INOVANCE

FORWARD, ALWAYS PROGRESSING

H5U and Easy Series Programmable Logic Controllers

Instruction Guide

Industrial Automation

Intelligent Elevator

New Energy Vehicle

Industrial Robot

Rail Transit

Data code 19012250 A12

Preface

Introduction

The H5U series small-sized high performance PLC carries compact structure and 16 inputs/14 outputs.

Easy series small- and medium-sized PLCs are available in eight models, covering the demands of automation equipment requiring small footprint, multi-axis motion control, accurate temperature control, and easy networking.

This guide describes basic and complex instructions and examples.

Target Audience

This manual is intended for the following audiences:

- Electrical engineers
- Software engineers
- System engineers

Cautions for New Users

Read this manual carefully if you use the PLC for the first time. If you have any problem concerning the functions or performance of the product, contact our technical support.

Related Manuals

Category	Document Name	Data No.
User Guide	*H5U Series Programmable Logic Controllers User Guide*	19011517
User Guide	*Easy Series Programmable Logic Controllers User Guide*	PS00006444
Programming and application guide	*H5U& Easy Series Programmable Logic Controllers Programming and Application Guide*	19012249

Change History

Date	Version	Description
May 2023	A12	Added description of the following LiteST explicit conversion instructions: INT_TO_BYTE, DINT_TO_BYTE, BOOL_TO_BYTE, REAL_TO_BYTE, BYTE_TO_<TYPE>, and TO_BYTE. Added description of the following instructions: MC_GearInPos, SerialSend, SerialRcv, MB_Master, and MB_Client. Corrected some instruction errors.
November 2022	A11	Added LiteST expression for some instructions. Added description of the following instructions: ENC_SetLineRotationMode, ENC_SetUnit, and MC_SetOverride. Corrected some instruction errors.
September 2022	A10	Added description of EIP communication instructions. Added description of the following instructions: SORTC, DSORTC, SORTR, DSORTR, RAMP, and DRAMP. Added description of Easy series PLC instructions. Corrected some instruction errors.

Date	Version	Description
May 2022	A09	Made minor changes.
August 2021	A08	Added description of the following instructions: DECO, DDECO, ENCO, and DENCO. Added description of the following instructions: ETC_RestartMaster, MC_FollowVelocity, MC_SetAxisConfigPara, and MC_DigitalCamSwitch. Updated the list of instructions. Updated the list of fault codes.
May 2021	A07	Kept the material versions consistent.
March 2021	A03	Corrected errors in the previous version. Added cam instructions and details. Added serial encoder axis instructions and details. Updated the list of error codes.
August 2020	A02	Corrected errors in the previous version.
June 2020	A01	Added axis group instructions and details. Updated the list of error codes. Corrected errors in the previous version.
December 2019	A00	First release.

Document Acquisition

This guide is not delivered with the product. You can obtain the PDF version by the following method:

- Visit Inovance's website (http://www.inovance.com) to download the PDF file.

Table of Contents

1 Overview

1.1 Instruction Composition

1.1.1 LD Instructions

An instruction consists of the opcode and operand.

- Opcode: Instruction function description
- Operand: Data used in the instruction

The operand includes the input data, output data, and numeric data.

Input (S)

The input indicates data used in the operation.

The usage of input data is described as follows according to the variables and elements specified in each instruction.

Table 1–1 Input data

Category	Description
Constant	A constant specifies a numerical value used in the operation. It cannot be changed during program execution since it is configured when the program is created.
Element and variable	During program execution, data used in the instruction can be changed by modifying data stored in the specified element.

Output (D)

The output element stores data after operations. Sometimes, data used in the operation needs to be stored in the target before the operation, depending on the specific instruction.

The following is an example of the addition operation of INT type data:

ADD	S1	S2	D
			↓ ①

①: Only the operation result is saved.

The D element must be configured with a variable or element for storing data.

Number of Elements/Transfers/Data/Strings (n)

In instructions involving specifying multiple elements, the number of repetitions, the number of groups of data to be processed, and the number of strings, the numbers of elements, transfers, data, and strings used are determined by n.

The following is an example of a block transfer instruction:

BMOV	S	D	n

①

①: The data to be transferred is specified by the BMOV instruction.

1.1.2 LiteST Instructions

A LiteST instruction consist of an instruction name, parameters, and return values, which are defined as follows:

- Instruction name: Instruction function description
- Parameters: Data used in the instruction
- Return values: Result obtained after the instruction is executed

Input (S)

The input indicates data used in the operation.

The usage of input data is described as follows according to the variables and elements specified in each instruction.

Table 1–2 Input data

Category	Description
Constant	A constant specifies a numerical value used in the operation. It cannot be changed during program execution since it is configured when the program is created.
Element and variable	During program execution, data used in the instruction can be changed by modifying data stored in the specified element.

Output (D)

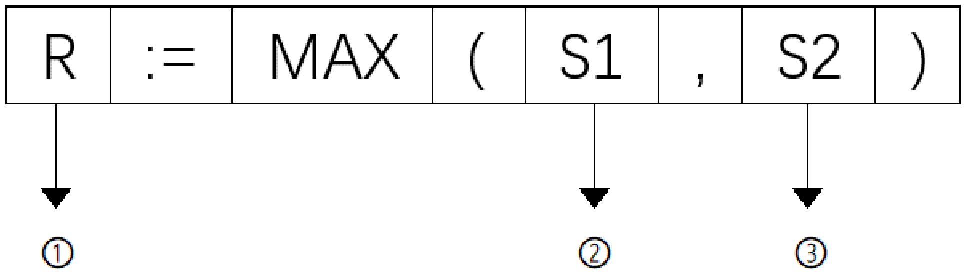
The output element stores data after operations. Sometimes, data used in the operation needs to be stored in the target before the operation, depending on the specific instruction.

Return Value (R)

The return value is the result obtained after the instruction is executed. Sometimes, data used to operate on the instruction result needs to be stored in the target after the operation, depending on the specific instruction.

The following is an example of the comparison operation of INT type data:

R:=MAX(S1, S2)



MAX(S1, S2)

①: Only the operation result

②: Input parameter

③: Input parameter

1.1.3 Lists of Elements and Variables

The PLC supports bit elements, word elements, special elements, variables, arrays, structures, and custom variables.

Bit Elements

Type	Range	Number of Points	Data Type	Description
X	X0 to X1777	1024 points, octal	BOOL	Not retained upon power failure
Y	Y0 to Y1777	1024 points, octal	BOOL	Not retained upon power failure
M	M0 to M7999	8000 points	BOOL	M0 to M999 not retained upon power failure, M1000 to M7999 retained upon power failure
S	S0 to S4095	4096 points	BOOL	S0 to S999 not retained upon power failure, S1000 to S4095 retained upon power failure
B	B0 to B32767	32768 points	BOOL	B0 to B999 not retained upon power failure, B1000 to B32767 retained upon power failure

Word Elements

Type	Range	Number of Points	Data Type	Description
D	D0 to D7999	8000 points	BOOL/INT/DINT/REAL	D0 to D999 not retained upon power failure, D1000 to D7999 retained upon power failure
R	R0 to R32767	32768 points	BOOL/INT/DINT/REAL	R0 to R999 not retained upon power failure, R1000 to R32767 retained upon power failure
W	W0 to W32767	32768 points	BOOL/INT/DINT/REAL	W0 to W999 not retained upon power failure, W1000 to W32767 retained upon power failure

Custom Variables

Type	Range	Capacity	Data Type	Description
Pointer	-	4096 points (32-bit)	BOOL/INT/DINT/REAL array	Not retained upon power failure
BOOL	-	2 MB (8-bit)	BOOL/BYTE/INT/DINT/ REAL/IP/STRING variable BOOL/BYTE/INT/DINT/ REAL/IP/STRING array BOOL/BYTE/INT/DINT/ REAL/IP/STRING compound structure	256 KB data retained upon power failure, other data not retained upon power failure
BYTE				
INT				
DINT				
REAL				
IP				
STRING				

Special Elements

Type	Function	Range	Number of Points	Description
L	Jump label	L0 to L1023	1024 points	Used in combination with the CJ and LBL instructions
K	Decimal	K-32,768 to K32,767 (16-bit), K-2,147,483,648 to K2,147,483,647 (32-bit)	-	-
H	Hexadecimal	H0000 to HFFFF (16-bit), H00000000 to HFFFFFFFF (32-bit)	-	-
E	Floating-point number, real number	$-1.0*2e^{128}$ to $-1.0*2e^{-126}$, 0, $1.0*2e^{-126}$ to $1.0*2e^{128}$ (32-bit)	-	-
Character	Character, string	-	-	Used as instruction parameters

M8000 and D8000 Special Elements

Special Element	Function Description	Access Permissions
M8000	ON during running of the user program	Read-only
M8001	Negated M8000 state	Read-only
M8002	ON in the first operation cycle of the user program	Read-only
M8003	Negated M8002 state	Read-only
-	-	-
M8011	Free-run clock with a cycle of 10 ms	Read-only
M8012	Free-run clock with a cycle of 100 ms	Read-only
M8013	Free-run clock with a cycle of 1s	Read-only
M8014	Free-run clock with a cycle of 1 min	Read-only
-	-	-
M8020	Zero flag	Read-only
M8021	Borrow flag	Read-only
M8022	Carry flag	Read-only
M8029	Multi-cycle instruction execution completion flag, applicable to the RAMP, SORTC, and SORTR instructions	Read-only

Special Element	Function Description	Access Permissions
-	-	-
M8040	SFC, SFC disabling flag	Read-write
-	-	-
M8161	OFF: 16-bit mode; ON: 8-bit mode; Bit processing mode for ASCII/HEX/CCD/LRC/CRC/RS	Read-write
M8163	BINDA instruction output character switchover flag (retained or switched to 0000h)	Read-write
M8165	SORTR instruction descending sort enabling flag	Read-write
M8168	SMOV instruction data format setting: OFF-BCD mode or ON-HEX mode	Read-write
M8333	Flag indicating all BKCMP instruction matrix comparison results are 1	Read-only

Other undefined elements after M8000 cannot be used in the programs.

Special Elements	Function Description	Access Permissions
D8066	Critical errors in user programs and instructions (triggered, not reset)	Read-only
D8067	Minor errors in user programs and instructions (triggered, not reset)	Read-only

The access permissions are described as follows:

- Read-only: The PLC output is read-only by the user. Data written by the user will be overwritten.
- Read-write: The PLC input can be read and written by the user.

1.2 Elements

1.2.1 Bit Elements

The PLC supports bit elements. The following table describes the specific type, range, number of points, and description of bit elements.

Type	Range	Number of Points	Data Type	Description
X	X0 to X1777	1024 points, octal	BOOL	Input
Y	Y0 to Y1777	1024 points, octal	BOOL	Output
M	M0 to M7999	8000 points	BOOL	M0 to M999 not retained upon power failure, M1000 to M7999 retained upon power failure
S	S0 to S4095	4096 points	BOOL	S0 to S999 not retained upon power failure, S1000 to S4095 retained upon power failure
B	B0 to B32767	32768 points	BOOL	B0 to B999 not retained upon power failure, B1000 to B32767 retained upon power failure

1.2.2 Word Elements

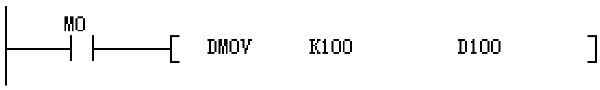
The PLC supports word elements. The following table describes the specific type, range, number of points, and description of word elements.

Type	Range	Number of Points	Data Type	Description
D	D0 to D7999	8000 points	BOOL/INT/DINT/REAL	D0 to D999 not retained upon power failure, D1000 to D7999 retained upon power failure
R	R0 to R32767	32768 points	BOOL/INT/DINT/REAL	R0 to R999 not retained upon power failure, R1000 to R32767 retained upon power failure
W	W0 to W32767	32768 points	BOOL/INT/DINT/REAL	W0 to W999 not retained upon power failure, W1000 to W32767 retained upon power failure

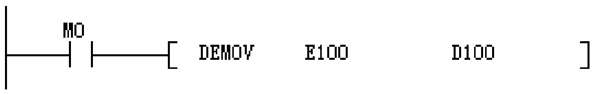
Example

1. Word element used as a 16-bit integer
   Use the 16-bit assignment instruction to assign the value 100 to the word element D100, which occupies D100.

2. Word element used as a 32-bit integer
   Use the 32-bit assignment instruction to assign the value 100 to the word element D100, which occupies occupy D100 (low-order) and D101 (high-order).



3. Word element used as a floating-point number
   Use the floating-point instruction to assign the value 100 to the word element D100, which occupies D100 and D101.

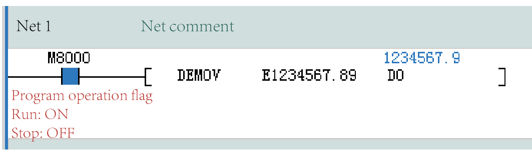


1.2.3 Special Elements

The PLC supports special elements. The following table describes the specific type, range, and description of special elements.

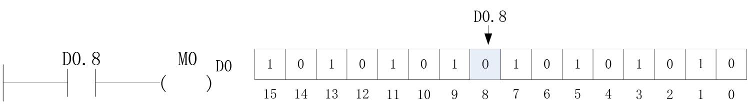
Type	Function	Range	Number of Points	Description
SBR	Subprogram label	SBR0 to SBR1023	1024	Used by the CALL instruction. Subprograms can be set as common subprograms or encrypted subprograms, which share the capacity of the system program area.
L	Jump label	L0 to L1023	1024 points	Used in combination with the CJ and LBL instructions
I	External interrupt	-	4	Interrupt label, X port rising edge, falling edge, rising and falling edge
	Timer interrupt	-	4	Timing duration (ms)
	Compare interrupt	-	16	Limited by the number of internal encoder axes (high-speed counters)
K	Decimal	K-32,768 to K32,767 (16-bit), K-2,147,483,648 to K2,147,483,647 (32-bit)	-	-
H	Hexadecimal	H0000 to HFFFF (16-bit), H00000000 to HFFFFFFFF (32-bit)	-	-
E	Floating-point number, real number	$-3.402823e^{+38}$ to $-1.175495e^{-38}$, 0, $+1.175495e^{-38}$ to $+3.402823e^{+38}$	-	Up to 7 decimal significant digits for a single-precision floating-point number (the excess will be automatically rounded off)
Character	Character, string	-	-	Used as instruction parameters

A single-precision floating-point number has a maximum of 7 significant decimal digits. If the 9-bit binary floating-point number 1234567.89 is transferred to the destination location D0, the actual value of D0 is 1234567.9. The precision is reduced.



1.2.4 Bit-based Operation on Word Elements

Bit-based operations on word elements can be implemented by using a dot (.). For example, writing D0.8 during programming indicates an operation on the 8th bit of the D0 word element.

Example: 

The bits of the word element are counted from the 0th bit. When the 8th bit of D0 is 0, the output M0 is OFF; when the 8th bit of D0 is 1, the output M0 is ON.

1.3 Variables

1.3.1 Custom Variables

In a PLC programming system, in addition to using direct addresses, such as the X, Y, M, D, R and other elements, for programming, you can also use variables without specific storage addresses for programming to implement the required control logic, or the complete control process of the application object, so as to facilitate code compiling and improve the readability of the code.

Table 1–3 Supported custom variables

<table>
<tr><th>Type</th><th>Capacity</th><th>Data Type</th><th>Description</th></tr>
<tr><td>Pointer</td><td>4096 points (32-bit)</td><td>BOOL/INT/DINT/REAL</td><td>Pointer Variable
Not retained upon power failure</td></tr>
<tr><td>BOOL</td><td rowspan="7">2 MB (8-bit)</td><td rowspan="7">BOOL/INT/DINT/REAL/IP/STRING/ BYTE variable
BOOL/INT/DINT/REAL/IP/STRING/ BYTE array
BOOL/INT/DINT/REAL/IP/STRING/ BYTE compound structure</td><td rowspan="7">256 KB data retained upon power failure
Other data not retained power failure</td></tr>
<tr><td>INT</td></tr>
<tr><td>DINT</td></tr>
<tr><td>REAL</td></tr>
<tr><td>IP</td></tr>
<tr><td>STRING</td></tr>
<tr><td>BYTE</td></tr>
</table>

1.3.2 Defining Variables

The PLC supports custom variables. You can define a global variable and directly use the variable name during programming. Abide by the following rules when naming a global variable:

- It contains only letters, digits, Chinese characters, and underscores (_) and does not start with a digit or underscore (_).
- It is not the same as the name of an element, constant, standard data type, instruction, subprogram, or interrupt subprogram.
- It cannot be keywords such as ARRAY, TRUE, FALSE, ON, OFF, and NULL.

Variable Data Types

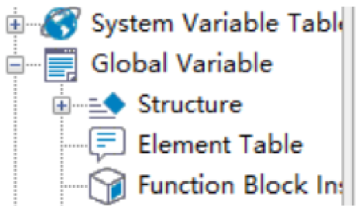
Structures and arrays are supported. The following table lists the supported data types.

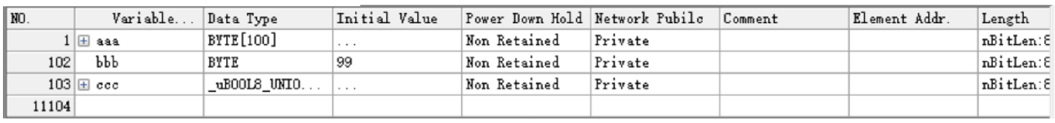
Table 1–4 Variable data types

Data Type	Description
BOOL	Boolean
INT	Single-word integer
DINT	Double word integer
REAL	Real number
STRING	String type
IP	IP
BYTE	Byte

Defining Global Variables

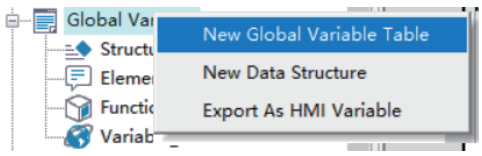
"Global Variable" in the project management window is used for variable management, allowing you to add, delete, and edit variables.





NO.	Variable...	Data Type	Initial Value	Power Down Hold	Network Public	Comment	Element Addr.	Length
1	aaa	BYTE[100]	...	Non Retained	Private			nBitLen:8
102	bbb	BYTE	99	Non Retained	Private			nBitLen:8
103	ccc	_uBOOL8_UNIO...	...	Non Retained	Private			nBitLen:8
11104								

1. Add a variable table and variables. Right-click "Global Variable" and choose "New Global Variable Table" to create a global variable table.



2. Double-click the variable table to go to the variable editing interface.

- Edit a variable: Double-click the text box to edit or click the drop-down box to select.
- Add a variable: Right-click and choose "Insert Row(&I)".
- Delete a variable: Right-click the row to be deleted and choose "Delete Row(&L)".

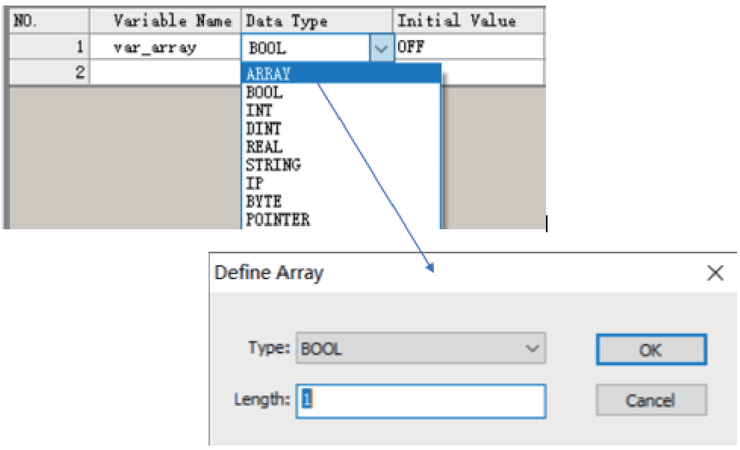
Parameter Name	Description
Variable Name	Custom variable name. You can directly use the variable name for programming.
Type	The data types include BOOL, INT, DINT, REAL, IP, STRING, and BYTE variables, BOOL, INT, DINT, REAL, IP, STRING, and BYTE arrays, and BOOL, INT, DINT, REAL, IP, STRING, and BYTE structures. If the data type is an array, you can set the type and length of the array variable in the displayed dialog box. If the data type is a pre-defined structure, you can define a structure variable.
Initial Value	You can assign an initial value to a variable. For arrays and structures, the initial value of each element can be specified individually.

Parameter Name	Description
Power Down Hold	"Power Down Hold" can be set to "Non Retained" or "Retained". The specified initial value is valid only when this parameter is set to "Non Retained".
Network Public	This parameter can be set to "Private", "Public", or "In/Out". For structure, specific union, structure array, and specific union array variables, this parameter must be set to "Private". When this parameter is set to "Public", a label configuration file named "LabelConfig.xml" will be generated in the "InteractiveFile" folder under the project directory after project compiling. Importing this configuration file into third-party software enables label communication.

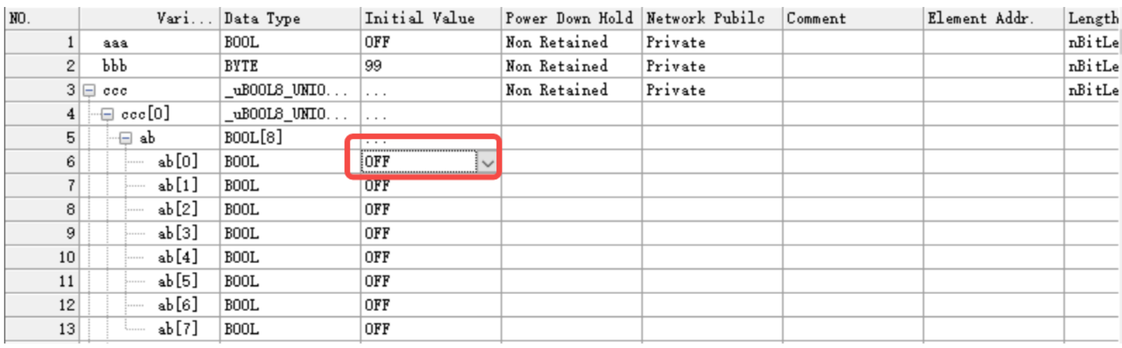
1.3.3 Defining Arrays

During user programming, if the data type is set to "ARRAY", an array can be defined.

1. Select the type and length of the array variable in the displayed dialog box and click "OK" to define an array.



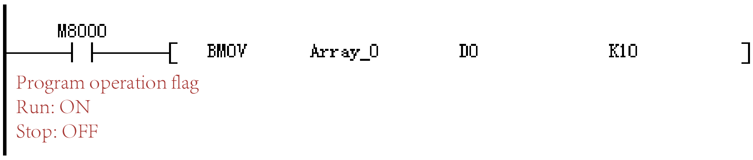
2. Click "+" next to the array variable to edit the initial values and comments of member variables.



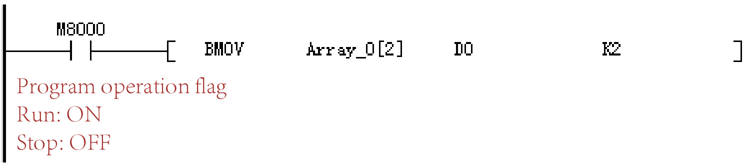
When an array is used in an instruction, if the array subscript is not specified, the access starts from the first element of the array. If the array subscript is specified, the access starts from the element specified by the subscript.

The following are two examples.

- Assign Array_0[0]–Array_0[9] to D0–D9.

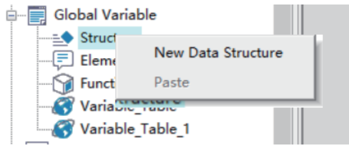


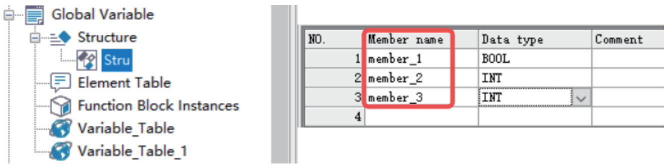
- Assign Array_0[2]–Array_0[3] to D0–D1.



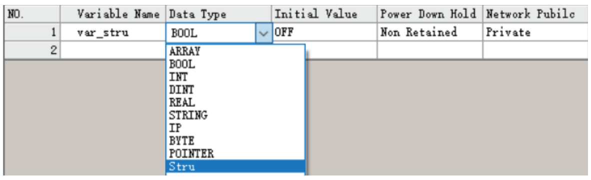
1.3.4 Defining Structures

To define a structure variable, you need to define the data structure of the structure in advance. Right-click "Structure" under "Global Variable", choose "New Data Structure", and enter a structure name. The structure is defined. When defining a variable in the variable table, you can select this structure as the data type of the variable to define the variable as a structure variable.





After the structure and member variables are created, you can select "Stru" in the "Data Type" column to define a structure variable.

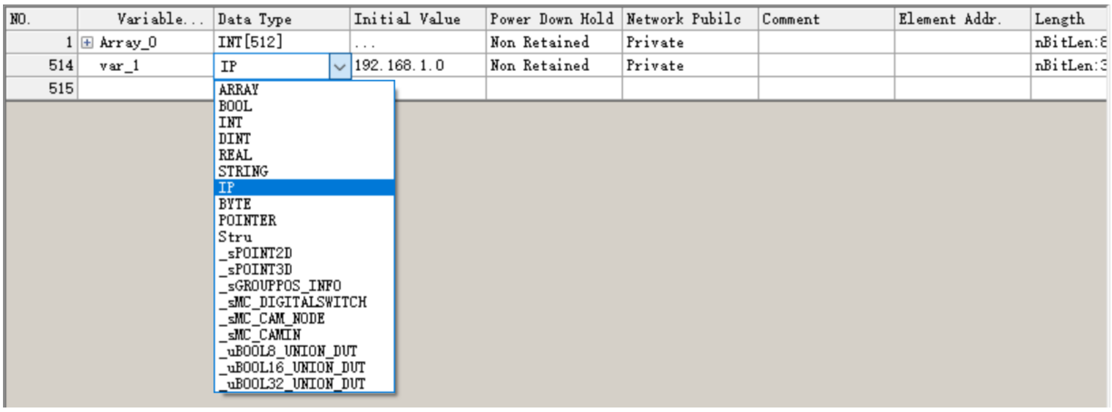


Click the "Initial Value" column of the structure variable to set the initial values of structure variable members.

1.3.5 Defining IP Variables

You can define IP variables in the variable table or program. An IP variable occupies 32 bits, and the default value is "192.168.1.0".

- Select "IP" from the "Type" drop-down list.



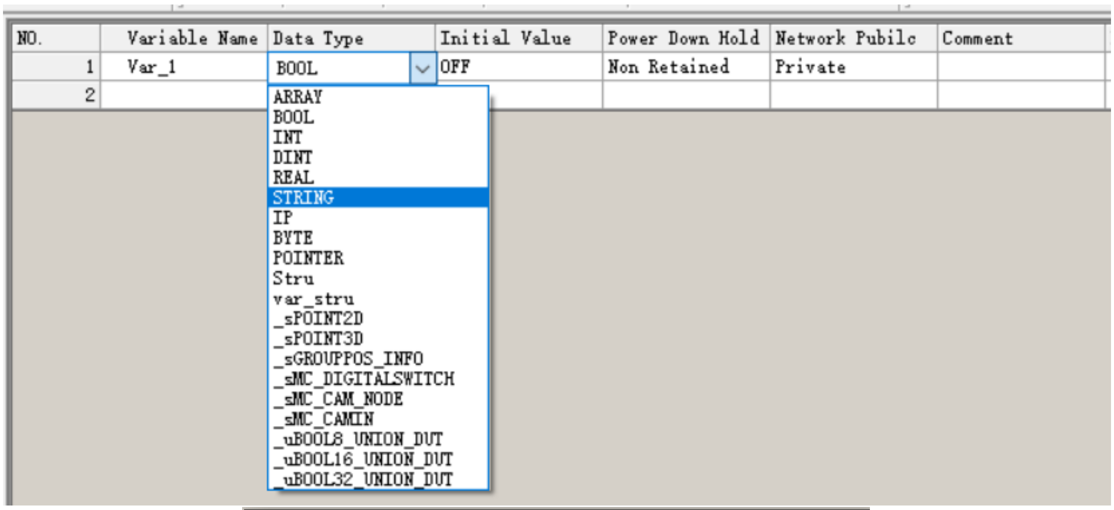
- Use an IP variable in the ST program, and assign a value to the IP variable by using single quotation marks.

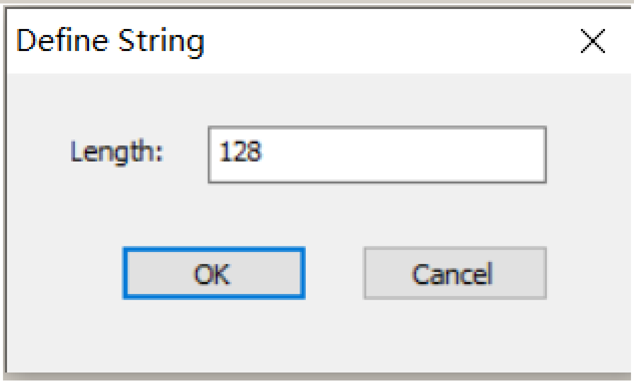
```
10  var_1:='10.45.121.90';
```

1.3.6 Defining Strings

You can define string variables in the variable table or program.

- Select "STRING" from the "Type" drop-down list of the variable table, and set the length of the string in the displayed dialog box.
  The default length is 128 bytes and the maximum length is 256 bytes. The last byte is the terminator by default.





- Use a string variable in the ST program, and assign a value to the string variable by using single quotation marks.

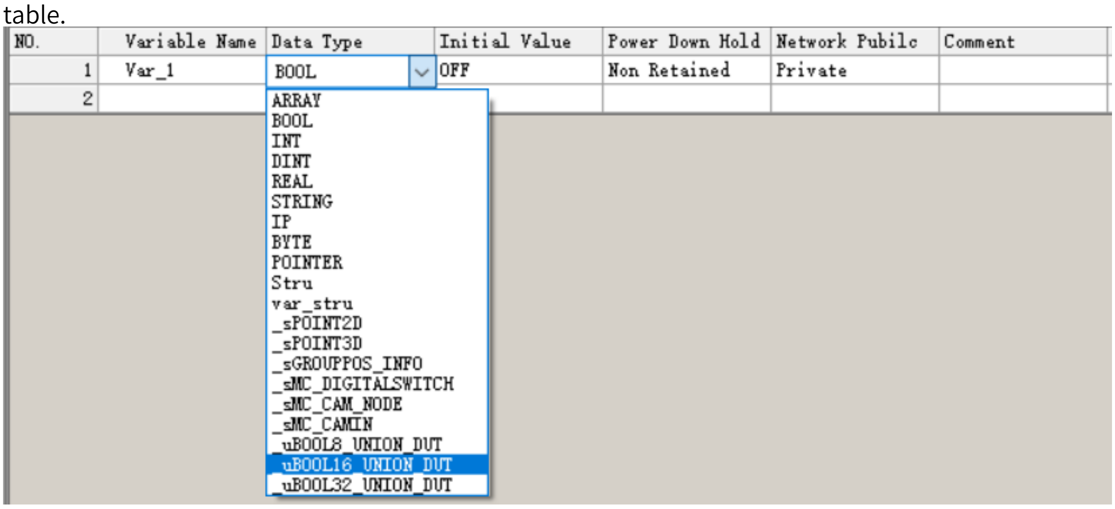
```
1 var_1:='abc';
```

1.3.7 Defining Specific Unions

A specific union is similar to a structure in that they both are collections of different types of elements. The difference lies in the fact that each member of a structure has its own independent storage space, while the members of a specific union share the same memory space (which is why a specific union is called a union). This will inevitably cause the members to overwrite each other, resulting in data loss. Therefore, the ideal application scenario for a specific union is when its members are not used simultaneously, but rather one after another.

You can define specific union variables in the variable table or program. There are three types of specific union variables: _uBOOL8_UNION_DUT, _uBOOL16_UNION_DUT, and _uBOOL32_UNION_DUT, corresponding to lengths of 1 byte, 2 bytes, and 4 bytes, respectively.

- Select the required specific union variable type from the "Type" drop-down list of the variable table.



Take _uBOOL32_UNION_DUT as an example. Create a variable in the variable table, and select "_uBOOL32_UNION_DUT" from the "Type" drop-down list.

NO.	Variab...	Data Type	Initial Value	Power Down Hold	Network Pubilc	Comment	Element Addr.	Length
1	var_1	_uBOOL32_UNI...	...	Non Retained	Private		D0	nBitLen:3
2	ab	BOOL[32]	...					
35	ai	INT[2]	...					
36	ai[0]	INT	0				D0	
37	ai[1]	INT	0				D1	
38	abyte	BYTE[4]	...					
39	abyte[0]	BYTE	0				D0	
40	abyte[1]	BYTE	0				D0.8	
41	abyte[2]	BYTE	0				D1	
42	abyte[3]	BYTE	0				D1.8	
43	byte0	BYTE	0				D0	
44	byte1	BYTE	0				D0.8	
45	byte2	BYTE	0				D1	
46	byte3	BYTE	0				D1.8	
47	i0	INT	0				D0	
48	i1	INT	0				D1	
49	f0	REAL	0.000000				D0	
50	di0	DINT	0				D0	
51								

- In a program, you can access different members of a specific union variable by using the dot operator ("."). This allows you to parse variables in different scenarios.

1.3.8 Using Variables

After a variable is defined, you can directly use the variable name for programming without assigning elements.

- When a common variable is used, directly use the variable name during programming.
- When an array variable is used, use "[Number]" to indicate an array element during programming. The number starts from 0.
- When a structure variable is used, use "Structure variable name.Member variable" to indicate a structure member during ST programming.
- When an IP or string variable is used, use a value enclosed in a pair of single quotation marks ('Value') to indicate the value of the variable.

```
var_1:='10.45.121.90';
```

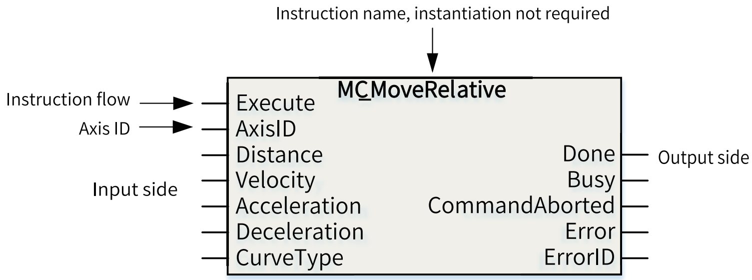
```
var_1:='abc';
```

For BYTE, INT, and DINT variables and arrays, you can perform bit operations using the syntax "variable_name.bit_number" in programming. For details, see *"1.2.4 Bit-based Operation on Word Elements" on page 21*.

1.4 Graphical Block Instructions

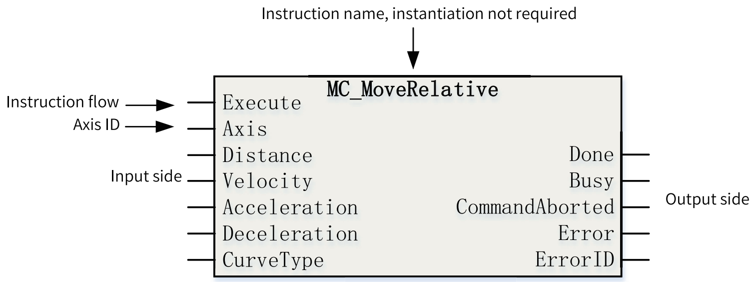
1.4.1 Instruction Composition

Some instructions support graphical block programming. An graphical block instruction is composed of the instruction name, flow signal, input side, and output side. The following figure shows the composition of a graphical block instruction of a motion control axis.



The floating-point numbers such as the target position and target velocity in the instructions are single-precision floating-point data. Therefore, the values in the instructions must meet the requirements of the range and precision of single-precision floating-point data when being processed in the PLC program. That is, a value should fall between –3.4E38 and +3.4E38, with a maximum of 7 significant digits. If a value has more than 7 significant digits, the excess part will be automatically rounded.

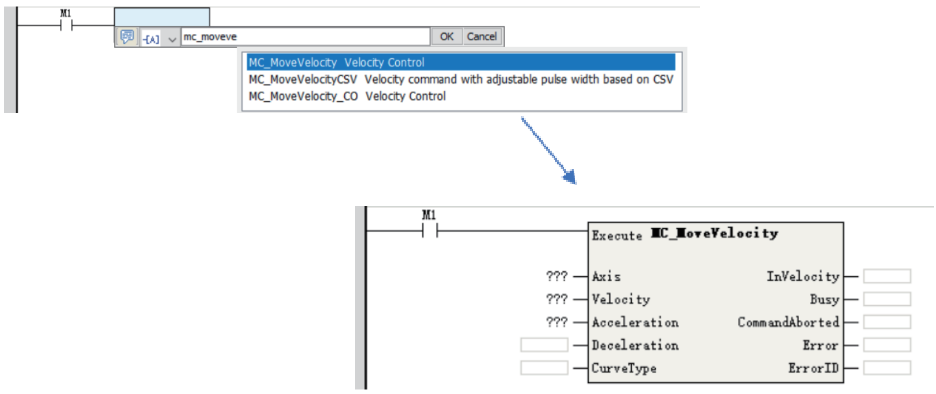
Since AutoShop 4.0.0.0 with PCB software 3.0.0.0, the motion control axis control instructions (EtherCAT/pulse output, pulse input) of graphical blocks support access by axis name. "AxisID" is changed to "Axis", and access by axis ID is still supported.



1.4.2 Programming

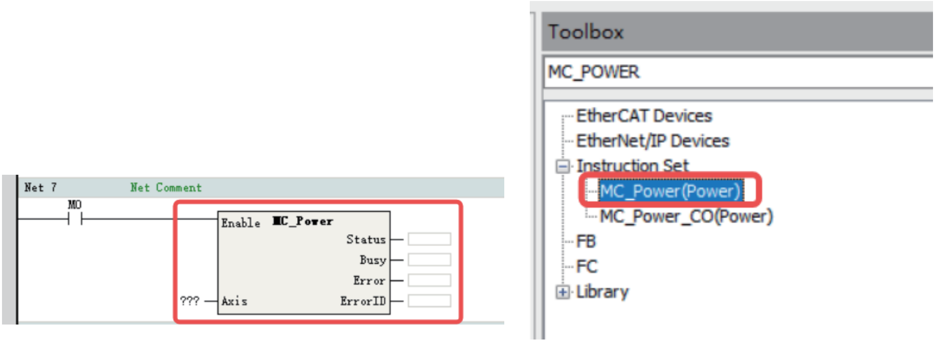
During programming, you only need to enter the name of a graphical block instruction and simply press the "Enter" key to add the graphical block instruction to the program network. You can also directly edit the instruction parameters.

- When editing a ladder diagram, enter an instruction name or select an instruction name according to the instruction prompt and click "OK". The graphical block instruction is added to the ladder diagram network.



- Enter parameters in the graphical block instruction to complete editing of the graphical block instruction.
  In the instruction, parameters (with "???") next to ① are mandatory, and parameters next to ② are optional. If a parameter is not used, the default parameter value is used automatically in the instruction input, and the state cannot be obtained in the instruction output in the program or during monitoring and debugging.

- All instructions under "Instruction Set" in the "Toolbox" pane are in graphical block mode. During programming, you can directly double-click an instruction under "Instruction Set" to add the instruction to the current focus position of the ladder diagram.



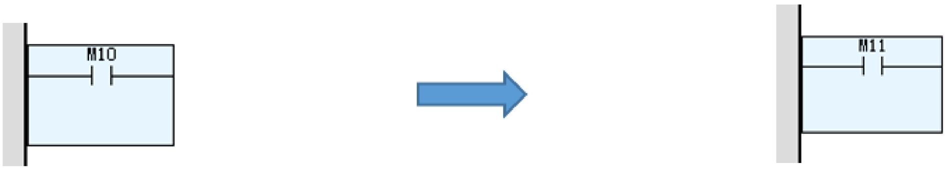
①: Double-click an instruction to add it to the ladder diagram. ②: The instruction is added successfully.

1.4.3 Labeling Function

Graphical blocks can be used to quickly increase or decrease label numbers and implement incremental paste.

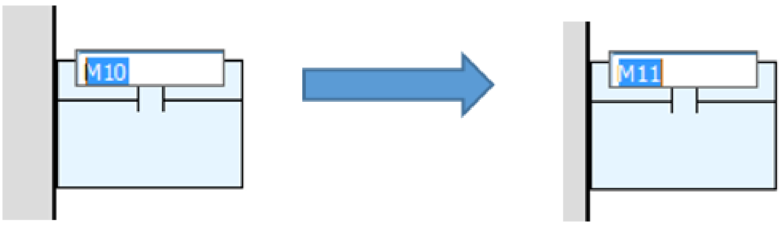
Quickly Increasing/Decreasing Label Numbers

When editing the ladder diagram, you can press "Alt"+"UP"/"DOWN" to quickly increase or decrease the label number of an element or array subscript.



Change “M10” to “M11” with keys Alt+UP after selecting “M10”.

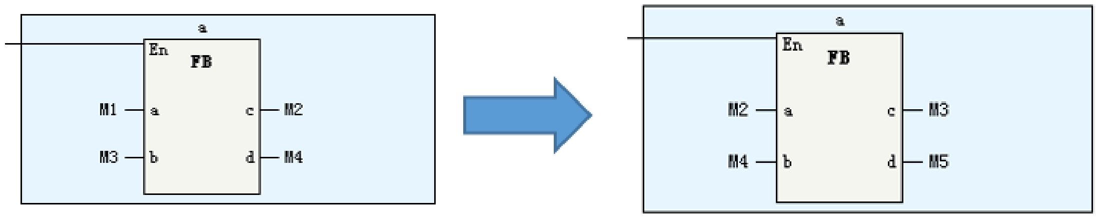
- This function can be used during command editing.



- For complex array variables, you can select the array subscript that needs to be increased or decreased.

Operate on the digit “7” through selecting the subscript of “A”.

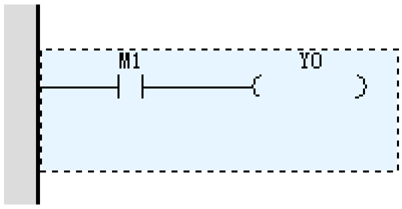
- When a function block is selected, the operation will be performed on all pins.



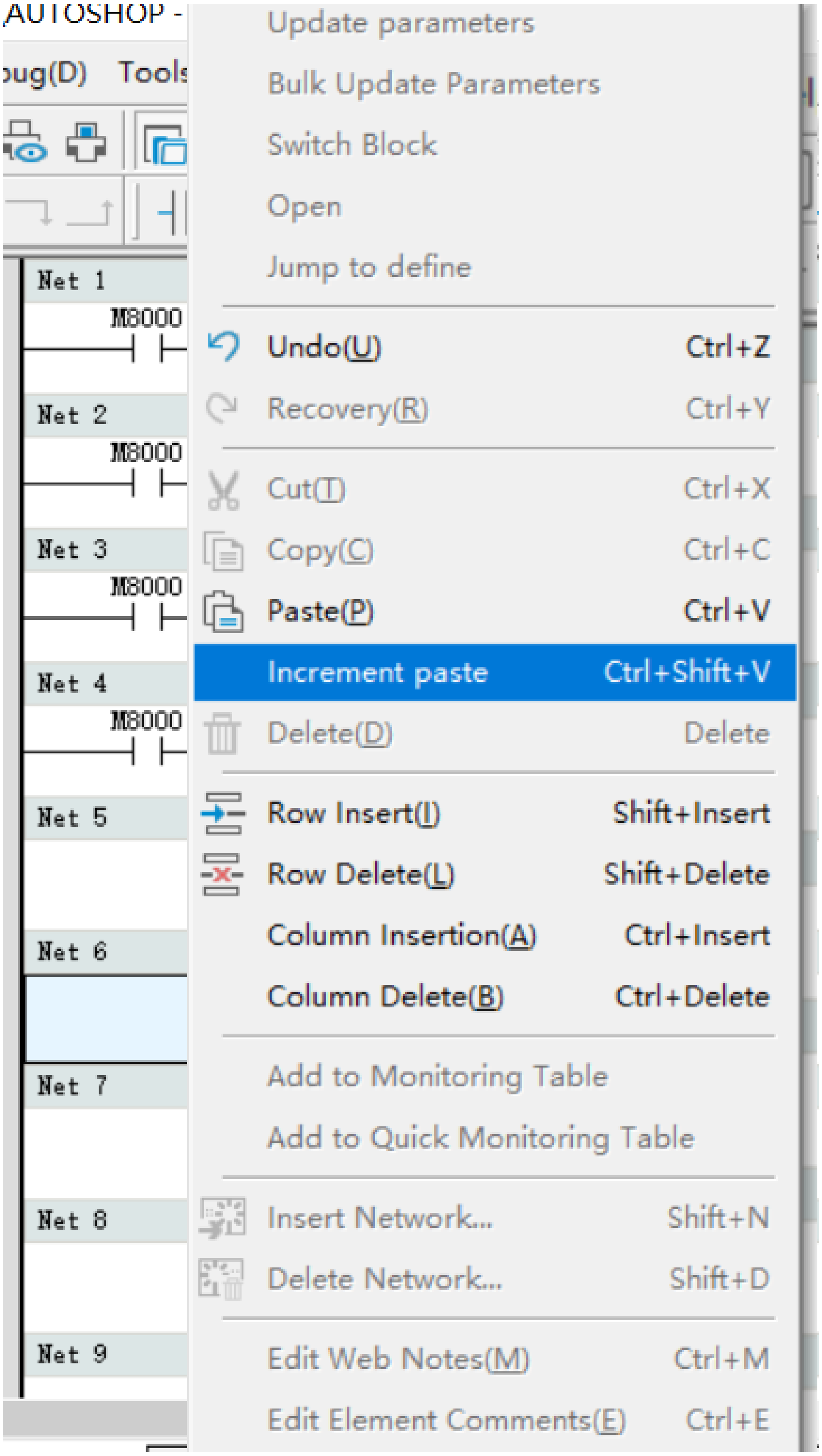
Incremental Paste

When editing the ladder diagram, you can use the incremental paste function to continuously paste the copied elements for multiple times. At the same time, the element number or array subscript can be specified during the process.

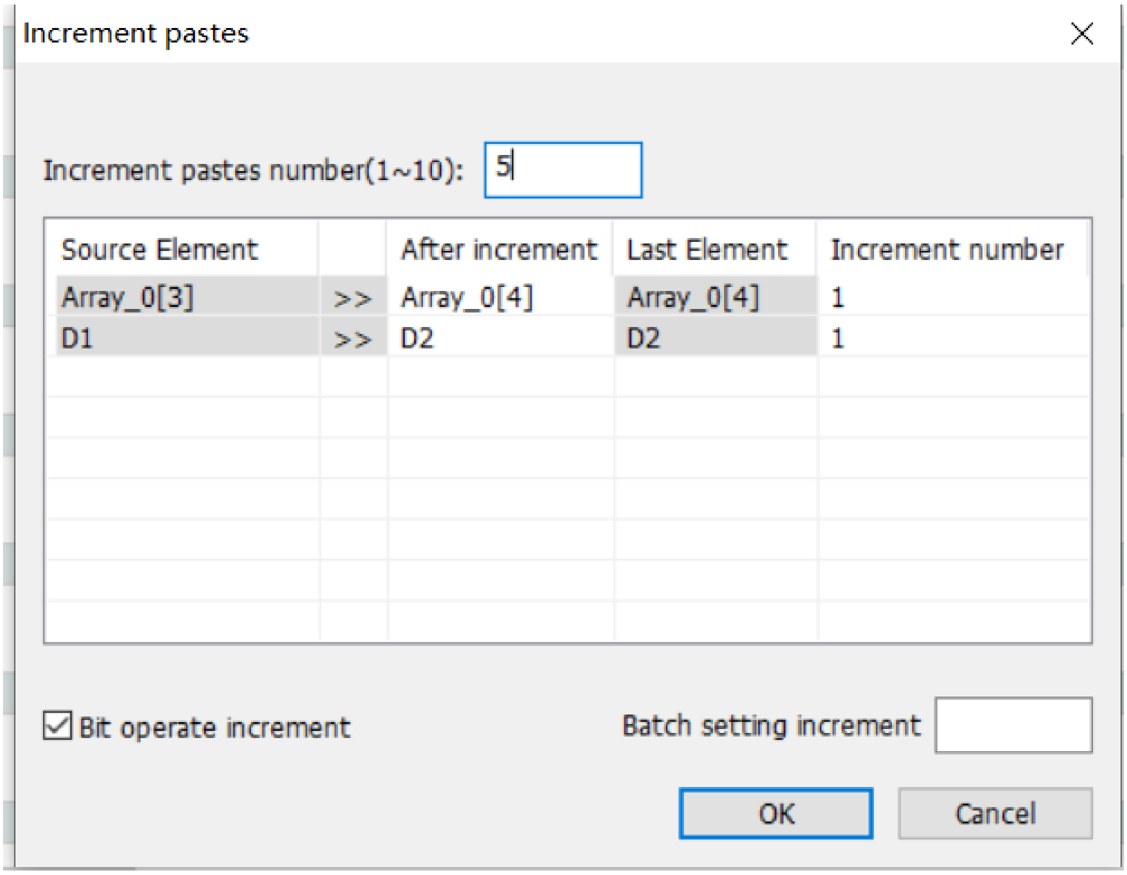
1. Select an element in the ladder diagram and press "Ctrl"+"C", or right-click the element and choose "Copy".



2. Right-click the destination position and choose "Increment paste" from the shortcut menu (or press "Ctrl"+"Shift"+"V").

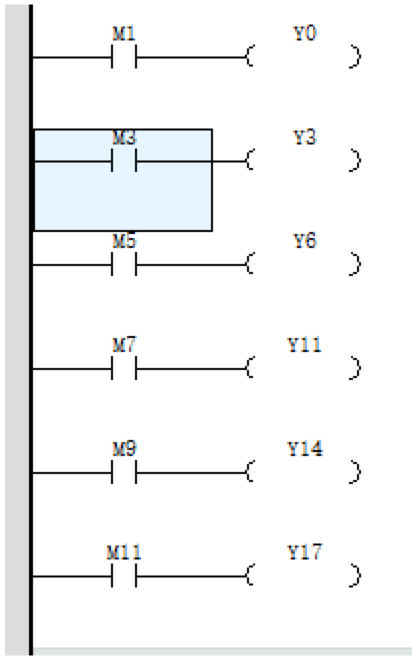


3. Specify the increment value and paste times in the displayed configuration window.



- "Incremental pastes number (1–10)": You can set the paste times.
- "After increment": You can enter the expected value after increment, and "Increment number" is automatically calculated based on this value.
- "Increment number": You can set the increment in the target element each time a paste operation is performed.
- "Bit operate increment": During bit operation of an element, if this option is selected, the increment applies to the bit operation of the target element.
- "Batch setting increment": You can set the increments in batches.

4. Click "OK". The paste operation is performed based on the configuration.



1.5 Function Blocks and Functions (FB/FC)

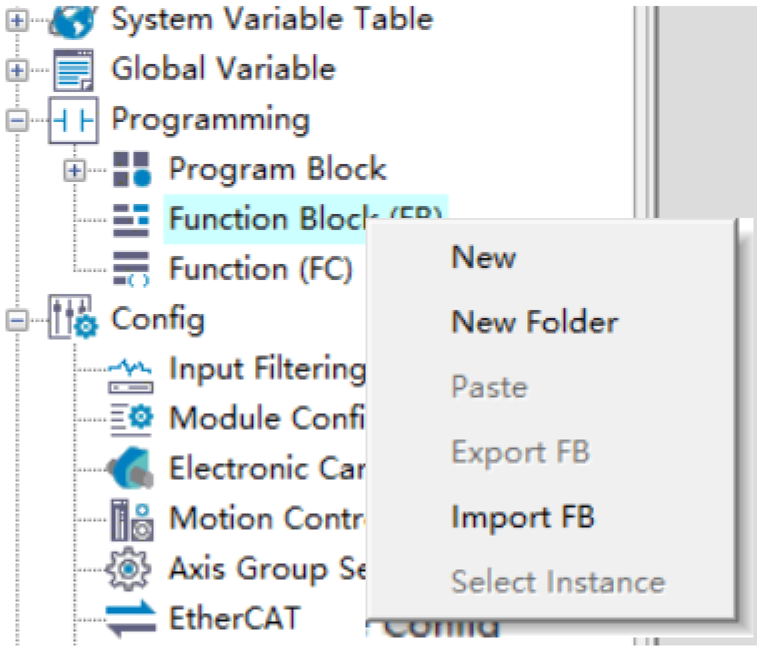
1.5.1 Function Blocks (FB)

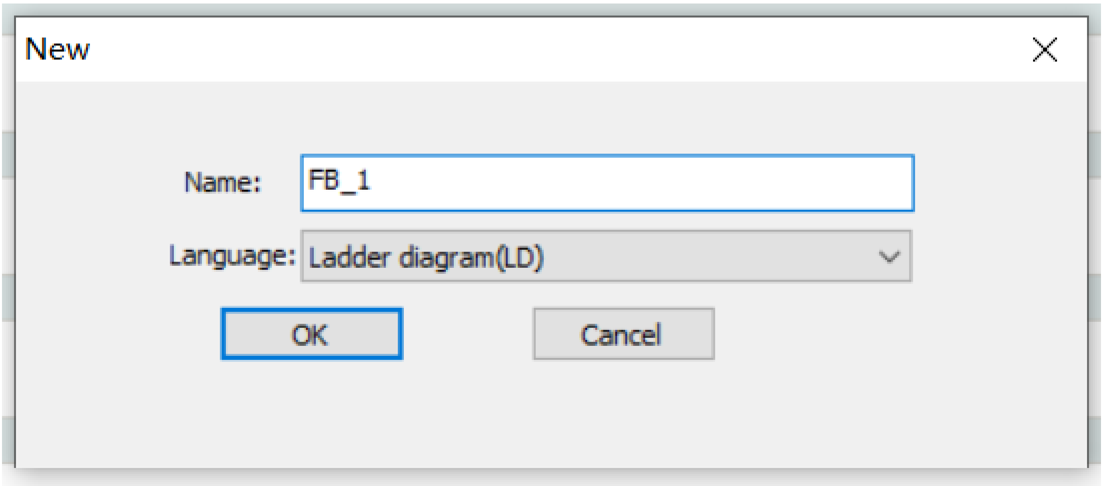
A function block (FB) abstractly encapsulates the part used repeatedly in a program into a general program block that can be called repeatedly within the program. Using encapsulated function blocks in programming can improve program development efficiency, reduce programming errors, and improve program quality.
Different instances can be created based on the same function block. These instances can output one or more values during execution. The system allocates memory for internal variables of each instance, and these variables describe the running state of the function block. With the same input parameters, different instances provide different calculation results.

The basic steps of using a function block are as follows: Create a function block -> Program the function block -> Instantiate the function block -> Run the function block -> Encapsulate the function block -> Import the function block.

Creating a Function Block

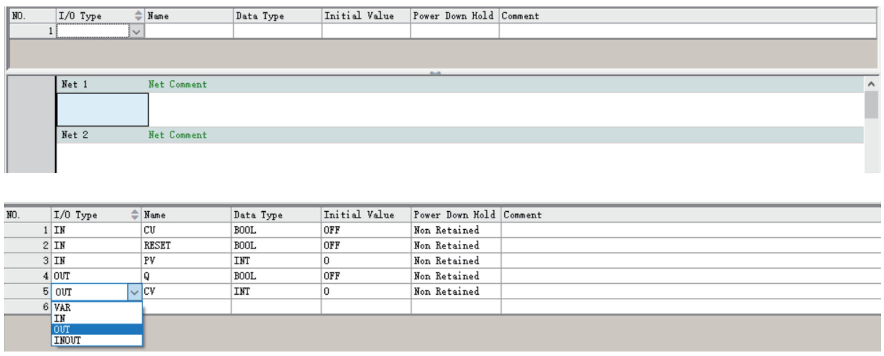
Expand the "Programming" node in the project management window, right-click "Function Block (FB)", or right-click a folder under "Function Block (FB)", choose "New", enter a name in the displayed dialog box, and click "OK". A function block is created successfully.





Programming the Function Block

Function blocks can be programmed in the ladder diagram or structured text. Double-click the created function block under "Function Block (FB)" to go to the function block program editing interface. Compared with ordinary program editing, the function block program editing interface has an additional input/output and local variable definition window.



①: Input/output and local variable definition window

1. "I/O Type": attribute of the function block variable

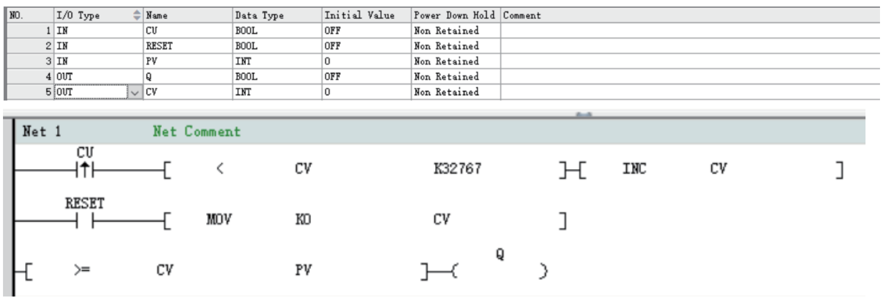
Variable Type	Type Description	Description
IN	Input variable	The parameter is provided by the logic block that calls the variable, and the input is transferred to the instruction of the logic block.
OUT	Output variable	The parameter is provided to the logic block that calls the variable, that is, structure data is output from the logic block.
INOUT	Input/Output variable	An input/output variable can not only be transferred to the called logic block, but also can be modified inside the called logic block.
VAR	Local variable	A local variable is only valid in the current logic block and cannot be accessed externally.

2. "Name": name of the variable
3. "Data Type"
   The supported data types include BOOL, INT, DINT, REAL, BYTE, IP, and STRING. You can also define array variables and structure variables. To use structure variables, you need to create structure members in the structure of global variables.
4. "Initial Value"
   You can set the initial value of a variable when execution starts.
5. "Power Down Hold"
   This attribute allows you to choose whether to retain the value of a variable upon power failure.
   - "Non Retained": The variable resumes the specified initial value after power-on.
   - "Retained": If you select "Re-initialize retentive variables when downloading", the variable resumes the specified initial value during program downloading; otherwise, it retains the previous value.

The function block program adopts ladder diagram programming. It can call functions (FC) or function blocks (FB) and supports up to 8 levels of nested calls.

In addition to variables, the function block program can also use supported elements, such as M8000, as global variables.

Example: Counting Up with FB Encapsulation

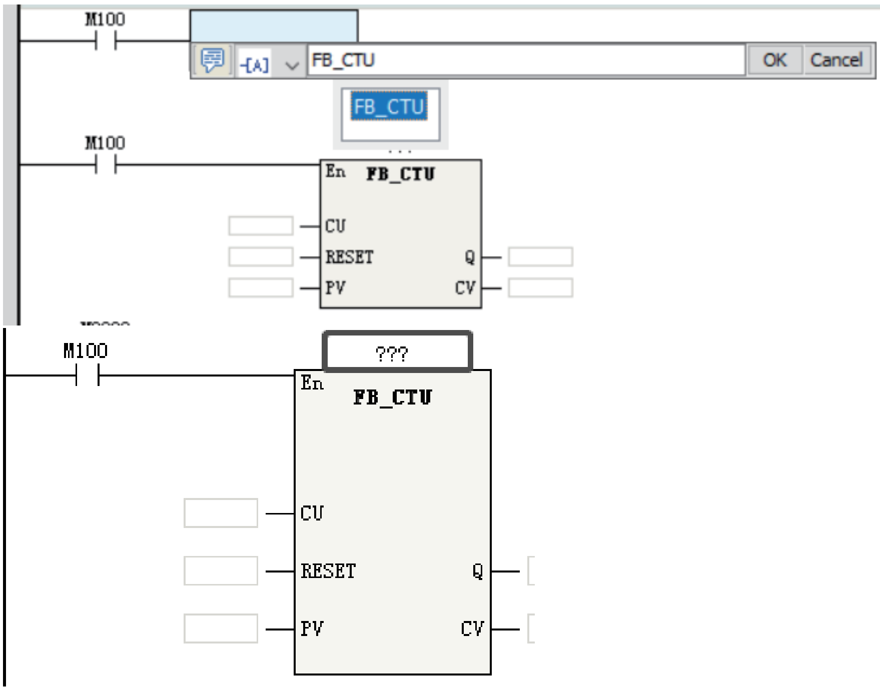


NO.	I/O Type	Name	Data Type	Initial Value	Power Down Hold	Comment
1	IN	CU	BOOL	OFF	Non Retained	
2	IN	RESET	BOOL	OFF	Non Retained	
3	IN	PV	INT	0	Non Retained	
4	OUT	Q	BOOL	OFF	Non Retained	
5	OUT	CV	INT	0	Non Retained	

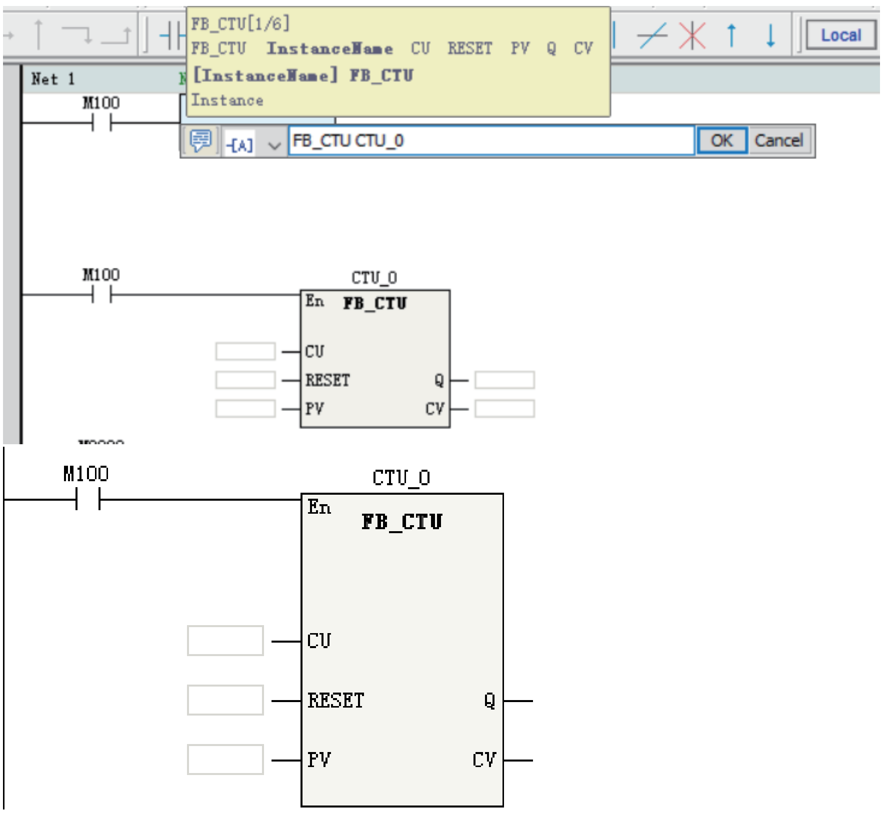
Instantiating and Calling the Function Block

After the FB program is compiled, the function block needs to be instantiated.

- Method 1: Directly enter the FB name in the ladder diagram application, and then enter the instance name in "???" at the top of the function block instruction to instantiate the function block.

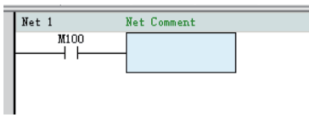


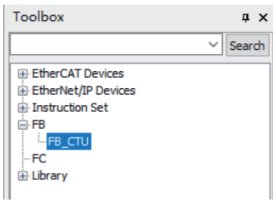
- Method 2: Directly enter the FB name+Instance name in the ladder diagram application and click "OK" to instantiate the function block.



After instantiation is completed, edit the instruction parameters in the FB instruction as required by the program to call the instantiated function block.

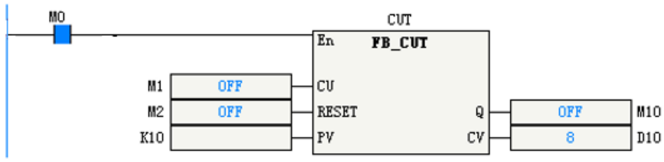
- Method 3: Double-click the FB instruction under "FB" of the "Toolbox" pane to add the FB instruction to the selected position in the ladder diagram. Then enter the instance name in the graphic block instruction to complete the instantiation definition.



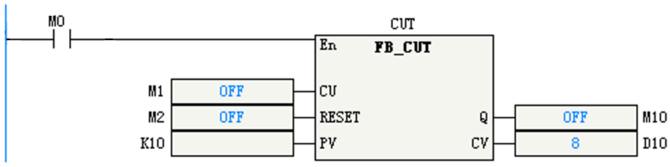


Running the Function Block

After the function block is instantiated, the En of the function block is connected to the ladder network. When the En network flow is ON, the function block program is executed, and the output of the function block changes with the input state and internal variable state. When the En network flow is OFF, the function block program is not executed, and output of the function block is not refreshed.



When the counter function block CUT flow is ON, the function block is executed. The output CV increases by 1 when the input condition CU changes on the rising edge.

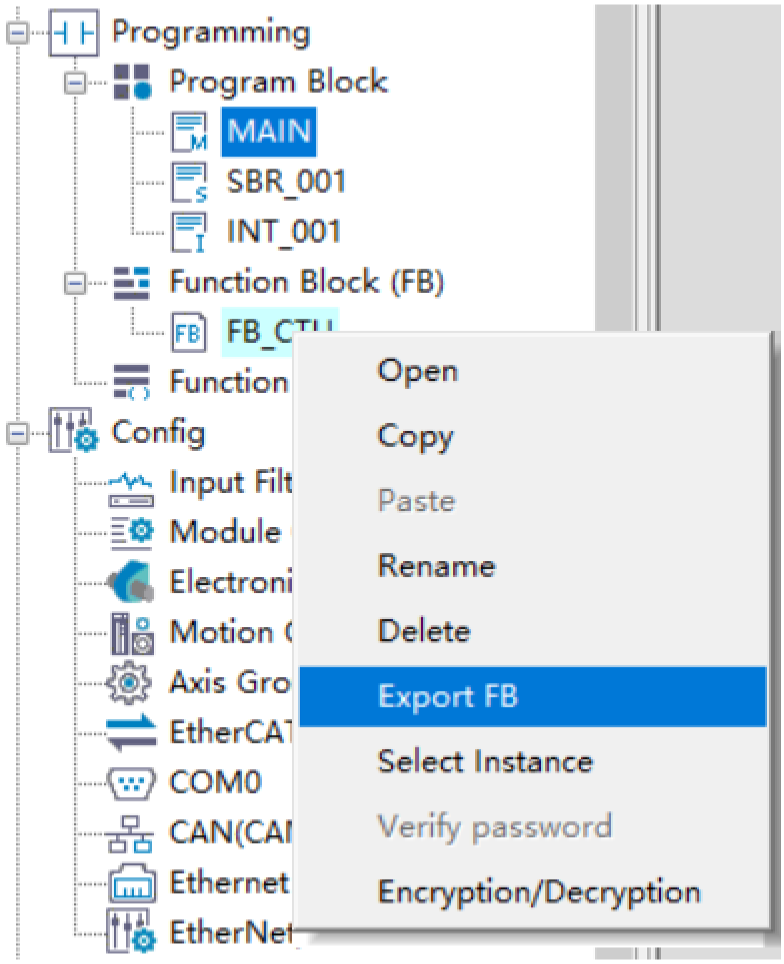


When the counter function block CUT flow is OFF, the function block is not executed. The output CV is not refreshed when the input condition CU changes on the rising edge.

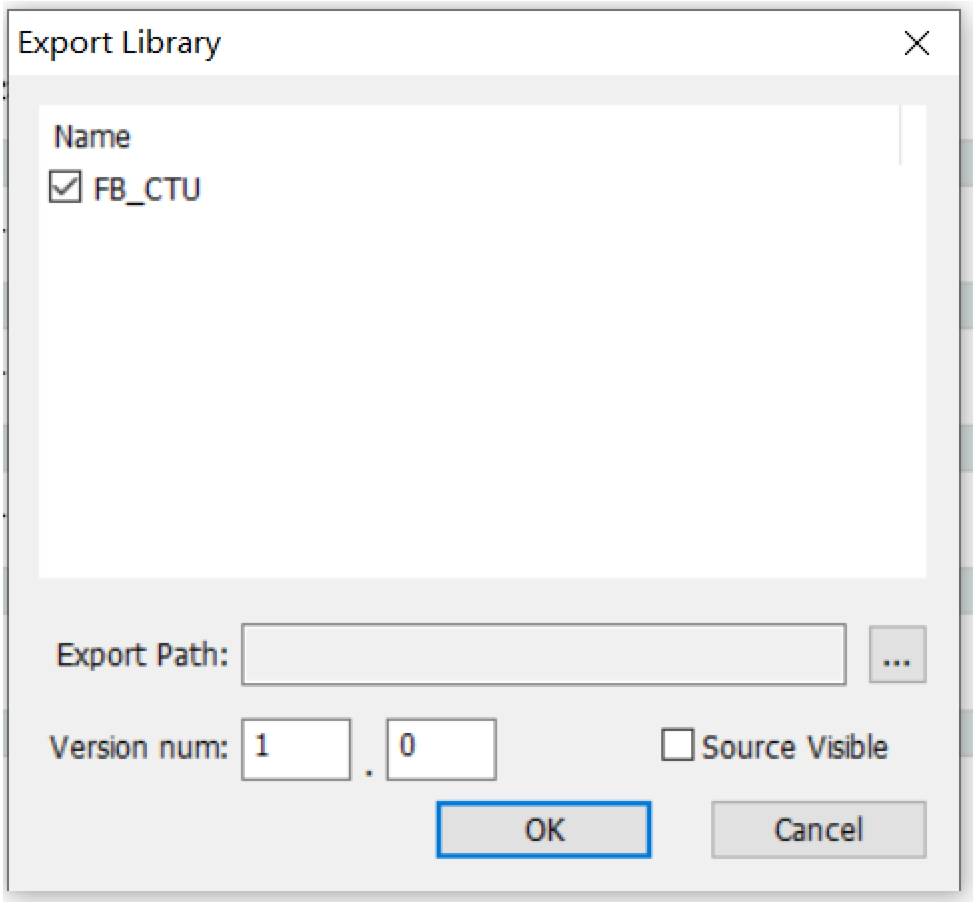
Encapsulating the Function Block

The function block can be encapsulated into a library after editing and debugging. The function block encapsulated into a library can be multiplexed in different programs through library management of AutoShop.

1. Right-click "Function Block (FB)" under "Programming" and choose "Export FB".



2. Select the function block to be encapsulated and set the version in the displayed "Export Library" window. Select "Source Visible" as required. If the source code is visible, after importing the library in the project, you can debug or modify the function block program. If the source code is invisible, after the library is imported, the function block program can only be called but not viewed or modified in the project.



3. Specify "Export Path" and click "OK". The FB is exported to the specified location, and a function block in .fe format is generated.

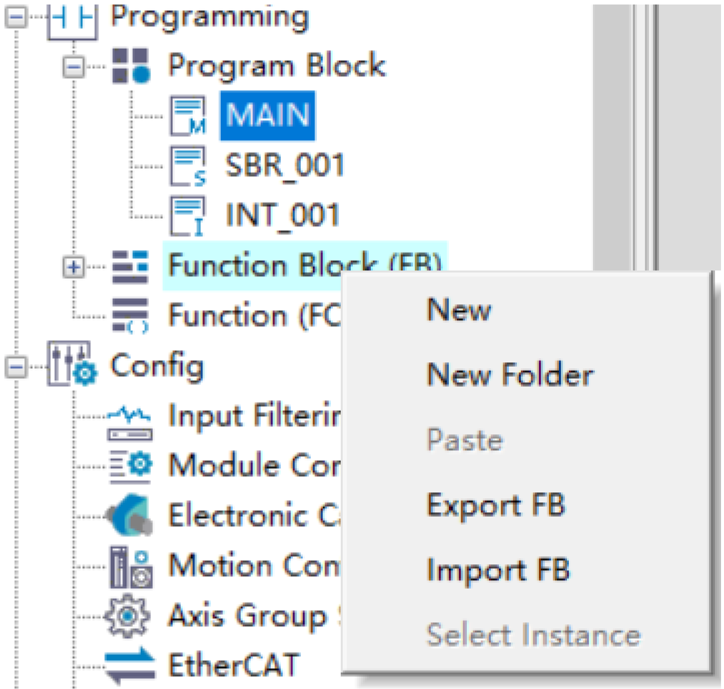
Caution

Encrypted function blocks, function blocks that call encrypted function blocks, and function blocks that call encrypted functions cannot be selected for export.

Importing the Function Block

After the function block is exported as a library, it can be called in other programs after being imported. You can import the function block library in either of the following two ways.

- Method 1: Right-click "Function Block (FB)" under "Programming" in the project management window and choose "Import FB" to import the library.



This method can only be used to import function blocks of which the source code is visible. After importing, you can double-click to open the function block program and edit and debug it. The function block library imported using this method is managed in the project. If you want to call the function block in a new project, you need to re-import the library.

- Method 2: Right-click "Library" in the "Toolbox" pane and choose "Import FB" to import the library. This method can be used to import function blocks of which the source code is visible or invisible. The libraries imported this way are managed as custom libraries, and the function blocks in the libraries can be used directly when a new project is created. You can double-click the function block library imported in the toolbox to directly add it to the ladder diagram program as an instruction. If you need to view or modify a function block program of which the source code is visible, you need to import it in the project management window.

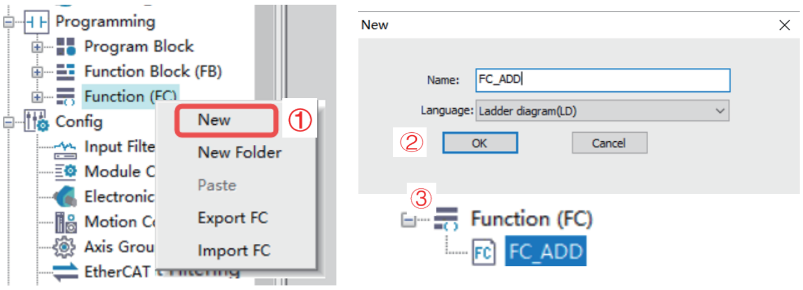
1.5.2 Functions (FC)

A function (FC) is an independently encapsulated program block. The program block can define input/output parameters and non-static internal variables. That is, when a function is called with the same input parameters, the output results are the same. An important feature of a function is that its internal variables are static, and there is no internal state storage. You will obtain the same output with the same input parameters. This is the main difference between a function and a function block.
FC, as a basic arithmetic unit, is often used in various mathematical operations. For example, sin(x) and sqrt(x) are typical functions.

The basic steps of using a function are as follows: Create a function -> Program the function -> Call the function -> Run the function -> Encapsulate the function.

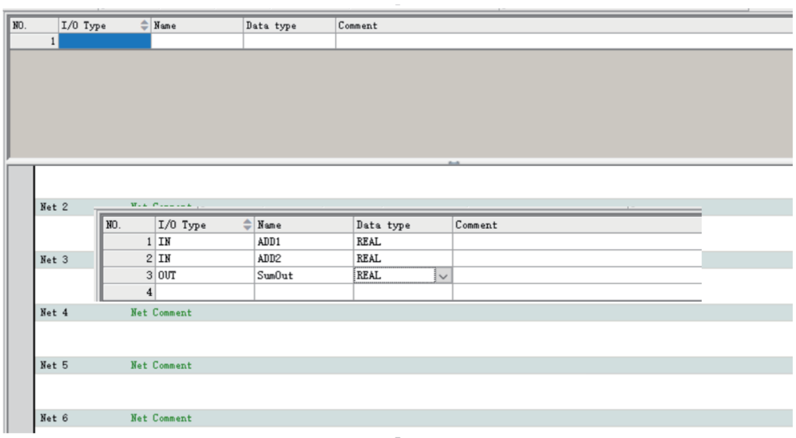
Creating a Function

Expand the "Programming" node in the project management window, right-click "Function (FC)", or right-click a folder under "Function (FC)", choose "New", enter a name in the displayed dialog box, and click "OK". A function is created successfully.



Programming the Function

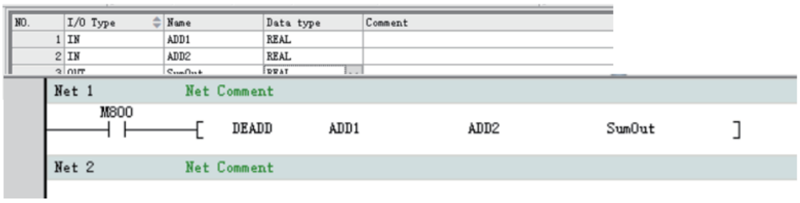
Functions can be programmed only in the ladder diagram. Double-click the created function under "Function (FC)" to go to the function program editing interface. The editing interface of the function program is similar to that of the function block. Compared with ordinary program editing, the function program editing interface has an additional input/output and local variable definition window.



In the input/output and local variable definition window, you can define the input (IN), output (OUT), input/output (INOUT), and local variable (VAR) of a function block. The supported data types include BOOL, INT, DINT, REAL, BYTE, IP, and STRING. You can also define array variables and structure variables. To use structure variables, you need to create structure members in the structure of global variables.

- Compared with variables of function blocks, variables of functions do not support configuration of initial values, and all local variables are non-retentive.
- The function program adopts ladder diagram programming. It can call functions. A function can be called by other functions, function blocks, and programs.
- In addition to variables, the function program can also use M8000 as an always ON variable.
- In a function program, instructions related to states or executed for multiple cycles, such as LDP and MC_Power, cannot be used.

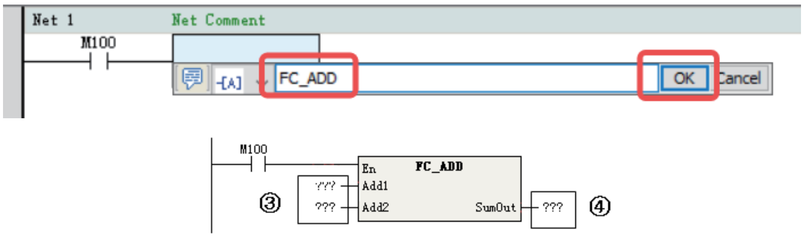
Example: Encapsulating the Addition Function



Calling the Function

The function program can be called directly or used in an application after it is compiled.

- Method 1: Directly enter the function name in the ladder diagram application, press "Enter", and then edit the input/output parameters in the graphic block instruction.

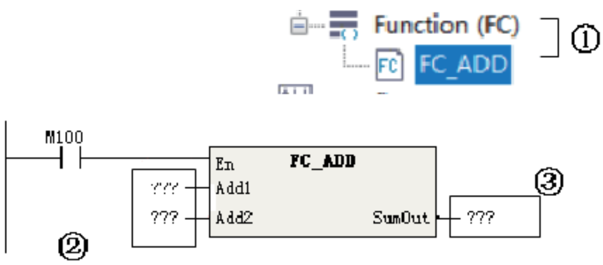


①: Enter the function name.

②: Click "OK".

③/④: Add input/output variables.

- Method 2: After a function program is created, the corresponding instruction is generated under "FC" in the "Toolbox" pane. Double-click the FC instruction under "FC" to add the FC instruction to the selected position in the ladder diagram.



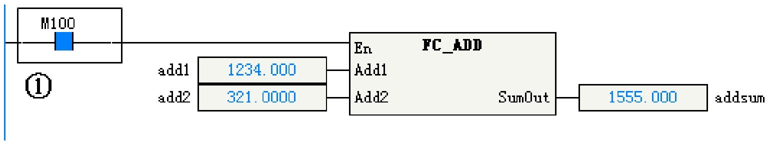
①: Double-click the FC instruction to add it.

②: Add input parameters.

③: Add output parameters.

Running the Function

After the function is called, the En of the function is connected to the ladder network. When the En network flow is ON, the function program is executed, and the output of the function is refreshed according to the input state operation. When the En network flow is OFF, the function program is not executed, and output of the function is not refreshed.



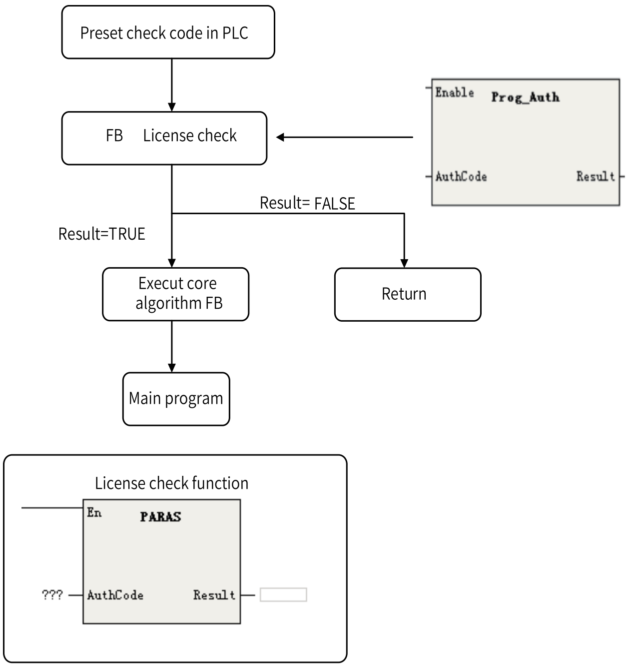
①: The function is executed when the En network flow is ON.

Encapsulating the Function

The encapsulation procedure of functions is similar to that of function blocks. For details, see the description of "Encapsulating the Function Block".

1.5.3 Authorization Function Block

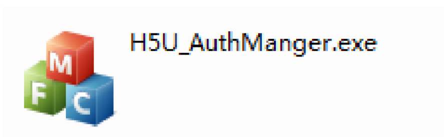
By using the Prog_Auth function, the core algorithm function block is controlled and compiled into a library file. Only authorized PLCs that pass the verification can use this library file, thus protecting the intellectual property of the equipment manufacturer.



1 - Check code
2 - Results returned (BOOL type); ON: Succeed; OFF: Failed

Setting Authorization Code

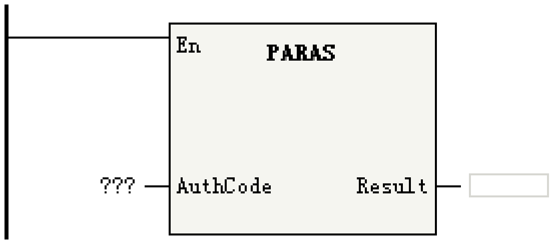
1. Run "H5U_AuthManger.exe" in the software installation directory.



2. Enter the IP address of the PLC, enter the 8-digit authorization code, and click "Set Authorization Code".
3. Click "Generate Verification Code". A string of characters is generated in the "Instruction Authorization Verification Code" text box.
4. You can also verify or clear the authorization code (only after you enter the authorization code) in the software.

Adding a Program Block

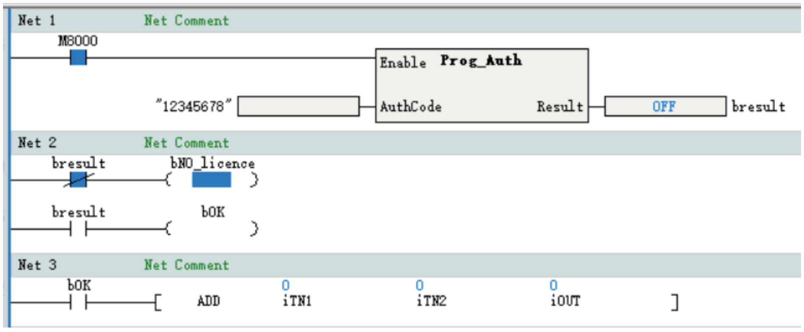
1. Open the function block to be authorized, and add the PARAS function block.



2. Enter the instruction authorization verification code generated by the software in "AuthCode".
3. The function block is authorized. If the authorization code of the PLC is inconsistent with that in the function block, the program in the function block cannot run.

Example

Since the verification code obtained by using Prog_Auth is inconsistent with the preset verification code in the PLC, the return value is "OFF", and the ADD instruction of the program is not executed.



1.5.4 FB Initial Values

The initial values of FB settings can be modified based on the FB type or FB instance.

- Modifying the initial values based on the FB type is equivalent to modifying the initial values of the type.
- Modifying the initial values based on an FB instance is equivalent to modifying the initial values of the instance.
- If the initial values of an instance are modified, the member variables of the FB instance display the values after modification, and the background color of the cells is yellow.
- If the initial values of an instance are not modified, the member variables of the FB instance display the default values, and the background color of the cells is white.

The initial values of the FB type are the default values of the instance. When the initial values of an instance are modified back to the default values, the background color of the cells changes from yellow to white.

