (high-speed counter interrupt instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to generate an interrupt when the pulse number reaches the set current value of the high-speed counter, interrupt routine is entered | VEC-VA-MP-005-MA |




- **Input parameters**

| name | Features | type of data | Setting range (Non-default) | The timing of the entry into force |
|---|---|---|---|---|
| CNT_ID (Associated counter number) | High count associated with the specified number, which is automatically associated with the counter number is equal to | WORD | 0 to 1 (Indispensable) | Execute from FALSE to TRUE |
| Enable (execute bit) | When the Enable FALSE to TRUE, the instruction is executed. | BOOL | TRUE or FALSE | Execute from FALSE to TRUE |
| Event_ID (interruption) | Set interrupt event number (0-5) is consistent with the interrupt routine event | WORD | 0-5 | Execute from FALSE to TRUE |

| | | | | |
|---|---|---|---|---|
| | number. | | | |
| Compare_Count (Set pulse number) | Set interrupt pulse number | DINT | A positive number (Non-default) | Execute from FALSE to TRUE |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| The Status (Status bits) | This parameter indicates when the output instruction is being executed is TRUE | BOOL | TRUE or FALSE |
| Valid (valid bit) | Represents the output parameter is TRUE instruction is controlling the shaft | BOOL | TRUE or FALSE |
| Error (error) | Retention | BOOL | TRUE or FALSE |
| ErrorID (error code) | Retention | WORD | Retention |

**Description:**

(1) NS_CC_CNTI command (count interrupt instruction) needs to be associated NS_CC_Counter instruction (high-speed counter) used together, whether the person can not be achieved.

(2) NS_CC_CNTI command (interrupt instruction count) is associated NS_CC_Counter instruction (high-speed counter), it is necessary CNT_ID (counter associated axis number) to fill the same with the high-speed counter AXIF_no (counts axis number) to

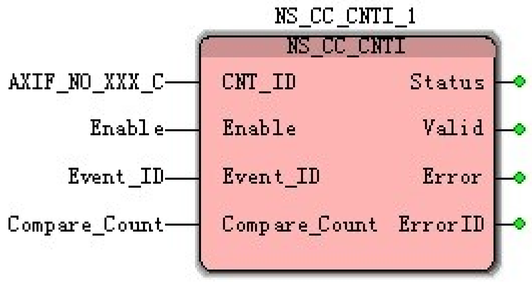
(3) Counter interrupt instruction currently supports two-way, high-speed counter current numbers 0 and 1 support this feature, and the remaining 2-6 count yet support interrupt function.

**Program example (0 high counter interrupt setting)**

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CNTI_1 | NS_CC_CNTI | |
| AXIF_NO_XXX_C | WORD | 0 |
| Enable | BOOL | 1 |
| Event_ID | WORD | 0 |
| Compare_Count | DINT | 5000 |

**Step 1:** High-speed counter call NS_CC_Counter_1 (refer to Note 1.4.4 counter shaft and the present instruction signal wave consistent, will not be repeated here, a high number of write counter variable)

**Step 2:** Call NS_CC_CNTI_1 module, as follows



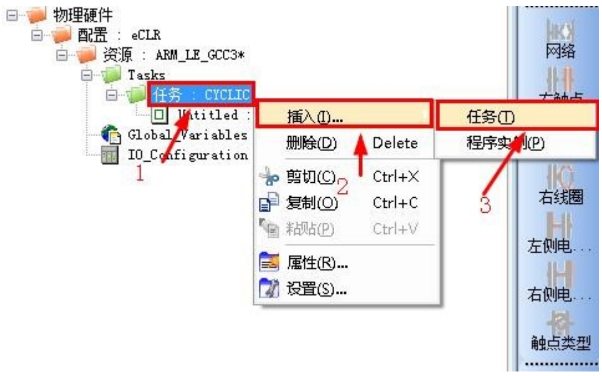

**Step 3:** The establishment of an interrupt routine in the "Project Tree Window" right "logic POU" select "Insert" and then select "Programs" in the window that pops up to interrupt program named "EN1" (no user name), and choose to write an interrupt routine language "LD language" below 10-10.


**Srep 4:** When finished, click OK, will automatically add the "logic the POU" to a "EN1" is the name of the program shown in Figure 10-11


**Step 5:** Just inserted "EN1" belongs to the category of the main program or program, as the new "the POU" software automatically selected when loaded into the new project "Task" CYCLIC type; 10-13 shown in FIG.


**Step 6:**Select under "Project Tree Window", "Hardware", under Resources "task", right-delete "EN1" program and click "OK" we need another new "task type", as shownure 10-14

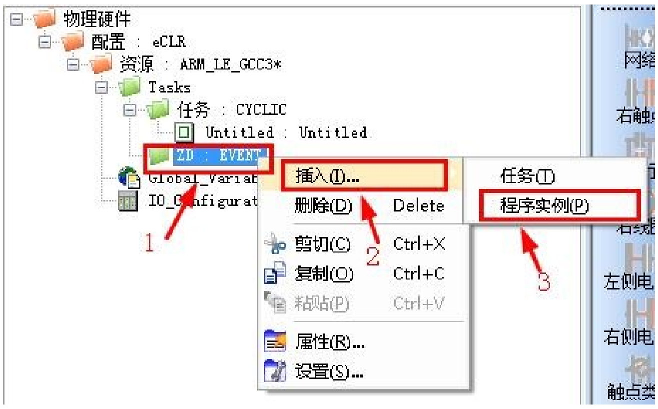

**Step 7:** New "Task Type" right "Tasks" click "insert", and then select the "Tasks" shown in Figure 10-15




**Step 8:** In the pop-up to "Insert" dialog box, fill in "ZD" in the name (no user name) Task Type Select "EVENT" are finished, click "OK." FIGS 10-16 shown in FIG.


**Step 9:**In the pop up "ZD task settings" Pick selection event number "0 Event" "Event_ID" is determined by the value on the interrupt module fill, Pick 0 (exemplary program "Event_ID = 0") 10-17 in FIG. as shown in (special attention not interrupt priority and the priority of the main program as whether those compiled by not downloading or 97% reported error)


**Step 10:**After the insertion procedure of example, the right task, "the EVENT", select "insert" and then select the "program example" 10-18 shown in FIG.

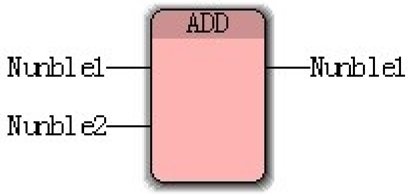



**Step 11:**In the pop-up to "insert box", for the example program named "ZD1 is", select the new program type "EN1"

Click "OK" to the final result shown in Figure 10-19


**Step 12:** "Project" Return "Project Tree Window" in the window, double-EN1 at POU join a summing module (purpose is to verify whether the interrupt routine is executed correctly)

| variable name | type of data | The initial value |
|---|---|---|
| ADD | ADD | |
| Nunble1 | INT | |
| Nunble2 | INT | 1 |

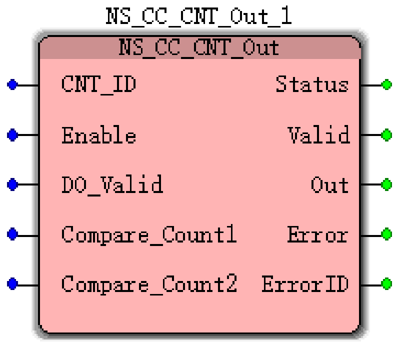



Interrupt this program to create and association has been completed, the following is the analysis process interrupt routine is executed.

**Case 1:**Start the high counter, allowing it to count, when the module is NS_CC_CNTI_1 Enable changes from FALSE to TRUE, the interrupt allowed to open, when the current value of the counter reaches the number of high set value 5000 interrupt, the interrupt program is executed once, this time variable Nunble1 a value of 1, after the completion of jump interrupt routine returns to the main. High again becomes the current value of counter 5000, the interrupt routine is executed again, this time is variable Nunble1 2.

## 11.6.8 NS_CC_CNT_Out (comparison output instruction section)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | The instructions for the high-speed counter current value reaches the set interval immediately output, DO outputted from the scan cycle Effect | VEC-VA-MP-005-MA |




- **Input parameters**

| name | Features | type of data | Setting range (Non-default) | The timing of the entry into force |
|---|---|---|---|---|
| CNT_ID (Counter associated axis number) | Associated with a respective number of high-speed counter shaft | WORD | 0-6 (Non-default) | - |
| Enable (Execute bit) | When the Enable FALSE to TRUE, the instruction is executed. | BOOL | TRUE or FALSE | - |
| DO_Valid (Allowing the Q output valid bit) | When the value is TRUE, the Q output to allow effective | BOOL | TRUE or FALSE | - |
| Compare_Count1 (Minimum value setting section min) | Min Min pulse interval is set (unit:) | DINT | A positive number | - |
| Compare_Count2 (Setpoint interval maximum value Max) | Setting a maximum value Max pulse interval (unit:) | DINT | A positive number | - |

**Description:**

(1) NS_CC_CNT_Out instruction (instruction comparison output section) needs to be associated NS_CC_Counter instruction (high-speed counter) used together, no meaning when used alone.

(2) NS_CC_CNT_Out instruction (output instruction section compare) instruction correlation NS_CC_Counter (high-speed counter), needs to be provided CNT_ID (number

associated with a counter shaft) filled with the high-speed counter AXIF_no (counts axis number) can be consistent.

(3) comparing the output section of the scan signal is not QXX cycle impact, using the internal FPAG satisfy the condition after the count output.

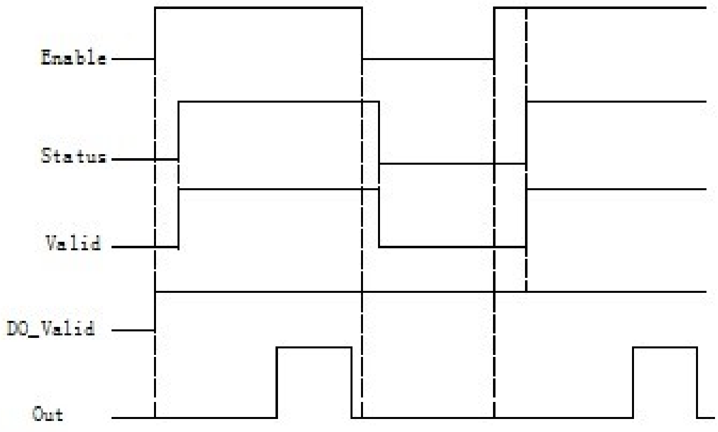
(4) DO_ID output number counter section, inside of which has specified, the specified relationship is as follows (designated DO_ID, when allowed to use a common DO)

| CNT_ID (counter associated axis number) | QXX (output) |
|---|---|
| 0 | Q0.0 |
| 1 | Q0.1 |
| 2 | Q0.2 |
| 3 | Q0.3 |
| 4 | Q0.4 |
| 5 | Q0.5 |
| 6 | Q0.6 |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| The Status (Status bits) | This parameter indicates when the output instruction is being executed is TRUE | BOOL | TRUE or FALSE |
| Valid (valid bit) | Represents the output parameter is TRUE instruction is controlling the shaft | BOOL | TRUE or FALSE |
| Out (Output signal) | This parameter indicates the output being TRUE Q output | BOOL | TRUE or FALSE |
| Error (error) | Retention | BOOL | Retention |
| ErrorID (error code) | Retention | WORD | Retention |

- **FIG timing variation output parameter**




**Case 1:**When the Enable FALSE to TRUE, Status Valid and becomes TRUE after a period and, FALSE when Enable changed from TRUE, Status Valid and from FALSE to TRUE and after a cycle.

2. When the case DO_Valid FALSE to TRUE, the conditions are satisfied by the Out FALSE to TRUE, while the output Q0.0, when the condition is not met by the automatically Out TRUE to FALSE while Q0.0 no output.

**Program Example**

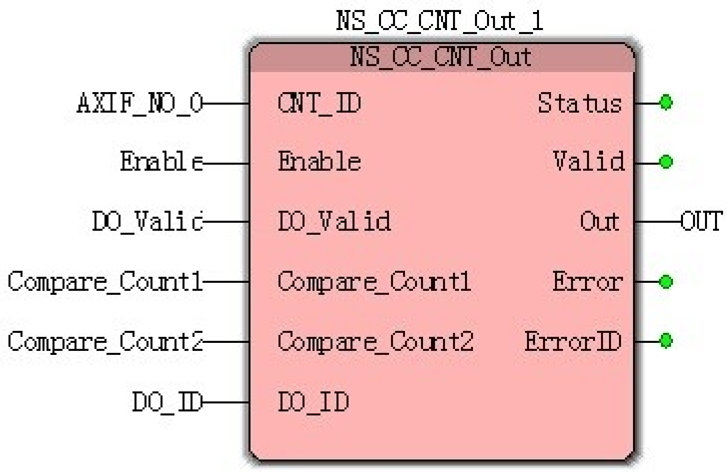
Example: high speed counter range between 1000 to 5000 output Q0.0;

**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| AXIF_NO_0 | WORD | 0 |
| Enable | BOOL | |
| DO_Valid | BOOL | 1 |
| Compare_Count1 | DONT | 1000 |
| Compare_Count2 | DONT | 5000 |
| DO_ID | WORD | 0 |
| OUT | BOOL | |

**Step 1:** High-speed counter call NS_CC_Counter_1 (refer to Note 11.7.7 counter shaft and the present instruction signal wave consistent, will not be repeated here write counter variable)

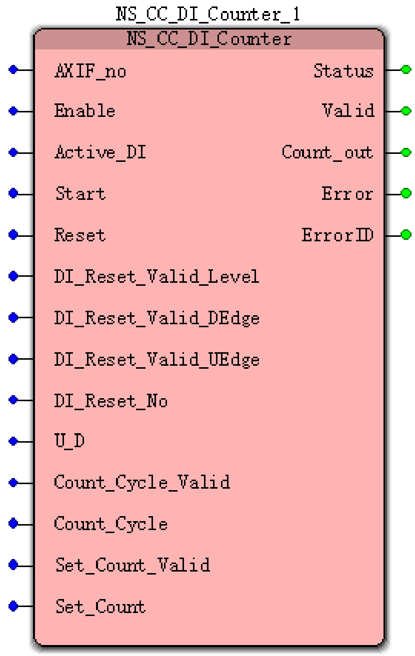
**Step 2 :** NS_CC_OUT_1 recall module configured as follows




**Case 1:**Start NS_CC_Counter_1 (high-speed counter module), when the Enable FALSE to TRUE, the execution instruction module, when the current value of the pulse high-speed counter at 1000 and 5000, Q0.0 will output, OUT will be FALSE to TRUE, the current pulse values outside 1000 and 5000, Q0.0 is not output, OUT by TRUE to FALSE.

# 11.6.9 NS_CC_DI_Counter (DI-speed count instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction for counting high-speed pulse port DI | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (Count axis number) | Set number counter shaft | WORD | 0-6 | Enbale is TRUE |
| Enbale (Execute bit) | when Enable by FALSE Changes toTRUE When the instruction is executed. | BOOL | TRUE or FALSE | |
| Active_DI (DI input channel number) | Set high-speed counting port DI | WORD | 0 ~ 15 | |
| Start (Start bit) | When Start is TRUE, DI start high-speed counter | BOOL | TRUE or FALSE | |
| Reset (Clear bit) | When Reset is TRUE, clears the counter current value | BOOL | TRUE or FALSE | |
| DI_Reset_Valid_ Level (External DI cleared | When DI_Reset_Valid_ Level is TRUE, DI cleared level Clears | BOOL | TRUE or FALSE | |

| | | | | |
|---|---|---|---|---|
| valid bit level) | | | | |
| DI_Reset_Valid_DEdge (External falling edge bit is cleared) | When DI_Reset_Valid_Dedge is TRUE, DI cleared the falling edge clear | BOOL | TRUE or FALSE | |
| DI_Reset_Valid_UEdge (External DI clearing rising significant bit) | When DI_Reset_Valid_Uedge is TRUE, DI cleared to rising cleared | BOOL | TRUE or FALSE | |
| DI_Reset_No (External DI cleared) | Specifies the external clear signal terminal, the input value of 0 to 7 corresponding to the input point I0 ~ I7,8 ~ 15 corresponding to the input point I10 ~ I17 | WORD | 0 ~ 15 | |
| U_D (Bit counting direction) | Counting direction specified counter<br>0: Negative counting direction<br>1: positive counting direction | DINT | 0 or 1 | |
| Count_Cycle_Valid (Cycle Count Valid bits) | When Count_Cycle_Valid is TRUE, start the counting cycles when the counter value reaches the counter Count_Cycle set value, the new count is automatically cleared, constant cycle. Count_Cycle_Valid to FALSE when not enable this feature. | BOOL | TRUE or FALSE | |
| Count_Cycle (Setting DI loop counter value) | When Count_Cycle_Valid is TRUE, start the counting cycles when the counter value reaches the counter Count_Cycle set value, the new count is automatically cleared, constant cycle. Count_Cycle_Valid to FALSE when not enable this feature. | DINT | Positive, negative, | |

| | | | | |
|---|---|---|---|---|
| Set_Count_Valid (Set the current count valid bit DI) | When the Set_Count_Valid is TRUE, the High-speed counter current value is set to Set_Count value | BOOL | TRUE or FALSE | |
| Set_Count (Set the current count value DI) | High-speed counter current value setting | DINT | Positive, negative, 0 | |

**Description:**

(1) body motion controller has 16 digital inputs DI, pure hardware count from the influences of the scan cycle, the maximum input frequency is 5KHz. 8 can be called high-speed counter modules AXIF_no (counter number) from 0 to 6, respectively, are opposite to each other counting methods, without disturbing each other.

(2) This command integrates feature-rich, with variable start / reset, the count is cleared and the external DI counting direction, cycle counting function, set the number of functions,

(3) sources of counter pulse can be selected by Active_DI (DI input channel number), specified in any of 0 to 15.

(4) DI-rich integration speed counter function, may be selected according to requirements, function of the input parameters to an unused blank.

➢ **Output parameters**

| **name** | **Features** | **type of data** | **Output range** |
|---|---|---|---|
| Status (Status bit) | The output parameter is TRUE Representing instructions being executed in | BOOL | TRUE or FALSE |
| Valid (Significant Bit Enable) | The output parameter is TRUE Is a command indicating the control shaft | BOOL | TRUE or FALSE |
| Count_out | Current count of pulses | DINT | Positive, negative, 0 |
| Error | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID | Error Error code when execution instruction | WORD | - |

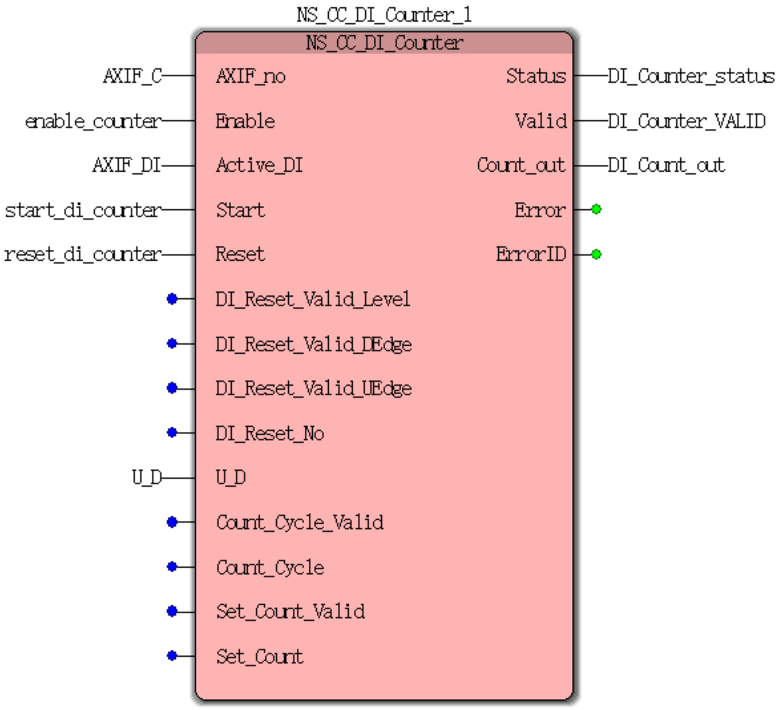
**Demonstration program** Ⅰ

Variable-speed counter start and reset DI

**1, variables, and procedures**

| Variable name | type of data | The initial value |
|---|---|---|

| | | |
|---|---|---|
| NS_CC_DI_Counter_1 | NS_CC_DI_Counter | |
| AXIF_0 | WORD | 0 |
| enbale_counter | BOOL | FALSE |
| AXIF_DI | WORD | 11 |
| start_di_counter | BOOL | FALSE |
| reset_di_counter | BOOL | FALSE |
| U_D | DINT | 1 |
| DI_Count_out | DINT | 0 |




**Case 1:**When enable_counter from FALSE to TRUE, and held, start_di_counter start permission from FALSE to TRUE DI-speed counter, DI_Count_out display the current count of pulses. When reset_di_counter a FALSE to TRUE, Count_out display the current pulse count value is 0, then the count pulse input DI, Count_out display the current pulse count remains at 0 when the Reset TRUE to FALSE, allowing counting resumes .

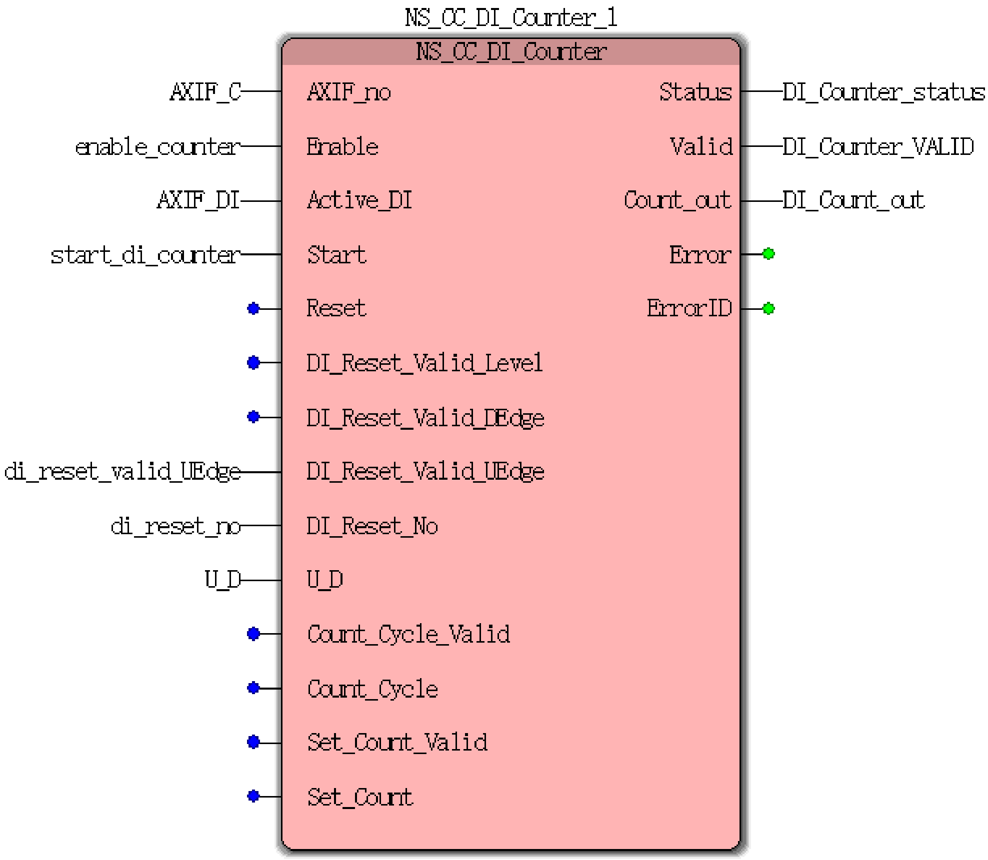
**Demonstration program Ⅱ**

Start and external variables DI DI perform high-speed counter reset

**1. variables, and procedures**

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_DI_Counter_1 | NS_CC_DI_Counter | |
| AXIF_0 | WORD | 0 |
| enbale_counter | BOOL | FALSE |
| AXIF_DI | WORD | 11 |
| start_di_counter | BOOL | FALSE |
| di_reset_no | BOOL | 0 |

| | | |
|---|---|---|
| U_D | DINT | 1 |
| DI_Count_out | DINT | 0 |




**Case 1:**When enable_counter from FALSE to TRUE, and held, start_di_counter start permission from FALSE to TRUE DI-speed counter, DI_Count_out display the current count of pulses. When the rising edge of DI di_reset_no specified, Count_out current pulse count value is cleared.