Modifying Initial Values When the FB Is Not Nested

Modify the initial value of the FB type from 0 to 10. Use the default value as the initial value of the FB instance, that is, the initial value 10 of the FB type.

NO.	I/O Type	Name	Data Type	Initial Value	Power Down Hold	Comment
1	VAR	param_1	INT	10	Non Retained	
2						

NO.	I/O Type	Name	Data Type	Initial Value	Power Dow...	Comment
1	VAR	var_1	FB	...	Non Retained	
2						

Toolbox

Variable Name	Initial Value	Type	Comment
var_1		FB	
param_1	10	INT	

Modify the initial value of the FB instance from 10 to 100. The initial value of the FB instance is 100. At this time, if you attempt to modify the initial value of the FB type to 11, you will find that the initial value of the FB instance remains unchanged (still 100).

Toolbox

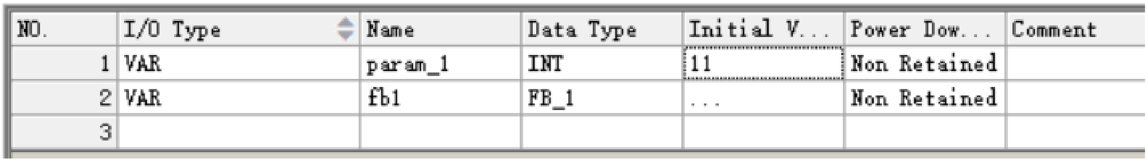
Variable Name	Initial Value	Type	Comment
var_1		FB	
param_1	100	INT	

In the ladder diagram, double-click "FB" to display the FB instance. At this time, the initial value of the FB instance is displayed in the FB view instead of the initial value of the FB type. If the initial value of the variable is modified to be inconsistent with the FB, the background color will be yellow. Modifying the initial value on this interface is the same as modifying the initial value of the instance in the function block instance table.

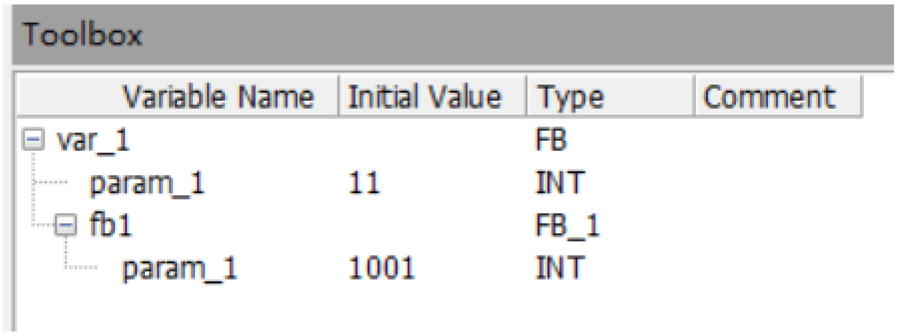
NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	100	Non Retained	
2						

Modifying Initial Values When the FB Is Nested

Add a variable fb1 in the FB type, and set the data type to "FB_1". Modify the initial value of FB_1 from 1000 to 1001. The member variable fb1 of the FB type automatically takes the default value 1001 as the initial value, and the member variable fb1 of the FB instance also automatically takes the default value 1001 as the initial value.

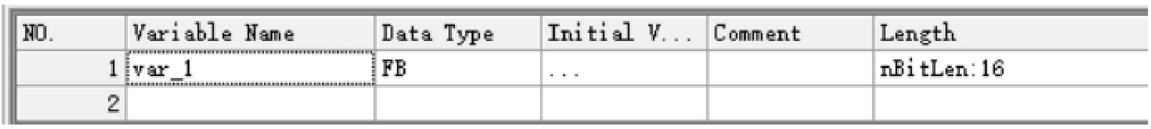


NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	11	Non Retained	
2	VAR	fb1	FB_1	...	Non Retained	
3						

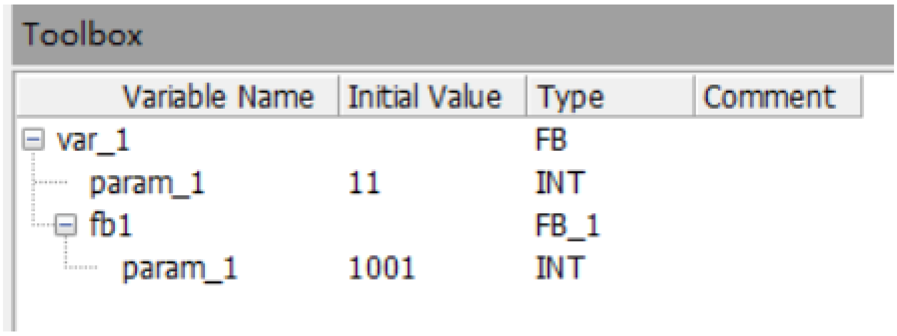


Toolbox

Variable Name	Initial Value	Type	Comment
var_1		FB	
param_1	11	INT	
fb1		FB_1	
param_1	1001	INT	



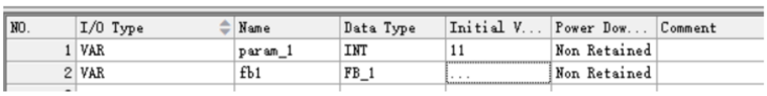
NO.	Variable Name	Data Type	Initial V...	Comment	Length
1	var_1	FB	...		nBitLen:16
2					



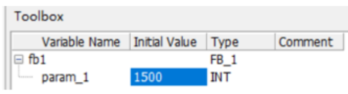
Toolbox

Variable Name	Initial Value	Type	Comment
var_1		FB	
param_1	11	INT	
fb1		FB_1	
param_1	1001	INT	

FB is the middle layer between the instance and FB_1. Modify the initial value of FB_1 to 1500 on the FB type interface. Then the initial value of the FB type changes to 1500, and the background color changes to yellow. At this time, the initial value of FB_1 of the FB instance is also 1500, but the background color is white, indicating that the default value is used.



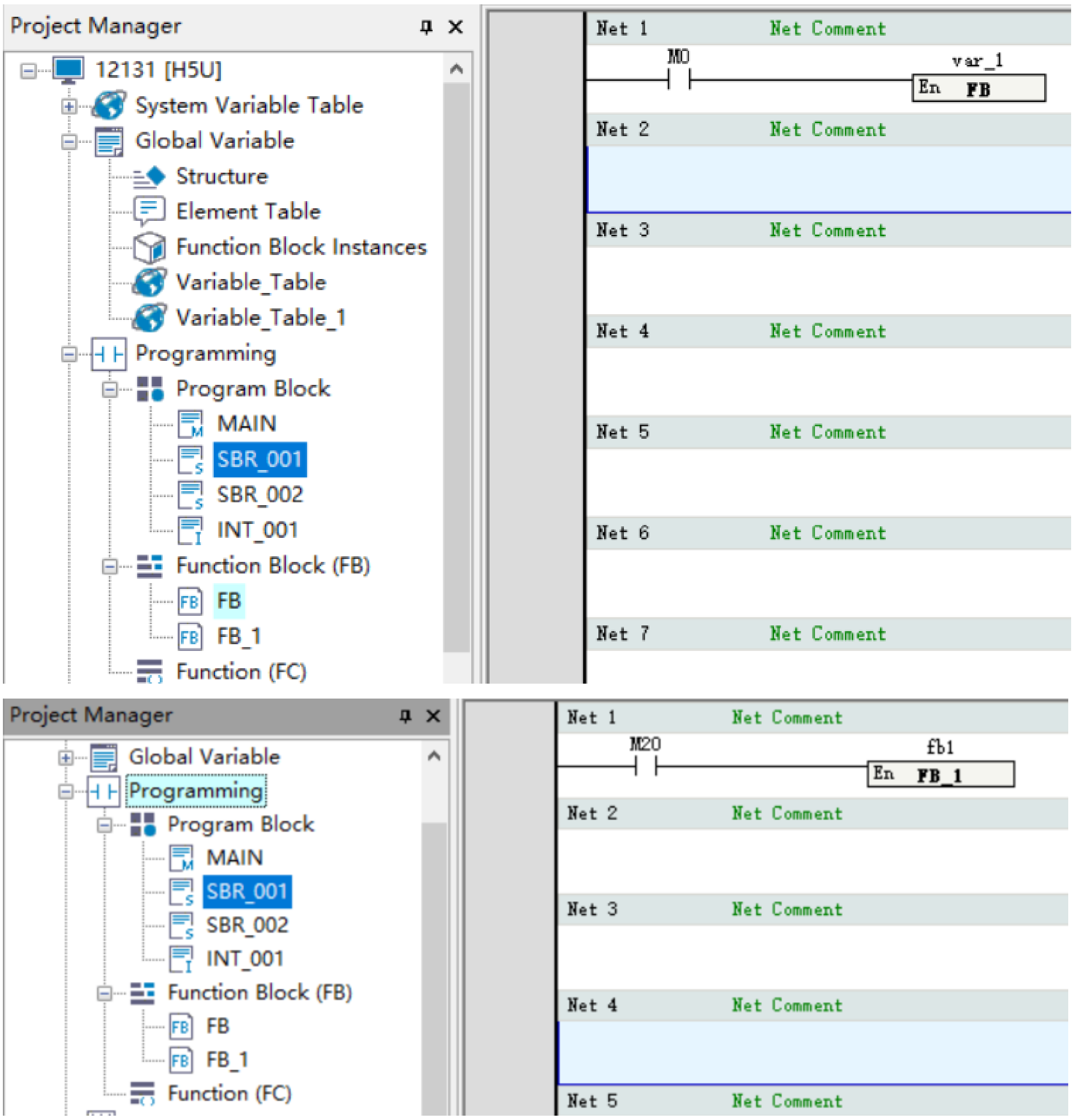
NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	11	Non Retained	
2	VAR	fb1	FB_1	...	Non Retained	



Toolbox

Variable Name	Initial Value	Type	Comment
fb1		FB_1	
param_1	1500	INT	

- Enter the instance interface from the main program. The initial value of the FB instance is displayed. Double-click "FB_1" to enter the FB_1 instance interface, and modify the initial value to 2000. Open the FB_1 type, and the initial value is still 1001. Open the FB instance FB_1, and the initial value is 2000.



At this time, the tab name is "FB_1(var_1.fb1)".

- Double-click "FB" in the "Project Manager" navigation tree. You can see that the initial value of FB_ 1 on the FB type interface is 1500. Double-click "FB_1" in the ladder diagram of the FB type interface to enter the FB_1 instance interface. You can see that the initial value of the FB_1 instance is 1500. Modify it to 2500. Then return to the FB type interface to check the initial value of FB_1. You will find that it also changes to 2500.

NO.	I/O Type	Name	Data Type	Initial Value	Power Dow...	Comment
1	VAR	param_1	INT	2500	Non Retained	

At this time, the tab name is "FB_1(fb1)".

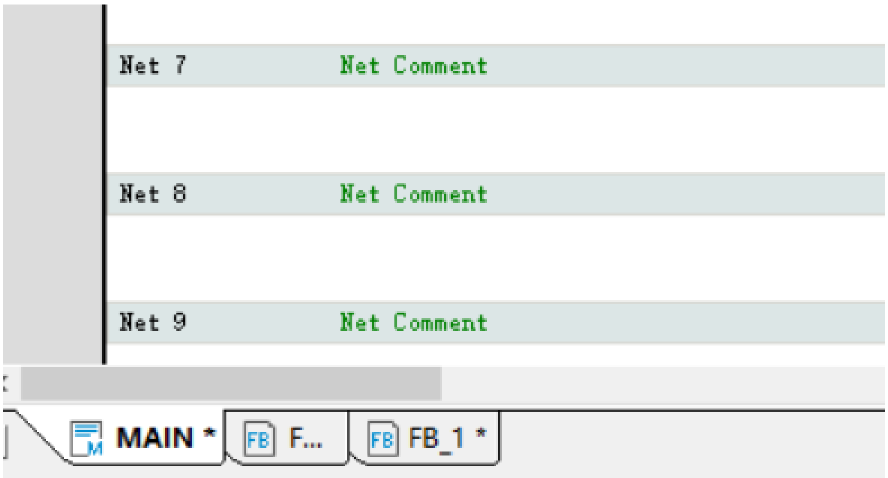
NO.	I/O Type	Name	Data Type	Initial V...	Power Dow...	Comment
1	VAR	param_1	INT	11	Non Retained	
2	VAR	fb1	FB_1	...	Non Retained	

Toolbox

Variable Name	Initial Value	Type	Comment
fb1		FB_1	
param_1	2500	INT	

Tab at the Bottom of the FB View

The tab displayed at the bottom of the FB view contains the following information from left to right: node name, instance name, and unsaved flag. The node name is the name of the project tree node, and the instance name refers to the instance name in parentheses. The following figures show the details.



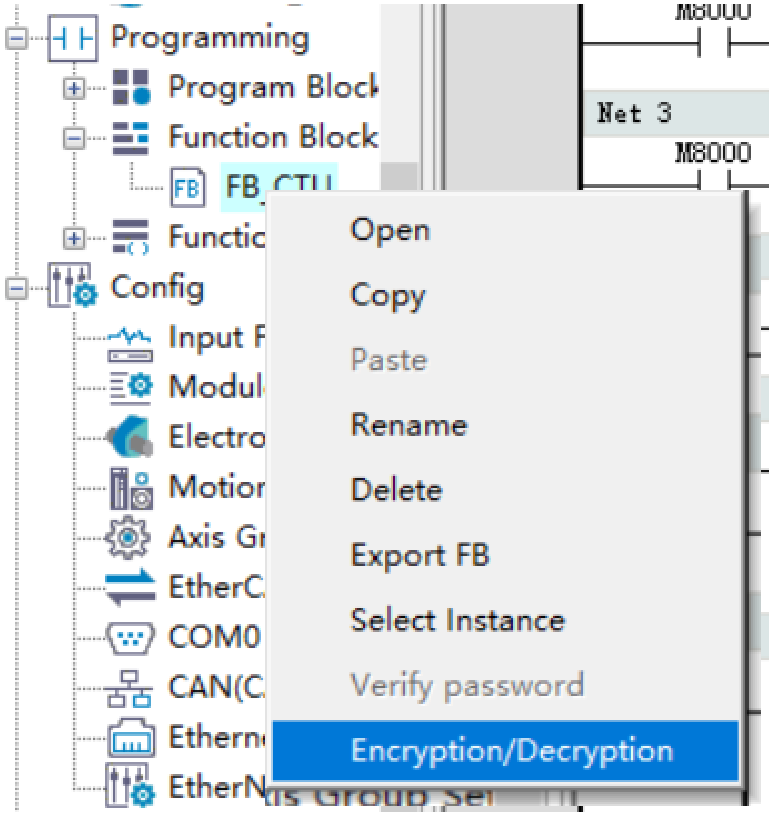
As shown in the preceding figure, "FB" is the node name, "var_1.fb1" is the instance name, and "*" indicates unsaved.

Since the tab needs to be parsed, characters including the period (.), asterisk (*), and parentheses (()) are not allowed when FBs and structures are renamed.

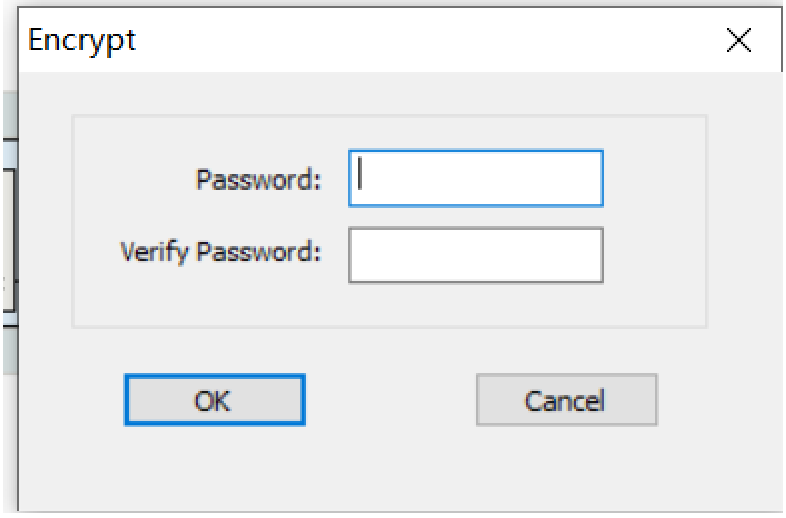
1.5.5 Encrypting FB or FC

This section takes encryption of function blocks as an example. The process is similar for encrypting functions. After encryption, the method of calling the function blocks or functions remains unchanged.

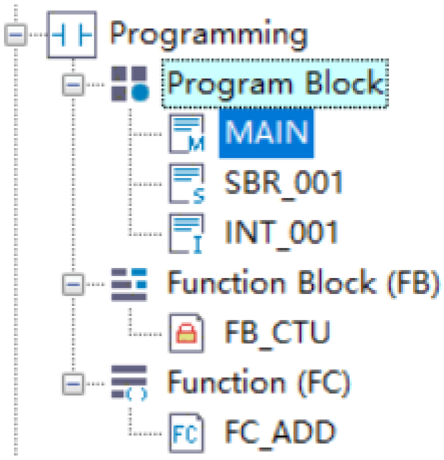
1. Choose "Programming" > "Function Block (FB)" in the project management window, right-click "FB", and choose "Encryption/Decryption".



2. Enter and confirm the password in the displayed "Encrypt" dialog box.

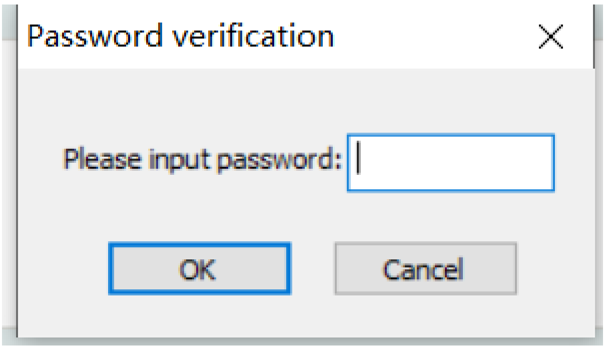


The following figure shows a function block after encryption.



Performing the preceding steps on an encrypted function block will decrypt it and restore it to its original unencrypted state.

To access an encrypted function block, you can double-click the encrypted node, or right-click the encrypted node and choose "Password verification" from the shortcut menu, and enter the correct password in the displayed dialog box.



2 Instruction List

2.1 LD & LiteST Instructions

All the instructions supported by the PLC are summarized and classified by function as follows.

Instruction Category	Instruction	Function Description	Language Support
Contact instruction	LD	Load NO contact	LD
	LDI	Load NC contact	LD
	AND	Serial connection of NO contacts	LD
	ANI	Serial connection of NC contacts	LD
	OR	Parallel connection of NO contacts	LD
	ORI	Parallel connection of NC contacts	LD
	LDP	Obtain pulse rising edge	LD
	LDF	Obtain pulse falling edge	LD
	ANDP	Serial connection of pulse rising edge	LD
	ANDF	Serial connection of pulse falling edge	LD
	ORP	Parallel connection of pulse rising edge	LD
	ORF	Parallel connection of pulse falling edge	LD
	MEP	Conversion of operation result to rising edge pulse	LD
	MEF	Conversion of operation result to falling edge pulse	LD
Output control instruction	OUT	Coil drive	LD
	SET	SET action storage coil instruction	LD
	RST	Contact or cache clearing	LD
	ZSET	Batch data setting	LD and LiteST
	ZRST	Batch data reset	LD and LiteST
	PLS	Pulse rising edge detection coil instruction	LD
	PLF	Pulse falling edge detection coil instruction	LD
	ALT	Alternate output	LD
	R_TRIG	Rising edge detection	LD and LiteST
	F_TRIG	Falling edge detection	LD and LiteST
Flow control instruction	INV	Operation result inversion	LD

Instruction Category	Instruction	Function Description	Language Support
Process control instruction	CJ	Conditional jump	LD
	LBL	Label	LD
	CALL	Call subprogram	LD
	SSRET	Conditional subprogram return	LD
	EI	Enable interrupt	LD and LiteST
	DI	Disable interrupt	LD and LiteST
	WDT	Watchdog timer reset	LD
	FOR	Start of a loop	LD
	NEXT	End of a loop	LD
SFC instruction	STL	Program jump to secondary bus	LD
	RET	Program return to primary bus	LD
	OUTSTL	Output program jump to secondary bus	LD
	SETSTL	Setting program jump to secondary bus	LD
	RSTSTL	Resetting program jump to secondary bus	LD
Contact comparison instruction	LD=	LD contact comparison equal to	LD
	LD>	LD contact comparison greater than	LD
	LD<	LD contact comparison less than	LD
	LD<>	LD contact comparison not equal to	LD
	LD>=	LD contact comparison greater than or equal to	LD
	LD<=	LD contact comparison less than or equal to	LD
	AND=	AND contact comparison equal to	LD
	AND>	AND contact comparison greater than	LD
	AND<	AND contact comparison less than	LD
	AND<>	AND contact comparison not equal to	LD
	AND>=	AND contact comparison greater than or equal to	LD
	AND<=	AND contact comparison less than or equal to	LD
	OR=	OR contact comparison equal to	LD
	OR>	OR contact comparison greater than	LD
	OR<	OR contact comparison less than	LD
	OR<>	OR contact comparison not equal to	LD

Instruction Category	Instruction	Function Description	Language Support
(Continued) Contact comparison instruction	OR>=	OR contact comparison greater than or equal to	LD
	OR<=	OR contact comparison less than or equal to	LD
	LD&	LD logical AND operation	LD
	LD|	LD logical OR operation	LD
	LD^	LD logical XOR operation	LD
	AND&	AND logical AND operation	LD
	AND|	AND logical OR operation	LD
	AND^	AND logical XOR operation	LD
	OR&	OR logical AND operation	LD
	OR|	OR logical OR operation	LD
	OR^	OR logical XOR operation	LD
	FLDD>	State contact of floating-point comparison >, conductive when S1 > S2	LD
	FLDD>=	State contact of floating-point comparison >=, conductive when S1 ⩾ S2	LD
	FLDD<	State contact of floating-point comparison <, conductive when S1 < S2	LD
(Continued) Contact comparison instruction	FLDD<=	State contact of floating-point comparison <=, conductive when S1 ⩽ S2	LD
	FLDD=	State contact of floating-point comparison =, conductive when S1 = S2	LD
	FLDD<>	State contact of floating-point comparison <>, conductive when S1 ≠ S2	LD
	FANDD>	AND state contact of floating-point comparison >, conductive when S1 > S2	LD
	FANDD>=	AND state contact of floating-point comparison >=, conductive when S1 ⩾ S2	LD
	FANDD<	AND state contact of floating-point comparison <, conductive when S1 < S2	LD
	FANDD<=	AND state contact of floating-point comparison <=, conductive when S1 ⩽ S2	LD
	FANDD=	AND state contact of floating-point comparison =, conductive when S1 = S2	LD
	FANDD<>	AND state contact of floating-point comparison <>, conductive when S1 ≠ S2	LD

Instruction Category	Instruction	Function Description	Language Support
Contact Comparison Instructions	FORD>	OR state contact of floating-point comparison >, conductive when S1 > S2	LD
	FORD>=	OR state contact of floating-point comparison >=, conductive when S1 ⩾ S2	LD
	FORD<	OR state contact of floating-point comparison <, conductive when S1 < S2	LD
	FORD<=	OR state contact of floating-point comparison <=, conductive when S1 ⩽ S2	LD
	FORD=	OR state contact of floating-point comparison =, conductive when S1 = S2	LD
	FORD<>	OR state contact of floating-point comparison <>, conductive when S1 ≠ S2	LD
	LDZ>	State contact of absolute value comparison >, conductive when |S1 – S2| > |S3|	LD
	LDZ>=	State contact of absolute value comparison >=, conductive when |S1 – S2| ⩾ |S3|	LD
	LDZ<	State contact of absolute value comparison <, conductive when |S1 – S2| < |S3|	LD
	LDZ<=	State contact of absolute value comparison <=, conductive when |S1 – S2| ⩽ |S3|	LD
	LDZ=	State contact of absolute value comparison =, conductive when |S1 – S2| = |S3|	LD
	LDZ<>	State contact of absolute value comparison <>, conductive when |S1 – S2| ≠ |S3|	LD
	ANDZ>	AND state contact of absolute value comparison >, conductive when |S1 – S2| > |S3|	LD

Instruction Category	Instruction	Function Description	Language Support
(Continued) Contact Comparison Instructions	ANDZ>=	AND state contact of absolute value comparison >=, conductive when $\lvert S1 - S2 \rvert \geqslant \lvert S3 \rvert$	LD
	ANDZ<	AND state contact of absolute value comparison <, conductive when $\lvert S1 - S2 \rvert < \lvert S3 \rvert$	LD
	ANDZ<=	AND state contact of absolute value comparison <=, conductive when $\lvert S1 - S2 \rvert \leqslant \lvert S3 \rvert$	LD
	ANDZ=	AND state contact of absolute value comparison =, conductive when $\lvert S1 - S2 \rvert = \lvert S3 \rvert$	LD
	ANDZ<>	AND state contact of absolute value comparison <>, conductive when $\lvert S1 - S2 \rvert \neq \lvert S3 \rvert$	LD
	ORZ>	OR state contact of absolute value comparison >, conductive when $\lvert S1 - S2 \rvert > \lvert S3 \rvert$	LD
	ORZ>=	OR state contact of absolute value comparison >=, conductive when $\lvert S1 - S2 \rvert \geqslant \lvert S3 \rvert$	LD
	ORZ<	OR state contact of absolute value comparison <, conductive when $\lvert S1 - S2 \rvert < \lvert S3 \rvert$	LD
	ORZ<=	OR state contact of absolute value comparison <=, conductive when $\lvert S1 - S2 \rvert \leqslant \lvert S3 \rvert$	LD
	ORZ=	OR state contact of absolute value comparison =, conductive when $\lvert S1 - S2 \rvert = \lvert S3 \rvert$	LD
	ORZ<>	OR state contact of absolute value comparison <>, conductive when $\lvert S1 - S2 \rvert \neq \lvert S3 \rvert$	LD
Arithmetic Operation Instructions	ADD	Binary data addition	LD
	SUB	Binary data subtraction	LD
	MUL	Binary data multiplication	LD
	DIV	Binary data division	LD
	MOD	Remainder by binary data division	LD and LiteST
	EADD	Binary floating-point addition	LD
	ESUB	Binary floating-point subtraction	LD
	EMUL	Binary floating-point multiplication	LD
	EDIV	Binary floating-point division	LD
	INC	Binary data increment by 1	LD
	DEC	Binary data decrement by 1	LD

Instruction Category	Instruction	Function Description	Language Support
Data logical operation instruction	WAND	Binary data logical AND	LD
	WOR	Binary data logical OR	LD
	WXOR	Binary data logical XOR	LD
	NEG	Binary data negation	LD
	ENEG	Binary floating-point sign negation	LD
Word bit operation instruction	BLD	Word or dword bit contact instruction	LD
	BLDI	Word or dword bit inversion contact instruction	LD
	BAND	Word or dword bit AND contact instruction	LD
	BANDI	Word or dword bit AND inversion contact instruction	LD
	BOR	Word or dword bit OR contact instruction	LD
	BORI	Word or dword bit OR inversion contact instruction	LD
	BOUT	Word or dword bit data output instruction	LD
	BSET	Word or dword bit data setting instruction	LD
	BRST	Word or dword bit data reset instruction	LD
Trigonometric function instruction	SIN	Floating-point SIN operation	LD
	COS	Floating-point COS operation	LD
	TAN	Floating-point TAN operation	LD
	ASIN	Binary floating-point ARCSIN operation	LD
	ACOS	Binary floating-point ARCCOS operation	LD
	ATAN	Binary floating-point ARCTAN operation	LD
	RAD	Binary floating-point degree-to-radian conversion	LD
	DEG	Binary floating-point radian-to-degree conversion	LD
	SINH	Binary floating-point SINH operation	LD
	COSH	Binary floating-point COSH operation	LD
	TANH	Binary floating-point TANH operation	LD
Table operation instruction	WSUM	Data sum calculation	LD
	MEAN	Mean calculation	LD
	LIMIT	Upper/Lower limit control	LD
	BZAND	Dead zone control	LD
	ZONE	Zone control	LD
	SCL	Coordinate determination (coordinate data of different points)	LD
	SCL2	Coordinate determination 2 (X and Y coordinates)	LD

Instruction Category	Instruction	Function Description	Language Support
Exponent operation instruction	EXP	Binary floating-point exponentiation operation	LD
	LOGE	Binary floating-point natural logarithm operation	LD
	LOG	Binary floating-point common logarithm operation	LD
	ESQR	Binary floating-point square root operation	LD
	SQR	Binary data square root operation	LD
	POW	Floating-point weight instruction	LD
Data conversion instruction	INT	Conversion from binary floating-point number into BIN integer	LD
	BCD	Conversion from binary into BCD	LD
	BIN	Conversion from BCD into binary	LD
	FLT	Conversion from binary into binary floating-point	LD
	EBCD	Conversion from binary floating-point into decimal floating-point	LD
	EBIN	Conversion from decimal floating-point into binary floating-point	LD
	DABIN	Conversion from decimal ASCII into BIN	LD
	BINDA	Conversion from BIN into decimal ASCII	LD
	WTOB	Conversion from word to byte	LD
	BITW	Conversion from bit to word	LD and LiteST
	BTOW	Conversion from byte to word	LD
	WBIT	Conversion from word to bit	LD and LiteST
(Continued Data conversion instruction	WTODW	Conversion from word to dword	LD
	DWTOW	Conversion from dword to word	LD
	MCPY	Data copy (memory copy, type conversion)	LD and LiteST
	MSET	Data setting (memory setting and reset)	LD and LiteST
	UNI	4-bit combination of 16-bit data	LD
	DIS	4-bit separation of 16-bit data	LD
	ASCI	Conversion from HEX into ASCII	LD
	HEX	Conversion from ASCII into HEX	LD
	DECO	Data decoding	LD
	ENCO	Data encoding	LD

Instruction Category	Instruction	Function Description	Language Support
Data transfer instruction	MOV	Move	LD
	EMOV	Binary floating-point move	LD
	SMOV	Shift move	LD
	BMOV	Batch move	LD
	FMOV	Multi-point move	LD
	CML	Complement	LD
	CMP	Comparison	LD
	ECMP	Floating-point comparison	LD
	ZCP	Zone comparison	LD
	EZCP	Floating-point zone comparison	LD
	SORTR	Data sorting	LD
	SORTC	Data sorting 2	LD
	SER	Data search	LD
	FDEL	Deletion of data from data table	LD
	FINS	Insertion of data to data table	LD
	POP	Last-in data read	LD
	RAMP	Ramp instruction	LD
Data shift instruction	ROR	Rotation right	LD
	ROL	Rotation left	LD
	RCR	Rotation right with carry	LD
	RCL	Rotation left with carry	LD
	SFTR	Bit shift right	LD
	SFTL	Bit shift left	LD
	WSFR	Word shift right	LD
	WSFL	Word shift left	LD
	SFWR	Shift write (FIFO)	LD
	SFRD	Shift read (FIFO)	LD
	SFR	Bit shift right with carry	LD
	SFL	Bit shift left with carry	LD
Other data processing instruction	SWAP	Byte swap	LD
	BON	Bit state check	LD
	SUM	Sum of ON bits	LD
	RAND	Random number generation within limits	LD
	XCH	Data exchange	LD
	ABS	Absolute value of integer	LD
	EABS	Absolute value of floating-point number	LD
	EFMOV	Multi-point floating-point move	LD
	CCD	Check code	LD
	CRC	CRC code calculation	LD
	LRC	LRC code calculation	LD

Instruction Category	Instruction	Function Description	Language Support
Matrix operation instruction	BK+	Block data addition	LD
	BK–	Block data subtraction	LD
	MAND	Matrix AND	LD
	MOR	Matrix OR	LD
	MXOR	Matrix XOR	LD
	MXNR	Matrix XNOR	LD
	MINV	Matrix inversion	LD
Matrix comparison instruction	BKCMP=	Matrix comparison equal to (S1 = S2)	LD
	BKCMP>	Matrix comparison greater than (S1 > S2)	LD
	BKCMP<	Matrix comparison less than (S1 < S2)	LD
	BKCMP<>	Matrix comparison not equal to (S1 ≠ S2)	LD
	BKCMP<=	Matrix comparison less than or equal to (S1 ⩽ S2)	LD
	BKCMP>=	Matrix comparison greater than or equal to (S1 ⩾ S2)	LD
String instruction	STR	Conversion from integer into string	LD
	STRMOV	String assignment	LD
	VAL	Conversion from string into integer	LD
	ESTR	Conversion from binary floating-point into string	LD
	EVAL	Conversion from string into binary floating-point	LD
	$ADD	Character string linking	LD
	LEN	Character string length detection	LD
	INSTR	Character string search	LD
	RIGHT	String data extraction from the right	LD
	LEFT	String data extraction from the left	LD
	MIDR	Random extraction of character string	LD
	MIDW	Random replacement of character string	LD
	$MOV	Character string transfer	LD
Clock instruction	TCMP	Clock data comparison	LD
	TZCP	Clock data zone comparison	LD
	TADD	Clock data addition	LD
	TSUB	Clock data subtraction	LD
	HTOS	Conversion from hour-minute-second into second	LD
	STOH	Conversion from second into hour-minute-second	LD
	TRD	Clock data read	LD
	TWR	Clock data write	LD
	HOUR	Hour meter	LD

Instruction Category	Instruction	Function Description	Language Support
High-speed counter instruction (H5U)	HC_Counter	High-speed counter enable	LD and LiteST
	HC_Preset	High-speed counter preset	LD and LiteST
	HC_TouchProbe	Probe	LD and LiteST
	HC_Compare	High-speed counter comparison	LD and LiteST
	HC_ ArrayCompare	High-speed counter array comparison	LD and LiteST
	HC_ StepCompare	High-speed counter step comparison	LD and LiteST
Bus encoder axis instruction (H5U)	ENC_Counter	Encoder enable	LD and LiteST
	ENC_Reset	Encoder reset	LD and LiteST
	ENC_Preset	Encoder preset	LD and LiteST
	ENC_ TouchProbe	Encoder probe	LD and LiteST
	ENC_ ArrayCompare	Encoder one-dimensional array comparison	LD and LiteST
	ENC_ StepCompare	Encoder one-dimensional step comparison	LD and LiteST
	ENC_ GroupArray-Compare	Encoder two-dimensional array comparison	LD and LiteST
	ENC_ ReadStatus	Encoder state read	LD and LiteST
	ENC_ DigitalOutput	Encoder DO control	LD and LiteST
	ENC_ ResetCompare	Encoder comparison output reset	LD and LiteST
	ENC_SetUnit	Gear ratio setting	LD and LiteST
	ENC_ SetLineRota-tionMode	Rotation mode setting	LD and LiteST

Instruction Category	Instruction	Function Description	Language Support
Encoder axis instruction (Easy)	ENC_Counter	Encoder enable	LD and LiteST
	ENC_Reset	Encoder reset	LD and LiteST
	ENC_Preset	Encoder preset	LD and LiteST
	ENC_ TouchProbe	Encoder probe	LD and LiteST
	ENC_ ArrayCompare	Encoder one-dimensional array comparison	LD and LiteST
	ENC_ StepCompare	Encoder one-dimensional step comparison	LD and LiteST
	ENC_Compare	Single-point comparison output	LD
	ENC_ GroupArray-Compare	Encoder two-dimensional array comparison	LD and LiteST
	ENC_ ReadStatus	Encoder state read	LD and LiteST
	ENC_ DigitalOutput	Encoder DO control	LD and LiteST
	ENC_ ResetCompare	Encoder comparison output reset	LD and LiteST
	ENC_SetUnit	Gear ratio setting	LD and LiteST
	ENC_ SetLineRota-tionMode	Rotation mode setting	LD and LiteST
Timer instruction	TPR	Pulse timer	LD and LiteST
	TONR	On-delay timer	LD and LiteST
	TOFR	Off-delay timer	LD and LiteST
	TACR	Accumulating timer	LD and LiteST
Pointer instruction	PTGET	Pointer variable assignment	LD
	PTINC	Pointer variable address incremented by 1	LD
	PTDEC	Pointer variable address decremented by 1	LD
	PTADD	Pointer variable address addition	LD
	PTSUB	Pointer variable address subtraction	LD
	PTSET	Pointer variable assignment	LD
	PTMOV	Pointer variable mutual assignment	LD
	PTLD>	Pointer variable contact comparison greater than	LD
	PTLD>=	Pointer variable contact comparison greater than or equal to	LD
	PTLD<=	Pointer variable contact comparison less than or equal to	LD
	PTLD=	Pointer variable contact comparison equal to	LD
	PTLD<>	Pointer variable contact comparison not equal to	LD

Instruction Category	Instruction	Function Description	Language Support
(Continued) Pointer instruction	PTAND>	Pointer variable AND contact comparison greater than	LD
	PTAND>=	Pointer variable AND contact comparison greater than or equal to	LD
	PTAND<	Pointer variable AND contact comparison less than	LD
	PTAND<=	Pointer variable AND contact comparison less than or equal to	LD
	PTAND=	Pointer variable AND contact comparison equal to	LD
	PTAND<>	Pointer variable AND contact comparison not equal to	LD
	PTOR>	Pointer variable OR contact comparison greater than	LD
	PTOR>=	Pointer variable OR contact comparison greater than or equal to	LD
	PTOR<	Pointer variable OR contact comparison less than	LD
	PTOR<=	Pointer variable OR contact comparison less than or equal to	LD
	PTOR=	Pointer variable OR contact comparison equal to	LD
	PTOR<>	Pointer variable OR contact comparison not equal to	LD
FB/FC instruction	PROG_AUTH	Program block (FB/FC) authorization verification	LD
Communication protocol instruction	SerialSR	Serial port free protocol transmission and reception	LD
	SerialSend	Serial port free protocol transmission	LD
	SerialRcv	Serial port free protocol reception	LD
	MB_Master	Transmission and reception of serial Modbus protocol	LD
	MB_Client	Transmission and reception of the Modbus TCP protocol	LD
	TCP_Listen	TCP listening	LD
	TCP_Accept	TCP connection request accept	LD
	TCP_Connect	TCP connection request initiation	LD
	TCP_Close	TCP connection close	LD
	TCP_Send	TCP data transmission	LD
	TCP_Receive	TCP data reception	LD
	UDP_Bind	UDP socket binding	LD
	UDP_Send	UDP data transmission	LD

Instruction Category	Instruction	Function Description	Language Support
(Continued Communication protocol instruction	UDP_Receive	UDP data reception	LD
	ETC_ ReadParameter_CoE	Reading SDO parameters of ETC_ RestartMaster slave	LD and LiteST
	ETC_ WriteParameter_CoE	Writing SDO parameters of ETC_ RestartMaster slave	LD and LiteST
	ETC_ RestartMaster	Restarting EtherCAT master	LD and LiteST
	EIP_Generic_ Service	Calling the "Generic" service	LD
	EIP_Get_ Attributes_All	Calling the "Get_Attributes_All" service	LD
	EIP_Get_ Attribute_Single	Calling the "Get_Attribute_Single" service	LD
	EIP_Set_ Attributes_All	Calling the "Set_Attributes_All" service	LD
	EIP_Set_ Attribute_Single	Calling the "Set_Attribute_Single" service	LD
	EIP_Apply_ Attributes	Calling the "Apply_Attributes" service	LD
	EIP_NOP	Calling the "NOP" service	LD
	EIP_Reset	Calling the "Reset" service	LD
	EIP_Start	Calling the "Start" service	LD
	EIP_Stop	Calling the "Stop" service	LD
EtherCAT/ Local high-speed pulse output motion control axis instruction	MC_Power	Axis enable control	LD and LiteST
	MC_Reset	Fault reset	LD and LiteST
	MC_ReadStatus	Axis state read	LD and LiteST
	MC_ ReadAxisError	Axis error read	LD and LiteST
	MC_ ReadDigitalInput	Digital input read	LD and LiteST
	MC_ ReadActualPosition	Current position read	LD and LiteST
	MC_ ReadActualVelocity	Current velocity read	LD and LiteST
	MC_ ReadActualTorque	Current torque read	LD and LiteST
	MC_SetPosition	Current position setting	LD and LiteST
	MC_TouchProbe	Probe	LD and LiteST
	MC_ MoveRelative	Relative positioning	LD and LiteST
	MC_ MoveAbsolute	Absolute positioning	LD and LiteST
	MC_ MoveVelocity	Velocity control	LD and LiteST

Instruction Category	Instruction	Function Description	Language Support
(Continued) EtherCAT/ Local high-speed pulse output motion control axis instruction	MC_Jog	Jogging	LD and LiteST
	MC_ TorqueControl	Torque control	LD and LiteST
	MC_Home	Homing	LD and LiteST
	MC_Stop	Axis stop	LD and LiteST
	MC_Halt	Axis halt	LD and LiteST
	MC_MoveFeed	Interrupt positioning	LD and LiteST
	MC_MoveBuffer	Multi-position positioning	LD and LiteST
	MC_ ImmediateStop	Immediate stop	LD and LiteST
	MC_ MoveSuperImposed	Motion superimposition	LD and LiteST
	MC_ MoveVelocityCSV	CSV-based velocity control with adjustable pulse width	LD and LiteST
	MC_ SyncMoveVelocity	CSV-based synchronous velocity control with adjustable pulse width	LD and LiteST
	MC_ FollowVelocity	CSP-based synchronous velocity control	LD and LiteST
	MC_ SyncTorqueControl	Synchronous torque control	LD and LiteST
	MC_ SetAxisConfigPara	Axis parameter configuration	LD and LiteST
Electronic cam instruction	MC_CamIn	Start cam operation	LD and LiteST
	MC_CamOut	End cam operation	LD and LiteST
	MC_ GetCamTablePhase	Obtain cam table phase	LD and LiteST
	MC_ GetCamTableDistance	Obtain cam table displacement	LD and LiteST
	MC_ DigitalCamSwitch	Electronic cam tappet control	LD and LiteST
	MC_GearIn	Start gear operation	LD and LiteST
	MC_GearOut	End gear operation	LD and LiteST
	MC_Phasing	Master axis phase shifting	LD and LiteST
	MC_ SaveCamTable	Save cam table	LD and LiteST
	MC_ GenerateCamTable	Update cam table	LD and LiteST
	MC_GearInPos	Start the gear operation at the specified position	LD and LiteST

Instruction Category	Instruction	Function Description	Language Support
Axis group control instruction	MC_MoveLinear	Linear interpolation	LD and LiteST
	MC_MoveCircular	Circular interpolation	LD and LiteST
	MC_MoveEllipse	Elliptical interpolation	LD and LiteST
	MC_GroupStop	Stop axis group operation	LD and LiteST
	MC_GroupPause	Pause axis group operation	LD and LiteST
CANopen motion control axis instruction	MC_Power_CO	Enable servo axis through communication	LD
	MC_Reset_CO	Reset servo axis fault through communication	LD
	MC_ReadActualPosition_CO	Read current position of axis through communication	LD
	MC_ReadActualVelocity_CO	Read current velocity of axis through communication	LD
	MC_Halt_CO	Stop servo axis through communication (can be aborted)	LD
	MC_Stop_CO	Stop servo axis through communication (cannot be aborted)	LD
	MC_MoveAbsolute_CO	Control absolute positioning of axis through communication	LD
	MC_MoveRelative_CO	Control relative positioning of axis through communication	LD
	MC_MoveVelocity_CO	Control axis velocity through communication	LD
	MC_Jog_CO	Control axis jogging through communication	LD
	MC_Home_CO	Control axis homing through communication	LD
	MC_WriteParameter_CO	Write axis parameters through communication	LD
	MC_ReadParameter_CO	Read axis parameters through communication	LD
Other instructions	PID	PID calculation	LD

2.2 LiteST Instructions

Instruction Category	Instruction	Function Description
Trigonometric function	SIN	Sine operation instruction
	COS	Cosine operation instruction
	TAN	Tangent operation instruction
	ASIN	Arcsine operation instruction
	ACOS	Arccosine operation instruction
	ATAN	Arctangent operation instruction
Exponent operation instruction	LOG	Base-10 logarithm
	LN	Base-e (2.71828) logarithm
	SQRT	Square root operation instruction
	EXPT	Power operation instruction
Explicit conversion instruction	INT_TO_<TYPE>	Convert the INT type into the type specified by <TYPE>.
	DINT_TO_<TYPE>	Convert the DINT type into the type specified by <TYPE>.
	BOOL_TO_<TYPE>	Convert the BOOL type into the type specified by <TYPE>.
	REAL_TO_<TYPE>	Convert the REAL type into the type specified by <TYPE>.
	BYTE_TO_<TYPE>	Convert the BYTE type into the type specified by <TYPE>.
	TO_<TYPE>	Convert the variable into the type specified by <TYPE>.
Comparison instruction	MAX	Max operation
	MIN	Min operation
Shift instruction	SHL	Shift left operation
	SHR	Shift right operation
Binary operation instruction	SEL	Binary operation
Absolute value operation instruction	ABS	Absolute value operation
Bit operation instruction	AND	AND operation
	OR	OR operation
	XOR	XOR operation
	NOT	NOT operation

3 Instruction Description (LD & LiteST)

3.1 Program Logic Instructions

3.1.1 Contact Instructions

3.1.1.1 Instruction List

The following table lists the contact instructions.

Instruction Category	Instruction	Function
Contact instruction	LD	Load NO contact
	LDI	Load NC contact
	AND	Serial connection of NO contacts
	ANI	Serial connection of NC contacts
	OR	Parallel connection of NO contacts
	ORI	Parallel connection of NC contacts
	LDP	Obtain pulse rising edge
	LDF	Obtain pulse falling edge
	ANDP	Serial connection of pulse rising edge
	ANDF	Serial connection of pulse falling edge
	ORP	Parallel connection of pulse rising edge
	ORF	Parallel connection of pulse falling edge
	MEP	Conversion of operation result to rising edge pulse
	MEF	Conversion of operation result to falling edge pulse

3.1.1.2 LD&LDI&LDP&LDF

LD – Load NO contact

LDI – Load NC contact

LDP – Obtain pulse rising edge

LDF – Obtain pulse falling edge

16-bit instruction	LD: Continuous execution
32-bit instruction	-
16-bit instruction	LDI: Continuous execution
32-bit instruction	-
16-bit instruction	LDP: Continuous execution
32-bit instruction	-
16-bit instruction	LDF: Continuous execution

32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

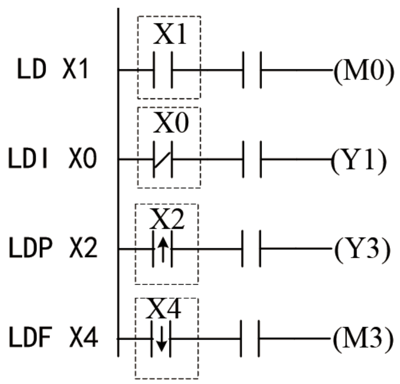
Table 3–1 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The LD, LDI, LDP, and LDF instructions are used for contacts starting from the left bus.

- The LDP instruction is used to detect the rising edge of a contact signal. If rising transition is detected in the signal, the contact becomes active, and it becomes inactive upon the next scan operation.
- The LDF instruction is used to detect the falling edge of a contact signal. If falling transition is detected in the signal, the contact becomes active, and it becomes inactive upon the next scan operation.



3.1.1.3 AND&ANDI&ANDP&ANDF

AND – Serial connection of NO contacts

ANDI – Serial connection of NC contacts

ANDP – Serial connection of pulse rising edge

ANDF – Serial connection of pulse falling edge

16-bit instruction	AND: Continuous execution
32-bit instruction	-
16-bit instruction	ANDI: Continuous execution
32-bit instruction	-

16-bit instruction	ANDP: Continuous execution			
32-bit instruction	-			
16-bit instruction	ANDF: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

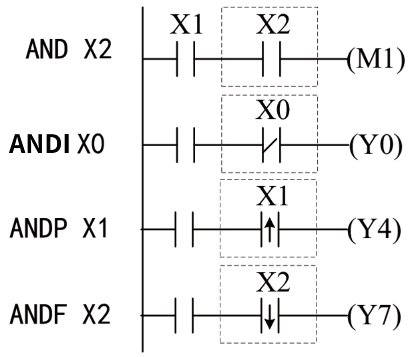
Table 3–2 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The AND, ANI, ANDP, and ANDF instructions are used for state operations of serial contacts. These instructions are used to read the state of the designated serial contact and perform an AND operation on the contact state and the contact's logical operation result. The AND result is stored in the accumulator.

- The ANDP instruction is used to obtain the rising edge transition state of the contact for an AND operation.
- The ANDP instruction is used to obtain the falling edge transition state of the contact for an AND operation.



3.1.1.4 OR&ORI&ORP&ORF

OR – Parallel connection of NO contacts

ORI – Parallel connection of NC contacts

ORP – Parallel connection of pulse rising edge

ORF – Parallel connection of pulse falling edge

16-bit instruction	OR: Continuous execution
32-bit instruction	-
16-bit instruction	ORI: Continuous execution
32-bit instruction	-
16-bit instruction	ORP: Continuous execution
32-bit instruction	-
16-bit instruction	ORF: Continuous execution
32-bit instruction	-

Operand	Name	Description	Range	Data Type
S	Bit element	Element or variable of which the flow state is to be determined	-	BOOL

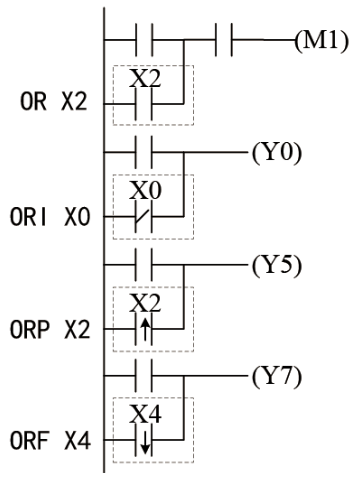
Table 3–3 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
S	√	√	√	-	-	√	-	-	-

Function and Instruction Description

The OR and ORI instructions are used for state operations of parallel contacts. These instructions are used to read the state of the designated parallel contact and perform an OR operation on the contact state and the contact's logical operation result. The OR result is stored in the accumulator.

- The ORP instruction is used to obtain the rising edge transition state of the contact for an OR operation.
- The ORF instruction is used to obtain the falling edge transition state of the contact for an OR operation.



3.1.1.5 MEP&MEF

MEP – Conversion of operation result to rising edge pulse

MEF– Conversion of operation result to falling edge pulse

16-bit instruction	MEP: Continuous execution			
32-bit instruction	-			
16-bit instruction	MEF: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	BOOL

Function and Instruction Description

MEP

The operation results up to the MEP instruction become conductive when the driving contacts turn from OFF to ON.

The use of MEP instructions simplifies the process of changing driving contacts to pulses when multiple contact points connect in a series.

MEF

The operation results up to the MEF instruction become conductive when the driving contacts turn from ON to OFF.

The use of MEF instructions simplifies the process of changing driving contacts to pulses when multiple contact points connect in a series.

Instruction Example

- MEP instruction (ON during rising edge of operation results)

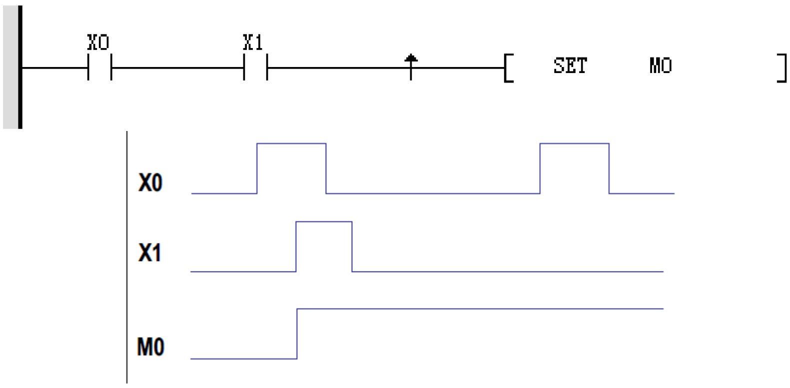
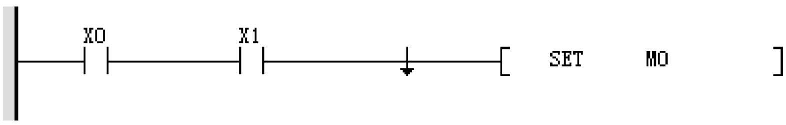


Figure 3-1 Timing Diagram

- MEF instruction (ON during falling edge of operation results)



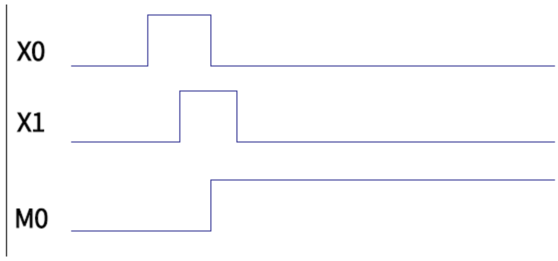


Figure 3-2

3.1.2 Output Control Instructions

3.1.2.1 Instruction List

The following table lists the output control instructions.

Instruction Category	Instruction	Function
Output control instruction	OUT	Coil drive
	SET	SET action storage coil instruction
	RST	Contact or cache clearing
	ZSET	Batch data setting
	ZRST	Batch data reset
	PLS	Pulse rising edge detection coil instruction
	PLF	Pulse falling edge detection coil instruction
	ALT	Alternate output

3.1.2.2 OUT

OUT – Coil drive

16-bit instruction	OUT: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

Table 3–4 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	√	-	-	-

Note[1] The X element is not supported. The S element applies to separate instructions. For details, see SFC instructions.

Function and Instruction Description

The OUT instruction outputs the logical operation results prior to this instruction to the designated element.

If the operand D is a pointer variable, use PGET to initialize the pointer variable. Otherwise, the system will report an error indicating that the address is invalid.

Instruction Example

3.1.2.3 SET

SET – SET action storage coil

16-bit instruction	SET: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

Table 3–5 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	√	-	-	-

Note[1] The X element is not supported. The S element applies to separate instructions. For details, see SFC instructions.

Function and Instruction Description

When the SET instruction is driven, the component designated by this instruction is set to ON and remains so regardless of whether the instruction is still driven. You can use the RST instruction to set the component to OFF.

If the operand D is a pointer variable, use PGET to initialize the pointer variable. Otherwise, the system will report an error indicating that the address is invalid.

Instruction Example

3.1.2.4 RST

RST – Contact or cache clearing

16-bit instruction	RST (bit): Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

Table 3–6 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	√	-	-	-

Note[1] The X element is not supported. The S element applies to separate instructions. For details, see SFC instructions.

Function and Instruction Description

When the RST instruction is driven, the component designated by this instruction is set to OFF and remains so regardless of whether the instruction is still driven. You can use the SET instruction to set the component to ON.

Instruction Example

3.1.2.5 ZSET

ZSET – Batch setting

Instruction	Name	LD Expression	LiteST Expression
ZSET	Batch data setting	-[ZSET ??? ???]	ZSET(???, ???);

16-bit instruction	ZSET (bit): Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D1	Element starting address	Starting address of elements or variables to be set in batches	-	BOOL
D2	Element ending address	Ending address of elements or variables to be set in batches	-	BOOL

Table 3–7 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-

Note[1] The X element is not supported.

Function and Instruction Description

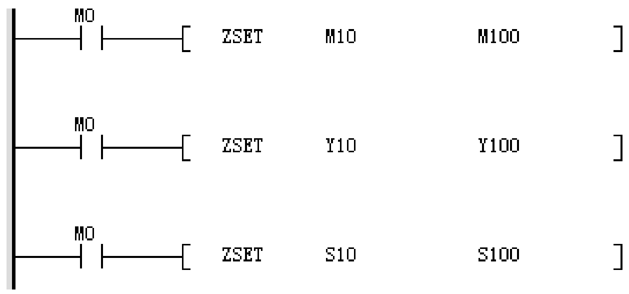
The ZSET instruction sets the values of all variables between D1 and D2 to 1. D1 and D2 can be set to Y, M, S, or B bit elements or other bit variables.

Note the following:

D1 and D2 must be of the same element type.

D1 cannot be greater than D2. If they are the same, only the specified element is set.

Instruction Example



Additional Information

Bit elements Y, M, S, and B can be set independently using the SET instruction.

3.1.2.6 ZRST

ZRST – Batch data reset

Instruction	Name	LD Expression	LiteST Expression
ZRST	Batch data reset	ZRST ??? ???	ZRST(???, ???);

16-bit instruction	ZRST: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type

D1	Element starting address	Starting address of elements or variables to be reset in batches	-	BOOL, INT, DINT Array*(D2–D1+1)
D2	Element ending address	Ending address of elements or variables to be reset in batches	-	BOOL

Table 3–8 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	√	√	√	√	-	-	-	-
D2	√[1]	√	√	√	√	-	-	-	-

Note[1] The X element is not supported.

Function and Instruction Description

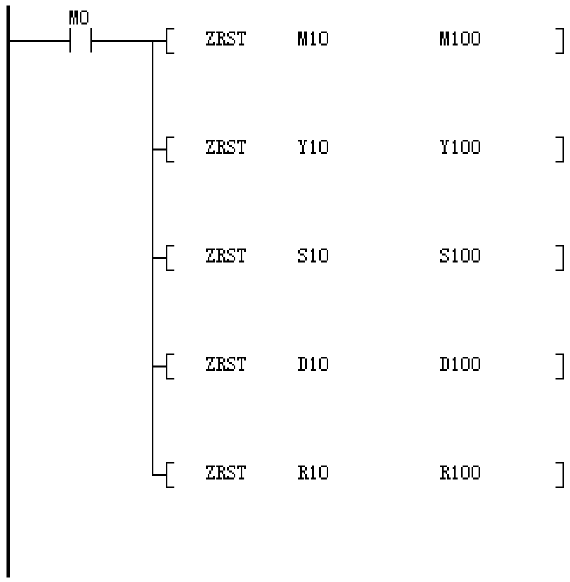
The ZRST instruction clears values of all variables between D1 and D2. D1 and D2 can be specified as word elements, word variables, bit elements, or bit variables.

Note the following:

D1 and D2 must be of the same element type.

D1 cannot be greater than D2. If they are the same, only the specified element is reset.

Instruction Example



Additional Information

Bit elements Y, M, S, and B can be reset independently using the RST instruction. Word elements D, R, and W can be reset independently using the ZRST instruction.

3.1.2.7 PLS&PLF

PLS – Pulse rising edge detection coil instruction

16-bit instruction	PLS: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

Table 3–9 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	-	-	-	-

Note[1] The X element is not supported.

PLF – Pulse falling edge detection coil instruction

16-bit instruction	PLF: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	Output element	Bit element or variable to output	-	BOOL

Table 3–10 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	-	-	-	-

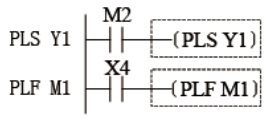
Note[1] The X element is not supported.

Function and Instruction Description

When the PLS instruction is driven by the rising edge, the element designated by this instruction is set to ON and remains so within only one scan cycle.

When the PLF instruction is driven by the falling edge, the component designated by this instruction is set to ON and remains so within only one scan cycle.

Instruction Example



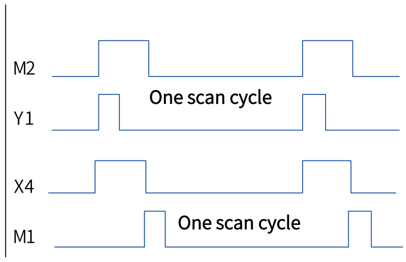


Figure 3-3 Timing Diagram

Figure 3-4 Timing Diagram

3.1.2.8 ALT

ALT – Alternate output

When the driving conditions are met, the ALT instruction switches the state (ON/OFF) of the bit element D.

16-bit instruction	ALT: Continuous execution/ALTP: Pulse execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	Execution device	Bit element	-	BOOL

Table 3–11 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	√[1]	√	√	-	-	-	-	-	-

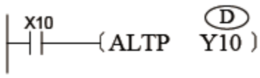
Note[1] The X element is not supported.

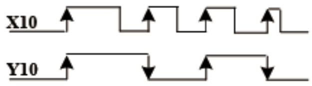
Function and Instruction Description

This instruction switches the state of the D element when the flow is active. D is a bit element or bit variable.

The pulse execution ALTP instruction is usually used.

Instruction Example





The action generated by the following instruction is the same as that generated by the ALTP instruction

3.1.2.9 R_TRIG

R_TRIG – Rising edge detection

The R_TRIG instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from FALSE to TRUE (rising edge).

Instruction	Name	LD Expression	LiteST Expression
R_TRIG	Rising edge detection		R_TRIG(CLK:=,Q=>);

16-bit instruction	Rising edge detection			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
CLK	Input value	Bit element	-	BOOL
Q	Input result	Bit element	-	BOOL

Table 3–12 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
CLK	√[1]	√	√	-	-	-	-	-	-
Q	√[1]	√	√	-	-	-	-	-	-

Note[1] The X element is not supported.

Function and Instruction Description

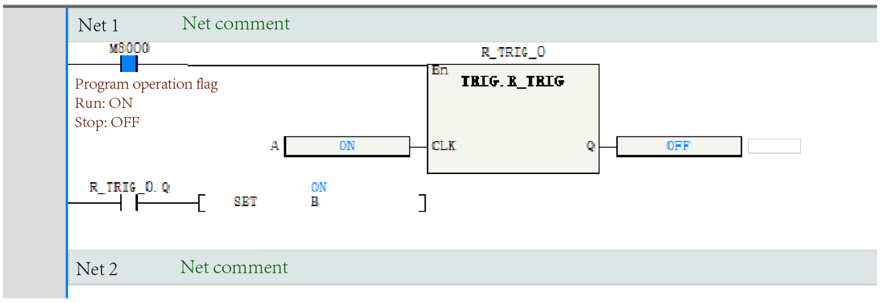
This instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from FALSE to TRUE (rising edge).

Note

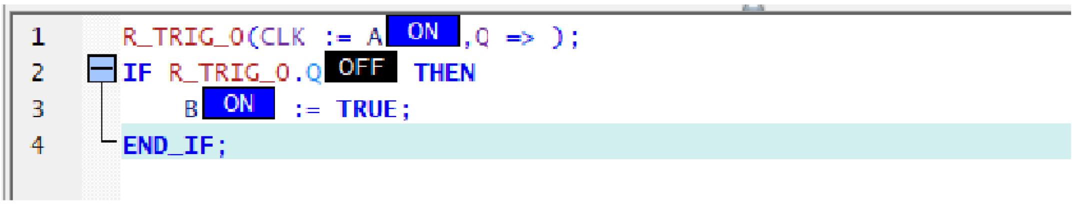
The R_TRIG instruction is used in the same way as a function block. You need to declare it in the instance table before calling.

Instruction Example

LD



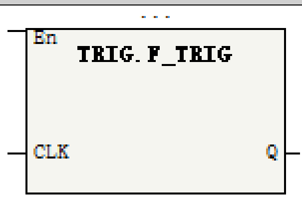
LiteST



3.1.2.10 F_TRIG

F_TRIG – Falling edge detection

The F_TRIG instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from TRUE to FALSE (falling edge).

Instruction	Name	LD Expression	LiteST Expression
F_TRIG	Falling edge detection		F_TRIG(CLK:=,Q=>);

16-bit instruction	Falling edge detection			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
CLK	Input value	Bit element	-	BOOL
Q	Input result	Bit element	-	BOOL

Table 3–13 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
CLK	√[1]	√	√	-	-	-	-	-	-
Q	√[1]	√	√	-	-	-	-	-	-

Note[1] The X element is not supported.

Function and Instruction Description

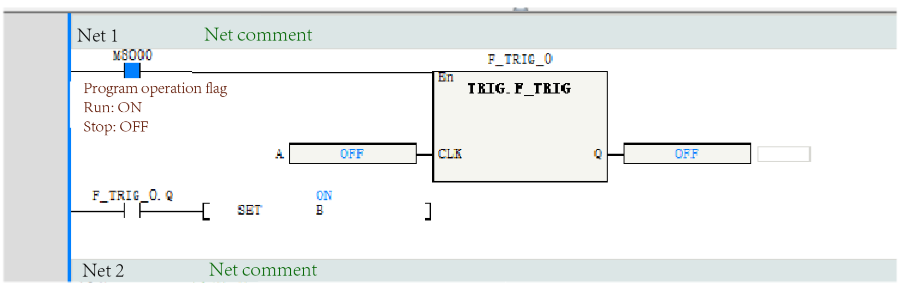
This instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from TRUE to FALSE (falling edge).

Note

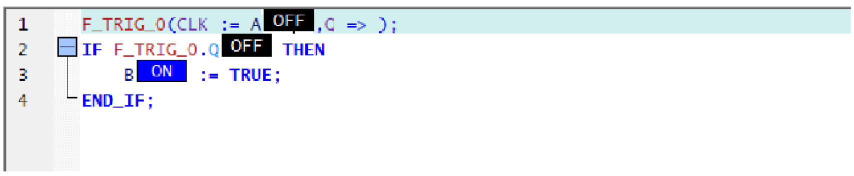
The F_TRIG instruction is used in the same way as a function block. You need to declare it in the instance table before calling.

Instruction Example

LD



LiteST



3.1.3 Flow Control Instruction

3.1.3.1 INV

INV – Operation result inversion

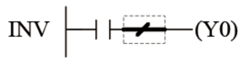
16-bit instruction	INV: Continuous execution
32-bit instruction	-

Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

The INV instruction inverts the logical operation result prior to this instruction. The result is stored in the accumulator. After the INV instruction is executed, the flow state switches from ON to OFF, or vice versa.

Instruction Example



3.2 Process Control Instructions

3.2.1 Instruction List

The following table lists the process control instructions.

Instruction Category	Instruction	Function
Process control instruction	CJ	Conditional jump
	LBL	Label
	CALL	Call subprogram
	SSRET	Conditional subprogram return
	EI	Enable interrupt
	DI	Disable interrupt
	WDT	Watchdog timer reset
	FOR	Start of a loop
	NEXT	End of a loop

3.2.2 CJ

CJ – Conditional jump

16-bit instruction	CJ: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	Target label	Target label to jump to	-	-

Table 3–14 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	L [1]

Note[1] Only the L element is supported.

3.2.3 LBL

LBL – Label

16-bit instruction	LBL: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	Label number	Current label number	-	-

Table 3–15 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	L [1]

Note[1] Only the L element is supported.

Function and Instruction Description

- When the flow is active, the program automatically jumps from the address of the CJ instruction to the address specified by L***. Program execution continues after the jump, and the program instructions in the intermediate addresses are skipped.
- When the flow is inactive, the program is executed without jump. The CJ instruction is not executed.

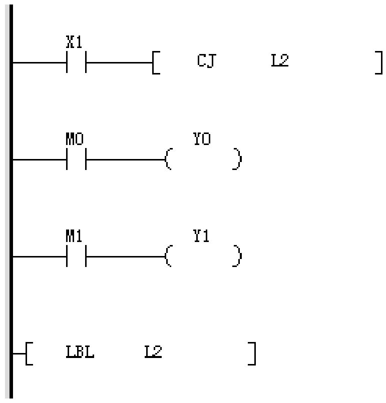
Note the following:

- The CJ instruction must work with the LBL instruction, and the target label must be located in the current program block. Jumping across program blocks is not allowed.
- The addresses defined by the label cannot be duplicate in the same program block.
- When part of a program does not need to be executed or two coils are used for output, this instruction can be used to avoid the double coil problem
- The CJ instruction can designate the same label repeatedly.

NoteThe CJ instruction cannot be used in subprograms, interrupt subprograms, FBs, and FCs.

Instruction Example

The CJ instruction is used as follows in AutoShop:



When X1 is disconnected, the program scans normally; when X1 is closed, the program jumps directly to L2 upon detecting the CJ instruction.

3.2.4 CALL

CALL – Call subprogram

16-bit instruction	CALL: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	Subprogram	Serial number of the target subprogram to be called	-	-

Table 3–16 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	P [1]

Note

- [1] Only the PL element is supported.
- It is added automatically by AutoShop without user input.
- Up to six nested layers (including the main program layer) of subprograms are supported.

Function and Instruction Description

When the flow is active, the program calls the subprogram specified by SBR_. After the subprogram is executed, the program returns to the next instruction of the CALL (or CALLP) statement to execute the subsequent statement.

Note the following about the SBR_ address pointer:

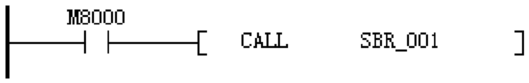
- The subprogram starting from SBR_ must be located after the end of the main program (ended with the FEND instruction).
- A subprogram must end with the SRET statement.

- The subprogram starting from SBR_ can be called in multiple locations or by another subprogram, but the number of nested layers cannot exceed six.
- A subprogram cannot be called within itself; otherwise, an infinite loop or program running timeout occurs.
- Subprograms are programmed in an independent window in AutoShop, which eliminates the problems of the FEND and SRET instructions. The names (including Chinese characters) of subprograms can be modified as needed.
- A subprogram cannot be called recursively.

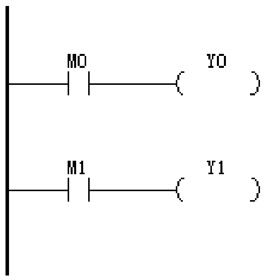
Instruction Example

The CALL instruction is used as follows in AutoShop:

- Main program



- Subprogram: SBR_001



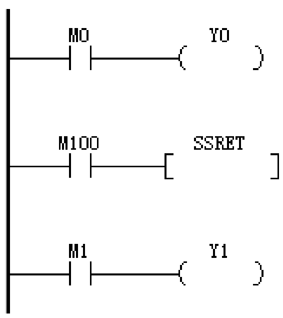
3.2.5 SSRET

SSRET – Conditional subprogram return

16-bit instruction	SSRET: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Instruction Example

As shown in the following figure, when M100 is ON, program execution directly returns to the main program (this instruction can be executed only in a subprogram).



3.2.6 EI & DI

EI – Enable interrupt

DI – Disable interrupt

Instruction	Name	LD Expression	LiteST Expression
EI	Enable interrupt	-[EI]	EI();
DI	Disable interrupt	-[DI]	DI();

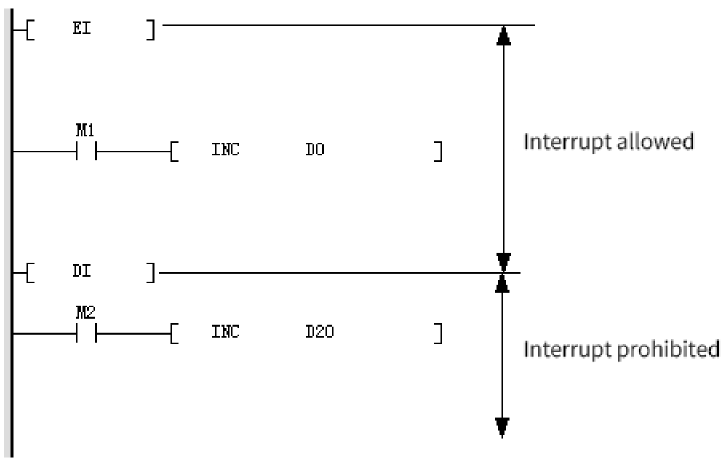
16-bit instruction	EI: Continuous execution			
32-bit instruction	-			
16-bit instruction	DI: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

When the PLC program starts running, the interrupt function is disabled by default. The EI statement enables the interrupt function, and the DI statement disables the interrupt function. The DI instruction is not required when the program does not have a range in which interrupt insertion is prohibited.

For the introduction and usage of the interrupt subprogram, see the "Interrupts" and "Subprograms" sections in the AutoShop Programming and Application Manual.

Interrupts are classified into the external signal input interrupt, high-speed count comparison interrupt, and timer interrupt.



3.2.7 WDT

WDT – Watchdog timer reset

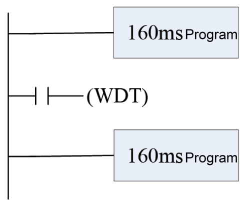
16-bit instruction	WDT: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

The PLC has a timer used to monitor the user program operation cycle. If the operation cycle times out, the program is stopped and an alarm is generated. The WDT instruction can reset the watchdog timer, allowing it to start timing again to avoid the timeout error.

An operation timeout error may occur when the operation performed by the user program is too complicated (for example, too many loop calculations). To avoid this error, you can insert the WDT instruction (for example, insert it between the FOR and NEXT instructions) when necessary during the programming process.

Instruction Example



If the watchdog timer for the program is set to 200 ms, and the program scan time is 320 ms, running the program directly will cause the watchdog timer to time out. In this case, you can insert the WDT instruction to divide the program into two segments, each with a scan time of less than 200 ms.

3.2.8 FOR&NEXT

FOR – Start of a loop

16-bit instruction	FOR: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	Number of loops	Number of loops	-	INT

Function and Instruction Description

The FOR instruction identifies the start position and specifies the number of repetitions of the loop. It must be used with the NEXT instruction. In the instruction, S is the variable that specifies the number of loops.

NEXT – End of a loop

16-bit instruction	NEXT: Continuous execution
32-bit instruction	-

Operand	Name	Description	Range	Data Type
-	-	-	-	-

Function and Instruction Description

The NEXT instruction identifies the end position of a loop. After the loop between the FOR and NEXT instructions is repeated for N times (N is specified by the FOR instruction), the PLC proceeds to subsequent execution.

The FOR-NEXT loop can be nested up to six levels (including the outermost level). The PLC parses each FOR-NEXT loop level in sequence during running. When the number of loops is too large, the PLC scan cycle becomes too long, which may result in an error when the watchdog timer times out. To avoid this error, you can insert the WDT instruction between the FOR and NEXT instructions.

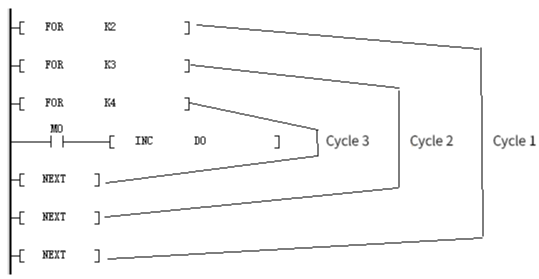
NoteThe FOR-NEXT loop can be nested for up to six layers.

Errors

- An error will be reported in the following cases:
- The NEXT instruction precedes the FOR instruction.
- The FOR instruction exists without the NEXT instruction.
- The FOR and NEXT instructions are not equal in quantity.

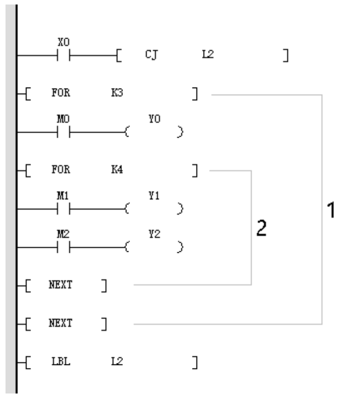
Instruction Example

- Example 1



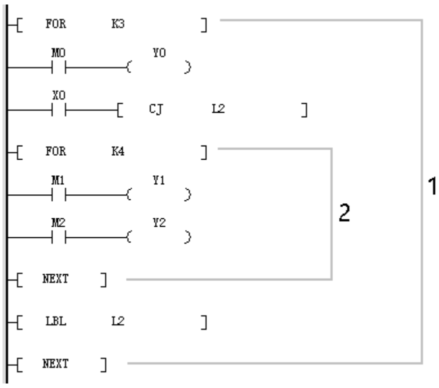
1 indicates loop 1, 2 indicates loop 2, and 3 indicates loop 3. After loop 1 is executed twice, program execution after the NEXT instruction continues. Each time loop 1 is executed, loop 2 is repeated 3 times; each time loop 2 is executed, loop 3 is repeated 4 times. Therefore, loop 3 is repeated 24 times, and loop 2 is repeated 6 times.

- Example 2



1 indicates loop 1, and 2 indicates loop 2. To skip the FOR-NEXT loops, you can insert a CJ instruction. In this example, when X0 is OFF, loop 1 and loop 2 are executed. When X0 is ON, program execution jumps from the CJ instruction to L2, and loop 1 and loop 2 are not executed.

- Example 3



1 indicates loop 1, and 2 indicates loop 2. To skip the FOR-NEXT loop nested in a loop, you can also insert a CJ instruction. In this example, when X0 is OFF, loop 2 inside loop 1 is executed. When X0 is ON, program execution jumps from the CJ instruction to L2, and FOR-NEXT loop 2 nested in loop 1 is skipped.

3.3 SFC Instructions

3.3.1 Instruction List

The following table lists the SFC instructions.

Instruction Category	Instruction	Function
SFC instruction	STL	Program jump to secondary bus
	RET	Program return to primary bus
	OUTSTL	Output program jump to secondary bus
	SETSTL	Setting program jump to secondary bus
	RSTSTL	Resetting program jump to secondary bus

Note

The SFC instruction is only used in the main program and cannot be used in subprograms and interrupt subprograms.

3.3.2 STL

STL – Program jump to secondary bus

16-bit instruction	STL: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S	STL number	S number of the STL statement to be executed	-	BOOL

Table 3–17 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	S [1]

Note

[1] Only the S element is supported.

3.3.3 RET

RET – Program return to primary bus

16-bit instruction	RET: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
-	-	-	-	-

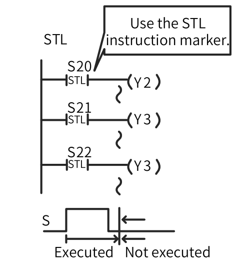
Function and Instruction Description

STL splits the running process of a controlled device into several states or procedures, performs logical programming based on each state, and then switches between states based on the signal condition. STL programming simplifies logical design and makes commissioning and maintenance easier.

The STL instruction can be represented by a ladder chart, where the state (S) is considered as a control procedure used for the sequential programming of input conditions and output control. This type of control separates the ongoing procedure from the preceding procedure and implements device control by executing various procedures in sequence.

STL and ladder charts differ in programming.

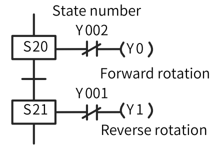
An STL program starts with the STL instruction (which is different from S used in general ladder charts) and ends with the RET instruction. The intermediate programs are guided by the S state. The operation logic of the S state is switched to the next state when conditions are met.



If the S contact of the STL instruction is connected, the circuit connected to this contact becomes active. If the S contact is disconnected, the circuit becomes inactive. The instruction is no longer executed (in jumping state) after one scan cycle.

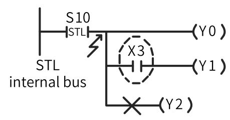
Different S states may correspond to the same output element. When S21 or S22 is connected, Y3 is output. The issue of dual coil processing also exists in the same S state. Special attention is required. S

S state numbers must be unique.



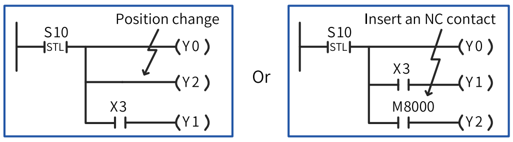
☆ **Output interlock**

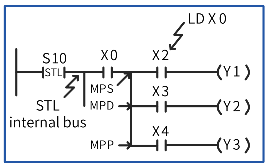
Two states are both active for a moment (one scan cycle) during state transition. To prevent simultaneous connection of a pair of outputs that should not be connected at the same time, configure external interlock for the PLC and configure interlock for the corresponding program.



☆ **Output driving**

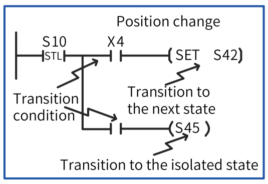
After the LD or LDI instruction is written from the internal bus, instructions that do not need contacts can no longer be used, as shown in the figure on the left. You need to modify the circuit according to the following figure.





☆ Locations of MPS, MRD, and MPP for stack operation

In the state, the MPS, MRD, and MPP instructions cannot be used directly in the STL internal bus. A program must be compiled after the LD or LDI instruction, as shown in the figure on the left.



☆ **State transition method**

The OUT and SET instructions have the same function (automatic reset of the transition source) for the state (S) after the STL instruction. Both instructions have the self-hold function.
However, the OUT instruction executes transition to the isolated state in SFC.

3.3.4 OUTSTL/SETSTL/RSTSTL

OUTSTL – Output program jump to secondary bus

SETSTL – Setting Program jump to secondary bus

RSTSTL – Resetting program jump to secondary bus

16-bit instruction	OUTSTL: Continuous execution
32-bit instruction	-
16-bit instruction	SETSTL: Continuous execution
32-bit instruction	-

16-bit instruction	RSTSTL: Continuous execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
D	STL number	S number of the STL statement to be executed	-	BOOL

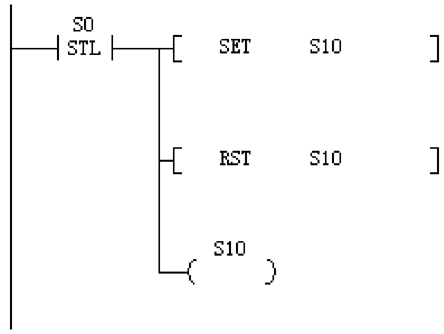
Table 3–18 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	-	-	-	S [1]

Note

[1] Only the S element is supported.

Instruction Example



3.4 Contact Operation Instructions

3.4.1 Contact Comparison Instructions

3.4.1.1 Instruction List

The following table lists the contact comparison instructions.

Instruction Category	Instruction	Function
Contact comparison instructions	LD=	LD contact comparison equal to
	LD>	LD contact comparison greater than
	LD<	LD contact comparison less than
	LD<>	LD contact comparison not equal to
	LD>=	LD contact comparison greater than or equal to
	LD<=	LD contact comparison less than or equal to
	AND=	AND contact comparison equal to
	AND>	AND contact comparison greater than
	AND<	AND contact comparison less than
	AND<>	AND contact comparison not equal to
	AND>=	AND contact comparison greater than or equal to
	AND<=	AND contact comparison less than or equal to
	OR=	OR contact comparison equal to
	OR>	OR contact comparison greater than
	OR<	OR contact comparison less than
(Continued) Contact comparison instructions	OR<>	OR contact comparison not equal to
	OR>=	OR contact comparison greater than or equal to
	OR<=	OR contact comparison less than or equal to
	FLDD>	State contact of floating-point comparison >, conductive when S1 > S2
	FLDD>=	State contact of floating-point comparison >=, conductive when S1 ⩾ S2
	FLDD<	State contact of floating-point comparison <, conductive when S1 < S2
	FLDD<=	State contact of floating-point comparison <=, conductive when S1 ⩽ S2
	FLDD=	State contact of floating-point comparison =, conductive when S1 = S2
	FLDD<>	State contact of floating-point comparison <>, conductive when S1 ≠ S2
	FANDD>	AND state contact of floating-point comparison >, conductive when S1 > S2
	FANDD>=	AND state contact of floating-point comparison >=, conductive when S1 ⩾ S2

Instruction Category	Instruction	Function
(Continued) Contact comparison instructions	FANDD<	AND state contact of floating-point comparison <, conductive when $S1 < S2$
	FANDD<=	AND state contact of floating-point comparison <=, conductive when $S1 \leqslant S2$
	FANDD=	AND state contact of floating-point comparison =, conductive when $S1 = S2$
	FANDD<>	AND state contact of floating-point comparison <>, conductive when $S1 \neq S2$
	FORD>	OR state contact of floating-point comparison >, conductive when $S1 > S2$
	FORD>=	OR state contact of floating-point comparison >=, conductive when $S1 \geqslant S2$
	FORD<	OR state contact of floating-point comparison <, conductive when $S1 < S2$
	FORD<=	OR state contact of floating-point comparison <=, conductive when $S1 \leqslant S2$
	FORD=	OR state contact of floating-point comparison =, conductive when $S1 = S2$
	FORD<>	OR state contact of floating-point comparison <>, conductive when $S1 \neq S2$
	LDZ>	State contact of absolute value comparison >, conductive when $\lvert S1 - S2 \rvert > \lvert S3 \rvert$
	LDZ>=	State contact of absolute value comparison >=, conductive when $\lvert S1 - S2 \rvert \geqslant \lvert S3 \rvert$
	LDZ<	State contact of absolute value comparison <, conductive when $\lvert S1 - S2 \rvert < \lvert S3 \rvert$
	LDZ<=	State contact of absolute value comparison <=, conductive when $\lvert S1 - S2 \rvert \leqslant \lvert S3 \rvert$
	LDZ=	State contact of absolute value comparison =, conductive when $\lvert S1 - S2 \rvert = \lvert S3 \rvert$
	LDZ<>	State contact of absolute value comparison <>, conductive when $\lvert S1 - S2 \rvert \neq \lvert S3 \rvert$

Instruction Category	Instruction	Function
(Continued) Contact Comparison Instructions	ANDZ>	AND state contact of absolute value comparison >, conductive when |S1 – S2| > |S3|
	ANDZ>=	AND state contact of absolute value comparison >=, conductive when |S1 – S2| ⩾ |S3|
	ANDZ<	AND state contact of absolute value comparison <, conductive when |S1 – S2| < |S3|
	ANDZ<=	AND state contact of absolute value comparison <=, conductive when |S1 – S2| ⩽ |S3|
	ANDZ=	AND state contact of absolute value comparison =, conductive when |S1 – S2| = |S3|
	ANDZ<>	AND state contact of absolute value comparison <>, conductive when |S1 – S2| ≠ |S3|
	ORZ>	OR state contact of absolute value comparison >, conductive when |S1 – S2| > |S3|
	ORZ>=	OR state contact of absolute value comparison >=, conductive when |S1 – S2| ⩾ |S3|
	ORZ<	OR state contact of absolute value comparison <, conductive when |S1 – S2| < |S3|
	ORZ<=	OR state contact of absolute value comparison <=, conductive when |S1 – S2| ⩽ |S3|
	ORZ=	OR state contact of absolute value comparison =, conductive when |S1 – S2| = |S3|
	ORZ<>	OR state contact of absolute value comparison <>, conductive when |S1 – S2| ≠ |S3|

3.4.1.2 AND#

Data comparison instructions – The AND# instruction compares two operands and outputs the comparison result as a logical state. The variables in comparison are processed as signed numbers.