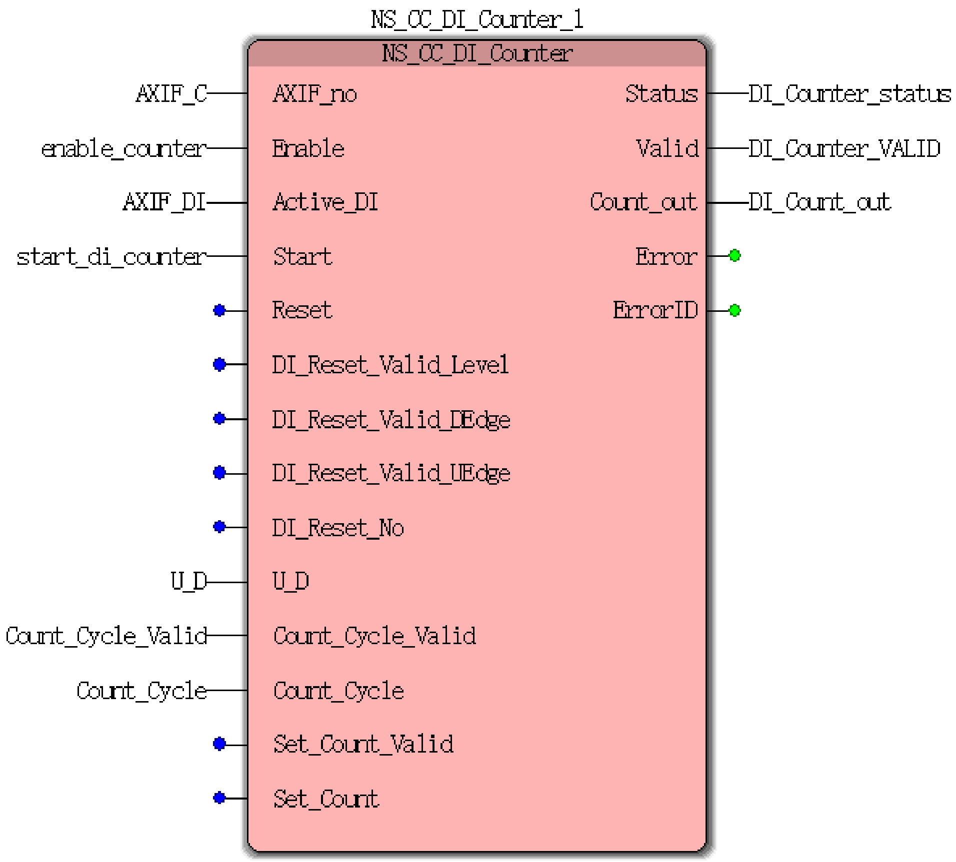
**Demonstration program Ⅲ**

Cycle-count mode for carrying out high-speed counter DI

**1, variables, and procedures**

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_DI_Counter_1 | NS_CC_DI_Counter | |
| AXIF_0 | WORD | 0 |
| enbale_counter | BOOL | FALSE |
| AXIF_DI | WORD | 11 |
| start_di_counter | BOOL | FALSE |
| U_D | DINT | 1 |
| Count_Cycle_Valid | BOOL | TRUE |
| Count_Cycle | DINT | 1000 |

| | | |
|---|---|---|
| DI_Count_out | DINT | 0 |



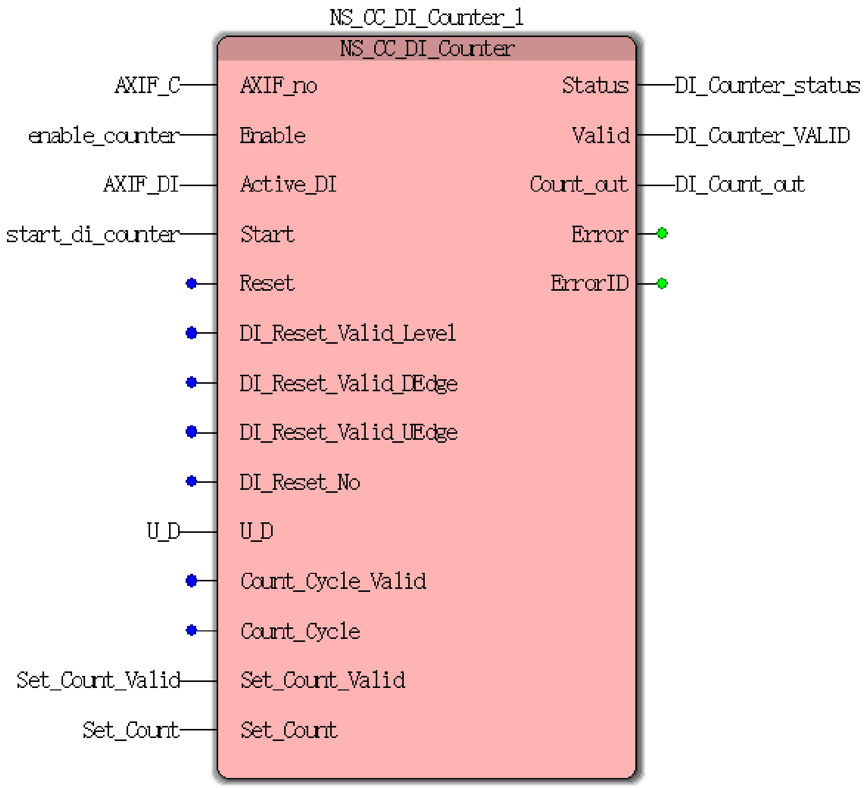

**Case 1:**When enable_counter from FALSE to TRUE, and held, start_di_counter start permission from FALSE to TRUE DI-speed counter, a pulse to DI11, DI_Count_out display the current pulse count is incremented, up to 1000 when DI_Count_out recounting is automatically cleared.

**Demonstration program Ⅳ**

Set the number of modes for carrying out high-speed counter DI

**1. variables, and procedures**

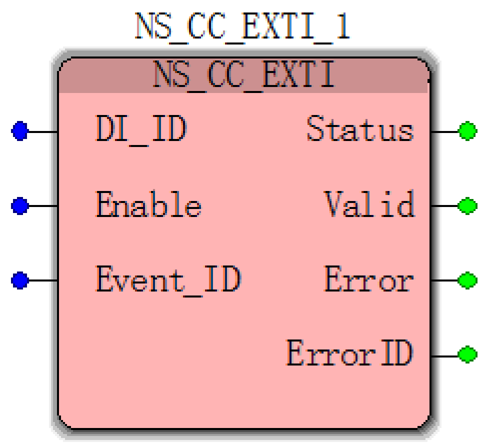
| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_DI_Counter_1 | NS_CC_DI_Counter | |
| AXIF_0 | WORD | 0 |
| enbale_counter | BOOL | FALSE |
| AXIF_DI | WORD | 11 |
| start_di_counter | BOOL | FALSE |
| U_D | DINT | 1 |
| Set_Count_Valid | BOOL | TRUE |
| Set_Count | DINT | 1000 |
| DI_Count_out | DINT | 0 |




**Case 1:**When counter_enable a FALSE to TRUE, start_di_counter start permission from FALSE to TRUE DI-speed counter, DI_Count_out display the current count of pulses. When the change from FALSE to TRUE Set_Count_Valid variable, high-speed counter current value be modified Count_out 1000, and 1000 remain, a TRUE when Set_Count_Valid variable becomes FALSE when DI11 pulse is received, will continue to count the number of pulses on the basis of 1000 increments.

## 11.6.10 NS_CC_EXTI (DI interrupt instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | An interrupt is generated when the specified effective DI, correlate events program will be executed once | VEC-VA-MP-005-MA |




- **Input parameters**

| name | Features | type of data | Setting range (Non-default) | The timing of the entry into force |
|---|---|---|---|---|
| DI_ID | When specifying DI active interrupt, the input value of 0 to 7 corresponding to the input points I0 ~ I7,8 ~ 15 corresponding to the input point I10 ~ I17 | WORD | Currently only supports two interrupt DI0 and DI1<br>0 to 1<br>(Non-default) | |
| Enable | When the Enable FALSE to TRUE, the command execution (open interrupts allowed). | BOOL | TRUE or FALSE | |
| Event_ID | Set interrupt event number (0-5) is consistent with the interrupt routine event number. | WORD | 0-5 (non-default) | |

**Description:**

**(1)**Currently only supports two interrupt DI, respectively external terminals DI0 and DI1

(2) NS_CC_EXTI instruction belongs to an interrupt instruction, when an interrupt is opened, designated DI0 or DI1 effective when an interrupt occurs, the program will branch to the interrupt to the program execution returns to the main program after the completion of the cycle continues to run.

- **Output parameters**

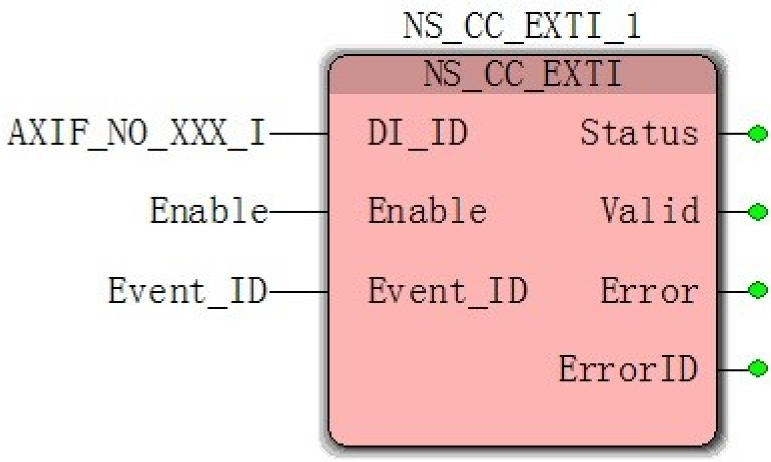
| name | Features | type of data | Output range |
|---|---|---|---|
| The Status (Status bits) | This parameter indicates when the output instruction is being executed is TRUE | BOOL | TRUE or FALSE |
| Valid (valid bit) | Represents the output parameter is TRUE instruction is controlling the shaft | BOOL | TRUE or FALSE |
| Error (error) | Retention | BOOL | TRUE or FALSE |
| ErrorID (error code) | Retention | WORD | Retention |

**Program Example**

**Example: Specify the external interrupt setting DI0**

| Variable name | type of data | The initial value |
|---|---|---|
| NS_CC_EXTI_1 | NS_CC_EXTI | |
| AXIF_NO_XXX_I | WORD | 0 |
| Enable | BOOL | 1 |
| Event_ID | WORD | 0 |

**Step 1:** Call NS_CC_EXTI module, programmed as follows



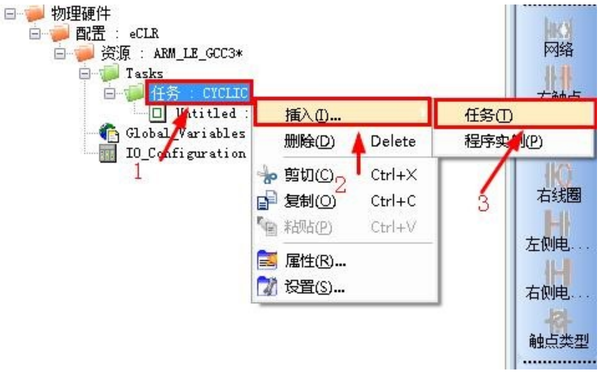

**Step 2:** Interrupt program established in the "Project Tree Window" right "logic POU" select "Insert" and then select "Programs" in the window that pops up to interrupt program named "EN1" (no user name), and choose to write an interrupt routine language "LD language" as shown below.


**Step 3:** When finished, click OK, will automatically add the "logic the POU" to a "EN1" is the name of the program as shown.


**Step 4:** Just inserted "EN1" belongs to the category of the main program or program, as the new "the POU" software automatically selected when loaded into the new project "Task" CYCLIC type; as shown.


**Step 5:** Select under "Project Tree Window", "Hardware", under Resources "task", right-delete "EN1" program and click "OK" we need another new "task type", as shown below.

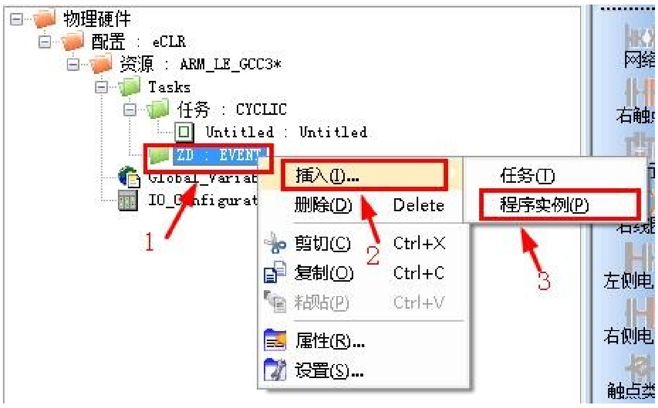
**Step 6:** New "Task Type" right "Tasks" click "insert", and then select the "Tasks" as shown.




**Step 7:** In the pop-up to "Insert" dialog box, fill in "ZD" in the name (no user name) Task Type Select "EVENT" are finished, click "OK." As shown below


**Step 8:** In the pop up "ZD task settings" Pick selection event number "0 Event" "Event_ID" is determined by the value on the interrupt module fill, Pick 0 (exemplary program "Event_ID = 0") 10-17 in FIG. as shown in (special attention not interrupt priority and the priority of the main program as whether those compiled by not downloading or 97% reported error)


**Step 9:** After the insertion procedure of example, the right task, "the EVENT", select "insert" and then select the "program instances" as shown.

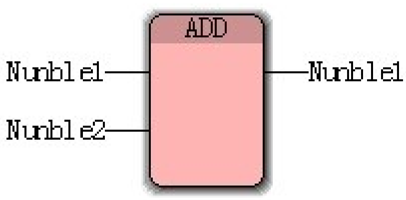



**Step 10:** In the pop-up to "insert box", for the example program named "ZD1 is", select the new program type "EN1"

Click "OK" in the final results are shown in FIG.


**Step 11:** "Project" Return "Project Tree Window" in the window, double-EN1 at POU join a summing module (purpose is to verify whether the interrupt routine is executed correctly)

| variable name | type of data | The initial value |
|---|---|---|
| ADD | ADD | |
| Nunble1 | INT | |
| Nunble2 | INT | 1 |

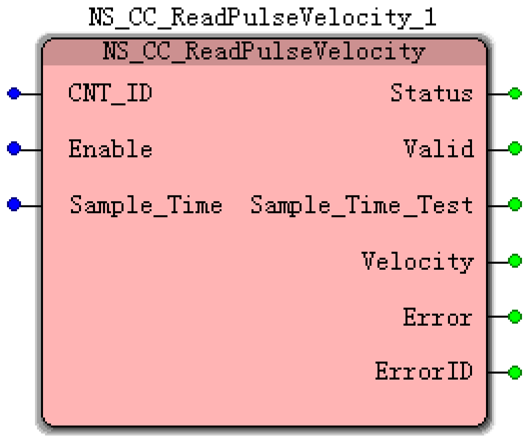



Interrupt this program to create and association has been completed, the following is the analysis process interrupt routine is executed.

**Case 1:**When there is a NS_CC_EXTI Enable module FALSE to TRUE, the interrupt allowed to open, when the external DI0 effective when: interrupt routine is executed once, when the variable value is 1 Nunble1, out after the completion of the interrupt routine returns to the main routine. When the external DI active again when the interrupt routine is executed again, this time is variable Nunble1 2.

## 11.6.11 NS_CC_ReadPulseVelocity (read-axis pulse rate controlled)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the number of motor encoder pulses at the set sampling period | VEC-VA-MP-005-MA |




- **Input parameters**

| name | Features | type of data | Setting range (Non-default) | The timing of the entry into force |
|---|---|---|---|---|
| CNT_ID (Axis No. ID) | To specify the number of motor shaft speed Mining (Corresponding to 0-4 AXIS0-AXIS4) | WORD | 0-4 (Indispensable) | Enable change from FALSE TRUE |
| Enable (execute bit) | When the Execute FALSE to TRUE, the instruction execution starts | BOOL | TRUE or FALSE | - |
| Sample_Time (sampling time) | Set the sampling time (unit: ms) This parameter should MC_AXIS_REF (parameter | REAL | A positive number | With immediate effect |

| | | | | |
|---|---|---|---|---|
| | setting axis) coincides Sample_Time. | | | |

**Description:**

(1) when the instruction starts execution Enable changes from FALSE to TRUE.

Unit reading speed is controlled axis Pulse / s, the sampling period is Sample_Time parameter, calculated from the current shaft speed:

Assumptions: display means of the terminal speed n units / s, the read pulses per second Velocity = 10000 Pulse / s, the number of pulses per motor revolution (moter_PPC) = 10000 Pulse / rad, the terminal mechanism Screw_Lead lead of 1000units the reduction gear ratio of 1: 2

then:

$$n = \text{unit / s} \frac{Velocity}{motor\_PPC} * Screw\_Lead * \frac{1}{2}$$

$$= 500 \text{ units / s}$$

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Status (state) | Retention | BOOL | TRUE or FALSE |
| Valid (effective) | Retention | BOOL | TRUE or FALSE |
| Sample_Time_Test | Retention | REAL | |
| Velocity (current rate) | Display speed of the current sample (unit: Pulse / s) | REAL | Real |
| Error (reserved) | Retention | BOOL | Retention |
| ErrorID (reserved) | Retention | WORD | Retention |

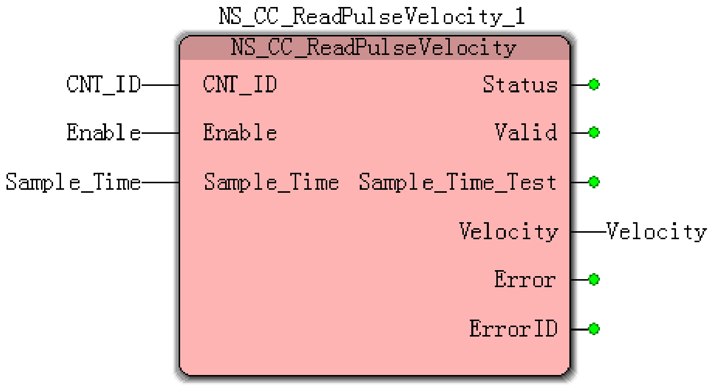
**Program Example**

Example: Read the current speed shaft AXIS0;

**1. variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_ReadPulseVelocity_1 | NS_CC_ReadPulseVelocity | |
| CNT_ID | WORD | 0 |
| Enable | BOOL | 1 |
| Sample_Time | REAL | 10 |
| Velocity | REAL | |

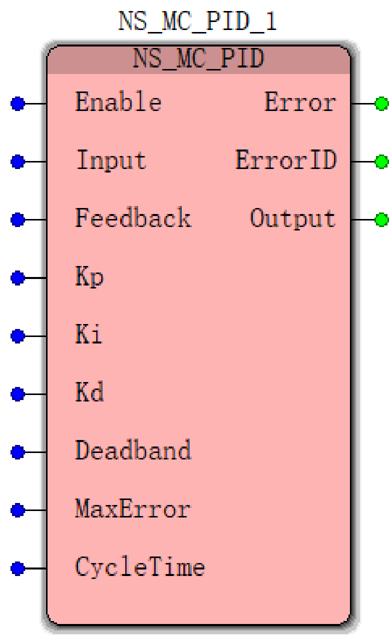
**Step 1:** Following programming instructions call NS_CC_ReadPulseVelocity_1 module




**Case 1:**In the servo axis operation when the Enable FALSE to TRUE, the output module Velocity servo axis current operating speed, if you want to interrupt the conversion speed consistent actuator need to be converted according to the above formula.

## 11.6.12 MC_PID (PID instruction)

| FB / FC | Explanation | Applicable models |
|---|---|---|
| FB | This command is used to specify the PID control variable | VEC-VA-MP-005-MA |




- **Input parameters**

| name | Features | type of data | Setting range (Non-default) | The timing of the entry into force |
|---|---|---|---|---|
| Enable (Execute bit) | When the Enable FALSE to TRUE, the instruction is executed. | BOOL | TRUE or FALSE | Enable from FALSE to TRUE |
| Input (Desired point) | Setting the reference value | LREAL | Real | - |
| Feedback (feedback value) | Feedback value | LREAL | Real | - |
| Kp (Scale factor) | Proportional control | LREAL | Real | - |
| Ki (Integral coefficient) | Integral control | LREAL | Real | - |
| Kd (Differential coefficient) | Differential Control | LREAL | Real | - |

| | | | | |
|---|---|---|---|---|
| Deadband (PID dead band) | PID deadband value represents a set value in the PID operation is not performed; | LREAL | Real | - |
| MaxError (maximum cumulative error) | The maximum cumulative error | LREAL | Real | - |
| CycleTime (PID sampling period) | PID sampling cycle unit (ms) | LREAL | Real | - |

Description:

1) PID control principle

PID regulation is Proportional (ratio), Integral (Integral), abbreviation Differential (differential) of the three, is the most widely used system of continuous adjustment method. The essence of PID regulator, according to a function of proportional, integral and differential value is calculated based on the deviation input, for outputting the calculation result controlled to achieve regulation.

Incremental PID control algorithm

$$\Delta \mathrm{u(n)} = \mathrm{u(n)} - \mathrm{u(n-1)}$$

$\Delta \mathrm{u(n)} = K_p * [e(n) - e(n-1)] + K_i * e(n) + K_d * [e(n) - 2 * e(n-1) + e(n-2)]$

or$\Delta \mathrm{u(n)} = P_{val} + I_{val} + D_{val}$

among them:

Kp: proportional coefficient, the ratio of the field practice of using Kp = 100;

Ki: integral coefficient;$K_i = K_p * T/T_i$

Kd: differential coefficient;$K_d = K_p * T_d/T$

Pval: the ratio of action;$P_{val} = K_p * [e(n) - e(n-1)]$

Ival: integral role; $I_{val} = K_i * e(n)$

Dval: differential effects; $D_{val} = K_d * [e(n) - 2 * e(n-1) + e(n-2)]$

Wherein U (n-1) is the actual control of the amount of time n-1, △ u (n) to control the amount of incremental time n, e (n), e (n-1) and e (n-2 ) are n, n-1 and n-2 time amount of the offset control and the actual value, Ti, Td, and T is the integral time, and derivative time sampling period (CycleTime), wherein the predetermined deviation as follows: setpoint deviation = -Measurements.

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Output (output) | After the PID output | REAL | Float |

## 11.6.13 RTC_S (special register clock)

| address | Explanation | Applicable models |
|---|---|---|
| Special register | And reading the motion controller for modifying an internal clock | VEC-VA-MP-005-MA |

**Write clock address**

| Special register address | Write Functions | Value range |
|---|---|---|
| % MB3.9543 | year | 0 to 255 |
| % MB3.9544 | month | 0 to 255 |
| % MB3.9545 | day | 0 to 255 |
| % MB3.9546 | week | 0 to 255 |
| % MB3.9547 | Time | 0 to 255 |
| % MB3.9548 | Minute | 0 to 255 |
| % MB3.9549 | second | 0 to 255 |
| **Special Note:**Due to the special register address of byte type, maximum value is larger than the value of the register 255 overflows, a modulo operation need to be greater than the value of If. For example, 2018, 2018 can not be filled, it is necessary to 100% by 2018 (modulo)% MB3.9543 to address this in the values into the register address is the address value in the display 18; | | |

**Read clock address**

| Special register address | Read function | Value range |
|---|---|---|
| % MB3.9550 | year | 0 to 255 |
| % MB3.9551 | month | 0 to 255 |
| % MB3.9552 | day | 0 to 255 |
| % MB3.9553 | week | 0 to 255 |
| % MB3.9554 | Time | 0 to 255 |
| % MB3.9555 | Minute | 0 to 255 |
| % MB3.9556 | second | 0 to 255 |

**Perform the modification Clock Address**

| Special register address | Features | Value range |
|---|---|---|
| % MB3.9542 | 1 when the value of the clock execution modification (the need to maintain an approximately 1s, the clock needs to modify the bytes successfully written as 0) | 0 to 255 |

## 11.7 G code instructions

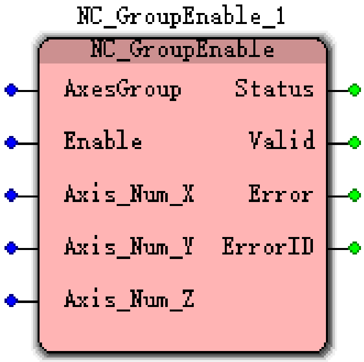
### G code input format

Our VA motion controller supports input format of the G code and the following table:

| Support code | Functional Description | Support axes |
|---|---|---|
| G0 | Rapid positioning | 3-axis |
| G1 | Linear interpolation | 3-axis |
| G2 | Clockwise circular interpolation | 3-axis |
| G3 | Counterclockwise circular interpolation | 3-axis |
| G4 | Timed pause | |
| G17 | Processing the XY plane | |
| G18 | Processing the XZ plane | |
| G19 | Processing the YZ plane | |
| G90 | Absolute size | |
| G91 | Relative size | |
| M0 | The program stops | |
| M1 | Conditional program stop | |
| M2 | End of program | |
| M30 | End of program and return to the program head | |

## 11.7.1 NC_GroupEnable (ENABLE command axis group)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to enable the shaft group | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AxesGroup (Axis group number) | Purports to set the axis of the group can | USINT | 0 | When Enable is TRUE |
| Enable (Enable) | When Enable is TRUE, the instruction is executed | BOOL | TRUE or FALSE | |
| Axis_Num_X (X-axis number) | X axis number setting must be set to 0 | USINT | 0 | When Enable is TRUE |
| Axis_Num_Y (Y-axis number) | Set the Y axis number, it must be set to 1 | USINT | 1 | When Enable is TRUE |
| Axis_Num_Z (Z-axis number) | Axis Z axis is set number must be set to 2 | USINT | 2 | When Enable is TRUE |

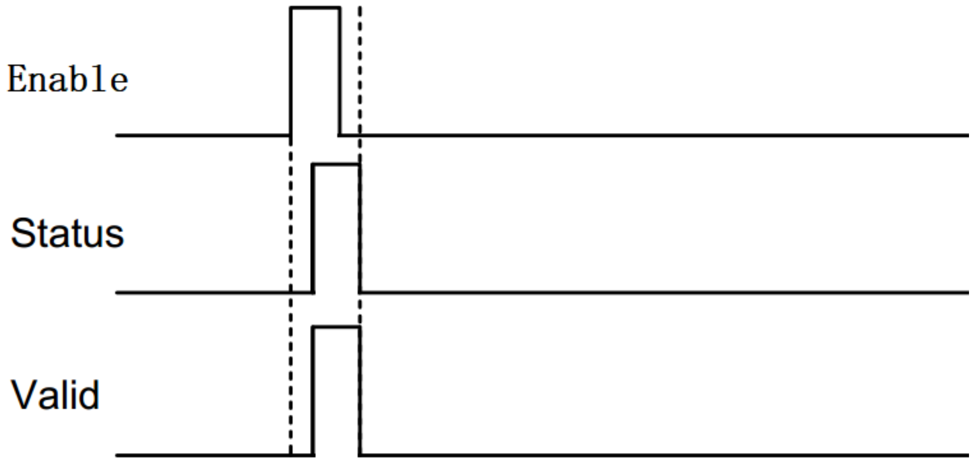
**Description:**It must be set to the control shaft after the shaft is enabled for the corresponding set of operation, when the shaft so that the group can not, linear interpolation (NC_MoveLiner), circular interpolation (NC_MoveCircula), Cartesian robots (NC_CartesianCoordinate) instruction can not be executed.