AND= – AND contact comparison equal to

AND> – AND contact comparison greater than

AND< – AND contact comparison less than

AND<> – AND contact comparison not equal to

AND>= – AND contact comparison greater than or equal to

AND<= – AND contact comparison less than or equal to

16-bit instruction	AND=: Continuous execution
32-bit instruction	ANDD=: Continuous execution
16-bit instruction	AND>: Continuous execution
32-bit instruction	ANDD>: Continuous execution
16-bit instruction	AND<: Continuous execution

32-bit instruction	ANDD<: Continuous execution			
16-bit instruction	AND<>: Continuous execution			
32-bit instruction	ANDD<>: Continuous execution			
16-bit instruction	AND<=: Continuous execution			
32-bit instruction	ANDD<=: Continuous execution			
16-bit instruction	AND>=: Continuous execution			
32-bit instruction	ANDD>=: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data source to be compared or data variable unit 1	-	INT/DINT
S2	Comparand 2	Data source to be compared or data variable unit 2	-	INT/DINT

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- The AND# instruction is preceded by other logical operations.
- This instruction compares two operands and outputs the comparison result as a logical state, which is used for a program flow operation. The variables in comparison are processed as signed numbers.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Table 3–19 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

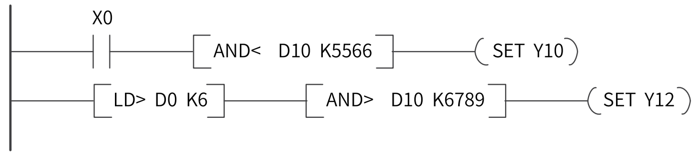
Function and Instruction Description

The following table lists the AND contact comparison modes.

Table 3–20

16-bit Instruction	FNC NO	32-bit Instruction	ON Condition	OFF Condition
AND=	232	ANDD=	S1 = S2	S1 ≠ S2
AND>	233	ANDD>	S1 > S2	S1 <= S2
AND<	234	ANDD<	S1 < S2	S1 >= S2
AND<>	236	ANDD<>	S1 <> S2	S1 = S2
AND<=	237	ANDD<=	S1 <= S2	S1 > S2
AND>=	238	ANDD>=	S1 >= S2	S1 < S2

Instruction Example



When X0 is ON and the value of D10 is smaller than that of K5566, Y10 is ON and remains ON.

When the value of D0 is greater than that of K6 and the value of D10 is greater than that of K6789, Y12 is ON and remains ON.

Use the 32-bit instruction ANDD# when comparing 32-bit variables; otherwise, an error will occur.

3.4.1.3 LD#

Contact comparison instructions – The LD# instruction compares two operands and outputs the comparison result as a logical state. The variables in comparison are processed as signed numbers.
LD= – Contact comparison equal to

LD> – Contact comparison greater than

LD< – Contact comparison less than

LD<> – Contact comparison not equal to

LD>= – Contact comparison greater than or equal to

LD<= – Contact comparison less than or equal to

16-bit instruction	LD=: Continuous execution
32-bit instruction	LDD=: Continuous execution
16-bit instruction	LD>: Continuous execution
32-bit instruction	LDD>: Continuous execution
16-bit instruction	LD<: Continuous execution
32-bit instruction	LDD<: Continuous execution
16-bit instruction	LD<>: Continuous execution
32-bit instruction	LDD<>: Continuous execution
16-bit instruction	LD<=: Continuous execution
32-bit instruction	LDD<=: Continuous execution
16-bit instruction	LD>=: Continuous execution
32-bit instruction	LDD>=: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data source to be compared or data variable unit 1	-	INT/DINT
S2	Comparand 2	Data source to be compared or data variable unit 2	-	INT/DINT

Note

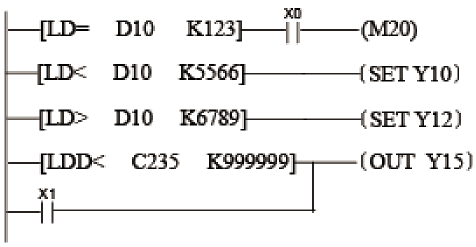
- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Function and Instruction Description

The following table lists the LD contact comparison modes.

16-bit Instruction	FNC NO	32-bit Instruction	ON Condition	OFF Condition
LD=	224	LDD=	S1 = S2	S1 ≠ S2
LD>	225	LDD>	S1 > S2	S1 <= S2
LD<	226	LDD<	S1 < S2	S1 >= S2
LD<>	228	LDD<>	S1 ≠ S2	S1 = S2
LD<=	229	LDD<=	S1 <= S2	S1 > S2
LD>=	230	LDD>=	S1 >= S2	S1 < S2

Instruction Example



Use the 32-bit instruction LDD# when comparing 32-bit variables; otherwise, an error occur.

3.4.1.4 OR#

Data comparison instructions – The OR# instruction compares two operands and outputs the comparison result as a logical state. The variables in comparison are processed as signed numbers.

OR= – OR contact comparison equal to

OR> – OR contact comparison greater than

OR< – OR contact comparison less than

OR<> – OR contact comparison not equal to

OR>= – OR contact comparison greater than or equal to

OR<= – OR contact comparison less than or equal to

16-bit instruction	OR=: Continuous execution
32-bit instruction	ORD=: Continuous execution

16-bit instruction	OR>: Continuous execution			
32-bit instruction	ORD>: Continuous execution			
16-bit instruction	OR<: Continuous execution			
32-bit instruction	ORD<: Continuous execution			
16-bit instruction	OR<>: Continuous execution			
32-bit instruction	ORD<>: Continuous execution			
16-bit instruction	OR<=: Continuous execution			
32-bit instruction	ORD<=: Continuous execution			
16-bit instruction	OR>=: Continuous execution			
32-bit instruction	ORD>=: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data source to be compared or data variable unit 1	-	INT/DINT
S2	Comparand 2	Data source to be compared or data variable unit 2	-	INT/DINT

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- This instruction compares two operands and outputs the comparison result as a logical state, which is used for a program flow operation. The variables in comparison are processed as signed numbers.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Table 3–21 List of elements

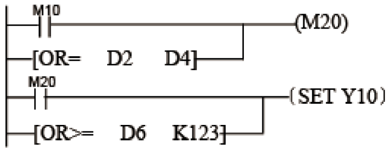
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The following table lists the OR contact comparison modes.

16-bit Instruction	FNC NO	32-bit Instruction	ON Condition	OFF Condition
OR=	240	ORD=	S1 = S2	S1 ≠ S2
OR>	241	ORD>	S1 > S2	S1 <= S2
OR<	242	ORD<	S1 < S2	S1 >= S2
OR<>	244	ORD<>	S1 <> S2	S1 = S2
OR<=	245	ORD<=	S1 <= S2	S1 > S2
OR>=	246	ORD>=	S1 >= S2	S1 < S2

Instruction Example



When M10 is ON, or the value of D2 is equal to that of D4, M20 is ON.

When M20 is ON, or the value of D6 is greater than or equal to that of K123, Y10 is ON and remains ON.

Use the 32-bit instruction ORD# when comparing 32-bit variables; otherwise, an error will occur.

3.4.1.5 FLDD#

Floating-point contact comparison – The FLDD# instruction compares two floating-point operands and then sets a contact (a node directly connected to the left-hand bus) to ON or OFF based on the comparison result.

FLDD= – Floating-point contact comparison equal to

FLDD> – Floating-point contact comparison greater than

FLDD< – Floating-point contact comparison less than

FLDD<> – Floating-point contact comparison not equal to

FLDD>= – Floating-point contact comparison greater than or equal to

FLDD<= – Floating-point contact comparison less than or equal to

16-bit instruction	-			
32-bit instruction	FLDD>: Continuous execution			
16-bit instruction	-			
32-bit instruction	FLDD>=: Continuous execution			
16-bit instruction	-			
32-bit instruction	FLDD<: Continuous execution			
16-bit instruction	-			
32-bit instruction	FLDD<=: Continuous execution			
16-bit instruction	-			
32-bit instruction	FLDD=: Continuous execution			
16-bit instruction	-			
32-bit instruction	FLDD<>: Continuous execution			
Operand	Name	Description	Range	Data Type

S1	Data 1	Element number of source data 1	-	REAL
S2	Data 2	Element number of source data 2	-	REAL

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the FLDD*, FANDD*, and FORD* instructions, the input is FLDD*, and the corresponding instructions are automatically generated at the background.

Table 3–22 List of elements

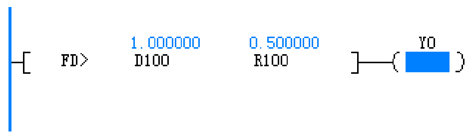
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

Function and Instruction Description

The FLDD# instruction compares S1 and S2. The contact becomes conductive (ON) when the conditions are met; otherwise, it is non-conductive (OFF).

32-bit Instruction	ON Condition	OFF Condition
FLDD>	S1 > S2	S1 <= S2
FLDD>=	S1 >= S2	S1 < S2
FLDD<	S1 < S2	S1 >= S2
FLDD<=	S1 <= S2	S1 > S2
FLDD=	S1 = S2	S1 <> S2
FLDD<>	S1 <> S2	S1 = S2

Instruction Example



3.4.1.6 FANDD#

Floating-point AND contact comparison – The FANDD# instruction compares two floating-point operands and sets a contact (a node connected to another node in series) to ON or OFF based on the comparison result.

FANDD= – Floating-point AND contact comparison equal to

FANDD> – Floating-point AND contact comparison greater than

FANDD< – Floating-point AND contact comparison less than

FANDD<> – Floating-point AND contact comparison not equal to

FANDD>= – Floating-point AND contact comparison greater than or equal to

FANDD<= – Floating-point AND contact comparison less than or equal to

16-bit instruction	-
32-bit instruction	FANDD>: Continuous execution
16-bit instruction	-
32-bit instruction	FANDD>=: Continuous execution
16-bit instruction	-
32-bit instruction	FANDD<: Continuous execution
16-bit instruction	-
32-bit instruction	FANDD<=: Continuous execution
16-bit instruction	-
32-bit instruction	FANDD=: Continuous execution
16-bit instruction	-
32-bit instruction	FANDD<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Data 1	Element number of source data 1	-	REAL
S2	Data 2	Element number of source data 2	-	REAL

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the FLDD*, FANDD*, and FORD* instructions, the input is FLDD*, and the corresponding instructions are automatically generated at the background.

Table 3–23 List of elements

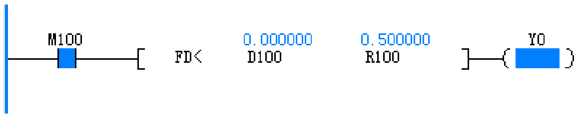
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

Function and Instruction Description

The FANDD# instruction compares S1 and S2. The contact becomes conductive (ON) when the conditions are met; otherwise, it is non-conductive (OFF).

32-bit Instruction	ON Condition	OFF Condition
FANDD>	S1 > S2	S1 <= S2
FANDD>=	S1 >= S2	S1 < S2
FANDD<	S1 < S2	S1 >= S2
FLD<=	S1 <= S2	S1 > S2
FANDD=	S1 = S2	S1 <> S2
FANDD<>	S1 <> S2	S1 = S2

Instruction Example



3.4.1.7 FORD#

Floating-point OR contact comparison – The FORD# instruction compares two floating-point operands and sets a contact (a node connected to another node in parallel) to ON or OFF based on the comparison result.

FORD= – Floating-point OR contact comparison equal to

FORD> – Floating-point OR contact comparison greater than

FORD< – Floating-point OR contact comparison less than

FORD<> – Floating-point OR contact comparison not equal to

FORD>= – Floating-point OR contact comparison greater than or equal to

FORD<= – Floating-point OR contact comparison less than or equal to

16-bit instruction	-
32-bit instruction	FORD>: Continuous execution
16-bit instruction	-
32-bit instruction	FORD>=: Continuous execution
16-bit instruction	-
32-bit instruction	FORD<: Continuous execution
16-bit instruction	-
32-bit instruction	FORD<=: Continuous execution
16-bit instruction	-
32-bit instruction	FORD=: Continuous execution
16-bit instruction	-
32-bit instruction	FORD<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Data 1	Element number of source data 1	-	REAL
S2	Data 2	Element number of source data 2	-	REAL

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the FLDD*, FANDD*, and FORD* instructions, the input is FLDD*, and the corresponding instructions are automatically generated at the background.

Table 3–24 List of elements

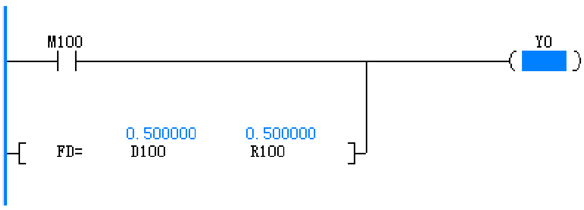
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-

Function and Instruction Description

The FORD# instruction compares S1 and S2. The contact becomes conductive (ON) when the conditions are met; otherwise, it is non-conductive (OFF).

32-bit Instruction	ON Condition	OFF Condition
FORD>	S1 > S2	S1 <= S2
FORD>=	S1 >= S2	S1 < S2
FORD<	S1 < S2	S1 >= S2
FORD<=	S1 <= S2	S1 > S2
FORD=	S1 = S2	S1 <> S2
FORD<>	S1 <> S2	S1 = S2

Instruction Example



3.4.1.8 LDZ#

Absolute value comparison contact – The LDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3 and sets a contact (a node directly connected to the left-hand bus) to ON or OFF based on the comparison result.

LDZ= – Absolute value contact comparison equal to

LDZ> – Absolute value contact comparison greater than

LDZ< – Absolute value contact comparison less than

LDZ<> – Absolute value contact comparison not equal to

LDZ>= – Absolute value contact comparison greater than or equal to

LDZ<= – Absolute value contact comparison less than or equal to

16-bit instruction	LDZ>: Continuous execution
32-bit instruction	LDDZ>: Continuous execution
16-bit instruction	LDZ>=: Continuous execution
32-bit instruction	LDDZ>=: Continuous execution
16-bit instruction	LDZ<: Continuous execution
32-bit instruction	LDDZ<: Continuous execution
16-bit instruction	LDZ<=: Continuous execution
32-bit instruction	LDDZ<=: Continuous execution
16-bit instruction	LDZ=: Continuous execution
32-bit instruction	LDDZ=: Continuous execution
16-bit instruction	LDZ<>: Continuous execution
32-bit instruction	LDDZ<>: Continuous execution

Operand	Name	Description	Range	Data Type
S1	Subtrahend	Source element of the subtrahend	-	INT/DINT
S2	Minuend	Source element of the minuend	-	INT/DINT
S3	Comparand	Source element of the comparand	-	INT/DINT

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LDZ*/LDDZ*, ANDZ*/ANDDZ*, and ORZ*/ORDZ* instructions, the input is LDZ*/LDDZ*, and the corresponding instructions are automatically generated at the background.

Table 3–25 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

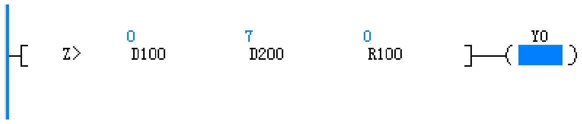
Function and Instruction Description

The LDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3. The contact becomes conductive (ON) when the conditions are met; otherwise, it is non-conductive (OFF).

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
LDZ>	LDDZ>	|S1 – S2| > |S3|	|S1 – S2| <= |S3|
LDZ>=	LDDZ>=	|S1 – S2| >= |S3|	|S1 – S2| < |S3|

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
LDZ<	LDDZ<	|S1 – S2| < |S3|	|S1 – S2| >= |S3|
LDZ<=	LDDZ<=	|S1 – S2| <= |S3|	|S1 – S2| > |S3|
LDZ=	LDDZ=	|S1 – S2| = |S3|	|S1 – S2| <> |S3|
LDZ<>	LDDZ<>	|S1 – S2| <> |S3|	|S1 – S2| = |S3|

Instruction Example



3.4.1.9 ANDZ#

Absolute value comparison AND contact – The ANDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3 and sets a contact (a node connected to another node in series) to ON or OFF based on the comparison result.

ANDZ= – Absolute value AND contact comparison equal to

ANDZ> – Absolute value AND contact comparison greater than

ANDZ< – Absolute value AND contact comparison less than

ANDZ<> – Absolute value AND contact comparison not equal to

ANDZ>= – Absolute value AND contact comparison greater than or equal to

ANDZ<= – Absolute value AND contact comparison less than or equal to

16-bit instruction	ANDZ>: Continuous execution			
32-bit instruction	ANDDZ>: Continuous execution			
16-bit instruction	ANDZ>=: Continuous execution			
32-bit instruction	ANDDZ>=: Continuous execution			
16-bit instruction	ANDZ<: Continuous execution			
32-bit instruction	ANDDZ<: Continuous execution			
16-bit instruction	ANDZ<=: Continuous execution			
32-bit instruction	ANDDZ<=: Continuous execution			
16-bit instruction	ANDZ=: Continuous execution			
32-bit instruction	ANDDZ=: Continuous execution			
16-bit instruction	ANDZ<>: Continuous execution			
32-bit instruction	ANDDZ<>: Continuous execution			
Operand	Name	Description	Range	Data Type

S1	Subtrahend	Source element of the subtrahend	-	INT/DINT
S2	Minuend	Source element of the minuend	-	INT/DINT
S3	Comparand	Source element of the comparand	-	INT/DINT

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LDZ*/LDDZ*, ANDZ*/ANDDZ*, and ORZ*/ORDZ* instructions, the input is LDZ*/LDDZ*, and the corresponding instructions are automatically generated at the background.

Table 3–26 List of elements

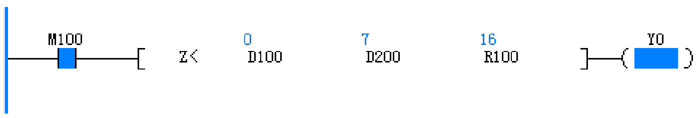
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ANDZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3. The contact becomes conductive (ON) when the conditions are met; otherwise, it is non-conductive (OFF).

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
ANDZ>	ANDDZ>	|S1 – S2| > |S3|	|S1 – S2| <= |S3|
ANDZ>=	ANDDZ>=	|S1 – S2| >= |S3|	|S1 – S2| < |S3|
ANDZ<	ANDDZ<	|S1 – S2| < |S3|	|S1 – S2| >= |S3|
ANDZ<=	ANDDZ<=	|S1 – S2| <= |S3|	|S1 – S2| > |S3|
ANDZ=	ANDDZ=	|S1 – S2| = |S3|	|S1 – S2| <> |S3|
ANDZ<>	ANDDZ<>	|S1 – S2| <> |S3|	|S1 – S2| = |S3|

Instruction Example



3.4.1.10 ORZ#

Absolute value comparison OR contact – The ORZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3 and sets a contact (a node connected to another node in parallel) to ON or OFF based on the comparison result.

ORZ= – Absolute value OR contact comparison equal to

ORZ> – Absolute value OR contact comparison greater than

ORZ< – Absolute value OR contact comparison less than

ORZ<> – Absolute value OR contact comparison not equal to

ORZ>= – Absolute value OR contact comparison greater than or equal to

ORZ<= – Absolute value OR contact comparison less than or equal to

16-bit instruction	ORZ>: Continuous execution			
32-bit instruction	ORDZ>: Continuous execution			
16-bit instruction	ORZ>=: Continuous execution			
32-bit instruction	ORDZ>=: Continuous execution			
16-bit instruction	ORZ<: Continuous execution			
32-bit instruction	ORDZ<: Continuous execution			
16-bit instruction	ORZ<=: Continuous execution			
32-bit instruction	ORDZ<=: Continuous execution			
16-bit instruction	ORZ=: Continuous execution			
32-bit instruction	ORDZ=: Continuous execution			
16-bit instruction	ORZ<>: Continuous execution			
32-bit instruction	ORDZ<>: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Subtrahend	Source element of the subtrahend	-	INT/DINT
S2	Minuend	Source element of the minuend	-	INT/DINT
S3	Comparand	Source element of the comparand	-	INT/DINT

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- For the LDZ*/LDDZ*, ANDZ*/ANDDZ*, and ORZ*/ORDZ* instructions, the input is LDZ*/LDDZ*, and the corresponding instructions are automatically generated at the background.

Table 3–27 List of elements

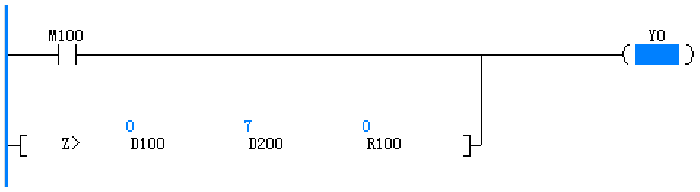
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ORZ# instruction compares the absolute value of the S1 and S2 subtraction result with the absolute value of S3. The contact becomes conductive (ON) when the conditions are met; otherwise, it is non-conductive (OFF).

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
ORZ>	ORDZ>	|S1 – S2| > |S3|	|S1 – S2| <= |S3|
ORZ>=	ORDZ >=	|S1 – S2| >= |S3|	|S1 – S2| < |S3|
ORZ<	ORDZ<	|S1 – S2| < |S3|	|S1 – S2| >= |S3|
ORZ<=	ORDZ<=	|S1 – S2| <= |S3|	|S1 – S2| > |S3|
ORZ=	ORDZ=	|S1 – S2| = |S3|	|S1 – S2| <> |S3|
ORZ<>	ORDZ<>	|S1 – S2| <> |S3|	|S1 – S2| = |S3|

Instruction Example



3.4.2 Contact Logical Operation Instructions

3.4.2.1 Instruction List

The following table lists the contact logical operation instructions.

Instruction Category	Instruction	Function
Contact Logical Operation Instructions	LD&	LD logical AND operation
	LD|	LD logical OR operation
	LD^	LD logical XOR operation
	AND&	AND logical AND operation
	AND|	AND logical OR operation
	AND^	AND logical XOR operation
	OR&	OR logical AND operation
	OR|	OR logical OR operation
	OR^	OR logical XOR operation

3.4.2.2 LD*

LD logical operation instructions – The bit logical operation result is used to determine whether the contact (a node directly connected to the left-hand bus) is conductive.

LD& – LD logical AND operation

LD| – LD logical OR operation

LD^ – LD logical XOR operation

16-bit instruction	LD&: Continuous execution
32-bit instruction	LDD&: Continuous execution

16-bit instruction	LD|: Continuous execution			
32-bit instruction	LDD|: Continuous execution			
16-bit instruction	LD^: Continuous execution			
32-bit instruction	LDD^: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Element number of source data 1	-	INT/DINT
S2	Data 2	Element number of source data 2	-	INT/DINT

Note

- * indicates &, |, or ^.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Table 3–28 List of elements

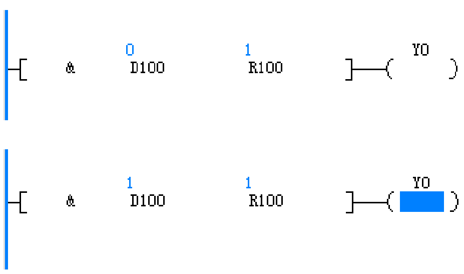
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The instruction performs a logical operation (AND: &; NOT: |; XOR: ^) on S1 and S2. It is conductive (ON) if the operation result is not 0 and non-conductive (OFF) if the operation result is 0. The execution results are as follows:

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
LD&	LDD&	S1&S2 ≠ 0	S1&S2 = 0
LD|	LDD|	S1|S2 ≠ 0	S1|S2 = 0
LD^	LDD^	S1^S2 ≠ 0	S1^S2 = 0

Instruction Example



3.4.2.3 AND*

The bit logical operation result is used to determine whether the contact (a node connected to another node in series) is conductive.

AND& – AND logical AND operation

AND| – AND logical OR operation

AND^ – AND logical XOR operation

16-bit instruction	AND&: Continuous execution			
32-bit instruction	ANDD&: Continuous execution			
16-bit instruction	AND|: Continuous execution			
32-bit instruction	ANDD|: Continuous execution			
16-bit instruction	AND^: Continuous execution			
32-bit instruction	ANDD^: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Element number of source data 1	-	INT/DINT
S2	Data 2	Element number of source data 2	-	INT/DINT

Note

- * indicates &, |, or ^.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Table 3–29 List of elements

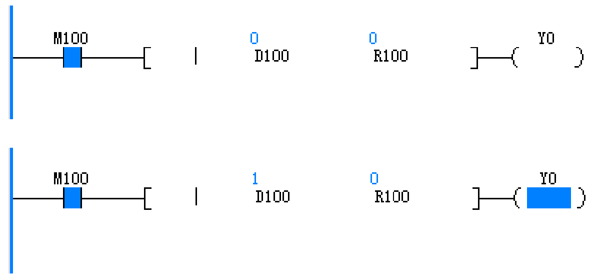
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The instruction performs a logical operation (AND: &; NOT: |; XOR: ^) on S1 and S2. It is conductive (ON) if the operation result is not 0 and non-conductive (OFF) if the operation result is 0. The execution results are as follows:

16-bit Instruction	32-bit Instruction	ON Condition	OFF Condition
AND&	ANDD&	S1&S2 ≠ 0	S1&S2 = 0
AND|	ANDD|	S1|S2 ≠ 0	S1|S2 = 0
AND^	ANDD^	S1^S2 ≠ 0	S1^S2 = 0

Instruction Example



3.4.2.4 OR*

The bit logical operation result is used to determine whether the contact (a node connected to another node in parallel) is conductive.

OR& – OR logical AND operation

OR| – OR logical OR operation

OR^ – OR logical XOR operation

16-Bit Instruction	OR&: Continuous execution			
32-Bit Instruction	ORD&: Continuous execution			
16-Bit Instruction	OR|: Continuous execution			
32-Bit Instruction	ORD|: Continuous execution			
16-Bit Instruction	OR^: Continuous execution			
32-Bit Instruction	ORD^: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Element number of source data 1	-	INT/DINT
S2	Data 2	Element number of source data 2	-	INT/DINT

Note

- * indicates &, |, or ^.
- For the LD*/LDD*, AND*/ANDD*, and OR*/ORD* instructions, the input is LD*/LDD*, and the corresponding instructions are automatically generated at the background.

Table 3–30 List of elements

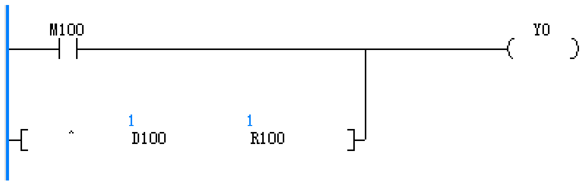
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

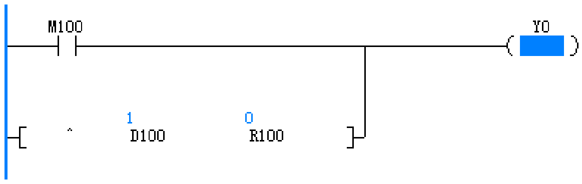
Function and Instruction Description

The instruction performs a logical operation (AND: &; NOT: |; XOR: ^) on S1 and S2. It is conductive (ON) if the operation result is not 0 and non-conductive (OFF) if the operation result is 0. The execution results are as follows:

16-Bit Instruction	32-Bit Instruction	ON Condition	OFF Condition
OR&	ORD&	S1&S2 ≠ 0	S1&S2 = 0
OR|	ORD|	S1|S2 ≠ 0	S1|S2 = 0
OR^	ORD^	S1^S2 ≠ 0	S1^S2 = 0

Instruction Example





3.5 Data Operation Instructions

3.5.1 Arithmetic Operation Instructions

3.5.1.1 Instruction List

The following table lists the arithmetic operation instructions.

Instruction Category	Instruction	Function
Arithmetic Operation Instructions	ADD	Binary data addition
	SUB	Binary data subtraction
	MUL	Binary data multiplication
	DIV	Binary data division
	MOD	Remainder by binary data division
	EADD	Binary floating-point addition
	ESUB	Binary floating-point subtraction
	EMUL	Binary floating-point multiplication
	EDIV	Binary floating-point division
	INC	Binary data increment by 1
	DEC	Binary data decrement by 1

3.5.1.2 ADD

ADD – Binary data addition

16-bit instruction	ADD: Continuous execution/ADDP: Pulse execution			
32-bit instruction	DADD: Continuous execution/DADDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Augend	Data, or address of the word element that stores the data	-	INT/DINT
S2	Addend	Data, or address of the word element that stores the data	-	INT/DINT
D	Sum	Address of the word element that stores the data	-	INT/DINT

Table 3–31 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

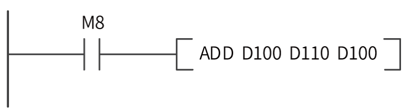
Function and Instruction Description

The ADD instruction requires contact driving and has three operands. It adds S1 and S2 algebraically in binary format, and stores the addition result in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

- The zero flag (M8020) is set if the operation result is 0.
- The carry flag (M8022) is set if the operation result is greater than 32,767 (in 16-bit operation) or 2,147,483,647 (in 32-bit operation).
- The borrow flag (M8021) is set if the operation result is less than –32,768 (in 16-bit operation) or –2,147,483,648 (in 32-bit operation).
- In 32-bit operation, the variable address in the ADD instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

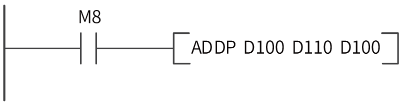
Instruction Example

Example 1



When M8 is set, D110 (addend) is added to D100 (augend), and the addition result is stored to D100. Assume that D100 is K8 and D110 is K-12, then D100 = 8 + (–12) = –4.

Example 2



D110 (addend) is added to D100 (augend) on the rising edge of M8, and the addition result is stored to D100.

3.5.1.3 SUB

SUB – Binary data subtraction

16-bit instruction	SUB: Continuous execution/SUBP: Pulse execution			
32-bit instruction	DSUB: Continuous execution/DSUBP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Subtrahend	Data, or address of the word element that stores the data	-	INT/DINT
S2	Minuend	Data, or address of the word element that stores the data	-	INT/DINT
D	Difference	Address of the word element that stores the data	-	INT/DINT

Table 3–32 List of elements

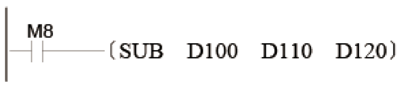
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The SUB instruction requires contact driving and has three operands. It performs subtraction on S1 and S2 algebraically in binary format, and stores the subtraction result in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

- The zero flag (M8020) is set if the operation result is 0.
- The carry flag (M8022) is set if the operation result is greater than 32,767 (in 16-bit operation) or -2,147,483,647 (in 32-bit operation).
- The borrow flag (M8021) is set if the operation result is less than –32,768 (in 16-bit operation) or –2,147,483,648 (in 32-bit operation).
- In 32-bit operation, the variable address in the SUB instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

Instruction Example



When M8 is set, D110 (minuend) is subtracted from D100 (subtrahend), and the subtraction result is stored to D120. Assume that D100 is K10 and D110 is K8, then D120 = 10 – 8 = K2.

3.5.1.4 MUL

MUL – Binary data multiplication

16-bit Instruction	MUL: Continuous execution/MULP: Pulse execution			
32-bit Instruction	DMUL: Continuous execution/DMULP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Multiplicand	Data, or address of the word element that stores the data	-	INT/DINT
S2	Multiplier	Data, or address of the word element that stores the data	-	INT/DINT
D	Product	Address of the word element that stores the data	-	DINT

Table 3–33 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

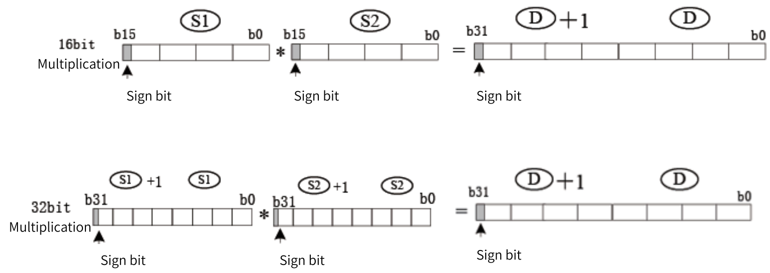
Function and Instruction Description

The MUL instruction requires contact driving and has three operands. It multiplies S1 by S2 algebraically in binary format, and stores the multiplication result in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

In 32-bit operation, the variable address in the MUL instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming. The operation result must be 32-bit data. If not, the floating-point operation instruction EMUL is recommended.

For 16-bit multiplication, the product is 32-bit data.

For 32-bit multiplication, the product is 32-bit data.



Instruction Example

Figure 3-5 Ladder chart and instruction list

When M8 is set, D100 (multiplicand) is multiplied by D110 (multiplier), and the multiplication result is stored in D120.

If D100 is K5 and D110 is K9, then D120 = 5 x 9 = K45.

If D100 is K1234 and D110 is K5678, then D120 and D121 = 1234 x 5678 = K7006652. Note that the product is greater than 16 bits at this time, and both D121 and D120 are used.

3.5.1.5 DIV

DIV – Binary data division

16-bit Instruction	DIV: Continuous execution/DIVP: Pulse execution			
32-bit Instruction	DDIV: Continuous execution/DDIVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Dividend	Data, or address of the word element that stores the data	-	INT/DINT
S2	Divisor	Data, or address of the word element that stores the data	-	INT/DINT
D	Quotient	Address of the word element that stores the data	-	INT/DINT

Table 3–34 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

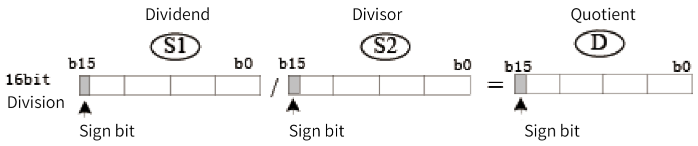
Function and Instruction Description

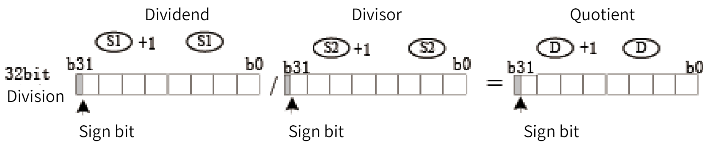
The DIV instruction requires contact driving and has three operands. It divides S1 by S2 algebraically in binary format and stores the quotient in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

In 32-bit operation, the variable address (S1 and S2) in the DIV instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming. The quotient is stored in the units pointed to by D and D+1.

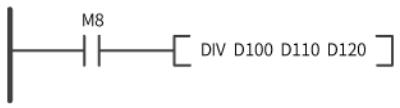
The DIV operation result involves only the quotient. To obtain the remainder, use the MOD instruction.

If the divisor S2 is 0, a calculation error occurs.





Instruction Example



LD M8
DIV D100 D110 D120

When M8 is set, D100 (dividend) is divided by D110 (divisor), and the quotient is stored in D120.

3.5.1.6 MOD

The MOD instruction is used to calculate the remainder produced by a division of two integers.

MOD – Remainder by division

Instruction	Name	LD Expression	LiteST Expression
MOD	Remainder by division	[MOD ??? ??? ???]	D := S1 MOD S2

16-bit instruction	MOD: Continuous execution/MODP: Pulse execution			
32-bit instruction	DMOD: Continuous execution/DMODP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Element number of source data 1	-	INT/DINT
S2	Data 2	Element number of source data 2	-	INT/DINT
D	Operation result	Start number of elements for storing the operation result	-	INT/DINT

Table 3–35 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The MOD instruction requires contact driving and has three operands. It divides S1 by S2 algebraically in binary format and stores the remainder in D. The variables in the algebraic operation are processed as signed numbers. The most significant bit is the sign bit. The value 0 indicates a positive number, whereas the value 1 indicates a negative number.

In 32-bit operation, the variable address in the DIV instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming. The remainder is stored in the units pointed to by D and D+1.

If the divisor S2 is 0, a calculation error occurs. If the dividend is negative, the remainder is negative.

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

		Element Name	Data Type	Display Format	Current Value
1	...	D100	INT	Decimal	10000
2	...	D200	INT	Decimal	9000
3	...	D300	INT	Decimal	1000
4	...				
5	...				

3.5.1.7 EADD

The EADD instruction adds two binary floating-point numbers together.

EADD – Floating-point addition

16-bit instruction	-			
32-bit instruction	DEADD: Continuous execution/DEADDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Augend	Binary floating-point augend	-	REAL
S2	Addend	Binary floating-point addend	-	REAL
D	Sum	Unit that stores the binary floating-point sum	-	REAL

Table 3–36 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EADD instruction performs addition of two binary floating-point numbers. Where,

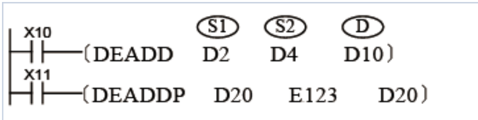
- S1 and S2 are respectively the binary floating-point augend and addend.
- D is the unit that stores the sum of S1 and S2.

The zero flag (M8020) is set if the operation result is 0.

The carry flag (M8022) is set if the absolute value of the operation result is greater than the maximum floating-point value.

The borrow flag (M8021) is set if the absolute value of the operation result is less than the minimum floating-point value.

Instruction Example



Example Explanation

When X10 is ON, two binary floating-point numbers in (D3, D2) and (D5, D4) are added together and the sum is stored in (D11, D10).

When X11 switches from OFF to ON, the floating-point number in (D21, D20) is incremented by 123.

If the sum is stored in the same unit as the augend or addend, use the DEADDP instruction of the pulse execution type. If the instruction of the continuous execution type is used, calculation is performed upon every program scan.

3.5.1.8 ESUB

The ESUB instruction performs subtraction on two binary floating-point numbers.
ESUB – Floating-point subtraction

16-bit instruction	-			
32-bit instruction	DESUB: Continuous execution/DESUBP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Subtrahend	Binary floating-point subtrahend	-	REAL
S2	Minuend	Binary floating-point minuend	-	REAL
D	Difference	Unit that stores the binary floating-point subtraction result	-	REAL

Table 3–37 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The ESUB instruction subtracts one binary floating-point number from another. Where,

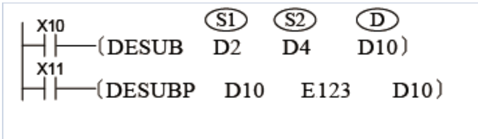
- S1 and S2 are respectively the binary floating-point subtrahend and minuend.
- D is the unit that stores the subtraction result.

The zero flag (M8020) is set if the operation result is 0.

The carry flag (M8022) is set if the absolute value of the operation result is greater than the maximum floating-point value.

The borrow flag (M8021) is set if the absolute value of the operation result is less than the minimum floating-point value.

Instruction Example



Example Explanation

When X10 is ON, the binary floating-point number in (D5, D4) is subtracted from that in (D3, D2) and the binary floating-point difference is stored in (D11, D10).

When X11 switches from OFF to ON, the floating-point number in (D11, D10) is decremented by 123.

If the difference is stored in the same unit as the subtrahend or minuend, use the DESUBP instruction of the pulse execution type. If the instruction of the continuous execution type is used, calculation is performed upon every program scan.

3.5.1.9 EMUL

The EMUL instruction multiplies two binary floating-point numbers together.
EMUL – Floating-point multiplication

16-bit instruction	-			
32-bit instruction	DEMUL: Continuous execution/DEMULP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Multiplicand	Binary floating-point multiplicand	-	REAL
S2	Multiplier	Binary floating-point multiplier	-	REAL
D	Product	Unit that stores the binary floating-point product	-	REAL

Table 3–38 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EMUL instruction multiplies two binary floating-point numbers together. Where,

- S1 and S2 are respectively the binary floating-point multiplicand and multiplier.

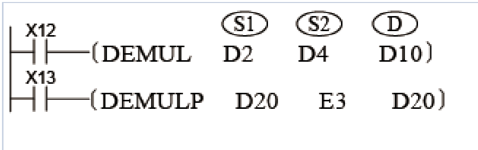
- D is the unit that stores the binary floating-point multiplication product.

The zero flag (M8020) is set if the operation result is 0.

The carry flag (M8022) is set if the absolute value of the operation result is greater than the maximum floating-point value.

The borrow flag (M8021) is set if the absolute value of the operation result is less than the minimum floating-point value.

Instruction Example



Example Explanation

When X12 is ON, the binary floating-point number in (D3, D2) is multiplied by that in (D5, D4) and the binary floating-point product is stored in (D11, D10).

When X13 switches from OFF to ON, the binary floating-point number in (D21, D20) is multiplied by 3 and the result is stored in (D21, D20)

If the product is stored in the same unit as the multiplicand or multiplier, use the DEMULP instruction of the pulse execution type. If the instruction of the continuous execution type is used, calculation is performed upon every program scan.

3.5.1.10 EDIV

The EDIV instruction divides one binary floating-point number by another.
EDIV – Floating-point division

16-bit instruction	-			
32-bit instruction	DEDIV: Continuous execution/DEDIVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Dividend	Binary floating-point dividend	-	REAL
S2	Divisor	Binary floating-point divisor	-	REAL
D	Quotient	Starting address of units that store the binary floating-point quotient	-	REAL

Table 3–39 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EDIV instruction divides one binary floating-point number by another. Where,

- S1 and S2 are respectively the binary floating-point dividend and divisor.
- D is the starting address of units for storing the binary floating-point quotient.

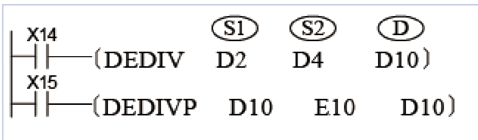
The zero flag (M8020) is set if the operation result is 0.

The carry flag (M8022) is set if the absolute value of the operation result is greater than the maximum floating-point value.

The borrow flag (M8021) is set if the absolute value of the operation result is less than the minimum floating-point value.

If the divisor is 0, a calculation error occurs.

Instruction Example



Example Explanation

When X14 is ON, the binary floating-point number in (D3, D2) is divided by that in (D5, D4) and the binary floating-point quotient is stored in (D11, D10).

When X15 switches from OFF to ON, the binary floating-point number in (D11, D10) is divided by 10 and the result is stored in (D11, D10).

If the quotient is stored in the same unit as the dividend or divisor, use the DEDIVP instruction of the pulse execution type. If the instruction of the continuous execution type is used, calculation is performed upon every program scan.

3.5.1.11 INC

The INC instruction increases the binary data by 1.
INC – Increment by 1

16-bit instruction	INC: Continuous execution/INCP: Pulse execution			
32-bit instruction	DINC: Continuous execution/DINCP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Cumulative result	Address of the word element that stores the cumulative result	-	INT/DINT

Table 3–40 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

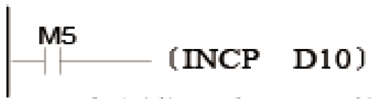
The INC instruction increases the value in D by 1 each time it is executed.

In 16-bit operation, when 32,767 increases by 1, the result is –32,768. In 32-bit operation, when 2,147,483,647 increases by 1, the result is –2,147,483,648.

This instruction does not refresh the zero flag, carry flag, and borrow flag.

In 32-bit operation, the variable address in the INC instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

Instruction Example



The value in D10 increases by 1 each time M5 is set to ON.

3.5.1.12 DEC

The DEC instruction decreases the binary data by 1.
DEC – Decrement by 1

16-bit instruction	DEC: Continuous execution/DECP: Pulse execution			
32-bit instruction	DDEC: Continuous execution/DDECP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Regressive result	Address of the word element that stores the regressive result	-	INT/DINT

Table 3–41 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	√	-	-

Function and Instruction Description

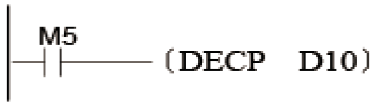
The DEC instruction decreases the value in D by 1 each time it is executed.

This instruction does not refresh the zero flag, carry flag, and borrow flag.

In 16-bit operation, when –32,768 decreases by 1, the result is 32,767. In 32-bit operation, when –2,147,483,648 decreases by 1, the result is 2,147,483,647.

In 32-bit operation, the variable address in the INC instruction is the low-order 16 bits of the address, and the adjacent subsequent address unit contains the high-order 16 bits. This prevents duplication or overwriting during programming.

Instruction Example



The value in D10 decreases by 1 each time M5 is set to ON.

3.5.2 Data Logical Operation Instructions

3.5.2.1 Instruction List

The following table lists the data logical operation instructions.

Instruction Category	Instruction	Function
Data Logical Operation Instructions	WAND	Binary data logical AND
	WOR	Binary data logical OR
	WXOR	Binary data logical XOR
	NEG	Binary data negation
	ENEG	Binary floating-point sign negation

3.5.2.2 WAND

When the driving conditions are met, the WAND instruction performs a logical AND on S1 and S2 by bit and stores the result in D.
WAND – Logical AND instruction

16-bit Instruction	WAND: Continuous execution/WANDP: Pulse execution			
32-bit Instruction	DWAND: Continuous execution/DWANDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Data in the AND operation, or address of the word element that stores the data	-	INT/DINT
S2	Data 2	Data in the AND operation, or address of the word element that stores the data	-	INT/DINT
D	Operation result	Address of the word element that stores the operation result	-	INT/DINT

Table 3–42 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

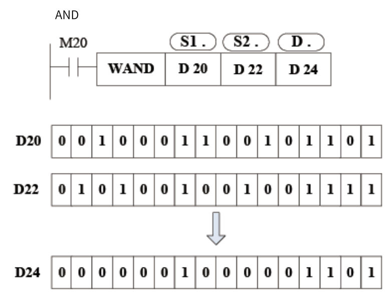
Function and Instruction Description

The WAND instruction performs logical AND on the binary values in S1 and S2 by bit and stores the operation result in the variable D.

The result of a logical AND operation is 0 if the value of either S1 or S2 is 0.

1∧1=1; 1∧0=0; 0∧1=0; 0∧0=0

Instruction Example



3.5.2.3 WOR

When the driving conditions are met, the WOR instruction performs a logical OR on S1 and S2 by bit and stores the result in D.

WOR – Logical OR instruction

16-bit Instruction	WOR: Continuous execution/WORP: Pulse execution			
32-bit Instruction	DWOR: Continuous execution/DWORP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Data in the OR operation, or address of the word element that stores the data	-	INT/DINT
S2	Data 2	Data in the OR operation, or address of the word element that stores the data	-	INT/DINT
D	Operation result	Address of the word element that stores the operation result	-	INT/DINT

Table 3–43 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The WOR instruction performs logical OR on the binary values in S1 and S2 by bit and stores the operation result in the variable D.

The result of a logical OR operation is 1 if the value of either S1 or S2 is 1.

1∨1=1; 1∨0=0; 0∨1=0; 0∨0=0

Instruction Example

3.5.2.4 WXOR

When the driving conditions are met, the WXOR instruction performs a logical XOR on S1 and S2 by bit and stores the result in D.

WXOR – Logical XOR instruction

16-bit Instruction	WXOR: Continuous execution/WXORP: Pulse execution			
32-bit Instruction	DWXOR: Continuous execution/DWXORP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Data 1	Data in the XOR operation, or address of the word element that stores the data	-	INT/DINT
S2	Data 2	Data in the XOR operation, or address of the word element that stores the data	-	INT/DINT
D	Operation result	Address of the word element that stores the operation result	-	INT/DINT

Table 3–44 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

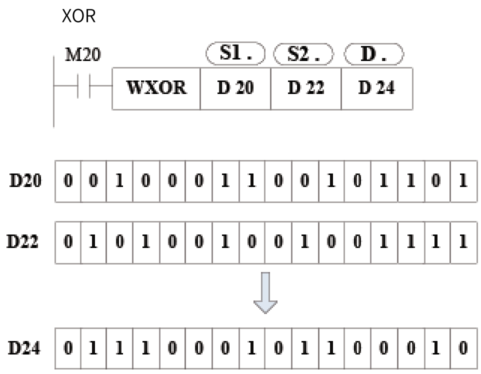
Function and Instruction Description

The WXOR instruction performs logical XOR on the binary values in S1 and S2 by bit and stores the operation result in the variable D.

The result of a logical XOR operation is 0 if S1 and S2 are the same and 1 if they are different.

$1\veebar 1=0$; $1\veebar 0=1$; $0\veebar 1=1$; $0\veebar 0=0$

Instruction Example



3.5.2.5 NEG

When the driving conditions are met, the NEG instruction inverts each bit of D, adds 1, and then writes the result to D.

NEG – Negation instruction

16-bit Instruction	NEG: Continuous execution/NEGP: Pulse execution			
32-bit Instruction	DNEG: Continuous execution/DNEGP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Operation result	Address of the word element that stores the data	-	INT/DINT

Table 3–45 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

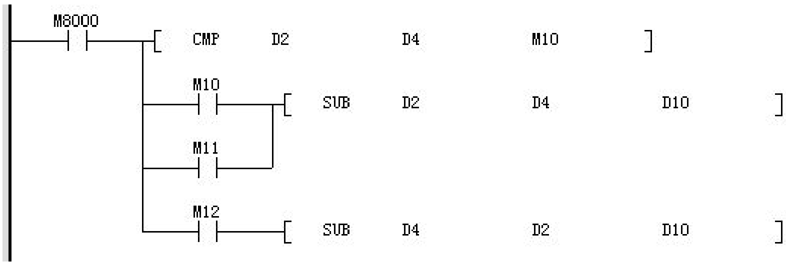
The NEG instruction requires contact driving and has one operand. It inverts each bit of D, adds 1, and then writes the result to D.

The instruction of the pulse execution type is used in normal cases.

The NEG instruction can be used to obtain the absolute value of a negative binary number.

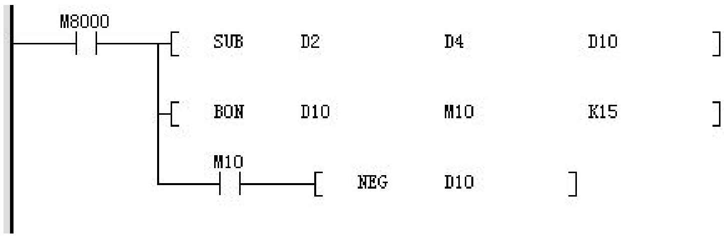
Instruction Example

The following example illustrates how to obtain the absolute value of the difference in a subtraction:



When the value of D2 is greater than that of D4, M10 is ON. When the value of D2 is equal to that of D4, M11 is ON. When the value of D2 is less than that of D4, M12 is ON. This ensures that the value in D10 is positive.

The preceding program is represented as follows:



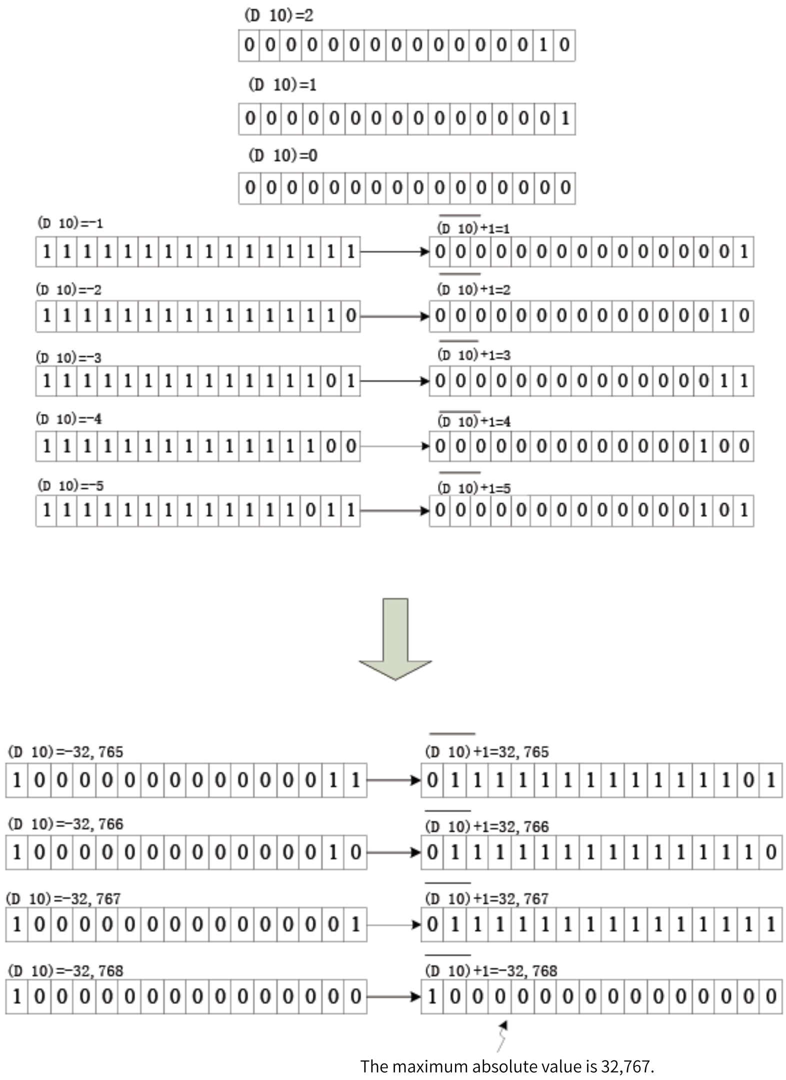
When bit 15 of D10 is 1 (indicating that the value in D10 is negative), M10 is ON. The NEG instruction can be used to obtain the absolute value of D10.

In the preceding examples, when D2 is K4 and D4 is K8, or D2 is K8 and D4 is K4, the result in D10 is K4.

Additional Information

The most significant bit (leftmost bit) of the register indicates whether a number is positive or negative. 0 indicates positive and 1 indicates negative.

When the most significant bit is 1, the NEG instruction can be used to obtain absolute value.



3.5.2.6 ENEG

The ENEG instruction inverts the sign of a binary floating-point number (real number).

ENEG – Floating-point sign negation instruction

16-bit Instruction	-			
32-bit Instruction	DENEG: Continuous execution/DENEGP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Operand	Start number of elements that store the binary floating-point number subject to a sign change	-	REAL

Table 3–46 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

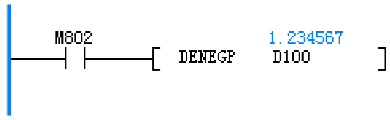
Function and Instruction Description

The ENEG instruction inverts the sign of the binary floating-point number in [D+1, D] and stores the result in [D+1, D]. The instruction of the pulse execution type is used in normal cases.

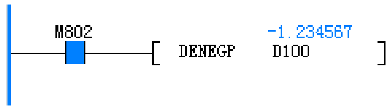
Instruction Example

Data in D100 and D101 is inverted, and the result is stored in D100 and D101.

- Before the instruction is executed



- After the instruction is executed



3.5.3 Word Bit Operation Instructions

3.5.3.1 Instruction List

The following table lists the word bit operation instructions.

Instruction Category	Instruction	Function
Word bit operation instruction	BLD	Word or dword bit contact instruction
	BLDI	Word or dword bit inversion contact instruction
	BAND	Word or dword bit AND contact instruction
	BANDI	Word or dword bit AND inversion contact instruction
	BOR	Word or dword bit OR contact instruction
	BORI	Word or dword bit OR inversion contact instruction
	BOUT	Word or dword bit data output instruction
	BSET	Word or dword bit data setting instruction
	BRST	Word or dword bit data reset instruction

3.5.3.2 BLD

The execution result (ON or OFF) of the BLD instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BLD – Word or dword bit contact instruction

16-bit Instruction	BLD: Continuous execution			
32-bit Instruction	DBLD: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Source data	Element number of the source data	-	INT/DINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Note

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

Table 3–47 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction uses the state of the specified bit of a word variable as the contact output.

Instruction Example

- n = 3:

- n = 4:

3.5.3.3 BLDI

The execution result (ON or OFF) of the BLDI instruction is determined based on the state (OFF or ON) of the specified bit of the source data.

BLDI – Word or dword bit inversion contact instruction

16-bit Instruction	BLDI: Continuous execution			
32-bit Instruction	DBLDI: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Source data	Element number of the source data	-	INT/DINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Note

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

Table 3–48 List of elements

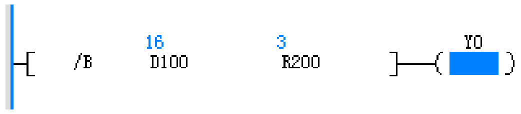
Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction inverts the state of the specified bit of a word variable and use the inversion result as the contact output.

Instruction Example

- n = 3:



- n = 4:

3.5.3.4 BAND

The execution result (ON or OFF) of the BAND instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BAND – Word or dword bit AND contact instruction

16-bit Instruction	BAND: Continuous execution			
32-bit Instruction	DBAND: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Source data	Element number of the source data	-	INT/DINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Note

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

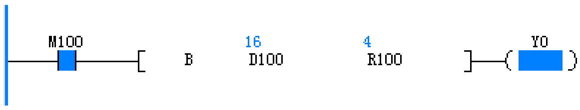
Table 3–49 List of elements

Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction uses the state of the specified bit of a word variable as the contact output.

Instruction Example



3.5.3.5 BANDI

The execution result (ON or OFF) of the BANDI instruction is determined based on the state (OFF or ON) of the specified bit of the source data.

BANDI: Word or dword bit AND inversion contact instruction

16-bit Instruction	BANDI: Continuous execution			
32-bit Instruction	DBANDI: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Source data	Element number of the source data	-	INT/DINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Note

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

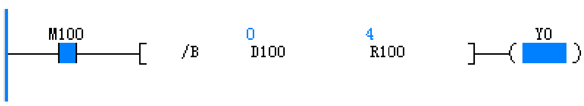
Table 3–50 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction inverts the state of the specified bit of a word variable and use the inversion result as the contact output.

Instruction Example



3.5.3.6 BOR

The execution result (ON or OFF) of the BOR instruction is determined based on the state (ON or OFF) of the specified bit of the source data.

BOR – Word or dword bit OR contact instruction

16-bit Instruction	BOR: Continuous execution			
32-bit Instruction	DBOR: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Source data	Element number of the source data	-	INT/DINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Note

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

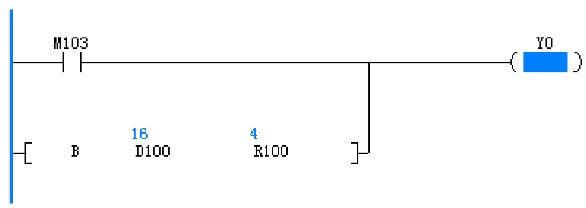
Table 3–51 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction uses the state of the specified bit of a word variable as the contact output.

Instruction Example



3.5.3.7 BORI

The execution result (ON or OFF) of the BORI instruction is determined based on the state (OFF or ON) of the specified bit of the source data.

BORI – Word or dword bit OR inversion contact instruction

16-bit Instruction	BORI: Continuous execution			
32-bit Instruction	DBORI: Continuous execution			
Operand	Name	Description	Range	Data Type

S	Source data	Element number of the source data	-	INT/DDINT
n	Load bit	Specified load bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DDINT

Note

For the BLD/DBLD, BLDI/DBLDI, BAND/DBAND, BANDI/DBANDI, BOR/DBOR, and BORI/DBORI instructions, the input is BLD/DBLD or BLDI/DBLDI, and the corresponding instructions are automatically generated at the background.

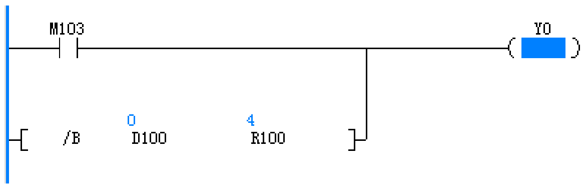
Table 3–52 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction inverts the state of the specified bit of a word variable and use the inversion result as the contact output.

Instruction Example



3.5.3.8 BOUT

The BOUT instruction outputs the logical operation result prior to this instruction to the specified bit.
BOUT – Word or dword bit data output instruction

16-bit Instruction	BOUT: Continuous execution			
32-bit Instruction	DBOUT: Continuous execution			
Operand	Name	Description	Range	Data Type
D	Output data	Element number of the output data	-	INT/DINT
n	Output bit	Specified output bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Table 3–53 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

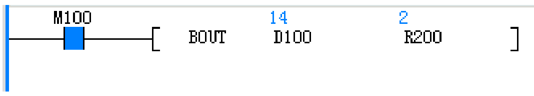
Function and Instruction Description

The BOUT instruction outputs the flow prior to this instruction to the specified bit of a word variable.

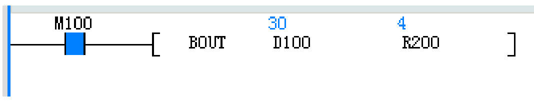
Instruction Example

The initial value of D100 is 2#1010 (decimal K10).

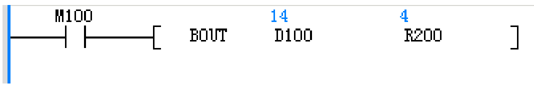
1. When R200 is 2 and M100 is ON, bit 2 of D100 is set and the value of D100 becomes 2#1110 (decimal K14).



2. When R200 is 4 and M100 is ON, bit 4 of D100 is set and the value of D100 becomes 2#11110 (decimal K30).



3. When M100 is OFF, bit 4 of D100 is reset and the value of D100 becomes 2#1110 (decimal K14).



3.5.3.9 BSET

When the BSET instruction is driven, the bit specified by this instruction is set to ON.
BSET – Word or dword bit data setting instruction

16-bit Instruction	BSET: Continuous execution			
32-bit Instruction	DBSET: Continuous execution			
Operand	Name	Description	Range	Data Type
D	Output data	Element number of the output data	-	INT/DINT
n	Output bit	Specified output bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Table 3–54 List of elements

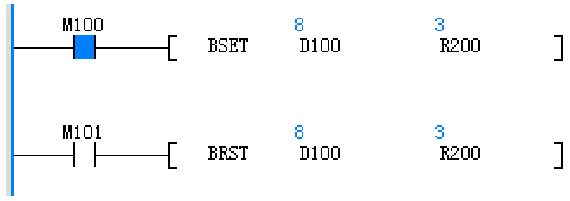
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

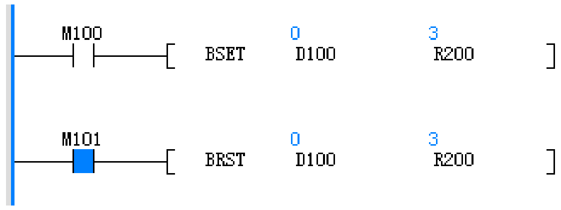
The BSET instruction sets the specified bit of a word variable to ON.

Instruction Example

- When M100 is ON:



- When M100 is OFF:



3.5.3.10 BRST

When the BRST instruction is driven, the bit specified by this instruction is set to OFF.

BRST – Word or dword bit data reset instruction

16-bit Instruction	BRST: Continuous execution			
32-bit Instruction	DBRST: Continuous execution			
Operand	Name	Description	Range	Data Type
D	Output data	Element number of the output data	-	INT/DINT
n	Output bit	Specified output bit, ranging from 0 to 15 (16-bit instruction) or 0 to 31 (32-bit instruction)	0 to 15/31	INT/DINT

Table 3–55 List of elements

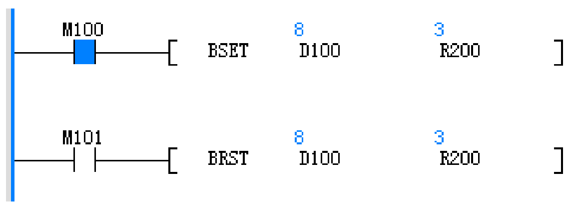
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

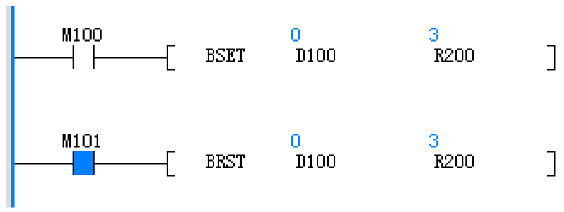
The BRST resets the specified bit of a word variable.

Instruction Example

- When M100 is OFF:



- When M101 is ON:



3.5.4 Trigonometric Function Instructions

3.5.4.1 Instruction List

The following table lists the trigonometric function instructions.

Instruction Category	Instruction	Function
Trigonometric function instruction	SIN	Floating-point SIN operation
	COS	Floating-point COS operation
	TAN	Floating-point TAN operation
	ASIN	Binary floating-point ARCSIN operation
	ACOS	Binary floating-point ARCCOS operation
	ATAN	Binary floating-point ARCTAN operation
	RAD	Binary floating-point degree-to-radian conversion
	DEG	Binary floating-point radian-to-degree conversion
	SINH	Binary floating-point SINH operation
	COSH	Binary floating-point COSH operation
	TANH	Binary floating-point TANH operation

3.5.4.2 SIN

The SIN instruction calculates the sine of the specified angle (in radians). The variable is a binary floating-point number.

SIN – Floating-point SIN operation instruction

16-bit Instruction	-			
32-bit Instruction	DSIN: Continuous execution/DSINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Angle variable (binary floating-point number, in radians) for which the sine is to be calculated; value range: $0 \leqslant \alpha \leqslant 2\pi$	-	REAL
D	Operation result	Storage unit for storing the sine value (binary floating-point)	-	REAL

Table 3–56 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The SIN instruction obtains the sine of the specified angle (in radians). The variable is a binary floating-point number. Where,

- S is the angle variable (binary floating-point number) for which the sine value is to be calculated, in the unit of rad. The value range is as follows: $0 \leqslant \alpha \leqslant 2\pi$.
- D is the storage unit for storing the SIN operation result (binary floating-point number).

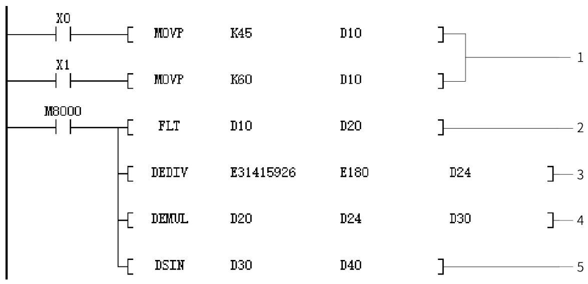
Instruction Example 1

The angle in radians specified in (D21, D20) is converted into the sine value and stored in (D31, D30).

Both the source data and SIN operation result are binary floating-point numbers.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

Instruction Example 2



- 1. X0 and X1 determine the angle in degrees, which is 45° or 60°. The value is stored in D10.
- 2. The decimal value in D10 is converted into a binary floating-point equivalent and stored in (D21, D20).
- 3. The floating-point number of (π/180) is calculated and stored in (D25, D24).
- 4. The floating-point angle in degrees in (D21, D20) is converted to the floating-point angle in radians and stored in (D31, D30).
- 5. The sine of the floating-point angle in radians (D31, D30) is calculated and stored in (D41, D40) as a floating-point number.

3.5.4.3 TAN

The TAN instruction calculates the tangent of the specified angle (in radians). The variable is a binary floating-point number.

TAN – Floating-point TAN operation instruction

16-bit Instruction	-			
32-bit Instruction	DTAN: Continuous execution/DTANP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Angle variable (binary floating-point number, in radians) for which the tangent is to be calculated; value range: 0 ⩽ α < 2π	-	REAL
D	Operation result	Storage unit for storing the tangent value (binary floating-point)	-	REAL

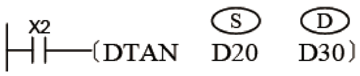
Table 3–57 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The TAN instruction obtains the tangent of the specified angle (in radians). The variable is a binary floating-point number.

Instruction Example



The angle in radians specified in (D21, D20) is converted into the tangent value and stored in (D31, D30).

Both the source data and TAN operation result are binary floating-point numbers.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

3.5.4.4 COS

The COS instruction calculates the cosine of the specified angle (in radians). The variable is a binary floating-point number.

COS – Floating-point COS operation instruction

16-bit Instruction	-			
32-bit Instruction	DCOS: Continuous execution/DCOSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Angle variable (binary floating-point number, in radians) for which the cosine is to be calculated; value range: 0 ⩽ α ⩽ 2π	-	REAL
D	Operation result	Storage unit for storing the cosine value (binary floating-point)	-	REAL

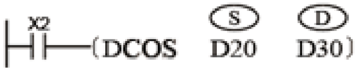
Table 3–58 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The COS instruction obtains the cosine of the specified angle (in radians). The variable is a binary floating-point number.

Instruction Example



The angle in radians specified in (D21, D20) is converted into the cosine value and stored in (D31, D30).

Both the source data and COS operation result are binary floating-point numbers.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

3.5.4.5 ASIN

The ASIN instruction calculates the angle in radians based on a sine value.
ASIN – Floating-point SIN^{-1} operation instruction

16-bit Instruction	-			
32-bit Instruction	DASIN: Continuous execution/DASINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which arcsine is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (–n/2 to +n/2)	-	REAL

Table 3–59 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

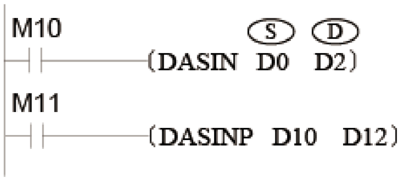
Function and Instruction Description

The ASIN instruction calculates the angle in radians based on a sine value.

Note

An operation error will occur if the value in S falls beyond the range of –1.0 to +1.0.

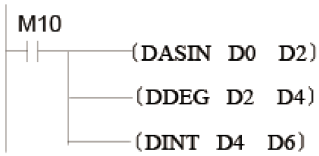
Instruction Example 1



When M10 is ON, the SIN^{-1} value of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

$$SIN^{-1}(D1、D0) \Longrightarrow (D3、D2)$$

Instruction Example 2



Assume that the value in (D1, D0) is 0.707106781. When M10 switches from OFF to ON, the value in (D3, D2) is 0.78539815, that in (D5, D4) is 45, and that in (D7, D6) is 45.

3.5.4.6 ACOS

The ACOS instruction calculates the angle in radians based a COS value.

ACOS – Floating-point COS^{-1} operation instruction

16-bit Instruction	-			
32-bit Instruction	DACOS: Continuous execution/DACOSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which arccosine is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (0 to n)	-	REAL

Table 3–60 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

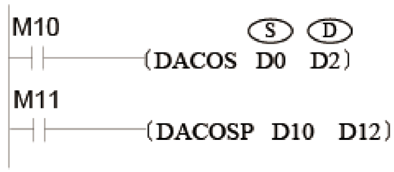
Function and Instruction Description

The ACOS instruction calculates the angle in radians based a COS value.

Note

An operation error will occur if the value in S falls beyond the range of –1.0 to +1.0.

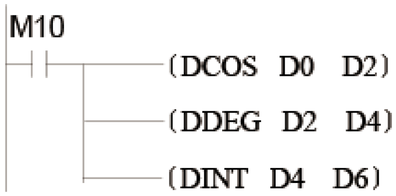
Instruction Example 1



When M10 is ON, the COS^{-1} value of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

$$\cos^{-1}(D1、D0) \Longrightarrow (D3、D2)$$

Instruction Example 2



Assume that the value in (D1, D0) is 0.866025404. When M10 switches from OFF to ON, the value in (D3, D2) is 0.52359877, that in (D5, D4) is 30, and that in (D7, D6) is 30.

3.5.4.7 ATAN

The ATAN instruction calculates the angle in radians based a TAN value.

ATAN – Floating-point TAN^{-1} operation instruction

16-bit Instruction	-			
32-bit Instruction	DATAN: Continuous execution/DATANP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which arctangent is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (–n/2 to +n/2)	-	REAL

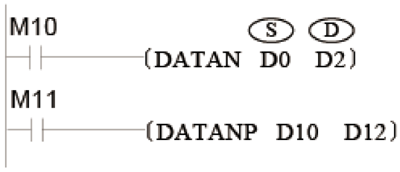
Table 3–61 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The ATAN instruction calculates the angle in radians based a TAN value.

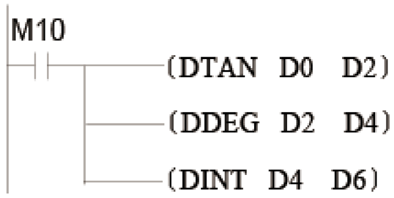
Instruction Example 1



When M10 is ON, the TAN^{-1} value of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

$$TAN^{-1}(D1、D0) \Rightarrow (D3、D2)$$

Instruction Example 2



Assume that the value in (D1, D0) is 1.732050808. When M10 switches from OFF to ON, the value in (D3, D2) is 1.04719753, that in (D5, D4) is 60, and that in (D7, D6) is 60.

3.5.4.8 RAD

The RAD instruction converts a binary floating-point value in degrees into a value in radians.
RAD – Floating-point degree-to-radian conversion instruction

16-bit Instruction	-			
32-bit Instruction	DRAD: Continuous execution/DRADP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point value in degrees to be converted to a value in radians	-	REAL
D	Operation result	Storage unit for storing the operation result	-	REAL

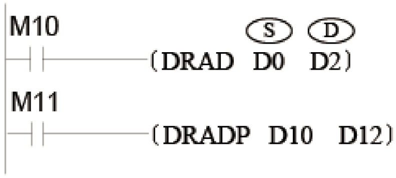
Table 3–62 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

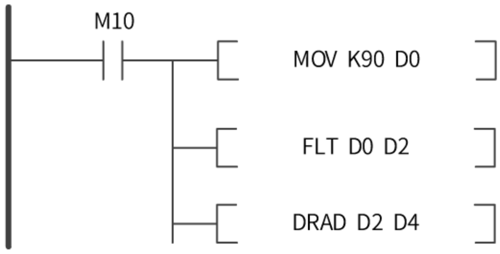
The RAD instruction converts a binary floating-point value in degrees into a value in radians. The formula is as follows: Value in radians = Value in degrees x n/180.

Instruction Example 1



When M10 is ON, the binary floating-point value in degrees in (D1, D0) is converted into a value in radians and stored in (D3, D2).

Instruction Example 2



When M10 switches from OFF to ON, the value 90 is assigned to D0. The integer in D0 is converted into a floating-point number, which is then assigned to (D3, D2). Degree-to-radian conversion is performed on (D3, D2) and the result is assigned to (D5, D4). The final value in (D5, D4) is π/2, that is, 1.570796.

3.5.4.9 DEG

The DEG instruction converts a binary floating-point value in radians into a value in degrees.
DEG – Floating-point radian-to-degree conversion instruction

16-bit Instruction	-			
32-bit Instruction	DDEG: Continuous execution/DDEGP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point value in radians to be converted to a value in degrees	-	REAL
D	Operation result	Storage unit for storing the operation result	-	REAL

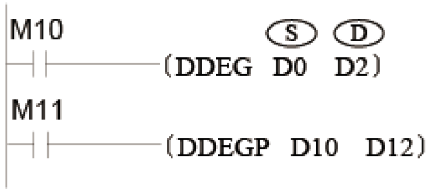
Table 3–63 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

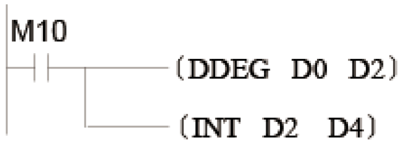
The DEG instruction converts a binary floating-point value in radians into a value in degrees. The formula is as follows: Value in degrees = Value in radians x 180/n.

Instruction Example 1



When M10 is ON, the binary floating-point value in radians in (D1, D0) is converted into a value in degrees and stored in (D3, D2).

Instruction Example 2



Assume that the value in (D1, D0) is 3.1415926. When M10 switches from OFF to ON, the value in (D3, D2) is 180. After the floating-point number is converted into an integer, the value in (D5, D4) is 180.

3.5.4.10 SINH

The SINH instruction calculates the hyperbolic sine of a binary floating-point number.

SINH – Floating-point SINH operation instruction

16-bit Instruction	-			
32-bit Instruction	DSINH: Continuous execution/DSINHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which the hyperbolic sine is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result (Error 6706 is returned if the operation result in D exceeds the floating-point range.)	-	REAL

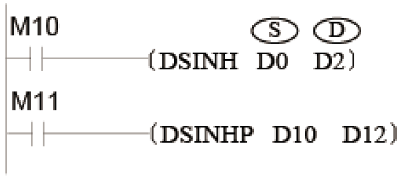
Table 3–64 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The SINH instruction calculates the hyperbolic sine of a binary floating-point number. The formula is as follows: sinh = $(e^{s} – e^{-s})/2$.

Instruction Example



When M10 is ON, the hyperbolic sine of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

3.5.4.11 COSH

The COSH instruction calculates the hyperbolic cosine of a binary floating-point number.
COSH – Floating-point COSH operation instruction

16-bit Instruction	-			
32-bit Instruction	DCOSH: Continuous execution/DCOSHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which the hyperbolic cosine is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result	-	REAL

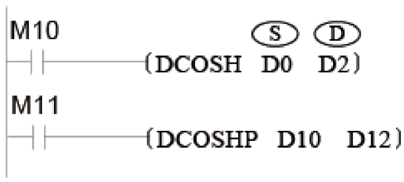
Table 3–65 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The COSH instruction calculates the hyperbolic cosine of a binary floating-point number. The formula is as follows: cosh = $(e^{s} + e^{-s})/2$.

Instruction Example



When M10 is ON, the hyperbolic cosine of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

3.5.4.12 TANH

The TANH instruction calculates the hyperbolic tangent of a binary floating-point number.
TANH – Floating-point TANH operation instruction

16-bit Instruction	-			
32-bit Instruction	DTANH: Continuous execution/DTANHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which the hyperbolic tangent is to be calculated	-	REAL
D	Operation result	Storage unit for storing the operation result	-	REAL

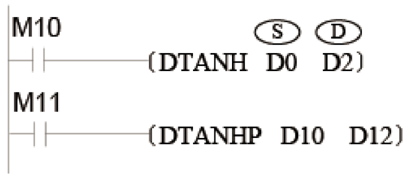
Table 3–66 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The TANH instruction calculates the hyperbolic tangent of a binary floating-point number. The formula is as follows: tanh = $(e^{s} – e^{-s})/(e^{s} + e^{-s})$.

Instruction Example



When M10 is ON, the hyperbolic tangent of the binary floating-point value in (D1, D0) is calculated and stored in (D3, D2).

3.5.5 Table Operation Instructions

3.5.5.1 Instruction List

The following table lists the table operation instructions.

Instruction Category	Instruction	Function
Table operation instruction	WSUM	Data sum calculation
	MEAN	Mean calculation
	LIMIT	Upper/Lower limit control
	BZAND	Dead zone control
	ZONE	Zone control
	SCL	Coordinate determination (coordinate data of different points)
	SCL2	Coordinate determination 2 (X and Y coordinates)

3.5.5.2 WSUM

The WSUM instruction calculates the sum of consecutive 16-bit or 32-bit data entries.

WSUM – Data sum calculation

16-bit Instruction	WSUM: Continuous execution/WSUMP: Pulse execution			
32-bit Instruction	DWSUM: Continuous execution/DWSUMP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store the data entries for which the sum is to be calculated	-	INT/DINT, array*n
D	Result	Start number of elements that store the sum	-	INT/DINT, array*2
n	Data count	Data count	2 to 256	INT/DINT

Table 3–67 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

- 16-bit instruction

  The WSUM instruction calculates the sum of n 16-bit data entries starting from [S] and stores the result as 32-bit data in [D+1, D].

- 32-bit instruction

  The WSUM instruction calculates the sum of n 32-bit data entries starting from [S+1, S] and stores the result as 64-bit data in [D+3, D+2, D+1, D].

Errors

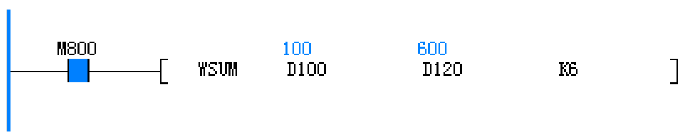
An error is returned in the following conditions:

The n elements starting from [S] are out of range.

[D] for data storage is out of range.

The operand n is less than or equal to 0.

Instruction Example



3.5.5.3 MEAN

When the driving conditions are met, the MEAN instruction calculates the mean value of n data entries starting from S and stores the result in D.

MEAN – Mean calculation

16-bit Instruction	MEAN: Continuous execution/MEANP: Pulse execution			
32-bit Instruction	DMEAN: Continuous execution/DMEANP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data start address	Start address of word elements that store the data entries for which the mean value is to be calculated	-	INT/DINT, array*n
D	Average value	Address of the word element that stores the mean value	-	INT/DINT
n	Data length	Immediate	1 to 256	INT/DINT

Table 3–68 List of elements

Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The MEAN instruction calculates the mean value of n variables starting from S by dividing the sum of the variables by n and then stores the result in D.

The remainder (if any) is discarded.

A calculation error occurs when n falls beyond the range of 1 to 256.

Instruction Example

(D10 + D11 + D12 + D13)/2 = D20

Assume that D10 is K5, D11 is K5, D12 is K15, and D13 is K52. Then D20 is K19, and the remainder 1 is discarded.

3.5.5.4 LIMIT

The LIMIT instruction sets the upper and lower limits of an input value to control the output.
LIMIT – Upper/Lower limit control

16-bit Instruction	LIMIT: Continuous execution/LIMITP: Pulse execution			
32-bit Instruction	DLIMIT: Continuous execution/DLIMITP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Lower limit	Minimum output limit	-	INT/DINT
S2	Upper limit	Maximum output limit	-	INT/DINT

S3	Input value	Input value to be controlled by lower and upper limits	-	INT/DINT
D	Output value	Start number of elements that store an output value under lower/upper limit control	-	INT/DINT

Table 3–69 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction

  The LIMIT instruction sets the upper and lower limits in [S1] and [S2] for the input in [S3] to control the output in [D].

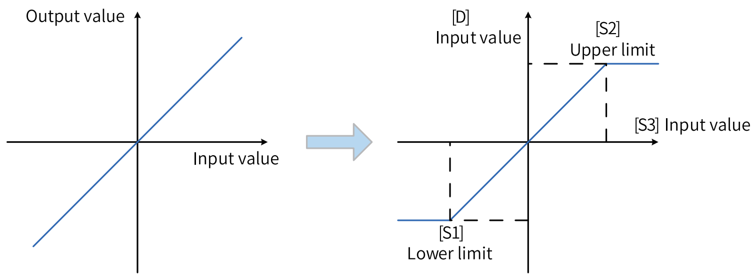
  When the input is less than the lower limit ([S1] > [S3]), [S1] is used as the output ([D]).

  When the input is greater than the upper limit ([S2] < [S3]), [S2] is used as the output ([D]).

  When the input falls between the upper and lower limits ([S1] ⩽ [S3] ⩽ [S2]), [S3] is used as the output ([D]).

  When controlling the output value using only the upper limit, set the lower limit specified in S1 to the minimum 16-bit signed value, that is, –32,768.

  When controlling the output value using only the lower limit, set the upper limit specified in S2 to the maximum 16-bit signed value, that is, 32767.



- 32-bit instruction

  The LIMIT instruction sets the upper and lower limits in [S1+1, S1] and [S2+1, S2] for the input in [S3+1, S3] to control the output in [D+1, D].

  When the input is less than the lower limit ([S1+1, S1] > [S3+1, S3]), [S1+1, S1] is used as the output ([D+1, D]).

  When the input is greater than the upper limit ([S2+1, S2] < [S3+1, S3]), [S2+1, S2] is used as the output ([D+1, D]).

  When the input falls between the upper and lower limits ([S1+1, S1] ⩽ [S3+1, S3] ⩽ [S2+1, S2]), [S3+1, S3] is used as the output ([D+1, D]).

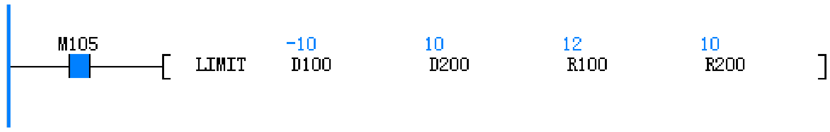
When controlling the output value using only the upper limit, set the lower limit specified in [S1+1, S1] to the minimum 32-bit signed value, that is, –2,147,483,648.