➢ Output parameters

| name | Features | type of data | Output range |
|---|---|---|---|
| Status | This parameter indicates when the instruction is TRUE control shaft | BOOL | TRUE or FALSE |

| | | | |
|---|---|---|---|
| Valid | The output parameter represents the effective output command is TRUE | BOOL | TRUE or FALSE |
| Error | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| ErrorID | Instruction execution error code error | WORD | - |

- **FIG timing variation output parameter**




- **Function Description**

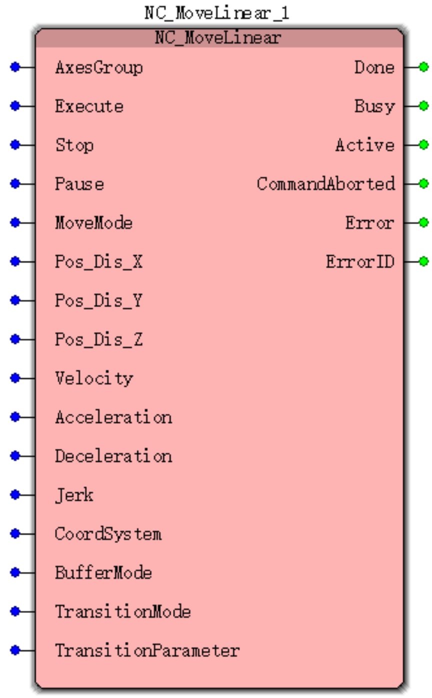
1, when the Enable FALSE to TRUE, a delay period, Status, Valid TRUE simultaneously

2, when the Enable TRUE to FALSE, a delay period, Status, Valid simultaneously FALSE

3. The instruction set for the servo axis controlled release enabled or enabled;

## 11.7.2 NC_MoveLiner (linear interpolation)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control axis linear interpolation function | VEC-VA-MP-005-MA |




➤ **Input parameters**

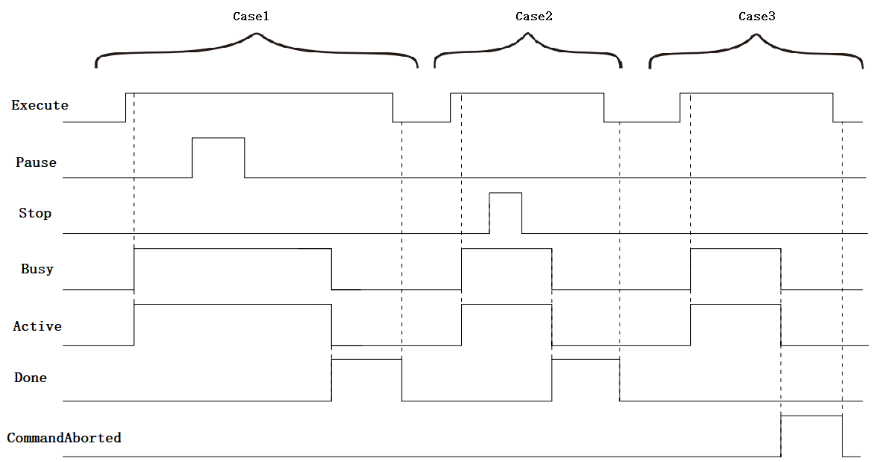
| Name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| AxesGroup (Axis group number) | Purports to set the axis of the group can | USINT | 0 | When the Execute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the execution instruction | BOOL | TRUE or FALSE | - |
| Stop (Stop Bit) | When a Stop FALSE to TRUE, the command to stop. | BOOL | TRUE or FALSE | - |
| Pause (Pause position) | When Pause is TRUE, suspend execution of the instruction | BOOL | TRUE or FALSE | - |
| MoveMode (Movement | When is MoveMode When TRUE, the target | BOOL | TRUE or FALSE | When the Execute |

| | | | | |
|---|---|---|---|---|
| pattern) | position X / Y / Z-axis of the absolute position<br>When FALSE, the target position X / Y / Z-axis relative position | | | from FALSE to TRUE |
| Pos_Dis_X (X-axis target position) | X axis target position setting<br>Unit: unit | LREAL | Positive, negative, 0 | When the Execute from FALSE to TRUE |
| Pos_Dis_Y (Y-axis target position) | Y-axis target position setting<br>Unit: unit | LREAL | Positive, negative, 0 | When the Execute from FALSE to TRUE |
| Pos_Dis_Z (Z-axis target position) | Setting a Z-axis target position<br>Unit: unit | LREAL | Positive, negative, 0 | When the Execute from FALSE to TRUE |
| Velocity (speed) | Synthesis of three axes set maximum speed unit: unit / min | LREAL | A positive number | When the Execute from FALSE to TRUE |
| Acceleration (Acceleration) | Set the maximum value of the three-axis composite acceleration<br>Unit: unit / min2 | LREAL | A positive number | When the Execute from FALSE to TRUE |
| Deceleration | Retention | Retention | Retention | Retention |
| Jerk | Retention | Retention | Retention | Retention |
| CoordSystem | Retention | Retention | Retention | Retention |
| BufferMode | Command transfer mode can only be set to 1 | INT | | |
| TransitionMode | Retention | Retention | Retention | Retention |
| TransitionParameter | Retention | Retention | Retention | Retention |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

- **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period, and Busy Active simultaneously become TRUE, during execution of instructions, a Pause FALSE to TRUE, the operation of the pause instruction is executed, but still Busy Active TRUE and, when changed from TRUE Pause after it is FALSE, the controlled shaft work to finish the operation. When the instruction is complete, Busy and Active becomes FALSE, while Done becomes TRUE. After the Execute cycle by TRUE to FALSE, Done becomes FALSE.

**Case 2:**When the Execute FALSE to TRUE, after a period, Busy becomes TRUE and Active Meanwhile, during the execution of instructions, Stop by the FALSE to TRUE, the end of the execution of instructions, but Busy and Active remains TRUE until the controlled axes stop, Busy and Active becomes FALSE, while Done becomes TRUE. After the Execute cycle by TRUE to FALSE, Done becomes FALSE.

**Case 3:**When the Execute FALSE to TRUE, the instruction is interrupted by another instruction, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE CommandAborted becomes FALSE.

■ **Function Description**

This command is used to set linear interpolation axis, a shaft may be controlled in a group or more axes.

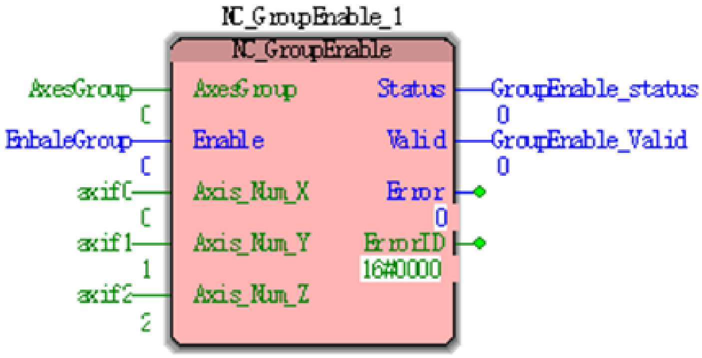
1, the parameter Velocity NC_MoveLiner target speed instruction terminating mechanism, the relationship between the terminal velocity of each shaft speed mechanism is as follows: the terminal means square of the speed of each shaft speed = sum of squares. The command parameter Acceleration, Deceleration target acceleration and target deceleration terminal means, the relationship between the acceleration and deceleration of the terminal means and the addition, the deceleration of each axis are: the terminal means plus (deceleration) = each axis plus (minus) and the square of the speed.

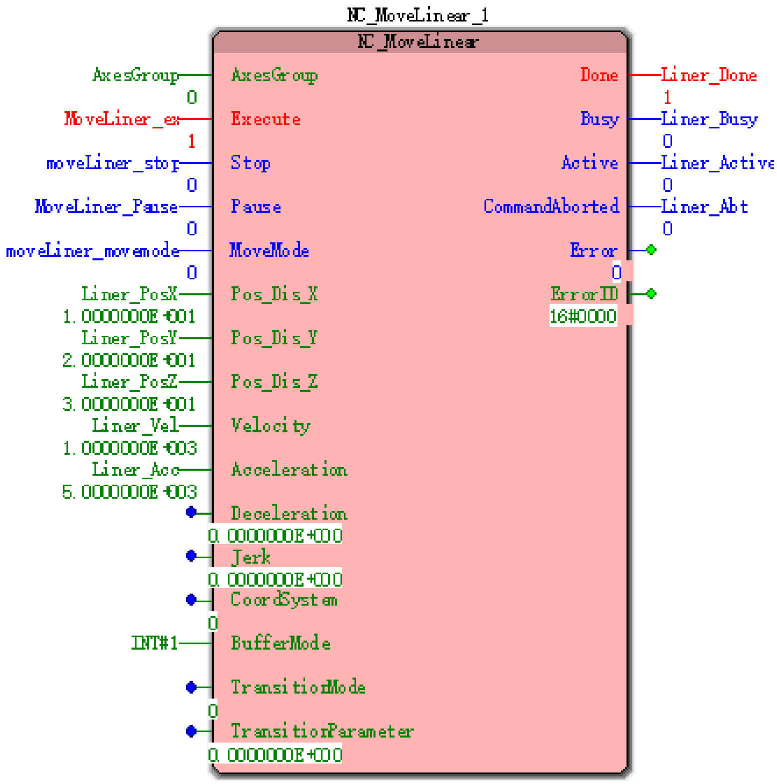
**Examples of a program**

Relative mode execution NC_MoveLiner

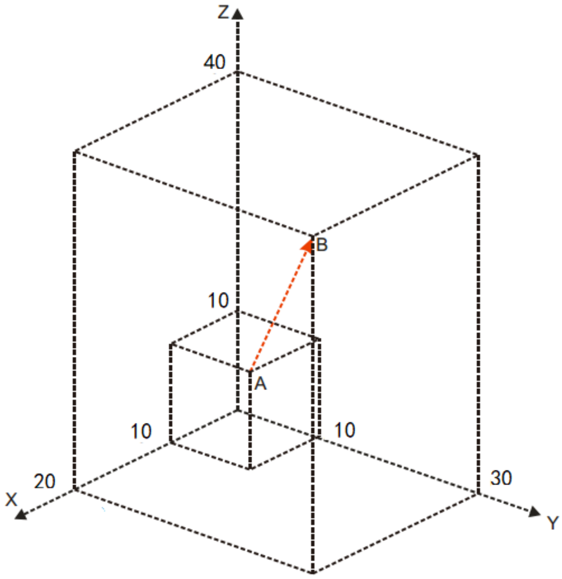
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NC_MoveLiner_1 | NC_MoveLiner | - |
| AxesGroup | USINT | 0 |
| MoveLiner_ex | BOOL | |
| MoveLiner_stop | BOOL | |
| MoveLiner_Pause | BOOL | |
| MoveLiner_Movemode | BOOL | |
| Liner_PosX | LREAL | 10.0 |
| Liner_PosY | LREAL | 20.0 |
| Liner_Vel | LREAL | 30.0 |
| Liner_Acc | LREAL | 1000.0 |
| Liner_Done | BOOL | 5000.0 |
| Liner_Busy | BOOL | |
| Liner_Active | BOOL | |
| Liner_Abt | BOOL | |



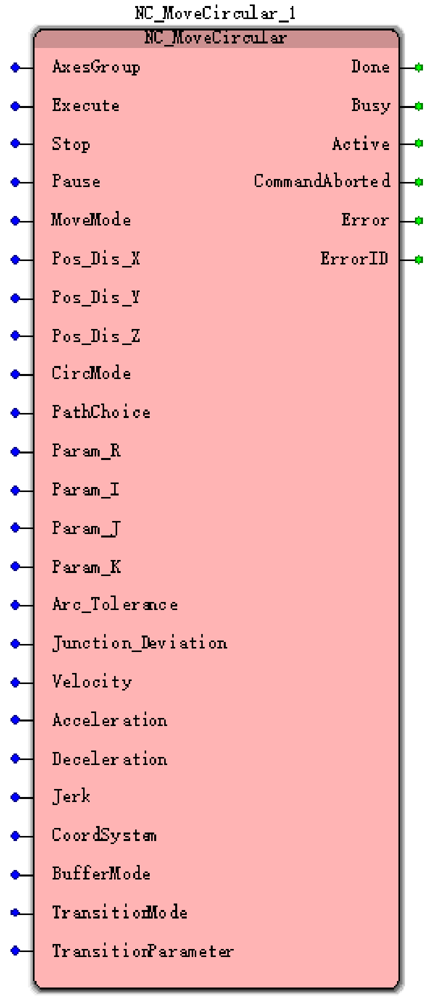




**2, after the instruction is executed, the entire movement as shown below:**


Z
40
B
10
A
10
10
20
30
X
Y

## 11.7.3 NC_MoveCircula (circular interpolation)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control axis linear interpolation function | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| AxesGroup (Axis group number) | Purports to set the axis of the group can | USINT | 0 | When the Execute from FALSE to TRUE |
| Execute (Execute bit) | When the Execute FALSE to TRUE, the execution instruction | BOOL | TRUE or FALSE | - |

| | | | | |
|---|---|---|---|---|
| Stop (Stop Bit) | When a Stop FALSE to TRUE, the command to stop. | BOOL | TRUE or FALSE | - |
| Pause (Pause position) | When Pause is TRUE, suspend execution of the instruction | BOOL | TRUE or FALSE | - |
| MoveMode (Movement pattern) | When is MoveMode When TRUE, the target position X / Y / Z-axis of the absolute position When FALSE, the target position X / Y / Z-axis relative position | BOOL | TRUE or FALSE | When the Execute from FALSE to TRUE |
| Pos_Dis_X (X-axis target position) | X axis target position setting Unit: unit | LREAL | Positive, negative, 0 | When the Execute from FALSE to TRUE |
| Pos_Dis_Y (Y-axis target position) | Y-axis target position setting Unit: unit | LREAL | Positive, negative, 0 | When the Execute from FALSE to TRUE |
| Pos_Dis_Z (Z-axis target position) | Setting a Z-axis target position Unit: unit | LREAL | Positive, negative, 0 | When the Execute from FALSE to TRUE |
| CircMode (Circular interpolation) | Set Circular interpolation<br>0: XY plane circle<br>1: ZX plane circle<br>2: YZ plane circle | INT | 0-2 | |
| PathChoice (Arcuate direction) | The direction of circular interpolation<br>0: clockwise<br>1: counterclockwise | INT | 0,1 | |
| Param_R (radius) | Planar circle radius method set radius and when Preferably selected arc radius is negative; Select inferior arc radius is positive; | LREAL | Positive, negative, 0 (0) | |

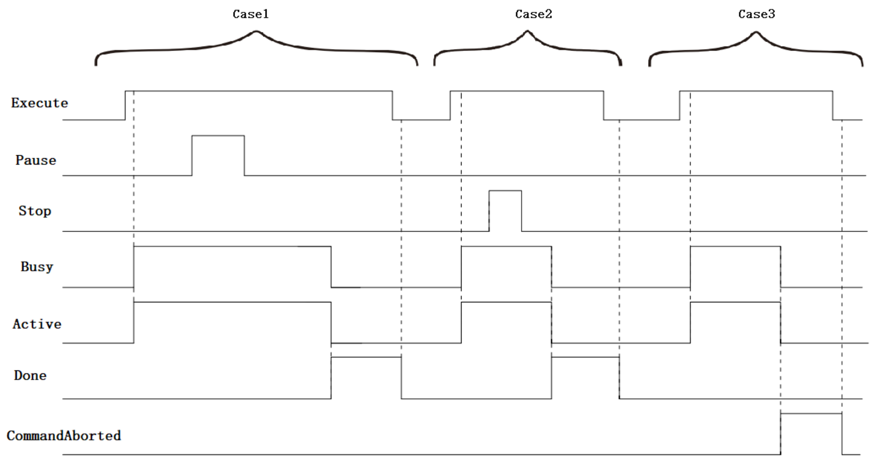
| | | | | |
|---|---|---|---|---|
| | Method selected radius center circle is 0:00 | | | |
| Param_I (X-axis center offset) | When setting method center circle, the center shift amount in the X-axis current position | LREAL | Positive, negative, 0 (0) | |
| Param_J (Y-axis offset center) | When the circle center setting method, the current center position of the Y-axis offset | LREAL | Positive, negative, 0 (0) | |
| Param_K (Z-axis center offset) | When setting method center circle, the circle center in the Z-axis offset current position | LREAL | Positive, negative, 0 (0) | |
| Arc_Tolerance (Arc chord tolerance) | The interpolation process of setting the maximum allowable arc chord tolerance. Arc chord tolerance interpolation = arc length of each segment - the chord length of each segment interpolation | LREAL | | |
| Junction_Deviation (Angle deviation) | Setting each piece of circular interpolation the maximum deviation angle | LREAL | | |
| Velocity (speed) | Maximum speed setting three axes Synthesis Unit: unit / min | LREAL | A positive number | When the Execute from FALSE to TRUE |
| Acceleration (Acceleration) | Set the maximum value of the three-axis composite acceleration Unit: unit / min2 | LREAL | A positive number | When the Execute from FALSE to TRUE |
| Deceleration | Retention | Retention | Retention | Retention |
| Jerk | Retention | Retention | Retention | Retention |
| CoordSystem | Retention | Retention | Retention | Retention |

| | | | | |
|---|---|---|---|---|
| BufferMode | Command transfer mode can only be set to 1 | INT | | |
| TransitionMode | Retention | Retention | Retention | Retention |
| TransitionParameter | Retention | Retention | Retention | Retention |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |

➢ **FIG timing variation output parameter**




**Case 1:**When the Execute FALSE to TRUE, after a period, and Busy Active simultaneously become TRUE, during execution of instructions, a Pause FALSE to TRUE, the operation of the pause instruction is executed, but still Busy Active TRUE and, when changed from TRUE Pause after it is FALSE, the controlled shaft work to finish the operation. When the instruction is complete, Busy and Active becomes FALSE, while Done becomes TRUE. After the Execute cycle by TRUE to FALSE, Done becomes FALSE.

**Case 2:**When the Execute FALSE to TRUE, after a period, Busy becomes TRUE and Active Meanwhile, during the execution of instructions, Stop by the FALSE to TRUE, the end of the execution of instructions, but Busy and Active remains TRUE until the controlled axes stop, Busy and Active becomes FALSE, while Done becomes TRUE. After the Execute cycle by TRUE to FALSE, Done becomes FALSE.

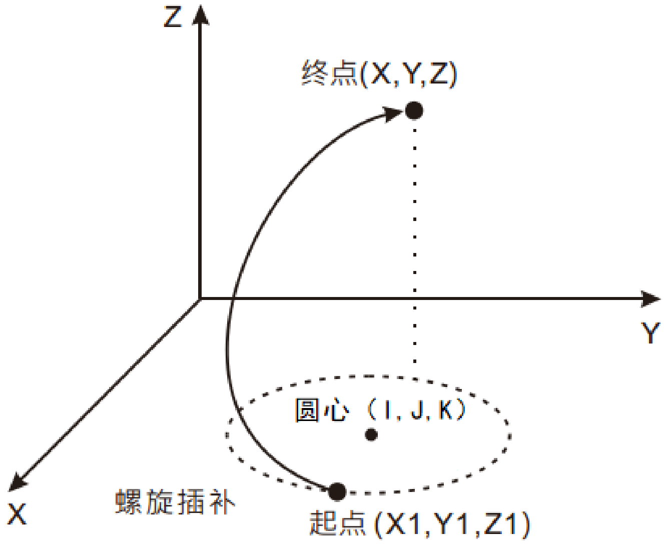
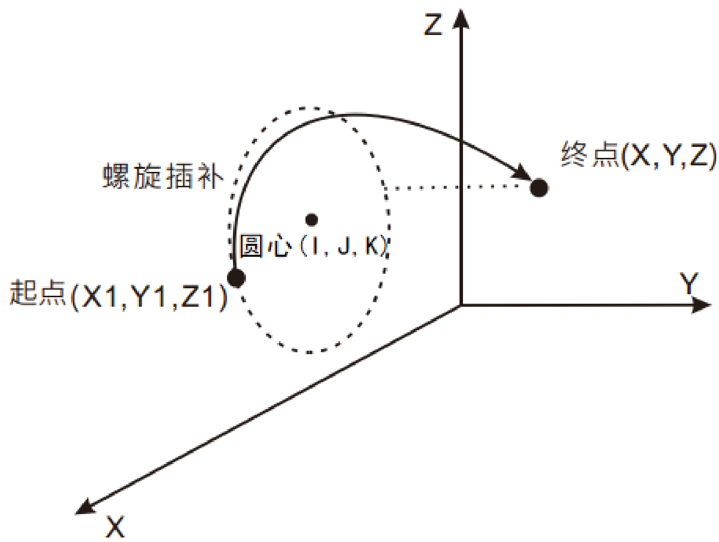
**Case 3:**When the Execute FALSE to TRUE, the instruction is interrupted by another instruction, CommandAborted becomes TRUE, and the Busy Active becomes FALSE; Execute when a TRUE to FALSE CommandAborted becomes FALSE.

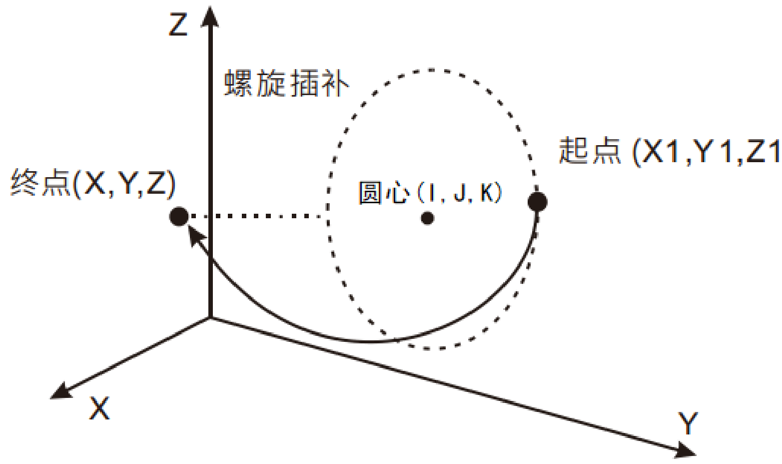
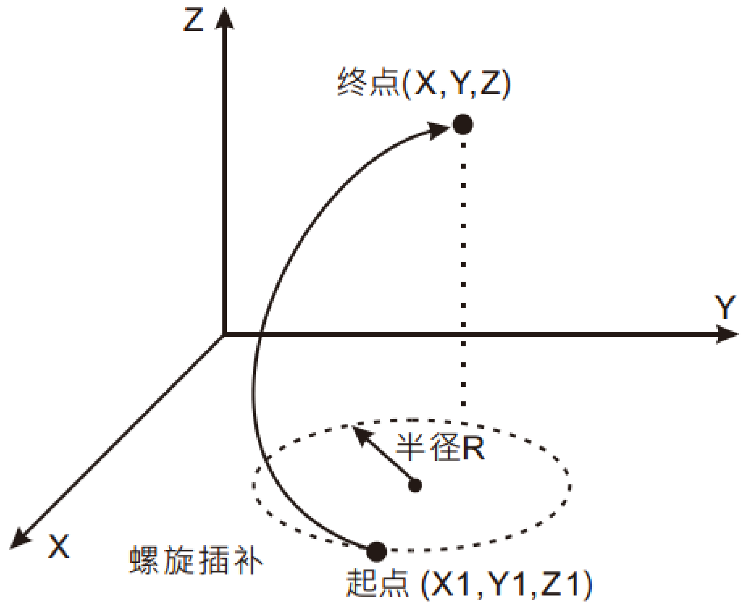
**Function Description:**

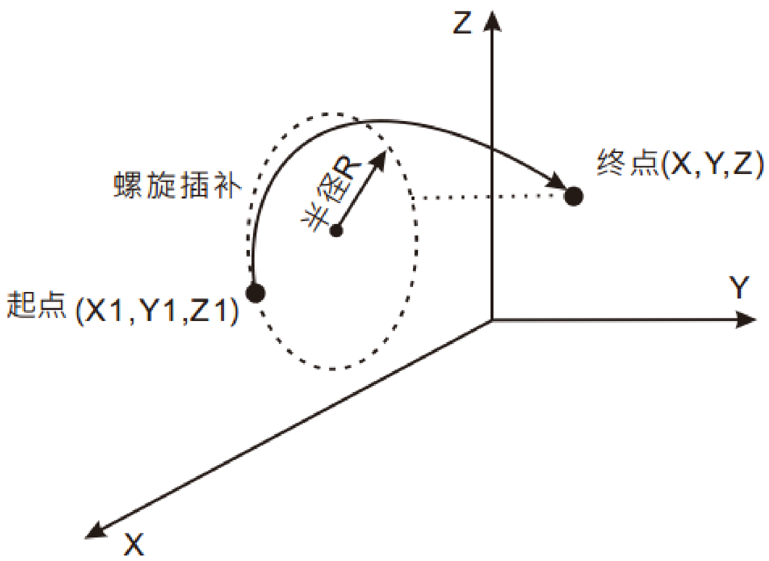
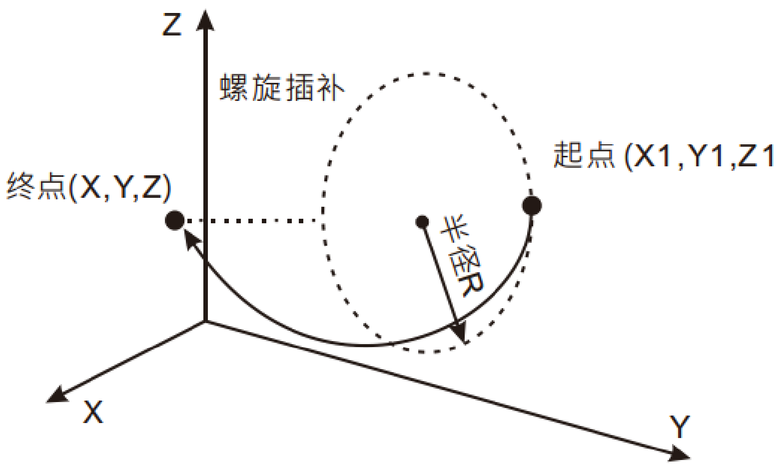
This set of instructions for circular interpolation axis