When controlling the output value using only the lower limit, set the upper limit specified in [S2+1, S2] to the maximum 32-bit signed value, that is, 2,147,483,647.

An error is returned in the following conditions:

The lower limit is greater than the upper limit in the 16-bit/32-bit instruction.

Instruction Example



3.5.5.5 BZAND

The BZAND instruction controls an output value based on whether the input value is within the specified dead zone range (defined by upper and lower limits).

BZAND – Dead zone control

16-bit Instruction	BZAND: Continuous execution/BZANDP: Pulse execution			
32-bit Instruction	DBZAND: Continuous execution/DBZANDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Lower limit	Lower limit of a dead zone (with no output)	-	INT/DINT
S2	Upper limit	Upper limit of a dead zone (with no output)	-	INT/DINT
S3	Input value	Input value subject to dead zone control	-	INT/DINT
D	Output value	Number of the element that stores an output value under dead zone control	-	INT/DINT

Table 3–70 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction

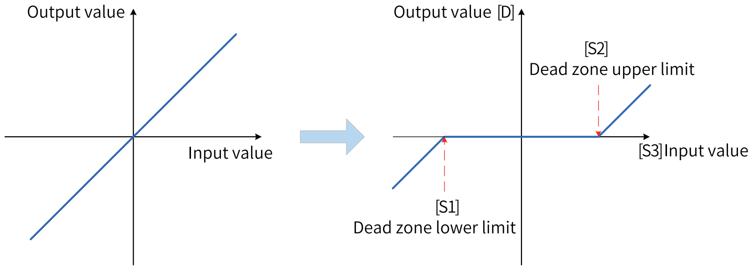
  The BZAND instruction sets a dead zone range in [S1] and [S2] for an input value in [S3] to control the output in [D].

  The output value is controlled as follows:

When the input is less than the lower limit of the dead zone ([S1] > [S3]), ([S3] – [S1]) is used as the output ([D]).

When the input is greater than the upper limit of the dead zone ([S2] < [S3]), ([S3] – [S2]) is used as the output ([D]).

When the input falls between the upper and lower limits of the dead zone ([S1] ⩽ [S3] ⩽ [S2]), 0 is used as the output ([D]).



- 32-bit instruction

The BZAND instruction sets a dead zone range in [S1+1, S1] and [S2+1, S2] for an input value in [S3 +1, S3] to control the output in [D+1, D].

When the input is less than the lower limit of the dead zone ([S1+1, S1] > [S3+1, S3]), ([S3+1, S3] – [S1+1, S1]) is used as the output ([D+1, D]).

When the input is greater than the upper limit of the dead zone ([S2+1, S2] < [S3+1, S3]), ([S3+1, S3] – [S2+1, S2]) is used as the output ([D+1, D]).

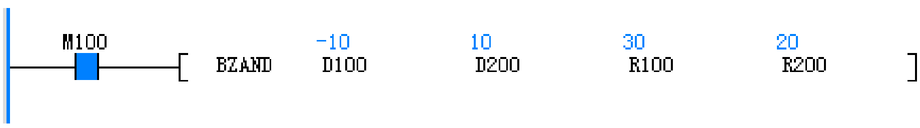
When the input falls between the upper and lower limits of the dead zone ([S1+1, S1] ⩽ [S3+1, S3] ⩽ [S2+1, S2]), 0 is used as the output ([D+1, D]).

Data overflow conforms to cyclical processing during instruction execution. That is, the minimum value is reached when the maximum value increases by 1; the maximum value is reached when the minimum value decreases by 1.

An error is returned in the following conditions:

The lower limit is greater than the upper limit in the 16-bit/32-bit instruction.

Instruction Example



3.5.5.6 ZONE

The ZONE instruction controls an output value by using the specified deviation based on whether the input value is positive or negative.

ZONE – Zone control

16-bit Instruction	ZONE: Continuous execution/ZONEP: Pulse execution			
32-bit Instruction	DZONE: Continuous execution/DZONEP: Pulse execution			
Operand	Name	Description	Range	Data Type

S1	Negative deviation	Negative deviation (which can be a positive or negative number or 0) added to an input value	-	INT/DINT
S2	Positive deviation	Positive deviation (which can be a positive or negative number or 0) added to an input value	-	INT/DINT
S3	Input value	Input value subject to zone control	-	INT/DINT
D	Output value	Start number of elements that store an output value under zone control	-	INT/DINT

Table 3–71 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction

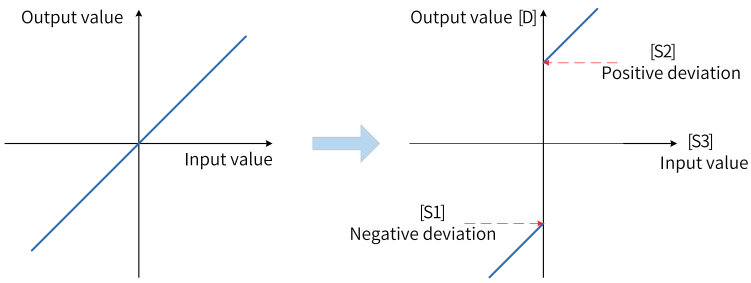
  The ZONE instruction adds the value in [S2] or [S1] to the input value in [S3] based on whether the input is positive or negative and stores the result in [D].

  When the input is less than 0 ([S3] < 0), ([S3] + [S1]) is used as the output ([D]).

  When the input is greater than 0 ([S3] > 0), ([S3] + [S2]) is used as the output ([D]).

  When the input is 0 ([S3] = 0), 0 is used as the output ([D]).

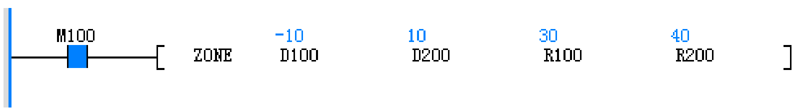
  The instruction is executed as follows:



- 32-bit instruction

  The ZONE instruction adds the value in [S2+1, S2] or [S1+1, S1] to the input value in [S3+1, S3] based on whether the input is positive or negative and stores the result in [D+1, D].

Instruction Example



3.5.5.7 SCL

The SCL instruction determines the coordinates of an input value based on the specified data table and outputs the result.

SCL – Coordinate determination (coordinates of different points)

16-bit Instruction	SCL: Continuous execution/SCLP: Pulse execution			
32-bit Instruction	DSCL: Continuous execution/DSCLP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Input value	Input value for which the coordinate is to be determined, or number of the element that stores the input value	-	INT/DINT
S2	Table data	Start number of elements that store the conversion table used for coordinate determination	1 to 256	INT/DINT, array*indeterminate
D	Output value	Number of the element that stores the output value under coordinate control	-	INT/DINT

Table 3–72 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction

  The SCL instruction determines the output value ([D]) corresponding to the input value in [S1] based on the graph determined by the table data in [S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

  The instruction is executed as follows:

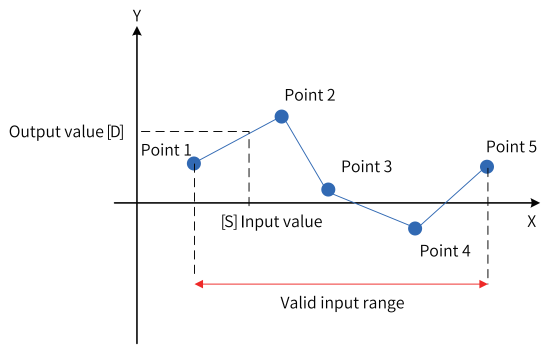


Table 3–73 [S2] in the 16-bit instruction

Setting		Element for Storage
Assume that the number of coordinate points is 5.		[S2]
Point 1	X coordinate	[S2+1]
	Y coordinate	[S2+2]
Point 2	X coordinate	[S2+3]
	Y coordinate	[S2+4]
Point 3	X coordinate	[S2+5]
	Y coordinate	[S2+6]
Point 4	X coordinate	[S2+7]
	Y coordinate	[S2+8]
Point 5	X coordinate	[S2+9]
	Y coordinate	[S2+10]

- 32-bit instruction

  The SCL instruction determines the output value ([D+1, D]) corresponding to the input value in [S1 +1, S1] based on the graph determined by the table data in [S2+1, S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

Table 3–74 [S2] in the 32-bit instruction

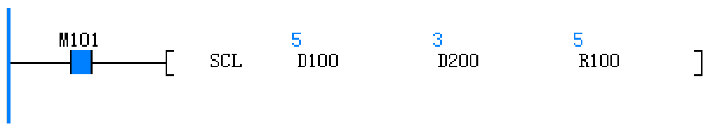
Setting		Element for Storage
Assume that the number of coordinate points is 5.		[S2+1, S]
Point 1	X coordinate	[S2+3, S2+2]
	Y coordinate	[S2+5, S2+4]
Point 2	X coordinate	[S2+7, S2+6]
	Y coordinate	[S2+9, S2+8]
Point 3	X coordinate	[S2+11, S2+10]
	Y coordinate	[S2+13, S2+12]
Point 4	X coordinate	[S2+15, S2+14]
	Y coordinate	[S2+17, S2+16]
Point 5	X coordinate	[S2+19, S2+18]
	Y coordinate	[S2+21, S2+20]

An error is returned in the following conditions:

The x coordinates of table data are not sorted in ascending order.

The value in [S1] is beyond the range of the table data.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	5
D200	16-bit INT	Decimal	3
D201	16-bit INT	Decimal	0
D202	16-bit INT	Decimal	0
D203	16-bit INT	Decimal	10
D204	16-bit INT	Decimal	10
D205	16-bit INT	Decimal	20
D206	16-bit INT	Decimal	0
	16-bit INT	Decimal	
R100	16-bit INT	Decimal	5
	16-bit INT	Decimal	

3.5.5.8 SCL2

The SCL2 instruction determines the coordinates of an input value based on the specified data table and outputs the result.

SCL2 – Coordinate determination 2 (X and Y coordinates)

16-bit Instruction	SCL2: Continuous execution/SCL2P: Pulse execution			
32-bit Instruction	DSCL2: Continuous execution/DSCL2P: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Input value	Input value for which the coordinate is to be determined, or number of the element that stores the input value	-	INT/DINT
S2	Table data	Start number of elements that store the conversion table used for coordinate determination	1 to 256	INT/DINT, array*indetermi-nate
D	Output value	Number of the element that stores the output value under coordinate control	-	INT/DINT

Table 3–75 List of elements

Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction

  The SCL2 instruction determines the output value ([D]) corresponding to the input value in [S1] based on the graph determined by the table data in [S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

  The instruction is executed as follows:

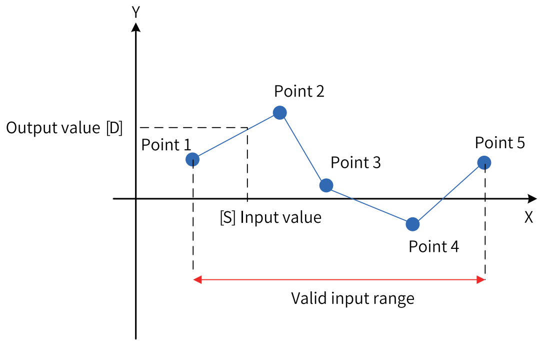


Table 3–76 [S2] in the 16-bit instruction

Setting		Element for Storage
Assume that the number of coordinate points is 5.		[S2]
X coordinate	Point 1	[S2+1]
	Point 2	[S2+2]
	Point 3	[S2+3]
	Point 4	[S2+4]
	Point 5	[S2+5]
Y coordinate	Point 1	[S2+6]
	Point 2	[S2+7]
	Point 3	[S2+8]
	Point 4	[S2+9]
	Point 5	[S2+10]

- 32-bit instruction

  The SCL2 instruction determines the output value ([D+1, D]) corresponding to the input value in [S1 +1, S1] based on the graph determined by the table data in [S2+1, S2]. If the output value is not an integer, the digit in the first decimal place is rounded.

Table 3–77 [S2+1, S2] in the 32-bit instruction

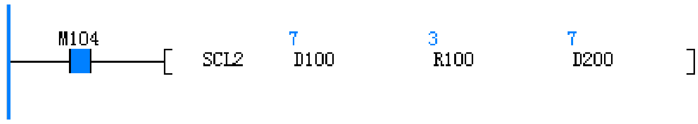
Setting		Element for Storage
Assume that the number of coordinate points is 5.		[S2+1, S2]
X coordinate	Point 1	[S2+3, S2+2]
	Point 2	[S2+5, S2+4]
	Point 3	[S2+7, S2+6]
	Point 4	[S2+9, S2+8]
	Point 5	[S2+11, S2+10]
Y coordinate	Point 1	[S2+13, S2+12]
	Point 2	[S2+15, S2+14]
	Point 3	[S2+17, S2+16]
	Point 4	[S2+19, S2+18]
	Point 5	[S2+21, S2+20]

An error is returned in the following conditions:

The x coordinates of table data are not sorted in ascending order.

The value in [S1] is beyond the range of the table data.

Instruction Example



3.5.6 Exponent Operation Instructions

3.5.6.1 Instruction List

The following table lists the exponent operation instructions.

Instruction Category	Instruction	Function
Exponent operation instruction	EXP	Binary floating-point exponentiation operation
	LOGE	Binary floating-point natural logarithm operation
	LOG	Binary floating-point common logarithm operation
	ESQR	Binary floating-point square root operation
	SQR	Binary data square root operation
	POW	Floating-point weight instruction

3.5.6.2 EXP

The EXP instruction performs exponentiation of a binary floating-point number with the base of mathematical constant e (2.71828).

EXP – Floating-point exponentiation operation

16-bit Instruction	-			
32-bit Instruction	DEXP: Continuous execution/DEXPP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable used as the exponent	-	REAL
D	Operation result	Unit that stores the result of exponentiation	-	REAL

Table 3–78 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

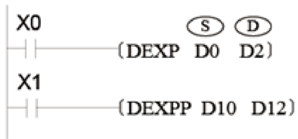
The EXP instruction performs exponentiation of a binary floating-point number with the base of mathematical constant e (2.71828). Where,

- S is the binary floating-point variable used as the exponent.
- D is the unit that stores the result of exponentiation.

Note

An operation error will occur when the operation result does not satisfy the following condition: $2^{-126} \leqslant$ Operation result < 2^{128}.

Instruction Example



When X0 is ON, exponentiation is performed for the binary floating-point value in (D1, D0) with e as the base, and the result is stored in (D3, D2). $e^{(D1, D0)}$→(D3, D2)

3.5.6.3 LOG

The LOG instruction calculates the common logarithm of a binary floating-point number with base 10. LOG – Floating-point common logarithm operation

16-bit Instruction	-			
32-bit Instruction	DLOG: Continuous execution/DLOGP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which common logarithm is to be calculated	-	REAL
D	Operation result	Unit that stores the operation result	-	REAL

Table 3–79 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

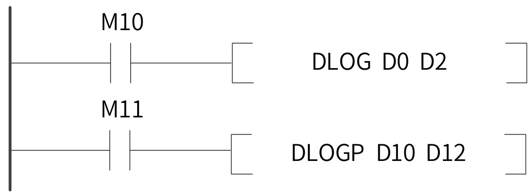
The LOG instruction calculates the common logarithm of a binary floating-point number with base 10. Where,

- S is the binary floating-point variable for which common logarithm is to be calculated.
- D is the unit that stores the natural logarithm result.

Note

The value in S must be positive. If it is 0 or negative, an operation error will occur.

Instruction Example



When M10 is ON, the common logarithm for the binary floating-point value in (D1, D0) is calculated with base 10 and the operation result is stored in (D3, D2).

Log_{10} (D1, D0)→(D3, D2)

3.5.6.4 LOGE

The LOGE instruction calculates the natural logarithm of a binary floating-point number with the base of mathematical constant e (2.71828).

LOGE – Floating-point natural logarithm operation

16-bit Instruction	-			
32-bit Instruction	DLOGE: Continuous execution/DLOGEP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable for which the natural logarithm is to be calculated	-	REAL
D	Operation result	Unit that stores the operation result	-	REAL

Table 3–80 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

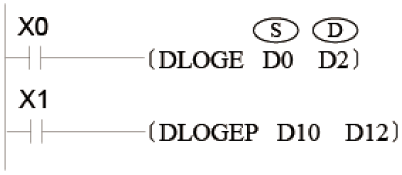
The LOGE instruction calculates the natural logarithm of a binary floating-point number with the base of mathematical constant e (2.71828). Where,

- S is the binary floating-point variable for which natural logarithm is to be calculated.
- D is the unit that stores the natural logarithm result.

Note

The value in S must be positive. If it is 0 or negative, an operation error will occur.

Instruction Example



When X0 is ON, the natural logarithm for the binary floating-point value in (D1, D0) is calculated with the base of mathematical constant e and the operation result is stored in (D3, D2).

$$\log_e(D1、D0) \Longrightarrow (D3、D2)$$

The formula for converting the natural logarithm to common logarithm is as follows (0.4342945 is used for common logarithm division):

$$10^{x} = e^{\frac{x}{0.4342945}}$$

3.5.6.5 ESQR

The ESQR instruction calculates the square root of a binary floating-point number.
ESQR – Floating-point square root operation

16-bit Instruction	-			
32-bit Instruction	DESQR: Continuous execution/DESQRP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable of which the square root is to be calculated	-	REAL
D	Operation result	Unit that stores the calculated square root	-	REAL

Table 3–81 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

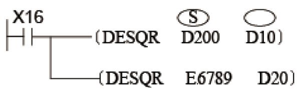
Function and Instruction Description

The ESQR instruction calculates the square root of a binary floating-point number.

The zero flag (M8020) is set if the operation result is 0.

The value in S must be positive. If it is negative, a calculation error occurs.

Instruction Example



The square root result of the binary floating-point number ($\sqrt{(D201,D200)}$) is stored in (D11, D10).

The square root of the binary floating-point number E6789 is calculated, and the result is stored in (D21, D20).

3.5.6.6 SQR

The SQR instruction calculates the square root of an integer.
SQR – Square root operation

16-bit Instruction	SQR: Continuous execution/SQRP: Pulse execution			
32-bit Instruction	DSQR: Continuous execution/DSQRP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Data of which the square root is to be calculated, or address of the word element that stores the data	-	INT/DINT
D	Operation result	Address of the word element that stores the calculated square root	-	INT/DINT

Table 3–82 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

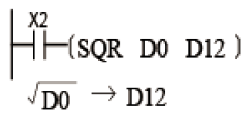
The SQR instruction calculates the square root of S in binary format and stores the result in D.

The value in S must be positive. If it is negative, an operation error occurs, the error flag M8067 is set to ON, and the instruction is not executed.

The operation result in D must be an integer. The borrow flag M8021 is set to ON when the decimal places (if any) of the operation result are discarded.

The zero flag M8020 is set to ON when the operation result is 0.

Instruction Example



If D0 is K100, D12 is K10 when X2 is set to ON.

If D0 is K110, D12 is K10 (the decimal places are discarded) when X2 is set to ON.

3.5.6.7 POW

The POW instruction performs a mathematical operation in which the binary floating-point number in [S1+1, S1] is raised to the power in [S2+1, S2] and stores the result in [D+1, D].
POW – Floating-point weight instruction

16-bit Instruction	-			
32-bit Instruction	DPOW: Continuous execution/DPOWP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Base	Start address of elements that store the base, which cannot be 0	-	REAL
S2	Power	Start address of elements that store the power	-	REAL
D	Result	Start address of elements that store the operation result	-	REAL

Table 3–83 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

Since the POW instruction uses only floating-point numbers, the values in [S1] and [S2] must be converted to floating-point numbers.

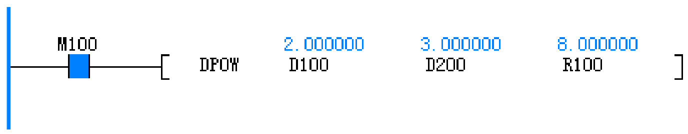
1. The carry flag M8022 is set to ON if the absolute value of the operation result is greater than the maximum floating-point value.

2. The borrow flag M8021 is set to ON if the absolute value of the operation result is less than the minimum floating-point value.

3. The zero flag M8020 is set to ON if the operation result is 0.

Instruction Example

If [S1] is 2 and [S2] is 3, then [D] = $2^3 = 8$.



3.6 Data Processing Instructions

3.6.1 Data Conversion Instructions

3.6.1.1 Instruction List

The following table lists the data conversion instructions.

Instruction Category	Instruction	Function
Data conversion instruction	INT	Conversion from binary floating-point number into BIN integer
	BCD	Conversion from binary into BCD
	BIN	Conversion from BCD into binary
	FLT	Conversion from binary into binary floating-point
	EBCD	Conversion from binary floating-point into decimal floating-point
	EBIN	Conversion from decimal floating-point into binary floating-point
	DABIN	Conversion from decimal ASCII into BIN
	BINDA	Conversion from BIN into decimal ASCII
	WTOB	Conversion from word to byte
	BITW	Conversion from bit to word
	BTOW	Conversion from byte to word
	WBIT	Conversion from word to bit
	WTODW	Conversion from word to dword
	DWTOW	Conversion from dword to word
	MCPY	Data copy (memory copy, type conversion)
	MSET	Data setting (memory setting and reset)
	UNI	4-bit combination of 16-bit data
	DIS	4-bit separation of 16-bit data
	ASCI	Conversion from HEX into ASCII
	HEX	Conversion from ASCII into HEX
	DECO	Data decoding
	ENCO	Data encoding

3.6.1.2 INT

The INT instruction rounds a binary floating-point number by discarding the decimal places and stores the result in D.

INT – Conversion from floating-point number to binary integer

16-bit Instruction	INT: Continuous execution/INTP: Pulse execution			
32-bit Instruction	DINT: Continuous execution/DINTP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable to be rounded	-	REAL, fixed to 32 bits
D	Operation result	Unit that stores the resulting binary integer	-	INT/DINT

Table 3–84 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√[1]	-
D	-	-	-	√	√	√	-	-	-

Note

[1]: The 16-bit instruction does not support the constant E.

Function and Instruction Description

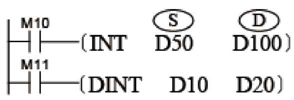
The INT instruction rounds a binary floating-point number by discarding the decimal places and stores the result in D.

- M8020 is set when S is 0.
- The borrow flag M8021 is set when the absolute value of S is less than or equal to 1 (|S| ⩽ 1).

The carry flag M8022 is set if the operation result falls beyond the following range (which results in an overflow):

- 16-bit instruction: –32,768 to +32,767
- 32-bit instruction: –2,147,483,648 to +2,147,483,647

Instruction Example



The floating-point number in (D51, D50) is rounded and then stored in D100.

The floating-point number in (D11, D10) is rounded and then stored in (D21, D20).

Note the difference in storing the operation result between the INT and DINT instructions.

3.6.1.3 BCD

The BCD instruction converts binary data into binary coded decimal (BCD) data.
BCD – Conversion from binary into BCD

16-bit Instruction	BCD: Continuous execution/BCDP: Pulse execution			
32-bit Instruction	DBCD: Continuous execution/DBCDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary data, or address of the word element that stores the binary data	-	INT/DINT
D	Conversion result	Address of the word element that stores the conversion result	-	INT/DINT

Table 3–85 List of elements

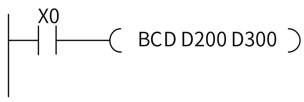
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

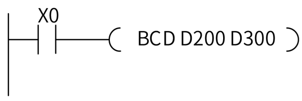
Function and Instruction Description

The BCD instruction requires contact driving and has two operands. It converts the binary number in S to a BCD number and stores the result in D. The BCD instruction is generally used for data format conversion before display.

- For the 16-bit instruction, the conversion result ranges from 0 to 9999. An error occurs when the conversion result exceeds 9999.
- For the 32-bit instruction, the conversion result ranges from 0 to 99,999,999. An error occurs when the conversion result exceeds 99,999,999.

Instruction Example





The binary number in D200 is converted into a BCD value, which is then stored in D300.

If the value in D200 is H000E (hexadecimal) or K14 (decimal), the conversion result in D300 is 10100 (binary number).

3.6.1.4 BIN

The BIN instruction converts BCD data into binary data.
BIN – Conversion from BCD into binary

16-bit Instruction	BIN: Continuous execution/BINP: Pulse execution			
32-bit Instruction	DBIN: Continuous execution/DBINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	BCD data, or address of the word element that stores the data	-	INT/DINT
D	Conversion result	Address of the word element that stores the conversion result	-	INT/DINT

Table 3–86 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

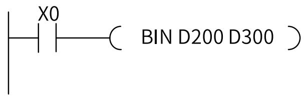
Function and Instruction Description

The BIN instruction requires contact driving and has two operands. It converts the BCD value in S to a binary number and stores the result in D. This instruction is generally used to convert the data (for example, encoder disk setting) read by external ports into binary data that can be directly used in operation.

The BCD value in S ranges from 0 to 9999 for the 16-bit instruction or from 0 to 99,999,999 for the 32-bit instruction.

If the data in S is not in BCD format (Hex indicates any digit beyond the range of 0 to 9), an operation error occurs.

Instruction Example



The BCD value in D200 is converted into a binary value, which is then stored in D300.

3.6.1.5 FLT

The FLT instruction converts a binary integer into a binary floating-point number.
FLT – Conversion from binary integer to binary floating-point

16-bit Instruction	FLT: Continuous execution/FLTP: Pulse execution			
32-bit Instruction	DFLT: Continuous execution/DFLTP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Integer	Binary integer to be converted, or address of the word element that stores the binary integer	-	INT/DINT
D	Floating-point number	Address of the word element that stores the floating-point number after conversion	-	REAL, fixed to 32 bits

Table 3–87 List of elements

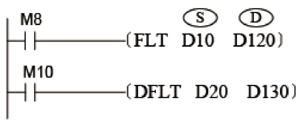
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The FLT instruction converts the integer in S into a floating-point number and stores the result in D and D+1.

This instruction implements the inverse function of the INT instruction (converting binary floating-point values into binary integers).

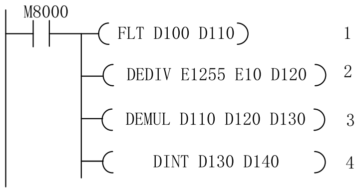
Instruction Example 1



- When M8 is ON, the 16-bit binary integer in D10 is converted into a binary floating-point number. The result is stored in (D121, D120).
- When M10 is ON, the 32-bit binary integer in (D21, D20) is converted into a binary floating-point number. The result is stored in (D131, D130).

Instruction Example 2

Instructions are executed to multiply D100 by 125.5 and convert the calculation result into an integer.



- 1. Convert the value in D100 into a floating-point number and store the result in D110.
- 2. Store the conversion result of 125.5 in D120.
- 3. Multiply the value in D110 by that in D120 and store the result in D130.
- 4. Convert the calculation result into an integer and store it in D140.

3.6.1.6 EBCD

EBCD – Conversion from binary floating-point to decimal floating-point

16-bit Instruction	-			
32-bit Instruction	DEBCD: Continuous execution/DEBCDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Binary floating-point variable	-	REAL
D	Operation result	Unit that stores the decimal floating-point number after conversion	-	DINT

Table 3–88 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EBCD instruction converts a binary floating-point number into a decimal floating-point number.

Instruction Example

The binary floating-point number in (D3, D2) is converted into a decimal floating-point number, which is then stored in (D11, D10).

For the binary floating-point number in (D3, D2), the real number occupies 23 bits, the exponent occupies eight bits, and the sign occupies one bit. For the decimal floating-point number in (D11, D10), the exponent (D3) and real number (D2) are expressed as D2 x 10^{D3} in scientific notation.

In floating-point operations of the PLC, all data is handled in binary floating-point format. Binary floating-point numbers are converted into decimal equivalents for easy monitoring.

3.6.1.7 EBIN

EBIN – Conversion from decimal floating-point to binary floating-point

16-bit Instruction	-			
32-bit Instruction	DEBIN: Continuous execution/DEBINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Decimal floating-point variable	-	DINT
D	Result	Unit that stores the binary floating-point number after conversion	-	REAL

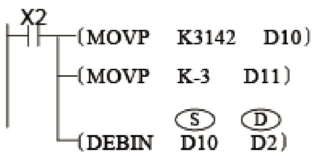
Table 3–89 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EBIN instruction converts a decimal floating-point number into a binary floating-point number.

Instruction Example



The decimal floating-point number 3.142 in (D11, D10) is converted into a binary floating-point number, which is then stored in (D3, D2).

3.6.1.8 DABIN

The DABIN instruction converts numeric data expressed in decimal ASCII codes (30H to 39H) into binary data.

DABIN – Conversion from decimal ASCII into BIN

16-bit Instruction	DABIN: Continuous execution/DABINP: Pulse execution			
32-bit Instruction	DDABIN: Continuous execution/DDABINP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Input value	Start number of elements that store the ASCII code to be converted to a binary number	-	INT/DINT, array*3
D	Output value	Number of the element that stores the conversion result	-	INT/DINT

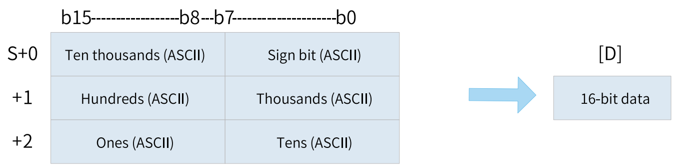
Table 3–90 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	√	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

16-bit instruction

The DABIN instruction converts the decimal ASCII code (30H to 39H) stored in [S] to [S+2] into a 16-bit binary number. The result is stored in [D].



The value stored in [S] to [S+2] ranges from –32,768 to +32,767.

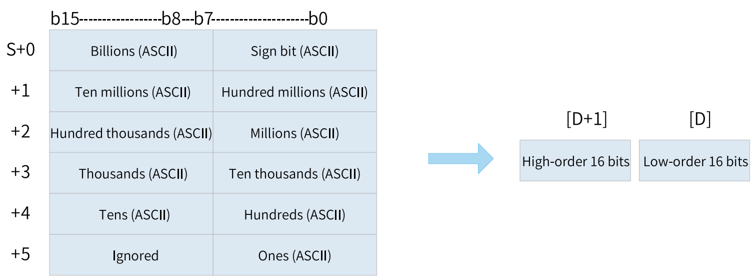
When the number to be converted is positive, the sign (lowest byte) is set to 20H (space); when it is negative, the sign is set to 2DH (–).

The ASCII code of each digit falls within the range of 30H to 39H.

When the ASCII code of each digit is 20H (space) or 00H (NULL), it is handled as 30H.

32-bit instruction

The DABIN instruction converts the decimal ASCII code (30H to 39H) stored in [S] to [S+5] into a 32-bit binary number. The result is stored in [D+1, D].



The value stored in [S] to [S+5] ranges from –2,147,483,648 to +2,147,483,647. The high-order byte in [S +5] is ignored.

The ASCII code of each digit falls within the range of 30H to 39H.

When the ASCII code of each digit is 20H (space) or 00H (NULL), it is handled as 30H.

Errors

An operation error occurs in the following conditions.

- The sign data is not 20H (space) or 2DH (–).
- The ASCII code of each digit is not 20H (space), 00H (NULL), or a value between 30H and 39H.
- The value to be converted is beyond the value range of the 16-bit or 32-bit signed number.
- The element [S+2] (16-bit operation) or [S+5] (32-bit operation) is out of range.

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x202D
D101	16-bit INT	Hex	0x3220
D102	16-bit INT	Hex	0x3637
	16-bit INT	Hex	
R100	16-bit INT	Hex	-276

3.6.1.9 BINDA

The BINDA instruction converts binary data into decimal ASCII codes (30H to 39H).

BINDA – Conversion from BIN to decimal ASCII

16-bit Instruction	BINDA: Continuous execution/BINDAP: Pulse execution			
32-bit Instruction	DBINDA: Continuous execution/DBINDAP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Input value	Number of the element that stores the binary number to be converted to ASCII code	-	INT/DINT, array*4
D	Output value	Number of the element that stores the conversion result	-	INT/DINT

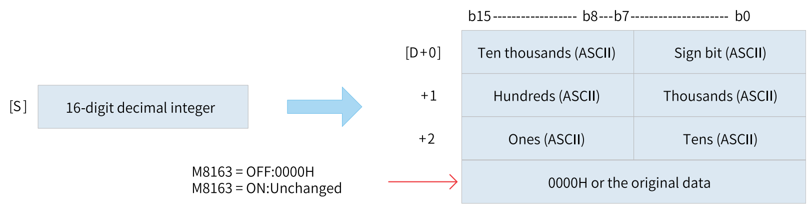
Table 3–91 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	√	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

16-bit instruction

The BINDA instruction converts each digit of the 16-bit binary data in [S] into an ASCII code (30H to 39H) in decimal format and stores the result in elements starting from [D].



The 16-bit data in [S] ranges from –32,768 to +32,767.

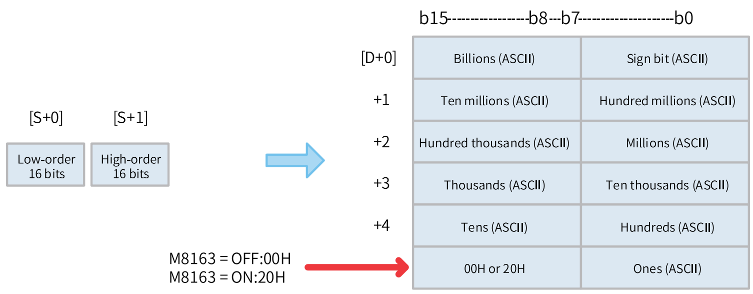
The operation result is as follows:
When the 16-bit number is positive, the sign bit is set to 20H (space). When it is negative, the sign bit is set to 2DH (–).

When 0 exists on the left of valid digits, the sign bit is set to 20H (space).

The value in [D+3] is determined based on whether M8163 is set to ON or OFF.

32-bit instruction

The BINDA instruction converts each digit of the 32-bit binary data into an ASCII code (30H to 39H) in decimal format and stores the result in elements starting from [D].



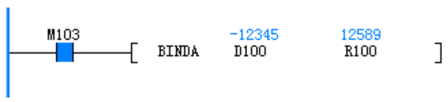
The 32-bit data in [S+1, S] ranges from –2,147,483,648 to +2,147,483,647.

The operation result is as follows:
When the 32-bit number is positive, the sign bit is set to 20H (space). When it is negative, the sign bit is set to 2DH (–).

When 0 exists on the left of valid digits, the sign bit is set to 20H (space).

The value in [D+5] is determined based on whether M8163 is set to ON or OFF.

Instruction Example



	Element Name	Data Type	Display Format	Current Value
1	... D100	INT	Dec	-12345
2	...			
3	... R100	INT	Hex	0x312D
4	... R101	INT	Hex	0x3332
5	... R102	INT	Hex	0x3534

3.6.1.10 WBIT

The WBIT instruction assigns the value of a word element to a combination of bit elements.

WBIT – Conversion from word to bit

Instruction	Name	LD Expression	LiteST Expression
WBIT	Conversion from word to bit	-[WBIT ??? ??? ???]	WBIT(???, ???, ???);

16-bit Instruction	WBIT: Continuous execution/WBITP: Pulse execution			
32-bit Instruction	DWBIT: Continuous execution/DWBITP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Value to be assigned to bit elements	-	INT/DINT
D	Bit element	Start number of bit elements	-	BOOL, array*n
n	Bit element quantity	Number of bit elements	1 to 16/1 to 32	INT/DINT

Table 3–92 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	√[1]	√	√	-	-	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WBIT instruction converts the binary number in S into bit states and assigns the conversion result to n bits starting from D.

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

		Element Name	Data Type	Display Format	Current Value
1	...	M0	BOOL	Binary	ON
2	...	M1	BOOL	Binary	OFF
3	...	M2	BOOL	Binary	OFF
4	...	M3	BOOL	Binary	OFF
5	...	M4	BOOL	Binary	OFF
6	...	M5	BOOL	Binary	OFF
7	...	M6	BOOL	Binary	OFF
8	...	M7	BOOL	Binary	ON
9	...	M8	BOOL	Binary	OFF
10	...	M9	BOOL	Binary	OFF
11	...	M10	BOOL	Binary	OFF
12	...	M11	BOOL	Binary	ON
13	...	M12	BOOL	Binary	OFF
14	...	M13	BOOL	Binary	OFF
15	...	M14	BOOL	Binary	OFF
16	...	M15	BOOL	Binary	ON
17	...	D100	INT	Binary	2#1000100010000001

3.6.1.11 UNI

The UNI instruction combines the low-order four bits of consecutive 16-bit data.

UNI – 4-bit combination of 16-bit data

16-bit Instruction	UNI: Continuous execution/UNIP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store the data to be combined	-	INT, array*n
D	Result	Number of the element that stores the data after combination	-	INT
n	Combined data count	Number of data entries to be combined (ranging from 0 to 4; no processing when n is 0)	0, 1 to 4	INT

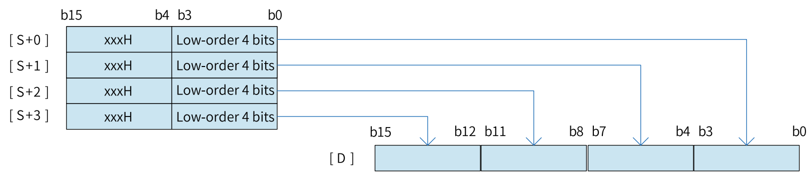
Table 3–93 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The UNI instruction combines the low-order four bits of each of the n 16-bit data entries starting from S into 16-bit data and stores the result in D.

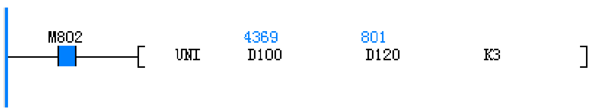
n ranges from 1 to 4. The instruction is not executed when n is 0. When n is 1, 2, or 3, the high-order bits are filled with 0s.



An operation error occurs in the following conditions.

- The element (S) is out of the specified range.
- n is out of the specified range.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1111
D101	16-bit INT	Hex	0x2222
D102	16-bit INT	Hex	0x3333
D104	16-bit INT	Hex	0x0
	16-bit INT	Hex	
D120	16-bit INT	Hex	0x321
	16-bit INT	Hex	

3.6.1.12 DWTOW

The DWTOW instruction assigns the values of 32-bit word elements to 16-bit word elements.

DWTOW – Conversion from dword to word

16-bit Instruction	-			
32-bit Instruction	DWTOW: Continuous execution/DWTOWP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Dword element	Start number of dword elements	-	DINT, array*n
D	Word element	Start number of word elements	-	INT, array*n
n	Element quantity	Number of elements	0, 1 to 256	DINT

Table 3–94 List of elements

Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WTODW instruction converts the 32-bit data in n elements starting from S into 16-bit data and stores the conversion result in n 16-bit registers starting from D. This instruction is used to convert the 32-bit data so that the data can be used in 16-bit instructions.

Note

When the value to be converted in this instruction is greater than the upper limit of 16-bit data, only the low-order bits are retained after conversion.

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

	Element Name	Data Type	Display Format	Current Value
1	... D100	DINT	Decimal	100000
2	... D102	DINT	Decimal	-10000
3	... D104	DINT	Decimal	20000
4	... D106	DINT	Decimal	30000
5	... data_16bit[0]	INT	Decimal	-31072
6	... data_16bit[1]	INT	Decimal	-10000
7	... data_16bit[2]	INT	Decimal	20000
8	... data_16bit[3]	INT	Decimal	30000

3.6.1.13 MCPY

The MCPY instruction assigns data with specified length (in byte) to the target address without any change.

MCPY – Data copy (memory copy, type conversion) instruction

Instruction	Name	LD Expression	LiteST Expression
MCPY	Data copy (memory copy, type conversion)	-[MCPY ??? ??? ???]	MCPY(???, ???, ???);

16-bit Instruction	-			
32-bit Instruction	MCPY: Continuous execution/MCPYP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Address of source data	Start number of elements that store the source data	-	Array*n, structure
D	Address of target data	Start number of elements that store the target data	-	Array*n, structure
n	Length	Data length, in byte	-	-

Table 3–95 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	√	√	-	-	-	-
D	√	√	√	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is a higher-order application and should be used with caution.

This instruction implements the data copy operation with no change in the data. It can also implement memory copy, or even type conversion, if used skillfully.

n is the length of the data to be copied (in byte). For example, when two 16-bit data entries are assigned to a 32-bit data entry, n is 4; when two 32-bit integers are copies to structures or 16-bit integer arrays of the same size, n is 8.

When the operand S or D is a bit element, the addresses of the bit element must be aligned by byte. Otherwise, an addressing error will occur. For example, if the instruction is MCPY M1 M15 K1, the system will report "Invalid variable address: variable non-existent".

Instruction Example

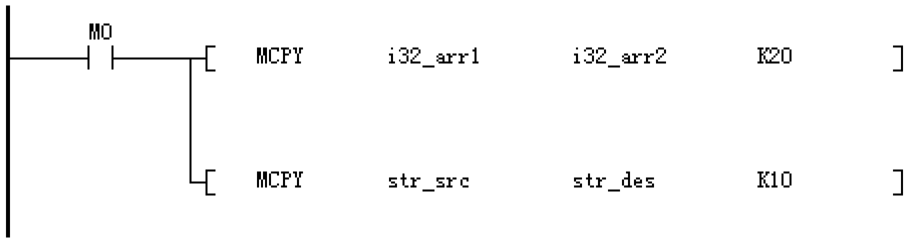
Program running flag

- Running: ON
- Stopped: OFF

If "Size" is 5, then:

D0 Low-order byte =0x34	—>	D100 Low-order byte =0x34
D0 High-order byte =0x12	—>	D100 High-order byte=0x12
D1 Low-order byte =0x78	—>	D101 Low-order byte = 0x78
D1 High-order byte =0x56	—>	D101 High-order byte=0x56
D2 Low-order byte =0x99	—>	D102 Low-order byte =0x99

The MCPY instruction can be used to assign values between arrays and structures of the same type. As shown in the following figure, if the length of the array is DINT[5], data in i32_arr1 can be assigned to i32_arr2 (in byte). As each DINT data entry is 4 bytes, the data to be copied is 20 bytes (4 x 5 = 20). If the data to be assigned is of the structure type, its length and size must be consistent with the size of the structure, so that assignment can be performed properly.



After the program is completely compiled, the lengths of arrays and structures can be obtained in the variable table. The obtained length is in the unit of bit. One byte is equal to 8 bits, one word is equal to 16 bits, and one dword is equal to 32 bits. The formula for calculating the variable length is as follows:

Size = (Length + 15)/16

3.6.1.14 MSET

The MSET instruction assigns data with specified length (in byte) to the target address without any change.

MSET – Data setting (memory setting and reset) instruction

Instruction	Name	LD Expression	LiteST Expression
MSET	Data setting (memory setting and reset)	-[MSET ??? ??? ???]	MSET(???, ???, ???);

16-bit Instruction	-			
32-bit Instruction	MSET: Continuous execution/MSETP: Pulse execution, 13 steps			
Operand	Name	Description	Range	Data Type
S	Data to be set	Value to be set. Only a single byte is valid.	-	-
D	Address of target data	Start number of elements that store the target data	-	Array*n, structure
n	Length	Data length, in byte	-	-

Table 3–96 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	√	√	√	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is a higher-order application and should be used with caution.

This instruction can set data in batches. It can be used to set fixed values in memory or reset the memory. It is typically used to clear structures or arrays.

n is the length of the data to be set (in byte). For example, when 0x12 is assigned to a 32-bit data entry, n is 4; when 0x1234 is set to two 32-bit elements, n is 8.

If you set 0x1234 to a 32-bit data entry in the unit of byte, the result is 0x34343434.

When the operand D is a bit element, the addresses of the bit element must be aligned by byte. Otherwise, an addressing error will occur. For example, if the instruction is MSET D0 M15 K1, the system will report "Invalid variable address: variable non-existent".

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

If "Size" is 5 and D0 is 0x1234, then:

D0 Low-order byte=0x34 —> D100 Low-order byte = 0x34

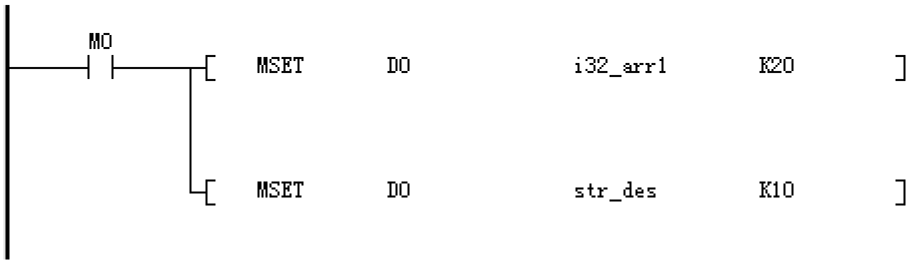
D0 Low-order byte=0x34 —> D100 High-order byte = 0x34

D0 Low-order byte=0x34 —> D101 Low-order byte = 0x34

D0 Low-order byte=0x34 —> D101 High-order byte = 0x34

D0 Low-order byte=0x34 —> D102 Low-order byte = 0x34

The MSET instruction can be used to clear structures and arrays.



After the program is completely compiled, the lengths of arrays and structures can be obtained in the variable table. The obtained length is in the unit of bit. One byte is equal to 8 bits, one word is equal to 16 bits, and one dword is equal to 32 bits. The formula for calculating the variable length is as follows:

Size = (Length + 15)/16

3.6.1.15 DIS

The DIS instruction separates 16-bit data in 4-bit units.
DIS – 4-bit separation of 16-bit data

16-bit Instruction	DIS: Continuous execution/DISP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store the data to be separated	-	INT
D	Result	Number of the element that stores the data after separation	-	INT, array*n
n	Number of data to be separated	Number of data entries to be separated (ranging from 0 to 4; no processing when n is 0)	0, 1 to 4	INT

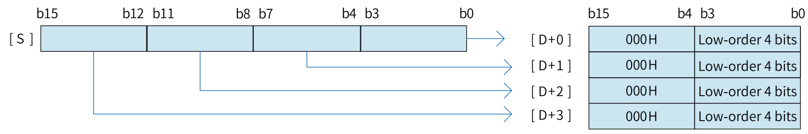
Table 3–97 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The DIS instruction separates the 16-bit data in S in 4-bit units and stores the separation results in the low-order four bits of each of the elements starting from D. The high-order 12 bits are filled with 0s.

n ranges from 1 to 4. The instruction is not executed when n is 0.

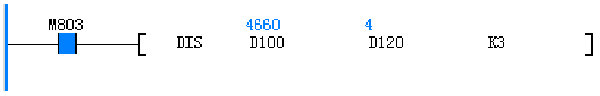


An operation error occurs in the following conditions.

- The element (D) is out of the specified range.
- n is out of the specified range.

Instruction Example

The 16-bit data in D100 is separated in 4-bit units. The results are stored in three consecutive D elements starting from D120.



	Element Name	Data Type	Display Format	Current Value
1	... D100	INT	Hex	0x1234
2	...			
3	... D120	INT	Dec	4
4	... D121	INT	Dec	3
5	... D122	INT	Dec	2
6	... D123	INT	Dec	0

3.6.1.16 BTOW

The BTOW instruction combines the low-order eight bits (low-order byte) of consecutive 16-bit/32-bit data.

BTOW – Conversion from byte to word

16-bit Instruction	BTOW: Continuous execution/BTOWP: Pulse execution			
32-bit Instruction	BTODW: Continuous execution/BTODWP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store the data to be combined by byte	-	INT, array*n
D	Result	Start number of elements that store the data after combination	-	INT/DINT, array*n/2
n	Combined data count	Number of bytes to be combined (n ⩾ 0; no processing when n = 0)	0, 1 to 256	INT/DINT

Table 3–98 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BTOW instruction combines the low-order eight bits of each of the n 16-bit data entries starting from [S] into 16-bit/32-bit data and stores the combination result in elements starting from [D]. The high-order eight bits of each source data entry ([S] and later) are ignored.

[S+0]	High-order byte	1st byte
[S+1]	High-order byte	2nd byte
	High-order byte	3rd byte
	High-order byte	4th byte
	...	...
	High-order byte	(n - 1)th byte
[S+n-1]	High-order byte	nth byte

2nd byte	1st byte	[D+0]
4th byte	3rd byte	[D+1]
...	...	
nth byte	(n - 1)th byte	

An error occurs when the elements starting from [S] or [D] are out of the specified range.

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x99AA
D105	16-bit INT	Hex	0xBBCC
D120	16-bit INT	Hex	0x4422
D121	16-bit INT	Hex	0x8866
D122	16-bit INT	Hex	0xCCAA
D123	16-bit INT	Hex	0x0

3.6.1.17 WTOB

The WTOB instruction separates consecutive 16-bit/32-bit data by byte (eight bits).

WTOB – Conversion from word to byte

16-bit Instruction	WTOB: Continuous execution/WTOBP: Pulse execution			
32-bit Instruction	DWTOB: Continuous execution/DWTOBP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store the data to be separated by byte	-	INT/DINT, array*n/2

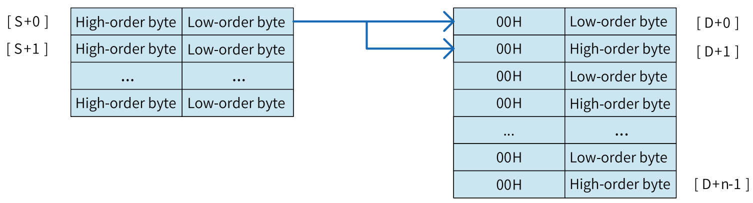
D	Result	Start number of elements that store the data separation result	-	INT, array*n
n	Number of data to be separated	Number of bytes to be separated (n ⩾ 0; no processing when n is 0)	0, 1 to 256	INT/DINT

Table 3–99 List of elements

Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WTOB instruction separates the 16-bit/32-bit data in elements starting from [S] by byte and stores the bytes to the low-order eight bits of each of the n elements starting from [D]. 00H is stored in the high-order eight bits of each element.



An error occurs when the elements starting from [S] or [D] are out of the specified range.

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D120	16-bit INT	Hex	0x22
D121	16-bit INT	Hex	0x11
D122	16-bit INT	Hex	0x44
D123	16-bit INT	Hex	0x33
D124	16-bit INT	Hex	0x66
D125	16-bit INT	Hex	0x55
	16-bit INT	Hex	

3.6.1.18 BITW

The BITW instruction assigns the values of a combination of bit elements to a word element.

BITW – Conversion from bit to word

Instruction	Name	LD Expression	LiteST Expression
BITW	Conversion from bit to word	-[BITW ??? ??? ???]	BITW(???, ???, ???);

16-bit Instruction	BITW: Continuous execution/BITWP: Pulse execution			
32-bit Instruction	BITDW: Continuous execution/BITDWP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Bit element	Start number of bit elements	-	BOOL, array*n
D	Target data	Combination value of bit elements	-	INT/DINT
n	Bit element quantity	Number of bit elements	1 to 16/1 to 32	INT/DINT

Table 3–100 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BITW instruction converts n bits starting from S into a binary number and transfers the conversion result to D.

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

		Element Name	Data Type	Display Format	Current Value
1	...	M0	BOOL	Binary	ON
2	...	M1	BOOL	Binary	OFF
3	...	M2	BOOL	Binary	OFF
4	...	M3	BOOL	Binary	OFF
5	...	M4	BOOL	Binary	OFF
6	...	M5	BOOL	Binary	OFF
7	...	M6	BOOL	Binary	OFF
8	...	M7	BOOL	Binary	ON
9	...	M8	BOOL	Binary	OFF
10	...	M9	BOOL	Binary	OFF
11	...	M10	BOOL	Binary	OFF
12	...	M11	BOOL	Binary	ON
13	...	M12	BOOL	Binary	OFF
14	...	M13	BOOL	Binary	OFF
15	...	M14	BOOL	Binary	OFF
16	...	M15	BOOL	Binary	ON
17	...	D100	INT	Binary	2#1000100010000001

3.6.1.19 WTODW

The WTODW instruction assigns the values of 16-bit word elements to 32-bit word elements.

WTODW – Conversion from word to dword

16-bit Instruction	-			
32-bit Instruction	WTODW: Continuous execution/WTODWP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Word element	Start number of word elements	-	INT, array*n
D	Dword element	Start number of dword elements	-	DINT, array*n
n	Element quantity	Number of elements	0, 1 to 256	DINT

Table 3–101 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WTODW instruction converts the 16-bit data in n elements starting from S into 32-bit data and stores the conversion result in n 32-bit registers starting from D. This instruction is used to convert the 16-bit data so that the data can be used in 32-bit instructions.

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

	Element Name	Data Type	Display Format	Current Value
1	D100	INT	Decimal	10000
2	D101	INT	Decimal	20000
3	D102	INT	Decimal	30000
4	D103	INT	Decimal	-20000
5	data_32bit[0]	DINT	Decimal	10000
6	data_32bit[1]	DINT	Decimal	20000
7	data_32bit[2]	DINT	Decimal	30000
8	data_32bit[3]	DINT	Decimal	-20000

3.6.1.20 ASCI

The ASCI instruction converts the value in S into ASCII codes and store the result in variables starting from D.

ASCI – Conversion from HEX into ASCII

16-bit Instruction	ASCI: Continuous execution/ASCIP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type

S	Data source	Address of the variable or the numeric constant to be converted	-	INT, array*n
D	Conversion result	Start address of elements that store the ASCII codes after conversion	-	INT, array*n
n	Converted character count	Number of converted characters, ranging from 1 to 256	1 to 256	INT

Table 3–102 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ASCI instruction converts the value in S into ASCII codes and store the result in variables starting from D. Where,

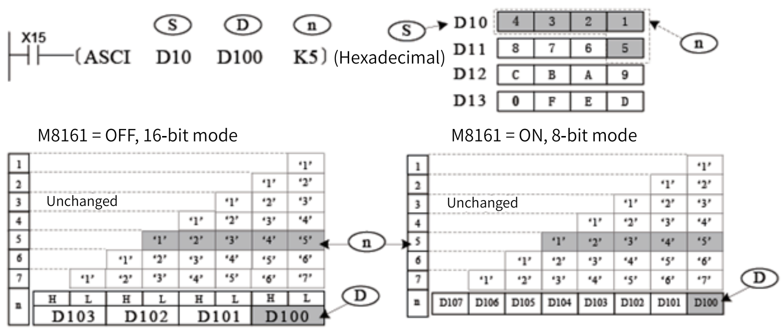
- S is the address of the variable or the numeric constant to be converted.
- D is the start address for storing the ASCII codes after conversion.
- n is the number of converted characters, which ranges from 1 to 256.

Note

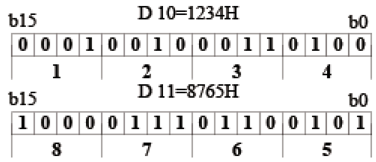
Note during programming that instructions including HEX, ASCI, and CCD share the M8161 mode flag.

The ASCI instruction conforms to the ASCII-hexadecimal mapping table. For example, the ASCII code 0 corresponds to H30 in hexadecimal format, and the ASCII code F corresponds to H46 in hexadecimal format. For details about hexadecimal-ASCII mapping, see " *“5.1 ASCII Code Conversion” on page 734*".

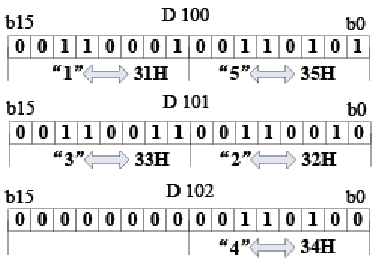
Instruction Example



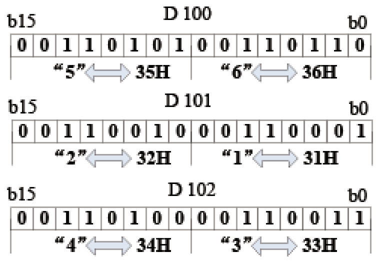
The M8161 flag determines the width mode of the target variable that stores the operation result. When M8161 is OFF, the 16-bit mode is used, whereby the operation result is stored in the high-order and low-order bytes of the variable separately. When M8161 is ON, the 8-bit mode is used, whereby the operation result is stored only in the low-order byte of the variable. In this case, the length of the actually used variable area is increased.



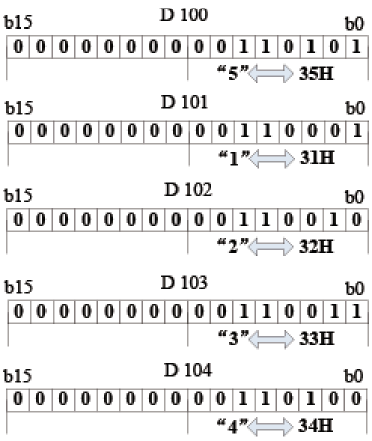
Bit composition when M8161 is OFF and n is 5
Conversion by using D10 to D11



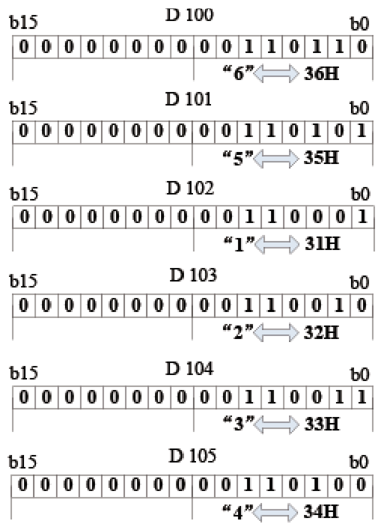
Bit composition when M8161 is OFF and n is 6
Conversion by using D10 to D11



Bit composition when M8161 is ON and n is 5
Conversion by using D10 to D11



Bit composition when M8161 is ON and n is 6
Conversion by using D10 to D11



3.6.1.21 HEX

The HEX instruction converts the values of variables starting from S to hexadecimal equivalents and stores the result in variables starting from D. The number of characters to convert and the storage mode are configurable.

HEX – Conversion from ASCII to HEX

16-bit Instruction	HEX: Continuous execution/HEXP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data source	Start address of variables or numeric constants to be converted. Register variables are converted and separated in units of 32 bits (four ASCII characters).	-	INT, array*n

D	Conversion result	Start address of variables that store the hexadecimal characters after conversion. The occupied variable space is related to S2.	-	INT, array*n
n	Converted character count	Number of converted characters	1 to 256	INT

Table 3–103 List of elements

Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

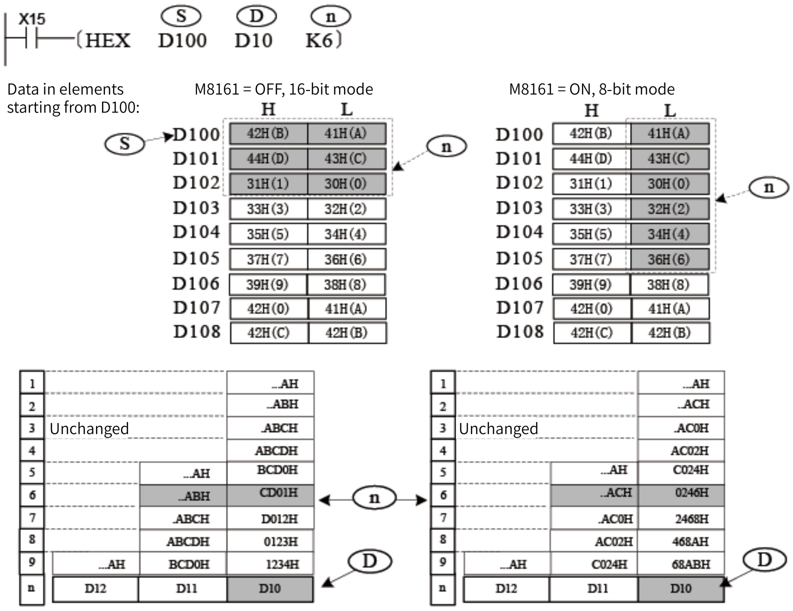
The HEX instruction converts the values of variables starting from S to hexadecimal equivalents and stores the result in variables starting from D. The number of characters to convert and the storage mode are configurable. Where,

- S is the start address of variables or the numeric constants to be converted. Register variables are converted and separated in units of 32 bits (four ASCII characters).
- D is the start address of variables for storing the hexadecimal characters after conversion. The occupied variable space is related to n.
- n is the number of converted characters, which ranges from 1 to 256.

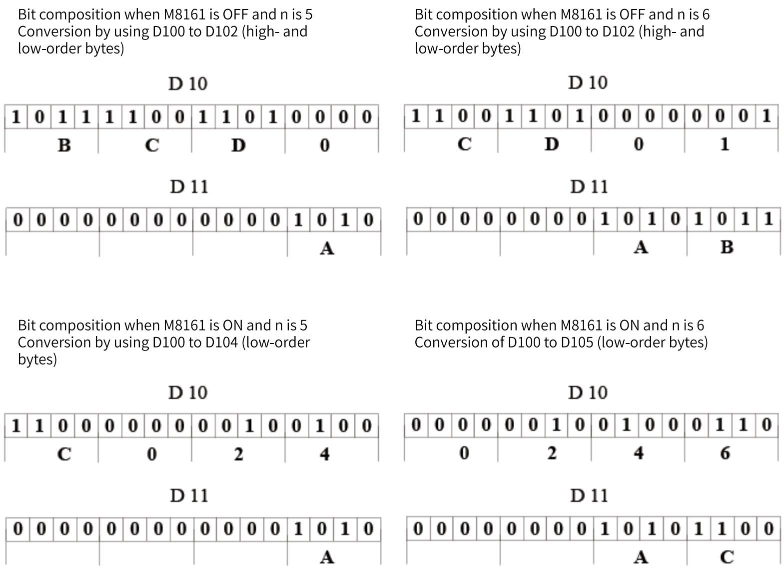
Note

- Note during programming that instructions including HEX, ASCI, and CCD share the M8161 mode flag.
- The source data in S must be ASCII code characters; otherwise, a conversion error occurs.
- If the format of the output data is BCD, BCD-to-BIN conversion is required on the hexadecimal characters after conversion to get the correct value.

Instruction Example



The M8161 flag determines the variable width mode. When M8161 is OFF, the 16-bit mode is used, whereby both the high- and low-order bytes of variables are taken for the operation. When M8161 is ON, the 8-bit mode is used, whereby only the low-order bytes of variables are taken for the operation and the high-order bytes are discarded. In this case, the length of the actually used variable area is increased.



3.6.1.22 DECO

The DECO instruction decodes data and stores the result.

DECO – Data decoding

16-bit Instruction	DECO: Continuous execution/DECOP: Pulse execution
32-bit Instruction	DDECO: Continuous execution/DDECOP: Pulse execution

Operand	Name	Description	Range	Data Type
S	Decoding source	Source data to be decoded	-	INT, DINT
D	Decoding result	Address of the element that stores the decoding result	-	BOOL
n	Bit length of the decoding source	Bit length of source data to be decoded	0 to 8	INT, DINT

Table 3–104 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	√[1]	√	√	-	-	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Note

[1] The X element is not supported.

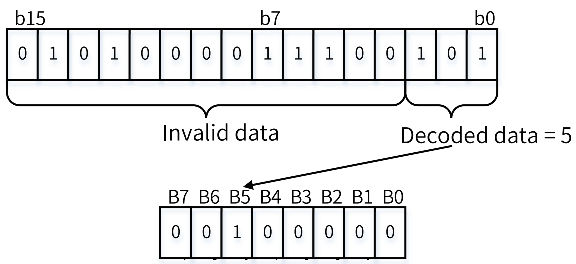
Function and Instruction Description

The DECO instruction calculates the value of the low-order n bits of S, sets the bit variable or element pointed to by this value among the 2^n elements starting from D to ON, and resets other bits.

- Words are decoded into bits.
- When n is 0, the instruction is not executed. When n is beyond the range of 0 to 8, an operation error occurs. When n is 8, D includes a total of 256 bit variables or elements.
- If the instruction flow is OFF, the instruction is not executed.
- The instruction of the pulse execution type is generally used.

Instruction Example

When the M345 flow is ON, the DECO instruction is executed. The low-order 3 bits of D400 is calculated to obtain 5. Then B5 among the 2^3 (8) bit elements starting from B0 is set to ON.



3.6.1.23 ENCO

The ENCO instruction encodes data and stores the result.

ENCO – Data encoding

16-bit Instruction	ENCO: Continuous execution/ENCOP: Pulse execution			
32-bit Instruction	DENCO: Continuous execution/DENCOP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Encoding source	Source data to be encoded	-	BOOL
D	Encoding result	Address of the element that stores the encoding result	-	INT, DINT
n	Bit length of the encoding result	Data bit length of the encoding result	0 to 8	INT, DINT

Table 3–105 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	-	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

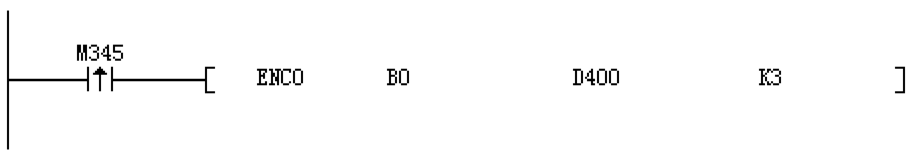
Function and Instruction Description

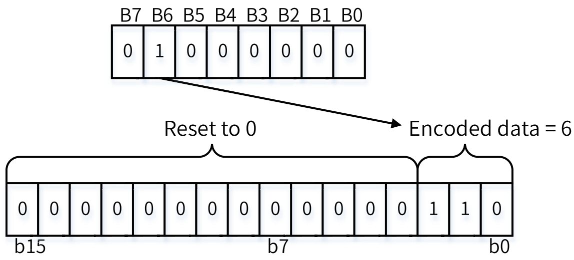
The ENCO instruction obtains the position of the bit variable or element that is set to ON among the 2^n S bit variables or elements, converts it into a value and assigns the value to the low-order n bits of D, and resets other bits of D.

- Bits are encoded into words.
- This instruction can be used to determine whether there is a bit element set to ON among multiple consecutive bit elements.
- When n is 0, the instruction is not executed. When n is beyond the range of 0 to 8, an operation error occurs. When n is 8, S includes a total of 256 bit variables or elements.
- If multiple bits in the source address are 1, only the first ON bit on the high-order side is calculated. If all bits in the source address S are 0, an operation error occurs.
- If the instruction flow is OFF, the instruction is not executed.
- The instruction of the pulse execution type is generally used.

Instruction Example

When the M345 flow is ON, the ENCO instruction is executed. The ON bit B6 is obtained among the 2^3 (8) bit elements starting from B0. Then the value 6 is assigned to the low-order 3 bits of D400, and other bits are cleared.





3.6.2 Data Transfer And Comparison Instructions

3.6.2.1 Instruction List

The following table lists the data transfer and comparison instructions.

Instruction Category	Instruction	Function
Data transfer and comparison instruction	MOV	Move
	EMOV	Binary floating-point move
	SMOV	Shift move
	BMOV	Batch move
	FMOV	Multi-point move
	CML	Complement
	CMP	Comparison
	ECMP	Floating-point comparison
	ZCP	Zone comparison
	EZCP	Floating-point zone comparison

3.6.2.2 MOV

The MOV instruction copies data in the source address S to the destination address.
MOV – Move

16-bit Instruction	MOV: Continuous execution/MOVP: Pulse execution			
32-bit Instruction	DMOV: Continuous execution/DMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Data to be moved, or address of the word element that stores the data	-	INT/DINT
D	Destination to which data is copied	Address of the word element to which data is copied	-	INT/DINT

Table 3–106 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	

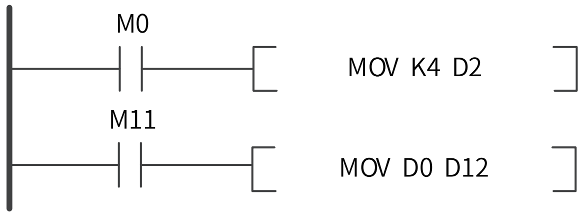
Function and Instruction Description

The MOV instruction requires contact driving and has two operands. It copies the value in S to D.

When the 32-bit instruction (DMOV) is executed, the operation involves the variable units (S+1, S) and (D+1, D).

For example, when the statement [DMOV D1 D5] is executed, data in D1 is moved to D5, and data in D2 is moved to D6.

Instruction Example



When X0 is ON, K4 is assigned to D2. When X0 switches from ON to OFF, K4 in D2 remains unchanged unless the value of D2 is changed again by the user program.

When the PLC is powered on again, the value of D2 becomes 0. The value of a register that is retentive upon power failure remains unchanged when the PLC is powered on or starts running after stop.

3.6.2.3 EMOV

The EMOV instruction transfers binary floating-point data. This instruction requires contact driving. When it is executed, the binary floating-point data in S is copied to D.

EMOV – Floating-point move

16-bit Instruction	-			
32-bit Instruction	DEMOV: Continuous execution/DEMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Source from which the binary floating-point data is transferred	-	REAL
D	Transfer destination	Unit that stores the transferred binary floating-point data	-	REAL

Table 3–107 List of elements

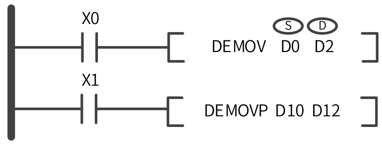
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The EMOV instruction transfers binary floating-point data. This instruction requires contact driving. When it is executed, the binary floating-point data in S is copied to D. Where,

- S is the source from which the binary floating-point data is transferred.
- D is the unit that stores the transferred binary floating-point data.

Instruction Example



Assume that the binary floating-point number in (D1, D0) is 12.3456. When X0 is ON, the binary floating-point number in (D3, D2) becomes 12.3456. When X0 switches from ON to OFF, 12.3456 in (D3, D2) remains unchanged, unless the value of (D3, D2) is changed again by the user program, or the PLC is powered on again or starts running after stop. The value of a register that is retentive upon power failure remains unchanged when the PLC is powered on or starts running after stop.

3.6.2.4 BMOV

When the driving conditions are met, the BMOV instruction transfers the data of n registers in addresses starting from S to the n registers in addresses starting from D.

BMOV – Batch move

16-Bit Instruction	BMOV: Continuous execution/BMOVP: Pulse execution			
32-bit Instruction	DBMOV: Continuous execution/DBMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source start address	Start address of word elements that store the data to be transferred in batches	-	INT/DINT, array*n
D	Transfer destination start address	Start address of word elements that store the transferred data	-	INT/DINT, array*n
n	Data length	Number of word elements of which data will be transferred in batches	1 to 512	INT/DINT

Table 3–108 List of elements

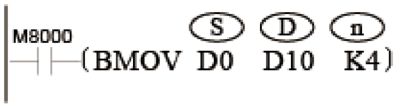
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BMOV instruction requires contact driving and has three operands. It copies the values of n variables in addresses starting from S to n units in addresses starting from D.

n ranges from 1 to 512.

Instruction Example



The operations are as follows:

D0→D10

D1→D11

D2→D12

D3→D13

3.6.2.5 SMOV

The SMOV instruction transfers m2 bits starting from the m1th bit in S to m2 bits starting from the nth bit in D.

SMOV – Shift move

16-bit Instruction	SMOV: Continuous execution/SMOVP: Pulse execution			
32-bit Instruction	DSMOV: Continuous execution/DSMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Address of the word element that stores the data to be transferred	-	INT/DINT
n1	Start bit to be transferred	Start position of bits to be transferred in S	1 to 4/1 to 8	INT/DINT
n2	Number of bits to be transferred	Number of bits to be transferred in S	1 to m1	INT/DINT
D	Destination device	Address of the word element that stores the transferred data	-	INT/DINT
n	Start bit at the destination	Start position of bits transferred to D	m2 to 4/m2 to 8	INT/DINT

Table 3–109 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

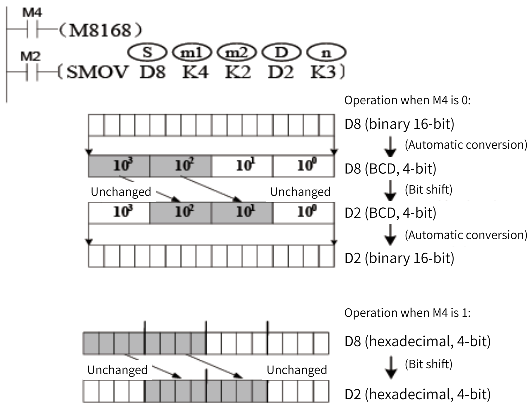
Function and Instruction Description

The SMOV instruction requires contact driving and has a maximum of five operands, which are described as follows:

- S is the data source variable. When M8168 is OFF, the BCD mode (decimal bits) is used. The S operand ranges from 0000 to 9999/00000000 to 99999999 and cannot be negative. When M8168 is ON, the BIN mode is used. The S operand can be a negative number.
- m1 is the start position of bits to be transferred. It ranges from 1 to 4/1 to 8.
- m2 is the number of bits to be transferred. It ranges from 1 to m1.
- D is the destination variable to which data is transferred.
- n is the start bit in the destination variable that stores transferred data. It ranges from m2 to 4/m2 to 8.

The data bit transfer process is related to the state of the special flag M8168. When M8168 is OFF, the BCD mode (decimal bits) is used When M8168 is ON, the BIN mode is used, whereby every four bits (hexadecimal) are transferred at a time as a whole unit.

Instruction Example



Assume that D8 is K1234 and D2 is K5678. When M8168 is OFF (BCD mode), the value in D2 changes to K5128 if M2 is set to ON.

When M8168 is ON (BIN mode), D8 is H04D2 (K1234), and D2 is H162E (K5678), then the value in D2 changes to H104E (K4174) if M2 is set to ON.

3.6.2.6 FMOV

When the driving conditions are met, the FMOV instruction transfers the data in S to n registers starting from the address specified in D.

FMOV – Multi-point move

16-bit Instruction	FMOV: Continuous execution/FMOVP: Pulse execution			
32-bit Instruction	DFMOV: Continuous execution/DFMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Data to be transferred, or address of the word element that stores the data	-	INT/DINT
D	Start address of the transfer destination	Start address of word elements that store the transferred data	-	INT/DINT, array*n
n	Target number	Number of points of the word element to which the data is transferred	1 to 512	INT/DINT

Table 3–110 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The FMOV instruction requires contact driving and has three operands. It copies the data in S to n units starting from the address specified in D.

n ranges from 1 to 512.

FMOV is a 16-bit multi-point transfer instruction, whereas DFMOV is a 32-bit multi-point transfer instruction.

Instruction Example

The operations are as follows when M8 is set to ON:

K100→D100

K100→D101

K100→D102

K100→D103

3.6.2.7 CML

The CML instruction inverts the data in S by bit and transfers the inverted data to D.
CML: Complement

16-bit Instruction	CML: Continuous execution/CMLP: Pulse execution			
32-bit Instruction	DCML: Continuous execution/DCMLP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data source	Data to be inverted, or address of the word element that stores the data	-	INT/DINT
D	Transfer destination	Address of the word element that stores the transferred inverted data	-	INT/DINT

Table 3–111 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The CML instruction requires contact driving and has two operands. It inverts the BIN value in S by bit and copies the inverted data to D.

When the number of bits in D is less than 16, the inverted data is aligned by low-order bits and then transferred to D.

When the 32-bit instruction (DCML) is executed, the operation involves the variable units (S+1, S) and (D+1, D).

For example, when the statement [DCML D1 D5] is executed, the operation result is as follows: /D1→D5; /D2→D6.

Instruction Example

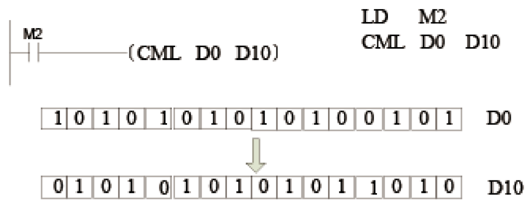


Figure 3-6 Ladder chart and instruction list

3.6.2.8 CMP

CMP – Comparison

16-bit Instruction	CMP: Continuous execution/CMPP: Pulse execution			
32-bit Instruction	DCMP: Continuous execution/DCMPP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Data of comparand 1, or address of the word element that stores the data	-	INT/DINT
S2	Comparand 2	Data of comparand 2, or address of the word element that stores the data	-	INT/DINT
D	Comparison result	Start address of three consecutive bits that store the comparison result (ON or OFF)	-	BOOL, array*3

Table 3–112 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	√[1]	√	√	-	-	√	-	-	-

Note

[1] The X element is not supported.

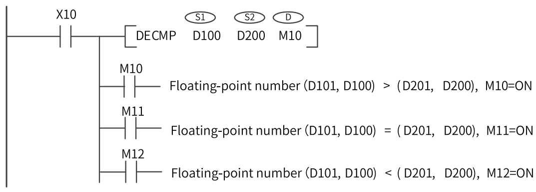
Function and Instruction Description

The CMP instruction compares the values of two operands and outputs the comparison result to the specified bit variable. The operands are handled as signed numbers in algebraic comparison.

When the driving conditions are met, the CMP instruction compares the values in S1 and S2 and then sets a bit element among D, D+1, and D+2 to ON based on the comparison result (S1 > S2, S1 = S2, or S1 < S2).

D is a bit variable that occupies three consecutive addresses.

Instruction Example



When X0 is ON, one among M0 to M2 is set to ON.

When X0 switches from ON to OFF and the CMP instruction is not executed, M0 to M2 remain in the state just before X0 switches from ON to OFF.

To clear the comparison result indicated by M0 to M2, use the RST or ZRST instruction.

To obtain the results of ⩾, ⩽, or ≠, connect M0 to M2 in series or in parallel.

3.6.2.9 ECMP

ECMP – Floating-point comparison

16-bit Instruction	-			
32-bit Instruction	DECMP: Continuous execution/DECMPP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Comparand 1	Binary floating-point number 1 to be compared	-	REAL

S2	Comparand 2	Binary floating-point number 2 to be compared	-	REAL
D	Comparison result	Comparison result storage unit, which occupies three (bit) variables	-	BOOL, array*3

Table 3–113 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
D	√[1]	√	√	-	-	√	-	-	-

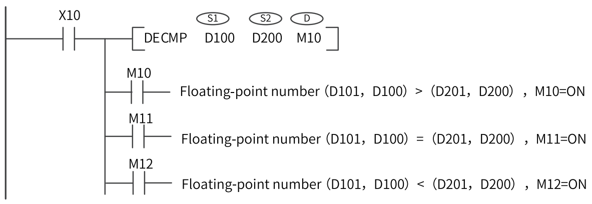
Note

[1] The X element is not supported.

Function and Instruction Description

The ECMP instruction compares two floating-point variables and outputs the comparison result to three variables.

Instruction Example



When X10 is ON, one among M10 to M12 is set to ON.

When X10 switches from ON to OFF and the DECMP instruction is not executed, M10 to M12 remain in the state just before X10 switches from ON to OFF. To clear the comparison result indicated by M10 to M12, use the RST or ZRST instruction.

To obtain the results of ⩾, ⩽, or ≠, connect M10 to M12 in series or in parallel.

3.6.2.10 ZCP

When the driving conditions are met, the ZCP instruction compares S with S1 and S2 and sets a bit element among D, D+1, and D+2 to ON based on the comparison result (S < S1, S1 ⩽ S ⩽ S2, or S > S2). ZCP – Zone comparison

16-bit Instruction	ZCP: Continuous execution/ZCPP: Pulse execution			
32-bit Instruction	DZCP: Continuous execution/DZCPP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Lower limit for comparison	Data, or address of the word element that stores the data	-	INT/DINT

S2	Upper limit for comparison	Data, or address of the word element that stores the data	-	INT/DINT
S	Comparison variable	Data, or address of the word element that stores the data	-	INT/DINT
D	Comparison result	Start address of three consecutive bits that store the comparison result (ON or OFF)	-	BOOL, array*3

Table 3–114 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S	-	-	-	√	√	√	√	-	-
D	√[1]	√	√	-	-	√	-	-	

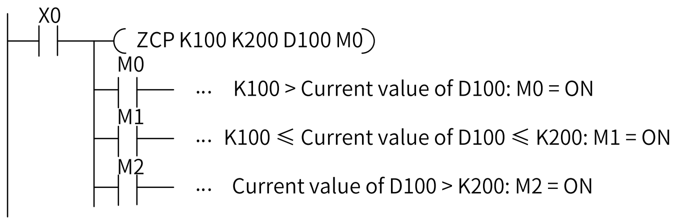
Note

[1] The X element is not supported.

Function and Instruction Description

The ZCP instruction requires contact driving and has four operands. When the control flow is active, the ZCP instruction algebraically compares the operands as signed numbers and stores the comparison result indicated by the position of S relative to S1 and S2 in three consecutive bit variables starting from D.

Instruction Example



When X0 is ON, one among M3 to M5 is set to ON.

When X0 switches from ON to OFF and the ZCP instruction is not executed, M3 to M5 remain in the state just before X0 switches from ON to OFF.

To clear the comparison result indicated by M3 to M5, use the RST or ZRST instruction.

3.6.2.11 EZCP

The EZCP instruction compares a binary floating-point variable with the upper and lower limits of a floating-point variable zone and outputs the comparison result to three variables starting from D.
EZCP – Floating-point zone comparison

16-bit Instruction	-
32-bit Instruction	DEZCP: Continuous execution/DEZCPP: Pulse execution

Operand	Name	Description	Range	Data Type
S1	Lower limit for comparison	Lower limit of a binary floating-point variable range	-	REAL
S2	Upper limit for comparison	Upper limit of a binary floating-point variable range	-	REAL
S	Comparand	Binary floating-point variable to be compared	-	REAL
D	Comparison result	Comparison result storage unit, which occupies three (bit) variables	-	BOOL, array*3

Table 3–115 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	√	-
S2	-	-	-	√	√	√	-	√	-
S	-	-	-	√	√	√	-	√	-
D	√[1]	√	√	-	-	√	-	-	

Note

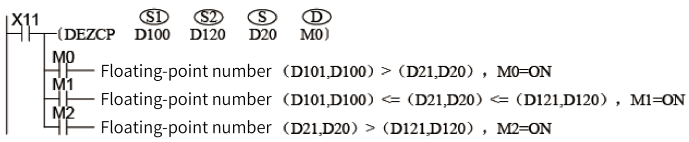
[1] The X element is not supported.

Function and Instruction Description

The EZCP instruction compares a binary floating-point variable with the upper and lower limits of a binary floating-point variable zone and outputs the comparison result to three variables starting from D. Where,

- S1 is the lower limit of the binary floating-point variable zone.
- S2 is the upper limit of the binary floating-point variable zone.
- S is the binary floating-point variable to be compared.
- D is the comparison result storage unit, which occupies three (bit) variables.

Instruction Example



When X11 is ON, one among M0 to M2 is set to ON.

When X11 switches from ON to OFF and the DEZCP instruction is not executed, M0 to M2 remain in the state just before X11 switches from ON to OFF.

To clear the comparison result indicated by M0 to M2, use the RST or ZRST instruction.

3.6.3 Table Operation Instructions

3.6.3.1 Instruction List

The following table lists the table operation instructions.

Instruction Category	Instruction	Function
Table operation instruction	SORTR	Data sorting by row
	SORTC	Data sorting by column
	SER	Data search
	FDEL	Deletion of data from data table
	FINS	Insertion of data to data table
	POP	Last-in data read
	RAMP	Ramp instruction

3.6.3.2 SORTR

When the driving conditions are met, the SORTR instruction sorts a data table (starting from address S) with m1 rows and m2 columns in ascending or descending order according to the data of the nth row, and stores the sorting result in a data table starting from address D1.

SORTR – Data sorting by row

16-bit Instruction	SORTR: Continuous execution			
32-bit Instruction	DSORTR: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Data to be sorted	Start address of the array to be sorted	-	INT/DINT, array*m1*m2
m1	Row count	Total number of rows	1 to 32	INT/DINT
m2	Column count	Total number of columns	1 to 32	INT/DINT
D1	Sorting result	Data sorting result	-	INT/DINT, array*m1*m2
n	Reference row for sorting	Number of the row which serves as the reference for sorting	1 to m1	INT/DINT
D2	Sorting process data	Process data during sorting	-	INT/DINT

Table 3–116 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
m1	-	-	-	√	√	√	√	-	-
m2	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-
D2	-	-	-	√	√	√	-	-	-

Function and Instruction Description

This instruction sorts an array of m1 rows and m2 columns (starting from address S) based on data of the nth row. The sorting result will then be stored in a variable area starting from D1. Where,

- S is the start unit for the first variable in the first row (or record).
- m1 is the number of rows (or records) in the array.
- m2 is the number of columns in the array, or the number of entries in each record.
- D1 is the start unit for storing the sorted data. The number of subsequent variable units occupied is the same as the number of array variables before the sorting.
- n is the number of the array row that serves as the reference for sorting. n ranges from 1 to m1.
- D2 stores the sorting process data, ranging from 0 to (m2 – 1).

The following illustrates the sorting process of a 3 x 3 data table:

Before sorting:

Row Number	Column Number		
	1	2	3
1	S	S+3	S+6
	1	2	8
2	S+1	S+4	S+7
	6	7	2
3	S+2	S+5	S+8
	3	4	3

After sorting (sorted in ascending order based on the second row)

Row Number	Column Number		
	1	2	3
1	D	D+3	D+6
	8	1	2
2	D+1	D+4	D+7
	2	6	7
3	D+2	D+5	D+8
	3	3	4

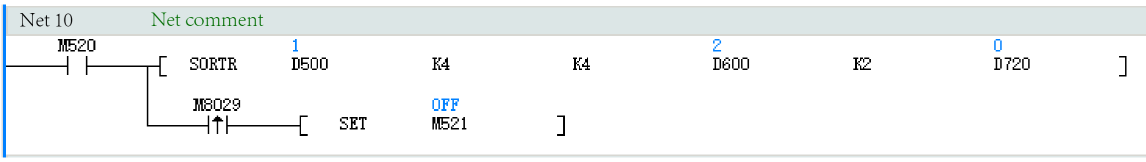
The sorting order is determined by the ON/OFF state of M8165. ON indicates descending and OFF indicates ascending.

Data sorting starts when the instruction flow becomes active. After m1 scan cycles, the sorting is completed, and the instruction completion flag, M8029, is set to ON.

Precautions

- Do not modify the operands during the execution of the SORTR instruction.
- Switch the flow from OFF to ON before you execute the instruction for the second time.
- Keep the operands and data unchanged during execution.
- Data in S and D1 can overlap completely or be staggered, but they cannot overlap partially. Otherwise, error 6705 will occur.
- The 32-bit instruction is used in the same way as the 16-bit instruction. The operands occupy two 16-bit elements.

Instruction Example



When M520 is set to ON, the flow of the SORTR instruction becomes active. The instruction sorts a 4x4 table starting from D500 in ascending order based on the values in the second row. The sorting result is stored in a table starting from D600, and the process data is stored in D720.

Once the sorting is completed, the flag M8029 will be set to ON. However, if there are multiple sorting instructions, the value of M8029 will be overwritten by the subsequent sorting instructions.

Before sorting:

	Element Name	Data Type	Display Format	Current Value
1	... D500	INT	Dec	1
2	... D501	INT	Dec	2
3	... D502	INT	Dec	3
4	... D503	INT	Dec	2
5	... D504	INT	Dec	6
6	... D505	INT	Dec	4
7	... D506	INT	Dec	8
8	... D507	INT	Dec	7
9	... D508	INT	Dec	3
10	... D509	INT	Dec	1
11	... D510	INT	Dec	2
12	... D511	INT	Dec	3
13	... D512	INT	Dec	2
14	... D513	INT	Dec	6
15	... D514	INT	Dec	4
16	... D515	INT	Dec	8

After sorting:

18	... D600	INT	Dec	3
19	... D601	INT	Dec	1
20	... D602	INT	Dec	2
21	... D603	INT	Dec	3
22	... D604	INT	Dec	1
23	... D605	INT	Dec	2
24	... D606	INT	Dec	3
25	... D607	INT	Dec	2
26	... D608	INT	Dec	6
27	... D609	INT	Dec	4
28	... D610	INT	Dec	8
29	... D611	INT	Dec	7
30	... D612	INT	Dec	2
31	... D613	INT	Dec	6
32	... D614	INT	Dec	4
33	... D615	INT	Dec	8

3.6.3.3 SORTC

When the driving conditions are met, the SORTC instruction sorts a data table (starting from address S) with m1 rows and m2 columns in ascending or descending order according to the data of the nth column, and stores the sorting result in a data table starting from address D.

SORTC – Data sorting by column

16-bit Instruction	SORTC: Continuous execution			
32-bit Instruction	DSORTC: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Data to be sorted	Start address of the array to be sorted	-	INT/DINT, array*m1*m2
m1	Row count	Total number of rows	1 to 32	INT/DINT
m2	Column count	Total number of columns	1 to 32	INT/DINT
D1	Sorting result	Data sorting result	-	INT/DINT, array*m1*m2
n	Reference column for sorting	Number of the column which serves as the reference for sorting	1 to m2	INT/DINT
D2	Sorting process data	Process data during sorting	-	INT/DINT

Table 3–117 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
m1	-	-	-	√	√	√	√	-	-
m2	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-
D2	-	-	-	√	√	√	-	-	-

Function and Instruction Description

This instruction sorts an array of m1 rows and m2 columns (starting from address S) based on data of the nth column. The sorting result will then be stored in a variable area starting from D1. Where,

- S is the start unit for the first variable in the first row (or record).
- m1 is the number of rows (or records) in the array.
- m2 is the number of columns in the array, or the number of entries in each record.
- D1 is the start unit for storing the sorted data. The number of subsequent variable units occupied is the same as the number of array variables before the sorting.
- n is the number of the array column that serves as the reference for sorting. n ranges from 1 to m2.
- D2 stores the sorting process data, ranging from 0 to (m1 – 1).

The sorting order is determined by the ON/OFF state of M8165. ON indicates descending and OFF indicates ascending.

Data sorting starts when the instruction flow becomes active. After m1 scan cycles, the sorting is completed, and the instruction completion flag, M8029, is set to ON.

The following illustrates the sorting process of a 3 x 3 data table:

Before sorting:

Row Number	Column Number		
	1	2	3
1	S	S+3	S+6
	1	2	8
2	S+1	S+4	S+7
	2	6	7
3	S+2	S+5	S+8
	3	4	3

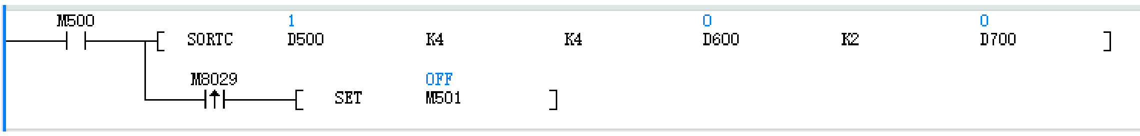
After sorting (sorted in ascending order based on the second column)

Row Number	Column Number		
	1	2	3
1	D	D+3	D+6
	1	2	8
2	D+1	D+4	D+7
	3	4	3
3	D+2	D+5	D+8
	2	6	7

Precautions

- Do not modify the operands during the execution of the SORTR instruction.
- Switch the flow from OFF to ON before you execute the instruction for the second time.
- Keep the operands and data unchanged during execution.
- Data in S and D1 can overlap completely or be staggered, but they cannot overlap partially. Otherwise, error 6705 will occur.
- The 32-bit instruction is used in the same way as the 16-bit instruction. The operands occupy two 16-bit elements.

Instruction Example



When M500 is set to ON, the flow of the SORTC instruction becomes active. The instruction sorts a 4x4 table starting from D500 in ascending order based on the values in the second column. The sorting result is stored in a table starting from D600, and the process data is stored in D700.

Once the sorting is completed, the flag M8029 will be set to ON. However, if there are multiple sorting instructions, the value of M8029 will be overwritten by the subsequent sorting instructions.

Before sorting:

	Element Name		Data Type	Display Format	Current Value
1	...	D500	INT	Dec	1
2	...	D501	INT	Dec	2
3	...	D502	INT	Dec	3
4	...	D503	INT	Dec	2
5	...	D504	INT	Dec	6
6	...	D505	INT	Dec	4
7	...	D506	INT	Dec	8
8	...	D507	INT	Dec	7
9	...	D508	INT	Dec	3
10	...	D509	INT	Dec	1
11	...	D510	INT	Dec	2
12	...	D511	INT	Dec	3
13	...	D512	INT	Dec	2
14	...	D513	INT	Dec	6
15	...	D514	INT	Dec	4
16	...	D515	INT	Dec	8

After sorting:

18	...	D600	INT	Dec	2
19	...	D601	INT	Dec	1
20	...	D602	INT	Dec	2
21	...	D603	INT	Dec	3
22	...	D604	INT	Dec	4
23	...	D605	INT	Dec	6
24	...	D606	INT	Dec	7
25	...	D607	INT	Dec	8
26	...	D608	INT	Dec	1
27	...	D609	INT	Dec	3
28	...	D610	INT	Dec	3
29	...	D611	INT	Dec	2
30	...	D612	INT	Dec	6
31	...	D613	INT	Dec	2
32	...	D614	INT	Dec	8
33	...	D615	INT	Dec	4

3.6.3.4 SER

When the driving conditions are met, the SER instruction searches n data entries starting from source address S1 to find the address of the data compliant with the condition specified by S2 and stores the result in five consecutive registers starting from D.

SER – Data search

16-bit Instruction	SER: Continuous execution/SERP: Pulse execution			
32-bit Instruction	DSER: Continuous execution/DSERP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Search start address	Start address of the data (n consecutive registers) to be searched	-	INT/DINT, array*n
S2	Data for comparison	Data for comparison, or address of the word element that stores the data	-	INT/DINT
D	Search result storage start address	Start address of word elements that store the search result	-	INT/DINT, array*5
n	Number of data entries to be searched	Number of data entries to be searched	1 to 256	INT/DINT

Table 3–118 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	√		
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

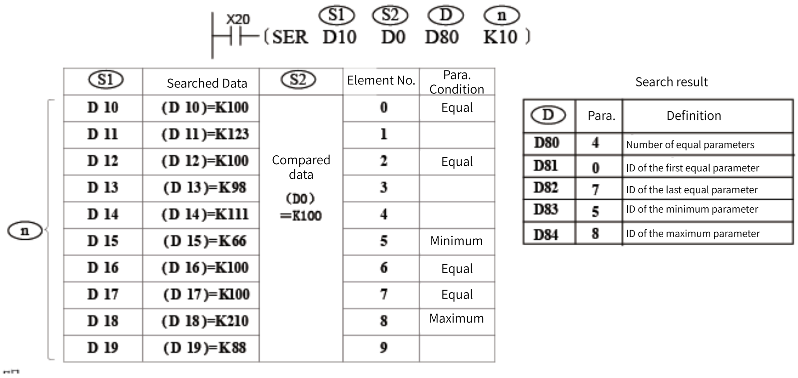
The SER instruction searches a defined data stack for the units with the same data as the data for comparison as well as the maximum and minimum values. Where,

- S1 is the start address of the searched data stack.
- S2 is the data to be searched for.
- D is the start address of elements for storing the search result.
- n is the length of the searched data area. It ranges from 1 to 256.

In 32-bit operation, S1, S2, and D point to 32-bit variables, and n is calculated based on the 32-bit variable width.

When the driving conditions are met, the SER instruction searches k data entries starting from source address S1 to find the address of the data compliant with the condition specified by S2 and stores the result in five consecutive registers starting from D.

Instruction Example



The example is explained as follows:

Comparison is performed only when X20 is ON. Signed numbers are compared algebraically, for example, –8 < +2.

When there are multiple minimum or maximum values, the element with the largest serial number is displayed.

The search result is stored in five consecutive units starting from D. If no data that meets the requirements is found, the values in D80 to D82 in the preceding example are all 0s.

3.6.3.5 FDEL

The FDEL instruction deletes any data from a table.

FDEL – Deletion of data from a table

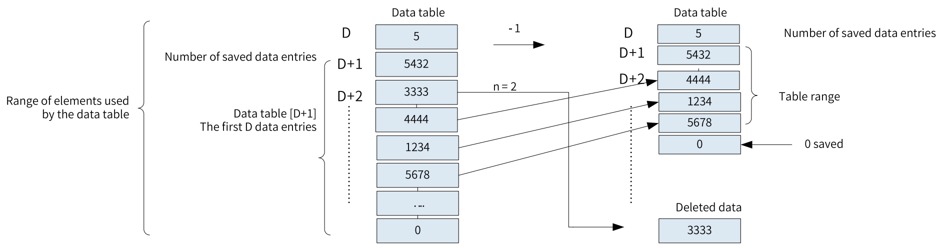
16-bit Instruction	FDEL: Continuous execution/FDELP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data to be deleted	Number of the element that stores the data to be deleted	-	INT
D	Data table information	Start number of elements that store the data table D: Number of stored data entries D+1: Start position of a data table	-	INT, array*(D+1)
n	Position of deletion	Position in a table at which data is deleted	1 to 256	INT

Table 3–119 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The FDEL instruction deletes the nth data entry from a table (starting from [D+1]) and stores the deleted data in [S]. The (n+1)th data entry and subsequent ones in the data table are shifted forward one by one, and the number of stored data entries (D) decreases by 1.



An error is returned in the following conditions:

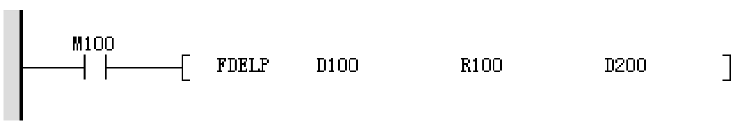
1. The number of stored data entries is out of range.

2. The table position n of the data to be deleted exceeds the number of stored data entries (D).

3. n is less than or equal to 0.

4. The number of stored data entries is less than or equal to 0.

Instruction Example



Before the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	0
R100	16-bit INT	Decimal	5
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	3333
R104	16-bit INT	Decimal	4444
R105	16-bit INT	Decimal	5555
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

After the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3333
R100	16-bit INT	Decimal	4
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	4444
R104	16-bit INT	Decimal	5555
R105	16-bit INT	Decimal	0
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

3.6.3.6 FINS

The FINS instruction inserts data at any position in a table.

FINS – Insertion of data to a table

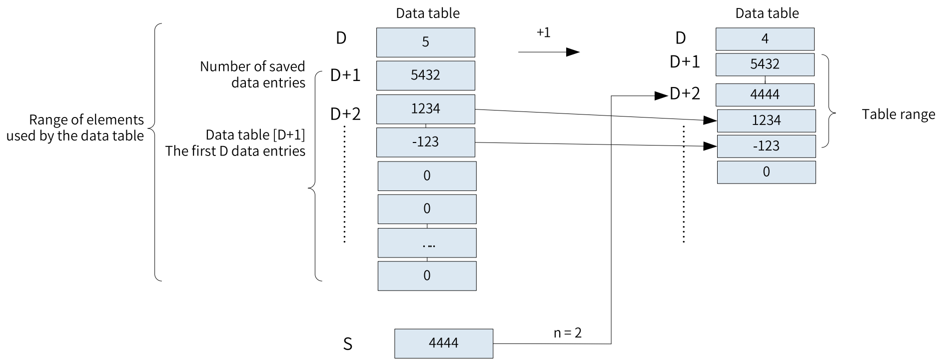
16-bit Instruction	FINS: Continuous execution/FINSP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data to be inserted	Number of the element that stores the data to be inserted	-	INT
D	Data table information	Start number of elements that store the data table D: Number of stored data entries D+1: Start position of a data table	-	INT, array*(D+2)
n	Position of insertion	Position in a table at which data is inserted	1 to 256	INT

Table 3–120 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The FINS instruction inserts the data stored in [S] at the nth data entry position in a data table starting from [D+1]. The original nth data entry and subsequent ones in the data table are shifted backward one by one, and the number of stored data entries (D) increases by 1.



An error is returned in the following conditions:

1. The number of stored data entries is out of range.

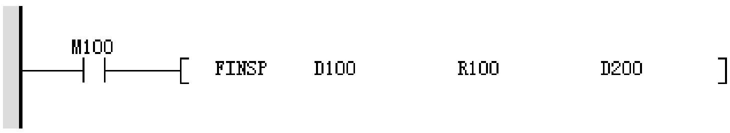
2. The data table range after insertion exceeds the corresponding element range.

3. The position of insertion (n) exceeds the number of stored data entries (D).

4. n is less than or equal to 0.

5. The number of stored data entries is less than or equal to 0.

Instruction Example



Before the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3333
R100	16-bit INT	Decimal	4
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	4444
R104	16-bit INT	Decimal	5555
R105	16-bit INT	Decimal	0
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

After the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3333
R100	16-bit INT	Decimal	5
R101	16-bit INT	Decimal	1111
R102	16-bit INT	Decimal	2222
R103	16-bit INT	Decimal	3333
R104	16-bit INT	Decimal	4444
R105	16-bit INT	Decimal	5555
R106	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	3

3.6.3.7 POP

The POP instruction reads the last data written by a shift write instruction (SFWR) for first in last out (FI-LO) control.

POP – Last-in data read

16-bit Instruction	POP: Continuous execution/POPP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data to be read	Start number of elements that store first-in data (including pointer data) S: Pointer data (number of stored data entries) S+1: Data area	-	INT, array*n
D	Stored result	Number of the element that stores the last-out data	-	INT
n	Data count	Count of stored data (Because pointer data is also included, set n to a value plus 1. Value range: 2 ⩽ n ⩽ 512.)	2 to 512	INT

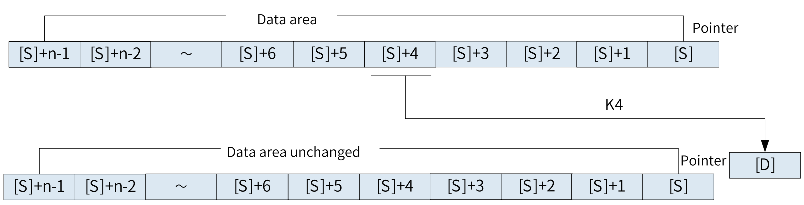
Table 3–121 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

For the word elements [S] to [S+n–1], the POP instruction reads the values in elements starting from S as well as the offset pointer (pointer data) in [S] and stores the result in [D]. The pointer in [S] decreases by 1. n ranges from 2 to 512.

Address	Content
S	Pointer data (number of stored data entries)
[S]+1	Data area (First-in data written by the SFWR instruction)
[S]+2	
[S]+3	
-	
[S]+n–3	
[S]+n-2	
[S]+n-1	



When the pointer in [S] is 0, the zero flag M8020 is set to ON and the POP instruction is not executed.

Check in advance using a comparison instruction whether the current value of [S] satisfies the following condition before executing the POP instruction: 1 ⩽ [S] ⩽ (n – 1).

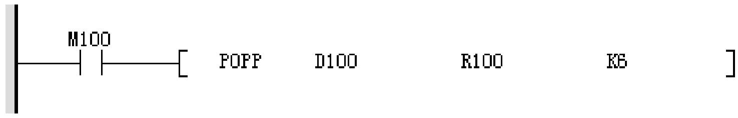
When the current value of the pointer in [S] is 1, 0 is written to [S] and the zero flag M8020 turns ON.

An error is returned in the following conditions:

[S] > n – 1

[S] < 0

Instruction Example



Before the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	4
D101	16-bit INT	Decimal	1111
D102	16-bit INT	Decimal	2222
D103	16-bit INT	Decimal	3333
D104	16-bit INT	Decimal	4444
D105	16-bit INT	Decimal	5555
D106	16-bit INT	Decimal	6666
D107	16-bit INT	Decimal	0
	16-bit INT	Decimal	
R100	16-bit INT	Decimal	0

After the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	3
D101	16-bit INT	Decimal	1111
D102	16-bit INT	Decimal	2222
D103	16-bit INT	Decimal	3333
D104	16-bit INT	Decimal	4444
D105	16-bit INT	Decimal	5555
D106	16-bit INT	Decimal	6666
D107	16-bit INT	Decimal	0
	16-bit INT	Decimal	
R100	16-bit INT	Decimal	4444

3.6.3.8 RAMP

When the driving conditions are met, the RAMP instruction changes the value in D linearly from S1 to S2 after a number (indicated by n) of scan cycles are completed.

RAMP – Ramp instruction

16-bit Instruction	RAMP: Continuous execution			
32-bit Instruction	DRAMP: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Start value	Address of the word element that stores the ramp start value	-	INT/DINT
S2	End value	Address of the word element that stores the ramp end value	-	INT/DINT

D	Current value	Address of the word element that stores the current ramp value	-	INT/DINT, array*2
n	Cycle count	Number of scan cycles required to complete a ramp change	≥ 1	INT/DINT

Table 3–122 List of elements

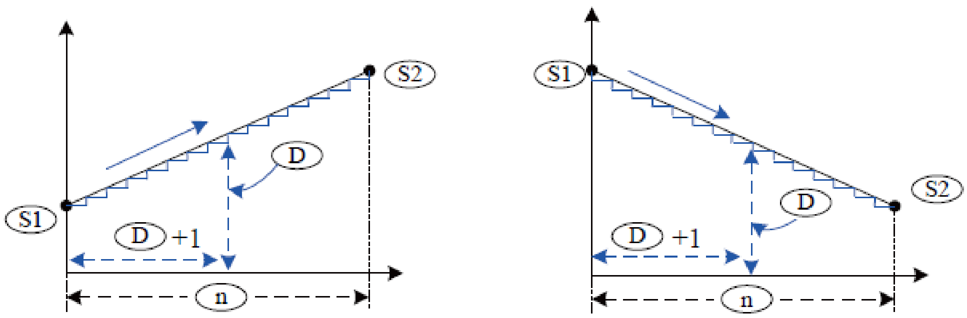
Operand	Bit			Word		Pointer	Constant		Oth ers
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	√	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

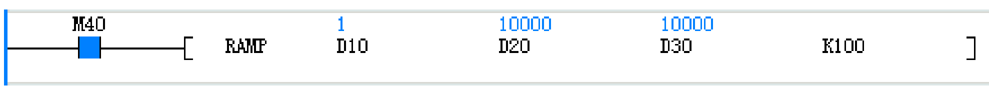
This instruction is used to perform linear interpolation between two given data entries a specified time interval. It outputs intermediate values in a sequential manner based on the scanning execution time, until the end value is reached. After each interpolation operation is completed, the M8029 flag is set to ON. Where,

- S1 stores the start value of a ramp signal.
- S2 stores the end value of a ramp signal.
- D stores the process values of a linear interpolation signal. The interpolation timer is stored in D+1.
- n is the number of program scans required to complete the interpolation. (For a 16-bit instruction, n ranges from 1 to 32,767.) Since interpolation output is performed within the normal main loop, set program execution to fixed scan mode to ensure linear output.

Interpolation is performed using integers, and the fractional parts are discarded. The function of the instruction is shown as follows:



Instruction Example



When M40 is ON, interpolation is performed 100 times directly from 1 to 10000. The output results are stored in D30, and the number of interpolations is displayed in D31.

Additional Information

Different from that in H3U, the RAMP instruction in H5U only supports single interpolation mode but not continuous interpolation.

3.6.4 Data Shift Instructions

3.6.4.1 Instruction List

The following table lists the data shift instructions.

Instruction Category	Instruction	Function
Data shift instruction	ROR	Rotation right
	ROL	Rotation left
	RCR	Rotation right with carry
	RCL	Rotation left with carry
	SFTR	Bit shift right
	SFTL	Bit shift left
	WSFR	Word shift right
	WSFL	Word shift left
	SFWR	Shift write (FIFO)
	SFRD	Shift read (FIFO)
	SFR	Bit shift right with carry
	SFL	Bit shift left with carry

3.6.4.2 ROR

When the driving conditions are met, the ROR instruction shifts and rotates the data in D rightwards by n bits. The low-order bits that are rotated out of D fill the high-order bits of D.
ROR – Rotation right

16-bit Instruction	ROR: Continuous execution/RORP: Pulse execution			
32-bit Instruction	DROR: Continuous execution/DRORP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Source data/Target data	Address of the word element that stores the data	-	INT/DINT
n	Number of bits to be rotated upon each execution	Value range: 1 ≤ n ≤16 (16-bit operation); 1 ≤ n ≤ 32 (32-bit operation)	1 to 16/1 to 32	INT/DINT

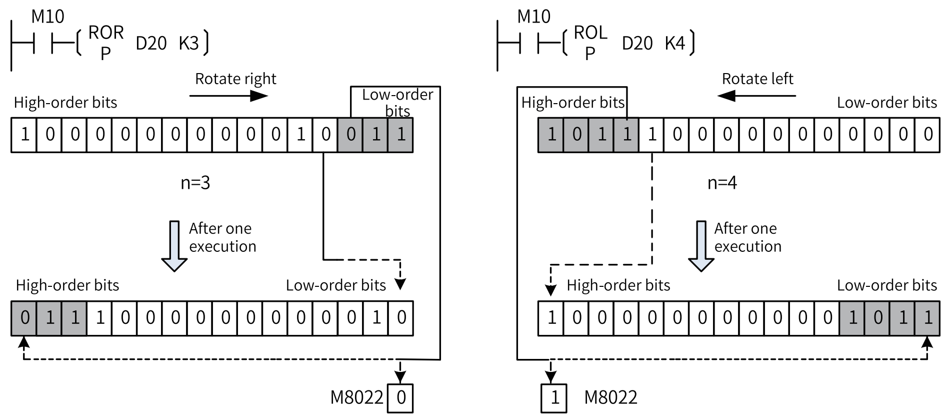
Table 3–123 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ROR instruction shifts and rotates the data in D with the carry flag M8022 rightwards by n bits. The instruction of the pulse execution type is generally used. When the 32-bit instruction is executed, the register variable occupies two consecutive units.

Instruction Example



3.6.4.3 ROL

When the driving conditions are met, the ROL instruction shifts and rotates the data in D leftwards by n bits. The high-order bits that are rotated out of D fill the low-order bits of D.

ROL – Rotation left

16-bit Instruction	ROL: Continuous execution/ROLP: Pulse execution			
32-bit Instruction	DROL: Continuous execution/DROLP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Source data/Target data	Address of the word element that stores the data	-	INT/DINT
n	Number of bits to be rotated upon each execution	Value range: 1 ⩽ n ⩽ 16 (16-bit operation); 1 ⩽ n ⩽ 32 (32-bit operation)	1 to 16/1 to 32	INT/DINT

Table 3–124 List of elements

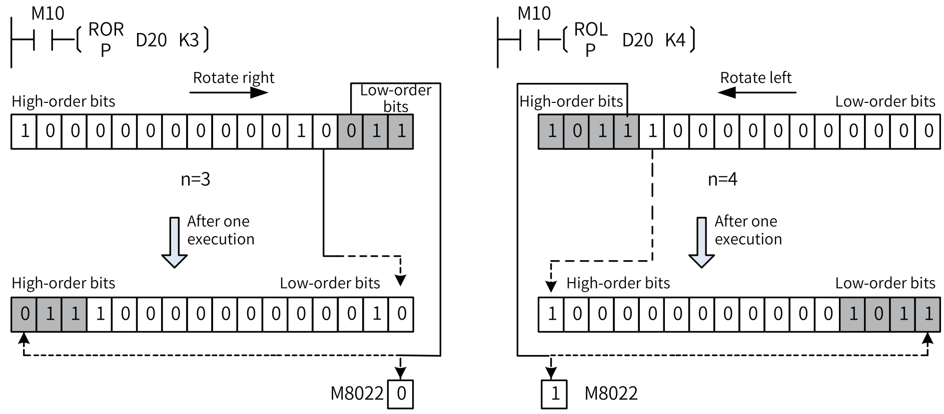
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The ROL instruction shifts and rotates the data in D leftwards by n bits. The instruction of the pulse execution type is generally used. When the 32-bit instruction is executed, the register variable occupies two consecutive units.

The final bit is stored in the carry flag.

Instruction Example



3.6.4.4 RCR

When the driving conditions are met, the RCR instruction shifts and rotates the data in D with the carry flag (M8022) rightwards by n bits. The low-order bits with the carry flag (M8022) that are rotated out of D fill the high-order bits of D.

RCR – Rotation right with carry

16-bit Instruction	RCR: Continuous execution/RCRP: Pulse execution			
32-bit Instruction	DRCR: Continuous execution/DRCRP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Source data/Target data	Address of the word element that stores the data	-	INT/DINT
n	Number of bits to be rotated upon each execution	Value range: 1 ⩽ n ⩽ 16 (16-bit operation); 1 ⩽ n ⩽ 32 (32-bit operation)	1 to 16/1 to 32	INT/DINT

Table 3–125 List of elements

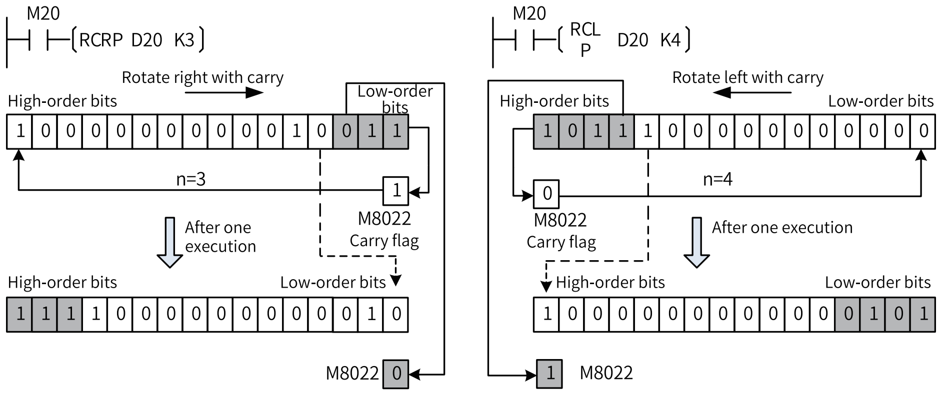
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The RCR instruction shifts and rotates the data in D with the carry flag M8022 rightwards by n bits.

The instruction of the pulse execution type is generally used. When the 32-bit instruction is executed, the register variable occupies two consecutive units.

Instruction Example



3.6.4.5 RCL

When the driving conditions are met, the RCL instruction shifts and rotates the data in D with the carry flag (M8022) leftwards by n bits. The high-order bits with the carry flag (M8022) that are rotated out of D fill the low-order bits of D.

RCL – Rotation left with carry

16-bit Instruction	RCL: Continuous execution/RCLP: Pulse execution			
32-bit Instruction	DRCL: Continuous execution/DRCLP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Source data/Target data	Address of the word element that stores the data	-	INT/DINT
n	Number of bits to be rotated upon each execution	Value range: 1 ⩽ n ⩽ 16 (16-bit operation); 1 ⩽ n ⩽ 32 (32-bit operation)	1 to 16/1 to 32	INT/DINT

Table 3–126 List of elements

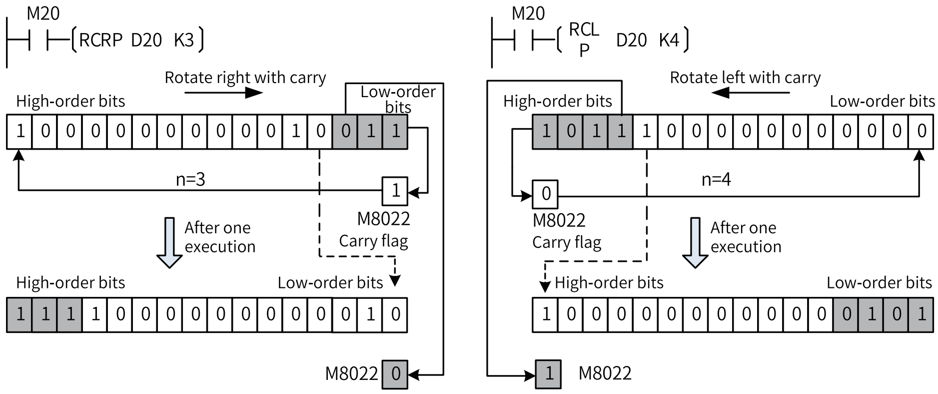
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The RCL instruction shifts and rotates the data in D with the carry flag (M8022) leftwards by n bits. The instruction of the pulse execution type is generally used.

When the 32-bit instruction is executed, the register variable occupies two consecutive units.

Instruction Example



3.6.4.6 SFTR

When the driving conditions are met, the SFTR instruction shifts n1 bit elements starting from D rightwards by n2 bits and transfers n2 bit elements starting from S to fill the high-order bits. The n2 low-order bits that are shifted out are discarded. The values in the bit elements starting from S remain unchanged.

SFTR – Bit shift right

16-bit Instruction	SFTR: Continuous execution/SFTRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element start address	Start address of bit elements to be shifted	-	BOOL, array*n2
D	Incoming bit start address	Start address of incoming bit elements	-	BOOL, array*n1
n1	Incoming bit count	Number of incoming bit elements	1 to 256	INT
n2	Bit element quantity	Number of shifted bit elements	1 to n1	INT

Table 3–127 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
D	√[1]	√	√	-	-	√	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

Note

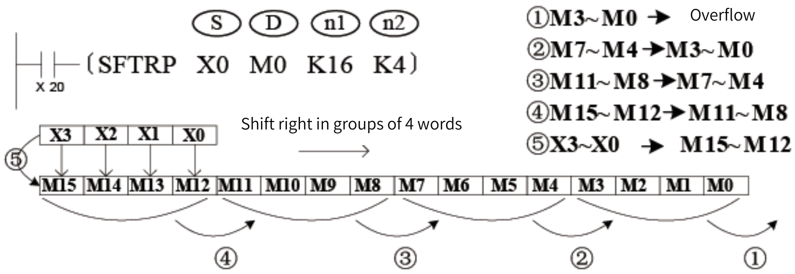
[1] The X element is not supported.

Function and Instruction Description

The SFTR instruction transfers n2 bit variables starting from S to D and shifts the bit variables rightwards by n2 bits.

The instruction of the pulse execution type is generally used.

Instruction Example



3.6.4.7 SFTL

When the driving conditions are met, the SFTL instruction shifts n1 bit elements starting from D leftwards by n2 bits and transfers n2 bit elements starting from S to fill the low-order bits. The n2 high-order bits that are shifted out are discarded. The values in the bit elements starting from S remain unchanged.

SFTL – Bit shift left

16-bit Instruction	SFTL: Continuous execution/SFTLP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Bit element start address	Start address of bit elements to be shifted	-	BOOL, array*n2
D	Incoming bit start address	Start address of incoming bit elements	-	BOOL, array*n1
n1	Incoming bit count	Number of incoming bit elements	1 to 256	INT
n2	Bit element quantity	Number of shifted bit elements	1 to n1	INT

Table 3–128 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	√	√	√	-	-	√	-	-	-
D	✓[1]	√	√	-	-	√	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

Note

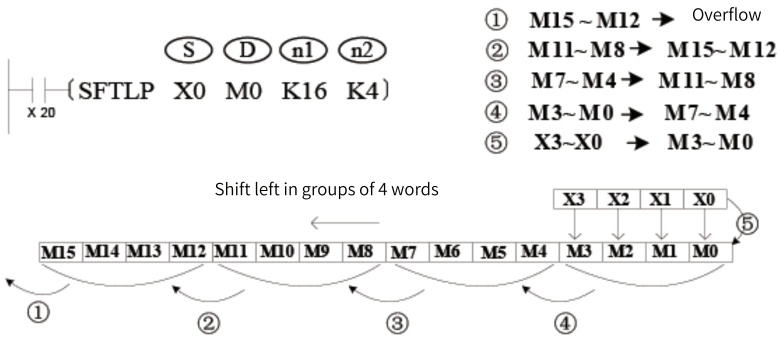
[1] The X element is not supported.

Function and Instruction Description

The SFTL instruction transfers n2 bit variables starting from S to D and shifts the bit variables leftwards by n2 bits.

The instruction of the pulse execution type is generally used.

Instruction Example



3.6.4.8 WSFR

When the driving conditions are met, the WSFR instruction shifts n1 word elements starting from D rightwards by n2 words and transfers n2 word elements starting from S to fill the high-order words. The n2 low-order words that are shifted out are discarded. The values in the word elements starting from S remain unchanged.

WSFR – Word shift right

16-bit Instruction	WSFR: Continuous execution/WSFRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Word element start address	Start address of word elements to be shifted	-	INT, array*n2
D	Incoming word start address	Start address of incoming word elements	-	INT, array*n1
n1	Incoming word count	Number of incoming word elements	1 to 256	INT
n2	Word element count	Number of shifted word elements	1 to n1	INT

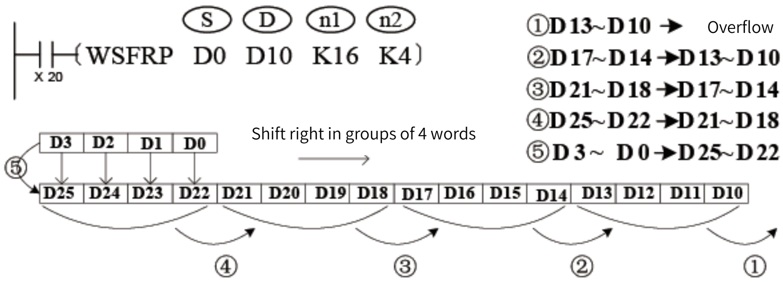
Table 3–129 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WSFR instruction shifts n1 word variables starting from D rightwards by n2 words and then transfers the n2 word variables starting from S to D to fill the high-order words. The instruction of the pulse execution type is generally used.

Instruction Example



3.6.4.9 WSFL

When the driving conditions are met, the WSFL instruction shifts n1 word elements starting from D leftwards by n2 words and transfers n2 word elements starting from S to fill the low-order words. The n2 high-order words that are shifted out are discarded. The values in the word elements starting from S remain unchanged.

WSFL – Word shift left

16-bit Instruction	WSFL: Continuous execution/WSFLP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Word element start address	Start address of word elements to be shifted	-	INT, array*n2
D	Incoming word start address	Start address of incoming word elements	-	INT, array*n1
n1	Incoming word count	Number of incoming word elements	1 to 256	INT
n2	Word element count	Number of shifted word elements	1 to n1	INT

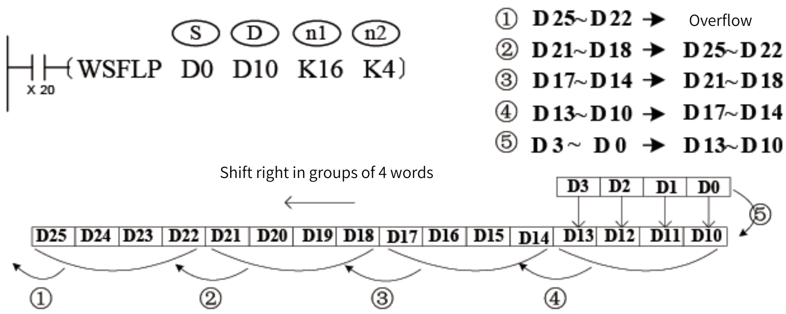
Table 3–130 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n1	-	-	-	√	√	√	√	-	-
n2	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The WSFL instruction shifts n1 word variables starting from D leftwards by n2 words and then transfers the n2 word variables starting from S to D to fill the low-order words. The instruction of the pulse execution type is generally used.

Instruction Example



3.6.4.10 SFWR

When the driving conditions are met, the SFWR instruction writes the current value in S to a data register starting from D+1 (with the length of n). The value of the pointer D increases by 1 each time a data entry is written to the database.

SFWR: Shift write (FIFO)

16-bit Instruction	SFWR: Continuous execution/SFWRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data source	Data to be written, or address of the word element that stores the data	-	INT
D	Data area start address	Start address of word elements that store the data	-	INT, array*n
n	Data area length	Length of the data area, including the pointer	2 to 512	INT

Table 3–131 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	√	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The SFWR instruction writes the value of S to the first in first out (FIFO) queue starting from D (with the length of n). The first device stores a pointer. When the instruction is executed, the pointer value increases by 1 and then the content of the device specified by S is written to the position specified by the pointer in the FIFO queue (D).

The instruction of the pulse execution type is generally used.

Instruction Example

When X0 is 1, the value in D0 is written to D2, and the value of D1 changes to 1. When X0 switches from OFF to ON again, the value in D0 is written to D3, and the value of D1 changes to 2, and so on. If the value in D1 exceeds the value of n minus 1, the instruction is not executed, and the carry flag M8022 is set to 1.

3.6.4.11 SFRD

When the driving conditions are met, the SFRD instruction reads the data in the data register starting from S+1 (with the length of n) to the destination register D.
SFRD: Shift read (FIFO)

16-bit Instruction	SFRD: Continuous execution/SFRDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data area start address	Start address of word elements that store the data	-	INT, array*n
D	Read data	Address for storing the read data	-	INT
n	Data area length	Length of the data area	2 to 512	INT

Table 3–132 List of elements

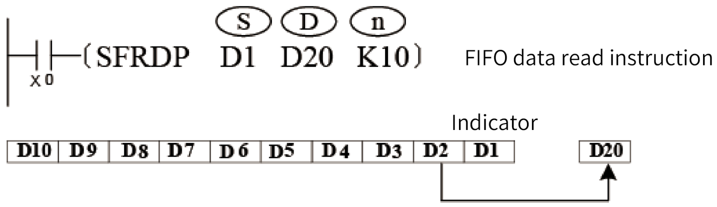
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The SFRD instruction reads the first data entry in the FIFO queue (S) to D and shifts the data within the queue rightwards by one word. The queue pointer decreases by 1. The first device stores a pointer. When the instruction is executed, the pointer value decreases by 1 and then the content of the device specified by S is written to position specified by the pointer in the FIFO queue (D). If the pointer is 0, the instruction is not executed and the zero flag M8020 is set to 1.

The instruction of the pulse execution type is generally used.

Instruction Example



When X0 switches from OFF to ON, the following operations are performed (the content in D10 remains unchanged):

1. The content of D2 is read and transferred to D20.

2. D10 to D3 are shifted rightwards by one register.

3. The value of pointer D1 decreases by 1.

3.6.4.12 SFR

The SFR instruction shifts the data in a word element rightwards by n bits.
SFR – Bit shift right with carry

16-bit Instruction	SFR: Continuous execution/SFRP: Pulse execution			
32-bit Instruction	DSFR: Continuous execution/DSFRP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Word to be shifted	Number of the element that stores the data to be shifted	-	INT/DINT
n	Shift times	Number of shift times 0 ⩽ n ⩽ 15 (16-bit operation); 0 ⩽ n ⩽ 31 (32-bit operation)	0 to 15/31	INT/DINT

Table 3–133 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

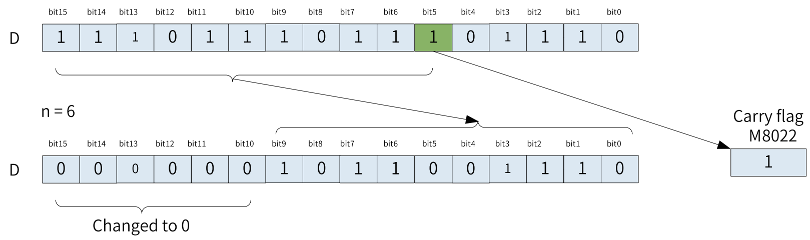
16-bit instruction:

n ranges from 0 to 15. When n is greater than or equal to 16, bits are shifted by the remainder of n%16. For example, when n is 20, bits are shifted rightwards by four bits (20%16 = 4).

32-bit instruction:

n ranges from 0 to 31. When n is greater than or equal to 32, bits are shifted by the remainder of n%32. For example, when n is 40, bits are shifted rightwards by eight bits (40%32 = 8).

The 1/0 state of the (bn – 1)th bit in [D] is written to the carry flag M8022. The n bits starting from the most significant bit in [D] are filled with 0s.



Errors

An error is reported when n is less than 0.

Instruction Example

Before the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	OFF
	16-bit INT	Dec	

After the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	ON
	16-bit INT	Dec	

3.6.4.13 SFL

The SFL instruction shifts the data in a word element leftwards by n bits.
SFL – Bit shift left with carry

16-bit Instruction	SFL: Continuous execution/SFLP: Pulse execution			
32-bit Instruction	DSFL: Continuous execution/DSFLP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Word to be shifted	Number of the element that stores the data to be shifted	-	INT/DINT
n	Shift times	Number of shift times 0 ⩽ n ⩽ 15 (16-bit operation); 0 ⩽ n ⩽ 31 (32-bit operation)	0 to 15/31	INT/DINT

Table 3–134 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

16-bit instruction:

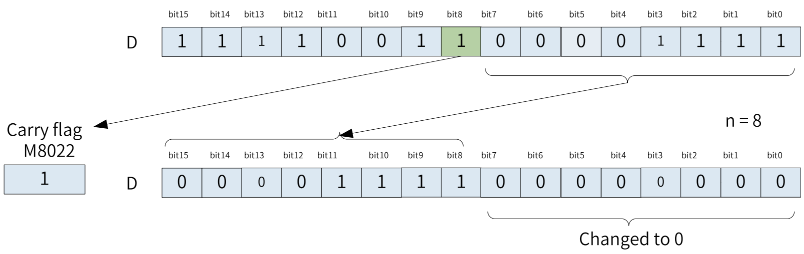
n ranges from 0 to 15. When n is greater than or equal to 16, bits are shifted by the remainder of n%16. For example, when n is 20, bits are shifted leftwards by four bits (20%16 = 4).

32-bit instruction:

n ranges from 0 to 31. When n is greater than or equal to 32, bits are shifted by the remainder of n%32. For example, when n is 40, bits are shifted leftwards by eight bits (40%32 = 8).

The 1/0 state of the bn bit in [D] is written to the carry flag M8022.

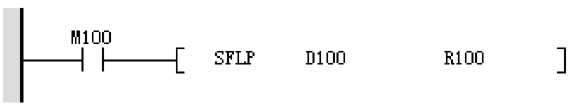
The n bits starting from the least significant bit in [D] are filled with 0s.



Errors

An error is reported when n is less than 0.

Instruction Example



Before the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x5555
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	OFF
	16-bit INT	Dec	

After the instruction is executed

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x5500
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x8
	16-bit INT	Dec	
M8022	BOOL	Bin	ON

3.6.5 Other Data Processing Instructions

3.6.5.1 Instruction List

The following table lists other data processing instructions.

Instruction Category	Instruction	Function
Other data processing instruction	SWAP	Byte swap
	BON	Bit state check
	SUM	Sum of ON bits
	RAND	Random number generation within limits
	XCH	Data exchange
	ABS	Absolute value of integer
	EABS	Absolute value of floating-point number
	EFMOV	Multi-point floating-point move
	CCD	Check code
	CRC	CRC code calculation
	LRC	LRC code calculation

3.6.5.2 SWAP

The SWAP instruction exchanges the upper and lower bytes of the variable in S.

SWAP: Byte swap

16-bit Instruction	SWAP: Continuous execution/SWAPP: Pulse execution			
32-bit Instruction	DSWAP: Continuous execution/DSWAPP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Operand	Unit that stores the data of which the upper and lower bytes will be exchanged	-	INT/DINT

Table 3–135 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The SWAP instruction exchanges the upper and lower bytes of the variable in S.

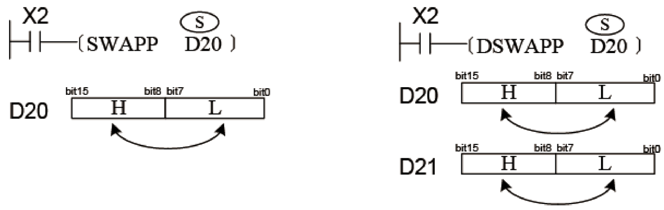
In 16-bit operation, the high-order 8 bits and the low-order 8 bits are swapped.

In 32-bit operation, the high-order 8 bits and the low-order 8 bits of each of the two registers are swapped.

Note

This instruction is generally programmed as the pulse execution type. If it is programmed as the continuous execution type, swap is performed on every program scan.

Instruction Example



In the figure on the left, the values of the high-order 8 bits and low-order 8 bits in D20 are swapped.

In the figure on the right, the values of the high-order 8 bits and low-order 8 bits in D20 are swapped.

The values of the high-order 8 bits and low-order 8 bits in D21 are swapped.

3.6.5.3 BON

When the driving conditions are met, the BON instruction checks the state of the nth bit of the binary data in S and outputs the result to D.

BON – Bit state check

16-bit Instruction	BON: Continuous execution/BONP: Pulse execution			
32-bit Instruction	DBON: Continuous execution/DBONP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Data, or address of the word element that stores the data	-	INT/DINT
D	Controlled bit	Controlled bit element	-	BOOL
n	Designated bit	Designated bit in S; value range: 0 ⩽ n ⩽ 15 (16-bit operation); 0 ⩽ n ⩽ 31 (32-bit operation)	0 to 15/31	INT/DINT

Table 3–136 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	✓[1]	√	√	-	-	√	-	-	-
n	-	-	-	√	√	√	√	-	-

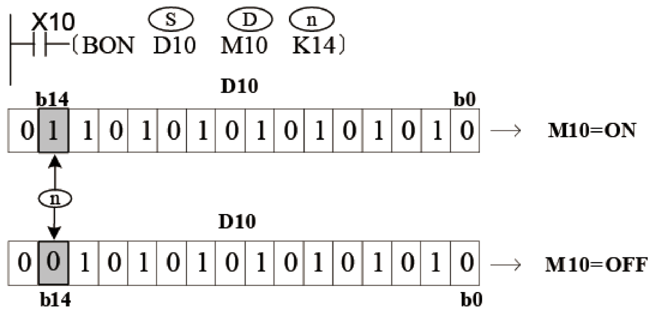
Note

[1] The X element is not supported.

Function and Instruction Description

The BON instruction checks the state of the nth bit in S and stores the result in D.

Instruction Example



When the 14th bit in D10 is 1, M10 is set to ON.

When the 14th bit in D10 is 0, M10 is reset.

When X10 switches from ON to OFF, the state of M10 remains unchanged.

3.6.5.4 SUM

When the driving conditions are met, the SUM instruction counts the ON bits (the value is 1) in the binary data in S and stores the result in D.

SUM – Sum of ON bits

16-bit Instruction	SUM: Continuous execution/SUMP: Pulse execution			
32-bit Instruction	DSUM: Continuous execution/DSUMP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data to be counted	Data to be counted, or address of the element that stores the data	-	INT/DINT
D	Counting result	Address of the element that stores the result	-	INT/DINT

Table 3–137 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

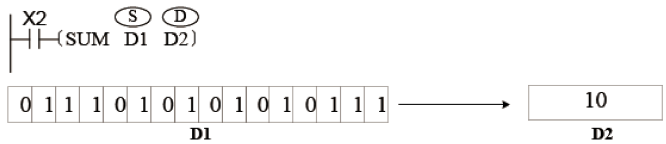
Function and Instruction Description

The SUM instruction counts the number of ON bits (the value is 1) in the binary data in S and stores the result in D.

When DSUM or DSUMP is executed, the number of ON bits (the value is 1) among the 32 bits in (S+1, S) is written to D, and all bits in D+1 are set to 0.

If all bits in S are 0, the zero flag M8020 is set to ON.

Instruction Example



The number of ON bits (the value is 1) in D1 is counted and the result is stored in D2.

3.6.5.5 RAND

The RAND instruction generates a random number within a specified range.
RAND – Random number generation within limits

16-bit Instruction	-			
32-bit Instruction	RAND: Continuous execution/RANDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Random number lower limit	Lower limit of the random number		DINT
S2	Random number upper limit	Upper limit of the random number		DINT
S3	Random seed	Random seed, which is used as input. The random number generated varies with the seed.		DINT
D	Random number	Generated random number		DINT

Table 3–138 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S3	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The RAND instruction generates a random number within a specified range. The random number is generated by the random seed. When the range is determined, there is a one-to-one correspondence between the random seed and the generated random number. That is, when the random seed changes, the generated random number changes accordingly.

The parameters of this instruction are described as follows:

- Lower limit: Lower limit of the random number
- Upper limit: Upper limit of the random number
- Random seed: Input for generating the random number, which is not restricted by the upper and lower limits
- Random number: Generated random number, which is between the upper and lower limits An error is returned when S1 is greater than S2.

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

		Element Name	Data Type	Display Format	Current Value
1	...	D100	DINT	Dec	0
2	...	D200	DINT	Dec	10000
3	...	D300	DINT	Dec	1000
4	...	D400	DINT	Dec	9247
5	...				

3.6.5.6 XCH

When the driving conditions are met, the XCH instruction exchanges the data in S and D.

XCH – Data exchange

16-bit Instruction	XCH: Continuous execution/XCHP: Pulse execution			
32-bit Instruction	DXCH: Continuous execution/DXCHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data 1	Word element 1 that stores the data to be exchanged	-	INT/DINT
D	Data 2	Word element 2 that stores the data to be exchanged	-	INT/DINT

Table 3–139 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

The XCH instruction requires contact driving and has two operands. It exchanges the values in S and D.

Instruction Example

Example 1

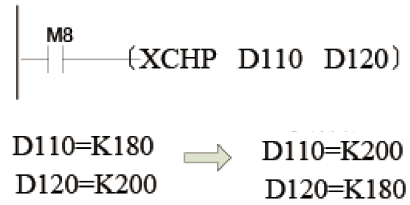


Figure 3-7 Before execution and after execution

Example 2

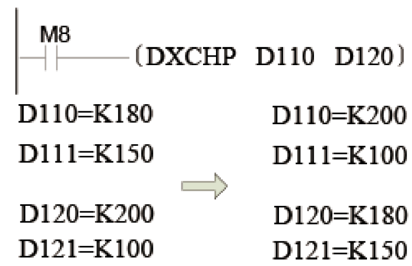


Figure 3-8 Before execution and after execution

3.6.5.7 ABS

The ABS instruction calculates the absolute value of an integer.
ABS – Absolute value of integer

16-bit Instruction	ABS: Continuous execution/ABSP: Pulse execution			
32-bit Instruction	DABS: Continuous execution/DABSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Source data for which the absolute value is calculated	-	INT/DINT
D	Absolute value	Obtained absolute value	-	INT/DINT

Table 3–140 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-

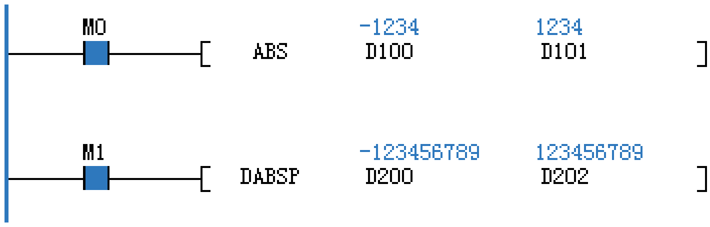
Function and Instruction Description

The ABS instruction requires contact driving and has two operands. It assigns the absolute value of the integer in S to D.

When the 32-bit instruction (DABS) is executed, the operation involves the variable units (S+1, S) and (D +1, D).

S: Integer for which the absolute value is to be calculated

D: Obtained absolute value



3.6.5.8 EABS

The EABS instruction calculates the absolute value of a floating-point number.

EABS – Absolute value of floating-point number

16-bit Instruction	-			
32-bit Instruction	DEABS: Continuous execution/DEABSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Source data for which the absolute value is calculated	-	REAL
D	Absolute value	Obtained absolute value	-	REAL

Table 3–141 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-

Function and Instruction Description

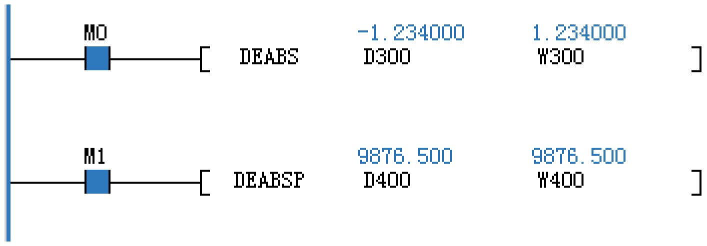
The EABS instruction calculates the absolute value a single-precision floating-point number. It requires contact driving. When it is executed, the absolute value of the floating-point number in S is assigned to D.

S: Floating-point number for which the absolute value is to be calculated

D: Obtained absolute value

Instruction Example

When the flow is active, the absolute value of the source floating-point number is calculated and then assigned to the target register or variable.



3.6.5.9 EFMOV

When the driving conditions are met, the EFMOV instruction transfers the floating-point number data in S to the n registers starting from the address specified in D.

EFMOV – Multi-point floating-point move

16-bit Instruction	-			
32-bit Instruction	DEFMOV: Continuous execution/DEFMOVP: Pulse execution, 13 steps			
Operand	Name	Description	Range	Data Type
S	Source data	Data to be transferred, or address of the floating-point element that stores the data	-	-
D	Start address of the transfer destination	Start address of word elements that store the transferred data	-	Array*n
n	Target number	Number of points of the word element to which the data is transferred	1 to 512	-

Table 3–142 List of elements

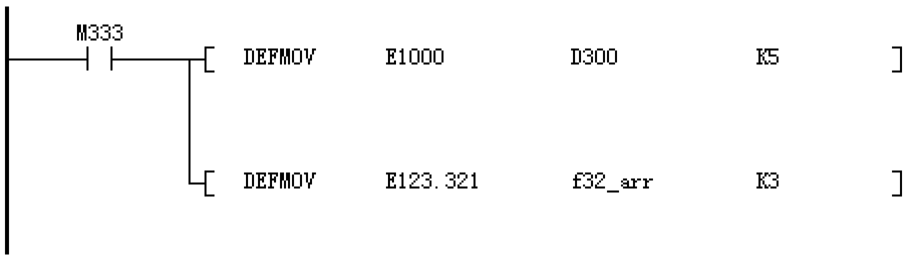
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	√	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The EFMOV instruction requires contact driving and has three operands. It copies the floating-point number data in S to the n units starting from the address specified in D.

n ranges from 1 to 512.

Instruction Example



When M333 is ON, the calculation result is as follows:

1000 Floating point —> D300 ,D301

1000 Floating point —> D3 02,D303

1000 Floating point —> D304 ,D305

1000 Floating point —> D306 ,D307

1000 Floating point —> D308 ,D309

123.321 Floating point —> f32_arr[0]

123.321 Floating point —> f32_arr[1]

123.321 Floating point —> f32_arr[2]

D300	REAL	Dec	1000.000
D302	REAL	Dec	1000.000
D304	REAL	Dec	1000.000
D306	REAL	Dec	1000.000
D308	REAL	Dec	1000.000
D310	REAL	Dec	0.000000
⊟ f32_arr	REAL[5]	Dec	
f32_arr[0]	REAL	Dec	123.3210
f32_arr[1]	REAL	Dec	123.3210
f32_arr[2]	REAL	Dec	123.3210
f32_arr[3]	REAL	Dec	0.000000
f32_arr[4]	REAL	Dec	0.000000

3.6.5.10 CCD

When the driving conditions are met, the CCD instruction calculates the checksum of the n data entries starting from S and stores the result in D. The XOR operation result is stored in D+1.

CCD – Check code

16-bit Instruction	CCD: Continuous execution/CCDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Data source	Start address of consecutive units that store the variables for which checksum will be calculated	-	INT, array*n
D	Operation result	Checksum result stored in D; XOR logical operation result stored in D+1	-	INT, array*2
n	Checked byte count	Number of bytes contained in checked variables	1 to 256	INT

Table 3–143 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The CCD instruction performs two types of checksum operations on n variables starting from S and stores the summation result in D and the XOR logical operation result in D+1. The string checksum operation ensures correct data transfer during communication. Where,

Summation is the process where the values of n variables are directly added together.

The XOR logical operation is described as follows:

- Convert the variables involved in the operation into binary numbers.
- Count the number of variables of which bit 0 is 1. If it is an even number, bit 0 of the XOR result is 0; if it is an odd number, bit 0 of the XOR result is 1.

- Count the number of variables of which bit 1 is 1. If it is an even number, bit 1 of the XOR result is 0; if it is an odd number, bit 1 of the XOR result is 1.

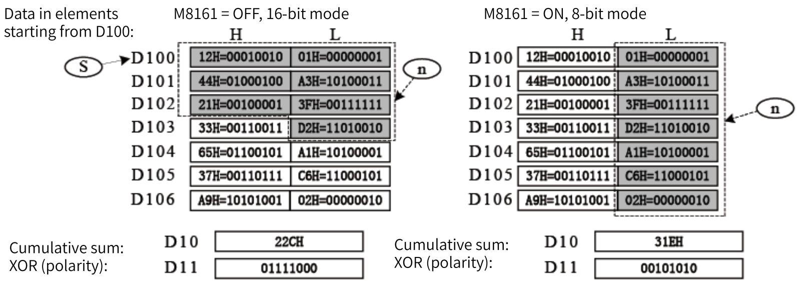
Calculate bit 2 to bit 7 in a similar way. Convert the resulting binary number into a hexadecimal equivalent, which is the XOR operation result (or called a polarity value).

Note

Note during programming that instructions including HEX, ASCI, and CCD share the M8161 mode flag.

Instruction Example

The M8161 flag determines the variable width mode. When M8161 is OFF, the 16-bit mode is used, whereby both the high- and low-order bytes of variables are taken for the operation. When M8161 is ON, the 8-bit mode is used, whereby only the low-order bytes of variables are taken for the operation and the high-order bytes are discarded. In this case, the length of the actually used variable area is increased.



3.6.5.11 CRC

Cyclic redundancy check (CRC) is commonly used during communication. The CRC instruction is used to calculate the CRC code.

CRC – CRC code calculation

16-bit Instruction	CRC: Continuous execution/CRCP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start address of elements that store the data for CRC code calculation (RTU mode)	-	INT, array*n
n	Data count	Number of operated data entries (K1 to K256)	1 to 256	INT
D	Result	Start address of elements that store the operation result	-	INT, array*2

Table 3–144 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	√	√	-	-
n	-	-	-	√	√	-	-	-	-

Function and Instruction Description

16-bit conversion mode: When M8161 is OFF, the CRC instruction takes the high-order 8 bits and low-order 8 bits (n data points in total) starting from [S] in the unit of 16 bits for CRC code calculation and stores the result in the high-order 8 bits and low-order 8 bits in [D].

8-bit conversion mode: When M8161 is ON, the CRC instruction takes the low-order 8 bits (n data points in total) starting from [S] in the unit of 8 bits for CRC code calculation and stores the low-order 8 bits of the result in [D] and the high-order 8 bits of the result in [D+1].

Errors

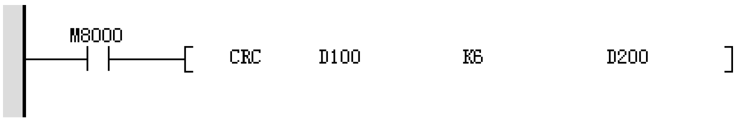
An error is returned in the following conditions:

The error flag M8067 is set to ON, and the error code is stored in D8067.

Error 6706 is returned when n is out of range.

Instruction Example

When M8161 is ON, the 8-bit conversion mode is used. The low-order 8 bits of elements D100 to D105 are taken for CRC code calculation. The result is stored in the low-order 8 bits of D200 and D201.



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1
D101	16-bit INT	Hex	0x2
D102	16-bit INT	Hex	0x4
D103	16-bit INT	Hex	0x20
D104	16-bit INT	Hex	0x0
D105	16-bit INT	Hex	0x12
	16-bit INT	Dec	
M8161	BOOL	Bin	ON
	16-bit INT	Dec	
D200	16-bit INT	Hex	0xF8
D201	16-bit INT	Hex	0xFD

When M8161 is OFF, the 16-bit conversion mode is used. The low-order 8 bits of elements D100 to D105 are taken for CRC code calculation. The result is stored in the high-order 8 bits and low-order 8 bits of D200.

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1
D101	16-bit INT	Hex	0x2
D102	16-bit INT	Hex	0x4
D103	16-bit INT	Hex	0x20
D104	16-bit INT	Hex	0x0
D105	16-bit INT	Hex	0x12
	16-bit INT	Dec	
M8161	BOOL	Bin	OFF
	16-bit INT	Dec	
D200	16-bit INT	Hex	0xB202
D201	16-bit INT	Hex	0x0

3.6.5.12 LRC

The LRC instruction calculates the longitudinal redundancy check (LRC) code in ASCII mode.

LRC – LRC code calculation

16-bit Instruction	LRC: Continuous execution/LRCP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start address of elements that store the data for LRC code calculation (ASCII mode)		INT, array*n
n	Data count	Number of operated data entries (value range: K1 to K256), which must be an even number	1 to 256	INT
D	Result	Register that stores the operation result		INT, array*2

Table 3–145 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The LRC code is acquired by calculating the two's complement of the sum of values within the range from the communication address to the end of the data content.

The following are two examples. 01 H + 03 H + 21 H + 02 H + 00 H + 02 H = 29 H, and the two's complement of the sum is D7H (which corresponds to the ASCII codes 44H and 37H).

16-bit conversion mode: When M8161 is OFF, the LRC instruction takes the high-order 8 bits and low-order 8 bits (n data points in total) starting from [S] in the unit of 16 bits for LRC code calculation and stores the result in the high-order 8 bits and low-order 8 bits in [D].

8-bit conversion mode: When M8161 is ON, the LRC instruction takes the low-order 8 bits (n data points in total) starting from [S] in the unit of 8 bits for LRC code calculation and stores the low-order 8 bits of the result in [D] and the high-order 8 bits of the result in [D+1].

Errors

An error is returned in the following conditions:

- n is out of the specified range.
- n is an odd number.

Instruction Example

1. 16-bit mode (M8161 = OFF)

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x3130
D101	16-bit INT	Hex	0x3330
D102	16-bit INT	Hex	0x3132
D103	16-bit INT	Hex	0x3230
D104	16-bit INT	Hex	0x3030
D105	16-bit INT	Hex	0x3230
D200	16-bit INT	Hex	0x3744
M8161	BOOL	Bin	OFF
	16-bit INT	Dec	

2. 8-bit mode (M8161 = ON)

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x30
D101	16-bit INT	Hex	0x31
D102	16-bit INT	Hex	0x30
D103	16-bit INT	Hex	0x33
D104	16-bit INT	Hex	0x32
D105	16-bit INT	Hex	0x31
D106	16-bit INT	Hex	0x30
D107	16-bit INT	Hex	0x32
D108	16-bit INT	Hex	0x30
D109	16-bit INT	Hex	0x30
D110	16-bit INT	Hex	0x30
D111	16-bit INT	Hex	0x32
D112	16-bit INT	Hex	0x0
D200	16-bit INT	Hex	0x44
D201	16-bit INT	Hex	0x37
M8161	BOOL	Bin	ON
	16-bit INT	Hex	

3.7 Matrix Instructions

3.7.1 Matrix Operation Instructions

3.7.1.1 Instruction List

The following table lists the matrix operation instructions.

Instruction Category	Instruction	Function
Matrix operation instruction	BK+	Block data addition
	BK–	Block data subtraction
	MAND	Matrix AND
	MOR	Matrix OR
	MXOR	Matrix XOR
	MXNR	Matrix XNOR
	MINV	Matrix inversion

3.7.1.2 BK+

The BK+ instruction adds binary block data.

BK+ – Block data addition

16-bit Instruction	BK+: Continuous execution/BK+P: Pulse execution			
32-bit Instruction	DBK+: Continuous execution/DBK+P: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Source address 1	Start number of elements that store the data for which the addition operation is performed	-	INT/DINT, array*n
S2	Source address 2	Constant for which the addition operation is performed, or start number of elements that store the data for which the addition operation is performed	-	INT/DINT, array*n
D	Destination address	Start number of elements that store the operation result	-	INT/DINT, array*n
n	Data count	Number of data entries involved in an operation	1 to 256	INT/DINT

Note

n indicates the number of data entries to be operated. If it is a constant, only a fixed number of data entries can be operated; if it is a variable, the number of data entries to be operated can be changed by adjusting the value of n.

Table 3–146 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BK+ instruction adds the n data entries (16- or 32-bit) starting from [S1] and the n data entries (16- or 32-bit) starting from [S2] together and stores the result in n units (16- or 32-bit) starting from [D].

[S1+0]	K1111
[S1+1]	K1111
...	...
[S1+n-2]	K1111
[S1+n-1]	K1111

+

[S2+0]	K1111
[S2+1]	K-2222
...	...
[S2+n-2]	K3333
[S2+n-1]	K4444

=

[D+0]	K2222
[D+1]	K-1111
...	...
[D+n-2]	K4444
[D+n-1]	K5555

A signed constant (16- or 32-bit) can be directly specified in [S2].

[S1+0]	K1111
[S1+1]	K1111
...	...
[S1+n-2]	K1111
[S1+n-1]	K1111

+ K2222 =

[D+0]	K3333
[D+1]	K3333
...	...
[D+n-2]	K3333
[D+n-1]	K3333

If the elements starting from [S1], [S2], or [D] are beyond the corresponding element range, an error is returned and the instruction is not executed.

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Decimal	1111
D101	16-bit INT	Decimal	1111
D102	16-bit INT	Decimal	1111
D103	16-bit INT	Decimal	1111
D104	16-bit INT	Decimal	1111
D105	16-bit INT	Decimal	0
D200	16-bit INT	Decimal	2222
D201	16-bit INT	Decimal	2222
D202	16-bit INT	Decimal	2222
D203	16-bit INT	Decimal	2222
D204	16-bit INT	Decimal	2222
D205	16-bit INT	Decimal	0
R100	16-bit INT	Decimal	3333
R101	16-bit INT	Decimal	3333
R102	16-bit INT	Decimal	3333
R103	16-bit INT	Decimal	3333
R104	16-bit INT	Decimal	3333
R105	16-bit INT	Decimal	0

3.7.1.3 BK–

The BK– instruction subtracts binary block data.

BK– – Block data subtraction

16-bit Instruction	BK–: Continuous execution/BK–P: Pulse execution			
32-bit Instruction	DBK–: Continuous execution/DBK–P: Pulse execution			
Operand	Name	Description	Range	Data Type

S1	Source address 1	Start number of elements that store the data for which the subtraction operation is performed		INT/DINT, array*n
S2	Source address 2	Constant for which the subtraction operation is performed, or start number of elements that store the data for which the subtraction operation is performed		INT/DINT, array*n
D	Destination address	Start number of elements that store the operation result		INT/DINT, array*n
n	Data count	Number of data entries involved in an operation	1 to 256	INT/DINT

Table 3–147 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The BK– instruction subtracts the n data entries (16- or 32-bit) starting from [S2] from the n data entries (16- or 32-bit) starting from [S1] and stores the result in n units (16- or 32-bit) starting from [D].

[S1+0]	K1111
[S1+1]	K1111
...	...
[S1+n-2]	K1111
[S1+n-1]	K1111

−

[S2+0]	K1111
[S2+1]	K-2222
...	...
[S2+n-2]	K3333
[S2+n-1]	K4444

=

[D+0]	K0
[D+1]	K3333
...	...
[D+n-2]	K-2222
[D+n-1]	K-3333

A signed constant (16- or 32-bit) can be directly specified in [S2].

[S1+0]	K1111
[S1+1]	K1111
...	...
[S1+n-2]	K1111
[S1+n-1]	K1111

− K-2222 =

[D+0]	K3333
[D+1]	K3333
...	...
[D+n-2]	K3333
[D+n-1]	K3333

If the elements starting from [S1], [S2], or [D] are beyond the corresponding element range, an error is returned and the instruction is not executed.

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Dec	3333
D101	16-bit INT	Dec	3333
D102	16-bit INT	Dec	3333
D103	16-bit INT	Dec	3333
D104	16-bit INT	Dec	3333
D105	16-bit INT	Dec	0
D200	16-bit INT	Dec	2222
D201	16-bit INT	Dec	2222
D202	16-bit INT	Dec	2222
D203	16-bit INT	Dec	2222
D204	16-bit INT	Dec	2222
D205	16-bit INT	Dec	0
R100	16-bit INT	Dec	1111
R101	16-bit INT	Dec	1111
R102	16-bit INT	Dec	1111
R103	16-bit INT	Dec	1111
R104	16-bit INT	Dec	1111
R105	16-bit INT	Dec	0

3.7.1.4 MAND

The MAND instruction performs an AND operation on the matrices and stores the result in D.

MAND – Matrix AND

16-bit Instruction	MAND: Continuous execution/MANDP: Pulse execution			
32-bit Instruction	DMAND: Continuous execution/DMANDP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Matrix 1	Operand element 1 for the operation	-	INT/DINT, array*n
S2	Matrix 2	Operand element 2 for the operation	-	INT/DINT, array*n
D	Operation result	Start number of elements for storing the operation result	-	INT/DINT, array*n
n	Data group quantity	Number of data groups involved in an operation; ranging from 1 to 256	1 to 256	INT/DINT

Table 3–148 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The MAND instruction performs an AND operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result of the AND operation is 1 when the values of both bits are 1; otherwise, the result is 0.

Assume that n is 4. A matrix AND operation is performed as follows.

Note

For the 16-bit instruction, n indicates the number of words; for the 32-bit instruction, n indicates the number of dwords.

	bit15															bit0
[S1+0]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
[S1+1]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
[S1+2]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
[S1+3]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0

AND

[S2+0]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1
[S2+1]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1
[S2+2]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1
[S2+3]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1

[S2+0]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1
[S2+1]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1
[S2+2]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1
[S2+3]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xFFFF
D101	16-bit INT	Hex	0xFFFF
D102	16-bit INT	Hex	0xFFFF
	16-bit INT	Dec	
D200	16-bit INT	Hex	0xFF
D201	16-bit INT	Hex	0xFF00
D202	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0xFF
R101	16-bit INT	Hex	0xFF00
R102	16-bit INT	Hex	0xAAAA

3.7.1.5 MOR

The MOR instruction performs an OR operation on the matrix and stores the result in D.

MOR – Matrix OR

16-bit Instruction	MOR: Continuous execution/MORP: Pulse execution			
32-bit Instruction	DMOR: Continuous execution/DMORP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Matrix 1	Operand element 1 for the operation	-	INT/DDINT, array*n
S2	Matrix 2	Operand element 2 for the operation	-	INT/DDINT, array*n
D	Operation result	Start number of elements for storing the operation result	-	INT/DDINT, array*n
n	Data group quantity	Number of data groups involved in an operation; ranging from 1 to 256	1 to 256	INT/DDINT

Table 3–149 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The MOR instruction performs an OR operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result of the OR operation is 1 when the value of either bit is 1; otherwise, the result is 0.

Assume that n is 4. A matrix OR operation is performed as follows.

Note

For the 16-bit instruction, n indicates the number of words; for the 32-bit instruction, n indicates the number of dwords.

	bit15															bit0
[S1+0]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
[+1]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
[+2]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
[+3]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0

OR

[S2+0]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+1]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1
[+2]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+3]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1

[D+0]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+1]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1
[+2]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+3]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1

Instruction Example

3.7.1.6 MXNR

The MXNR instruction performs an XNOR operation on the matrix and stores the result in D.
MXNR: Matrix XNOR

16-bit Instruction	MXNR: Continuous execution/MXNRP: Pulse execution			
32-bit Instruction	DMXNR: Continuous execution/DMXNRP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Matrix 1	Operand element 1 for the operation	-	INT/DINT, array*n
S2	Matrix 2	Operand element 2 for the operation	-	INT/DINT, array*n
D	Operation result	Start number of elements for storing the operation result	-	INT/DINT, array*n
n	Data group quantity	Number of data groups involved in an operation; ranging from 1 to 256	1 to 256	INT/DINT

Table 3–150 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The MXNR instruction performs an XNOR operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result of the XNOR operation is 1 when the values of the two bits are different; otherwise, the result is 0.

Assume that n is 4. A matrix XNR operation is performed as follows.

Note

For the 16-bit instruction, n indicates the number of words; for the 32-bit instruction, n indicates the number of dwords.

	bit15															bit0
[S1+0]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
[S1+1]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0
[S1+2]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
[S1+3]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0

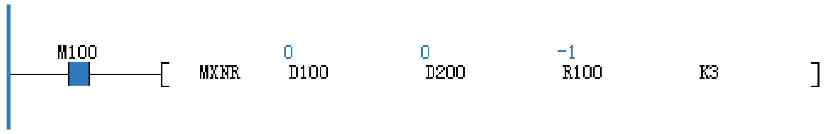
XNR

[S2+0]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1
[S2+1]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1
[S2+2]	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1
[S2+3]	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1

↓

[S2+0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
[S2+1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
[S2+2]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
[S2+3]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x0
D101	16-bit INT	Hex	0x0
D102	16-bit INT	Hex	0x0
	16-bit INT	Dec	
D200	16-bit INT	Hex	0x0
D201	16-bit INT	Hex	0x0
D202	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0xFFFF
R101	16-bit INT	Hex	0xFFFF
R102	16-bit INT	Hex	0x5555

3.7.1.7 MXOR

The MXOR instruction performs an XOR operation on the matrix and stores the result in D.

MXOR – Matrix XOR

16-bit Instruction	MXOR: Continuous execution/MXORP: Pulse execution			
32-bit Instruction	DMXOR: Continuous execution/DMXORP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Matrix 1	Operand element 1 for the operation	-	INT/DINT, array*n
S2	Matrix 2	Operand element 2 for the operation	-	INT/DINT, array*n
D	Operation result	Start number of elements for storing the operation result	-	INT/DINT, array*n
n	Data group quantity	Number of data groups involved in an operation	1 to 256	INT/DINT

Table 3–151 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The MXOR instruction performs an XOR operation by bit on the n groups of data starting from [S1] and the n groups of data starting from [S2] and stores the result in elements starting from [D].

The result of the XOR operation is 1 when the values of the two bits are different; otherwise, the result is 0.

Assume that n is 4. A matrix XOR operation is performed as follows.

Note

For the 16-bit instruction, n indicates the number of words; for the 32-bit instruction, n indicates the number of dwords.

---

	bit15															bit0
[S1+0]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+1]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1
[+2]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
[+3]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0

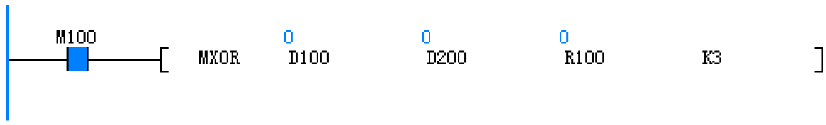
XOR (bit-based XOR)

[S2+0]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+1]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1
[+2]	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1
[+3]	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1

↓

[D+0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[+1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[+2]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
[+3]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x0
D101	16-bit INT	Hex	0x0
D102	16-bit INT	Hex	0x5555
	16-bit INT	Dec	
D200	16-bit INT	Hex	0x0
D201	16-bit INT	Hex	0x0
D202	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x0
R101	16-bit INT	Hex	0x0
R102	16-bit INT	Hex	0xFFFF

3.7.1.8 MINV

The MINV instruction inverts all bits of the specified matrix.

MINV – Matrix inversion

16-bit Instruction	MINV: Continuous execution/MINVP: Pulse execution			
32-bit Instruction	DMINV: Continuous execution/DMINVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Matrix	Operand element for the operation	-	INT/DINT, array*n
D	Operation result	Start number of elements for storing the operation result	-	INT/DINT, array*n
n	Data group quantity	Number of data groups involved in an operation; ranging from 1 to 256	1 to 256	INT/DINT

Table 3–152 List of elements

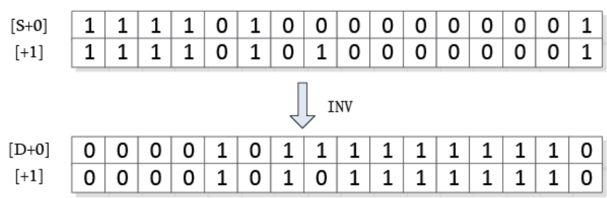
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	√	-	-	-
D	-	-	-	√	√	√	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

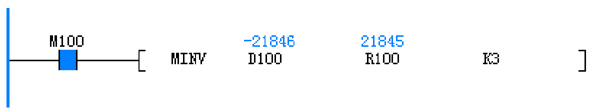
The MINV instruction inverts the n groups of data starting from [S] by bit and stores the result in elements starting from [D].

Note

For the 16-bit instruction, n indicates the number of words; for the 32-bit instruction, n indicates the number of dwords.



Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0xAAAA
D101	16-bit INT	Hex	0xAAAA
D102	16-bit INT	Hex	0xAAAA
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x5555
R101	16-bit INT	Hex	0x5555
R102	16-bit INT	Hex	0x5555

3.7.2 Matrix Comparison Instructions

3.7.2.1 Instruction List

The following table lists the matrix comparison instructions.

Instruction Category	Instruction	Function
Matrix comparison instruction	BKCMP=	Matrix comparison equal to (S1 = S2)
	BKCMP>	Matrix comparison greater than (S1 > S2)
	BKCMP<	Matrix comparison less than (S1 < S2)
	BKCMP<>	Matrix comparison not equal to (S1 ≠ S2)
	BKCMP<=	Matrix comparison less than or equal to (S1 ⩽ S2)
	BKCMP>=	Matrix comparison greater than or equal to (S1 ⩾ S2)

3.7.2.2 BKCMP#

The following instructions compare block data according to the comparison condition set in each instruction.

BKCMP= – Matrix comparison equal to(S1 = S2)

BKCMP> – Matrix comparison greater than(S1 > S2)

BKCMP< – Matrix comparison less than(S1 < S2)

BKCMP<> – Matrix comparison not equal to(S1 ≠ S2)

BKCMP<= – Matrix comparison less than or equal to(S1 ⩽ S2)

BKCMP>= – Matrix comparison greater than or equal to(S1 ⩾ S2)

16-bit Instruction	BKCMP=: Continuous execution/BKCMP=P: Pulse execution
32-bit Instruction	DBKCMP=: Continuous execution/DBKCMP=P: Pulse execution
16-bit Instruction	BKCMP>: Continuous execution/BKCMP>P: Pulse execution
32-bit Instruction	DBKCMP>: Continuous execution/DBKCMP>P: Pulse execution
16-bit Instruction	BKCMP<: Continuous execution/BKCMP<P: Pulse execution
32-bit Instruction	DBKCMP<: Continuous execution/DBKCMP<P: Pulse execution
16-bit Instruction	BKCMP<>: Continuous execution/BKCMP<>P: Pulse execution
32-bit Instruction	DBKCMP<>: Continuous execution/DBKCMP<>P: Pulse execution
16-bit Instruction	BKCMP>=: Continuous execution/BKCMP>=P: Pulse execution
32-bit Instruction	DBKCMP>=: Continuous execution/DBKCMP>=P: Pulse execution
16-bit Instruction	BKCMP<=: Continuous execution/BKCMP<=P: Pulse execution
32-bit Instruction	DBKCMP<=: Continuous execution/DBKCMP<=P: Pulse execution

Operand	Name	Description	Range	Data Type
S1	Comparand	Comparand, or number of the element that stores the comparand	-	INT/DINT, array*n
S2	Compared value	Start number of elements that store the source data to be compared	-	INT/DINT, array*n
D	Destination address	Start number of elements that store the comparison result	-	BOOL, array*n
n	Data count	Number of data entries involved in an operation	1 to 256	INT/DINT

Table 3–153 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
D	√[1]	√	√	-	-	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Note

- # indicates the comparison operator =, >, <, <>, <=, or >=.
- [1] The X element is not supported.

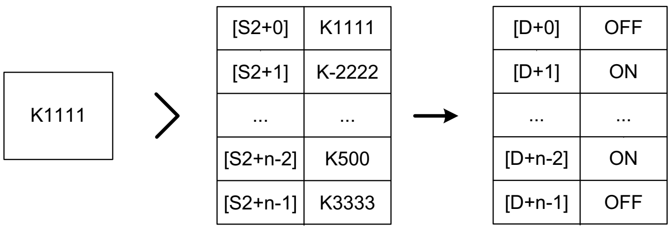
Function and Instruction Description

The BKCMP# instruction compares the n data entries (16- or 32-bit) starting from [S1] with the n data entries (16- or 32-bit) starting from [S2] and stores the comparison result in n units (16- or 32-bit) starting from [D].

Take the BKCMP> instruction an example.

A signed constant (16- or 32-bit) can be directly specified in [S1].

Take the BKCMP> instruction an example.



M8333 is set to ON when all of the n results starting from [D] are ON.

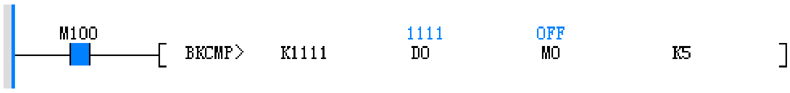
An error is returned in the following conditions, in which the instruction is not executed:

1. The elements starting from [S1], [S2], or [D] are beyond the corresponding element range.

2. 32-bit variables are used in a 16-bit instruction.

You need to use 32-bit instructions (such as DBKCMP=, DBKCMP>, and DBKCMP<) to compare 32-bit variables.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D0	16-bit INT	Dec	1111
D1	16-bit INT	Dec	-2222
D2	16-bit INT	Dec	0
D3	16-bit INT	Dec	500
D4	16-bit INT	Dec	3333
M100	BOOL	Bin	ON
	16-bit INT	Dec	
M0	BOOL	Bin	OFF
M1	BOOL	Bin	ON
M2	BOOL	Bin	ON
M3	BOOL	Bin	ON
M4	BOOL	Bin	OFF

3.8 String Instructions

3.8.1 Instruction List

The following table lists the string instructions.

Instruction Category	Instruction	Function
String instruction	STR	Conversion from integer into string
	STRMOV	String assignment
	VAL	Conversion from string into integer
	ESTR	Conversion from binary floating-point into string
	EVAL	Conversion from string into binary floating-point
	$ADD	Character string linking
	LEN	Character string length detection
	INSTR	Character string search
	RIGHT	String data extraction from the right
	LEFT	String data extraction from the left
	MIDR	Random extraction of character string
	MIDW	Random replacement of character string
	$MOV	Character string transfer

3.8.2 STR

The STR instruction converts integers into character strings (ASCII codes).