- **CirMode, Param_R, Param_I, Param_J, Param_K joint decision circular interpolation mode**

| Combinations | Explanation |
|---|---|
| CirMode = 0, Param_R = 0<br>Param_I = A, A $\neq$ 0<br>Param_J = B, B $\neq$ 0<br>Param_K = 0 | Method XY plane circle center. When using this method, Param_I representative of the center position with respect to the X axis offset of the starting position, Param_J representative of the center position with respect to the Y-axis offset of the starting position. |

| | |
|---|---|
| | <br>Figure above, center coordinates I = X1 + Param_I, J = Y1 + Param_J, K = Z1<br>End point coordinates X = Pos_Dis_X, Y = Pos_Dis_Y, Z = Pos_Dis_Z. |
| CirMode = 1,<br>Param_R = 0<br>Param_I = A, A ≠ 0<br>Param_K = B, B ≠ 0<br>Param_J = 0 | Method ZX plane center circle. When using this method, Param_I representative of the center position with respect to the X axis offset of the starting position, Param_K representative of the center position with respect to the Z-axis offset of the starting position.<br><br>Figure above, center coordinates I = X1 + Param_I, J = Y1, K = Z1 + Param_K,<br>End point coordinates X = Pos_Dis_X, Y = Pos_Dis_Y, Z = Pos_Dis_Z. |
| CirMode = 2,<br>Param_R = 0<br>Param_J = A, A ≠ 0<br>Param_K = B, B ≠ 0 | Method YZ plane center circle. When using this method, Param_J representative of the center position with respect to the Y-axis offset of the starting position, Param_K representative of the center position in the Z-axis offset relative to the start position. |

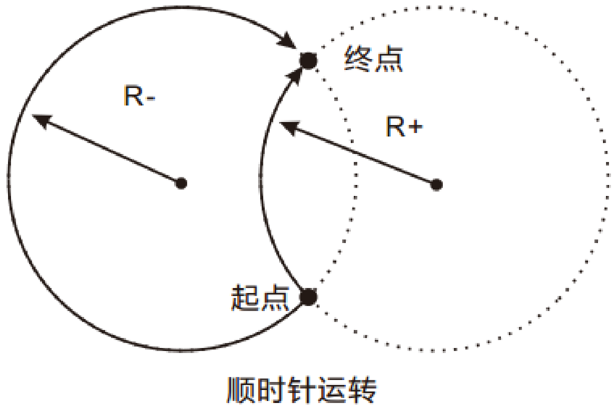
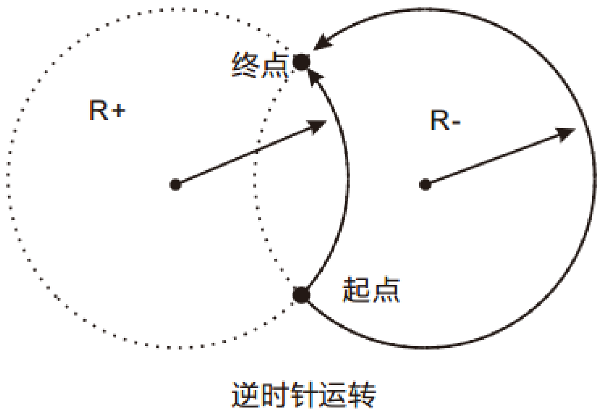
| | |
|---|---|
| Param_I = 0 |  <br>Figure above, the circle center, I = X1, J = Y1 + Param_J, K = Z1 + Param_K<br>End point coordinates X = Pos_Dis_X, Y = Pos_Dis_Y, Z = Pos_Dis_Z. |
| CirMode = 0,<br>Param_R = R, R ≠ 0 | XY plane circle radius method. When using this method, the value represents the radius of the circle Param_R on the XY plane, then Param_R greater than 0, the minor arc of the circular arc; Param_R less than 0, preferably circular arc-arc<br> <br>The radii of the above figure R.<br>End point coordinates X = Pos_Dis_X, Y = Pos_Dis_Y, Z = Pos_Dis_Z. |
| CirMode = 1,<br>Param_R = R, R ≠ 0 | ZX plane circle radius method. When using this method, the value represents the radius of the circle Param_R ZX plane, this time Param_R greater than 0, the minor arc of the circular arc; Param_R less than 0, preferably circular arc-arc |

| | |
|---|---|
| | <br>The radii of the above figure R.<br>End point coordinates X = Pos_Dis_X, Y = Pos_Dis_Y, Z = Pos_Dis_Z. |
| CirMode = 2,<br>Param_R = R, R ≠ 0 | YZ plane circle radius method. When using this method, the value represents the radius of the circle Param_R the YZ plane, this time Param_R greater than 0, the minor arc of the circular arc; Param_R less than 0, preferably circular arc-arc<br><br>The radii of the above figure R.<br>End point coordinates X = Pos_Dis_X, Y = Pos_Dis_Y, Z = Pos_Dis_Z. |

- **PathChoice**

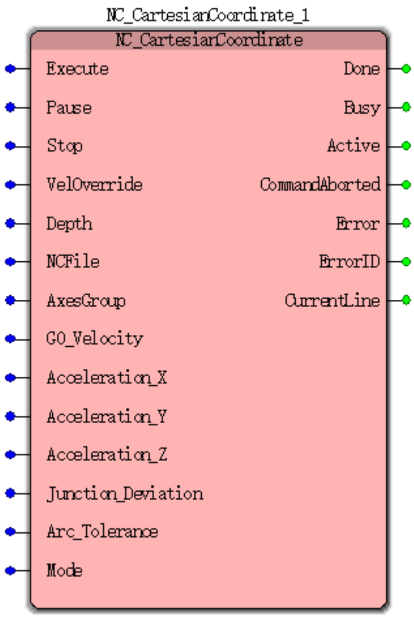
This parameter determines the direction of circular interpolation

| Parameter Value | Explanation |
|---|---|
| 0 | Group shaft clockwise circular interpolation on the specified plane and a circular arc. Radius method as an example, a circular arc trajectory as shown below: |

| | |
|---|---|
| |  |
| 1 | Group shaft counterclockwise circular interpolation on the specified plane and a circular arc. Radius method as an example, a circular arc trajectory as shown below:<br> |

## 11.7.4 NC_CartesianCoordinate (Cartesian robot command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to control the motion interpolated Cartesian axes according to G code | VEC-VA-MP-005-MA |




➢ **Input parameters**

| name | Features | type of data | Predetermined area (Default value) | The timing of the entry into force |
|---|---|---|---|---|
| Execute (Execute bit) | When the Execute FALSE to TRUE, the execution instruction | BOOL | TRUE or FALSE | When the Execute FALSE to TRUE- |
| Pause (Pause position) | When Pause is TRUE, suspend execution of the instruction | BOOL | TRUE or FALSE | - |
| Stop (Stop Bit) | When a Stop FALSE to TRUE, the command to stop. | BOOL | TRUE or FALSE | - |
| VelOverride (Speed overshoot value) | Speed overshoot value (%) | LREAL | 1≤ VelOverride ≤500 | |
| Depth (Depth buffer) | Depth buffer (provided 2 represents 1, 16 represents 15 is provided). 16 normal setting, if each small segment wants to start and | UINT | | |

| | | | | |
|---|---|---|---|---|
| | stop speed setting 2 0 | | | |
| NCFile (NC file) | NC file selection | UINT | | |
| AxesGroup (Axis groups) | Axis Group number must be set to 0 | USINT | 0 | |
| G0_Velocity (G0 speed) | Maximum speed setting command G0 Unit: unit / min | LREAL | | |
| Acceleration_X (X-axis acceleration) | Setting the maximum X-axis acceleration Unit: unit / min2 | LREAL | | |
| Acceleration_Y (Y-axis acceleration) | Set the maximum Y-axis acceleration Unit: unit / min2 | LREAL | | |
| Acceleration_Z (Z-axis acceleration) | Set the maximum Z-axis acceleration Unit: unit / min2 | LREAL | | |
| Junction_Deviation (Angle deviation) | Setting each piece of circular interpolation the maximum deviation angle | LREAL | | |
| Arc_Tolerance (Arc chord tolerance) | The interpolation process of setting the maximum allowable arc chord tolerance. Arc chord tolerance interpolation = arc length of each segment - the chord length of each segment interpolation | LREAL | | |
| Mode (End processing mode) | When the value is 0, no interpolation processing of the small end of each segment. When the value is 1, the processing of the small end of each interpolation segment | INT | 0,1 | |

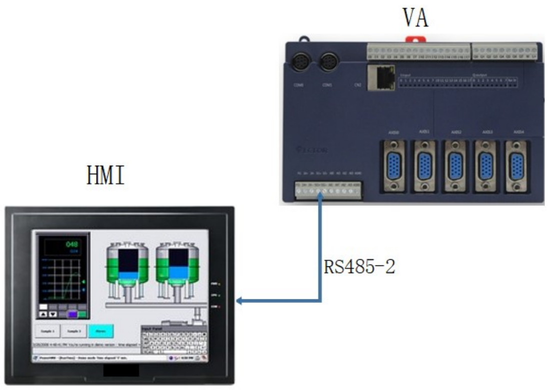
➢ **Output parameters**

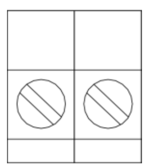
| name | Features | type of data | Output range |
|---|---|---|---|
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| CommandAborted (interruption) | The output parameter is TRUE representing instructions is interrupted | BOOL | TRUE or FALSE |
| Error (error) | It represents execution of the faulting instruction when the output instruction is TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Error Error code when execution instruction | WORD | - |
| CurrentLine | G code number of the currently executing row | UDINT | |

# XII Communication Settings

## 12.1 motion controller and HMI communication

### 12.1.1 motion controller and human-machine wiring shown below




<table>
<tr><td rowspan="3">RS485-2</td><td rowspan="3"></td><td>Terminals</td><td>definition</td></tr>
<tr><td>SG +</td><td>RS-485 signals are being</td></tr>
<tr><td>SG-</td><td>RS-485 signals negative</td></tr>
</table>

### 12.1.2 HMI and motion controller communication format

<table>
<tr><th>function name</th><th>format</th><th>Factory settings</th></tr>
<tr><td>PLC communication protocol</td><td>MODBUS RTU</td><td>MODBUS RTU</td></tr>
<tr><td>Communication Interface Type</td><td>RS 485-2</td><td>RS 485-2</td></tr>
<tr><td>Baud Rate</td><td>9600</td><td>9600</td></tr>
<tr><td>Data bits</td><td>8</td><td>8</td></tr>
<tr><td>Parity</td><td>Even parity</td><td>Even parity</td></tr>
<tr><td>Stop bits</td><td>1</td><td>1</td></tr>
<tr><td>Station No</td><td>111</td><td>111</td></tr>
<tr><td colspan="3">Default factory settings when communicating with the peripheral controller operation, the user can modify the station number and baud rate via a special register;<br>Special Registers:% MB3.4010 (station number);% MB3.4011 (baud rate);<br>Special Note: * = 4800 baud rate input value (e.g.: 9600% MB3.4011 then the filler 2; then the baud rate of 19200% MB3.4011 fill 4, and so on)</td></tr>
</table>

For example: the station number 10 to 19200 baud other formats remain unchanged. Programming the following initial value% MB3.4010 filler 10;% MB3.4011 initial value fill 4; download compiled, after power controller station number and baud rate modification after successful completion.

## 12.1.3 motion controller and human-machine communication address correspondence address

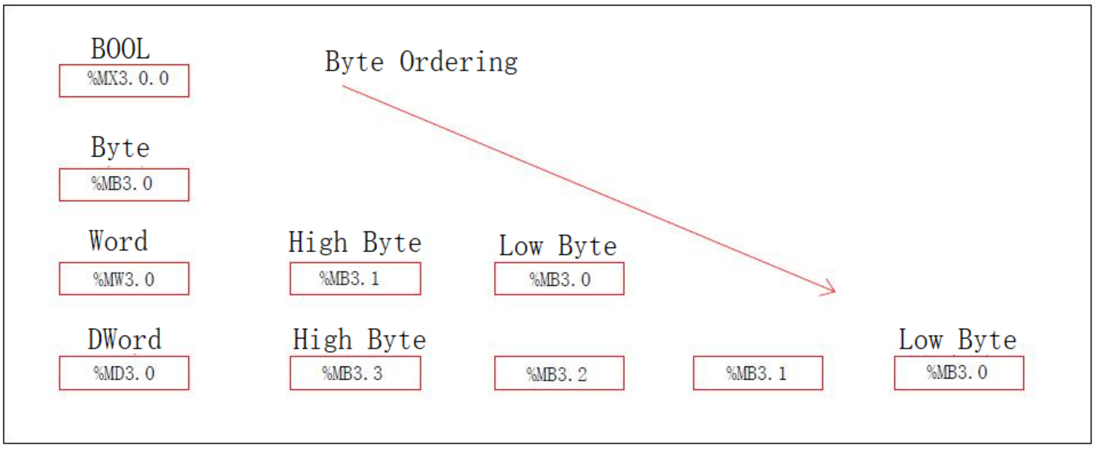
(1) addresses the relationship between motion controller

Data access must specify the address, the beginning address% MX3, wherein "X" may be a bit "% MX3.";. May be byte "% MB3."; The word may be "% MW3."; May also be double word "% MD3."; "." integer plus a decimal point with stored address, expressed as% MX3.0.0 data area memory map byte 0 bit 0, the characteristic data table address

| No. | Prefix | | Agreed definitions | type of data |
|---|---|---|---|---|
| 1 | Location prefix | I | Input mapping area | |
| 2 | | Q | Output mapping area | |
| 3 | | M | Intermediate variables mapping area | |
| 4 | The size prefixes | X | Place | BOOL |
| 5 | | B | Byte (8 bits) | BYTE |
| 6 | | W | Word (16 bits) | WORD |
| 7 | | D | Double word (32) | DWORD |
| 8 | | L | Long (64-bit) | LREAL |

(2) the relationship between the motion controller address

The relationship between the address byte, word and double word is a double word contains two words or four bytes comprising, in the following% MX3.0.0,% MB3.0,% MW3.0 and% MD3.0 an example of the relationship between an address byte, word and double word and the data arrangement as shown below:




Such as: a hexadecimal number stored in 16% MW3.0 # 1234 in the presence of 16% MB3.0 # 34, the # 12 is stored in 16% MB3.1 in. If the procedure for bit operations, it will affect the place where the byte, word and double-word and vice versa.

Examples of variable address

% IX1.3 denotes digital input bit map area 3 of a byte;

% QX0.0 digital output mapping area indicates the first byte 0 bit 0;

% MX3.0.0 represents the variable region of the intermediate byte 0 bit 0;

% MD3.4 represents the variable region of the intermediate 4 1 byte double word;

(3) motion controller and human machine address correspondence:

Motion controller address = (HMI address -1) * 2 (Wei-lun pass easily and Traditional HMI)

**Bit operation:**

| PLC Address Type | HMI Address Type |
|---|---|
| % IX0.0 | no |
| % QX0.0 | no |
| % MX3.0.0 | 0X |
| Example: PLC address type %MX3.6.0 corresponds to HMI address :Type 0X Address 4 | |

**Byte operation:**

| PLC Address Type | HMI Address Type |
|---|---|
| % IB0.0 | no |
| % QB0.0 | no |
| % MB3.0.0 | 4 (3) X |
| Example: PLC address type %MB3.10 corresponds to HMI address : Type 4(3)X Address 6 | |

**Word operation:**

| PLC Address Type | HMI Address Type |
|---|---|
| % IW0.0 | no |
| % QW0.0 | no |
| % MW3.0.0 | 4 (3) X |

| |
|---|
| Example: PLC address type %MW3.14 corresponds to HMI address : Type 4(3)X Address 8 |

**Double operation**

| PLC Address Type | HMI Address Type |
|---|---|
| % ID0.0 | no |
| % QD0.0 | no |
| % MD3.0.0 | 4 (3) X |
| Example: PLC address type %MD.20 corresponds to HMI address : Type 4(3)X Address 11 | |

# Appendix I Programming Considerations

When using the ladder programming language, to avoid many of the same variable operation with the same instruction. Scan sequence ladder program, the program is from top to bottom, left to right. When programming, do not copy and paste commands in the editing area (function blocks), or compiler errors, you can drag the editing area from the right side of the Edit wizard, or type the command (function blocks) name directly in the editing area.

The following command does not support:

GET_CHAR
GET_ERROR
GET_ERROR CATALOG
GET_SYM
CLR_OUT
COLD_RESTART
CONTINUE
HO_ RESTART
WRITE_RETAIN
WARM_RESTART

Written instructions PDD variables, such as WR_BOOL_BY_SYMFOR cycle: the number of steps can not be too long, such as 15,000, or may error, a program can not be more larger the FOR loop (such as 10,000 steps) appears, or it may be wrong. The maximum length of the array ARRAY is 32767.

# Appendix II ASCII Code Table

ASCII stands for American Standard Code for Information Interchange, is the American Standard Code for Information Interchange acronym, letters or numbers or symbols ASCII code definitions from one hundred twenty-eight numbers 0-127 of the representatives of all computers use the ASCII between each other in the same document can be read without the different results and significance. Since only seven bits (BIT) can represent from 0 to 127, most computers use to access 8-bit characters (CHARACTER SET), in Table 1 and Table 2. A number between 128 to 255 can be used to represent another group of one hundred twenty-eight symbols, called EXTENDED ASCII, see Table 3. Table 1 decimal 0 to 31 and 127 to control the character table 232 for the printable character to 126; 128 to 255 in Table 3 for the extended ASCII code.

Table 1 control character (Table Hx hexadecimal, Dec decimal)

| Dec | HX | character | description | Dec | Hx | description |
|---|---|---|---|---|---|---|
| 0 | 0 | NULL | Null / null character | 17 | 11 | Device Control 1/1 Control Device |
| 1 | 1 | SOX | Start Of Heading / title start | 18 | 12 | Device Control 1/2 Control Device |
| 2 | 2 | STX | Start Of Text / text start | 19 | 13 | Device Control 1 / controlling apparatus 3 |
| 3 | 3 | ETX | End End Of Text / text | 20 | 14 | Device Control 1/4 control device |
| 4 | 4 | EOT | End Of Transmission / End of Transmission | twenty one | 15 | Negtive Acknowledge / reject |
| 5 | 5 | ENO | Enquiry / Request | twenty two | 16 | Synchronous Idle / sync idle |
| 6 | 6 | ACK | Acknowledge / notified | twenty three | 17 | End of Trans. Block / End transport block |
| 7 | 7 | BEL | Bell / Bell | twenty four | 18 | Cancel / Cancel |
| 8 | 8 | BS | Backspace / Backspace | 25 | 19 | End of Medium / medium interrupted |
| 9 | 9 | HT | Horizontal Tab / horizontal tab | 26 | 1A | Substitute / replacement |
| 10 | OA | LF | NL line feed, new line / line feed | 27 | 1B | Escape / overflow |
| 11 | OB | VT | Vertical Tab / | 28 | 1C | File Separator / file delimiter |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | vertical breaks made table | | | |
| 12 | OC | FF | NP form feed, new page / feed | 29 | 1D | Group Separator / packet identifier |
| 13 | OD | CR | Carriage Return / Enter | 30 | 1E | Record Separator / record separators |
| 14 | OE | SO | Shift Out / stop switch n | 31 | 1F | Unit Separator / separator unit |
| 15 | 0F | SL | Shift In / enable switch | 127 | 7F | Delete / Delete |
| 16 | 10 | DLE | Data Link Escape / Data Link Escape | | | |

Table 2 printable characters (Table Hx hexadecimal, Dec decimal)

| Dec | Hx | control system word symbol | Dec | Hx | Control characters | Dec | Hx | Control characters | Dec | Hx | control system word symbol |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 20 | (Space) | 56 | 38 | 8 | 80 | 50 | P | 104 | 68 | h |
| 33 | twenty one | ! | 57 | 39 | 9 | 81 | 51 | Q | 105 | 69 | i |
| 34 | twenty two | " | 58 | 3A | : | 82 | 52 | R | 106 | 6A | j |
| 35 | twenty three | # | 59 | 3B | ; | 83 | 53 | X | 107 | 6B | k |
| 36 | twenty four | $ | 60 | 3C | < | 84 | 54 | T | 108 | 6C | l |
| 37 | 25 | % | 61 | 3D | = | 85 | 55 | U | 109 | 6D | m |
| 38 | 26 | & | 62 | 3E | > | 86 | 56 | V | 110 | 6E | n |
| 39 | 27 | , | 63 | 3F | ? | 87 | 57 | W | 111 | 6F | o |
| 40 | 28 | ( | 64 | 40 | @ | 88 | 58 | X | 112 | 70 | p |
| 41 | 29 | ) | 65 | 41 | A | 89 | 59 | Y | 113 | 71 | q |
| 42 | 2A | * | 66 | 42 | B | 90 | 5A | Z | 114 | 72 | r |
| 43 | 2B | + | 67 | 43 | C | 91 | 5B | [ | 115 | 73 | s |
| 44 | 2 C | , | 68 | 44 | D | 92 | 5C | / | 116 | 74 | t |
| 45 | 2D | - | 69 | 45 | E | 93 | 5D | ] | 117 | 75 | u |
| 46 | 2E | . | 70 | 46 | F | 94 | 5E | ^ | 118 | 76 | v |
| 47 | 2F | / | 71 | 47 | G | 95 | 5F | - | 119 | 77 | w |
| 48 | 30 | 0 | 72 | 48 | H | 96 | 60 | , | 120 | 78 | x |
| 49 | 31 | 1 | 73 | 49 | I | 97 | 61 | a | 121 | 79 | y |
| 50 | 32 | 2 | 74 | 4A | J | 98 | 62 | b | 122 | 7A | z |
| 51 | 33 | 3 | 75 | 4B | K | 99 | 63 | c | 123 | 7B | { |
| 52 | 34 | 4 | 76 | 4C | L | 100 | 64 | d | 124 | 7C | \| |
| 53 | 35 | 5 | 77 | 4D | M | 101 | 65 | e | 125 | 7D | } |
| 54 | 36 | 6 | 78 | 4E | N | 102 | 66 | f | 126 | 7E | ~ |
| 55 | 37 | 7 | 79 | 4F | O | 103 | 67 | g | | | |

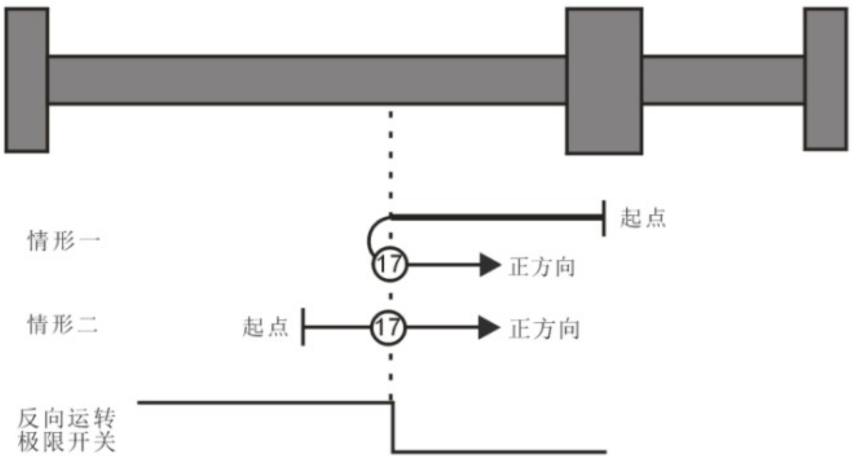
# Appendix III Homing Mode Description

Our motion controllerThere are many back homeReturnmode. The user can select the appropriate mode depending on the origin of the reset to zero field conditions and process requirements .

- **Homing mode operated in reverse limit switch 17 depending on the origin regression**

Case 1: MC_Home instruction execution when the reverse limit switch is in the low state, the shaft starts to first speed reverse movement, when the reverse limit switch is encountered high, changing the direction of motion and starts to move to 2nd speed, when faced with the reverse limit switch is in the low position is the home position.

Case 2: when the reverse limit switch is performed at a high state MC_Home command axis starts moving forward 2nd speed, when reverse limit switch is encountered in the low position is the home position.

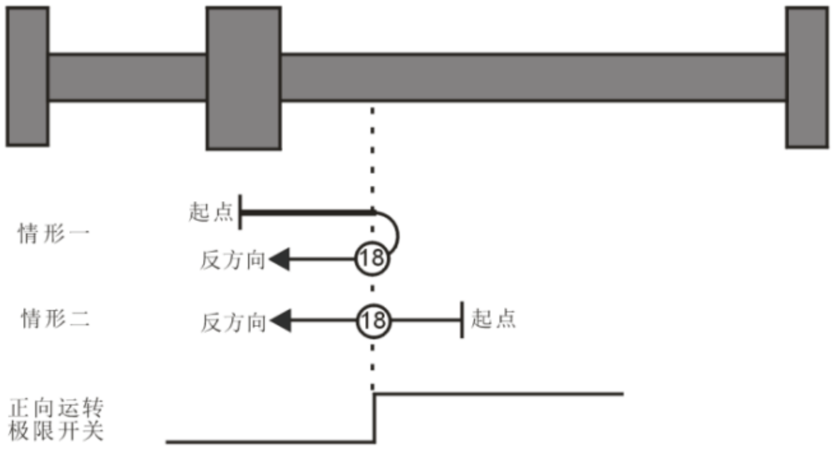



取决于反向运转极限开关的原点回零，上图中的表示17原点回零模式17

- **Homing mode operation 18 depends on the forward limit switch Homing**

Case 1: when a forward instruction execution MC_Home in the low limit switch, starting with the shaft moving forward first speed, when it encounters a forward operation at a high limit switch, changing the direction of movement and starts to move to 2nd speed, in the forward position limit switch operation state is in the low position of the origin.

Case 2: MC_Home instruction execution when a forward operation limit switch at a high state, the shaft directly in 2nd speed start reverse motion, the forward operation limit position when the switch is in the low state origin position

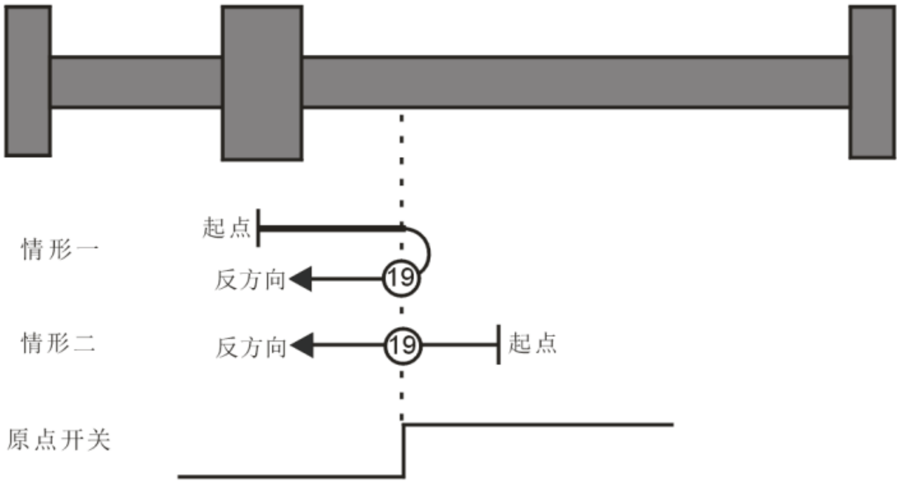



取决于正向运转极限开关的原点回零，上图中的表示18原点回零模式18

- **Homing mode switch 19 depending on the origin of the OPR**

Case 1: When performed at a low MC_Home home switch command axis starts forward motion to first speed, when it comes at a high origin switch, changing the direction of motion and starts to move to 2nd speed, when faced origin switch is in the low position is the home position.

Case 2: When performed in a high MC_Home home switch command, the shaft directly in 2nd speed reverse movement begins, when it comes to the home switch in the low position is the home position.

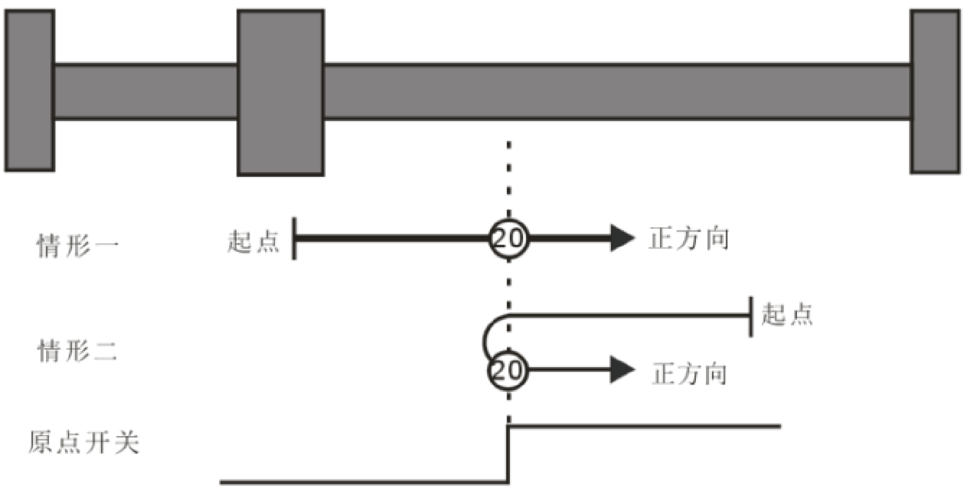



取决于原点开关的原点回零，上图中的表示19原点回零模式19

- **Homing mode switch 20 depending on the origin of the OPR**

Case 1: When performed at a low MC_Home home switch command, the shaft starts first speed forward motion, when faced with the origin position is the home position switch is high.

Case 2: When performed in a high MC_Home home switch command, the shaft directly in 2nd speed reverse movement begins, the home switch changes encountered when the movement direction and are low in 2nd speed starts to move. When faced with the origin switch again at a high position is the home position.

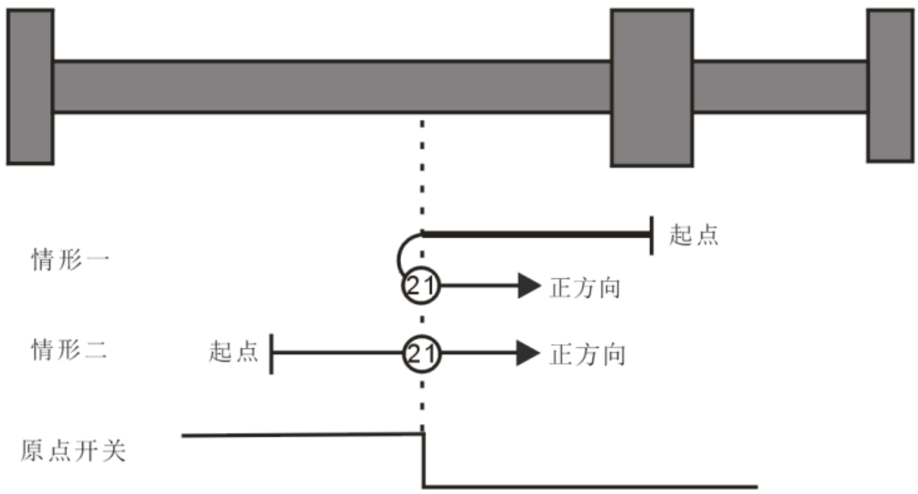



取决于原点开关的原点回零，上图中的表示20原点回零模式20

- **Homing mode switch 21 depending on the origin of homing**

Case 1: When the instruction execution MC_Home home switch is low, the shaft starts to first speed reverse movement, when the switch is in the home encounters high, changing the direction of motion and starts to move to 2nd speed, when faced origin switch is in the low position is the home position.

Case 2: When performed in a high MC_Home home switch instruction, the start of direct axis 2nd speed forward motion, when faced with the home switch in the low position is the home position.

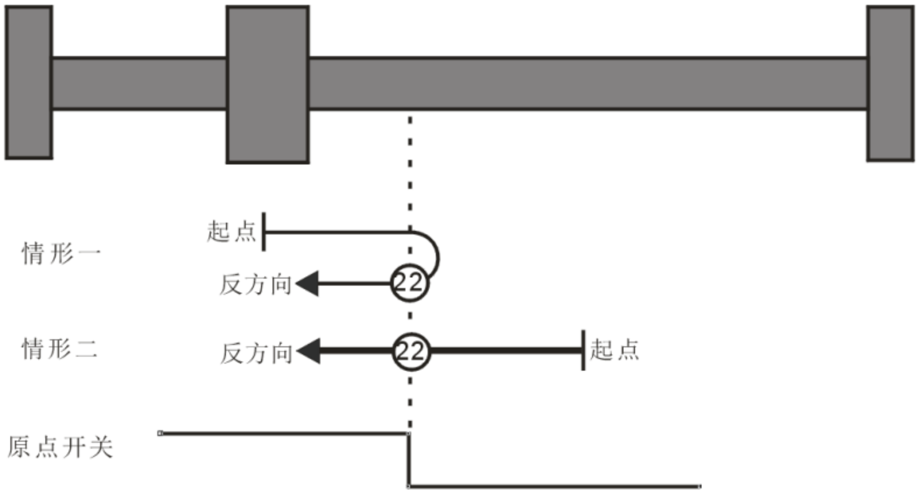



取决于原点开关的原点回零，上图中的表示21原点回零模式21

- **Homing mode switch 22 depending on the origin of the OPR**

Case 1: When performed in a high MC_Home home switch instruction, the start of direct axis 2nd speed forward motion, when the home switch is encountered when changing the direction of motion and at a low speed starts to move the second segment. When faced with the home switch in the high position is the home position.

Case 2: When the instruction execution MC_Home home switch is low, starts moving shaft is the first speed reverse, when the home switch is encountered when a high position is the home position.



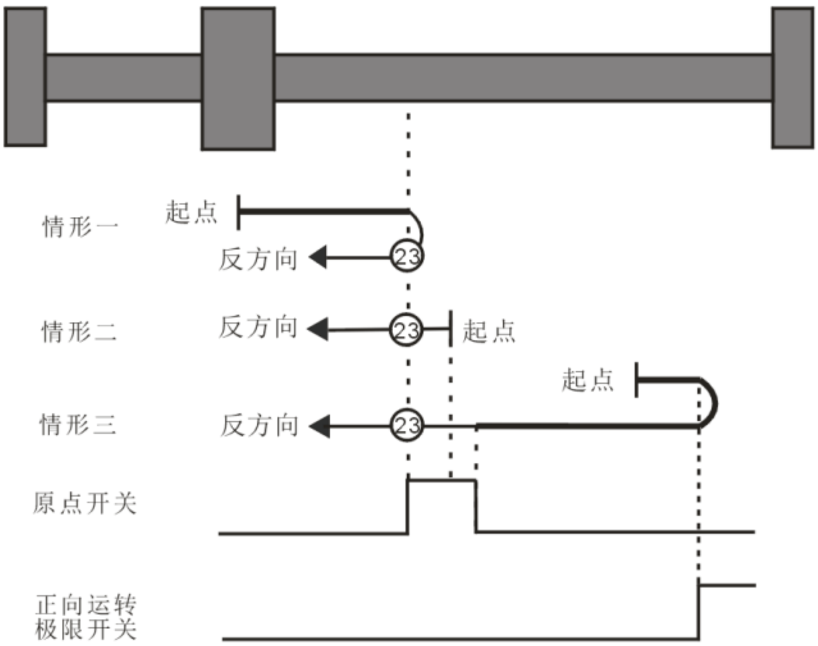

取决于原点开关的原点回零，上图中的表示22原点回零模式22

- **Homing mode Homing 23 depending on the origin switch, Forward limit switch**

Case 1: When performed at a low MC_Home home switch command axis starts forward motion to first speed, when it comes at a high origin switch, changing the direction of motion and starts to move in 2nd speed, the switching state at the origin position of the origin is in the low position.

Case 2: When performed in a high MC_Home home switch command, the shaft directly in 2nd speed reverse movement starts at the home switch is in the low position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, the shaft starts moving forward first speed, when the switch is in the home and encounters a low forward operation at a high limit switch, changing the direction of movement and in the first stage movement start speed, when it comes at a high origin switch, starts to move in 2nd speed, the home switch is in the low position is the home position.



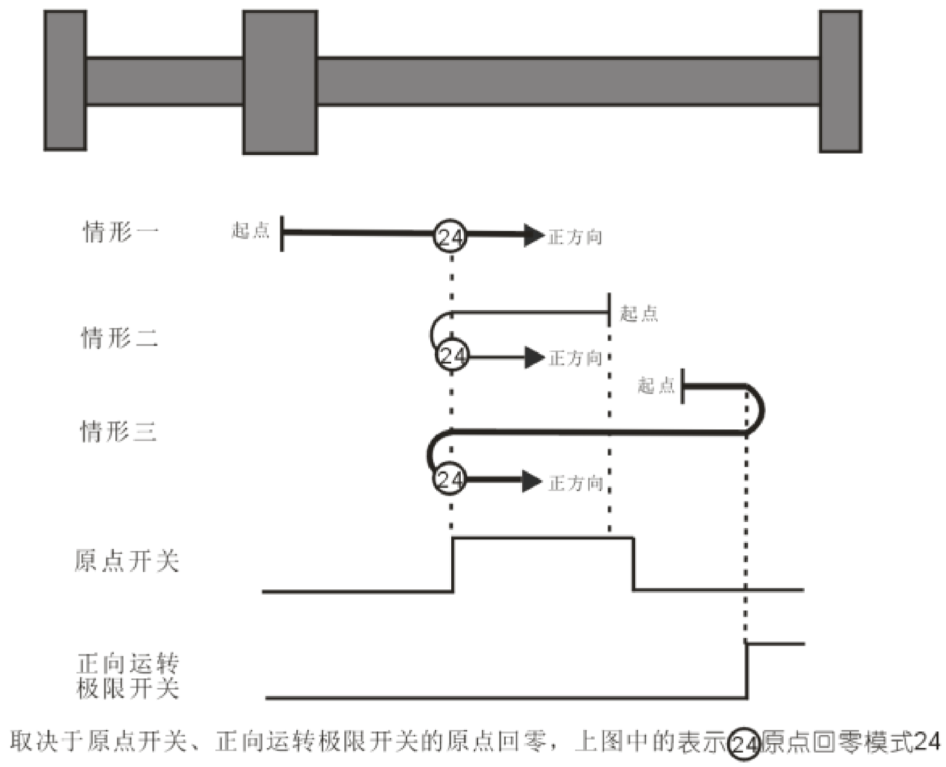

取决于原点开关、正向运转极限开关的原点回零，上图中的表示23原点回零模式23

- **Homing mode Homing 24 depending on the origin switch, Forward limit switch**

Case 1: When performed at a low MC_Home home switch command, the shaft starts first speed forward motion, when faced with the origin position is the home position switch is high.

Case 2: When performed in a high MC_Home home switch command, the shaft directly in 2nd speed reverse movement begins, the home switch changes encountered when the movement direction and are low in 2nd speed starts to move. When faced with the origin switch again at a high position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, the shaft starts moving forward first speed, when the switch is in the home and encounters a low forward operation at a high limit switch, changing the direction of movement and in the first stage movement start speed, when it comes to the home switch high, first speed is still moving, when the home switch is low, the direction of movement and at first speed change starts to move, the high position of the origin is found in the home switch position.

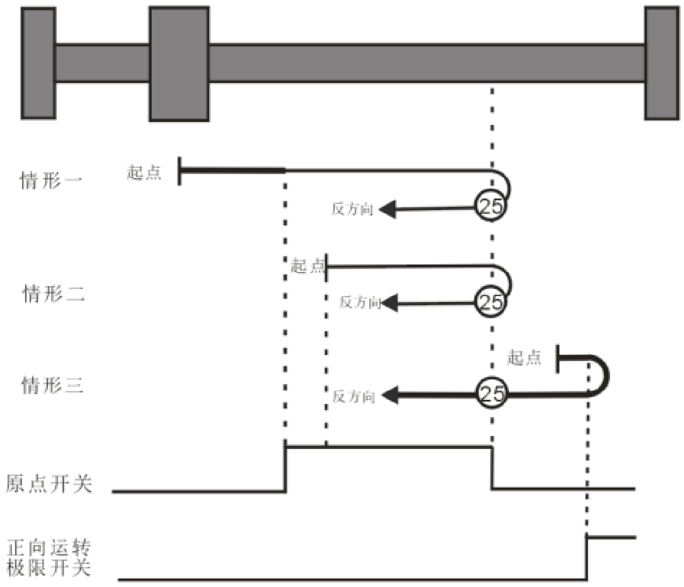



- **Homing mode Homing 25 depending on the origin switch, Forward limit switch**

Case 1: When performed at a low MC_Home home switch command axis starts forward motion to first speed, when it comes at a high origin switch, start 2nd speed forward motion, when the home switch is encountered when low, changing the direction of motion and starts to move to 2nd speed, when it comes to the home switch in the high position is the home position.

Case 2: When the instruction execution MC_Home home switch is high, axis starts moving forward 2nd speed, when it comes to the home switch is low, and changing the direction of movement in 2nd speed starts to move, when encountering origin when the switch is in the upper position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, the shaft starts moving forward first speed, when the switch is in the home and encounters a low forward operation at a high limit switch, changing the direction of movement and in the first stage movement start speed, when it comes to the home switch in the high position is the home position.



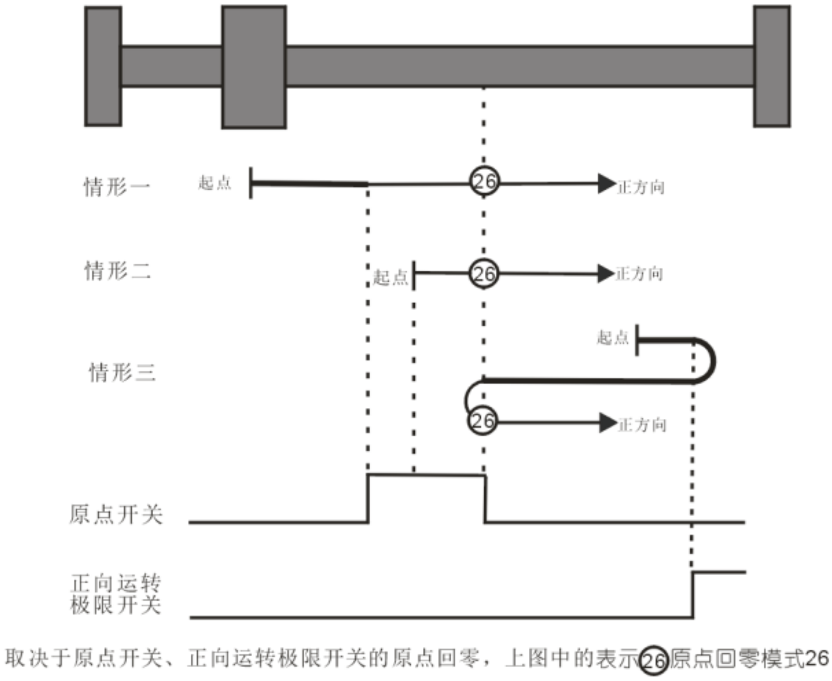

取决于原点开关、正向运转极限开关的原点回零，上图中的表示25原点回零模式25

- **Homing mode Homing 26 depending on the origin switch, Forward limit switch**

Case 1: When the instruction execution MC_Home home switch is low, the shaft starts moving forward first speed, when it comes to the home switch is high, starts to move to 2nd speed when it encounters the home switch in the low the position is the home position.

Case 2: When the home switch performed at a high MC_Home command axis starts moving forward 2nd speed, when faced with the origin position switch is in the low position of the origin.

Case 3: When the instruction execution MC_Home home switch is low, the shaft starts moving forward first speed, when the switch is in the home and encounters a low forward operation at a high limit switch, changing the direction of movement and in the first stage movement start speed, when it comes to the home switch is high, and once again changing the direction of movement in 2nd speed starts to move, when the home switch in the low position is the home position.

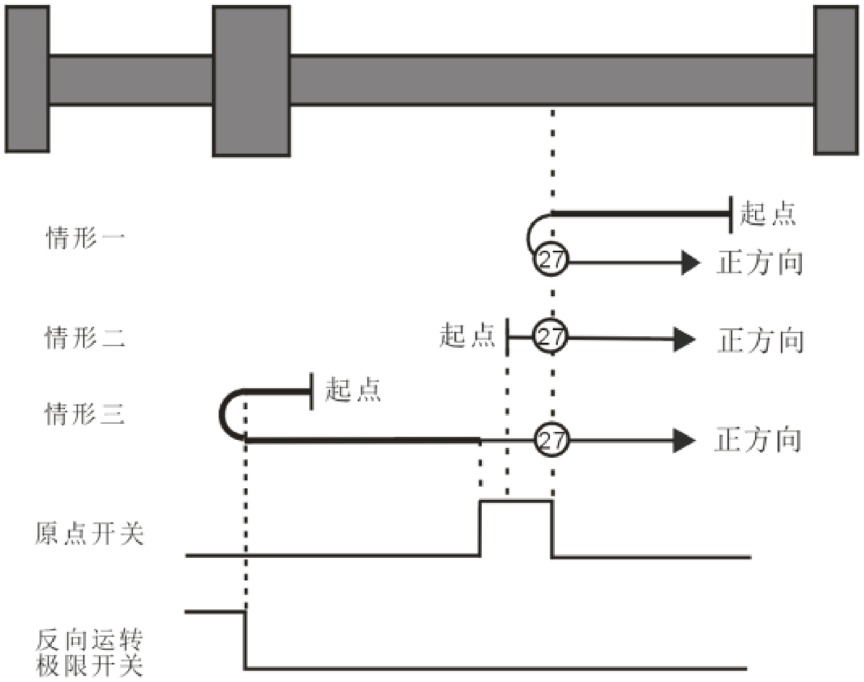



- **Homing mode switch 27 depending on the origin and reverse operation of the limit switch Homing**

Case 1: When the instruction execution MC_Home home switch is low, the shaft speed begins to reverse movement of the first segment, when faced with the home switch is high, the direction of motion and changes in motion in 2nd speed, the home switch position of the origin is in the low position.

Case 2: When performed in a high MC_Home home switch instruction, the start of direct axis 2nd speed forward motion, the home switch is in the low position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, starts moving shaft is the first speed reverse, when the home switch is in the low and reverse operation encountered when the limit switch is high, and to change the direction of movement of the first section movement start speed, when it comes at a high origin switch, starts to move in 2nd speed, the home switch is in the low position is the home position.



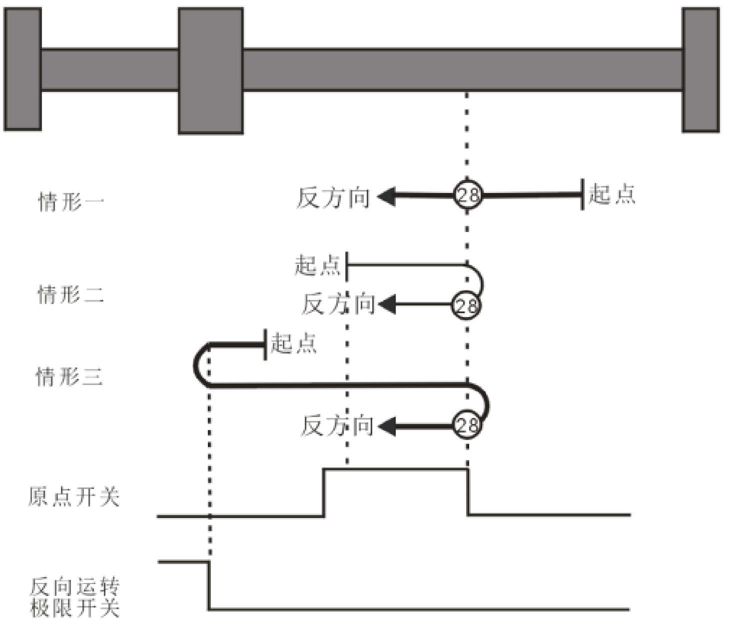

取决于原点开关、反向运转极限开关的原点回零，上图中的表示27原点回零模式27

- **Homing mode switch 28 depending on the origin and reverse operation of the limit switch Homing**

Case 1: When the instruction execution MC_Home home switch is low, the shaft starts to first speed reverse movement, when the switch is in the home position encounters the origin position is high.

Case 2: When performed in a high MC_Home home switch instruction, the start of direct axis 2nd speed forward motion, to change the direction of movement in the home switch in the low speed and at the second section starts to move, when the home switch is high position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, starts moving shaft is the first speed reverse, when the home switch is in the low and reverse operation encountered when the limit switch is high, and to change the direction of movement of the first section movement start speed, when it comes to the home switch high, first speed is still moving, when the home switch is low, the direction of movement to change the first speed and starts to move, the home switch is in the high position is the home position.



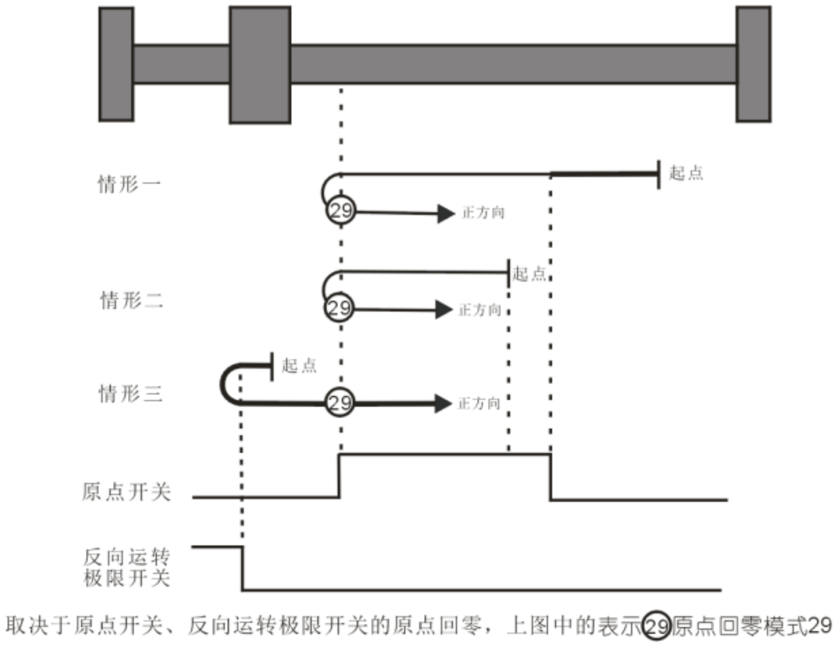

取决于原点开关、反向运转极限开关的原点回零，上图中的表示28原点回零模式28

- **Homing mode switch 29 depending on the origin and reverse operation of the limit switch Homing**

Case 1: When the instruction execution MC_Home the home switch in the low to first speed reverse movement, when the home switch is high encounter, 2nd speed starts to move, when the shaft begin to encounter the home switch in the low changing the direction of movement and runs in 2nd speed, when the switch is again encountered in the home position when the home position is high.

Case 2: When the instruction execution MC_Home home switch is high, the shaft speed begins to reverse movement of the second segment, to change the direction of the origin when the switch is in a low encounter and 2nd speed starts to move, when the home switch is encountered high position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, starts moving shaft is the first speed reverse, when the home switch is in the low and reverse operation encountered when the limit switch is high, and to change the direction of movement of the first section movement start speed, encountered when an origin position switch is in the home position is high.

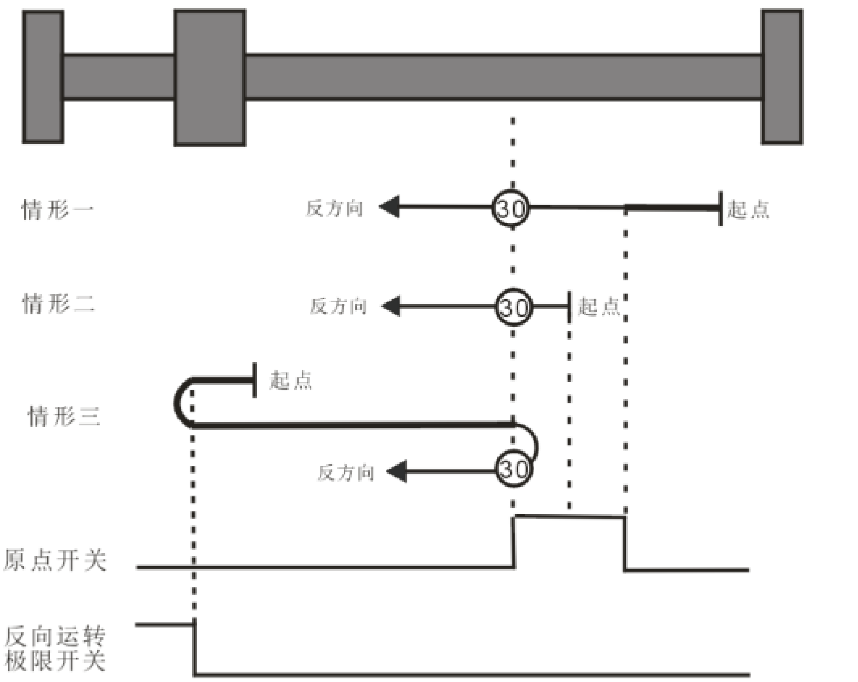



- **Homing mode switch 30 depending on the origin and reverse operation of the limit switch Homing**

Case 1: When the instruction execution MC_Home the home switch in the low to first speed reverse movement, when the home switch is high encounter, 2nd speed starts to move, when the shaft begin to encounter the home switch in the low the position is the home position.