STR – Conversion from integer into string

16-bit Instruction	STR: Continuous execution/STRP: Pulse execution
32-bit Instruction	DSTR: Continuous execution/DSTRP: Pulse execution

Operand	Name	Description	Range	Data Type
S1	Data to be converted	Number of the element that stores the integer to be converted	-	INT/DINT, array*2
S2	Data to be converted	Start number of elements that store the total number of characters contained in a string after conversion	-	INT/DINT
D	Output	Start number of elements that store the character string after conversion	-	INT/DINT, array*indeterminate

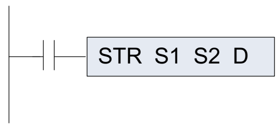
Table 3–154 List of elements

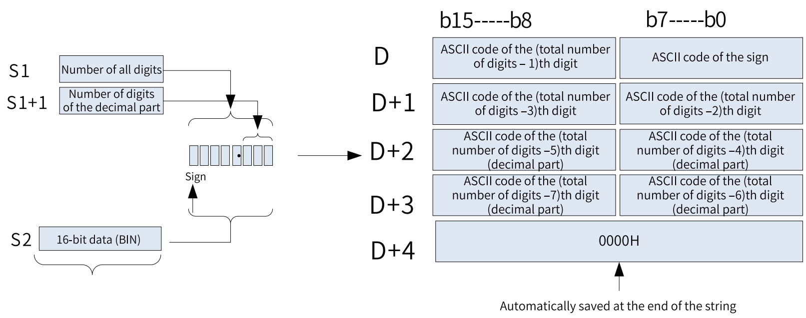
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

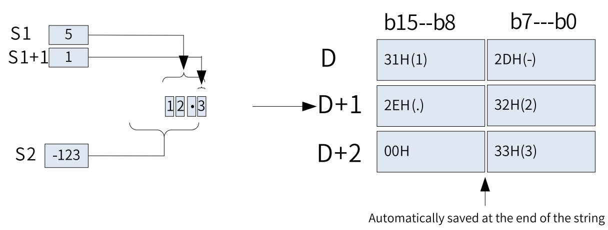
Function and Instruction Description

1. 16-bit operation (STR and STRP)

   The STR/STRP instruction inserts a decimal point in the position specified by [S1] and [S1+1], converts the 16-bit binary number in [S2] into a character string, and stores the result in elements starting from [D].

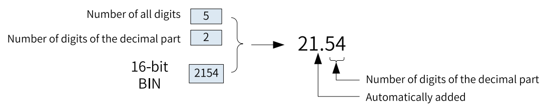




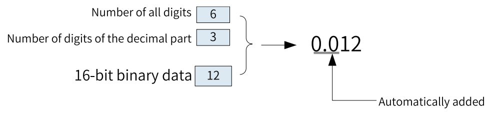


- The total number of digits specified in [S1] ranges from 2 to 8.
- The number of digits of the decimal part specified in [S1+1] ranges from 0 to 5. Make sure that the following condition is met: [S1+1] ⩽ [S1] – 3.
- The 16-bit binary number to be converted ranges from –32768 to +32767. The character string after conversion is stored in elements starting from [D], as shown in the following figure.

- The sign bit stores "space" (20H) when the 16-bit binary number in [S2] is positive or "–" (2DH) when it is negative.
- When the number of digits of the decimal part in [S1+1] is set to any value other than 0, the decimal point "." (2EH) is automatically added in the "number of digits of the decimal part + 1"th digit. No decimal point is inserted when the value in [S1+1] is 0.

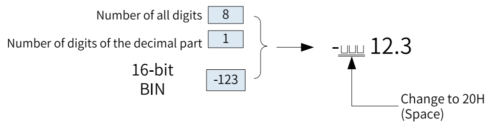


If the number of digits of the decimal part in [S1+1] is greater than the number of digits of 16-bit binary data in [S2], data is automatically aligned to the right and "0" (30H) is automatically added on the left during conversion.



If the number of all digits in [S1] excluding the sign and decimal point is greater than the number of digits of 16-bit binary data in [S2], "space" (20H) is inserted between the sign and the numeric value.

If the number of digits of 16-bit binary data in [S2] is larger, an error occurs.

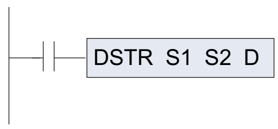


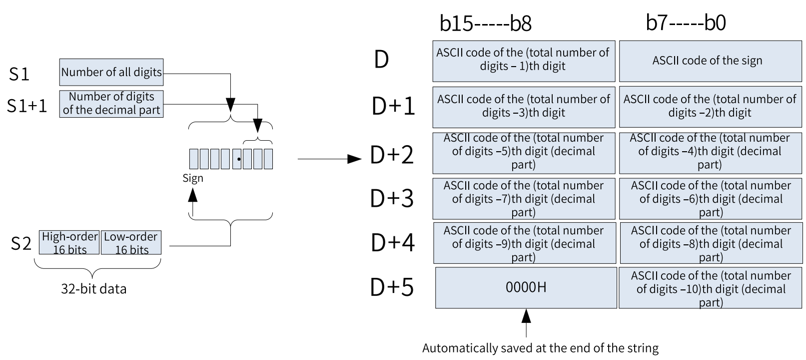
"00H" indicating the end of a character string is automatically added at the end of a converted character string.

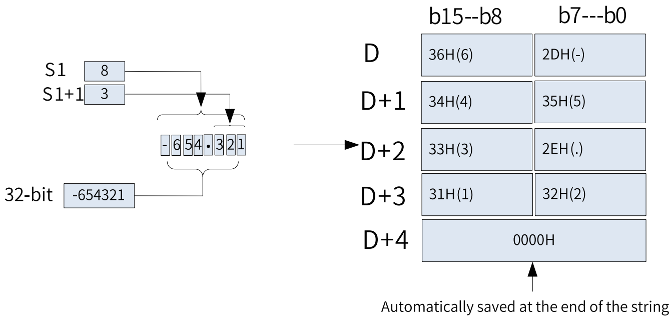
When the total number of digits is even, "0000H" is stored in the element after the one that stores the last character. When the total number of digits is odd, "00H" is stored in the high-order byte (8 bits) of the element that stores the last character.

2. 32-bit operation (DSTR and DSTRP)

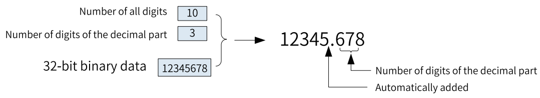
The DSTR/DSTRP instruction inserts a decimal point in the position specified by [S1+1], converts the 32-bit binary number in [S2+1, S2] into a character string, and stores the result in elements starting from [D].



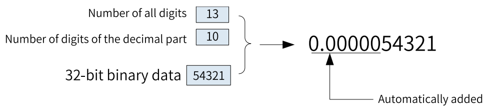




- The total number of digits specified in [S1] ranges from 2 to 13.
- The number of digits of the decimal part specified in [S1+1] ranges from 0 to 10. Make sure that the following condition is met: [S1+1] ⩽ [S1] – 3.
- The 32-bit binary number to be converted ranges from –2147483648 to +2147483647. The character string after conversion is stored in elements starting from [D], as shown in the following figure.
- The sign bit stores "space" (20H) when the 32-bit binary number in [S2] is positive or "–" (2DH) when it is negative.
- When the number of digits of the decimal part in [S1+1] is set to any value other than 0, the decimal point "." (2EH) is automatically added in the "number of digits of the decimal part + 1"th digit. No decimal point is inserted when the value in [S1+1] is 0.

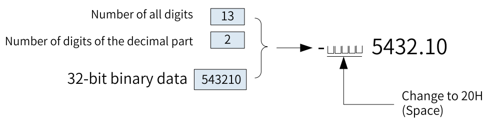


If the number of digits of the decimal part in [S1+1] is greater than the number of digits of 16-bit binary data in [S2], data is automatically aligned to the right and "0" (30H) is automatically added on the left during conversion.



If the number of all digits in [S1] excluding the sign and decimal point is greater than the number of digits of 32-bit binary data in [S2], "space" (20H) is inserted between the sign and the numeric value.

If the number of digits of 32-bit binary data in [S2] is larger, an error occurs.



"00H" indicating the end of a character string is automatically added at the end of a converted character string.

When the total number of digits is even, "0000H" is stored in the element after the one that stores the last character. When the total number of digits is odd, "00H" is stored in the high-order byte (8 bits) of the element that stores the last character.

Errors

An operation error occurs in the following conditions.

- The value in [S1] is out of the following range.

Operation	Value Range
16-bit operation	2 to 8
32-bit operation	2 to 13

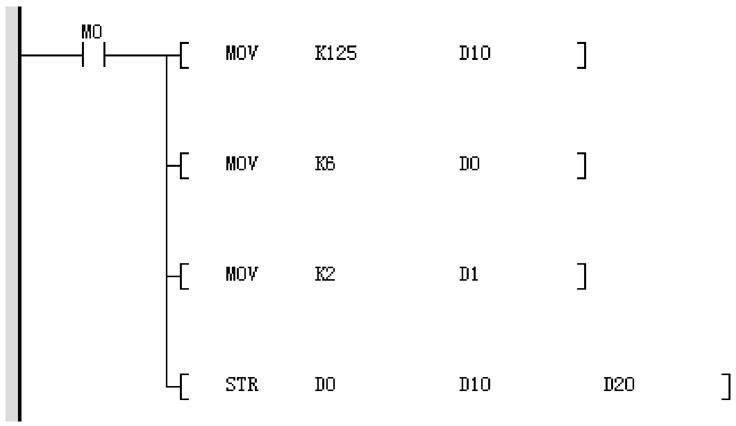
- The value in [S1+1] is out of the following range.

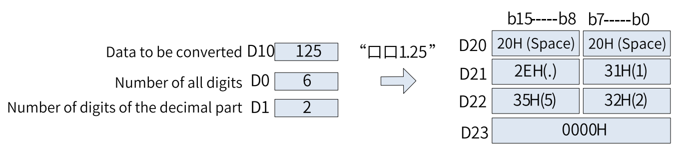
Operation	Value Range
16-bit operation	0 to 5
32-bit operation	0 to 10

- The relationship between the number of all digits specified in [S1] and the number of digits of the decimal part specified in [S1+1] does not meet the following requirements:
  - Total number of digits – 3 ⩾ Number of digits of the decimal part
  - Total number of digits ([S1]) including the digits for the sign and the decimal point < Number of digits of the binary data stored in [S2]
  - The elements starting from [D] for storing the character string are beyond the corresponding element range.

Instruction Example

When M0 is ON, the 16-bit binary number in D10 is converted to a character string in accordance with the digit numbers specified by D0 and D1. The result is stored in D20 to D23.





3.8.3 STRMOV

The STRMOV instruction directly assigns character strings.
STRMOV – String assignment

Instruction	Name	LD Expression
STRMOV	String assignment	—[STRMOV ??? ???]

16-bit Instruction	STRMOV: Continuous execution/STRMOVP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Character string data	Character string data to be assigned	1 to 127	String
D	Storage register	Destination storage register	-	INT, array*strlen [1]

Note

[1]: strlen indicates the string length. One character occupies one byte.

Table 3–155 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	-	-	String [1]
D	-	-	-	√	√	-	-	-	-

Note

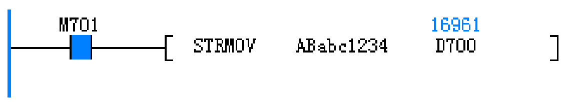
[1]: Only the string data type constants can be input directly.

Function and Instruction Description

The STRMOV instruction supports direct input of character strings. It can be used to receive and transmit character string data in communication.

The character string data is stored in sequence.

Instruction Example



	Element Name	Data Type	Display Format	Current Value
1	D700	16-bit INT	Hex	0x4241
2	D701	16-bit INT	Hex	0x6261
3	D702	16-bit INT	Hex	0x3163
4	D703	16-bit INT	Hex	0x3332
5	D704	16-bit INT	Hex	0x34
6	D705	16-bit INT	Dec	0
7	D706	16-bit INT	Dec	0

3.8.4 VAL

The VAL instruction converts character strings (ASCII codes) into integers.

VAL – Conversion from string into integer

16-bit Instruction	VAL: Continuous execution/VALP: Pulse execution			
32-bit Instruction	DVAL: Continuous execution/DVALP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Data to be converted	Start number of elements that store the string to be converted	-	INT/DINT, array*indetermi-nate
D1	Data to be converted	Number of the element that stores the total number of characters contained in the string	-	INT/DINT, array*2
D2	Output	Start number of elements that store the character string after conversion	-	INT/DINT

Table 3–156 List of elements

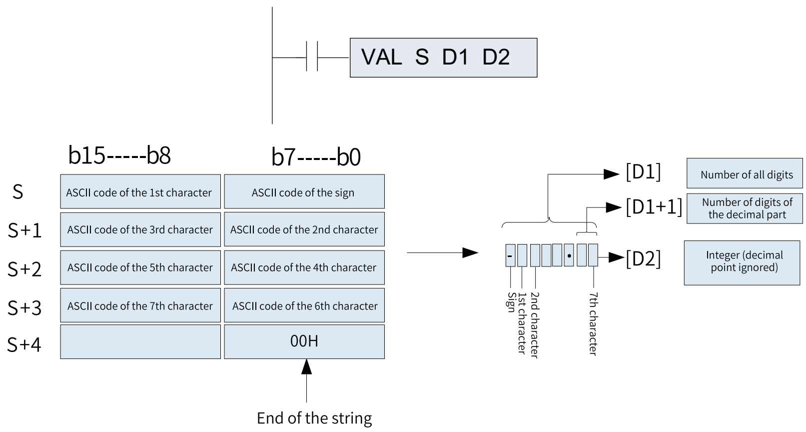
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D1	-	-	-	√	√	-	-	-	-
D2	-	-	-	√	√	-	-	-	-

Function and Instruction Description

1. 16-bit operation (VAL and VALP)

   The VAL/VALP instruction converts the string stored in elements starting from [S] into a 16-bit binary number. The total number of digits of the obtained binary data is stored in [D1], the number of digits of the decimal part is stored in [D1+1], and the binary data is stored in [D2].

   During the conversion, the data stored within the range from [S] to the element that stores 00H is handled as a character string in the unit of byte.



- String data to be converted

  The number of characters in and the value range (decimal point ignored) of the character string to be converted must meet the following requirements:

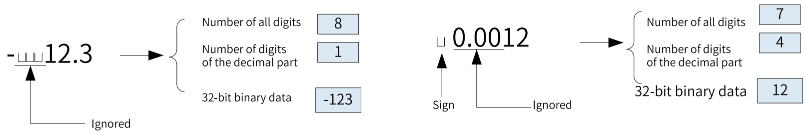
Item	Value Range
Total number of characters	2 to 8
Number of characters of the decimal part	0 to 5
Value range (decimal point ignored)	–32768 to +32767 For example, 123.45 is processed as 12345.

The types of characters that can be used in the character string to be converted are as follows:

Item	Character Type
Positive number	Space (20H)
Negative number	– (2DH)
Decimal point	. (2EH)
Digit	0 (30H) to 9 (39H)

- [D1] stores the total number of digits, including the digits, sign, and decimal point.
- [D1+1] stores the number of digits of the decimal part, that is, the characters to the right of the decimal point "." (2EH).
- [D2] stores the 16-bit binary data converted from a character string with the decimal point ignored.

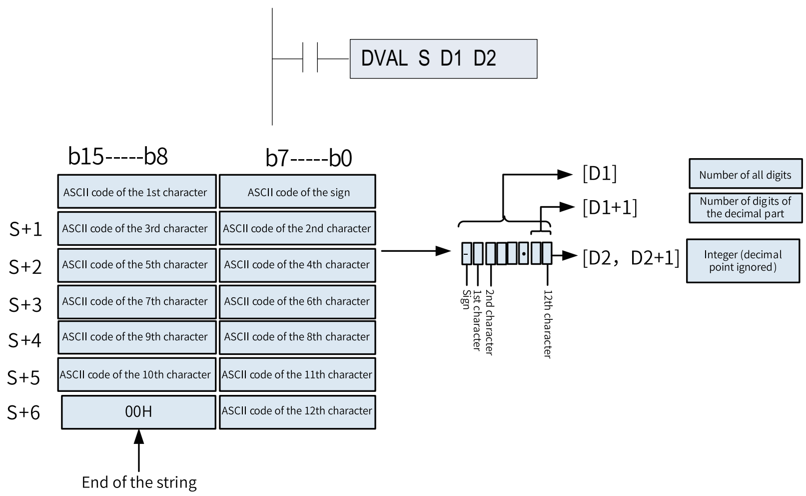
In the character string, "space" (20H) and "0" (30H) characters between the sign and the first number other than "0" are ignored in the conversion to 16-bit binary data.



2. 32-bit operation (DVAL and DVALP)

The DVAL/DVALP instruction converts the string stored in elements starting from [S] into a 32-bit binary number. The total number of digits of the obtained binary data is stored in [D1], the number of digits of the decimal part is stored in [D1+1], and the binary data is stored in [D2+1, D2].

During the conversion, the data stored within the range from [S] to the element that stores 00H is handled as a character string in the unit of byte.



- String data to be converted

The number of characters in and the value range (decimal point ignored) of the character string to be converted must meet the following requirements:

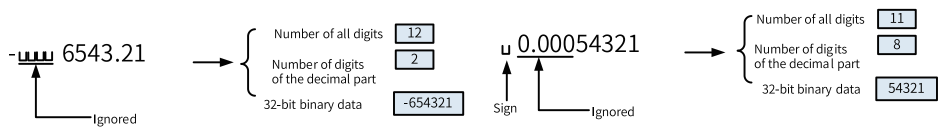
Item	Value Range
Total number of characters	2 to 8
Number of characters of the decimal part	0 to 10
Value range (decimal point ignored)	–2147483648 to +2147483647 For example, 123.45 is processed as 12345.

The types of characters that can be used in the character string to be converted are as follows:

Item	Character Type
Positive number	Space (20H)
Negative number	– (2DH)
Decimal point	. (2EH)
Digit	0 (30H) to 9 (39H)

- [D1] stores the total number of digits, including the digits, sign, and decimal point.
- [D1+1] stores the number of digits of the decimal part, that is, the characters to the right of the decimal point "." (2EH).
- [D2+1, D2] stores the 32-bit binary data converted from a character string with the decimal point ignored.

In the character string, "space" (20H) and "0" (30H) characters between the sign and the first number other than "0" are ignored in the conversion to 32-bit binary data.



Note

- The sign data, spaces (20H) or – (2DH), must be stored in the first byte (low-order 8 bits of the elements starting from [S]).
- Only digits 0 (30H) to 9 (39H), spaces (20H), and decimal points (2EH) can be stored in the ASCII code data area within the range from the second byte of [S] to the string end "00H". An operation error will occur when "–" (2DH) is stored after the second byte.

Errors

An operation error occurs in the following conditions.

- The total number of digits of the character string to be converted is out of the following range.

Operation	Value Range
16-bit operation	2 to 8
32-bit operation	2 to 13

- The number of digits of the decimal part of the character string to be converted is out of the following range.

Operation	Value Range
16-bit operation	0 to 5
32-bit operation	0 to 10

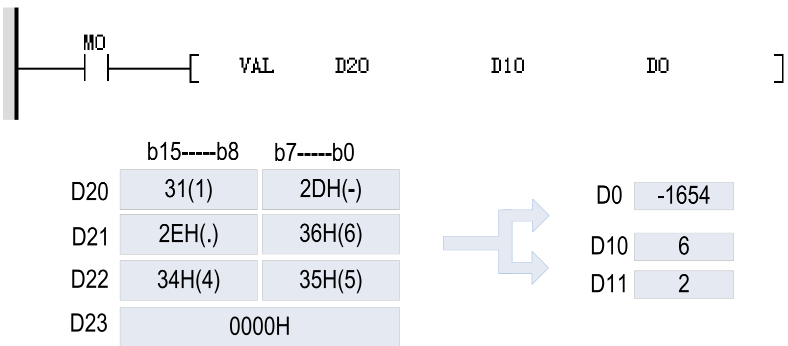
- The relationship between the number of all digits and the number of digits of the decimal part of the character string to be converted (starting from [S]) does not meet the following requirements:
  - Total number of digits – 3 ⩾ Number of digits of the decimal part
  - The sign is set to any ASCII code other than "space" (20H) and "–" (2DH).
  - A digit of a number is set to any ASCII code other than "0" (30H) to "9" (39H) or a decimal point "." (2EH).
  - The character string (starting from [S]) to be converted contains multiple decimal points "." (2EH).
- The binary data after conversion is out of the following range.

Operation	Value Range
16-bit operation	–32768 to +32767
32-bit operation	–2147483648 to +2147483647

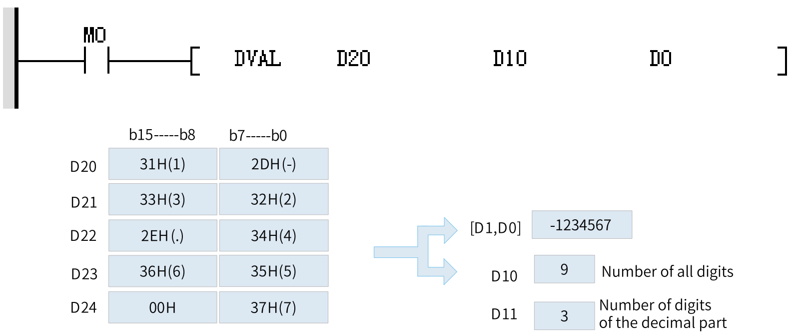
- "00H" does not exist in elements starting from [S].

Instruction Example

- When M0 is ON, the character string data stored in D20 to D22 is regarded as an integer value, converted into a binary value, and stored in D0.



- When M0 is ON, the character string data stored in D20 to D24 is regarded as an integer value, converted into a binary value, and stored in [D1, D0].



3.8.5 ESTR

The ESTR instruction converts binary floating-point data (real number) into a character string (ASCII codes) with specified number of digits.

ESTR – Conversion from binary floating-point into string

16-bit Instruction	-			
32-bit Instruction	DESTR: Continuous execution/DESTRP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Operand	Start number of elements that store the binary floating-point number to be converted	-	REAL
S2	Start number	Start number of elements that store the display format of the value to be converted	-	INT, array*3
D	Result	Start number of elements that store the character string after conversion	-	DINT, array*indeterminate

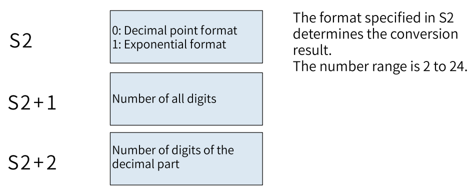
Table 3–157 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

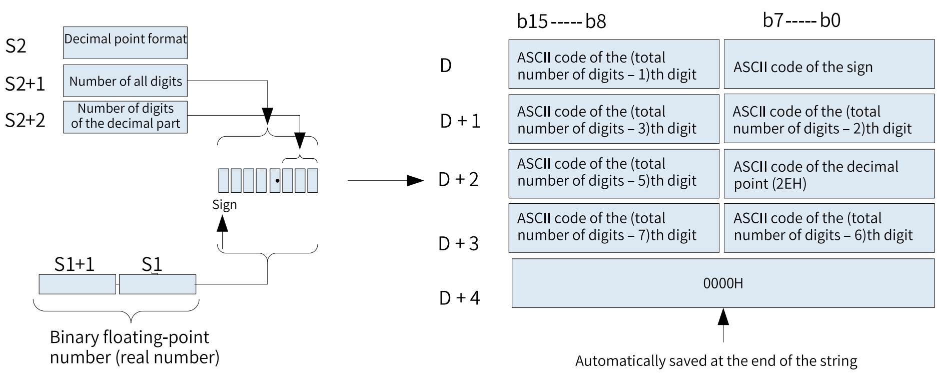
Function and Instruction Description

32-bit operation (DESTR)

The DESTR instruction converts the binary floating-point number in [S1 +1, S] into a character string based on the content of [S2, S2+1, S2+2] and stores the result in elements starting from D.



- Decimal point format

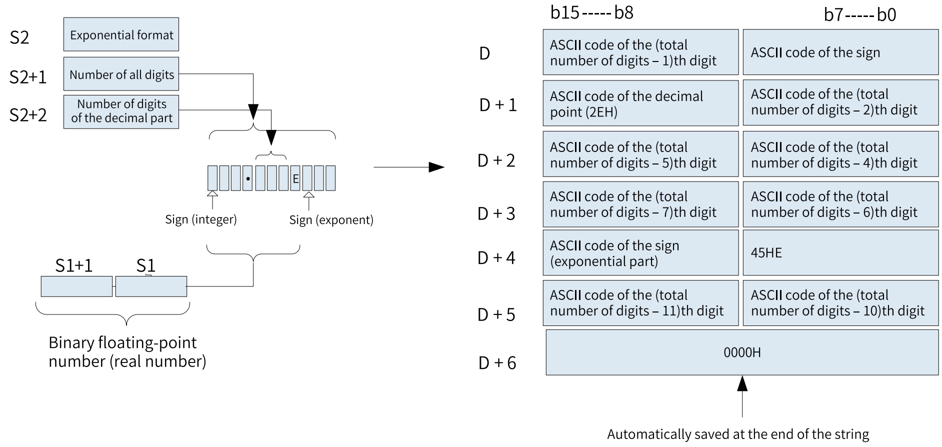


The total number of digits (max. 24 digits) is specified in [S2+1] based on the following rules:

- When the number of digits of the decimal part is 0, the total number of digits is greater than or equal to 2.
- When the number of digits of the decimal part is not 0, the total number of digits is greater than or equal to the number of digits of the decimal part plus 3.

The number of digits of the decimal part specified in [S2+2] ranges from 0 to 7. Meanwhile, it cannot be greater than the total number of digits minus 3.

- Exponent format



The total number of digits (max. 24 digits) is specified in [S2+1] based on the following rules:

- When the number of digits of the decimal part is 0, the total number of digits is greater than or equal to 6.
- When the number of digits of the decimal part is not 0, the total number of digits is greater than or equal to the number of digits of the decimal part plus +7.

The number of digits of the decimal part specified in [S2+2] ranges from 0 to 7. Meanwhile, it cannot be greater than the total number of digits minus 3.

Errors

An operation error occurs in the following conditions. The error flag M8067 turns ON and the error code is stored in D8067.

- The value in [S1] is out of range. (Error code: K6706)
- The value in [S2] is neither 0 nor 1. (Error code: K6706)
- The total number of digits specified in [S2+1] is out of range. (Error code: K6706)

Decimal point format:

When the number of digits of the decimal part is 0, the total number of digits is greater than or equal to 2.

When the number of digits of the decimal part is not 0, the total number of digits is greater than or equal to the number of digits of the decimal part plus 3.

Exponent format:

When the number of digits of the decimal part is 0, the total number of digits is greater than or equal to 6.

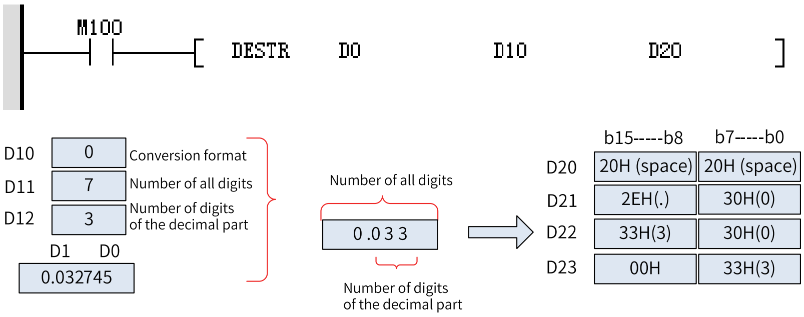
When the number of digits of the decimal part is not 0, the total number of digits is greater than or equal to the number of digits of the decimal part plus +7.

- The number of digits of the decimal part specified in [S2+2] is out of range. (Error code: K6706)
Decimal point format: Number of digits of the decimal part ⩽ Total number of digits – 3

  Exponent format: Number of digits of the decimal part ⩽ Total number of digits – 7
- The elements starting from [D] for storing the character string are out of the corresponding element range. (Error code: K6705)
- The number of digits in the conversion result exceeds the specified total number of digits. (Error code: K6705)

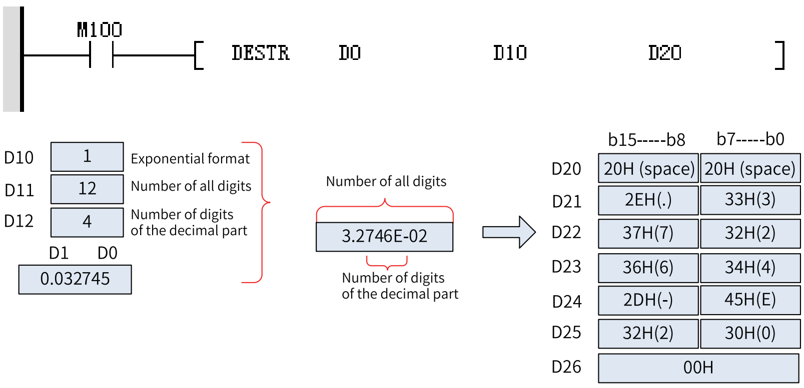
Instruction Example

- When M100 is ON, the binary floating-point number in D0 and D1 is converted based on the content (decimal form) of D10 to D12. The result is stored in elements starting from D20.



Element Name	Data Type	Display Format	Current Value
D0	Float	Dec	0.032745
D10	16-bit INT	Dec	0
D11	16-bit INT	Dec	7
D12	16-bit INT	Dec	3
M100	BOOL	Bin	ON
D20	16-bit INT	Hex	0x2020
D21	16-bit INT	Hex	0x2E30
D22	16-bit INT	Hex	0x3330
D23	16-bit INT	Hex	0x33

- When M100 is ON, the binary floating-point number in D0 and D1 is converted based on the content (exponential form) of D10 to D12. The result is stored in elements starting from D20.



Element Name	Data Type	Display Format	Current Value
D0	Float	Dec	0.032745
D10	16-bit INT	Dec	1
D11	16-bit INT	Dec	12
D12	16-bit INT	Dec	4
M100	BOOL	Bin	ON
D20	16-bit INT	Hex	0x2020
D21	16-bit INT	Hex	0x2E33
D22	16-bit INT	Hex	0x3732
D23	16-bit INT	Hex	0x3534
D24	16-bit INT	Hex	0x2D45
D25	16-bit INT	Hex	0x3230
D26	16-bit INT	Hex	0x0

3.8.6 EVAL

The EVAL instruction converts a character string (ASCII codes) into binary floating-point data.

EVAL – Conversion from string into binary floating-point

16-bit Instruction	-			
32-bit Instruction	DEVAL: Continuous execution/DEVALP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Operand	Start number of elements that store the character string to be converted	-	DINT, array*indeterminate
D	Result	Start number of elements that store the binary floating-point number after conversion	-	REAL

Table 3–158 List of elements

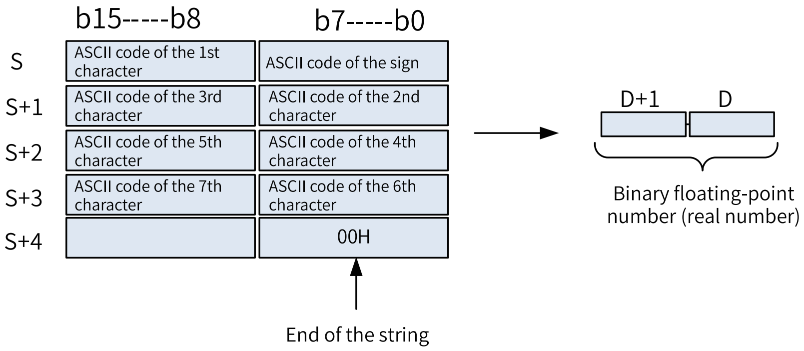
Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

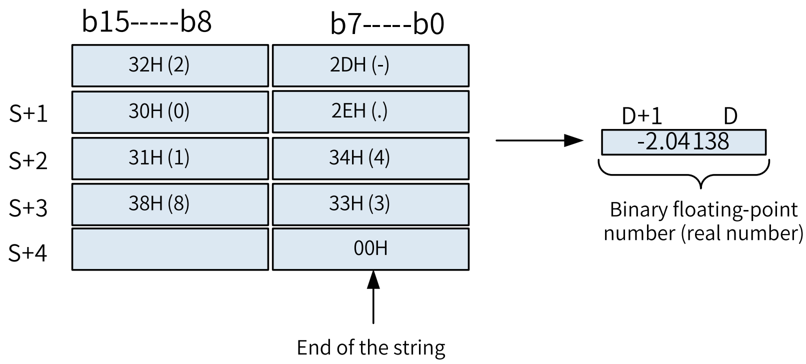
32-bit operation (DEVAL)

The DEVAL instruction converts the character string stored in elements starting from [S] into a binary floating-point number and stores the result in [D+1, D].

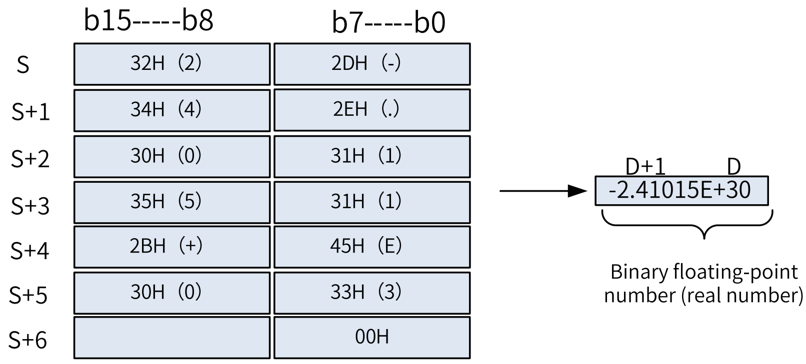
A specified character string may be in the decimal point format or exponent format. A character string in either format can be converted into binary floating-point data.



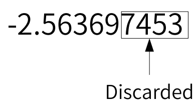
- Decimal point format

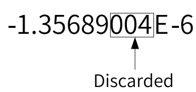


- Exponent format



When a character string to be converted into binary floating-point specified by [S] has 7 digits or more excluding the sign, decimal point, and exponent part, the digits after the 7th digit are discarded.





When "2BH" (+) is specified as the sign in the decimal point format or when the sign is omitted, a character string is converted into a positive value. When "2DH" (–) is specified as the sign, a character string is converted into a negative value.

When "2BH" (+) is specified as the sign in the exponent format or when the sign is omitted, a character string is converted into a positive exponent. When "2DH" (–) is specified as the sign, a character string is converted into a negative exponent.

If the source string specified in [S] contains 20H (space) or 30H (0) between digits other than the first 0, 20H or 30H is ignored when the string is converted.

-□01.245

Ignored

The source string can contain a maximum of 24 characters, including 20H (space) and 30H (0).

Related Elements

Element	Name	Condition	Action
M8020	Zero flag	The conversion result is true 0 (The mantissa part is 0).	The zero flag M8020 turns ON.
M8021	Borrow flag	The absolute value of the conversion result is less than 2^{-126}.	The value in D is 2^{-126} (the minimum value of 32-bit real numbers), and the borrow flag M8021 turns ON.
M8022	Carry flag	The absolute value of the conversion result is greater than or equal to 2^{128}.	The value in D is 2^{128} (the maximum value of 32-bit real numbers), and the carry flag M8022 turns ON.

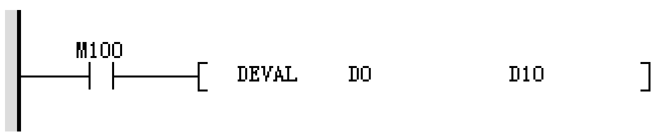
Errors

An operation error occurs in the following conditions.

- The integer or decimal part contains characters other than 30H (0) to 39H (9).
- The character string starting from [S] contains two or more decimal points (2EH).
- The exponent contains characters other than "45H" (E), "2BH" (+), and "2DH" (–), or multiple exponents exist.
- "00H" does not exist in the corresponding element range starting from [S].
- The number of characters after [S] is 0 or more than 24.

Instruction Example

- When M101 is ON, the character string stored in elements starting from D0 is converted into a binary floating-point number (in decimal point format). The result is stored in D10 and D11.



	b15-----b8	b7-----b0
D0	20H (space)	2DH(-)
D1	31H(1)	30H(0)
D2	32H(2)	2EH(.)
D3	34H(4)	33H(3)
D4	32H(2)	35H(5)
D5	00H	31H(1)

⇨

D11 D10
-1.234521

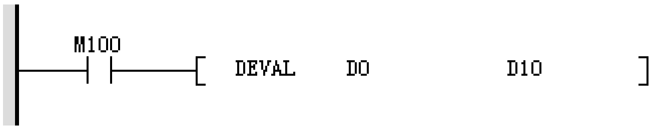
-□01.234521

⇧

Space ignored

Element Name	Data Type	Display Format	Current Value
D10	Float	Dec	-1.234521
M100	BOOL	Bin	ON
D0	16-bit INT	Hex	0x202D
D1	16-bit INT	Hex	0x3130
D2	16-bit INT	Hex	0x322E
D3	16-bit INT	Hex	0x3433
D4	16-bit INT	Hex	0x3235
D5	16-bit INT	Hex	0x31

- When M100 is ON, the character string stored in elements starting from D0 is converted into a binary floating-point number (in exponent format). The result is stored in D0 and D11.



	b15-----b8	b7-----b0
D0	20H (space)	20H (space)
D1	2EH(.)	31H(1)
D2	33H(3)	32H(2)
D3	35H(5)	34H(4)
D4	2DH(-)	45H(E)
D5	32H(2)	30H(0)
D6	00H	00H

⇨

D11 D10
0.012345

□□1.2345E-02

⇧

Space ignored

Element Name	Data Type	Display Format	Current Value
D10	Float	Dec	0.012345
M100	BOOL	Bin	ON
D0	16-bit INT	Hex	0x2020
D1	16-bit INT	Hex	0x2E31
D2	16-bit INT	Hex	0x3332
D3	16-bit INT	Hex	0x3534
D4	16-bit INT	Hex	0x2D45
D5	16-bit INT	Hex	0x3230
D6	16-bit INT	Hex	0x0

3.8.7 $ADD

The $ADD instruction links a character string to another character string.

$ADD – Character string linking

16-bit Instruction	$ADD: Continuous execution/$ADDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	String to be linked	Start number of elements that store the source data (character string) to be linked, or a directly specified character string	-	INT, array*indeterminate
S2	String to be linked to the source string	Start number of elements that store the data (character string) to be linked to the source string, or a directly specified character string	-	INT, array*indeterminate
D	Link result	Start number of elements that store the data (character string) after linking	-	INT, array*indeterminate

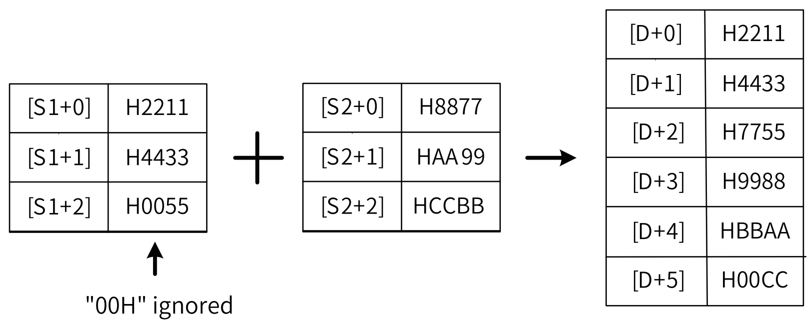
Table 3–159 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The $ADD instruction links the character string in elements starting from [S2] to the end of the character string in elements starting from [S1] and stores the resulting character string in [D].

A character string stored in [S1] or [S2] is organized by byte and ends with the first "00H" byte.



During linking, "00H" indicating the end of a string is ignored and the last character of a character string is linked to the last character of another specified character string. After a character string is linked, "00H" is automatically added at the end.

When the number of characters in the new character string after linking is odd, "00H" is stored in the high-order byte of the element that stores the last character.

When the number of characters in the new character string after linking is even, "0000H" is stored in the element after the one that stores the last character.

Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from[S1] or [S2].
- The number of elements required to store the linking result is beyond the element range starting from [D].

Instruction Example

Program running flag

- Running: ON
- Stopped: OFF

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x55
	16-bit INT	Dec	
D200	16-bit INT	Hex	0x8877
D201	16-bit INT	Hex	0xAA99
D202	16-bit INT	Hex	0xCCBB
D203	16-bit INT	Hex	0xDD
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x2211
R101	16-bit INT	Hex	0x4433
R102	16-bit INT	Hex	0x7755
R103	16-bit INT	Hex	0x9988
R104	16-bit INT	Hex	0xBBAA
R105	16-bit INT	Hex	0xDDCC
R106	16-bit INT	Dec	0

3.8.8 LEN

The LEN instruction detects the number of characters (bytes) of a specified character string.

LEN: Character string length detection

16-bit Instruction	LEN: Continuous execution/LENP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Checked data	Start number of elements that store the character string of which the length is to be detected	-	INT, array*indeterminate
D	Detection result	Number of the element that stores the detected number of characters (bytes) contained in the string	-	INT, array*indeterminate

Table 3–160 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

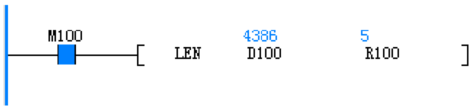
The LEN instruction detects the number of characters in the character string stored in elements starting from [S] and stores the result in [D]. Data starting from [S] to the first element that stores "00H" is handled as a character string in the unit of byte.

An error is returned in the following conditions:

1. "00H" is not found within the corresponding element range starting from [S].

2. The detected number of characters is greater than 32,767.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x55
D103	16-bit INT	Hex	0x0
	16-bit INT	Dec	

3.8.9 INSTR

The INSTR instruction searches a specified character string within another character string.

INSTR – Character string search

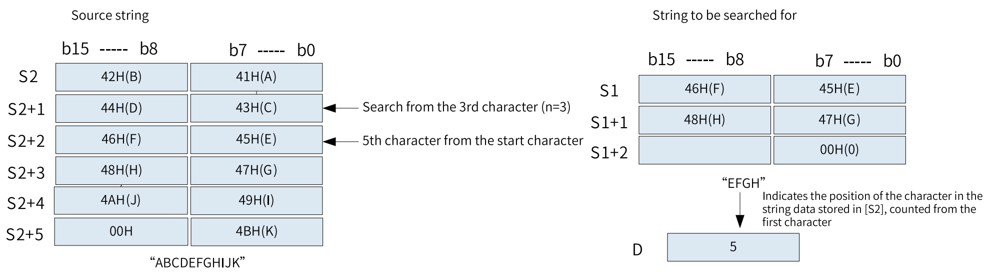
16-bit Instruction	INSTR: Continuous execution/INSTRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Source data	Start number of elements that store the character string to be searched for	-	INT, array*indeterminate
S2	Search source	Start number of elements that store the character string to be searched	-	INT, array*indeterminate
D	Search result	Start number of elements that store the search result	-	INT
n	Search start position	Position from which the search starts	1 to 32767	INT

Table 3–161 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

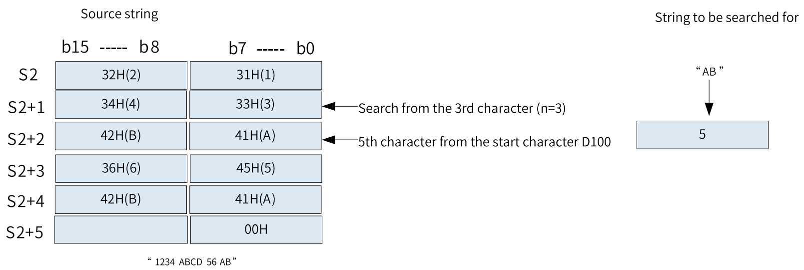
Function and Instruction Description

The INSTR instruction searches for the character string stored in elements starting from [S1] in the source data stored in elements starting from [S2]. The search begins at the nth character from the left end (start of a string) of [S2] and the search result (start position of the searched character string, that is, position of the first matching character located from the left end) is stored in [D].



If no character string in elements starting from [S2] matches the character string in elements starting from [S1], 0 is stored in [D].

If n (search start position) is negative or 0, the instruction is not executed.



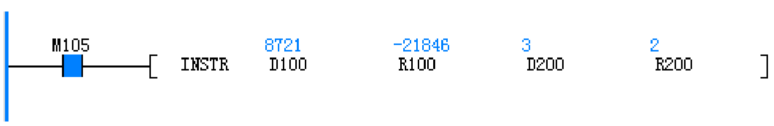
Errors

An error is returned in the following conditions:

1. The value of n (search start position) is greater than the number of characters stored in elements starting from [S2].

2. "00H" is not found within the corresponding element range starting from [S1] or [S2].

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x0
	16-bit INT	Hex	
D200	16-bit INT	Hex	0x3
R200	16-bit INT	Hex	0x2
	16-bit INT	Hex	
R100	16-bit INT	Hex	0xAAAA
R101	16-bit INT	Hex	0x2211
R102	16-bit INT	Hex	0x4433
R103	16-bit INT	Hex	0xAAAA
R104	16-bit INT	Hex	0xAAAA
R105	16-bit INT	Hex	0x0

3.8.10 RIGHT

The RIGHT instruction extracts a specified number of characters from the right end of a specified character string.

RIGHT – String data extraction from the right

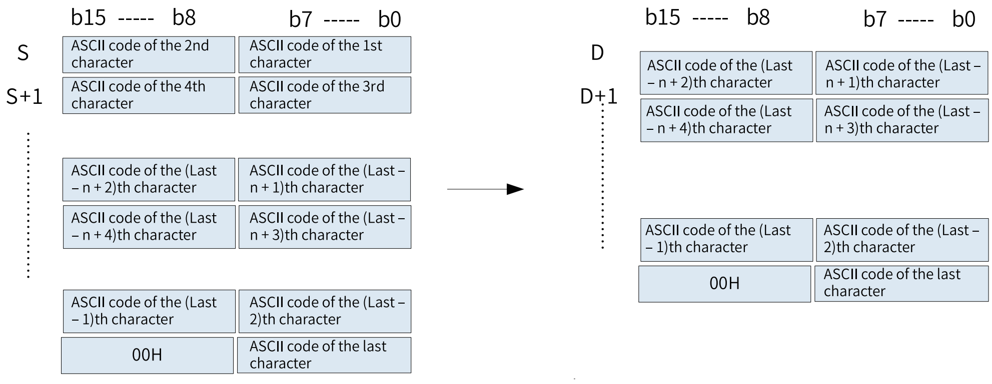
16-bit Instruction	RIGHT: Continuous execution/RIGHTP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store a character string	-	INT, array*indeterminate
D	Extraction result	Start number of elements that store the extracted character string	-	INT, array*indeterminate
n	Extracted character count	Number of characters to be extracted	1 to 32767	INT

Table 3–162 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The RIGHT instruction extracts n characters from the right end (that is, from the end) of the character string stored in elements starting from [S] and stores the extraction result to elements starting from [D].



"00H" is automatically added at the end of the extracted characters.

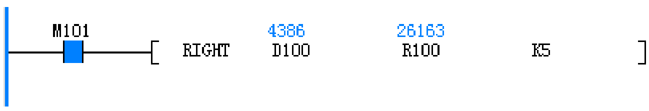
- When the number of extracted characters is odd, "00H" is stored in the high-order byte of the element that stores the last character.
- When the number of extracted characters is even, "0000H" is stored in the element after the one that stores the last character.
- When the number of bytes to be extracted is 0, "0000H" is stored in [D].

Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S].
- The number of elements starting from [D] is smaller than the number of elements required to store the extracted n characters.
- n is greater than the number of characters stored in elements starting from [S].
- n is a negative number.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x0
	16-bit INT	Dec	
R100	16-bit INT	Hex	0x6633
R101	16-bit INT	Hex	0x8855
R102	16-bit INT	Hex	0x77
R103	16-bit INT	Hex	0x0

3.8.11 LEFT

The LEFT instruction extracts a specified number of characters from the left end of a specified character string.

LEFT – String data extraction from the left

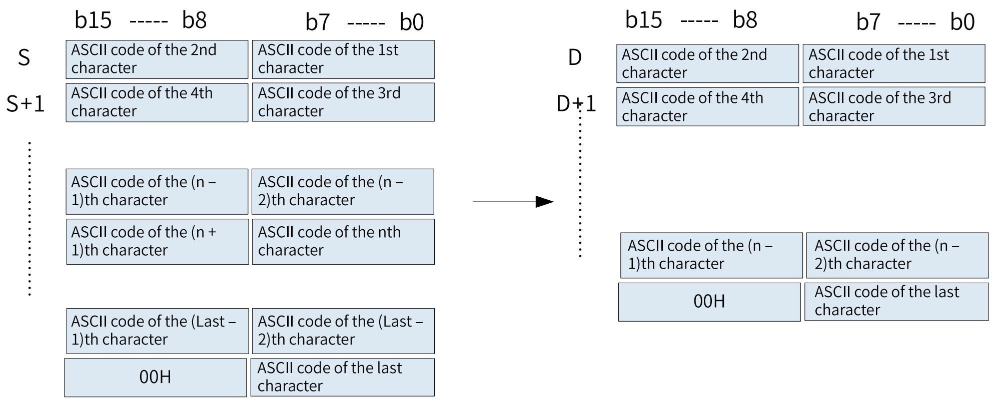
16-bit Instruction	LEFT: Continuous execution/LEFTP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store a character string	-	INT, array*indeterminate
D	Extraction result	Start number of elements that store the extracted character string	-	INT, array*indeterminate
n	Extracted character count	Number of characters to be extracted	1 to 32767	INT

Table 3–163 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
n	-	-	-	√	√	√	√	-	-

Function and Instruction Description

The LEFT instruction extracts n characters from the left end (that is, from the start) of the character string stored in elements starting from [S] and stores the extraction result to elements starting from [D].



"00H" is automatically added at the end of the extracted characters.

- When the number of extracted characters is odd, "00H" is stored in the high-order byte of the element that stores the last character.
- When the number of extracted characters is even, "0000H" is stored in the element after the one that stores the last character.
- When the number of bytes to be extracted is 0, "0000H" is stored in [D].

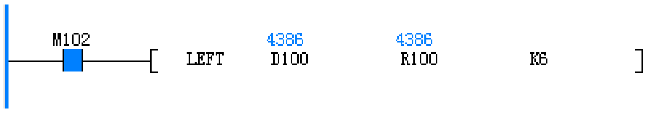
Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S].

- The number of elements starting from [D] is smaller than the number of elements required to store the extracted n characters.
- n is greater than the number of characters stored in elements starting from [S].
- n is a negative number.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x0
	16-bit INT		
R100	16-bit INT	Hex	0x1122
R101	16-bit INT	Hex	0x3344
R102	16-bit INT	Hex	0x5566
R103	16-bit INT	Hex	0x0

3.8.12 MIDW

The MIDW instruction replaces the characters in arbitrary positions of a specified character string with characters in another specified character string.

MIDW – Random replacement of character string

16-bit Instruction	MIDW: Continuous execution/MIDWP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Source data	Start number of elements that store the source character string	-	INT, array*indeterminate
D	Replacement result	Start number of elements that store the character string after replacement	-	INT, array*indeterminate
S2	Replacement position	Start number of elements that specify the start position of replacement and the number of characters to be replaced S2: Position of the first character of the character string to be replaced S2+1: Number of characters to be replaced	-	INT, array*2

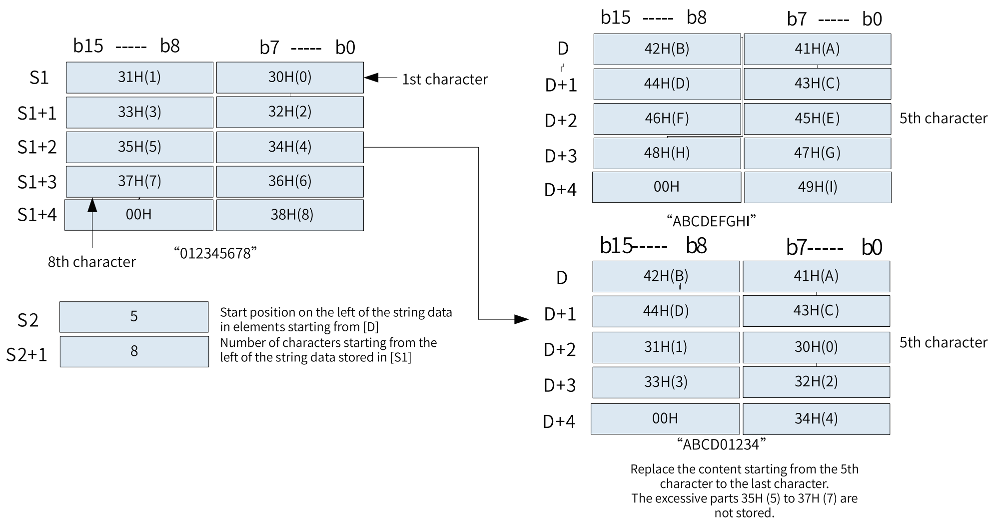
Table 3–164 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The MIDW instruction extracts [S2+1] characters from the left (that is, the start) of the character string stored in elements starting from [S1] and stores the extracted data to the position specified by [S2] of the character string stored in elements starting from [D].

- The character string specified in [S1] indicates data stored in elements starting from [S1] and ending with the element that stores the first "00H".
- When the value in [S2+1] is 0, the instruction is not executed.
- When the value in [S2+1] is –1, the entire character string in elements starting from [S1] is stored to elements starting from [D].
- If the value in [S2+1] exceeds the number of characters starting from the character specified by [S2] in elements starting from [D], data is stored up to the last character in elements starting from [D], and redundant characters of the source string are discarded.

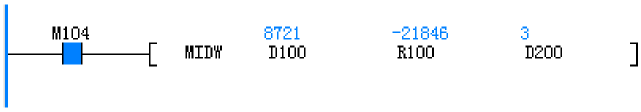


Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S1] or [D].
- The value specified in [S2] is greater than the number of characters of the character string stored in elements starting from [D].
- The value specified in [S2] is negative.
- The value specified in [S2+1] is –2 or less.
- The value specified in [S2+1] exceeds the number of characters stored in elements starting from [S1].

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x6655
D103	16-bit INT	Hex	0x0
D200	16-bit INT	Hex	0x3
D201	16-bit INT	Hex	0x4
D202	16-bit INT	Hex	0x0
R100	16-bit INT	Hex	0xAAAA
R101	16-bit INT	Hex	0x2211
R102	16-bit INT	Hex	0x4433
R103	16-bit INT	Hex	0xAAAA
R104	16-bit INT	Hex	0xAAAA
R105	16-bit INT	Hex	0x0

3.8.13 MIDR

The MIDR instruction extracts a specified number of characters from arbitrary positions of a specified character string.

MIDR – Random extraction of character string

16-bit Instruction	MIDR: Continuous execution/MIDRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Source data	Start number of elements that store a character string	-	INT, array*indeterminate
D	Extraction result	Start number of elements that store the extracted character string	-	INT, array*indeterminate
S2	Extraction position	Start number of elements that specify the start position of characters to be extracted and the number of characters to be extracted S2: Start position of characters to be extracted S2+1: Number of characters to be extracted	-	INT, array*2

Table 3–165 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-

Function and Instruction Description

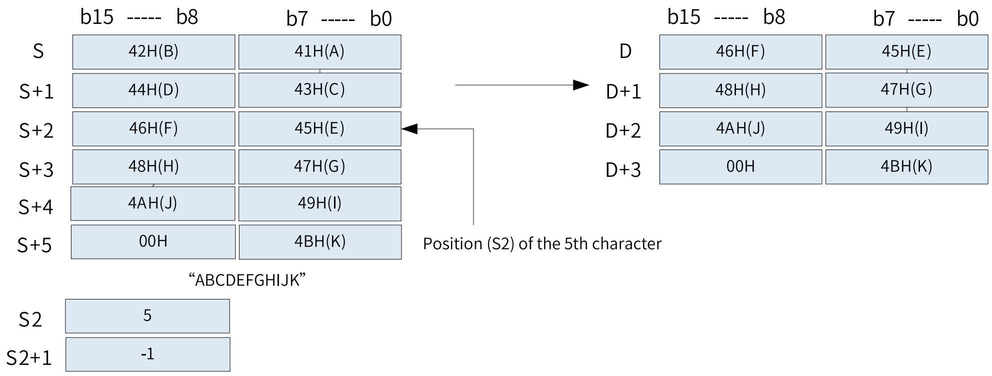
The MIDR instruction extracts [S2+1] characters from the position specified by [S2] (starting from the left of the character string, that is, from the start) of the character string data stored in elements starting from [S1] and stores the extraction result to elements starting from [D].

- When the number of extracted characters specified in [S2+1] is odd, "00H" is stored in the high-order byte of the element that stores the last character.
- When the number of extracted characters specified in [S2+1] is even, "0000H" is stored in the element after the one that stores the last character.

The character string specified in [S1] indicates data stored in elements starting from [S1] and ending with the element that stores the first "00H".

When the value in [S2+1] is 0, the instruction is not executed.

When the value in [S2+1] is –1, all data within the range from the character specified by [S2] to the last character stored in elements starting from [S1] is stored to elements starting from [D].



Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S1].
- The value specified in [S2] is greater than the number of characters of the character string stored in elements starting from [S1].
- The number of elements starting from [D] is smaller than the number of elements required to store the extracted [S2+1] characters.
- The value specified in [S2] is negative.
- The value specified in [S2+1] is –2 or less.
- The value specified in [S2+1] exceeds the number of characters stored in elements starting from [S1].

Instruction Example

Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x1122
D101	16-bit INT	Hex	0x3344
D102	16-bit INT	Hex	0x5566
D103	16-bit INT	Hex	0x7788
D104	16-bit INT	Hex	0x0
D200	16-bit INT	Dec	3
D201	16-bit INT	Dec	4
R100	16-bit INT	Hex	0x3344
R101	16-bit INT	Hex	0x5566
R102	16-bit INT	Hex	0x0

3.8.14 $MOV

The $MOV instruction transfers character string data.

$MOV – Character string transfer

16-bit Instruction	$MOV: Continuous execution/$MOVP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Source address	Character string (a maximum of 32 characters) directly specified in the transfer source, or start number of elements that store the character string	-	INT, array*indeterminate
D	Destination address	Start number of elements that store the transferred character string	-	INT, array*indeterminate

Table 3–166 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

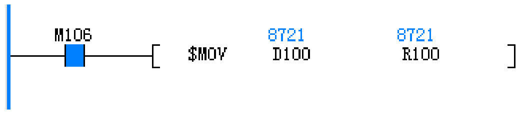
The $MOV instruction copies the character string data in elements starting from [S] to elements starting from [D]. The character string data stored in elements starting from [S] and ending with the element that stores the first "00H" is transferred at a time, together with the terminator "00H" or "0000H".

Errors

An error is returned in the following conditions:

- "00H" is not found within the corresponding element range starting from [S].
- The number of elements starting from [D] is smaller than the number of elements required to store the transferred character string data.

Instruction Example



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Hex	0x2211
D101	16-bit INT	Hex	0x4433
D102	16-bit INT	Hex	0x6655
D103	16-bit INT	Hex	0x0
R100	16-bit INT	Hex	0x2211
R101	16-bit INT	Hex	0x4433
R102	16-bit INT	Hex	0x6655
R103	16-bit INT	Hex	0x0

3.9 Clock Instructions

3.9.1 Instruction List

The following table lists the clock instructions.

Instruction Category	Instruction	Function
Clock instruction	TCMP	Clock data comparison
	TZCP	Clock data zone comparison
	TADD	Clock data addition
	TSUB	Clock data subtraction
	HTOS	Conversion from hour-minute-second into second
	STOH	Conversion from second into hour-minute-second
	TRD	Clock data read
	TWR	Clock data write
	HOUR	Hour meter

3.9.2 TCMP

The TCMP instruction compares the specified time (hour, minute, and second) with the time of an internal real-time clock and outputs the comparison result.

TCMP – Clock data comparison

16-bit Instruction	TCMP: Continuous execution/TCMPP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Hour	Hour of the comparison time, ranging from 0 to 23	-	INT
S2	Minute	Minute of the comparison time, ranging from 0 to 59	-	INT
S3	Second	Second of the comparison time, ranging from 0 to 59	-	INT

S	PLC time data start address	Start address of time registers that store the current time value of a real-time clock, which is usually the data read by the TRD or MOV instruction	-	INT, array*3
D	Comparison result	Start address of three consecutive variable units that store the comparison result	-	BOOL, array*3

Table 3–167 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S3	-	-	-	√	√	-	√	-	-
S	-	-	-	√	√	-	-	-	-
D	√[1]	√	√	-	-	-	-	-	-

Note

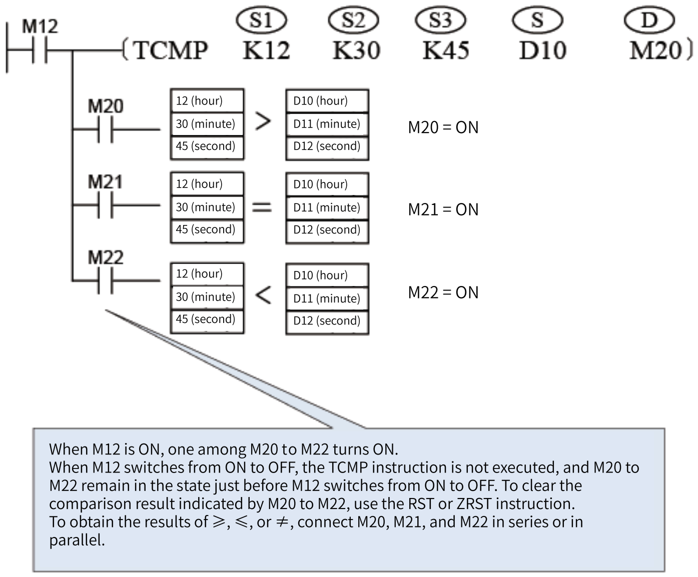
[1] The X element is not supported.

Function and Instruction Description

The TCMP instruction compares the specified time (hour, minute, and second) with the time of an internal real-time clock and outputs the comparison result. Where,

- S1 is the hour of the comparison time, which ranges from 0 to 23.
- S2 is the minute of the comparison time, which ranges from 0 to 59.
- S3 is the second of the comparison time, which ranges from 0 to 59.
- S is the start address of time registers that store the current time value of a real-time clock, which is usually the data read by the TRD or MOV instruction.
- D is the start address of three consecutive variable units that store the comparison result.

Instruction Example



3.9.3 TZCP

比较结果的存放变量启始地址，占用后续共3个变量单元。

TZCP – Clock data zone comparison

16-bit Instruction	TZCP: Continuous execution/TZCPP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Lower limit	Lower limit (hour, minute, and second) of the comparison time zone, which occupies three consecutive variable units	-	INT, array*3
S2	Upper limit	Upper limit (hour, minute, and second) of the comparison time zone, which occupies three consecutive variable units	-	INT, array*3
S	PLC time data head address	Start address of time registers that store the current time value of a real-time clock, which is usually the data read by the TRD or MOV instruction	-	INT, array*3
D	Comparison result	Start address of three consecutive variable units that store the comparison result	-	BOOL, array*3

Table 3–168 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	√	-	-
S2	-	-	-	√	√	-	√	-	-
S	-	-	-	√	√	-	-	-	-
D	√[1]	√	√	-	-	-	-	-	-

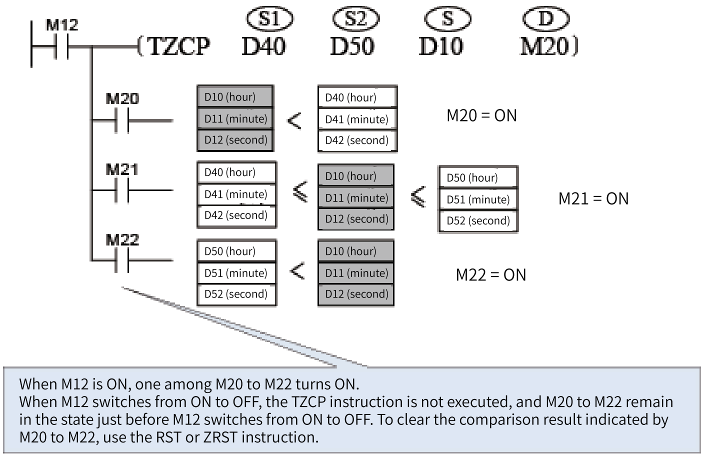
Note

[1] The X element is not supported.

Function and Instruction Description

The TZCP instruction compares the time data of the internal real-time clock with two specified sets of preset values, including hour (0 to 23), minute (0 to 59), and second (0 to 59), and outputs the comparison result.

Instruction Example



3.9.4 TADD

The TADD instruction adds two time values (hour, minute, and second) together and stores the result in the specified variables.

TADD – Clock data addition

16-bit Instruction	TADD: Continuous execution/TADDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Time augend	Time augend, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3
S2	Time addend	Time addend, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3
D	Sum of two time values	Sum of two time values, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3

Table 3–169 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

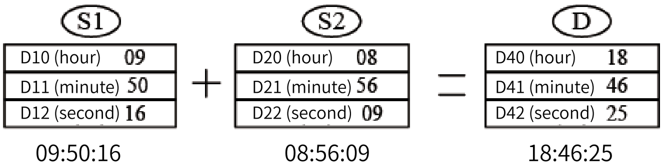
The TADD instruction adds two time values (hour, minute, and second) together and stores the result in the specified variables. Where,

If the operation result exceeds 24 hours, the carry flag M8022 turns ON, and the value simply acquired by addition subtracted by 24 hours is stored as the operation result.

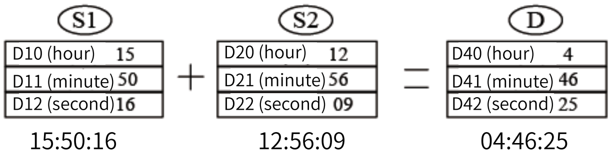
If the operation result is 00:00:00, the zero flag M8020 turns ON.

Instruction Example

The operation is performed as follows:



If the addition result is greater than 24 hours, the carry flag M8022 turns ON.



3.9.5 TSUB

The TSUB instruction subtracts one time value (hour, minute, and second) from another and stores the result in the specified variables.

TSUB – Clock data subtraction

16-bit Instruction	TSUB: Continuous execution/TSUBP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Time subtrahend	Time subtrahend, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3

S2	Time minuend	Time minuend, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3
D	Time difference	Difference between two time values, which occupies three consecutive variable units to store the hour, minute, and second data respectively	-	INT, array*3

Table 3–170 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
S2	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

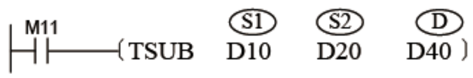
Function and Instruction Description

The TSUB instruction subtracts one time value (hour, minute, and second) from another and stores the result in the specified variables.

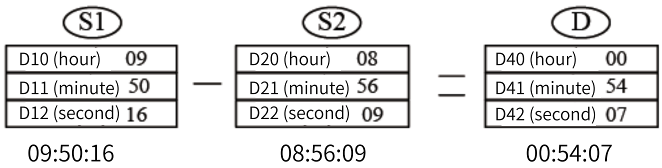
If the operation result is a negative value, the borrow flag M8021 turns ON, and the value simply acquired by subtraction added by 24 hours is stored as the operation result.

If the subtraction result is 00:00:00, the zero flag M8020 turns ON.

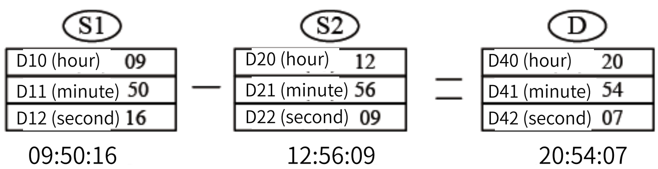
Instruction Example



The operation is performed as follows:



If the subtraction result is a negative value, the borrow flag M8021 turns ON.



3.9.6 HTOS

The HTOS instruction converts the time data in the unit of hour-minute-second into data in the unit of second.

HTOS – Conversion from hour-minute-second into second

16-bit Instruction	HTOS: Continuous execution/HTOSP: Pulse execution			
32-bit Instruction	DHTOS: Continuous execution/DHTOSP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Start number of elements that store the time data (in the unit of hour-minute-second) to be converted	-	INT, array*3 (3 registers)
D	Result	Number of the element that stores the time data (in the unit of second) after conversion	-	INT/DINT

Table 3–171 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction
  The HTOS instruction converts the time data in the unit of hour-minute-second stored in [S, S+1, S+2] into data in the unit of second and stores the result in D.
  - The hour value ranges from 0 to 9.
  - The minute value ranges from 0 to 59.
  - The second value ranges from 0 to 59.
- 32-bit instruction
  The HTOS instruction converts the time data in the unit of hour-minute-second stored in [S, S+1, S+2] into data in the unit of second and stores the result in [D, D+1].
  - The hour value ranges from 0 to 32767.
  - The minute value ranges from 0 to 59.
  - The second value ranges from 0 to 59.

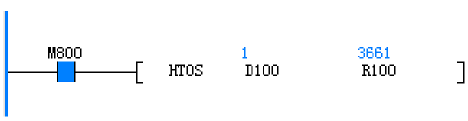
Errors

An error is returned and the instruction is not executed in the following conditions:

- The operands of the 16-bit or 32-bit instruction are out of range.
- The conversion result obtained by the 16-bit instruction is greater than 32,767.
- The time data in [S, S+1, S+2] is out of range.

Instruction Example

The time data in the unit of hour-minute-second stored in D100, D101, and D102 is converted into data in the unit of second. The conversion result is stored in R100.



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Dec	1
D101	16-bit INT	Dec	1
D102	16-bit INT	Dec	1
	16-bit INT	Dec	
R100	16-bit INT	Dec	3661
	16-bit INT	Dec	

3.9.7 STOH

The STOH instruction converts the time data in the unit of second into data in the unit of hour-minute-second.

STOH – Conversion from second into hour-minute-second

16-bit Instruction	STOH: Continuous execution/STOHP: Pulse execution			
32-bit Instruction	DSTOH: Continuous execution/DSTOHP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source data	Number of the element that stores time data (in the unit of second) to be converted	-	INT/DINT
D	Result	Start number of elements that store the time data (in the unit of hour-minute-second) after conversion	-	INT Array*3 (3 registers)

Table 3–172 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

- 16-bit instruction

  The STOH instruction converts the time data in the unit of second stored in [S] into data in the unit of hour-minute-second and stores the result in [D, D+1, D+2].

  The value in [S] ranges from 0 to 32,767.

- 32-bit instruction

  The STOH instruction converts the time data in the unit of second stored in [S, S+1] into data in the unit of hour-minute-second and stores the result in [D, D+1, D+2].

  The value in [S, S+1] ranges from 0 to 117,964,799.

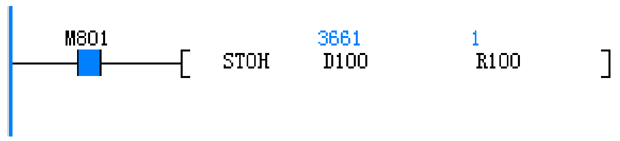
Errors

An error is returned and the instruction is not executed in the following conditions:

- The operands of the 16-bit or 32-bit instruction are out of range.
- The time data (in the unit of second) to be converted in the 16-bit or 32-bit instruction is out of range.

Instruction Example

The time data in the unit of second stored in D100 is converted into data in the unit of hour-minute-second. The conversion result is stored in R100, R101, and R102.



Element Name	Data Type	Display Format	Current Value
D100	16-bit INT	Dec	3661
	16-bit INT	Dec	
R100	16-bit INT	Dec	1
R101	16-bit INT	Dec	1
R102	16-bit INT	Dec	1
	16-bit INT	Dec	

3.9.8 TRD

The TRD instruction reads the data (year, month, day, hour, minute, second, day of week, and millisecond) of the internal real-time clock of the PLC and stores the read data in specified registers.

TRD – Clock data read

16-bit Instruction	TRD: Continuous execution/TRDP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
D	Time storage start address	Start address of eight consecutive variable units that store the year, month, day, hour, minute, second, day of week, and millisecond in turn (from low to high)	-	INT, array*8

Table 3–173 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The TRD instruction reads the data (year, month, day, hour, minute, second, day of week, and millisecond) of the internal real-time clock of the PLC and stores the read data in specified registers.

The instruction of the pulse execution type (TRDP) is recommended.

D is the start address of the eight consecutive variable units that store the year, month, day, hour, minute, second, day of week, and millisecond data in turn (from low address to high address).

Instruction Example

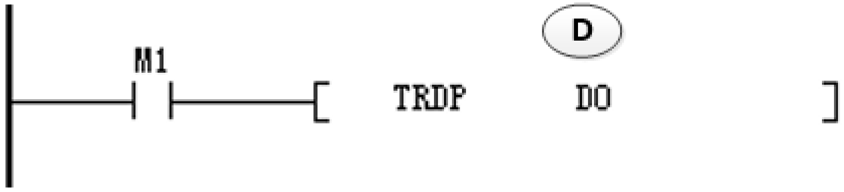


Table 3–174 Conversion process

Item	→	Target Value
Year (2000 to 2038)	→	D0
Month (1 to 12)	→	D1
Day (1 to 31)	→	D2
Hour (0 to 23)	→	D3
Minute (0 to 59)	→	D4
Second (0 to 59)	→	D5
Day (0 to 6: Sunday to Saturday)	→	D6
Millisecond (0 to 999)	→	D7

Note

Generally, to use the clock of the PLC, you need to run the TDR instruction to read the clock data into the D registers.

3.9.9 TWR

The TWR instruction writes the clock data (year, month, day, hour, minute, second, and day of week) specified in S to the internal real-time clock of the PLC.

TWR – Clock data write

16-bit Instruction	TWR: Continuous execution/TWRP: Pulse execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S	Start address for time data to be written	Start address of seven consecutive variable units that store the year, month, day, hour, minute, second, and day of week data to be written in turn (from low to high)	-	INT, array*7

Table 3–175 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-

Function and Instruction Description

The TWR instruction writes the clock data (year, month, day, hour, minute, second, and day of week) specified in S to the internal real-time clock of the PLC.

The instruction of the pulse execution type (TWRP) is recommended.

S is the start address of the seven consecutive variable units that store the year, month, day, hour, minute, second, and day of week data to be written in turn (from low address to high address).

Instruction Example

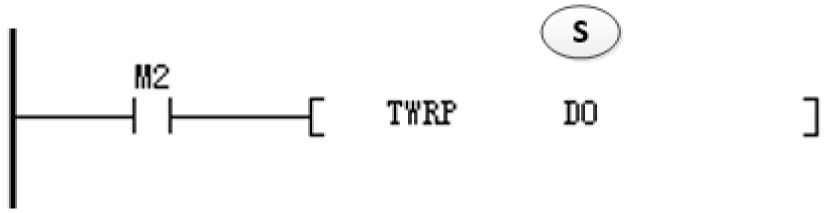


Table 3–176 Conversion process

Data source	→	Item
D0	→	Year (2000 to 2038)
D1	→	Month (1 to 12)
D2	→	Day (1 to 31)
D3	→	Hour (0 to 23)
D4	→	Minute (0 to 59)
D5	→	Second (0 to 59)
D6	→	Day (0 to 6: Sunday to Saturday)

Note

All of the seven data entries will be written to the clock. Therefore, each of the seven variables needs to be specified. For example, if the week is not set, the default value 0 is used, which indicates Sunday. If the month is not set, the default value 0 is considered an error by the PLC which makes the modification on the clock data invalid.

3.9.10 HOUR

When the driving conditions are met, the HOUR instruction records time cumulatively. When the cumulative time reaches the preset value, a specified output becomes active.

HOUR – Hour meter

16-bit Instruction	HOUR: Continuous execution			
32-bit Instruction	DHOUR: Continuous execution			
Operand	Name	Description	Range	Data Type
S	Preset time	Preset time in the unit of hour. When the cumulative time reaches the preset time, the specified output becomes active.	-	INT/DINT
D1	Cumulative time storage start unit	Start number of units that store the cumulative time	-	INT/DINT, array*2
D2	Time reach flag	Variable unit that outputs a time reach alarm. When the cumulative time reaches the preset value, this unit becomes active.	-	BOOL

Table 3–177 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	√	√	-	-	-	-
D1	-	-	-	√	√	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

When the driving conditions are met, the HOUR instruction records time cumulatively. When the cumulative time reaches the preset value, a specified output becomes active. Where,

- S is the preset time in the unit of hour. When the cumulative time reaches the preset value, the specified output becomes active.
- D1 is the start number of units that store the cumulative time.
- D2 is the variable unit that outputs a time reach alarm. When the cumulative time reaches the preset value, this unit becomes active.
- In 16-bit operation, the value in D1 ranges from K0 to K32,767, in the unit of hour. D1+1 stores the current time value less than 1 hour. The value ranges from K0 to K3599, in the unit of second. D1 occupies two units.
- In 32-bit operation, the value stored in D1+1 and D1 ranges from K0 to K2,147,483,647, in the unit of hour. D1+3 and D1+2 store the current time value less than 1 hour. The value ranges from K0 to K3599, in the unit of second. D1 occupies four units.

The time value in D1 cannot be negative. If D1 is specified as a register that is not retentive upon power failure, the value in D1 is cleared when the PLC mode switches from STOP to RUN or when a power failure occurs. If you need to retain the current data in the case of a power failure, specify D1 as a register that is retentive upon power failure.

Instruction Example

When M200 is ON, the time during which M200 remains ON is recorded cumulatively and stored in D300. If the time value is less than 1 hour, the equivalent value in the unit of second is recorded in D301. When the cumulative time in D300 reaches 2000 hours, Y10 turns ON.

After the cumulative time counted from when the timing condition is met reaches the preset value in S, the cumulative time value continues to increase. Timing stops when the current time value in D300 reaches 32,767 hours or the value in D301 reaches 3599s. To restart timing, clear the values in D300 and D301.

3.10 MC Axis Control Instructions (EtherCAT&Pulse Output)

3.10.1 Basic Instructions

3.10.1.1 Instruction List

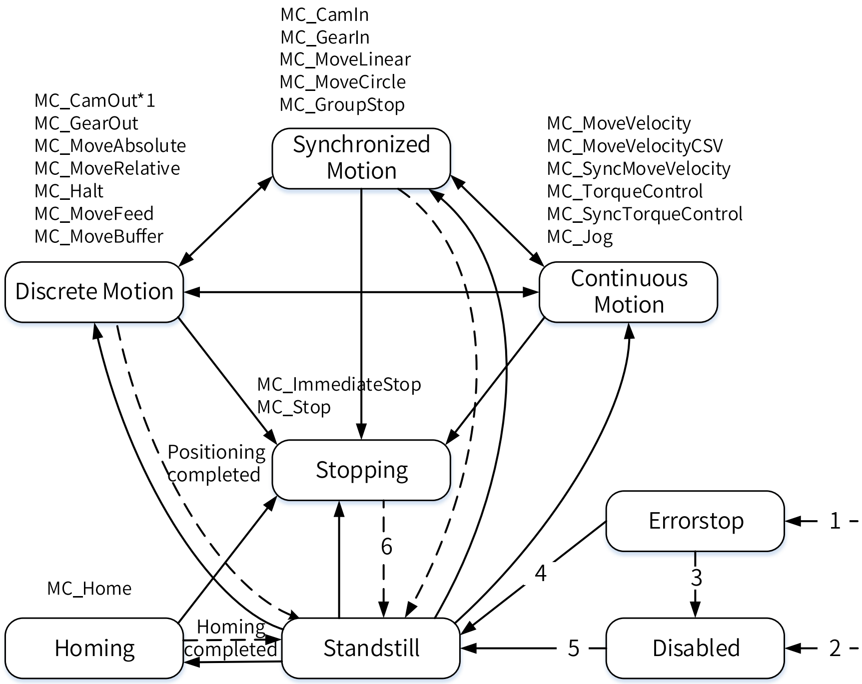
The following table lists the motion control axis instructions.

Table 3–178 Motion control axis instructions

Instruction	Name
MC_Power	Enable control
MC_Reset	Fault reset
MC_ReadStatus	Axis state read
MC_ReadAxisError	Axis error read
MC_ReadDigitalInput	Digital input read
MC_ReadActualPosition	Current position read
MC_ReadActualVelocity	Current velocity read
MC_ReadActualTorque	Current torque read
MC_SetPosition	Current position setting
MC_TouchProbe	Probe
MC_MoveRelative	Relative positioning
MC_MoveAbsolute	Absolute positioning
MC_MoveVelocity	Speed reference
MC_Jog	Jogging
MC_TorqueControl	Torque control
MC_Home	Homing
MC_Stop	Stop
MC_Halt	Halt
MC_ImmediateStop	Immediate stop
MC_MoveFeed	Interrupt positioning
MC_MoveBuffer	Multi-position positioning
MC_MoveSuperImposed	Motion superimposition
MC_MoveVelocityCSV	CSV-based velocity control with adjustable pulse width
MC_SyncMoveVelocity	CSV-based synchronous velocity control with adjustable pulse width
MC_FollowVelocity	CSP-based synchronous velocity control
MC_SyncTorqueControl	Synchronous torque control
MC_SetAxisConfigPara	Axis parameter configuration

3.10.1.2 MC Axis State Machine

The MC axis state machine manages the states and motions of axes based on the PLCOpen state machine.



The state machine is described as follows:

Table 3–179 State definitions

Status Value	Status	Function Description
0	Disabled	Disabled
1	ErrorStop	Fault reaction
2	Stopping	Stopping
3	StandStill	Enabled
4	DiscreteMotion	Discrete motion
5	ContinuousMotion	Continuous motion
7	Homing	Homing
8	SynchronizedMotion	Synchronized motion

Table 3–180 State transition conditions

Transition	Transition Condition
1	The fault detection logic of the axis has detected a fault.
2	The axis has no fault and MC_Power.Enable is OFF.
3	MC_Reset is executed to reset the axis fault and MC_Power.Status is OFF.
4	MC_Reset is executed to reset the axis fault and MC_Power.Status is ON.
5	Both MC_Power.Enable and MC_Power.Status are ON.
6	MC_Stop(MC_ImmediateStop).Done is ON and MC_Stop(MC_ImmediateStop).Execute is OFF.

Precision of Instruction Parameters

The floating-point numbers such as the target position and target velocity in the instructions are single-precision floating-point data. Therefore, the values in the instructions must meet the requirements of the range and precision of single-precision floating-point data when being processed in the PLC program. That is, a value should fall between –3.4E38 and +3.4E38, with a maximum of 7 significant digits. If a value has more than 7 significant digits, the excess part will be automatically rounded.

3.10.1.3 MC_Power

MC_Power – Enable control

Graphic Block

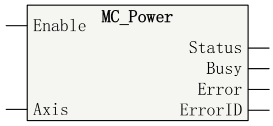
Instruction	Name	LD Expression	LiteST Expression
MC_Power	Enable control		MC_Power(Enable := ???, Axis := ???, Status => , Busy => , Error => , ErrorID =>);

Table 3–181 Instruction format

16-bit Instruction	MC_Power: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Status	Axis enable flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Fault code*1	Yes	0	-	INT

Note

*1: See *“3.10.1.30 Axis Fault Codes” on page 404*.

Table 3–182 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	
D2	√[1]	√	√	-	-	√	-	-	
D3	√[1]	√	√	-	-	√	-	-	
D4	-	-	-	√	√	√	-	-	

Note

[1] The X element is not supported.

Function and Instruction Description

Applicable to the EtherCAT bus axis and the local axis, the MC_Power instruction is used to set the enable state of the axes and is active high.

- Specifying axis
  The axis specified by Axis is latched at the rising edge of Enable.

  If you access an axis by using the axis number and modify Axis when Enable is ON, an instruction error occurs and the previously controlled axis is disabled.

  When you access an axis by using the axis number and modify Axis when Enable is OFF, if the axis specified by Axis is enabled and the instruction is the last Power instruction executed in a PLC scan cycle, the axis specified by Axis will be disabled and become inoperable.
- Function description
  When Enable is set to ON, the axis is enabled, and the Status signal of the instruction is active.

  The PLCOpen state machine of the axis transitions from the Disabled state to the StandStill state.

  After the axis is enabled, it can execute motion instructions such as MC_MoveRelative.

  When Enable is set to OFF, the enable state of the axis is cleared and the execution of motion instructions (such as MC_MoveAbsolute) is interrupted. You cannot control the axis because it does not acknowledge motion instructions. However, you can still execute non-motion instructions such as MC_Power, MC_Reset, and MC_SetPosition.

  When the axis enters the ErrorStop state upon a fault, re-enabling MC_Power cannot switch the axis to the StandStill state. You must call the MC_Reset instruction to reset the axis fault first.
- Multi-execution
  When multiple MC_Power instructions are executed, the control flow of the last MC_Power instruction executed in a cycle shall prevail.

Errors

An error is returned in the following conditions: The axis number does not exist.