Case 2: When the instruction execution MC_Home home switch is high, the shaft speed begins to reverse movement of the second segment, when faced with the home switch in the low position is the home position.

Case 3: When the instruction execution MC_Home home switch is low, starts moving shaft is the first speed reverse, when the home switch is in the low and reverse operation encountered when the limit switch is high, and to change the direction of movement of the first section movement start speed, when it comes to the home switch is high, and the direction of movement changes again begins to move to 2nd speed when it encounters the home switch in the low position is the home position.




取决于原点开关、反向运转极限开关的原点回零，上图中的表示30原点回零模式30

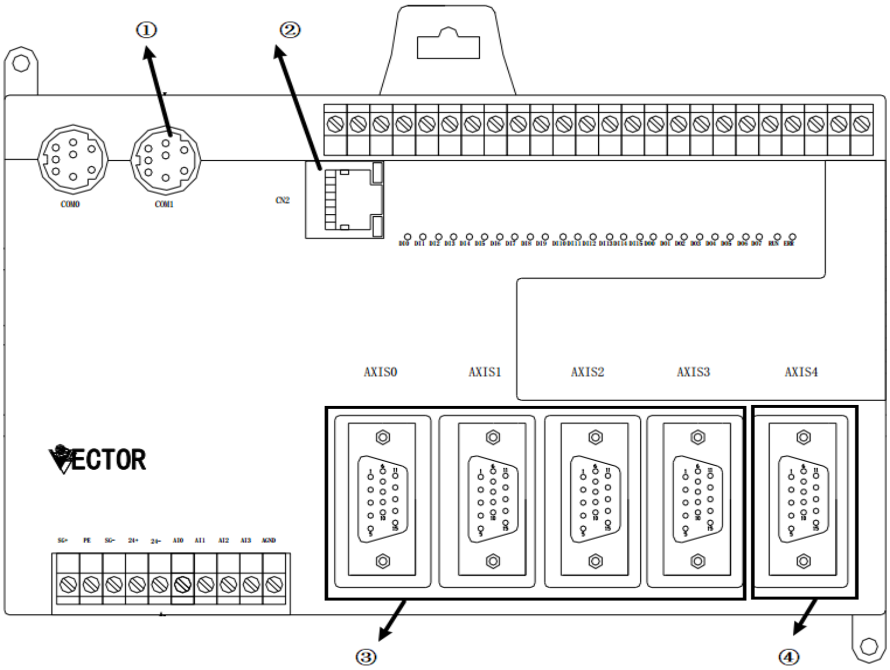
- **The current position of the origin return mode, the shaft 35 is considered a position OPR**

In mode 35, the instruction execution MC_Home, the shaft is not moving, the current position of the axis is considered to be the position of the OPR.

# Appendix IV CANopen Instructions

## 1. CANopen communication connection

### 1.1 Description Motion Controller Connection Ports




As the picture shows:

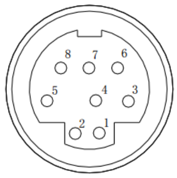
①CANopen communication port, through this interface the data exchange with the slave node, and a transceiver transmitting a synchronization signal sync packet;

② 100M Ethernet port, through this interface to upload and download programs online monitoring;

③AXIS0 ~ AXIS3 invalid type for CANopen;

④AXIS4 spindle interface axis number is 16, only instructions for the spindle to make a multi-axis (or the encoder connected to the pulse generator), it is noted that the same analog with other interface functions.

## 1.2 CANopen communication port pin definitions

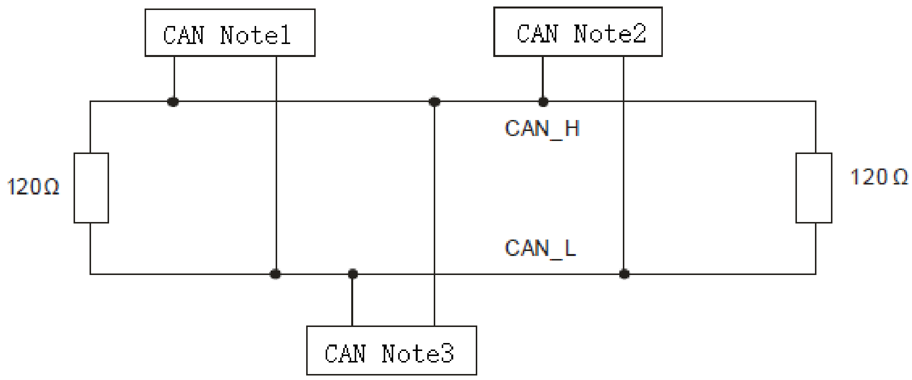
<table>
<tr><td rowspan="9">RS-232<br>(COM1)</td><td rowspan="9"></td><td>stitch</td><td>definition</td></tr>
<tr><td>1</td><td>NC</td></tr>
<tr><td>2</td><td>NC</td></tr>
<tr><td>3</td><td>NC</td></tr>
<tr><td>4</td><td>NC</td></tr>
<tr><td>5</td><td>NC</td></tr>
<tr><td>6</td><td>CANL</td></tr>
<tr><td>7</td><td>CANH</td></tr>
<tr><td>8</td><td>GND</td></tr>
</table>

CANL signal corresponding to the negative (-); CANH signal corresponding to the positive (+); GND signal, the signal needs to be connected in common with the device, please note that the boss pins and the bonding wire direction.

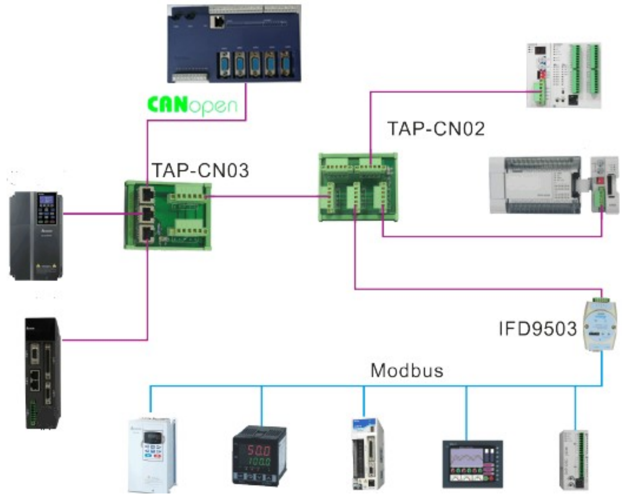
## 1.3 CANopen communication port LAN

CANopen bus terminal and the network topology:

To enhance the stability CANopen communication, two terminals CANopen bus network for an access terminal 120 ohm resistor. The following figure shows a schematic view of the basic CANopen network topology.




CANopen bus network topologies


CANopen
TAP-CN03
TAP-CN02
IFD9503
Modbus

## 1.4 CANopen communication port communication speed and communication distance

Supported CANopen communication speed: 20K, 50K, 125K, 250K, 500K, the maximum transmission 1Mbps, the communication rate of each frequency band with a communication distance, the distance corresponding to FIG.

| Transmission speed (bits per second) | 20K | 50K | 125K | 250K | 500K | 1M |
|---|---|---|---|---|---|---|
| Maximum communication distance (m) | 2500 | 1000 | 500 | 250 | 100 | 25 |

# 2. CANopen protocol basics

## 2.1 Network management (NMT)

Support NMT (Network Management Object: Network Management Objects) Master services, including resetting the network, stop, pre-operation, start and so on.

Support NMT error control, NMT error control station for monitoring whether from dropped. NMT Error Control Heartbeat and NodeGuarding into two types, native support Heartbeat.

## 2.2 Service data (SDO)

Support the use of the ladder in the PLC ladder in non-real time data read from the service station, reference should be read-write area defined by the manufacturer.

## 2.3 Process data (PDO)

Support PDO (Process Data Object: Process Data Object) services:

RxPDO maximum support 200, the amount of data to support the maximum 1000 bytes

TxPDO maximum support 200, the amount of data to support the maximum 1000 bytes

Each configurable TxPDO up to four and four slaves RxPDO

PDO transmission types: support event-triggered, time-triggered, periodic synchronization, synchronous aperiodic

PDO mapping: PDO mapping may each be a maximum of 32 bytes

Support for mapping data type:

| storage space | type of data |
|---|---|
| 1bit | BOOL |
| 8bit | SINT USINT BYTE |
| 16bit | INT UINT WORD |
| 32bit | DINT UDINT REAL DWORD |
| 64bit | LINT ULINT LREAL LWORD |

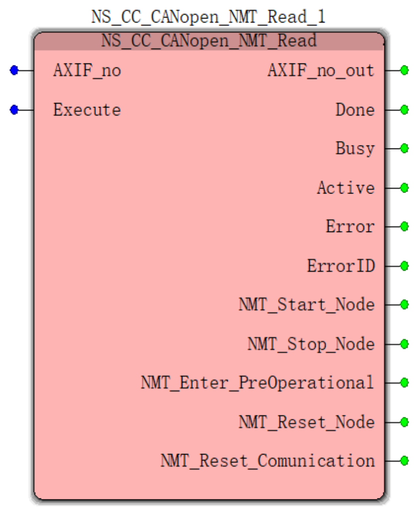
Please refer to the standard CANopen DS402 protocol DS301v4.02 and on motor-driven sub-protocol.

# 3. Software Features

## 3.1 Bus Initialization Configuration Module

### 3.1.1. NS_CC_CANopen_NMT_Read(Network status read instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | Network state This instruction is used to read the current network device is located | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction, the master reading function only supports (18) of the network state | WORD | 18 | Excute to TRUE |
| Excute (execute bit) | When Excute is True, the instruction is executed. | BOOL | TRUE or FALSE (FALSE) | Excute to TRUE |

- **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| AXIF_no_out (node number output) | This parameter is the output node number of instructions executed | WORD | 18 |
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| Error (error output) | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| ErrorID (error code) | Instruction execution error code error | UDINT | - |
| NMT_Start_Node (network node starts) | This parameter indicates the status of the network node is a start-up state is TRUE | BOOL | TRUE or FALSE |
| NMT_Stop_Node (network node stops) | This parameter indicates the status of the network node in a stopped state is TRUE | BOOL | TRUE or FALSE |
| NMT_Enter_Preoperational (pre network entry mode of operation) | This parameter indicates the status of the network node is TRUE when the pre-operation state | BOOL | TRUE or FALSE |
| NMT_Reset_Node (network node reset) | This parameter indicates the network node is TRUE state to the reset state | BOOL | TRUE or FALSE |
| NMT_Reset_Communication (communication reset) | This parameter indicates the status of the communication network when the reset state is TRUE | BOOL | TRUE or FALSE |

- **Function Description**

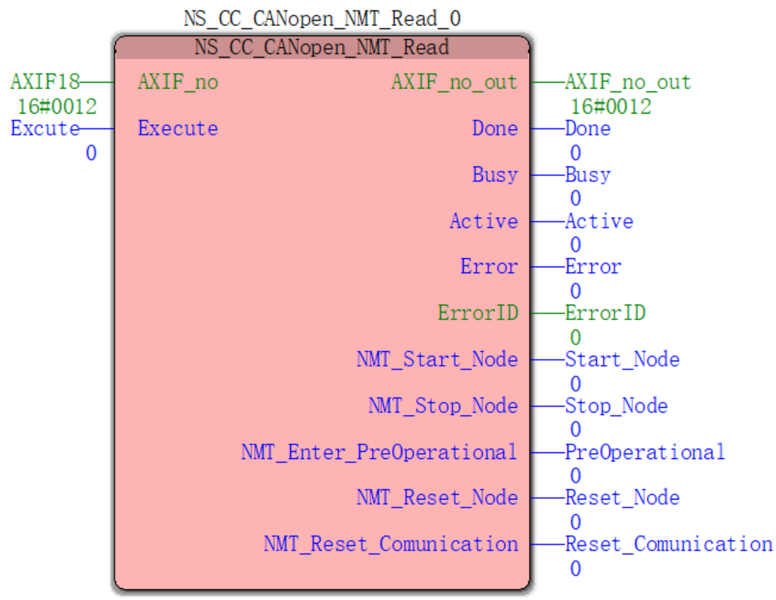
A given number axis (including the Master station number), the trigger module can read the current status of the network in which the network master device, the read module will be reflected on the output terminal when the corresponding state successfully read, the output bits Done becomes Ture, the current status of the corresponding output BOOL variable becomes TRUE.

**Program Example**

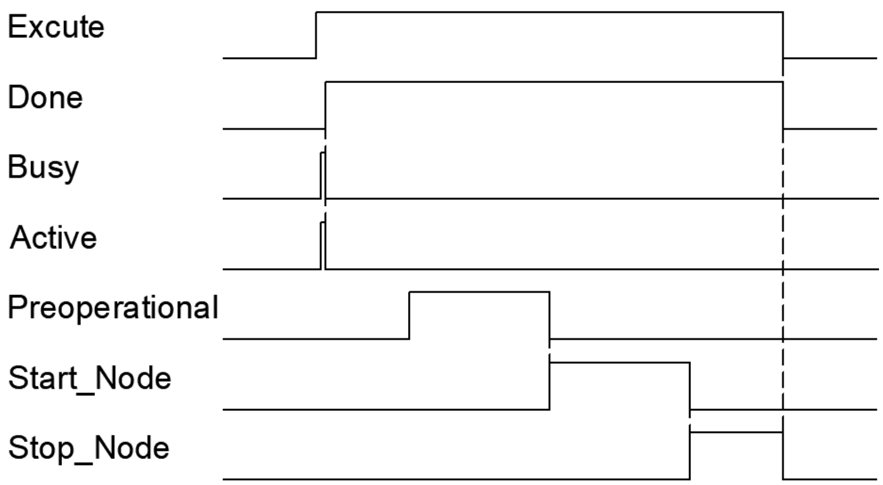
In the example below when NS_CC_CANopen_NMT_Read instruction execution alone.

**3、Variables and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CANopen_NMT_Read_0 | NS_CC_CANopen_NMT_Read | |
| AXIF18 | USINT | 18 |
| Excute | BOOL | FALSE |
| AXIF_no_out | WORD | 1 |
| Done | BOOL | 0 |
| Busy | BOOL | |
| Active | BOOL | |
| Error | BOOL | |
| ErrorID | USINT | |
| Start_Node | BOOL | |
| Stop_Node | BOOL | |
| PreOperational | BOOL | |
| Reset_Node | BOOL | |
| Reset_Comunication | BOOL | |

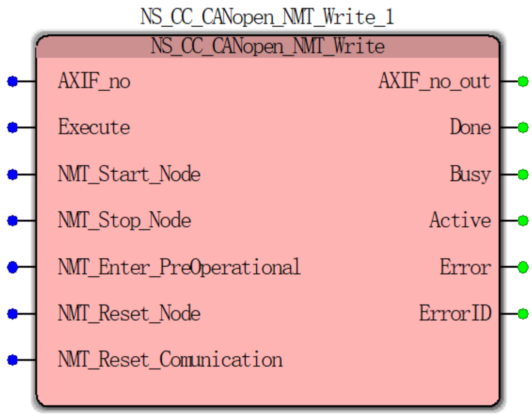



**4、Timing diagram**


Excute
Done
Busy
Active
Preoperational
Start_Node
Stop_Node

### 3.1.2. NS_CC_CANopen_NMT_Write(Network state write command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to write the current status of network devices in which the respective network node in the network | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16, 18 | Excute to TRUE |
| Excute (execute bit) | When Excute is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | |
| NMT_Start_ Node (start node) | This parameter is TRUE, start the network node | BOOL | TRUE or FALSE (FALSE) | |
| NMT_Stop_ Node (stop node) | This parameter is TRUE, the network node is stopped | BOOL | TRUE or FALSE (FALSE) | |
| NMT_Enter_ Preoperational (pre | This parameter is | BOOL | TRUE or FALSE (FALSE) | |

| | | | | |
|---|---|---|---|---|
| network entry mode of operation) | TRUE, into the pre-operational state of the network | | | |
| NMT_Reset_ Node (reset node) | When this parameter is TRUE, the reset network node | BOOL | TRUE or FALSE (FALSE) | |
| NMT_Reset_ Communication (reset communication) | When this parameter is TRUE, the reset network traffic | BOOL | TRUE or FALSE (FALSE) | |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| AXIF_no_out (node number output) | This parameter is the output node number of instructions executed | WORD | 1 to 16, 18 |
| Done | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| Error | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| ErrorID | Instruction execution error code error | WORD | - |

➢ **Function Description**

1, a given number of nodes (including the Master station number), the first trigger node status, and then Excute trigger (the rising edge of input), can be written in the network status of the network node corresponding to the device, after writing is completed, the output becomes the Done Ture ;

2, NMT_Start_Node (start node), NMT_Stop_Node (stop node), NMT_Enter_Preoperational (pre network entry mode of operation), NMT_Reset_Node (reset node), NMT_Reset_Communication (reset communication), any two or more inputs can not trigger;

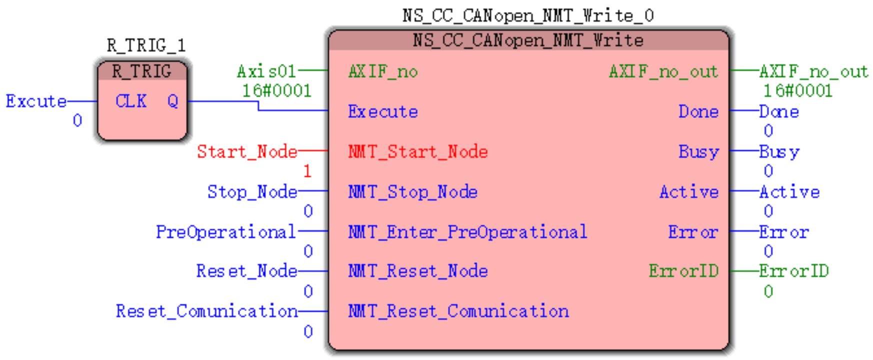
3, the instruction must trigger the triggering edge module, it is not always Excute TURE; Done when the output signal indicating write completion status of the selected network.

**Program Example**

In the example below when NS_CC_CANopen_NMT_Read instruction execution alone.

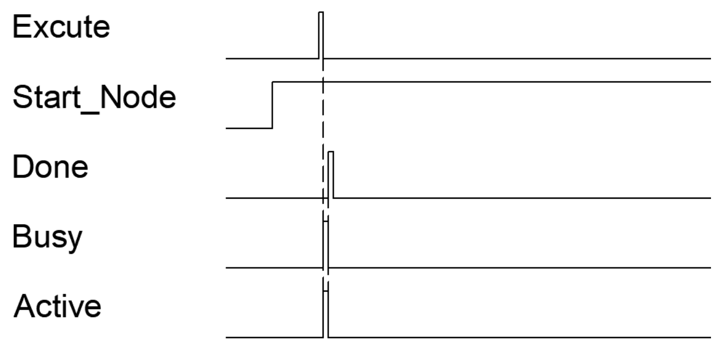
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CANopen_NMT_Read_0 | NS_CC_CANopen_NMT_Read | |
| AXIF18 | USINT | 1 |
| Excute | BOOL | FALSE |
| Start_Node | BOOL | FALSE |
| Stop_Node | BOOL | FALSE |
| PreOperational | BOOL | FALSE |
| Reset_Node | BOOL | FALSE |
| Reset_Comunication | BOOL | FALSE |
| AXIF_no_out | WORD | |
| Done | BOOL | |
| Busy | BOOL | |
| Active | BOOL | |
| Error | BOOL | |
| ErrorID | USINT | |



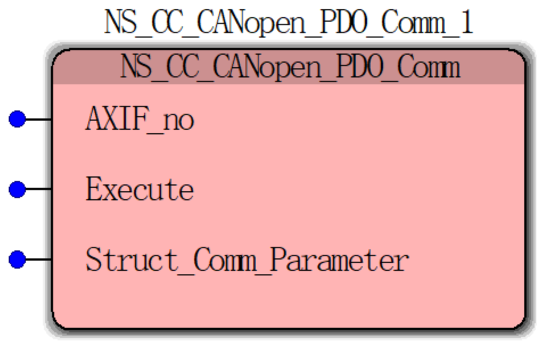

**2, a timing diagram**

As shown, a given mode of operation, such as Start_Node turned Excute trigger module, the master node of the network 1 will be a start-up operation, the output operation is successful "Done" signal.


Excute
Start_Node
Done
Busy
Active

### 3.1.3 NS_CC_CANopen_PDO_Comm(PDO process data communication configuration parameters)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to configure the communication parameters PDO process data | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16 | Excute to TRUE |
| Excute (execute bit) | When Excute is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | Excute to TRUE |
| Struct_Comm_Parameter (structure parameter) | Structure parameters, see the functional description format | ANY | | Excute to TRUE |

➢ **Function Description**

1, the instruction must trigger the triggering edge of the module, it is not always Excute TURE;

2, "Struct_Comm_Parameter" data structure as that defined in accordance with the contents specified content DS301 protocol, defined after completion trigger module, i.e. a communication parameter from the subject into the corresponding dictionary station, the communication parameters corresponding to the object dictionary is "0X1400H～0X14FF "or"

0X1800H ~ 0X18FF "," CommPara "structure as the data structure, protocol defines fixed format do not make changes.

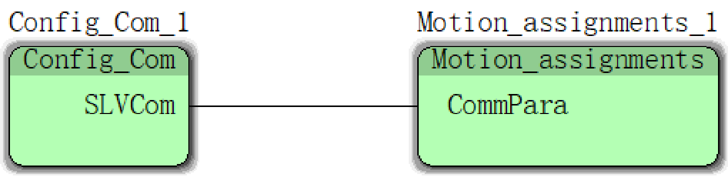
```
TYPE
  AIXComm:
    STRUCT
        Index:UINT ;
        Num_Of_SubIndex:BYTE ;
        CobID:UDINT ;
        Transmission_Type:BYTE ;
        Inhibit_Time:UINT ;
        Compatibility_Entry:BYTE ;
        Event_Timer:UINT ;
    END_STRUCT;
END_TYPE
```

**Program Example**

NS_CC_CANopen_PDO_Comm as shown in the example of instruction execution when alone.

**1, variables, and procedures**

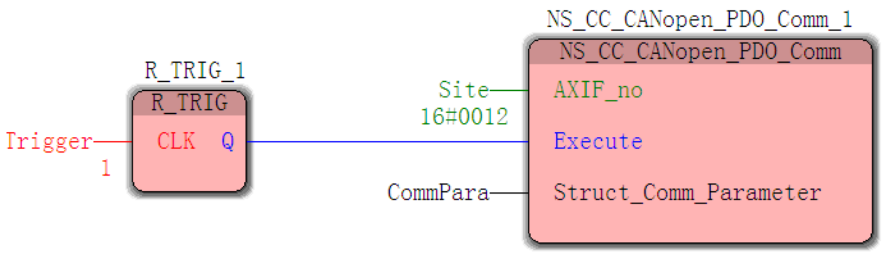
| variable name | type of data | The initial value |
|---|---|---|
| Config_Com_1 | Config_Com | |
| Motion_assignments_1 | Motion_assignments | |
| R_TRIG_1 | R_TRIG | |
| NS_CC_CANopen_PDO_Comm_1 | NS_CC_CANopen_PDO_Comm | |
| SLVCom | VAR_OUTPUT | |
| CommPara | VAR_INPUT | |




Config_Com_1 configuration is as follows:

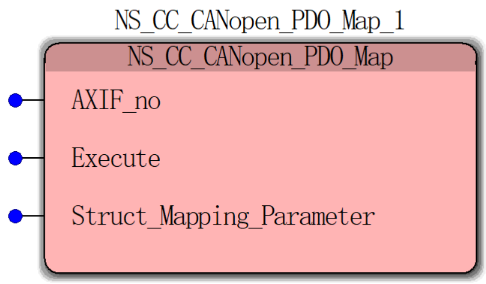
```
  TRUE |IF Trigger = TRUE THEN (*配置主站对从站的RPDO通信参数*)
  6144 |SLVCom.Index := UINT#16#1800;
 16#02 |SLVCom.Num_Of_SubIndex := INT_TO_BYTE (2);
   385 |SLVCom.CobID := UDINT#16#180 + WORD_TO_UDINT(AXIF_no+WORD#1);
 16#01 |SLVCom.Transmission_Type := INT_TO_BYTE (1);
     0 |SLVCom.Inhibit_Time := INT_TO_UINT (0);
 16#00 |SLVCom.Compatibility_Entry := INT_TO_BYTE (0);
     0 |SLVCom.Event_Timer :=INT_TO_UINT (0);
```

Motion_ assignment_1 configured as follows:


NS_CC_CANopen_PDO_Comm_1
NS_CC_CANopen_PDO_Comm
R_TRIG_1
R_TRIG
Site
16#0012
AXIF_no
Trigger
1
CLK
Q
Execute
CommPara
Struct_Comm_Parameter

### 3.1.4 NS_CC_CANopen_PDO_Map(PDO process data configuration parameter map)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to process data PDO configuration parameter map | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16 | Excute to TRUE |
| Excute (execute bit) | When Excute is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | |
| Struct_Mapping_Parameter (structure parameter) | Structure parameters, see the functional description format | ANY | | Excute to TRUE |

➢ **Function Description**

1, the instruction must trigger the triggering edge of the module, it is not always Excute TURE;

2, "Struct_Mapping_Parameter" data structure as that defined in accordance with the contents specified content DS301 protocol module departure Once defined, i.e. a communication parameter from the subject into the corresponding dictionary station, the communication parameters corresponding to the object dictionary is "0X1600H ~ 0X16FF "or" 0X1A00H ~ 0X1AFF "," MapPara "structure as the data structure, protocol defines fixed format do not make changes.