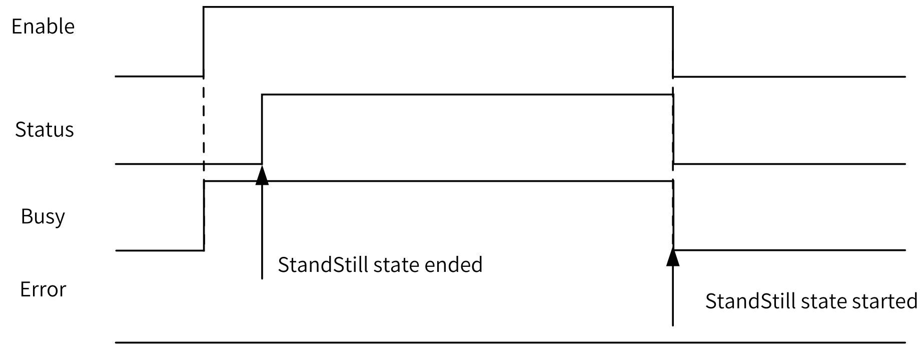
The axis type is incorrect.

Axis initialization fails.

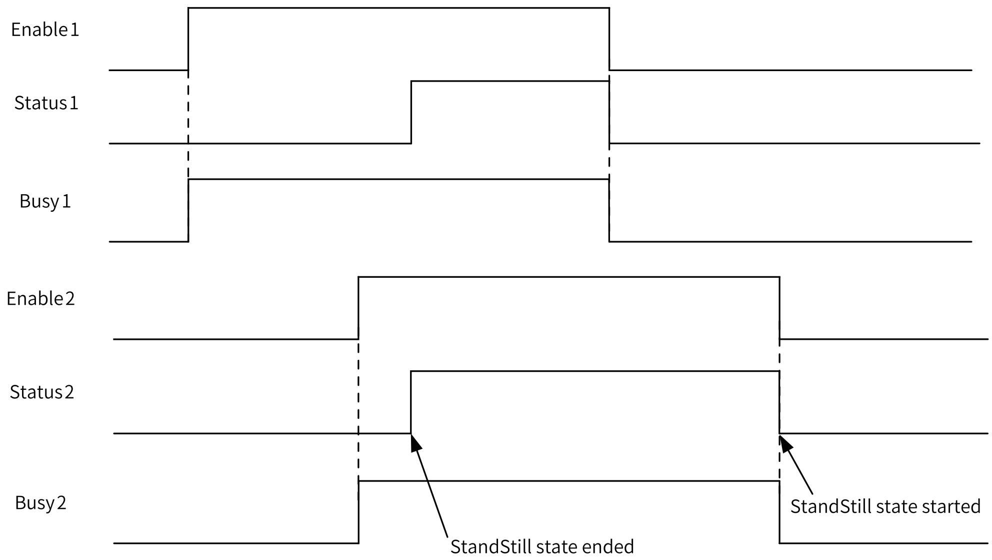
The control word, status word, target position, and current position are not configured in the PDO of the axis.

Timing Diagram

- When an MC_Power instruction is executed to enable an axis properly



- When two MC_Power instructions are executed to enable an axis properly



3.10.1.4 MC_Reset

MC_Reset – Fault reset

Graphic Block

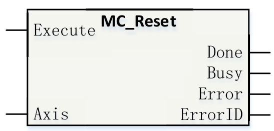
Instruction	Name	LD Expression	LiteST Expression
MC_Reset	Reset fault		MC_Reset(Execute := ???, Axis := ???, Done => , Busy => , Error => , ErrorID =>);

Table 3–183 Instruction format

16-Bit Instruction	MC_Reset: Continuous execution					
32-Bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
D1	Done	Reset completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Fault code*1	Yes	0	-	INT

Note

*1: See *"3.10.1.30 Axis Fault Codes" on page 404*.

Table 3–184 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√ [1]	√	√	-	-	√	-	-	
D2	√[1]	√	√	-	-	√	-	-	
D3	√ [1]	√	√	-	-	√	-	-	
D4	-	-	-	√	√	√	-	-	

Note

[1] The X element is not supported.

Function and Instruction Description

Applicable to the EtherCAT bus axis and the local axis, the MC_Reset instruction is used to reset faults of the axes and is triggered on the rising edge.

The MC_Reset instruction attempts to reset the fault of the axis on the rising edge of the Execute signal. If the reset is successful, the Done output is active; otherwise, the Error signal is active, and ErrorID specifies the reason for the reset failure.

After the reset is successful, the PLCOpen state machine of the axis transitions into the StandStill state if the drive is enabled or the Disabled state if the drive is not enabled.

Abortion

This instruction has no abortion output signal and cannot be aborted during execution.

If there are two reset instructions in one scan cycle, the program will start to execute the reset logic as long as one reset instruction is active. If the reset is successful, the Done signal output of the triggered instruction becomes active.

Errors

An error is returned in the following conditions: The axis number does not exist.

The axis type is incorrect.

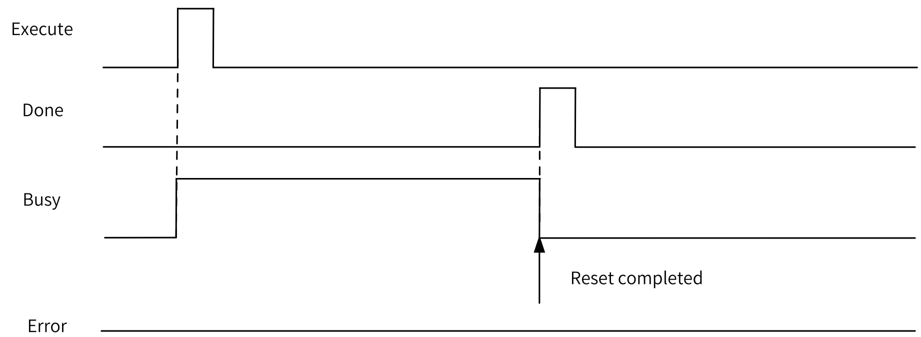
Axis initialization fails.

This instruction is executed when the axis has no fault.

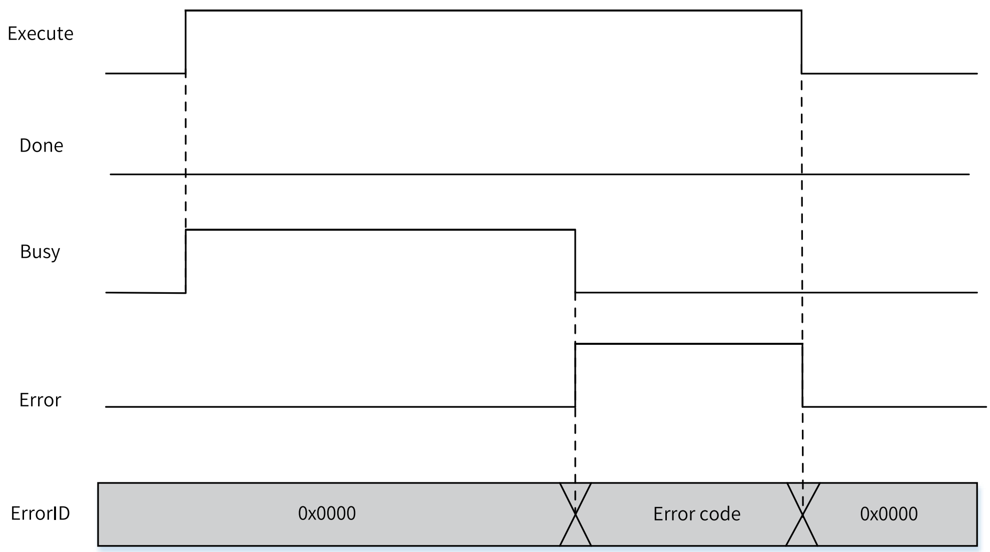
The axis cannot be reset.

Timing Diagram

- A fault occurs on the axis, and the MC_Reset instruction is executed to reset the axis fault successfully.



- A non-resettable fault occurs on the drive.



3.10.1.5 MC_ReadStatus

MC_ReadStatus – Axis state read

Graphic Block

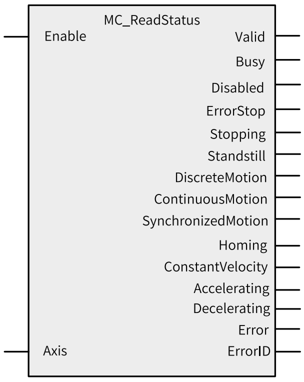
Instruction	Name	LD Expression	LiteST Expression
MC_ReadStatus	Read the Axis state		MC_ReadStatus(Enable := ???, Axis := ???, Valid => , Busy => , Disabled => , ErrorStop => , Stopping => , Standstill => , DiscreteMotion => , ContinuousMotion => , SynchronizedMotion => , Homing => , ConstantVelocity => , Accelerating => , Decelerating => , Error => , ErrorID =>);

Table 3–185 Instruction format

16-bit Instruction	MC_ReadStatus: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Disabled	PLCOpen state machine, disabled	Yes	OFF	ON/OFF	BOOL
D4	ErrorStop	PLCOpen state machine, stopping upon a fault	Yes	OFF	ON/OFF	BOOL
D5	Stopping	PLCOpen state machine, stopping	Yes	OFF	ON/OFF	BOOL
D6	StandStill	PLCOpen state machine, enabled and not running	Yes	OFF	ON/OFF	BOOL
D7	DiscreteMotion	PLCOpen state machine, discreate motion mode	Yes	OFF	ON/OFF	BOOL

D8	ContinuousMo-tion	PLCOpen state machine, continuous motion mode	Yes	OFF	ON/OFF	BOOL
D9	Synchronized-Motion	PLCOpen state machine, synchronized motion mode	Yes	OFF	ON/OFF	BOOL
D10	Homing	PLCOpen state machine, homing	Yes	OFF	ON/OFF	BOOL
D11	ConstantVelocity	The axis velocity is 0. The axis is moving at a constant speed. Invalid in torque mode	Yes	OFF	ON/OFF	BOOL
D12	Accelerating	The axis is accelerating (the absolute value of the velocity is increasing). Invalid in torque mode	Yes	OFF	ON/OFF	BOOL
D13	Decelerating	The axis is decelerating (the absolute value of the velocity is decreasing). Invalid in torque mode	Yes	OFF	ON/OFF	BOOL
D14	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D15	ErrorID	Fault Code	Yes	0	*1	INT

Note

*1: See "3.10.1.30 Axis Fault Codes" on page 404.

Table 3–186 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	√[1]	√	√	-	-	√	-	-	-
D6	√[1]	√	√	-	-	√	-	-	-
D7	√[1]	√	√	-	-	√	-	-	-
D8	√[1]	√	√	-	-	√	-	-	-
D9	√[1]	√	√	-	-	√	-	-	-
D10	√[1]	√	√	-	-	√	-	-	-
D11	√[1]	√	√	-	-	√	-	-	-
D12	√[1]	√	√	-	-	√	-	-	-
D13	√[1]	√	√	-	-	√	-	-	-
D14	√[1]	√	√	-	-	√	-	-	
D15	-	-	-	√	√	√	-	-	

Note

[1] The X element is not supported.

Function and Instruction Description

When Enable is ON, this instruction reads the acceleration/deceleration state and state of the PLCOpen state machine of the axis.

In torque mode, ConstantVelocity, Acceleration, and Deceleration are always OFF.

The priority of a EtherCAT task is higher than that of a PLC master task. If the state of the axis exists only for one EtherCAT cycle in the EtherCAT task, the state cannot be obtained in the PLC master task.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

Axis initialization fails.

The axis type is incorrect.

Timing Diagram

Omitted.

3.10.1.6 MC_ReadAxisError

MC_ReadAxisError – Read axis errors

Graphic Block

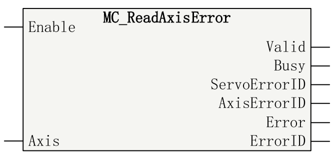
Instruction	Name	LD Expression	LiteST Expression
MC_ReadAxisError	Read axis errors.		MC_ReadAxisError(Enable := ???, Axis := ???, Valid => , Busy => , ServoErrorID => , AxisErrorID => , Error => , ErrorID =>);

Table 3–187 Instruction format

16-bit Instruction	MC_ReadAxisError: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ServoErrorID	If 0x603F is configured in the PDO, the value of 0x603F of the EtherCAT bus driver displayed; otherwise, 0 is displayed.	Yes	0	*2	INT
D4	AxisErrorID	Axis fault code	Yes	0	*1	INT
D5	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D6	ErrorID	Fault code	Yes	0	*1	INT

Note

*1: See *“3.10.1.30 Axis Fault Codes” on page 404*

*2: For a local pulse axis, see the list of local pulse axis fault codes; for an EtherCAT bus drive, see the relevant manual of the EtherCAT bus drive.

Table 3–188 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	√[1]	√	√	-	-	√	-	-	-
D6	-	-	-	√	√	√	-	-	

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction is used to read the fault of the EtherCAT bus axis or the local axis.

When Enable is ON, the Valid signal becomes active if the requested axis exists and no configuration failure occurs. AxisErrorID displays the fault code of the axis in real time. When no fault occurs, it is 0;

when an axis fault occurs, it displays the fault code. If 0x603F is configured in the PDO of the bus driver, ServoErrorID displays the value of 0x603F in real time; otherwise, it displays 0.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

Axis initialization fails.

The axis type is incorrect.

Timing Diagram

Omitted.

3.10.1.7 MC_ReadDigitalInput

MC_ReadDigitalInput – Digital input read

Graphic Block

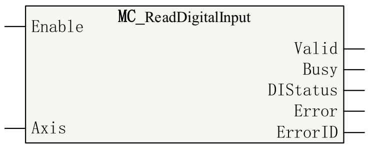
Instruction	Name	LD Expression	LiteST Expression
MC_ ReadDigitalInput	Read the digital input (DI) status		MC_ReadDigitalInput(Enable := ???, Axis := ???, Valid => , Busy => , DIStatus => , Error => , ErrorID =>);

Table 3–189 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ReadDigitalInput: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
D1	Valid	Active	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL

D3	DIStatus	DI terminal state. The standard format compliant with CiA402 is defined as follows: Bit0: Reverse limit signal; Bit1: Forward limit signal Bit2: Home signal; Bit3 to bit31: Customized	Yes	0	-	DINT
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Note *1: See *"3.10.1.30 Axis Fault Codes" on page 404*.

Table 3–190 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

Applicable to the EtherCAT bus axis and the local pulse axis, the MC_ReadDigitalInput instruction is used to read the DI terminal state of the axis. It does not support the imaginary axis mode.

When Enable is ON, if 0x60fd is configured in the PDO of the requested EtherCAT bus axis or any of the left and right limit and home signals of the local pulse axis is not empty, the Valid signal is active.

For the EtherCAT bus axis, DIStatus displays the digital input 0x60fd of the EtherCAT bus driver in real time. For details, see the relevant driver manual.

For the local pulse axis, DIStatus displays the input states of the limit and home signals or 0.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

Axis initialization fails.

The axis type is incorrect.

0x60fd is not configured in the PDO of the EtherCAT bus axis.

Timing Diagram

Omitted.

3.10.1.8 MC_ReadActualPosition

MC_ReadActualPosition – Current position read

Graphic Block

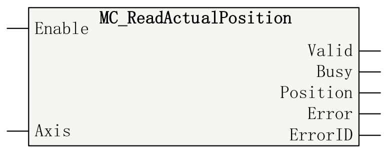
Instruction	Name	LD Expression	LiteST Expression
MC_ ReadActualPosition	Current position read		MC_ReadActualPosition(Enable := ???, Axis := ???, Valid => , Busy => , Position => , Error => , ErrorID =>);

Table 3–191 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualPosition: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Position	Current position	Yes	0	-	REAL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

Table 3–192 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction is used to read the feedback position of the EtherCAT bus axis or the local pulse axis. It is active high.

When Enable is ON, if 0x6064 is configured in the PDO of the EtherCAT bus axis, the Valid signal is active, and Position displays the feedback position of the axis.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

Axis initialization fails.

The axis type is incorrect.

0x6064 is not configured in the PDO of the EtherCAT bus axis.

Timing Diagram

Omitted.

3.10.1.9 MC_ReadActualTorque

MC_ReadActualTorque – Current torque read

Graphic Block

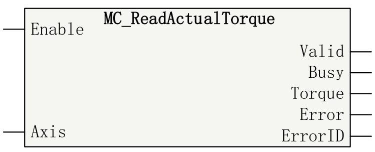
Instruction	Name	LD Expression	LiteST Expression
MC_ReadActualTorque	Read the current torque		MC_ReadActualTorque(Enable := ???, Axis := ???, Valid => , Busy => , Torque => , Error => , ErrorID =>);

Table 3–193 Instruction format

16-bit Instruction	
32-bit Instruction	MC_ReadActualTorque: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Torque	Current torque (unit: % 1)	Yes	0	Positive number, negative number, or 0	REAL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

Table 3–194 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction reads the feedback torque of the EtherCAT bus axis. It is active high. It does not support the imaginary axis mode.

When Enable is ON, if 0x6077 is configured in the PDO of the EtherCAT bus axis, the Valid signal is active, and Torque displays the feedback torque of the axis.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

The axis type is incorrect.

Axis initialization fails.

0x6077 is not configured in the PDO of the EtherCAT bus axis.

Timing Diagram

Omitted.

3.10.1.10 MC_ReadActualVelocity

MC_ReadActualVelocity – Current velocity read

Graphic Block

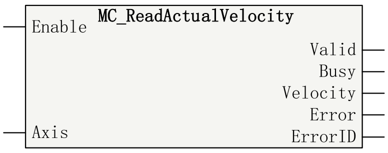
Instruction	Name	LD Expression	LiteST Expression
MC_ ReadActualVelocity	Read the current velocity		MC_ReadActualVelocity(Enable := ???, Axis := ???, Valid => , Busy => , Velocity => , Error => , ErrorID =>);

Table 3–195 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Valid	Active	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Velocity	Current velocity	Yes	0	-	REAL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–196 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction calculates the actual velocity based on the feedback velocity of the EtherCAT bus axis or the local pulse axis. It is active high.

When Enable is ON, if 0x6064 is configured in the PDO of the EtherCAT bus axis, the Valid signal is active, and Velocity displays the calculated velocity of the axis.

Abortion

This instruction has no abortion flag, and multiple instructions can be executed at the same time.

Errors

An error is returned in the following conditions: The axis number does not exist.

Axis initialization fails.

The axis type is incorrect.

0x6064 is not configured in the PDO of the EtherCAT bus axis.

Timing Diagram

Omitted.

3.10.1.11 MC_SetPosition

MC_SetPosition – Current position setting

Graphic Block

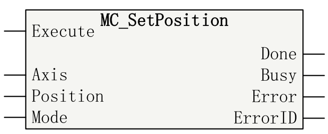
Instruction	Name	LD Expression	LiteST Expression
MC_SetPosition	Set the current position		MC_SetPosition(Execute := ???, Axis := ???, Position := ???, Mode := , Done => , Busy => , Error => , ErrorID =>);

Table 3–197 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_SetPosition: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Position	Target position	No	-	Positive number, negative number, or 0	REAL
S3	Mode	Control mode selection 0: Absolute mode (Write the value of Position as the current position.) 1: Relative mode (Add the value of Position based on the current position.)	Yes	0	0 to 1	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Fault code	Yes	0	*1	INT

Table 3–198 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	-	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction is used set the current position of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

This instruction can be executed only when the PLCOpen state of the axis is Disabled, StandStill, or ErrorStop; otherwise, an error occurs.

- When Mode is set to 0 (absolute mode), this instruction writes the value of Position as the current position of the axis on the rising edge of the Execute signal.
- When Mode is set to 1 (relative mode), this instruction adds the value of Position based on the current position of the axis on the rising edge of the Execute signal.

Abortion

This instruction does not support abortion. If there are several MC_SetPosition instructions in one scan cycle at the same time, the first active instruction will be executed; if other SetPosition instructions are executed during the period when the Busy signal of this instruction is valid, other instructions will report an error.

Errors

An error is returned in the following conditions: The axis number does not exist.

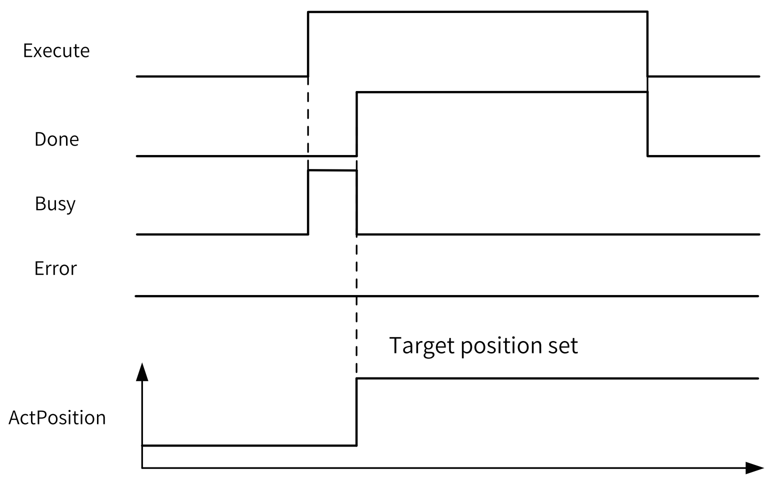
The axis type is incorrect.

Axis initialization fails.

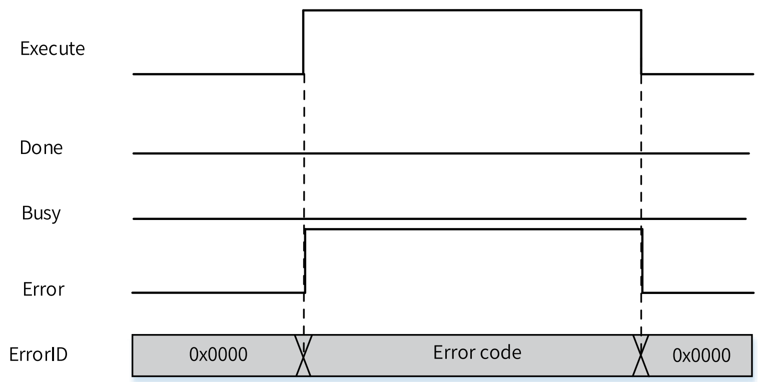
The MC_SetPosition instruction takes effect only when the axis has stopped operation, and it reports an error if the axis is in another state.

Timing Diagram

- The MC_SetPosition instruction (relative mode) is executed when the axis is in StandStill state.



- The MC_SetPosition instruction is executed during execution of the MC_Jog instruction.



3.10.1.12 MC_TouchProbe

MC_TouchProbe – Probe

Graphic Block

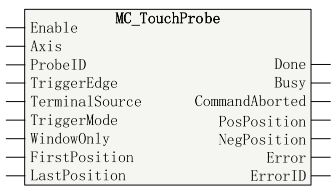
Instruction	Name	LD Expression	LiteST Expression
MC_TouchProbe	Probe		MC_TouchProbe(Enable := ???, Axis := ???, ProbeID := ???, TriggerEdge := ???, TerminalSource := , TriggerMode := , WindowOnly := ,

Table 3–199 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_TouchProbe: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	ProbeID	Probe ID 0: Probe 1 1: Probe 2	No	-	0 to 1	INT
S3	TriggerEdge	Trigger edge 0: Only rising edge 1: Only falling edge 2: Both rising edge and falling edge	No	-	0 to 2	INT

S4	Terminal-Source	Probe signal source (only for setting bus servo drive) 0: DI terminal 1: Encoder Z signal	Yes	0	0 to 1	INT
S5	TriggerMode	Trigger mode 0: Single trigger 1: Continuous trigger	Yes	0	0 to 1	INT
S6	WindowOnly	Probe window enable 0: Disabled. Probe signals are detected at all positions. 1: Enabled. Probe signals are detected only when the current position is between FirstPosition and LastPosition (included).	Yes	OFF	ON/OFF	BOOL
S7	FirstPosition	Probe window start position	Yes	0	Positive number, negative number, or 0	REAL
S8	LastPosition	Probe window end position	Yes	0	> Position	REAL
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion	Yes	OFF	ON/OFF	BOOL
D4	PosPosition	Position latched on the rising edge	Yes	0	Positive number, negative number, or 0	REAL
D5	NegPosition	Position latched on the falling edge	Yes	0	Positive number, negative number, or 0	REAL
D6	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D7	ErrorID	Fault code	Yes	0	*1	INT

Table 3–200 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	√	√	√	-	-	√	-	-	-
S7	-	-	-	√	√	√	-	√	-
S8	-	-	-	√	√	√	-	√	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-
D6	-	-	-	√	√	-	-	-	-
D7	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

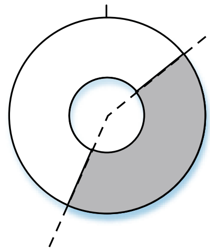
Function and Instruction Description

This instruction implements the probe function of the EtherCAT bus axis or the local pulse axis. It is active high. It does not support the imaginary axis mode.

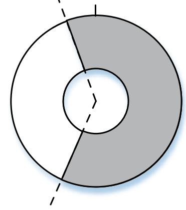
In EtherCAT bus axis mode, the probe function (0x60b8), probe state (0x60b9), and latched position (0x60ba/0x60bb/0x60bc/0x60bd) need to be configured for the drive.

In the local pulse axis mode, the probe signal source needs to be configured.

- On the rising edge, the instruction latches the input parameters on the left, such as ProbeID and TriggerEdge, and other state update parameters are invalid.
- When Enable is ON, the function block latches the current position of the axis when the instruction detects that the input of the probe specified by ProbeID is active and meets the probe detection conditions.
- When WindowOnly is OFF, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.
- When WindowOnly is ON, the window detection function is enabled.
- In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.
- In ring mode, the instruction first uses FirstPosition and LastPosition to mod the cycle to obtain FirstPosition_p and LastPosition_p in a cycle.
  - When FirstPosition_p is less than LastPosition_p, the valid window range is as follows:



  - When FirstPosition_p is greater than LastPosition_p, the valid window range is as follows:



This instruction can detect the rising edge or falling edge of the probe signal separately or both the rising edge and the falling edge at the same time.

When detecting only the rising edge (falling edge), the instruction writes the value detected on the rising edge (falling edge) into PosPosition (NegPosition). At this time, the Done signal is set to ON when a detection cycle is completed.

If the rising edge and falling edge are detected at the same time, after the Enable signal is active, the instruction immediately writes the position into PosPosition upon detecting the rising edge and writes the position into NegPosition upon detecting the falling edge. After that, the detection cycle is completed and the Done signal is output. There is no requirement on the input sequence of the rising edge and falling edge.

- For the EtherCAT bus driver, the input TerminalSource of this instruction can be used to set the terminal type to DI or Z signal of the motor (driver support required). No error is reported if the driver does not support the Z signal.
- This instruction supports the single trigger and continuous trigger modes. If the single trigger mode is used, instruction execution ends when the Done signal output is active. If the continuous trigger mode is used, the Done output active signal is reset after one PLC scan cycle, and the instruction automatically starts to detect new probe input signals.

Note

When the window function is enabled, probe signal loss or detection out-of-range may occur near the window area. The following is an example:

- In linear mode, the window range is 10 to 100, the EtherCAT cycle is set to 8 ms, and the velocity is 100. Then the axis moves 0.8 per EtherCAT cycle. If the current position at the moment when an EtherCAT cycle starts is 9.9, the probe signal is not detected within this EtherCAT cycle. The current position changes to 10.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 10 and 10.7 are lost. If the current position at the moment when an EtherCAT cycle starts is 99.9, the probe signal is detected within this EtherCAT cycle. The current position changes to 100.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 100 and 100.7 are responded.
- In continuous mode, if the input frequency of the probe signal is greater than the frequency of the PLC scan cycle, some probe signals are lost.

Abortion

The MC_TouchProbe instruction supports the detection probe 1 and probe 2. If two probe instructions are defined in the program and the probe IDs of the two instructions are different, the two probe instructions will work independently. If the probe IDs are the same, the probe instruction executed later will abort the previous probe instruction.

The MC_MoveFeed instruction also uses the probe signal. The execution rules of these two instructions are as follows:

- When an MC_TouchProbe instruction and an MC_MoveFeed coexist in a program, if their probe IDs are different, the two instructions work independently.
- If their probe IDs are the same, the situation is as follows: If the MC_TouchProbe instruction is enabled first, and the Mc_MoveFeed instruction is enabled while the Busy signal of the MC_TouchProbe instruction is still active, the MC_TouchProbe instruction is aborted. If the Mc_MoveFeed instruction is enabled first, and the MC_TouchProbe instruction is triggered while the Busy signal of the Mc_MoveFeed instruction is active but the InFeed signal is inactive, the MC_TouchProbe instruction reports an error.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

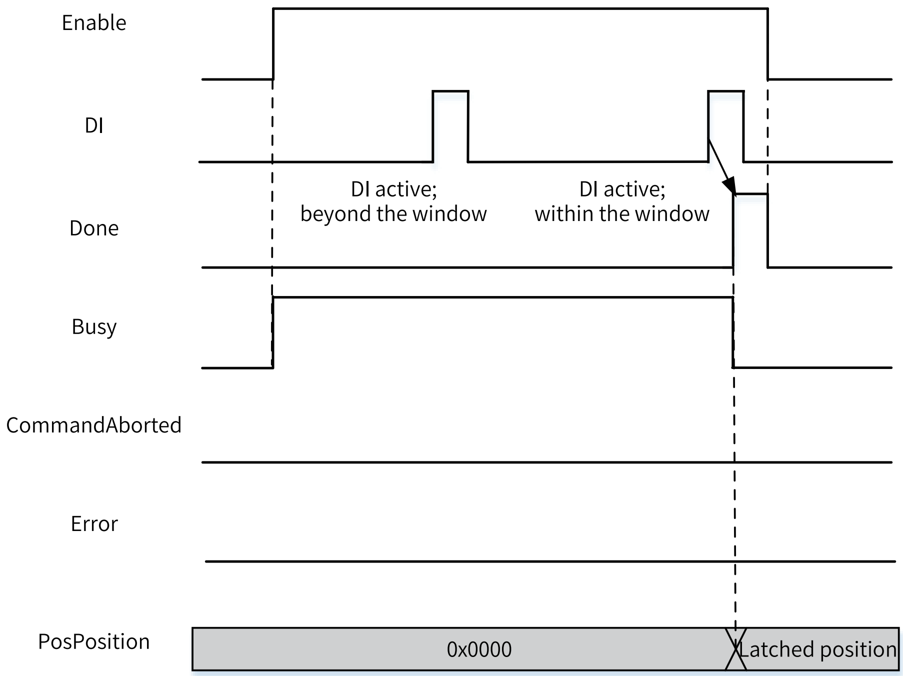
Error 9133 is reported when the imaginary axis mode is enabled.

A fault is reported when the corresponding PDO instruction is not configured.

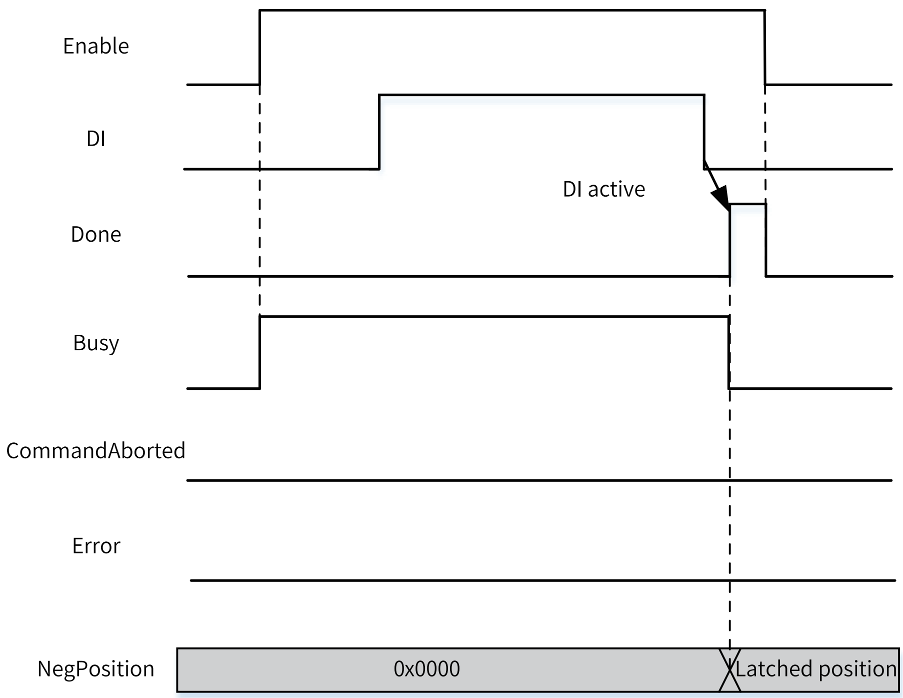
A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Enable input.

Timing Diagram

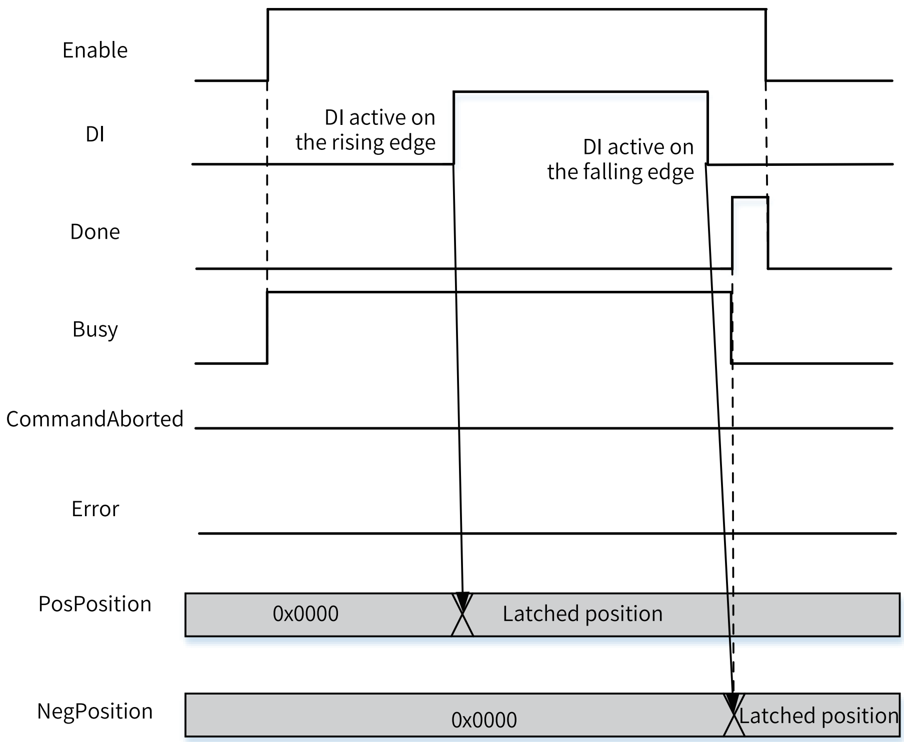
- Probe 1, active on the rising edge, DI signal trigger source, single trigger mode, window function enabled



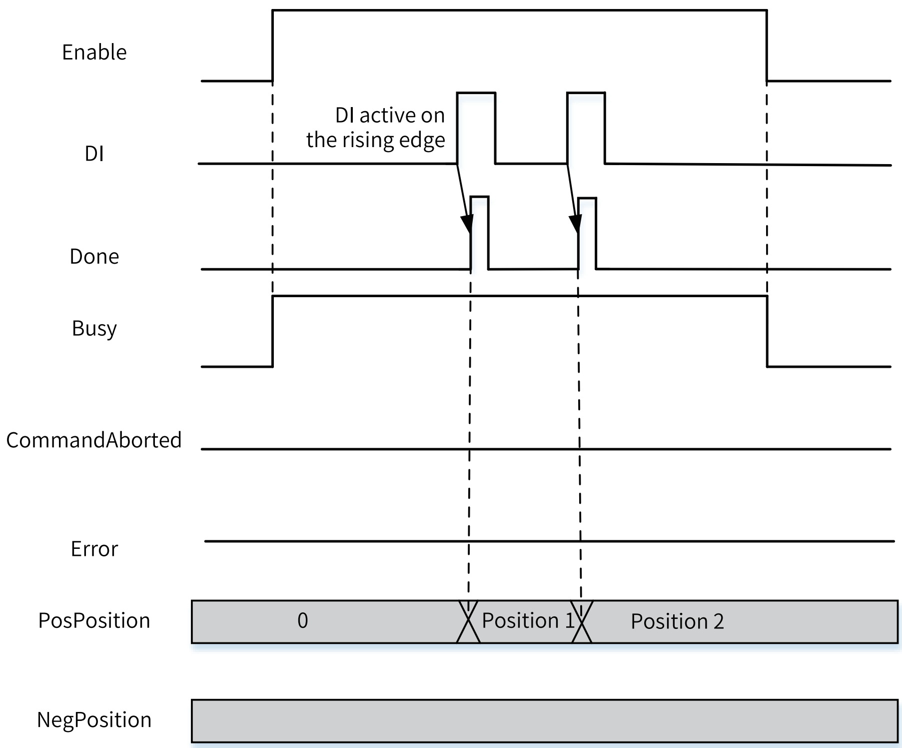
- Probe 1, active on the falling edge, DI signal trigger source, single trigger mode, window function disabled



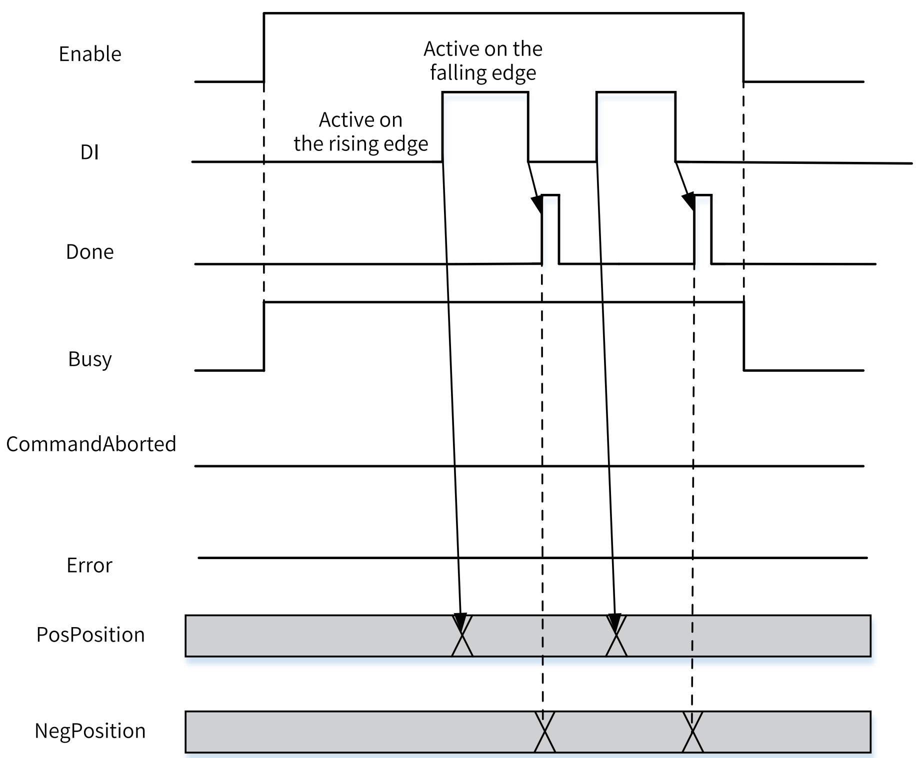
- Probe 1, active on both the rising edge and falling edge, DI signal trigger source, single trigger mode, window function disabled



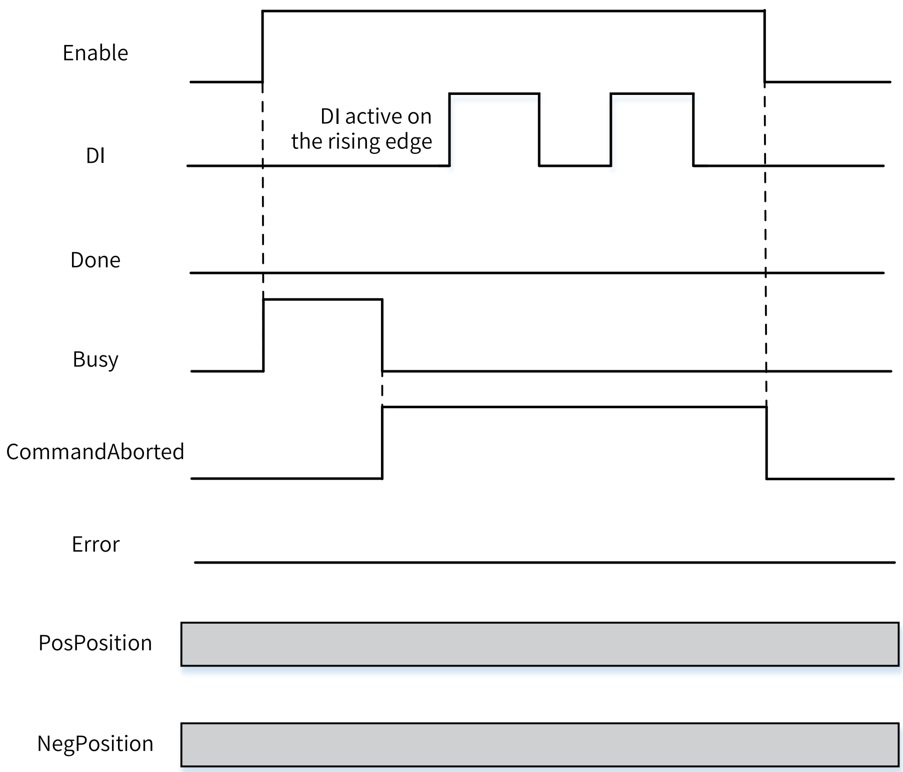
- Probe 1, active on the rising edge, DI signal trigger source, continuous trigger mode, window function disabled



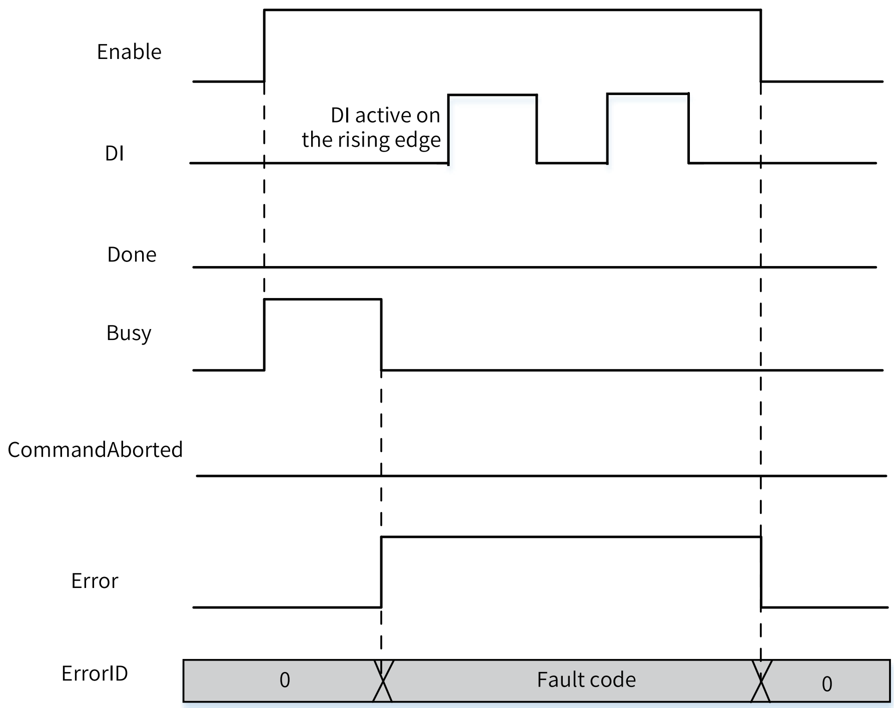
- Probe 1, active on both the rising edge and falling edge, DI signal trigger source, continuous trigger mode (the Done signal is active for a cycle after the DI signal is active on both the rising and falling edges), window function disabled



- Probe 1, aborted by another probe-related instruction, window function disabled



- Probe 1 instruction error



3.10.1.13 MC_MoveRelative

MC_MoveRelative – Relative positioning

Graphic Block

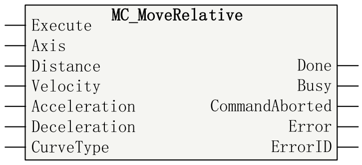
Instruction	Name	LD Expression	LiteST Expression
MC_MoveRelative	Relative positioning		MC_MoveRelative(Execute := ???, Axis := ???, Distance := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–201 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveRelative: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Distance	Target position	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Target velocity	No	-	Positive number, less than the maximum velocity	REAL
S4	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S6	CurveType	Curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT

D1	Done	Target position reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

Table 3–202 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

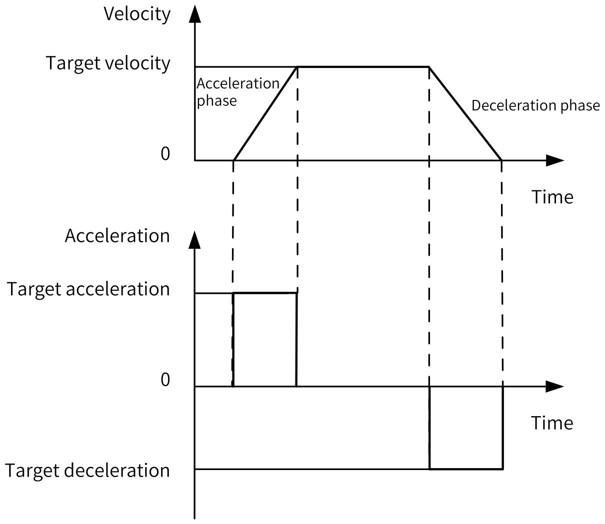
This instruction implements relative positioning of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Distance and Velocity, triggers the relative positioning function, and switches the PLCOpen state machine of the axis to the DiscreteMotion state.

  - Distance specifies the distance for relative positioning. No matter in linear mode or ring mode, if Distance is positive, the axis travels the distance specified by Distance in the forward direction; if Distance is negative, the axis travels in the reverse direction by the distance specified by |Distance|.

- CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.



Target position: Final position of the axis in the relative positioning instruction; unit: Unit (user unit)

Target velocity: Maximum allowable velocity of the axis during running; unit: Unit/s

Target acceleration: Change in velocity per second during acceleration; unit: Unit/s^2

Target deceleration: Change in velocity per second during deceleration; unit: Unit/s^2

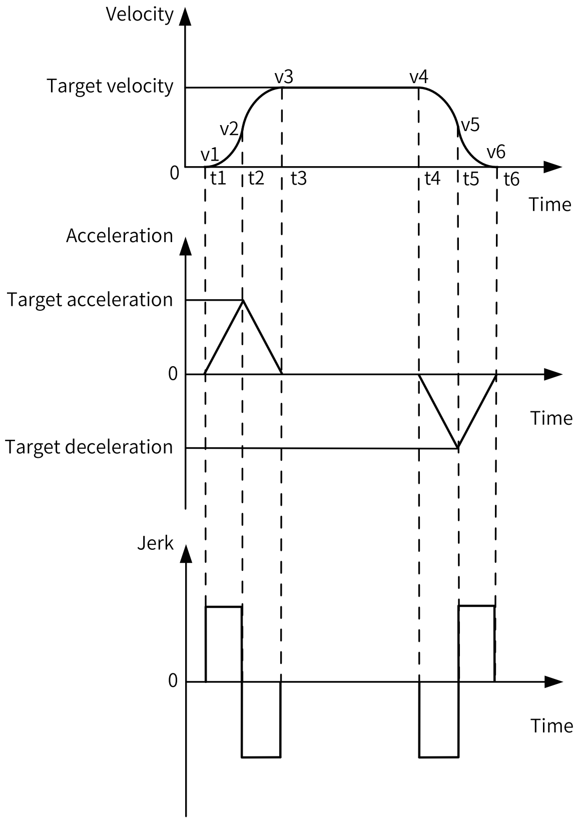
During acceleration, assume that the initial velocity of the axis is Vs, the target velocity is Vt, and the target acceleration is Acc. Then the acceleration time is calculated as follows:

“Tacc = (Vt – Vs)/Acc”

During deceleration, assume that the initial velocity of the axis is Vs, the target velocity is Ve, and the target deceleration is Dec. Then the deceleration time is calculated as follows:

“Tdec = (Vs – Ve)/Dec”

- If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.



The 5-segment S-curve is divided into five segments based on the acceleration state: increasing-acceleration, decreasing-acceleration, constant velocity, increasing-deceleration, and decreasing-deceleration. Constant acceleration or deceleration does not exist. The actual jerk during the variable-acceleration phase (such as increasing-acceleration and increasing-deceleration) is calculated internally by the PLC and cannot be set by the user.

Target position: Final position of the axis in the relative positioning instruction; unit: Unit (user unit)

Target velocity: Maximum allowable velocity of the axis during running; unit: Unit/s

Target acceleration: Maximum change in velocity per second during variable-acceleration operation; unit: Unit/s^2. The acceleration at the moment (t2) when the velocity changes from the increasing-acceleration segment to the decreasing-acceleration segment in the velocity curve must be the target acceleration.

Target deceleration: Maximum change in velocity per second during variable-deceleration operation; unit: Unit/s^2. The deceleration at the moment (t5) when the velocity changes from the decreasing-acceleration segment to the decreasing-deceleration segment in the velocity curve must be the target deceleration.

During acceleration, assume that the initial velocity of the axis is V1, the target velocity is V3, and the target acceleration is Acc. Then the acceleration time is calculated as follows:

“Tacc = 2 x (V3 – V1)/Acc”
During deceleration, assume that the initial velocity of the axis is V4, the target velocity is V6, and the target deceleration is Dec. Then the deceleration time is calculated as follows:

"Tdec = 2 x (V4 – V6)/Dec"

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

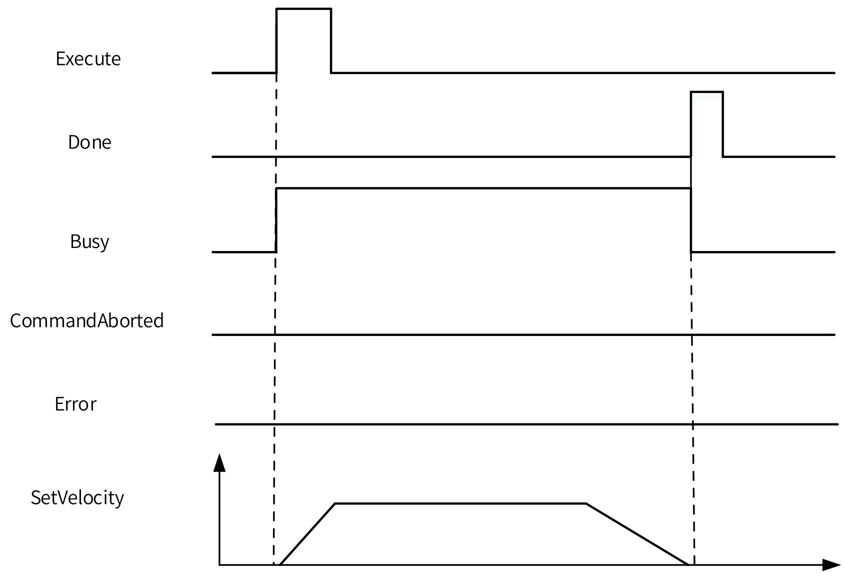
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

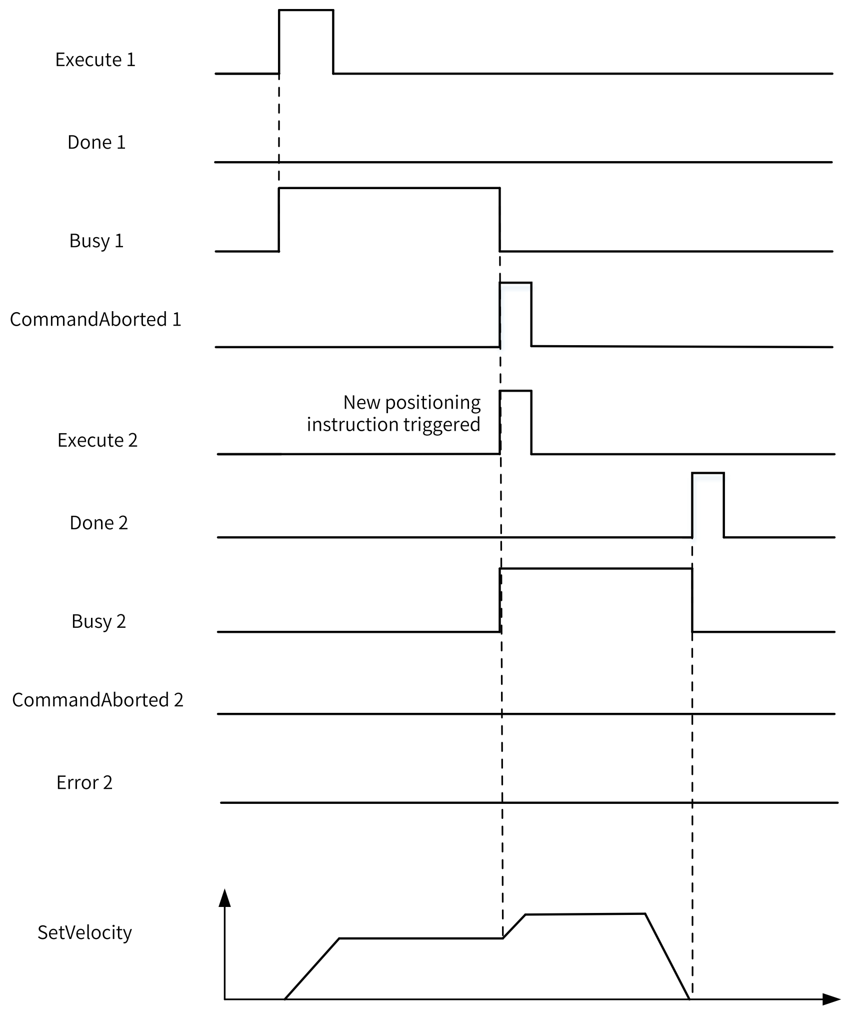
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

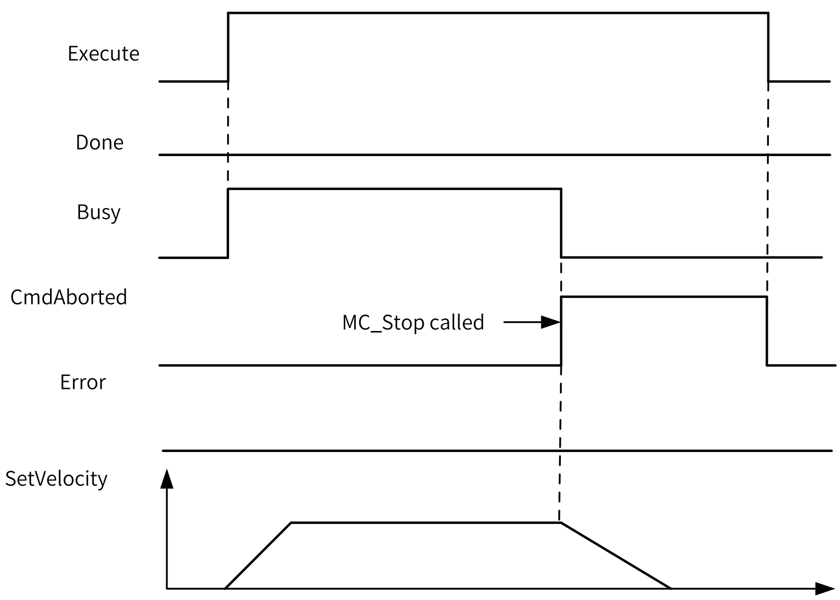
- The MC_MoveRelative instruction is executed to implement relative positioning based on the T-shaped curve when the axis is in StandStill state.



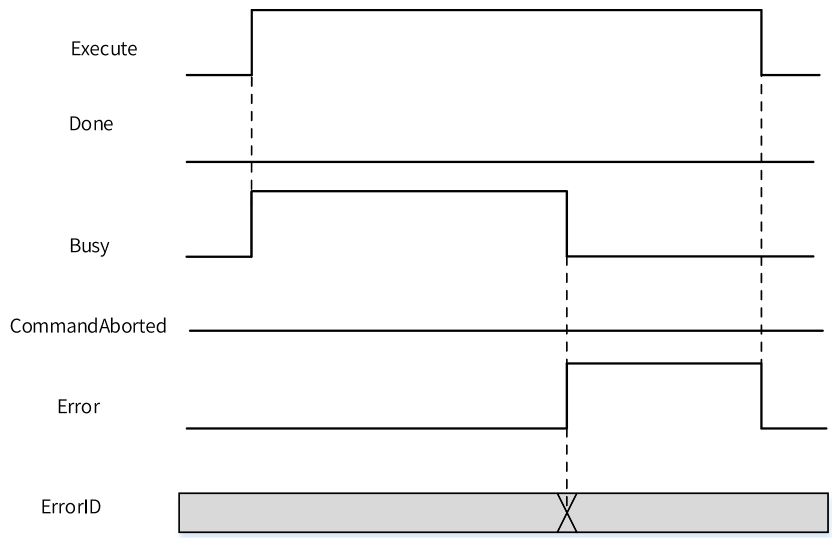
- Another relative positioning instruction is triggered during relative positioning.



- Relative positioning of the axis is aborted by the Mc_Stop instruction.



- The drive fails during motion of the axis.



3.10.1.14 MC_MoveVelocity

MC_MoveVelocity – Velocity control

Graphic Block

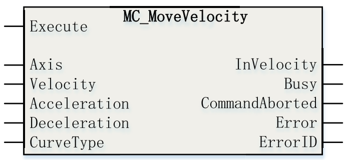
Instruction	Name	LD Expression	LiteST Expression
MC_MoveVelocity	Speed reference		MC_MoveVelocity(Execute := ???, Axis := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–203 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO

S2	Velocity	Target velocity	No	-	Positive number/ number/0, absolute value less than the maximum velocity	REAL
S3	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S4	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S5	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
D1	InVelocity	Velocity reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–204 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements absolute positioning of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis

- Axis is latched on the rising edge of the Execute input.
- If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
- If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.

- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the Execute input, the instruction latches the input parameters on the left, such as Velocity and Acceleration, triggers the axis to run at the velocity specified by Velocity, and switches the PLCOpen state machine of the axis to the ContinuousMotion state.

  CurveType specifies the type of the velocity curve.

  - If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.
  - If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.

During execution of this instruction, you can call MC_Stop, Mc_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Enable is ON and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

When Enable is ON and the Busy signal is active, if deceleration to stop needs to be performed upon a limit signal, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

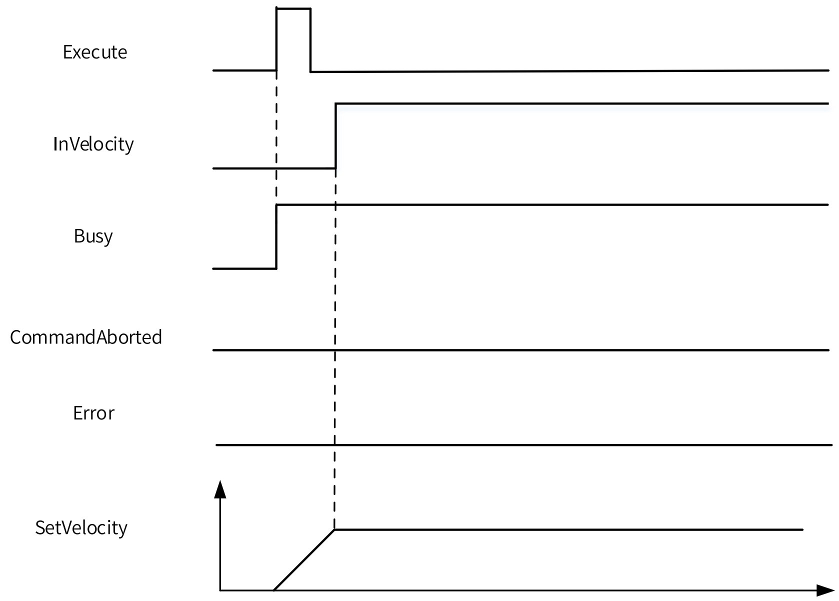
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

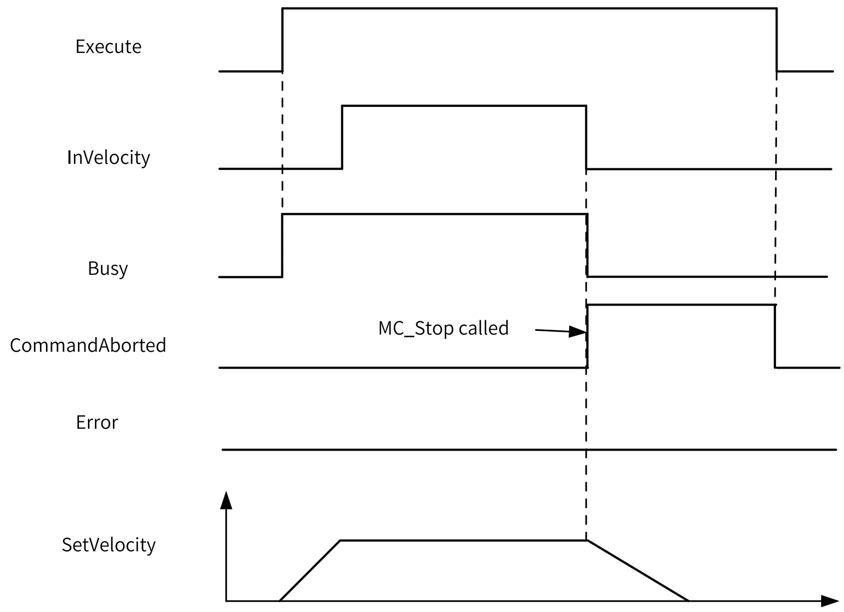
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

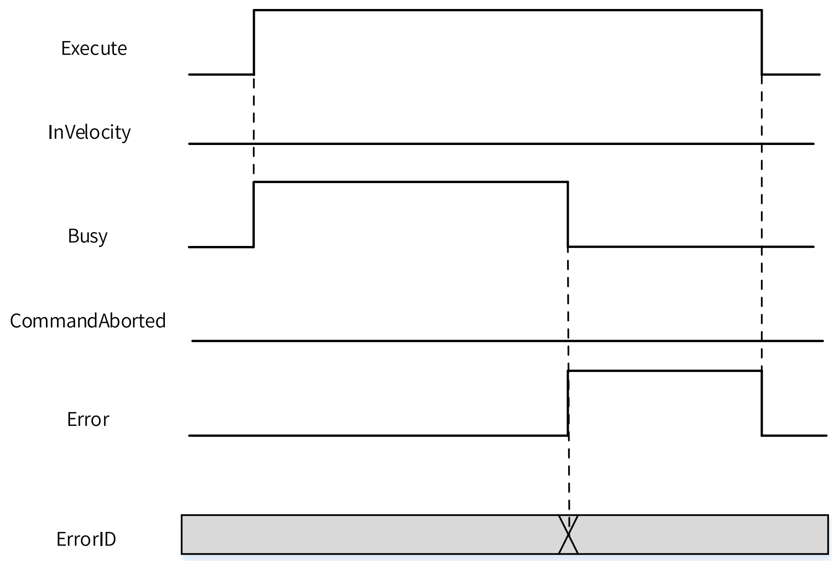
- The MC_MoveVelocity instruction is executed to implement continuous motion based on the T-shaped curve when the axis is in StandStill state.



- Axis motion is aborted by the MC_Stop instruction.



- The drive fails during acceleration of the axis.



3.10.1.15 MC_MoveAbsolute

MC_MoveAbsolute – Absolute positioning

Graphic Block

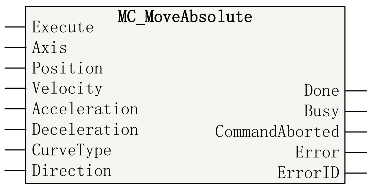
Instruction	Name	LD Expression	LiteST Expression
MC_MoveAbsolute	Absolute positioning		MC_MoveAbsolute(Execute := ???, Axis := ???, Position := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Direction := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–205 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveAbsolute: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Position	Target position	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Target velocity	No	-	Positive number, less than the maximum velocity	REAL
S4	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S6	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
S7	Direction	Direction (applicable to only the ring mode) 0: Forward (velocity > 0) 1: Reverse (velocity < 0) 2: Minimum distance 3: Current direction	Yes	0	0 to 3	INT
D1	Done	Target position reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–206 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

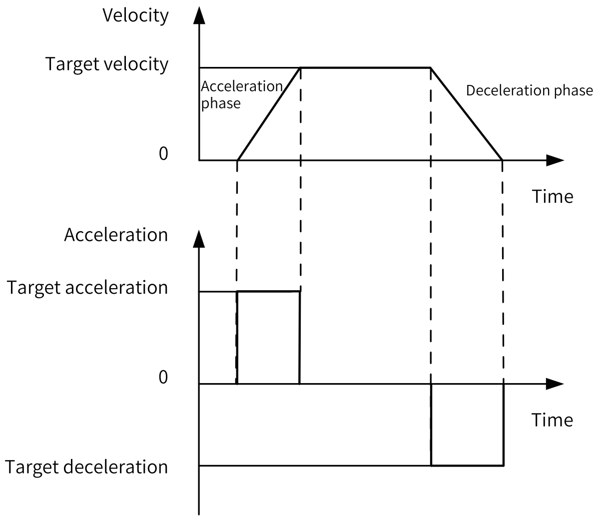
Function and Instruction Description

This instruction implements absolute positioning of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

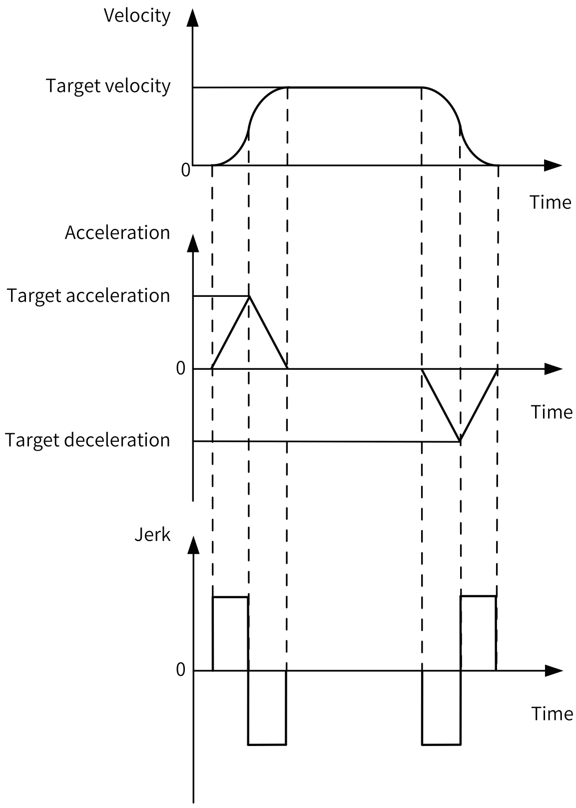
  On the rising edge of the Execute input, the instruction latches the input parameters on left, such as Position and Velocity, triggers the absolute positioning function, and switches the PLCOpen state machine of the axis to the DiscreteMotion state.

  - In linear mode, Position specifies the target position for absolution positioning. If the current position is less than the target position, the axis moves forward to reach the position specified by Position. If the current position is greater than the target position, the axis moves in the reverse direction to reach the position specified by Position.
  - CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.



For details about the T-shaped curve, see the relative positioning instruction section.

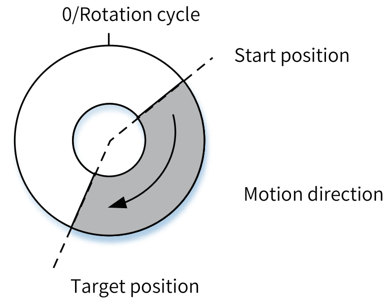
- If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.



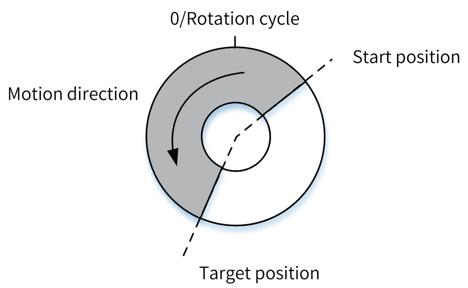
For details about the S-curve, see the relative positioning instruction section.

In ring mode, the instruction first uses Position to mod the revolution cycle to obtain the absolute position Position_p in a revolution cycle. The actual direction of the axis is determined based on the following four conditions:

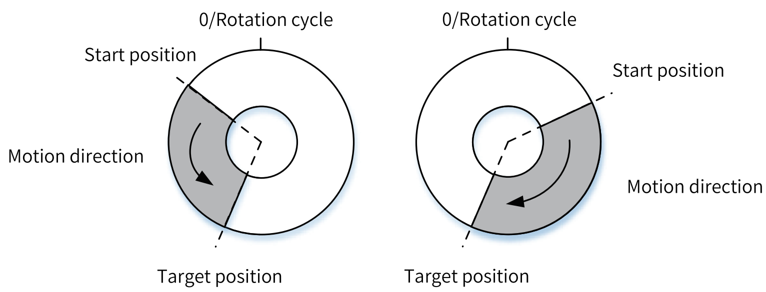
1. Direction = 0 (Forward, target velocity > 0). If the current velocity is greater than 0, the axis continues to run in the current direction and stops at the position specified by Position_p; if the current velocity is less than 0, the axis decelerates to 0 and then starts to move at reserve velocity until it reaches the position specified by Position_p; if the current position is just the position specified by Position_p***1**, the axis does not move.



2. Direction = 1 (Reverse, target velocity < 0). If the current velocity is less than 0, the axis continues to run in the current direction and stops at the position specified by Position_p; if the current velocity is greater than 0, the axis decelerates to 0 and then starts to move at reserve velocity until it reaches the position specified by Position_p; if the current position is just the position specified by Position_ p***1**, the axis does not move.



3. Direction = 2 (Minimum distance). The current position of the axis is recorded on the rising edge of the Execute signal. Assume that the current velocity is 0. Distance indicates the distance that the axis moves forward from 0 velocity to the position specified by Position_p. If Distance is less than or equal to 0.5*revolution cycle, the axis moves forward; if it is greater than 0.5*revolution cycle, the axis moves in the reverse direction; if the current position is just the position specified by Position_ p***1**, the axis does not move.



4. Direction = 3 (Current direction). On the rising edge of the Execute signal, the axis moves in the direction same as that before the rising edge of the Execute signal until it reaches the position specified by Position_p. If the machine is powered on for the first time, the axis moves in the forward direction (target velocity > 0). If the current position is just the position specified by Position_p***1**, the axis does not move.

Note

***1:** In ring mode, if the target position is greater than the ring cycle, the instruction uses the target position to mod the ring period to obtain a new target position. If the absolute value of the difference between the new target position and the set position of the axis is less than 0.001, the two values are considered equal.

Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Enable is ON and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

When Enable is ON and the Busy signal is active, if deceleration to stop needs to be performed upon a limit signal, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

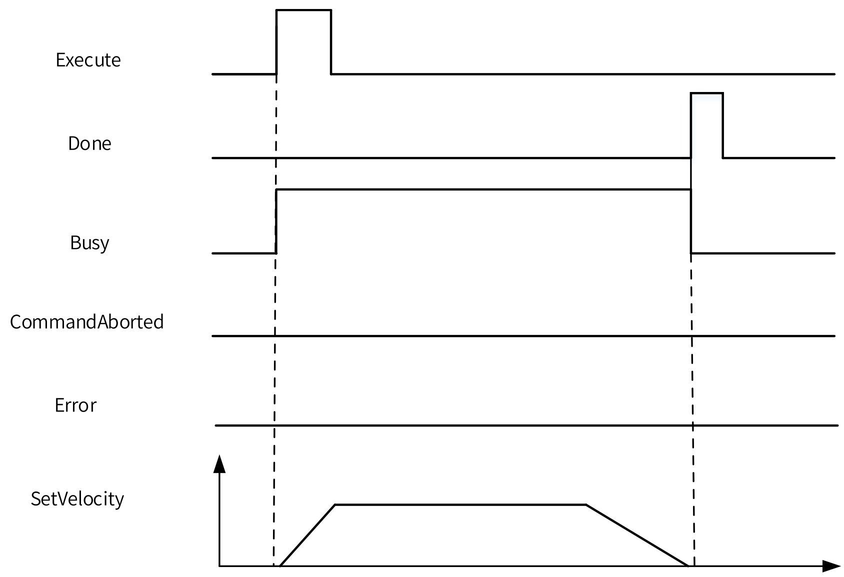
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

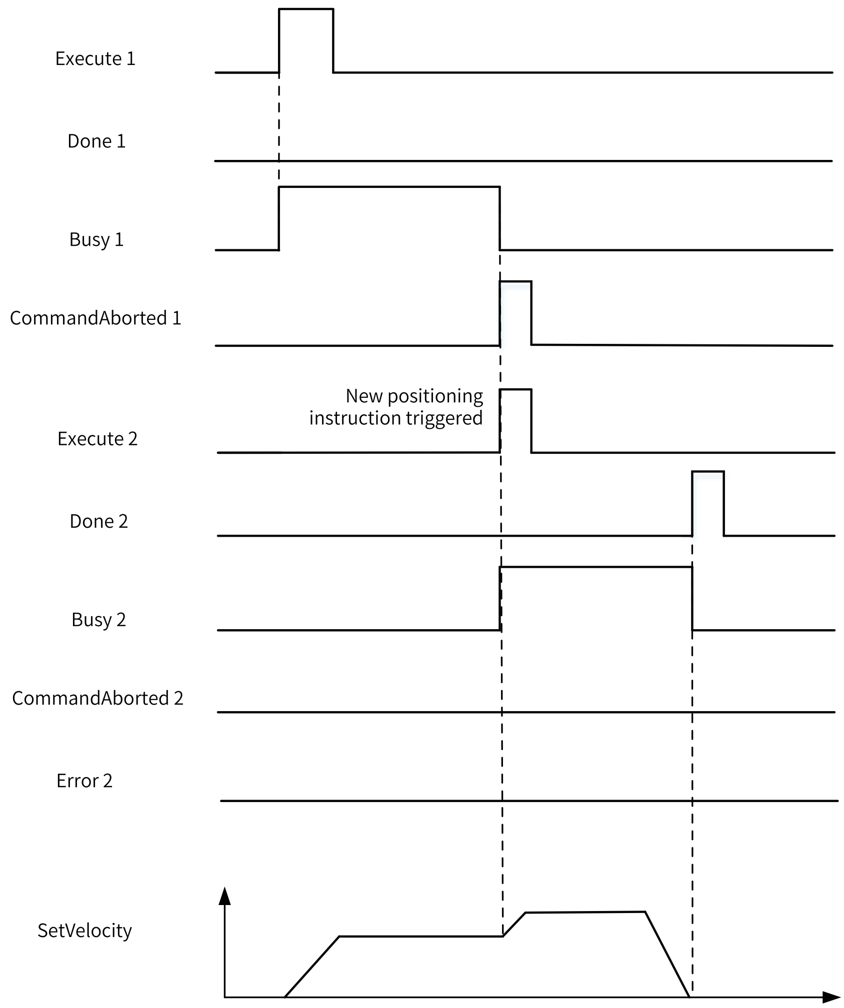
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

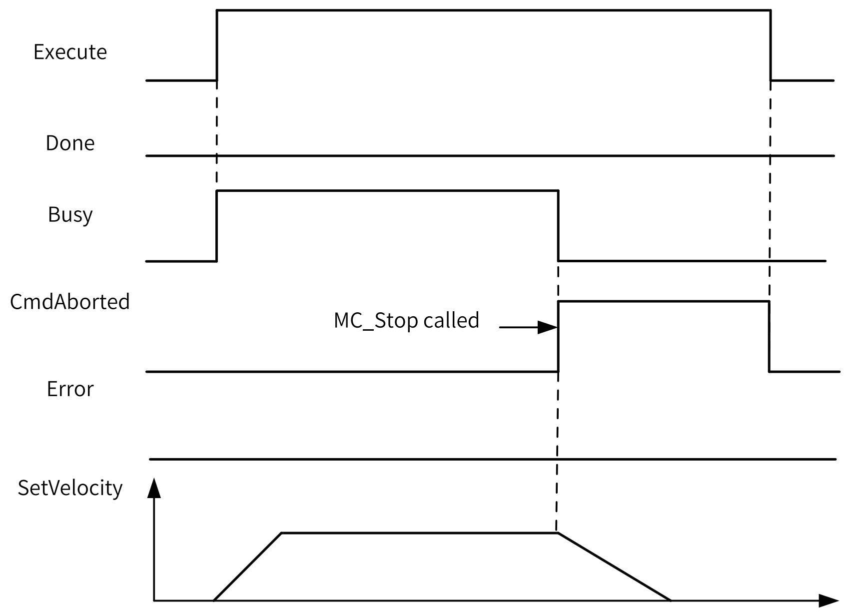
- The MC_MoveAbsolute instruction is executed to implement absolute positioning based on the T-shaped curve when the axis is in StandStill state.



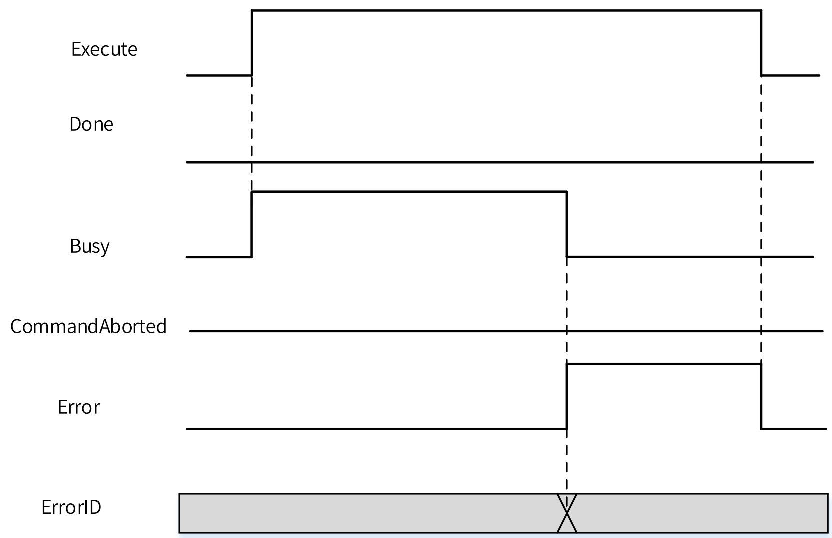
- Another absolute positioning instruction is triggered during absolute positioning.



- Absolute positioning of the axis is aborted by the MC_Stop instruction.



- The drive fails during motion of the axis.



3.10.1.16 MC_Jog

MC_Jog – Jogging

Graphic Block

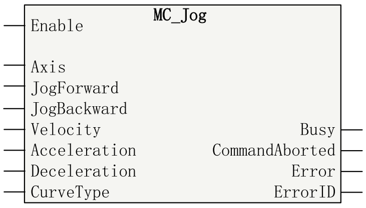
Instruction	Name	LD Expression	LiteST Expression
MC_Jog	Jog		MC_Jog(Enable := ???, Axis := ???, JogForward := ???, JogBackward := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–207 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_Jog: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	JogForward	Jogging in forward direction, triggered on the rising edge	No	-	-	BOOL
S3	JogBackward	Jogging in reverse direction, triggered on the rising edge	No	-	-	BOOL
S4	Velocity	Target velocity	No	-	Positive number, less than the maximum velocity, less than the maximum jogging velocity	REAL
S5	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S6	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S7	CurveType	Curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT

D1	Busy	Busy flag	Yes	OFF	-	BOOL
D2	CommandA-borted	Abortion of execution	Yes	OFF	-	BOOL
D3	Error	Error flag	Yes	OFF	-	BOOL
D4	ErrorID	Fault code	Yes	0	*1	INT

Table 3–208 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	-	-	-	-
S3	√	√	√	-	-	-	-	-	-
S4	-	-	-	√	√	-	-	√	-
S5	-	-	-	√	√	-	-	√	-
S6	-	-	-	√	√	-	-	√	-
S7	-	-	-	√	√	-	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the jogging function of the EtherCAT bus axis or the local pulse axis. It is active high.

- Specifying axis
  - Axis is latched on the rising edge of the Enable input.
  - If Axis specifies the axis number, when it is modified while Enable is ON, the previously controlled axis enters the ErrorStop state.
  - If Axis specifies the axis number, modification on Axis is valid when Enable is OFF.
- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block latches the input parameters, such as Velocity, Acceleration, Deceleration, and CurveType, and switches the state machine of the axis to the ContinuousMotion state to start jogging.

  - When Enable is ON, if instructions such as MC_Stop and MC_MoveRelative are called, the MC_Jog will be aborted, and the CommandAborted output of MC_Jog instruction becomes active.
  - When JogForward is active, the axis moves forward at the velocity specified by Velocity; when JogBackward is active, the axis moves in reverse direction at the velocity specified by Velocity.

When both JogForward and JogBackward are active, the axis stops but does not enter the ErrorStop state, and the instruction reports a fault.

- When Enable is ON, if the axis reaches the limit when moving toward one direction, the instruction reports a fault, and the axis stops but does not enter the ErrorStop state. When the MC_Jog instruction is triggered again, the axis will move toward the opposite direction.
- CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Enable is ON and the Busy signal is active, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

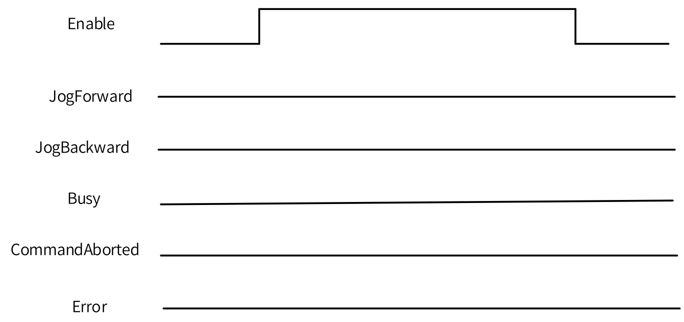
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

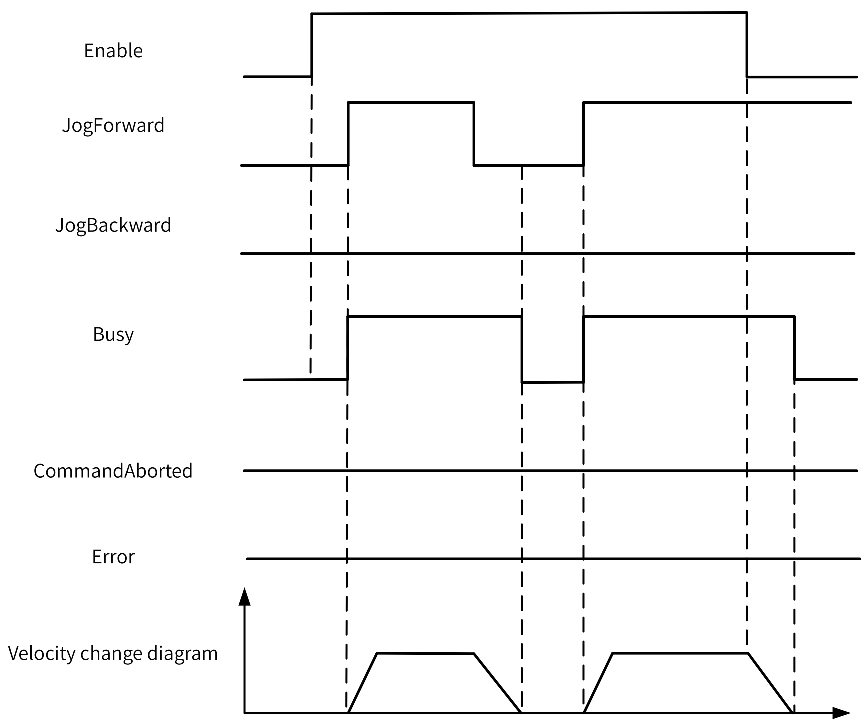
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Enable input.
- Error 9106 is reported if the axis is decelerating upon a fault on the rising edge of the Enable input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Enable input.
- Error 9116 is reported if the axis enters the commissioning state when Enable is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Enable is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

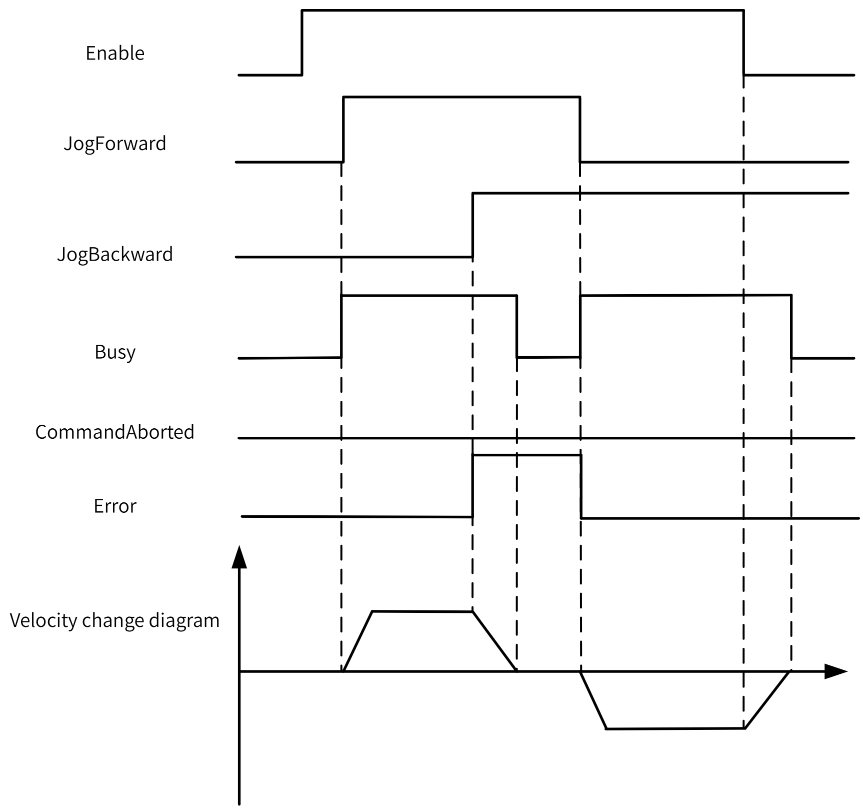
- The instruction has no action when only then Enable input is active.