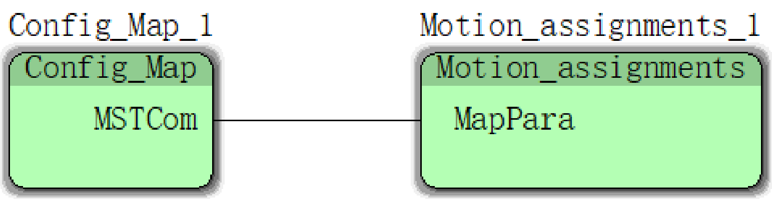
```
TYPE
    AIXMap:
    STRUCT
        Index:UINT;
        Num_Of_SubIndex:BYTE;
        SubIndex_Mapping_1:UDINT;
        SubIndex_Mapping_2:UDINT;
        SubIndex_Mapping_3:UDINT;
        SubIndex_Mapping_4:UDINT;
        SubIndex_Mapping_5:UDINT;
        SubIndex_Mapping_6:UDINT;
        SubIndex_Mapping_7:UDINT;
        SubIndex_Mapping_8:UDINT;
    END_STRUCT;
END_TYPE
```

**Program Example**

In the example below when NS_CC_CANopen_PDO_Map instruction execution alone.

**1, variables, and procedures**

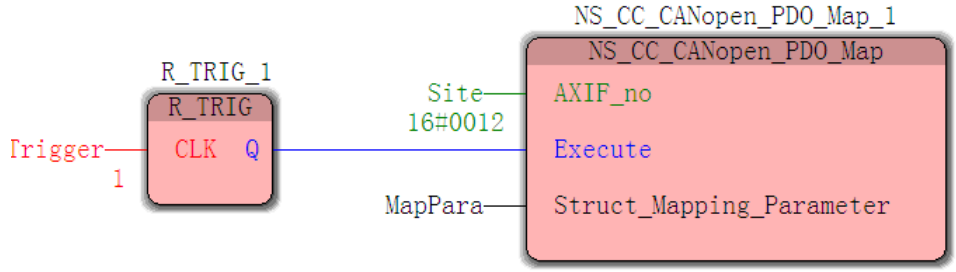
| variable name | type of data | The initial value |
|---|---|---|
| Config_Map_1 | Config_Map | |
| Motion_assignments_1 | Motion_assignments | |
| R_TRIG_1 | R_TRIG | |
| NS_CC_CANopen_PDO_Map_1 | NS_CC_CANopen_PDO_Map | |
| MSTCom | VAR_OUTPUT | |
| MapPara | VAR_INPUT | |




Config_Map_1 configuration is as follows:

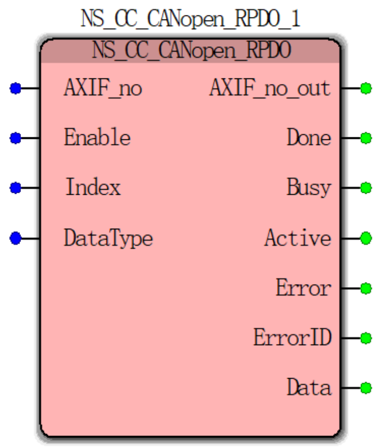
```
TRUE  IF Trigger = TRUE THEN
5120  MSTCom.Index := UINT#16#1400 + WORD_TO_UINT(AXIF_no * WORD#2) ;
16#02 MSTCom.Num_Of_SubIndex := INT_TO_BYTE (2);
385   MSTCom.CobID := UDINT#16#180 + WORD_TO_UDINT(AXIF_no+WORD#1);
16#01 MSTCom.Transmission_Type := INT_TO_BYTE (1);
0     MSTCom.Inhibit_Time := INT_TO_UINT (0);
16#00 MSTCom.Compatibility_Entry := INT_TO_BYTE (0);
0     MSTCom.Event_Timer :=INT_TO_UINT (0);
```

Motion_assignments_1 configuration is as follows:


NS_CC_CANopen_PDO_Map_1
NS_CC_CANopen_PDO_Map
R_TRIG_1
R_TRIG
Site
16#0012
AXIF_no
Trigger
1
CLK
Q
Execute
MapPara
Struct_Mapping_Parameter

## 3.1.5 NS_CC_CANopen_RPDO(PDO data mapping area read command)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to map the data area read command PDO | |




➤ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16 | Enable is TRUE |
| Enable (execute bit) | When Enable is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | |
| Index (Index) | Data mapping area index | WORD | 0000 ~ FFFF | Enable is TRUE |
| DataType | type of data | WORD | 0000 ~ FFFF | Enable is TRUE |

➤ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| AXIF_no_out (node number output) | This parameter is the output node number of instructions executed | WORD | 0 ~ 16 |
| Done (execution | The output parameter to | BOOL | TRUE or |

| | | | |
|---|---|---|---|
| is complete) | TRUE indicates instructions are executed | | FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| Error (error output bit) | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| The ErrorID (error code) | Instruction execution error code error | WORD | 0000 ~ FFFF |
| Data (data content) | Data output content | WORD | 0000 ~ FFFF |

➢ **Function Description**

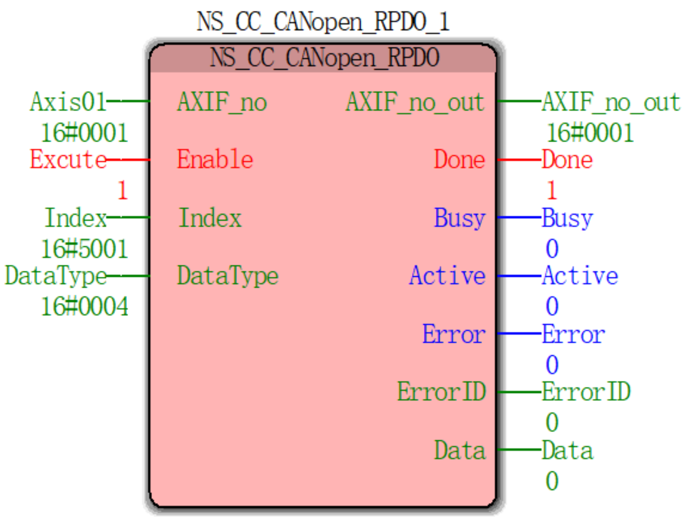
Function module for reading map data content from a certain sub station node TPDO index data mapping stored in the address zone data, when the state of the corresponding successful reading, the output becomes the Done Ture.

**Program Example**

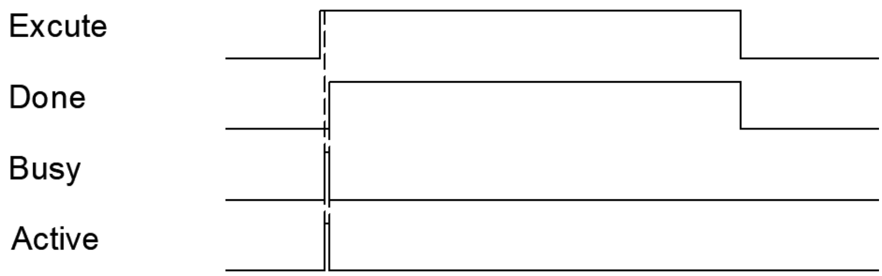
In the example below when NS_CC_CANopen_RPDO instruction execution alone.

**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CANopen_RPDO_1 | NS_CC_CANopen_RPDO | |
| AXIF01 | USINT | 1 |
| Excute | BOOL | FALSE |
| Index | WORD | 16 # 5001 |
| DataType | WORD | 16 # 0004 |
| AXIF_no_out | WORD | |
| Done | BOOL | |
| Busy | BOOL | |
| Active | BOOL | |
| Error | BOOL | |
| ErrorID | USINT | |
| Data | UDINT | |




**2. A timing diagram**



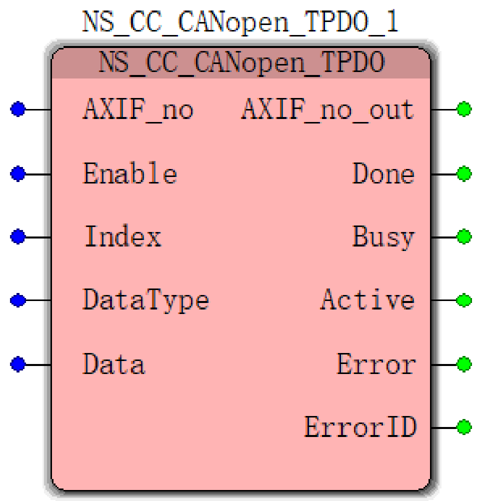

As shown, 0X60FF TPDO mapping data mapping area, the index 0x5001 (master defined), then the "Index = 16 # 5001", Type "DataType" is defined as follows:

```
数据类型表示：
02 signed8
03 signed16
04 signed32
05 unsigned8
06 unsigned16
07 unsigned32
```

Because 0X60FF to DINT type, the "DataType = 16 # 04", the trigger module reads the corresponding profile of the velocity set value.

## 3.1.6 NS_CC_CANopen_TPDO(PDO data mapping area assignment instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to map the data area assigned PDO | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16 | Enable is TRUE |
| Enable (execute bit) | When Enable is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | Enable is TRUE |
| Index (Index) | Data mapping area index | WORD | 0000 ~ FFFF | Enable is TRUE |
| The DataType (data type) | type of data | WORD | 0000 ~ FFFF | Enable is TRUE |
| Data (data content) | Data content | WORD | 0000 ~ FFFF | Enable is TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|

| | | | |
|---|---|---|---|
| AXIF_no_out (node number output) | This parameter is the output node number of instructions executed | WORD | 1 to 16 |
| Done (execution is complete) | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| Error (error output bit) | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| The ErrorID (error code) | Instruction execution error code error | WORD | |

- **Function Description**

1, the function of this module, the address assignment to the data from a certain sub station node RPDO index data mapping stored in the area;

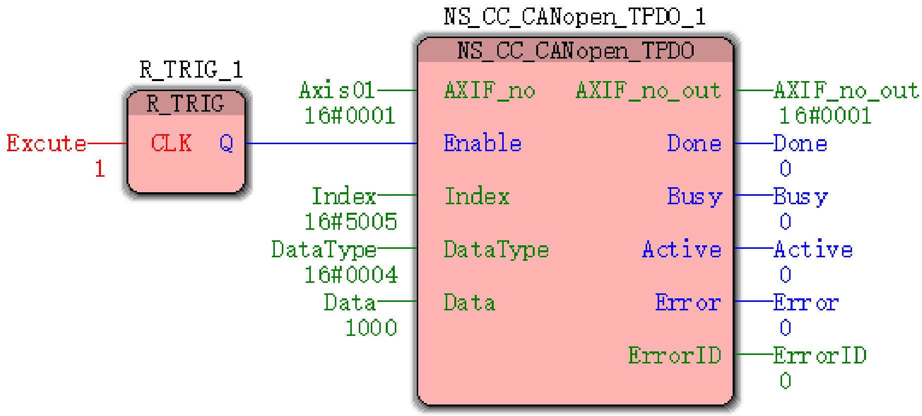
2, the instruction must trigger the triggering edge module, it is not always Excute TURE.

**Program Example**

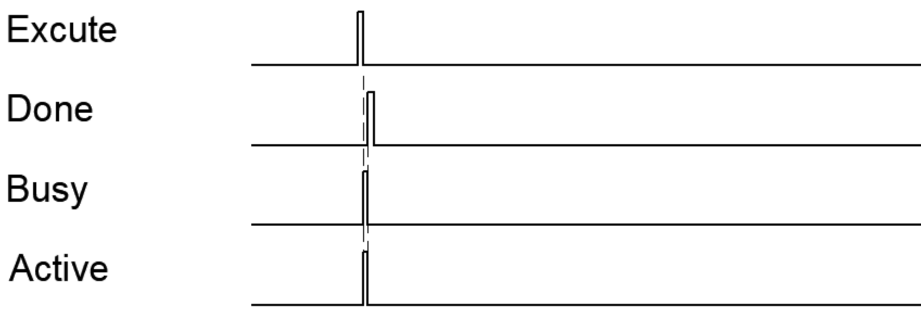
In the example below when NS_CC_CANopen_TPDO instruction execution alone.

**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CANopen_TPDO_1 | NS_CC_CANopen_TPDO | |
| AXIF01 | USINT | 1 |
| Excute | BOOL | FALSE |
| Index | WORD | 16 # 5001 |
| DataType | WORD | 16 # 0004 |
| Data | UDINT | 1000 |
| AXIF_no_out | WORD | |
| Done | BOOL | |
| Busy | BOOL | |
| Active | BOOL | |
| Error | BOOL | |
| ErrorID | USINT | |



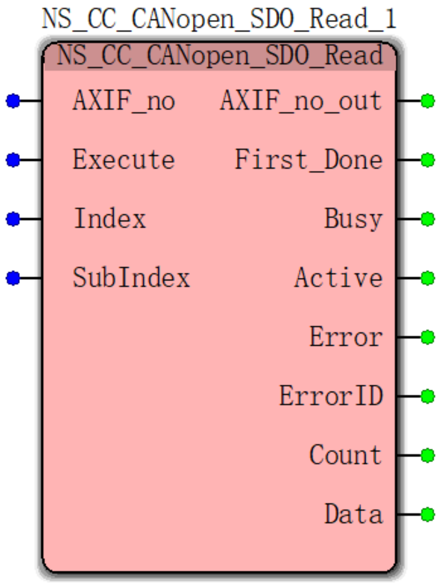

**2. A timing diagram**




As shown, 0X60FF RPDO mapping data mapping area, index 0X5005 (custom master station), then the "Index = 16 # 5005", due to DINT 0X60FF type, the "DataType = 16 # 04", "Data = 1000 "1000 will be assigned to the trigger module node address 0X60FF, it indicates that the current node profile speed setting value is set to 1000.

## 3.1.7 NS_CC_CANopen_SDO_Read(Service data reading instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This instruction is used to read the data service | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16 | Excute to TRUE |
| Excute (execute bit) | When Excute is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | |
| Index (Index) | Inode address | WORD | 0 ~ FFFF | Excute to TRUE |
| SubIndex (sub-index) | Subindex node address | WORD | 0 ~ FFFF | Excute to TRUE |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| AXIF_no_out (node number output) | This parameter is the output node number of | WORD | 0 ~ 16 |

| | | | |
|---|---|---|---|
| | instructions executed | | |
| First_Done (first execution is complete) | The output parameter to TRUE represents the first instruction execution is complete, the trigger again, the parameter is still Ture | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| Error (error output bit) | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| The ErrorID (error code) | Instruction execution error code error | WORD | |
| Count | Byte length | WORD | 0000 ~ FFFF |
| Data | Output Data | UDINT | 0000 ~ FFFF |

➢ **Function Description**

Service data read module, and read by the specified index subindex manner a node address in the contents of the address Youwenyouda, slow reading speed, reading normally takes a period of two sync, applied to non-real time data read operation.

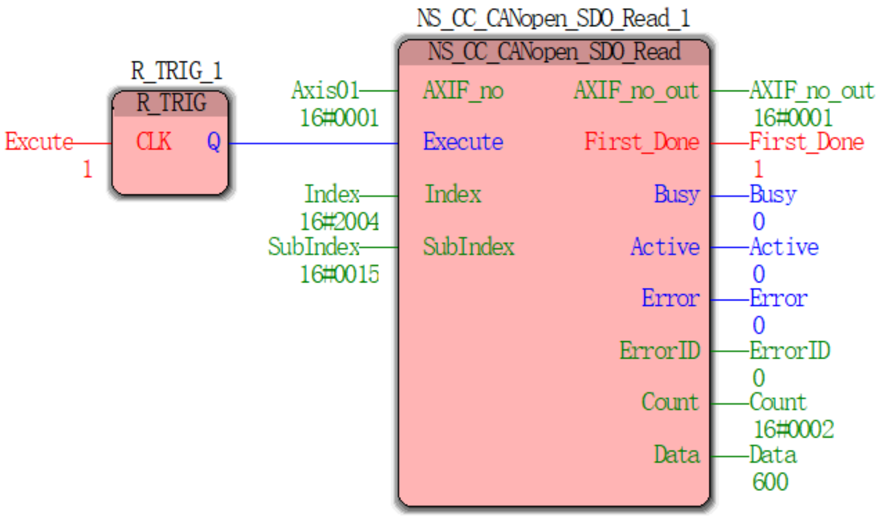
**Program Example**

In the example below when NS_CC_CANopen_SDO_Read instruction execution alone.

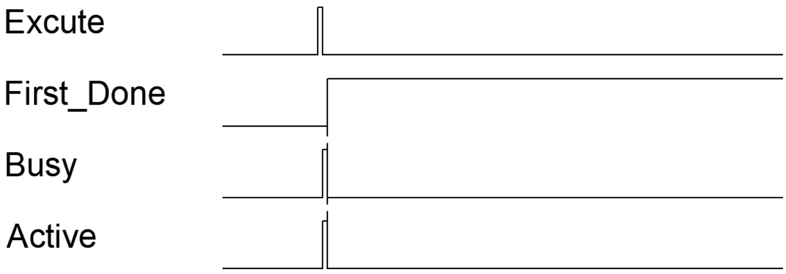
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CANopen_SDO_Read_1 | NS_CC_CANopen_SDO_Read | |
| AXIF01 | USINT | 1 |
| Excute | BOOL | FALSE |
| Index | WORD | 16 # 2004 |
| SubIndex | WORD | 16 # 0015 |
| AXIF_no_out | WORD | |
| Done | BOOL | |
| Busy | BOOL | |
| Active | BOOL | |
| Error | BOOL | |

| | | |
|---|---|---|
| ErrorID | USINT | |
| Count | WORD | |
| Data | UDINT | |



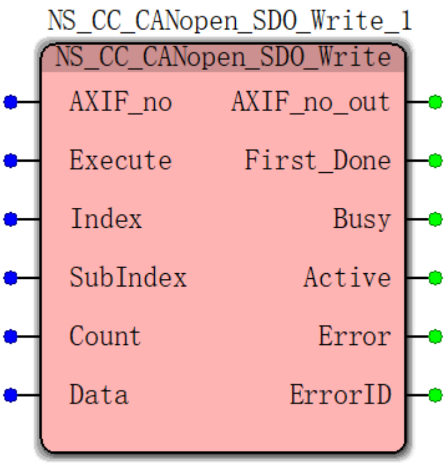

**2. A timing diagram**




As shown, the servo read parameter P04.21 Vector (the current rotation speed, the unit r / min), the corresponding index: 2000 + # 16 # 16 16 # 4 = 2004, corresponding to the sub-index 21 16 # 15, the trigger module No. 1 reads the current speed of the servo node 600r / min.

## 3.1.8 NS_CC_CANopen_SDO_Write(Service Data assignment instruction)

| FB / FC | Explanation | Applicable model |
|---|---|---|
| FB | This command is used to assign data service | |




➢ **Input parameters**

| name | Features | type of data | Range setting (default value) | The timing of the entry into force |
|---|---|---|---|---|
| AXIF_no (node number) | To control node setting instruction | WORD | 1 to 16 | Excute to TRUE |
| Excute (execute bit) | When Excute is True, the instruction is executed | BOOL | TRUE or FALSE (FALSE) | |
| Index (Index) | Inode address | WORD | | |
| SubIndex (sub-index) | Subindex node address | WORD | | |
| Count | Byte length | WORD | | |
| Data | data input | UDINT | | |

➢ **Output parameters**

| name | Features | type of data | Output range |
|---|---|---|---|
| AXIF_no_out (node number output) | This parameter is the output node number of instructions executed | WORD | 0 ~ 16 |

| | | | |
|---|---|---|---|
| Done (execution is complete) | The output parameter to TRUE indicates instructions are executed | BOOL | TRUE or FALSE |
| Busy (execution) | This parameter indicates to TRUE output instruction is executed | BOOL | TRUE or FALSE |
| The Active (control) | When this parameter is TRUE indicates output command under the control shaft | BOOL | TRUE or FALSE |
| Error (error output bit) | This parameter indicates the instruction execution error to TRUE | BOOL | TRUE or FALSE |
| The ErrorID (error code) | Instruction execution error code error | WORD | - |

- **Function Description**

1, the service data assignment module to assign the address specified by the index address and a node index from a direct manner, Youwenyouda, assignment slow speed, reading normally takes a period of two sync, applied to non-real time data write operations.

2, the instruction must trigger the triggering edge module, it is not always Excute TURE;

3, differs from that of the read module, a multi-byte variable "Count" (byte length), and "Data" (variable data), the byte length is defined as follows:

```
数据字节长度表示：
1,2,3,4 COUNT数
```

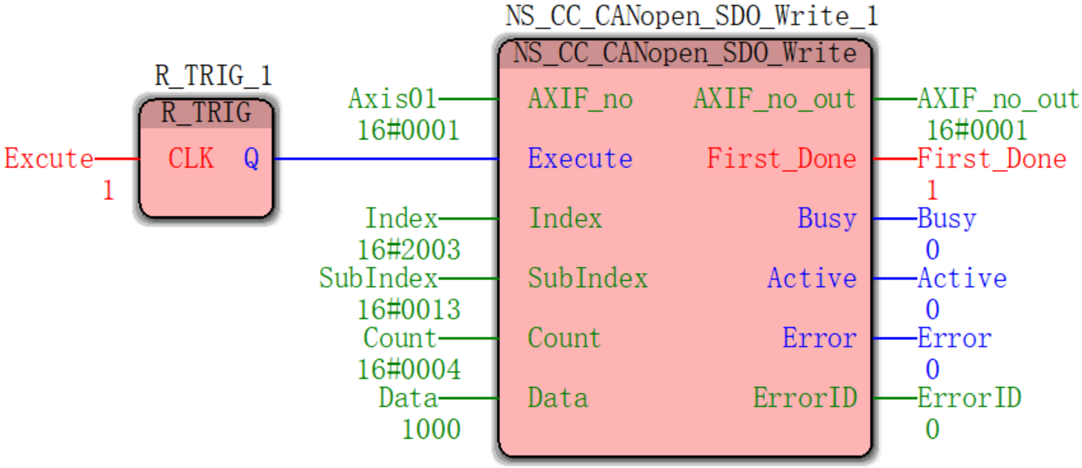
**Program Example**

NS_CC_CANopen_SDO_Write as shown in the example of instruction execution when alone.

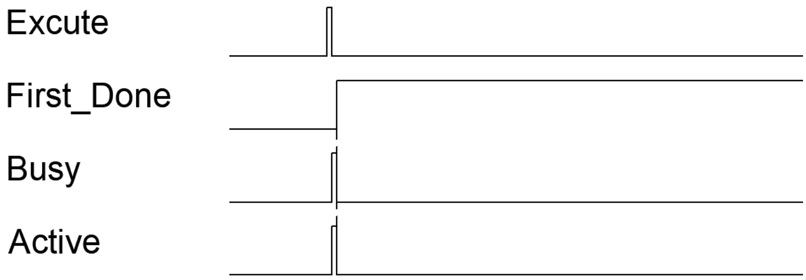
**1, variables, and procedures**

| variable name | type of data | The initial value |
|---|---|---|
| NS_CC_CANopen_SDO_Write_1 | NS_CC_CANopen_SDO_Write | |
| AXIF01 | USINT | 1 |
| Excute | BOOL | FALSE |
| Index | WORD | 16 # 2004 |
| SubIndex | WORD | 16 # 0015 |
| Count | WORD | |
| Data | UDINT | |
| AXIF_no_out | WORD | |
| Done | BOOL | |
| Busy | BOOL | |
| Active | BOOL | |

| | | |
|---|---|---|
| Error | BOOL | |
| ErrorID | USINT | |




**2. A timing diagram**




When the above, the servo parameter assignment Vector P03.19 (position error value is too large), the corresponding index: 2000 + # 16 # 16 16 # 3 = 2004, corresponding to the sub-index 19 16 # 13, a length of 4 bytes , the current No. 1 trigger module assignment node position is too large value.

### 3.2 Motion Control Module

. A module when the servo parameters P08.42 = 0, the default of the servo motion controller bus VEC support for Vector brand:

MC_AXIS_REF
MC_Power
MC_CamIn
MC_CamOut
MC_CombineAxes
MC_GearIn
MC_GearOut
MC_Halt
MC_Home
MC_MoveAbsolute
MC_MoveAdditive
MC_MoveRelative
MC_MoveVelocity
MC_Stop
NS_MC_StopByPos
MC_SpecialMoveAbsolute
NS_MC_RotaryCutIn
NS_MC_SpecialCamIn
NS_MC_SpecialCombineAxes
MC_HaltSuperimposed
MC_MoveSuperimposed
MC_Phasing
NS_MC_Jog
MC_SetOverride
MC_SetPosition
MC_TouchProbe
MC_AbortTrigger
NS_MC_CamReadPoint
NS_MC_CamReadTappetStatus
NS_MC_CamReadTappetValue
NS_MC_CamSet
NS_MC_CamWritePoint
NS_MC_CamWriteTappetValue
MC_ReadActualPosition
MC_ReadActualVelocity
MC_ReadMotionState
MC_ReadStatus
NS_MC_ReadParameter

MC_Reset

39 related to the motion control module, the maximum control shaft 16.

Bus motion controller first edition is to increase the CANopen protocol layers based on the analog version of the launch of an upgraded version of the product, the product uses the motion control function module is consistent with the analog products, the description does not function module then repeat, please refer to the "VA series motion controller programming Manual," a book to learn more about.

b. When the servo parameter P08.42 = 1, A2 series Delta default servo CANopen

By Delta DVP15MC11T motion controller Motion test, VC bus servo suitable for most applications function sets up the controller. Motion port can not be used a total of six functions:

MC_SetPosition (position setting command);

The MC_ReadAxisError (read axis error command);

MC_TouchProbe (position capture command, when capturing the servo position is defined by different terminals DI can not be used);

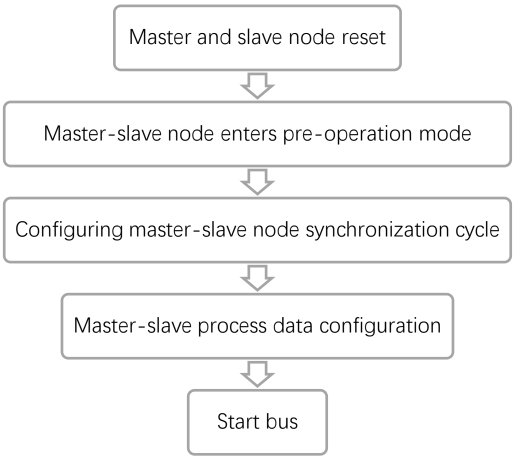
DMC_NC (G instruction code analysis);

DMC_ControlAxisByPos (NC shift instruction);

In addition to the main function NC function is not available, the other out of the listed functions tested, can be used.

# 4. Example Configuration

Process describes the software configuration of the bus when the motion controller using the Vector CANopen servo, configuration process is as follows:




## 4.1 Motion Control Shaft Arranged

The main job is to process the configuration data PDO, network synchronization period, following the second package module through the gradual movement of the shaft describes the configuration process, the process of configuring a servo, the node number is 1.

### 4.1.1 Communication Configuration

According to the hardware topology to build a good network, start building communication:

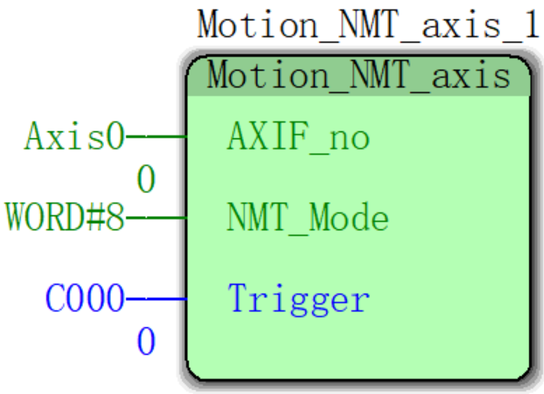
1, the slave set, the servo P08.41 = 1 (node number), P08.40 = 800 (baud rate);

2, the master station is provided, the motion controller in the master station 18 is the default number, register address:% MB3.4012, communication baud rate register address:% MW3.4013, set up as follows

| variable name | type of data | The initial value | address |
|---|---|---|---|
| BaudRate | WORD | 800 | |
| Com_ BaudRate | WORD | | % MW3.4013 |
| MainSite | Byte | 18 | |
| Com_ MainSite | Byte | | % MB3.4012 |

```
BaudRate———Com_BaudRate     MainSite———Com_MainSite
   800       800                  18       18
```

## 4.1.2 Reset the Master-slave Node



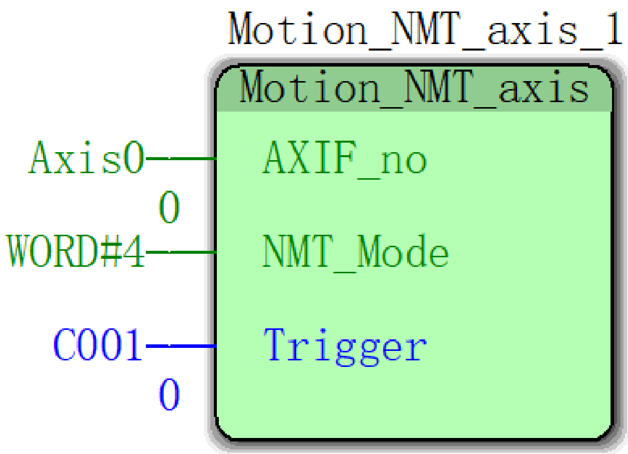

➢ **Input parameters**

| variable name | Features | type of data | The initial value |
|---|---|---|---|
| Axis0<br>(axis number) | Node number = axis number +1 | USINT | 0 |
| C000<br>(execute bit) | When the Execute FALSE to TRUE, the instruction is executed | BOOL | FALSE |

1, NMT_Mode (network model) is customizable, template is defined as: 1 (Start Network), 2 (stop the network), 4 (the network into a pre-operation), 8 (reset node), 16 (reset communication), this selected at NMT_Mode = 8, reset the network;

2, the internal module functions NS_CC_CANopen_NMT_Write two modules, one master station 18 is reset, one pair of the station 1 is reset from the node number, the condition "C000" trigger time delay waiting for approximately 1s and then go to the next step. (See detailed configuration template "Vector CANopen Configuration")

## 4.1.3 Master-slave Node Enters the Pre-main Operation



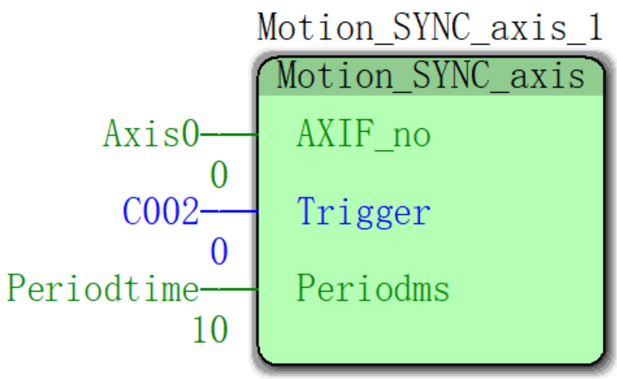

➢ **Input parameters**

| variable name | Features | type of data | The initial value |
|---|---|---|---|
| Axis0 (axis number) | Node number = axis number +1 | USINT | 0 |
| C001 (execute bit) | When the Execute FALSE to TRUE, the instruction is executed | BOOL | FALSE |

1, NMT_Mode (network model) is customizable, template is defined as: 1 (Start Network), 2 (stop the network), 4 (the network into a pre-operation), 8 (reset node), 16 (reset communication), this selected at NMT_Mode = 4, the network into a pre-operation;

2, the internal module functions NS_CC_CANopen_NMT_Write two modules, one master station 18 is pre-operation, operation from one pair of pre-node station No. 1, the condition "C001" trigger delay waiting for the same time is probably 1s, then go Next. (See detailed configuration template "Vector CANopen Configuration")

### 4.1.4 Configure the Synchronizing Cycle of Master-slave Node




➢ **Input parameters**

| variable name | Features | type of data | The initial value |
|---|---|---|---|
| Axis0 (axis number) | Node number = axis number +1 | nUSINT | 0 |
| C002 (execute bit) | When the Execute FALSE to TRUE, the instruction is executed | BOOL | FALSE |
| Periodtime (synchronization period) | Provided from the master node synchronization period (unit: ms) | Time | 10 |

1, the master and slave are the next pre-operational mode, the configuration synchronization cycle at this step, internal modules respectively of the master set from 1006H target station with NS_CC_CANopen_SDO_Write module, the primary must be the same synchronization cycle station is provided from, otherwise lead to control errors!

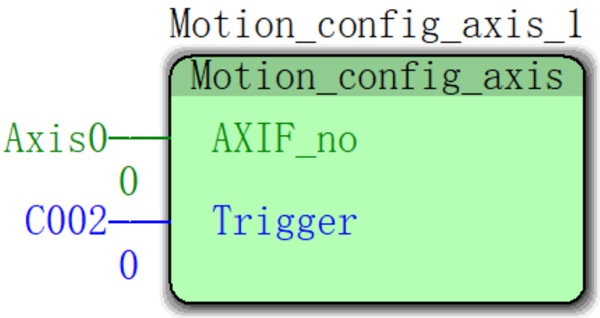
2, C002 trigger condition, configure synchronization cycle to enter the next step without delay;

3, setting the reference standard isochronous period 1006H:

Periodtime = (0.114 * 1.3 * Number of PDO * 1000 / * baud +1+ axes 0.125), the number of templates configured for PDO 4;

(See detailed configuration template "Vector CANopen Configuration")

## 4.1.5 Configure the Process Data of Master-slave Node Master

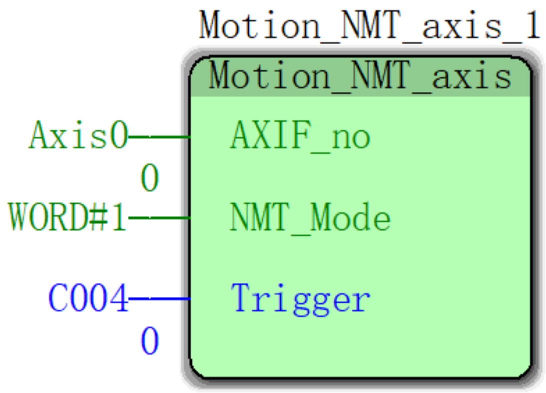



➢ **Input parameters**

| variable name | Features | type of data | The initial value |
|---|---|---|---|
| Axis0 (axis number) | Node number = axis number +1 | USINT | 0 |
| C002 (execute bit) | When the Execute FALSE to TRUE, the instruction is executed | BOOL | FALSE |

This step is a step PDO data configuration communication parameters and mapping parameters, must be configured in a servo interpolation position required by the model object and the format, or can not use the motion control module functions, this step configuration requires careful estimated time configuration consumed to delay, delay time is too short to make some axis configuration fails, making it impossible to control, affect the efficiency too long. (See detailed configuration template "Vector CANopen Configuration")

### 4.1.6 Start Bus




➢ **Input parameters**

| variable name | Features | type of data | The initial value |
|---|---|---|---|
| Axis0 (axis number) | Node number = axis number +1 | USINT | 0 |
| C000 (execute bit) | When the Execute FALSE to TRUE, the instruction is executed | BOOL | FALSE |

1, NMT_Mode (network model) is customizable, template is defined as: 1 (Start Network), 2 (stop the network), 4 (the network into a pre-operation), 8 (reset node), 16 (reset communication), this selected at NMT_Mode = 1, start the network;

2, there are two internal module NS_CC_CANopen_NMT_Write functional modules, each of the master station 18 and the start node number 1, the condition "C004" trigger the bus run mode, then using the motion control module can be controlled from the shaft. (See detailed configuration template "Vector CANopen Configuration")

If you are interested in more detail the configuration process understanding, please refer to the configuration template program.

### 4.2 tension control shaft arranged

4.2.1 Communication configuration------same as: 4.1.1 Communication configuration

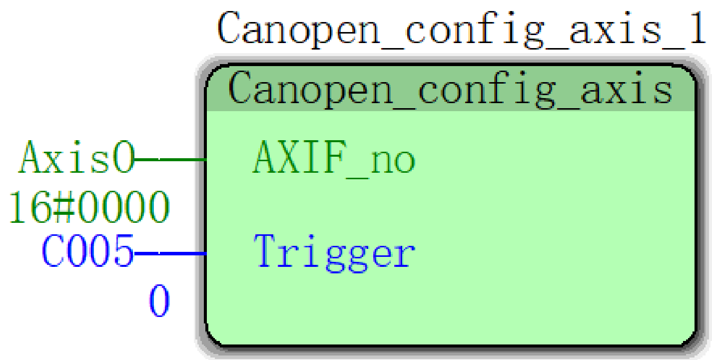
4.2.2 Reset the master-slave node ------- same as: 4.1.2 Reset the master-slave node

4.2.3 Master-slave node Enters the pre-main operation ------- same as: 4.1.3 Master-slave node Enters the pre-main operation

4.2.4 Configure the synchronizing cycle of master-slave node---------------- same as: 4.1.4 Configure the synchronizing cycle of master-slave node

4.2.6 Start Bus ---------------- 4.1.6 Start Bus

4.2.5 Configure the Process data of master-slave node master



# 5. Key Considerations

. A set of reference standards 1006H synchronization cycle: synchronization period = (0.114 * 1.3 * Number of PDO * 1000 / baud +1+ axes * 0.125) ms;

. B setup software CYCLETIME, scan cycle = (+1 synchronization period or 2) ms;

. C configuration data time: Processed tension axis = * 26 * 2ms synchronization period; motion axis = * 12 * 2ms synchronization period;

. D Bus JITTER * 3.56 / 1000 = MS synchronization period;

. E velocity source electronic gear% MB3.9690; CANopen communication baud rate% MW3.4013;

f. If the green light flashes motion controller, the controller and each check station communication station and baud rate settings are correct, or to check the line is disconnected, or there is no terminating resistor connected check or check the signal ground there is no communication together, or to check whether the servo grounded;

. G program modules (except read bus is not encapsulated, write, configuration module) Axis number less than the number corresponding to the station 1, for example, module 0 axis, the actual operation of the station 1;

. H called read bus is not encapsulated, write, configuration modules, and the one-axis number, a so-axis corresponds to a station;

j. spindle default station number 1, i.e. 1, the node can do this as a motion of the spindle axis and the axis number, the tension shaft speed Station No. 1 are read values are stored P14.63 used for tension control, data without master station, receiving from the hair;

I. reliable grounding system requirements, preventing interference.

# Appendix V Register Description

Register Category:

1, 0-1999 power-down does not save;

2, 2000-3999 power-down save;

3, 4000-4095 power-down save of special registers;

4, 4096-9499 power-down does not save;

5, 9500-9999 down does not save special registers;

6, 10000-19999 down without saving.

Special registers:

<table>
<tr><td>% MB3.4010</td><td>MODBUS communication station number</td></tr>
<tr><td>% MB3.4011</td><td>MODBUS communication baud rate baud rate of 4800 *% MB3.4011 in value</td></tr>
<tr><td>% MB3.4012</td><td>CAN Communication station number</td></tr>
<tr><td>% MD3.4013</td><td>CAN communication baud rate, for example, 500K to 500,000</td></tr>
<tr><td>% MB3.4015</td><td>EtherNET IP address of the last 192.168.1.% MB3.4015 number in the range of 93-124</td></tr>
<tr><td>% MB3.4016</td><td>7 MODBUS data length of seven eight 8</td></tr>
<tr><td>% MB3.4017</td><td>MODBUS 0 Even 1 Odd Parity None 2</td></tr>
<tr><td>% MB3.4018</td><td>MODBUS Stop bits 0 1 stop bit for the two stop bits</td></tr>
<tr><td>% MB3.4020</td><td>MODBUS communication delay 2-100 default 2 ms</td></tr>
<tr><td>% MW3.4021</td><td>Cycle time, unit: subtle</td></tr>
<tr><td>% MB3.9720</td><td>The number of cycles required to perform a calculation, Unit:</td></tr>
<tr><td>% MB3.9721</td><td>Axes for participating in the operation, the value of at least 5</td></tr>
<tr><td>% MB3.9536</td><td>By default, left untouched</td></tr>
<tr><td>% MB3.9538</td><td>The value of the assigned register section 2000 to 3999,4000 to 4095 for all registers 0</td></tr>
<tr><td>% MB3.9542 ~<br>% MB3.9556</td><td>Clock register, see 11.6.10 RTC_S (special register clock)</td></tr>
<tr><td>% MB3.9654</td><td>Encoder direction, can not be used together with the shaft information. Setting a corresponding bit corresponding to the encoder shaft reverse</td></tr>
<tr><td>% MW3.9690</td><td>Source encoder provided in bits 0-4
<table>
<tr><td>Shaft mouth</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr>
<tr><td>4</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>3</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
</table>
DI 8-12 provided source (16 represented by point Z<br>Edge edge 13 disposed DI bit to 0 indicates an invalid by the external signal becomes active</td></tr>
<tr><td>% MW3.9692</td><td>Bits 0-4 provided 8-12 source encoder disposed DI sources (Z 16 represents a</td></tr>
</table>

| | |
|---|---|
| | point) 13 is disposed rim edge bit 0 indicates invalid DI by the external signal becomes active |
| % MD3.9694 | The number of pulses between two signals (the pulse source register which is set by the sampling signal DI MW3.9690, need to be used in conjunction with MW3.9690) |
| % MD3.9704 | The number of pulses between two signals (the pulse source register which is set by the sampling signal DI MW3.9692, be used in conjunction with MW3.9692) |
| % MB3.9698 | 0 to 1 over normal Connaught |
| % MB3.9702 | 0 is the overcurrent protection DO, DO 1 is not protected |
| % MB3.9710 | 0 off pulse function virtual imaginary axis, an imaginary axis for the open dummy pulse function |
| % MB3.9711 | Pulse generating imaginary axis designated virtual axis number |
| % MW3.9995 | ARM version number |
| % MW3.9997 | FPGA version number |

# Appendix VI Error Codes

| | |
|---|---|
| 0x1001 | Axis axis number is set beyond the permitted range |
| 0x1002 | Acceleration Acceleration setting beyond the permitted range |
| 0x1003 | Deceleration deceleration setting beyond the permitted range |
| 0x1004 | Jerk Jerk is set outside the permitted range |
| 0x1005 | Velocity speed setting beyond the permitted range |
| 0x1006 | Location PositionOutside the allowable range is set |
| 0x1007 | Direction setting direction beyond the permitted range |
| 0x1008 | Outside the permitted range set BufferMode |
| 0x1009 | ReferenceType setting function beyond the permitted range SetPosition |
| 0x100b | Electronic cam beyond the permitted range table |
| 0x100c | Spindle axis number MasterSetting error |
| 0x100d | Electronic Cam Start Mode StartModeOutside the allowable range is set |
| 0x100e | Electronic cam beyond the permitted range set MasterScaling |
| 0x100f | Electronic cam beyond the permitted range set SlaveScaling |
| 0x1010 | Spindle Source MasterValueSourceOutside the allowable range is set |
| 0x1011 | From the main shaft number conflicts |
| 0x1012 | Electronic gear numeratorRatioNumeratorOutside the allowable range is set (≥0) |
| 0x1013 | Electronic gear denominatorRatioDoutside the permitted range set enominator (> 0) |
| 0x1014 | VelFactor MC_SetOverride parameter setting function outside the permitted range (0 to 500) |
| 0x1015 | Range Error electronic cam SlaveRange |
| 0x1018 | TriggerInput MC_TouchProbe feature set of range (0 ~15) |
| 0x1019 | Mode MC_TouchProbe function setting error |
| 0x1021 | RotaryAxisRadius set out of range (> 0) |
| 0x1022 | FeedAxisRadius set out of range (> 0) |
| 0x1023 | CutLength set out of range (> 0) |
| 0x1026 | SyncAngle setting out of the allowable range (0 ~360) |
| 0x1027 | Peeling is no such function parameters |

| | |
|---|---|
| 0x1028 | RotaryAxisKnifeNum set outside the permitted range (1-16) |
| 0x1045 | NS_MC_SpecialCamIn is equal to 1/2 Mode, MaterValueSource not be 0 |
| | When the mode is equal to 2/3/4 NS_MC_RotaryCutIn, MaterValueSource not be 0 |
| 0x1046 | MC_AXIS_REF, Sample_Time set beyond the permitted range |
| 0x1047 | MC_AXIS_REF, Closed_Loop_Scaling set beyond the permitted range |
| 0x1048 | MC_AXIS_REF, Reductor_Den set beyond the permitted range (> 0) |
| 0x1049 | MC_AXIS_REF, Reductor_Num set beyond the permitted range (> 0) |
| 0x1050 | MC_AXIS_REF, Screw_Lead beyond the permitted range, and setting Disc_Circumference (> 0) |
| 0x1051 | MC_AXIS_REF, Revolving_Axes 1 is set beyond the permitted range Modulo (> 0) |
| 0x1052 | MC_AXIS_REF, ControlMode set beyond the permitted range |
| 0x1053 | MC_AXIS_REF, Moter_Max_V set beyond the permitted range |
| 0x1054 | MC_AXIS_REF, Moter_PPC set beyond the permitted range |
| 0x1055 | MC_AXIS_REF, Offset_Max_V set beyond the permitted range |
| 0x1056 | MC_CamIn, ActivationMode is 2, ActivationPosition less than 0 or greater than the mold |
| | NS_MC_SpecialCamIn, ActivationPosition setOut of the allowable range (≧ 0) |
| 0x1057 | NS_MC_SpecialCamIn, DistanceOffset_Master setOut of the allowable range (≧ 0) |
| 0x1058 | DistanceAdd set outside the permitted range (≧ 0) |
| 0x1059 | DistanceSync set outside the permitted range (≧ 0) |
| 0x1060 | DistanceDec set outside the permitted range (≧ 0) |
| 0x1061 | NC_CartesianCoordinate beyond the permitted range setting module Depth |
| 0x1062 | NC_CartesianCoordinate beyond the permitted range setting module Junction_Deviation |
| 0x1063 | NC_CartesianCoordinate beyond the permitted range setting module Arc_Tolerance |
| 0x1064 | NS_MC_SpecialCombineAxes beyond the permitted range setting module Cam_DistanceOffset_Master (> 0) |
| 0x1068 | NS_MC_SpecialCombineAxes module Cam_Pulse_Per_Unit_M set outside the permitted range (> 0) |

| | |
|---|---|
| 0x1069 | NS_MC_SpecialCombineAxes module NCFile specified file was not found |
| 0x1070 | NC_MoveCircular, CircMode set out of the allowable range (0 to 2) |
| 0x1071 | NC_MoveCircular, PathChoice set out of the allowable range (0 to 1) |
| 0x1072 | NC_MoveCircular module, Param_R, Param_I, Param_J, Param_K all 0 |
| 0x1073 | NC_GroupEnable, the current state of the shaft when the shaft is not present 0/1/2 as Standstill, the shaft can not enable the group |
| 0x1074 | NC_MoveLinear / NC_MoveCircular/ NC_CartesianCoordinate when executed, not using NC_GroupEnableEnable axis groups |
| 0x1075 | NC_GroupEnable beyond the permitted range setting module Axis_Num_X (0) |
| 0x1076 | NC_GroupEnable beyond the permitted range setting module Axis_Num_Y (1) |
| 0x1077 | NC_GroupEnable beyond the permitted range setting module Axis_Num_Z (2) |
| 0x1078 | NS_MC_RotaryCutIn, cut length is set smaller than the knife CutLength circumference$\frac{1}{10}$ |
| 0x1079 | NS_MC_RotaryCutIn, Cut_DI_Num set beyond the permitted range (0 ~15) |
| 0x1080 | NS_MC_RotaryCutIn, Mark_DI_Num set beyond the permitted range (0 ~15) |
| 0x1081 | MC_CombineAxes, CombineMode set beyond the permitted range (0 ~1) |
| 0x1082 | NS_MC_SpecialCombineAxes, Periodic_Master_Units input out of range (> 0) |
| 0x1083 | This information shaft axis error command charged |
| 0x1084 | Spindle axis information corresponding to the instruction of this error |
| 0X2001 | The MC_Power, servo master slave returns a status word, a failure message from the station |
| 0x2002 | The MC_Power, there is an error on the bus, interference such as a bus, unequal baud |

| | |
|---|---|
| 0x2003 | NS_CC_CANopen_NMT_Read, read the state does not make sense |
| 0x2004 | NS_CC_CANopen_NMT_Write, write the state does not make sense |
| 0x2005 | NS_CC_CANopen_SDO_Read, NS_CC_CANopen_SDO_Write, over buffer |
| 0x2006 | NS_CC_CANopen_SDO_Write wrong data type, only 1,2,4 |
| 0x2007 | NS_CC_CANopen_SDO_Read, NS_CC_CANopen_SDO_Write slave reply timeout |
| 0x2008 | NS_CC_CANopen_TPDO, NS_CC_CANopen_RPDO index out of bounds |
| 0x2009 | NS_CC_CANopen_TPDO, NS_CC_CANopen_RPDO type of error, normal range of 2,3,4,5,6,7 |
| 0x2401 | Axis_no sets the function block out of the allowable range (0-6) |
| 0x2402 | Active_Axis sets the function block beyond the permitted range |
| 0x2403 | Outside the permitted range set CNT_ID |
| 0x2404 | Outside the permitted range set Event_ID |
| 0x2405 | Outside the permitted range set DI_ID |
| 0X4000 | The same axis with the same module exceeds a predetermined number, please refer to allowed number rangePrecautions: |
| 0X4001 | NS_NC_ReadParameterP modulearameterNoutside the permitted range set umber |
| 0x4101 | The current operating state of the shaft to ErrorStop or Disabled, can not perform any movement instruction. |
| 0x4102 | Axis current operating state of the Stopping, can not be performed in addition to MC_Any movement commands other than Stopping. |
| 0x4103 | Execution MC_Home instruction requires the axis to StandStill state |
| 0x4104 | MC_CamOutOnly modules in the currently running instructionMC_CamIn when to run |
| 0x4105 | MC_GearOut module only if the current command is runMC_GYou can run the earIn |
| 0x4106 | Current operating status of the shaftHoming, can not be performed in addition to MC_SAny motion command other than topping |
| 0x4107 | Current BufferMode cache beyond the permitted range, please readBufferMode cache description |
| 0x4108 | RunCommandUnder no pointer command, the bottom part of the error |

| | |
|---|---|
| 0x4150 | Error current state of the shaftStop state, can not be executed NS_MC_Jog |
| 0x4151 | Disabled axis current state of the state, not performing NS_MC_Jog |
| 0x4152 | Homing axis current state of the state, not performing NS_MC_Jog |
| 0x4153 | Axis current state of the Stopping state, can not be executed NS_MC_Jog |
| 0x4201 | Error current state of the shaftStop/ Disabled, can not perform MC_Phasing instruction |
| 0x4202 | Axis current state of the Stopping, M can not be performedC_Phasing instruction |
| 0x4203 | This shaft MC_CamIFollowing spindle, M n next instructionC_PMaster hasing spindle specified instructions follow the master axis according to the present non- |
| 0x4205 | This shaft MC_GeanIFollowing spindle, M n next instructionC_PMaster hasing spindle specified instructions follow the master axis according to the present non- |
| 0x4207 | This follows the spindle axis at NS_MC_RotaryCutIn instruction, MC_PMaster hasing spindle specified instructions follow the master axis according to the present non- |
| 0x4209 | MC_Phasing command setting spindle shaft from the Master and Slave master-slave follower relationship, this instruction is executed Invalid |
| 0x4210 | This axis NS_MC_SpecialCamIFollowing spindle, M n next instructionC_PMaster hasing spindle specified instructions follow the master axis according to the present non- |
| 0x4211 | This shaftNS_MC_SpecialCombineAFollowing spindle, M xes next instructionC_Phasing instruction specifies a spindle shaft according to the present non-Master following spindle |
| 0x4212 | This shaft is in Mode 1,NS_MC_SpecialCombineAFollowing spindle, M xes next instructionC_Phasing instruction can not be executed |
| 0x4251 | Error current state of the shaftStop/ Disabled, can not perform MC_MoveSuperimposed instruction |
| 0x4252 | Axis current state of the Stopping, M can not be performedC_MoveSuperimposed instruction |
| 0x4351 | MC_Home, FirstVecocity set out of range (> 0) |
| 0x4352 | MC_Home, SecondVecocity set out of range (> 0) |
| 0x4353 | MC_Home, Mode setting is outside the range |
| 0x4400 | Mode rotating shaft (Revolving_Axes =TRUEUnder), StopByPos set position is out of range |

## Welcome your valuable feedback!

We would like to wholeheartedly serve you, and strive to improve the already white. As the editor is limited, mistakes are inevitable urge readers to hesitate to correct me. We hope that you read this book, when using the product, such as an error is found, discusses the use of unknown or can not find the appropriate interpretation, please call us or fill in the feedback form send it to us, we sincerely look forward to your comments. Call us:

Customer service hotline: 40008-50004