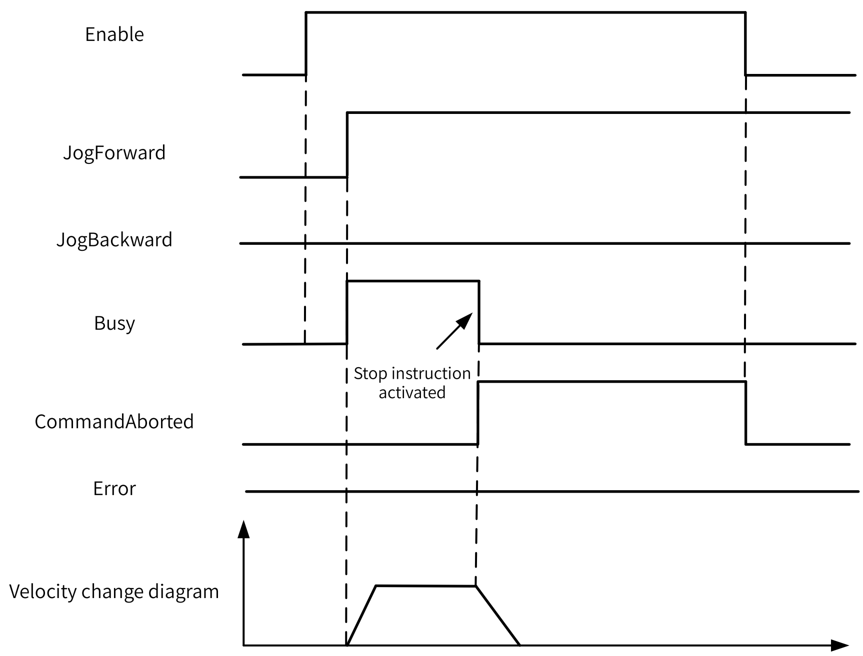
- The Enable and JogForward inputs are active.



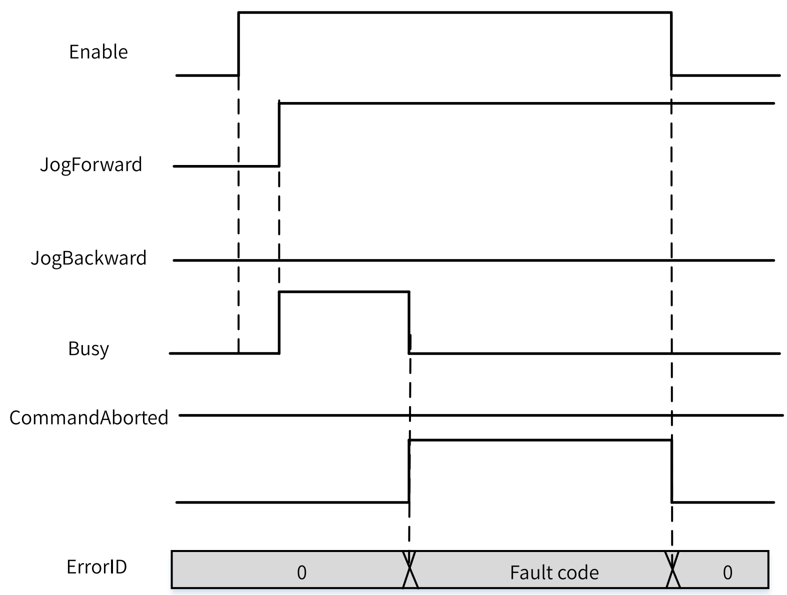
- The Enable and JogForward inputs are active, and JogBackward is set to ON.



- Instruction execution is aborted by the MC_Stop instruction.



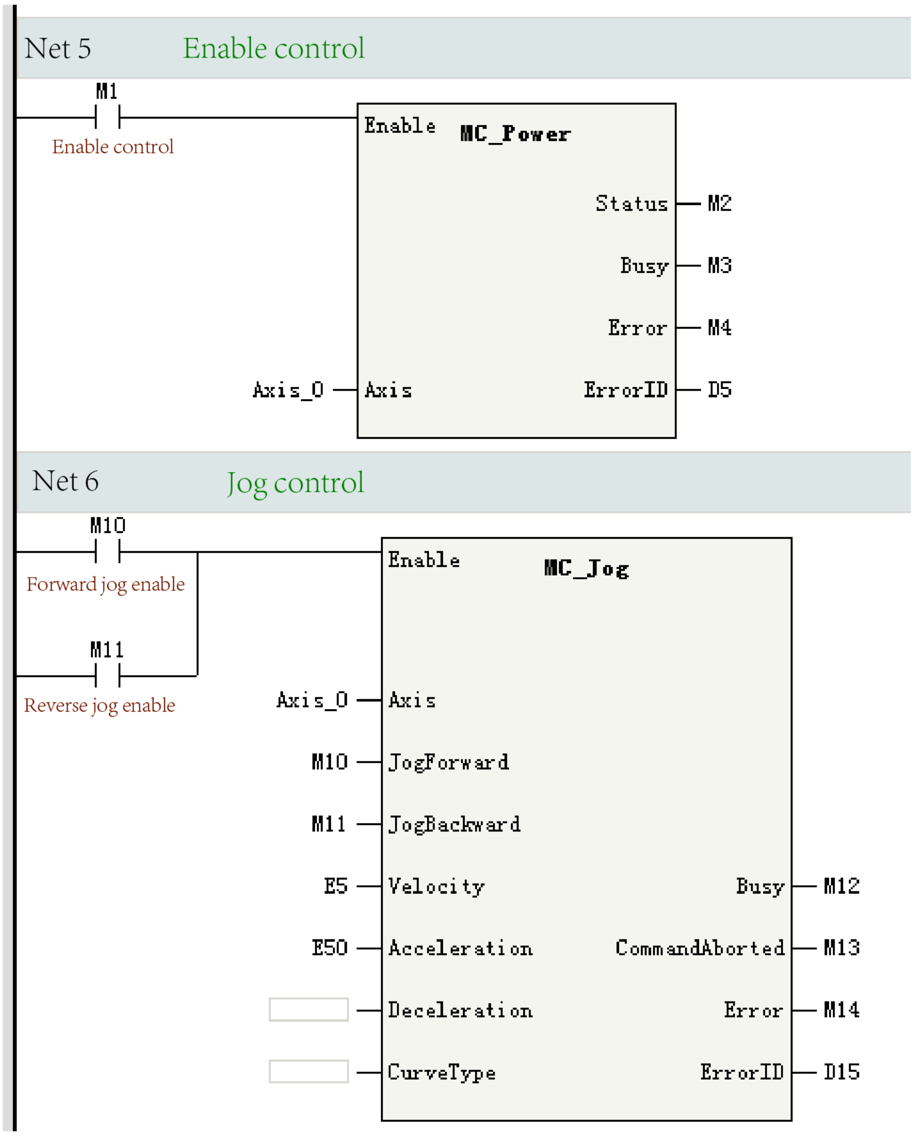
- The axis reports an error.



Routines

The following are some routines.

1. After M1 is set to ON, Axis_0 is enabled.
2. After M10 is set to ON, Axis_0 runs in forward direction at 5 unit/s.
3. After M10 is set to OFF, Axis_0 stops running.
4. After M11 is set to ON, Axis_0 runs in reverse direction at 5 unit/s.
5. After M11 is set to OFF, Axis_0 stops running.



3.10.1.17 MC_TorqueControl

MC_TorqueControl – Torque control

Graphic Block

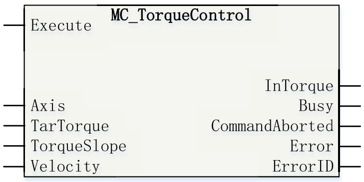
Instruction	Name	LD Expression	LiteST Expression
MC_TorqueControl	Torque control		MC_TorqueControl(Execute := ???, Axis := ???, TarTorque := ???, TorqueSlope := ???, Velocity := , InTorque => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–209 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_TorqueControl: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
S2	TarTorque	Target torque (unit: 1%)	No	-	Positive number, negative number, or 0	REAL
S3	TorqueSlope	Torque slope (unit: 1%)	No	-	Positive number	REAL
S4	Velocity	Velocity limit	No	-	Positive number or 0	REAL
D1	InTorque	Torque reached The output is active when the set torque reaches the target torque and the absolute value of the difference between the feedback torque and the target torque is less than 5%.	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CommandA-borted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–210 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction is used to implement the torque control function only for the EtherCAT bus axis. It is active on the rising edge and does not support the imaginary axis mode.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  The torque instruction can be used only when the following PDOs are configured: 0x6040, 0x6041, 0x6060, 0x6061, 0x6071, and 0x6077. Otherwise, a fault is reported.

  This instruction adopts the synchronous torque mode of the drive to implement the torque control function.

  The function block latches the input parameters TarTorque, TorqueSlope, and Velocity on the rising edge of the instruction. The axis enters the ContinuousMotion state and performs torque motion.

  - TarTorque: Target torque, in the unit of 1%. Only one decimal place after the decimal point is valid in the program, and the subsequent ones are directly discarded. The actual torque of the drive is limited by the maximum positive and negative torque specified in the configuration parameters.
  - TorqueSlope: Torque slope, in the unit of 1%. Only one decimal place after the decimal point is valid in the program, and the subsequent ones are directly discarded.
- Velocity control in torque mode

  For servo drives of Inovance, if 0x607f is mapped, this instruction limits the maximum velocity of the servo motor through 0x607f. If 0x607f is not mapped, the velocity limit is invalid.

On the rising edge of Execute, the instruction converts the velocity limit specified by Velocity into pulse unit and writes it into 0x607f through PDO.

If the torque instruction is aborted by another instruction, the maximum velocity of the axis can be limited by specifying Max. Velocity on the configuration interface.

For third-party drives, Velocity can be used as the velocity limit only when the following conditions are met:

- The maximum velocity of the servo motor can be limited by using 0x607F.
- 0x607F can be configured in the PDO.
- The unit of 0x607F is a pulse unit, not a rotation velocity unit.

- Stop Control in Torque Mode
  In torque mode, the MC_Stop instruction can be executed to stop the drive. Upon receiving the stop instruction, the drive switches to the synchronous position mode and decelerates according to the deceleration specified in the stop instruction.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

Error 9113 is reported when the MC_TorqueControl instruction is called after the imaginary axis mode is enabled.

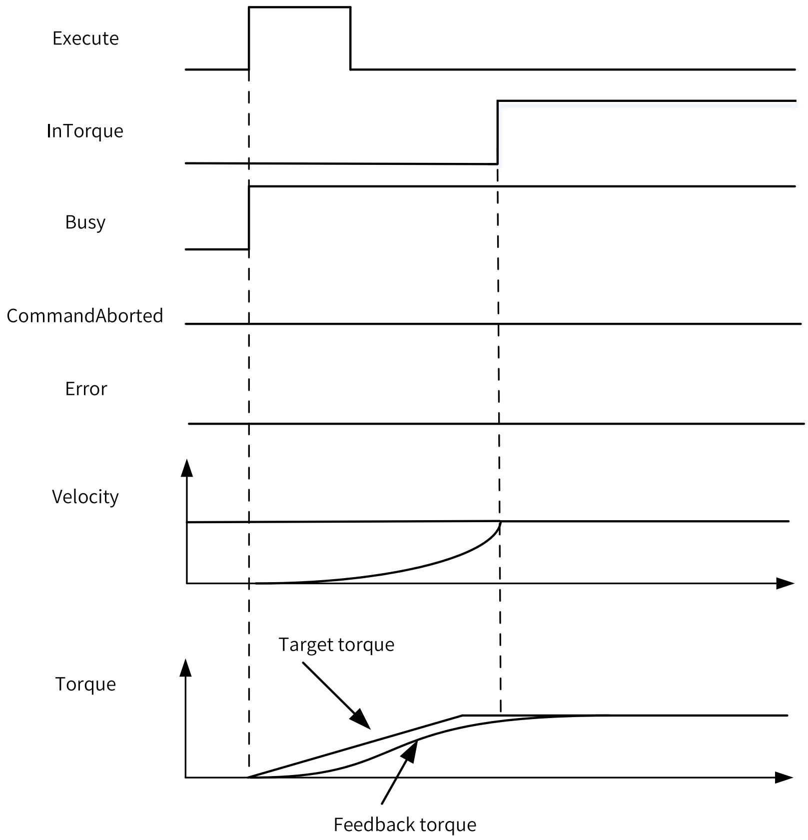
A PDO configuration fault is reported when the required PDO is not configured.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

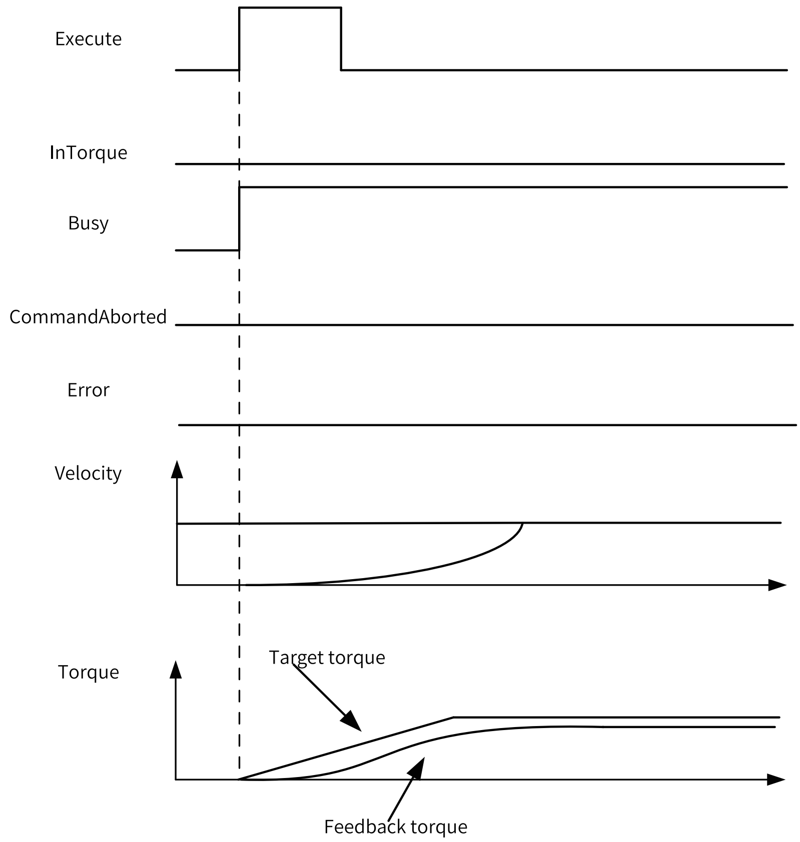
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

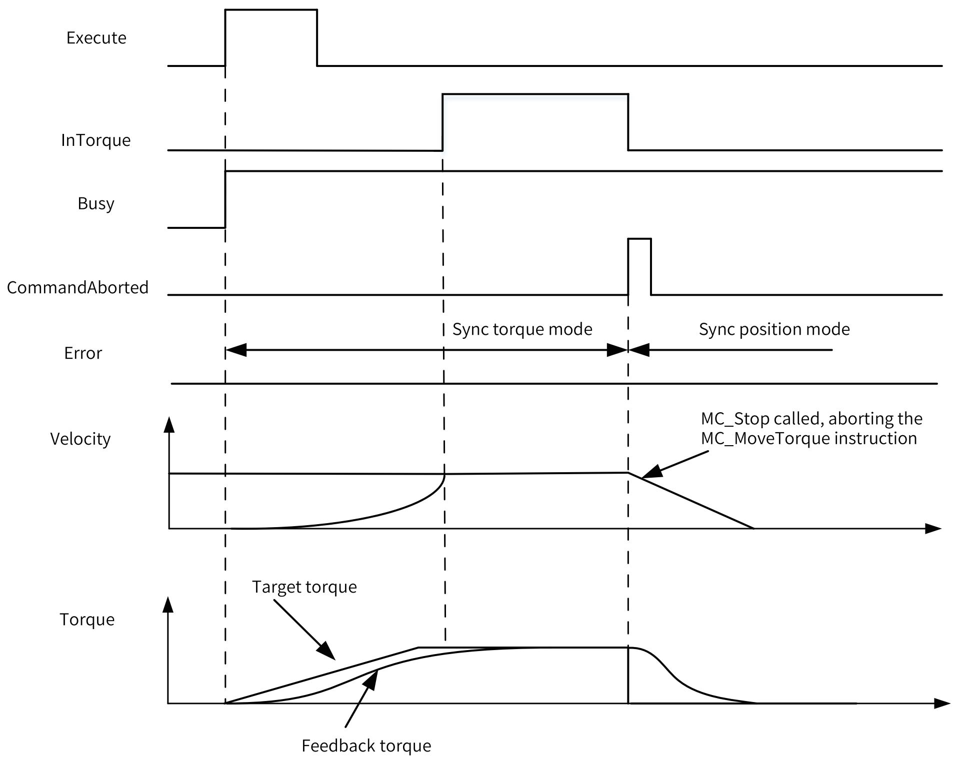
- The instruction is triggered after the target torque is specified, and the actual output torque can reach the target torque.



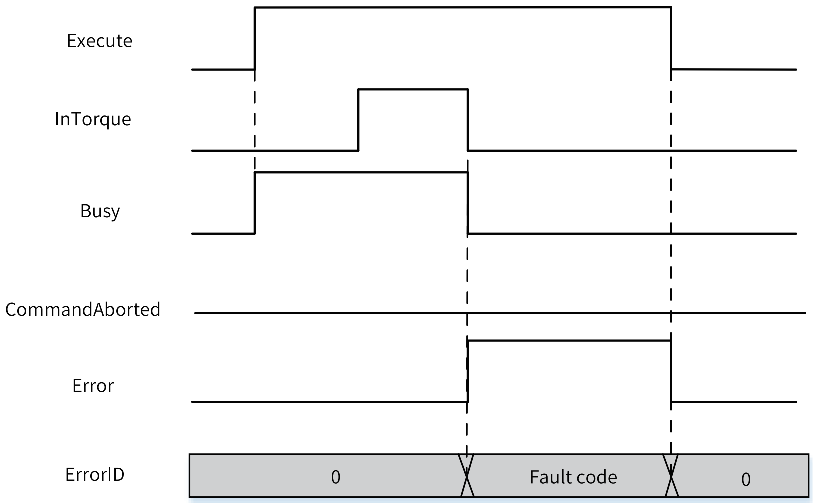
- The instruction is triggered after the target torque is specified, and the actual output torque cannot reach the target torque.



- The MC_Stop instruction aborts the instruction execution during torque operation.



- The drive reports an error during torque operation.



3.10.1.18 MC_Home

MC_Home – Homing

Graphic Block

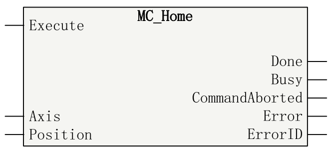
Instruction	Name	LD Expression	LiteST Expression
MC_Home	Homing		MC_Home(Execute := ???, Axis := ???, Position := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–211 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_Home: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Position	Home offset	Yes	0	Positive number, negative number, or 0	REAL
D1	Done	Homing completed	Yes	OFF	OFF/ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF/ON	BOOL

D3	CommandA-borted	Abortion of execution	Yes	OFF	OFF/ON	BOOL
D4	Error	Error flag	Yes	OFF	OFF/ON	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–212 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the homing function of the EtherCAT bus axis and the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description
  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  The function block latches the input parameter Position on the rising edge of the instruction. The axis enters the Homing state and performs homing.

  Position specifies the home offset.

  When this instruction is called in imaginary axis mode, homing is performed according to method 35 in CiA402.
- Multi-execution
  The homing instruction does not support multi-execution. After an MC_Home instruction is executed to perform homing, if another MC_Home instruction is called, the instruction called later reports an error.

Note

The MC_Home instruction does not support the synchronous motion mode. When the master axis is homing, the position type of the synchronization instruction is set to the instruction position, and the slave axis does not perform synchronous motion.

Abortion

When this instruction is active, the axis is in Homing state in PLCOpen. This instruction can be aborted by MC_Stop and MC_ImmediateStop, which can make the axis enter Stopping state. CommandAborted is active when this instruction is aborted.

If the MC_Power instruction is inactive when Enable is ON and the Done signal is active, the axis is disabled and CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

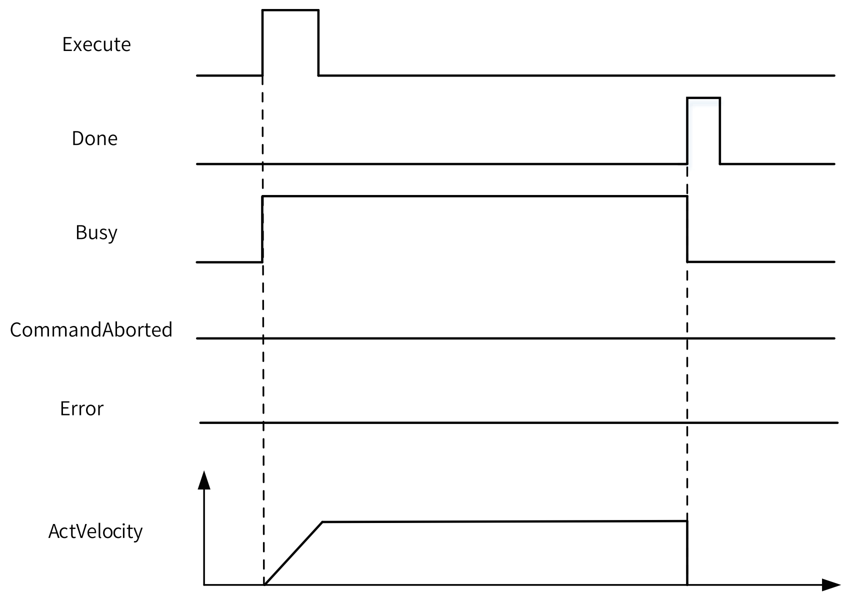
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is not in StandStill state.

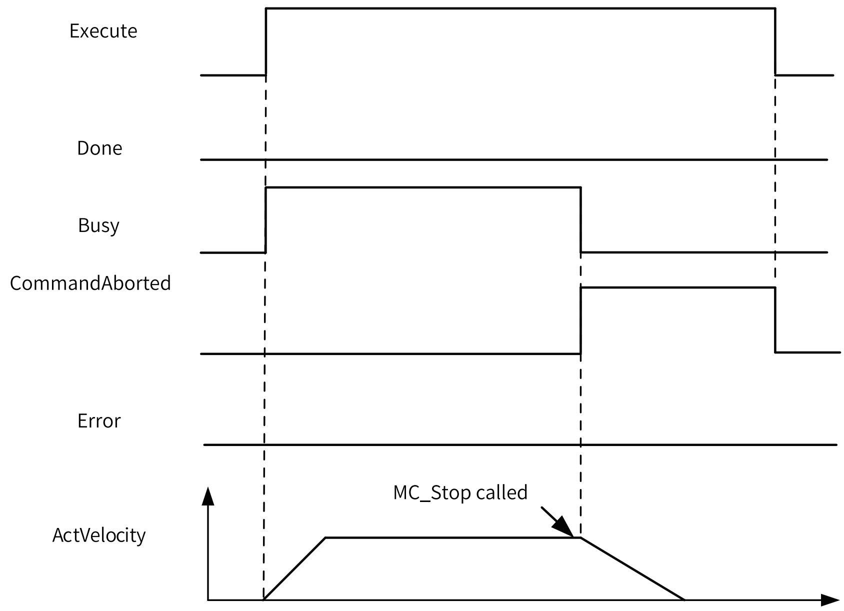
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

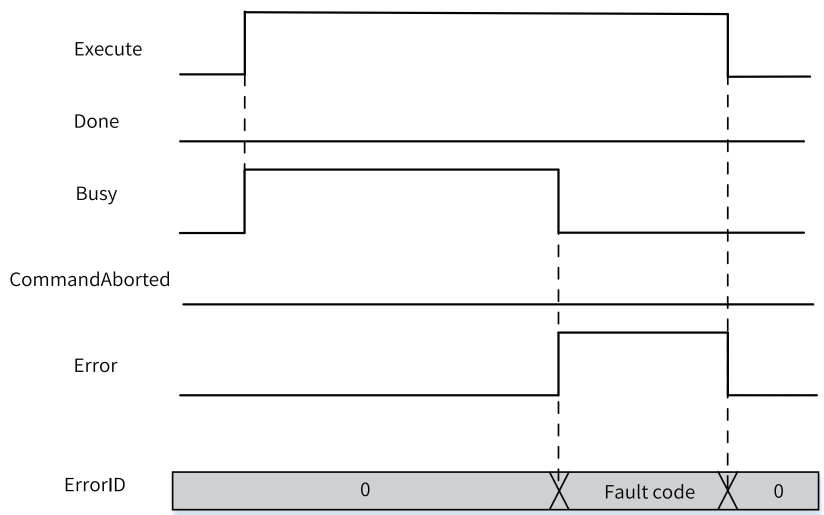
- The drive performs homing properly.



- Homing is aborted by the MC_Stop instruction.



- The drive fails during the homing process.



3.10.1.19 MC_Stop

MC_Stop – Stop

Graphic Block

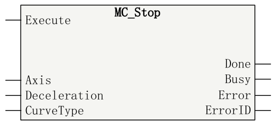
Instruction	Name	LD Expression	LiteST Expression
MC_Stop	Stop		MC_Stop(Execute := ???, Axis := ???, Deceleration := ???, CurveType := , Done => , Busy => , Error => , ErrorID =>);

Table 3–213 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_Stop: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
S2	Deceleration	Deceleration	No	-	Positive number, less than the maximum acceleration	REAL
S3	CurveType	Curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	OFF	OFF/ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF/ON	BOOL
D3	Error	Error flag	Yes	OFF	OFF/ON	BOOL
D4	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–214 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	-	√	-
S3	-	-	-	√	√	-	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the stop function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  The function block latches the input parameters such as Deceleration and CurveType on the rising edge of the Execute input. The axis enters the Stopping state and performs deceleration.

  After deceleration is completed, the Done signal becomes active, and the axis remains in the Stopping state when Execute is ON.

  When Execute turns OFF and Done is ON, the axis switches from the Stopping state to the StandStill state.

  The stop mode varies according to the running state of the axis when this instruction is executed.

  1. If the axis is executing the positioning instruction or running continuously when this instruction is called, CurveType specifies the type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis decelerates based on the value of Deceleration. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Deceleration indicates the maximum deceleration of the axis during deceleration.
  2. When the axis is in Homing state, this instruction triggers the Halt flag of the control word of the drive, and the drive decelerates according to the preset parameters. CurveType and Deceleration are invalid.

Re-execution

The same stop instruction can be executed repeatedly. If the same stop instruction is re-triggered during deceleration, the drive decelerates to stop according to the deceleration specified by the instruction triggered last.

Multi-execution

The MC_Stop instruction does not support multi-execution. If a stop instruction is called while another stop instruction is still active, it reports a fault.

Abortion

When this instruction is active, the axis is in Stopping state, and the instruction cannot be aborted by other motion instructions. When this instruction becomes inactive, the axis switches from the Stopping state to the StandStill state, and other motion control instructions can run.

This instruction can be aborted by the MC_ImmediateStop instruction. If the MC_ImmediateStop instruction is called while MC_Stop is active, the MC_Stop instruction reports an error.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

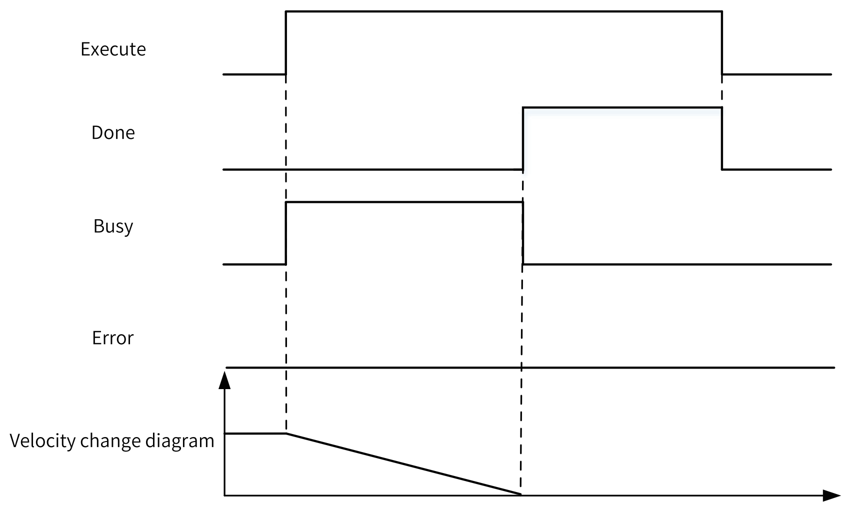
Error 9108 is reported if this instruction is executed when the axis is in Disabled or ErrorStop state.

Error 9142 is reported if this instruction is executed after the MC_ImmediateStop instruction is executed to put the axis in Stopping state.

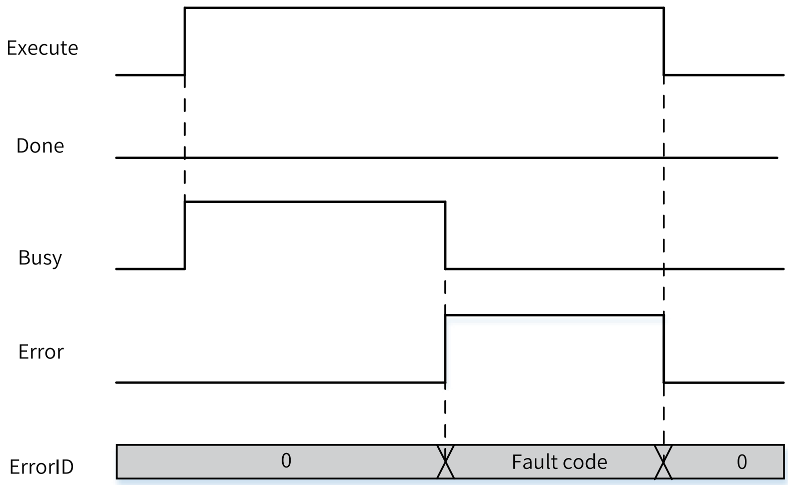
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

- The MC_Stop instruction is executed after the MC_MoveVelocity instruction.



- The drive fails during instruction execution.



3.10.1.20 MC_Halt

MC_Halt – Halt

Graphic Block

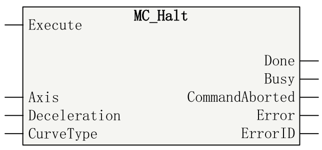
Instruction	Name	LD Expression	LiteST Expression
MC_Halt	Halt (Not recoverable)		MC_Halt(Execute := ???, Axis := ???, Deceleration := ???, CurveType := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–215 Instruction format

16-bit Instruction	-					
32-Bit Instruction	MC_Halt: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
S2	Deceleration	Deceleration	No	-	Positive number, less than the maximum acceleration	REAL

S3	CurveType	Curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	OFF	OFF/ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF/ON	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	OFF/ON	BOOL
D4	Error	Error flag	Yes	OFF	OFF ON	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

Table 3–216 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the halt function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block latches input parameters such as Deceleration and CurveType, and the axis performs deceleration. This instruction can be aborted by other instructions.

  CurveType specifies the type of the velocity curve.

  1. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration.

2. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.

Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion or ContinuousMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

- When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.
- When Execute is ON and the Done signal is inactive, if the MC_Power instruction is inactive, which causes the axis to be disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in Disabled, ErrorStop, or Homing state.

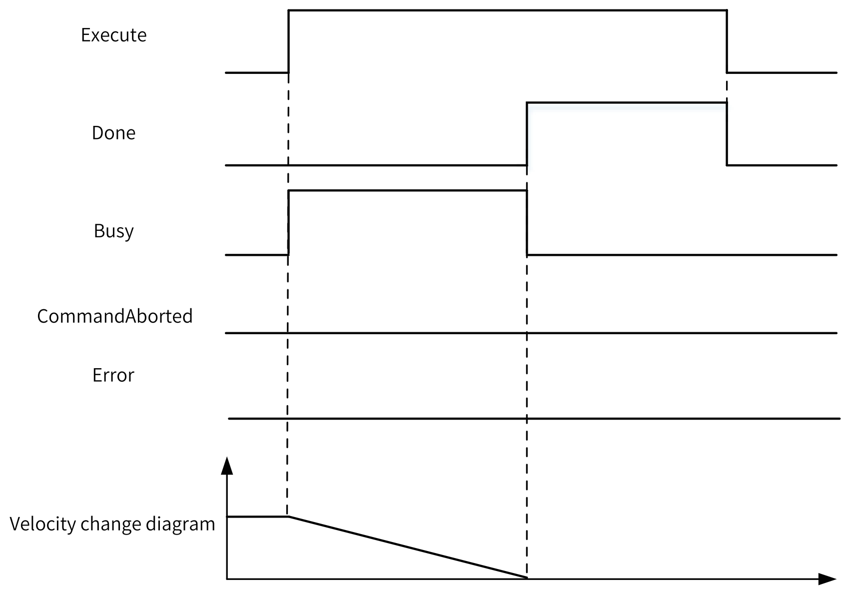
Error 9115 is reported if this instruction is executed after the MC_Stop instruction is executed to put the axis in Stopping state.

Error 9142 is reported if this instruction is executed after the MC_ImmediateStop instruction is executed to put the axis in Stopping state.

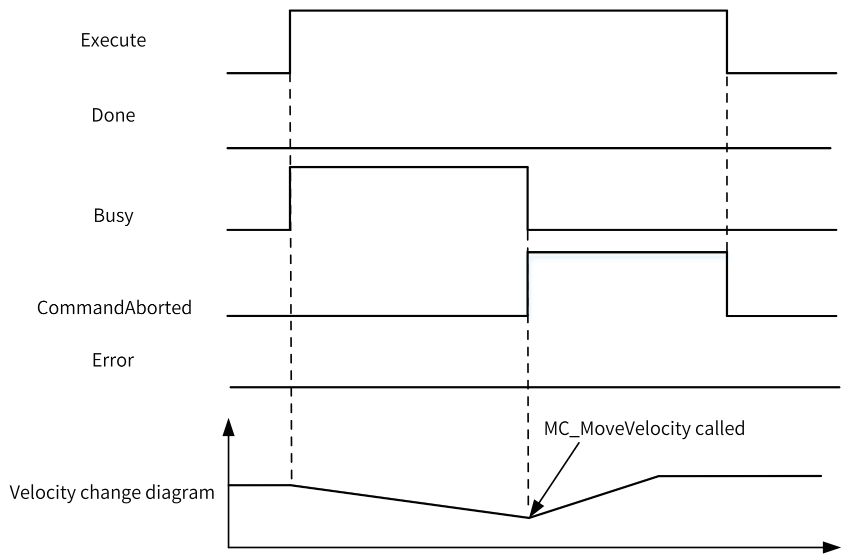
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

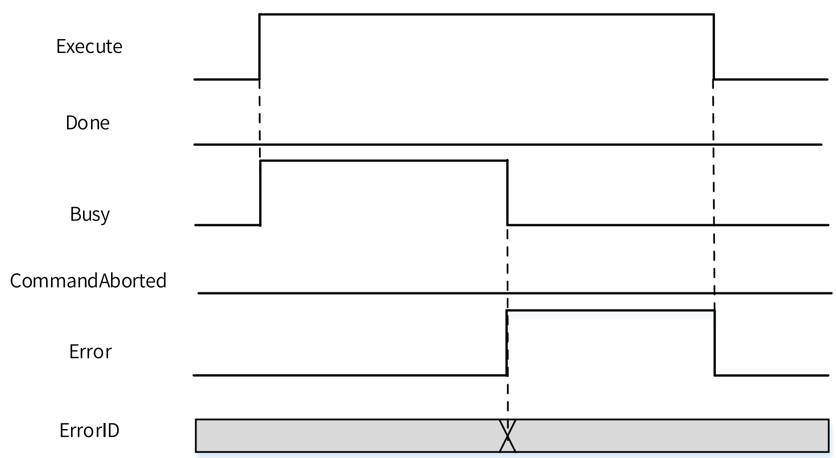
- After the positioning instruction is called, the MC_Halt instruction is triggered.



- After the MC_Halt instruction is triggered, the velocity instruction is called to abort the execution of the MC_Halt instruction.



- The drive stops upon a fault during execution of the MC_Halt instruction.



3.10.1.21 MC_MoveFeed

MC_MoveFeed – Interrupt positioning

Graphic Block

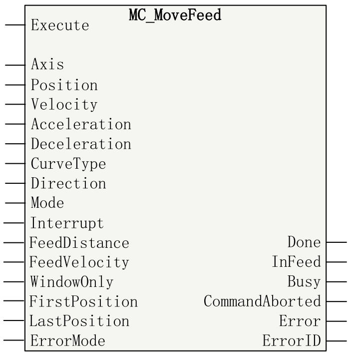
Instruction	Name	LD Expression	LiteST Expression
MC_MoveFeed	Interrupt positioning		MC_MoveFeed(Execute := ???, Axis := ???, Position := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Direction := , Mode := , Interrupt := , FeedDistance := ???, FeedVelocity := , WindowOnly := , FirstPosition := , LastPosition := , ErrorMode := , Done => , InFeed => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–217 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveFeed: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
S2	Position	Target position	No	-	Positive number, negative number, or 0	REAL

S3	Velocity	Target velocity	No	-	Positive number, less than the maximum velocity	REAL
S4	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S5	Deceleration	Deceleration	Yes	Same as acceleration	Positive number, less than the minimum acceleration	REAL
S6	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve	Yes	0	0 to 1	INT
S7	Direction	Motion direction of absolute positioning in ring mode 0: Forward (Target velocity > 0) 1: Reverse (Target velocity < 0) 2: Minimum distance 3: Current direction	Yes	0	0 to 3	INT
S8	Mode	Mode 0: Absolute positioning mode 1: Relative positioning mode 2: Speed mode	Yes	0	0 to 2	INT
S9	Interrupt	Interrupt source 0: Probe 1 1: Probe 2	Yes	0	0 to 1	INT

S10	FeedDistance	Travel distance after the interrupt feed input Positive: Feed in the same direction as the axis was moving before the interrupt input for the distance specified by FeedDistance. Negative: Feed in the opposite direction as the axis was moving before the interrupt input for the distance specified by FeedDistance.	No	-	Positive number, negative number, or 0	REAL
S11	FeedVelocity	Target velocity after the interrupt feed input	Yes	Same as Velocity	Positive number, less than the maximum velocity	REAL
S12	WindowOnly	Interrupt source window enable 0: Disabled 1: Enabled	Yes	OFF	OFF/ON	BOOL
S13	FirstPosition	Start position of the interrupt source window	Yes	0	Positive number, negative number, or 0	REAL
S14	LastPosition	End position of the interrupt source window	Yes	0	> FirstPosition	REAL
S15	ErrorMode	Fault mode OFF: After the position specified by Position is reached, if no interrupt signal is detected, the Done signal is set to ON, and the instruction does not report a fault. ON: After the position specified by Position is reached, if no interrupt signal is detected, the Error signal is set to ON, and the instruction reports a fault.	Yes	OFF	OFF/ON	BOOL

D1	Done	Target position reached	Yes	OFF	-	BOOL
D2	InFeed	Interrupt signal active	Yes	OFF	-	BOOL
D3	Busy	Busy flag	Yes	OFF		BOOL
D4	CmdAborted	Abortion of execution	Yes	OFF	-	BOOL
D5	Error	Error flag	Yes	OFF	-	BOOL
D6	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–218 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	-	√	-
S3	-	-	-	√	√	-	-	√	-
S4	-	-	-	√	√	-	-	√	-
S5	-	-	-	√	√	-	-	√	-
S6	-	-	-	√	√	-	√	-	-
S7	-	-	-	√	√	-	√	-	-
S8	-	-	-	√	√	-	√	-	-
S9	-	-	-	√	√	-	√	-	-
S10	-	-	-	√	√	-	-	√	-
S11	-	-	-	√	√	-	-	√	-
S12	√	√	√	-	-	-	-	-	-
S13	-	-	-	√	√	-	√	-	-
S14	-	-	-	√	√	-	√	-	-
S15	√	√	√	-	-	-	-	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	√[1]	√	√	-	-	-	-	-	-
D6	-	-	-	√	√	-	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the interrupt positioning function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge, and it does not support the imaginary axis mode.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.

- If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
- If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.

- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block latches input parameters such as Position, Velocity, Direction, Acceleration, and Deceleration.

  Before the interrupt arrives, the axis performs absolute positioning (Mode = 0), relative positioning (Mode = 1), or continuous motion (Mode = 2) based on parameters including Position, Velocity, Direction, and Mode. After an interrupt signal is generated by the interrupt source specified by Interrupt, the axis performs relative movement at the position where the interrupt arrives based on FeedDistance and FeedVelocity.

  - Position: Target position of the axis before the interrupt arrives when Mode is set to 0 (absolute positioning) or 1 (relative positioning).
  - Velocity: Target velocity of the axis before the interrupt arrives.
  - CurveType: Type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.
  - Direction: Direction of rotation, which is the same as Direction of the MC_MoveAbsolute instruction. It specifies the axis rotation direction in ring mode when Mode is set to 0 (absolute positioning). The value 0 indicates forward, 1 indicates reverse, 2 indicates the shortest path, and 3 indicates the current direction.
  - Mode: Movement mode before the interrupt arrives. When Mode is set to 0, the axis performs absolute positioning before the interrupt feed input; when Mode is set to 1, the axis performs relative positioning before the interrupt feed input; when Mode is set to 2, the axis performs continuous motion before the interrupt feed input.
  - Interrupt: Interrupt source. When it is set to 0, the interrupt source is probe 1, and the interrupt is active on the rising edge of probe 1. When it is set to 1, the interrupt source is probe 2, and the interrupt is active on the rising edge of probe 2.
  - FeedDistance: Target travel distance after the interrupt feed input. If the value is positive, the axis runs in the current direction for the distance specified by FeedDistance when the interrupt signal arrives. If the value is negative, the axis runs in the opposite direction for the distance specified by FeedDistance when the interrupt signal arrives.
  - FeedVelocity: Target velocity after the interrupt feed input.
  - ErrorMode: Fault handling mode when there is no interrupt. In absolute or relative positioning mode, when no interrupt signal is detected after the position (travel distance) specified by Position is reached, the instruction reports a fault if ErrorMode is set to ON and does not report a fault if ErrorMode is set to OFF.
  - InFeed: The InFeed output becomes active after the interrupt signal arrives.

Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

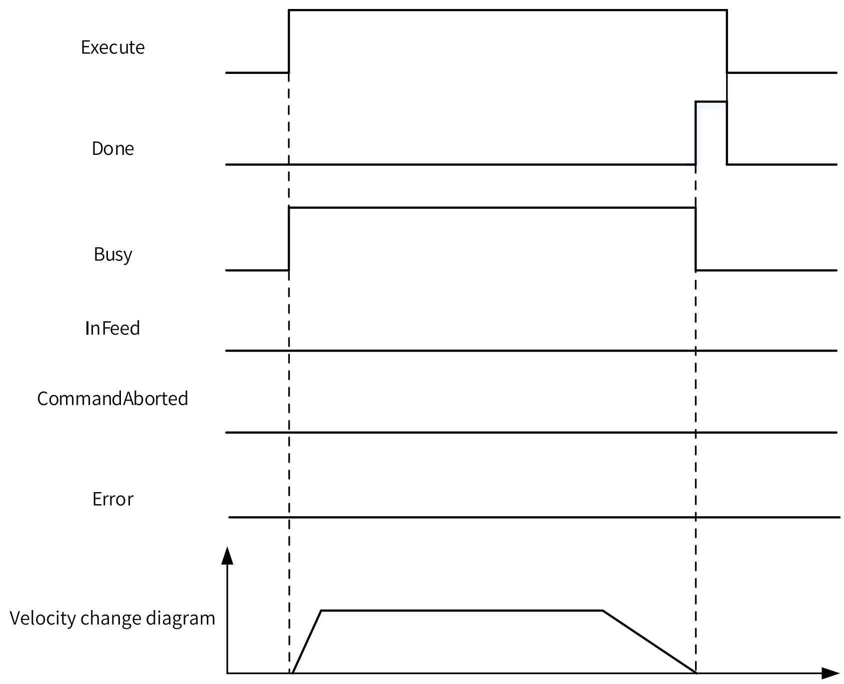
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

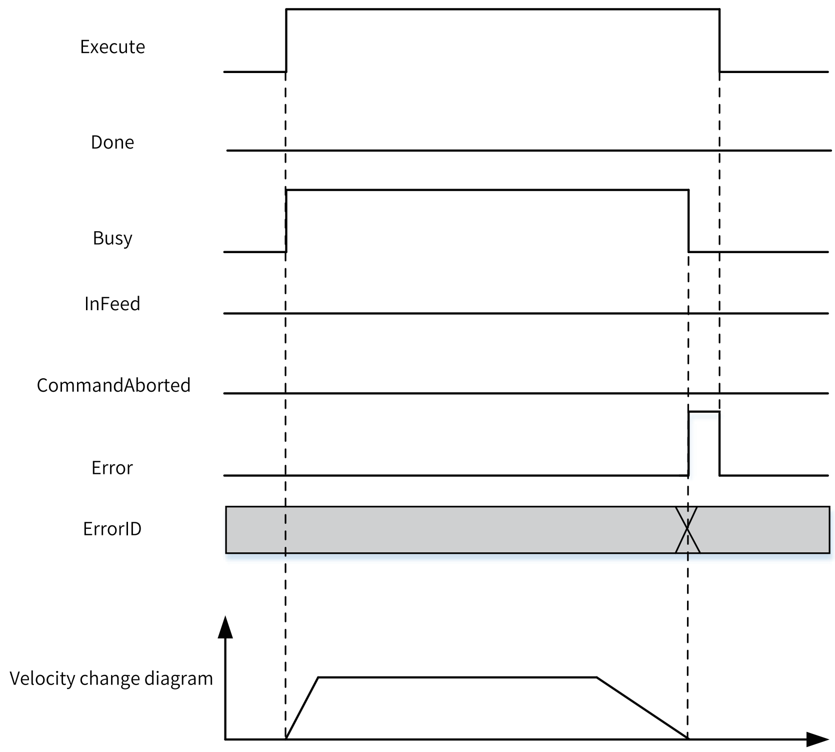
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9133 is reported when the imaginary axis mode is enabled on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

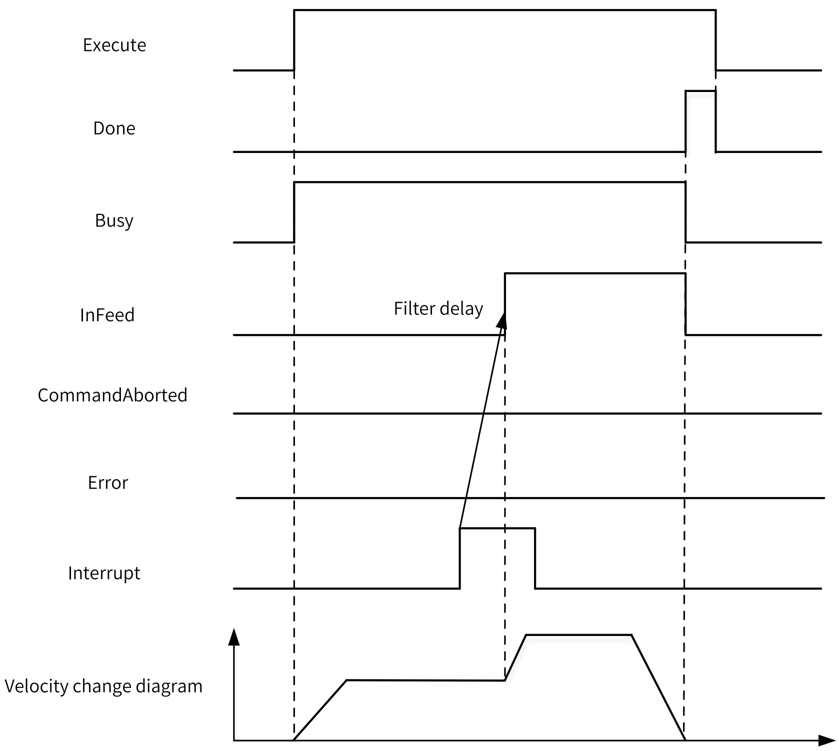
- The relative or absolute positioning mode is selected, the interrupt signal is not triggered, and ErrorMode is set to OFF.



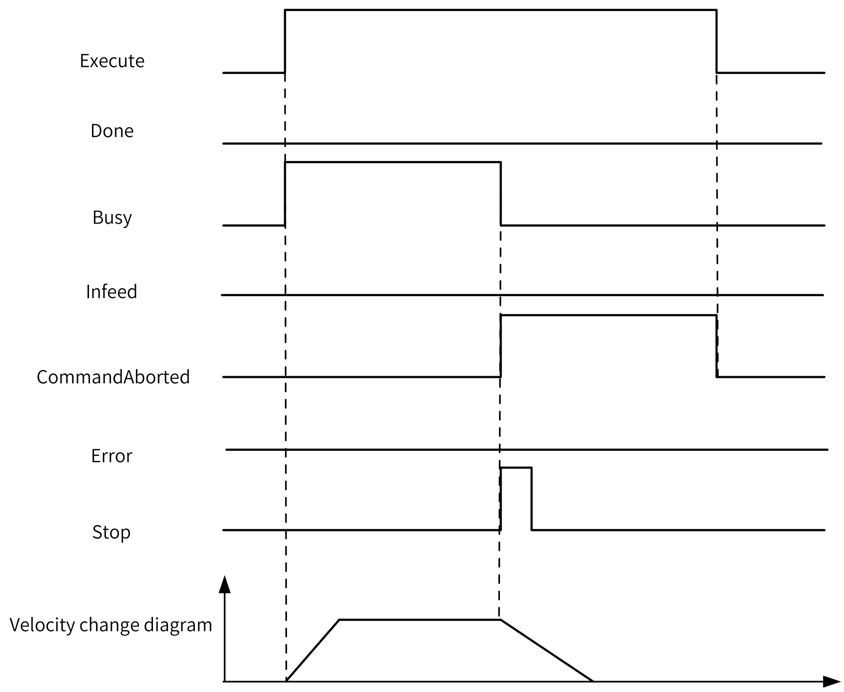
- The relative or absolute positioning mode is selected, the interrupt signal is not triggered, and ErrorMode is set to ON.



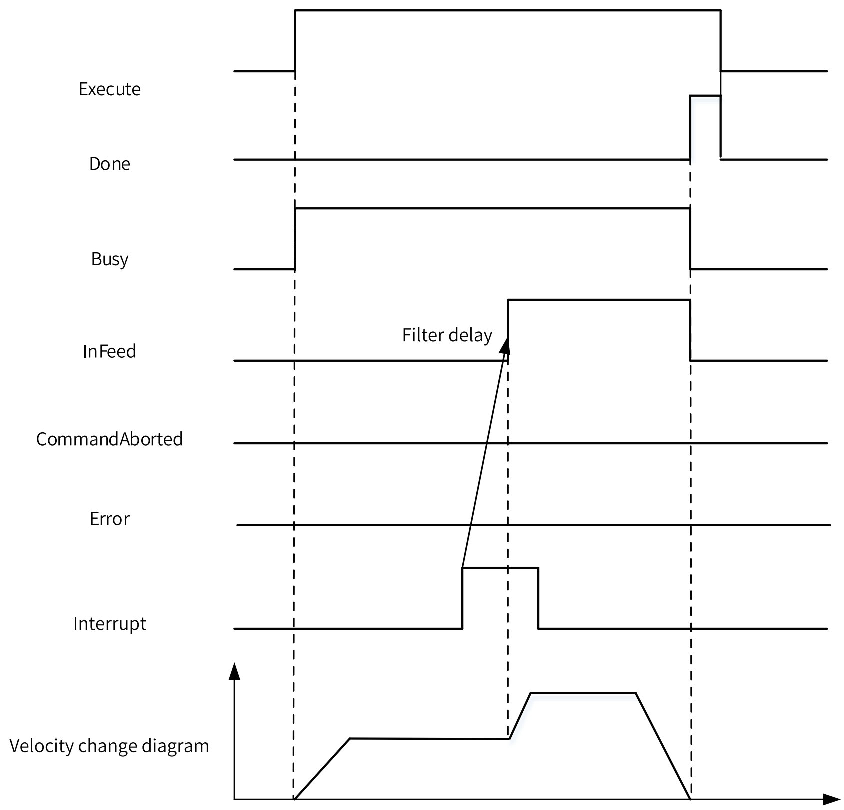
- The relative or absolute positioning mode is selected and the interrupt signal is triggered.



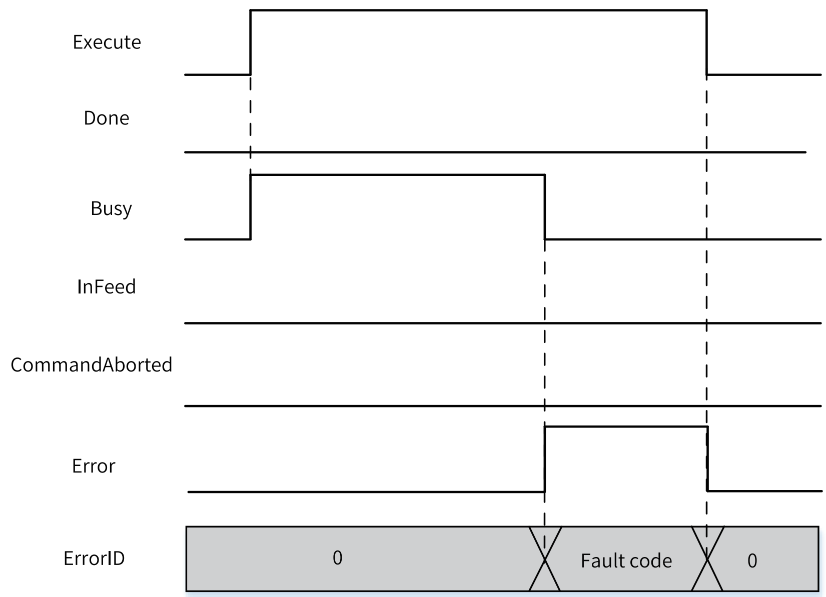
- The velocity mode is selected, the interrupt is not triggered, and the MC_Stop instruction is called to abort this instruction after it is executed for a period of time.



- The velocity mode is selected and the interrupt is triggered.



- A fault occurs during instruction execution.



3.10.1.22 MC_MoveBuffer

MC_MoveBuffer – Multi-position positioning

Graphic Block

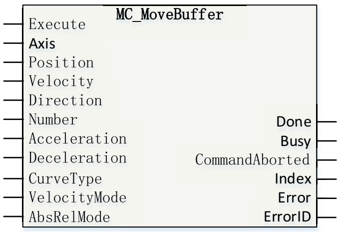
Instruction	Name	LD Expression	LiteST Expression
MC_MoveBuffer	Multi-position		MC_MoveBuffer(Execute := ???, Axis := ???, Position := ???, Velocity := ???, Direction := , Number := ???, Acceleration := ???, Deceleration := , CurveType := , VelocityMode := , AbsRelMode := , Done => , Busy => , CommandAborted => , Index => , Error => , ErrorID =>);

Table 3–219 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveBuffer: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_ INFO
S2	Position	Start address of the target position	No	-	Positive number Negative number 0	FLT32, array*16
S3	Velocity	Start address of the target velocity	No	-	Positive number	FLT32, array*16

S4	Direction	Start address of absolute positioning direction in ring mode 0: Forward (Target velocity > 0) 1: Reverse (Target velocity < 0) 2: Minimum distance 3: Current direction	Yes	0	0 to 3	INT, array*16
S5	Number	Number of buffer pairs	No	-	1 to 16	INT
S6	Acceleration	Acceleration	No	-	-	FLT32
S7	Deceleration	Deceleration	Yes	Acceleration	-	FLT32
S8	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve Others: T-shaped velocity curve	Yes	0	-	INT
S9	VelocityMode	Velocity switching mode 0: Decelerate to 0 and start the next segment 1: Keep the current velocity and start the next segment	Yes	0	0 to 1	INT
S10	AbsRelMode	Positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
D1	Done	Stop completed	Yes	OFF	-	BOOL
D2	Busy	Busy flag	Yes	OFF	-	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	-	BOOL
D4	Index	Current segment	Yes	0	0 to 15	INT
D5	Error	Error flag	Yes	OFF	-	BOOL
D6	ErrorID	Fault Code	Yes	0	*1	INT

Table 3–220 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	-	-	-
S4	-	-	-	√	√	√	-	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	-	√	-
S7	-	-	-	√	√	√	-	√	-
S8	-	-	-	√	√	√	-	√	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S9	-	-	-	√	√	√	√	-	-
S10	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	√[1]	√	√			√	-	-	-
D6	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the multi-position positioning function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the Execute input, the function block latches input parameters such as Position, Velocity, Direction, Number, Acceleration, and Deceleration.

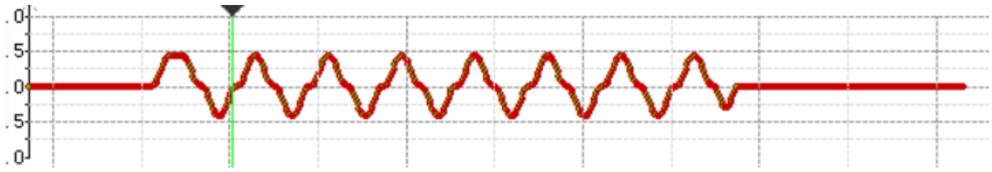
  The axis performs absolute positioning (AbsRelMode = 0) or relative positioning (AbsRelMode = 1) in buffer mode based on the value of AbsRelMode. This instruction supports up to 16-segment positions.

  - Position: Target position, array type, up to 16 segments. It specifies the target absolute position of the axis in absolute positioning mode or target travel distance of the axis in relative positioning mode.
  - Velocity: Target velocity, array type, up to 16 segments.
  - Direction: Target direction of absolute positioning in ring mode, same as Direction of the MC_ MoveAbsolute instruction.
  - Number: Number of groups of target position, target velocity, and direction to be buffered. It ranges from 1 to 16. A fault is reported if the value is out of range.
  - CurveType: Type of the velocity curve. If CurveType is set to 0, the T-shaped curve is used. In this case, the axis accelerates or decelerates based on the value of Acceleration or Deceleration. If CurveType is set to 1, the 5-segment S-curve is used. In this case, Acceleration and Deceleration indicate the maximum acceleration and minimum deceleration of the axis during acceleration and deceleration.

- VelocityMode: Velocity switching mode. When it is set to 0, the axis decelerates to 0 before reaching a target position and then starts to run from 0 velocity to the next target position; when it is set to 1, the axis runs to a target position at the current target velocity and then switches to a new velocity based on the acceleration (deceleration) to move to the next target position.

Note

A special situation may arise during absolute positioning when the velocity is retained. Assume that 3 position segments are set. The target position of segment 1 is 10, that of segment 2 is 10.1, and that of segment 3 is 10.2. The target velocity is 100, and the feedback velocity is also 100. The EtherCAT task cycle is 8 ms, and the increment of the target travel distance per EtherCAT cycle is 0.8. During segment 1, if the current position is 9.9 when an EtherCAT cycle starts, the current position changes to 10.7 when the next EtherCAT cycle starts, which has exceeded the target position of segment 2. In this case, the axis needs to decelerate and run in the reverse direction, and this may produce the velocity curve as shown in the following figure:



Abortion

When this instruction is active, the axis is in DiscreteMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

Errors

Error 9101 is reported when the axis number does not exist or the axis type does not match.

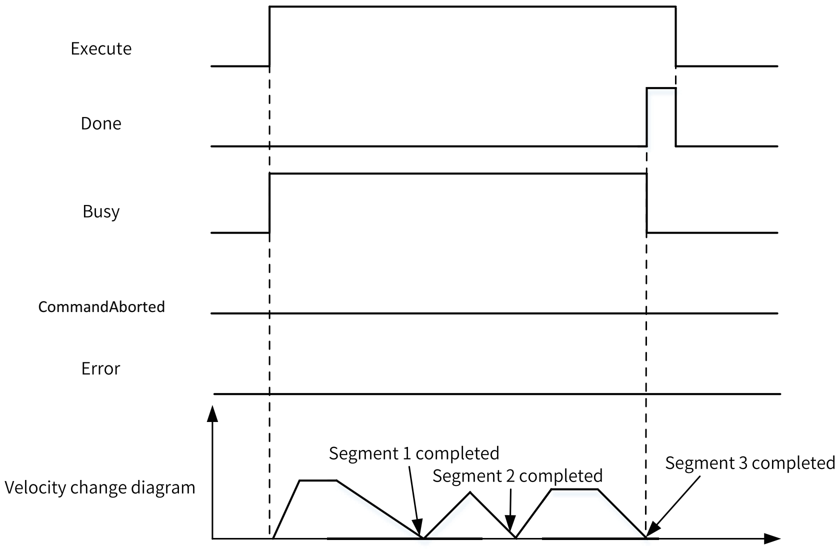
Error 9102 is reported when axis initialization fails.

Error 9108 is reported if this instruction is executed when the axis is in a state other than StandStill, DiscreteMotion, and ContinuousMotion.

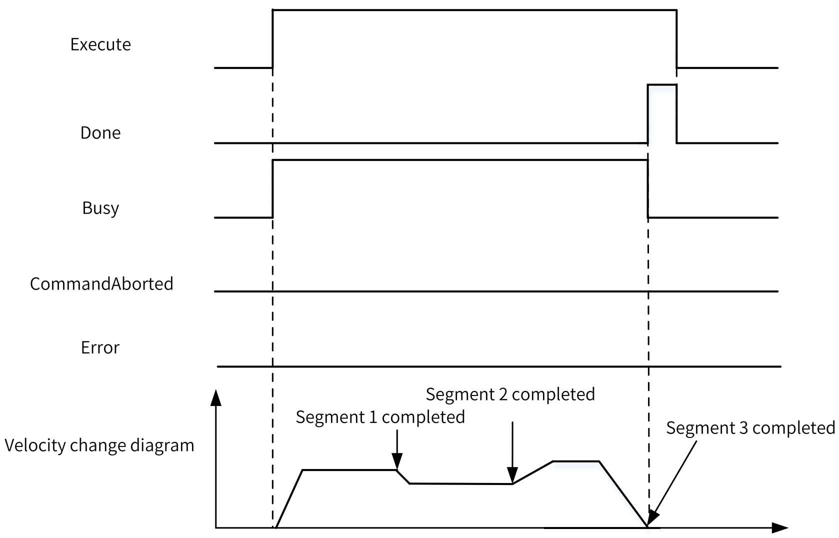
- Error 9116 is reported if the axis is in online commissioning state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- A fault is reported when the parameters on the left of the instruction are out of range or improper on the rising edge of the Execute input.
- Error 9116 is reported if the axis enters the commissioning state when Execute is ON and the Done signal is inactive.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

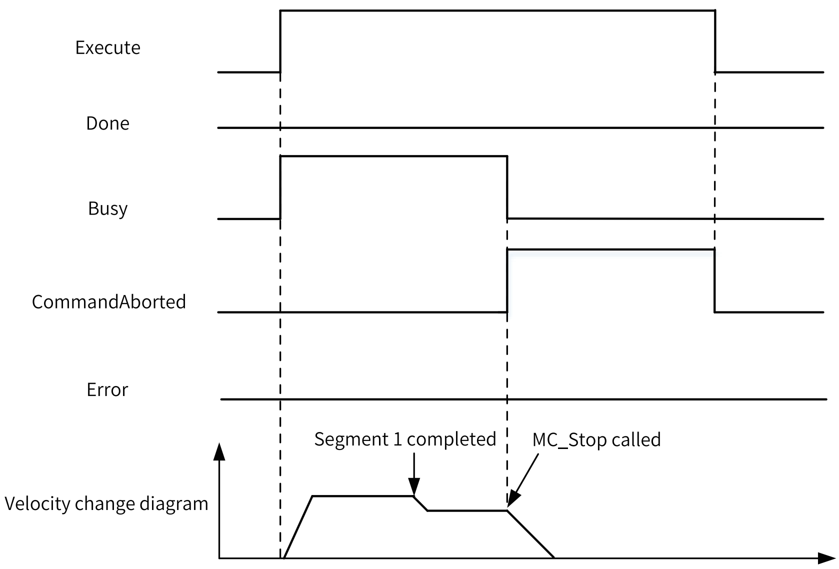
- In 3-segment buffer mode, VelocityMode is set to 0.



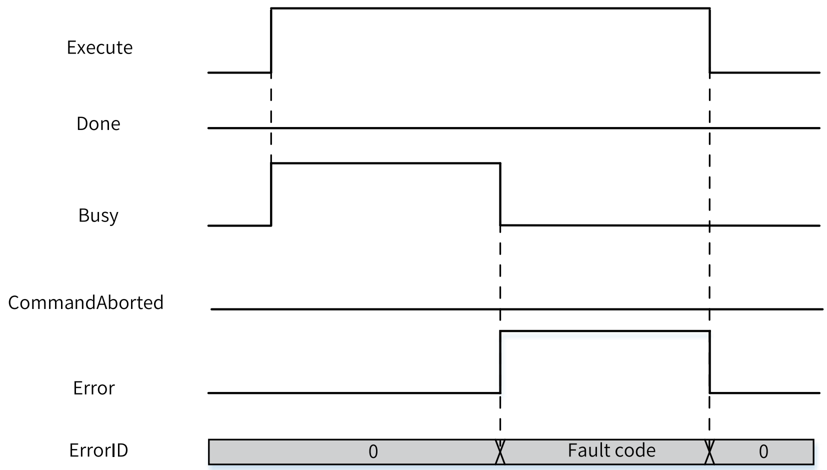
- In 3-segment buffer mode, VelocityMode is set to 1.



- In 3-segment buffer mode, instruction execution is aborted by the MC_Stop instruction.



- In 3-segment buffer mode, an error occurs during instruction execution.



3.10.1.23 MC_ImmediateStop

MC_ImmediateStop – Immediate stop

Graphic Block

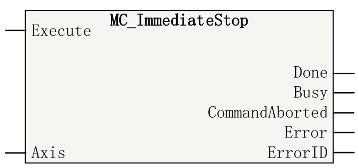
Instruction	Name	LD Expression	LiteST Expression
MC_ImmediateStop	Emergency stop		MC_ImmediateStop(Execute := ???, Axis := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–221 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ImmediateStop: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	0 to 32767	INT _sMCAXIS_INFO
D1	Done	Stop completed	Yes	OFF	OFF/ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF/ON	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	OFF/ON	BOOL
D4	Error	Error flag	Yes	OFF	OFF/ON	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

Table 3–222 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	-	-	-	-
D2	√[1]	√	√	-	-	-	-	-	-
D3	√[1]	√	√	-	-	-	-	-	-
D4	√[1]	√	√	-	-	-	-	-	-
D5	-	-	-	√	√		-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the immediate stop function of the EtherCAT bus axis or the local pulse axis. It is active on the rising edge.

- Specifying axis
  - Axis is latched on the rising edge of the Execute input.
  - If Axis specifies the axis name, modification on Axis is invalid when Execute is ON.
  - If Axis specifies the axis number, modification on Axis is valid when Execute is OFF.
- Function description

  This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

  On the rising edge of the instruction, the function block switches the PLCOpen state machine of the axis to the Stopping state, and switches the CiA 402 state machine of the drive to quick stop state. The drive stops running according to the stop mode specified in 0x605A. For the Inovance servo IS620N, the stop mode is described as follows:

- When the servo is in CSP mode:

<table>
<tr><th>Setpoint</th><th>Stop Mode</th></tr>
<tr><td>0</td><td>Coast to stop, keeping de-energized status
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>1</td><td rowspan="3">Stop at the emergency stop torque in 2007-10h, keeping the free-run state
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>2</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td>N/A</td></tr>
<tr><td>5</td><td rowspan="3">Stop at the emergency stop torque in 2007-10h, maintaining the locked position</td></tr>
<tr><td>6</td></tr>
<tr><td>7</td></tr>
</table>

- When the servo is in HM mode:

<table>
<tr><th>Setpoint</th><th>Stop Mode</th></tr>
<tr><td>0</td><td>Coast to stop, keeping de-energized status
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>1</td><td>Stop according to ramp in 6084h (HM: 609Ah), keeping the free-run state
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>2</td><td>Stop according to ramp in 6085h, keeping the free-run state
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>3</td><td>Stop at the emergency stop torque, keeping the free-run state
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>4</td><td>N/A</td></tr>
<tr><td>5</td><td>Stop according to ramp in 6084h (HM: 609Ah), maintaining the locked position</td></tr>
<tr><td>6</td><td>Stop according to ramp in 6085h, maintaining the locked position</td></tr>
<tr><td>7</td><td>Stop at the emergency stop torque in 2007-10h, maintaining the locked position</td></tr>
</table>

- When the servo is in CST mode:

<table>
<tr><th>Setpoint</th><th>Stop Mode</th></tr>
<tr><td>0</td><td>Coast to stop, keeping de-energized status
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>1</td><td rowspan="2">Stop according to ramp in 6087h, keeping the free-run state
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>2</td></tr>
<tr><td>3</td><td>Coast to stop, keeping de-energized status
If this mode is selected, the feedback velocity of the servo is not necessarily 0 after the Done signal output of the instruction becomes active.</td></tr>
<tr><td>4</td><td>N/A</td></tr>
<tr><td>5</td><td rowspan="2">Stop according to ramp in 6087h, maintaining the locked position</td></tr>
<tr><td>6</td></tr>
<tr><td>7</td><td>Coast to stop, maintaining the locked position</td></tr>
</table>

- Re-execution
  The same MC_ImmediateStop instruction can be executed repeatedly. If the same MC_ ImmediateStop instruction is re-triggered during deceleration, the drive stops according to the stop mode specified by the instruction triggered last.
- Multi-execution
  When multiple MC_ImmediateStop instructions are called, the instruction of which the rising edge is triggered first shall prevail. Other instructions report fault 9143 (repeatedly calling the immediate stop instruction).

When the axis is in the Stopping state, this instruction cannot be aborted by other motion instructions.

When the axis switches from the Stopping state to the StandStill state on the falling edge of the instruction flow, other motion control instructions can run.

This instruction takes priority over the MC_Stop instruction. If this instruction is called while the MC_ Stop instruction is still active, the MC_Stop instruction reports an error.

Abortion

The axis disable interrupt signal output is active when Execute is ON.

Errors

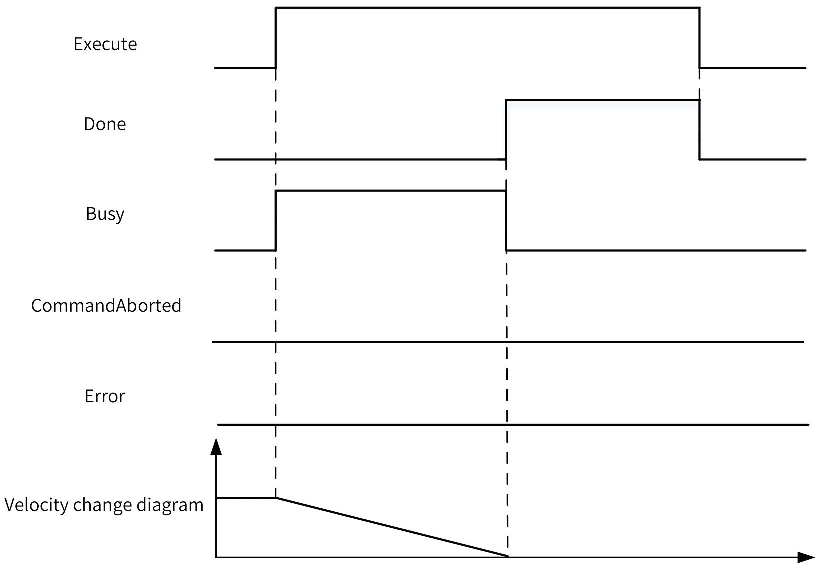
Error 9101 is reported when the axis number does not exist or the axis type does not match.

Error 9102 is reported when axis initialization fails.

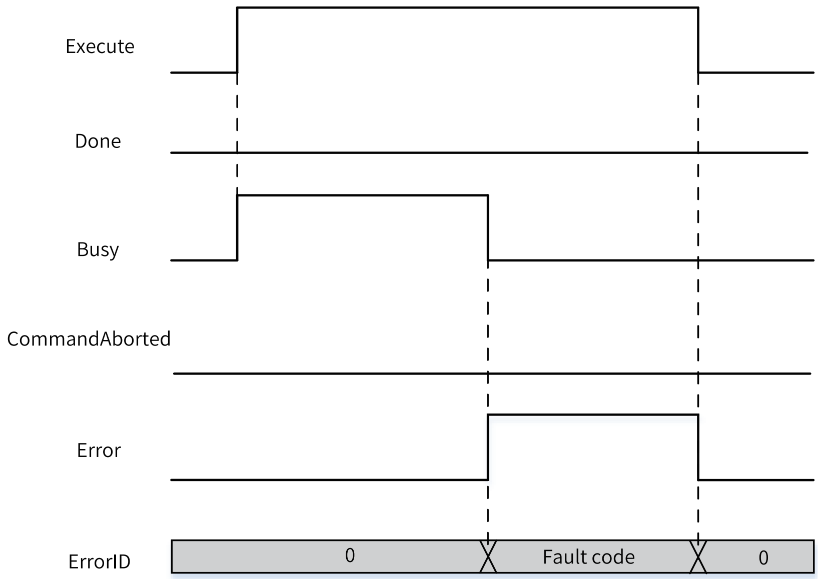
- Error 9108 is reported if the axis is in Disabled or ErrorStop state on the rising edge of the Execute input.
- Error 9106 is reported if the axis is decelerating in ErrorStop state on the rising edge of the Execute input.
- If the axis fails and enters the ErrorStop state when Execute is ON, the instruction displays the fault code of the axis in the ErrorStop state.

Timing Diagram

- The MC_Stop instruction is executed after the MC_MoveVelocity instruction.



- The drive fails during instruction execution.



3.10.1.24 MC_MoveSuperImposed

MC_MoveSuperImposed – Motion superimposition

Graphic Block

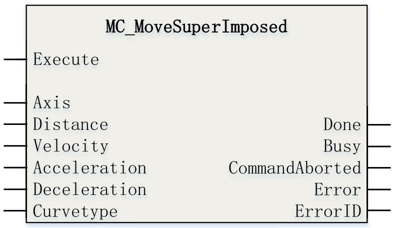
Instruction	Name	LD Expression	LiteST Expression
MC_ MoveSuperImposed	Motion super-impo-sition		MC_MoveSuperImposed(Execute := ???, Axis := ???, Distance := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–223 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveSuperImposed: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-		_sMCAXIS_ INFO
S2	Distance	Phase compensation	No	-	Positive number 0 Negative number	REAL
S3	Velocity	Target velocity	No	-	Positive number	REAL
S4	Acceleration	Acceleration	No	-	Positive number	REAL
S5	Deceleration	Deceleration	Yes	Acc	Positive number	REAL
S6	Curvetype	Curve type 0: T-shaped velocity curve	Yes	0	0	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL

D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	ON OFF	INT

Function Description

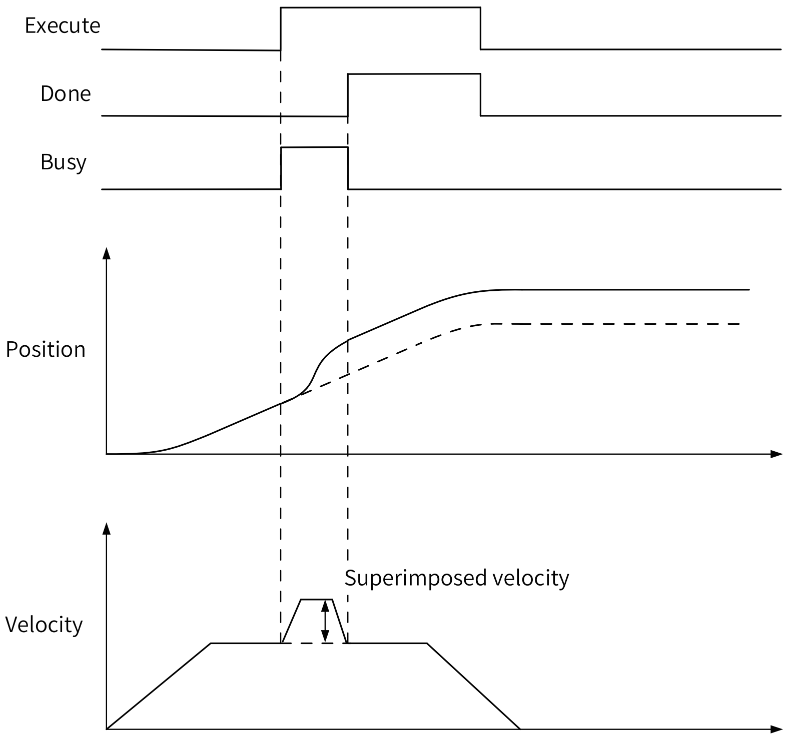
On the rising edge of the Execute input, this instruction superimposes a phase positioning based on the original control mode of the axis according to the input parameters. Distance specifies the travel distance of the superimposed motion, and Velocity specifies the velocity.

Instruction Execution Under Different Control Modes

- **Single-Axis Positioning Instructions**

  When called separately, this instruction controls the axis to perform relative positioning. The PLCOpen state machine switches from the StandStill state to the DiscreteMotion state.

  When called during execution of an instruction that can make the servo axis work in CSP mode, this instruction implements motion superimposition.



  If another instruction that can make the servo axis work in CSP mode, such as MC_MoveAbsolute, is triggered during execution of this instruction, this instruction is aborted.

  This instruction can be aborted by MC_Stop, MC_Halt, and MC_ImmediateStop.

  The motion superimposition instruction cannot be executed while the MC_Halt instruction is active.

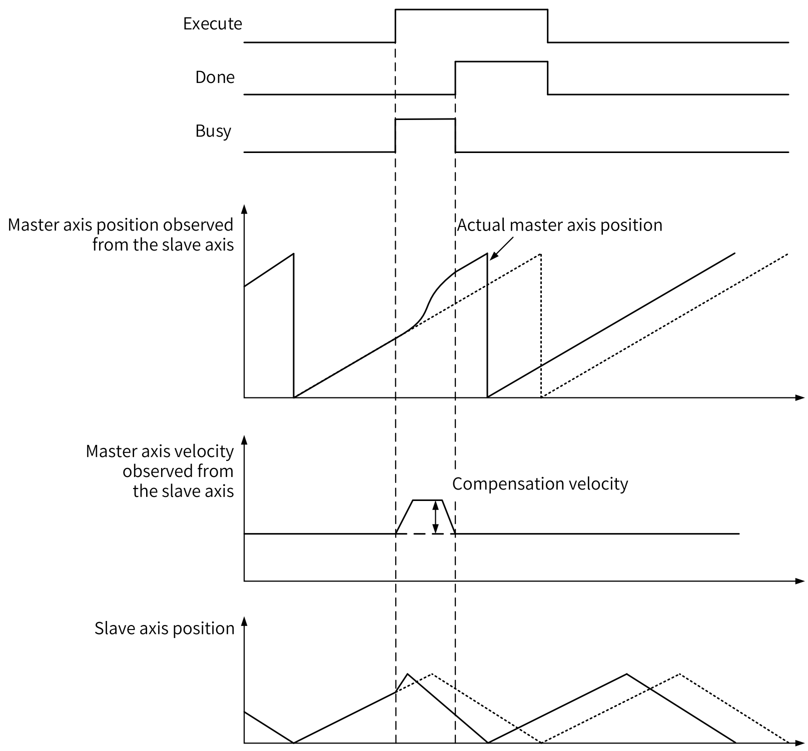
  If a non-CSP motion instruction such as MC_TorqueControl and MC_Home is executed during execution of this instruction, it reports an error.

- **Axis Group Instructions**

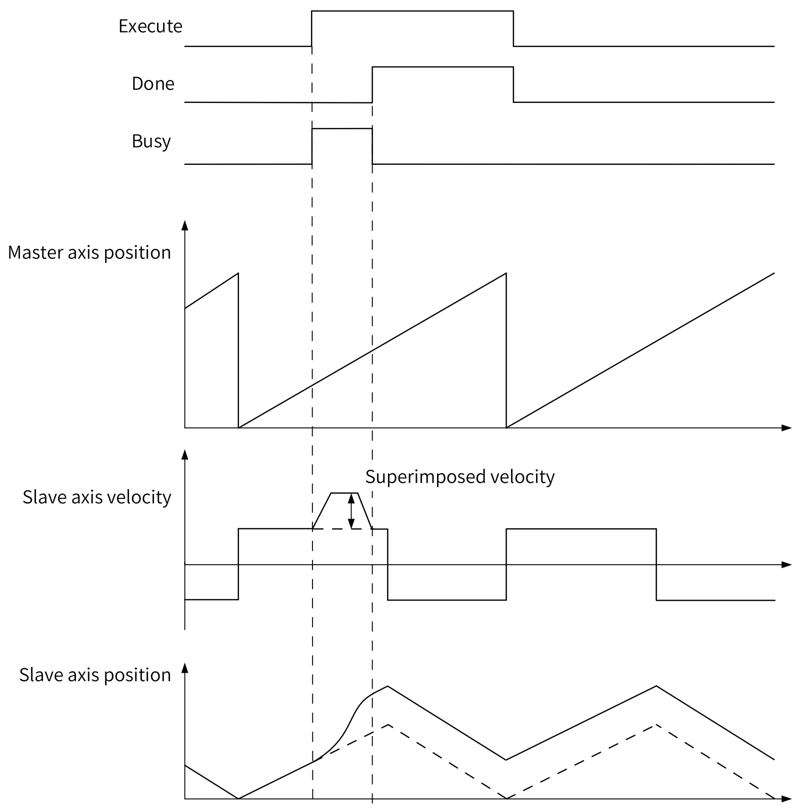
If this instruction is called during execution of an axis group instruction, it reports an error and does not implement motion superimposition.

- **Cam/Gear Instructions**

  For operations on the master axis, see the rules described in the single-axis and axis group instruction sections.



- When this instruction is called by the slave axis during gear and cam following, it performs motion superimposition.



This instruction is aborted if the gear instruction (MC_GearIn) is re-triggered during execution of this instruction.

When the cam instruction (MC_CamIn) is re-triggered during execution of this instruction, if the buffer mode is to switch immediately, this instruction is aborted; if the buffer mode is to wait for the completion of execution of the previous cam, execution of this instruction continues.

MC_GearOut and MC_CamOut can abort execution of this instruction.

3.10.1.25 MC_MoveVelocityCSV

MC_MoveVelocityCSV – CSV-based velocity control with adjustable pulse width

Graphic Block

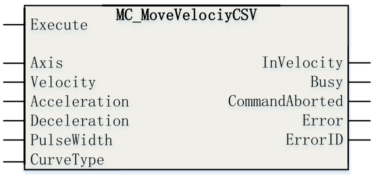
Instruction	Name	LD Expression	LiteST Expression
MC_ MoveVelocityCSV	CSV-based velocity control with adjustable pulse width		MC_MoveVelocityCSV(Execute := ???, Axis := ???, Velocity := ???, Acceleration := ???, Deceleration := , PulseWidth := , CurveType := , InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–224 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveVelocityCSV: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	-	_sMCAXIS_INFO
S2	Velocity	Target velocity	No	-	Positive number/ number/0, absolute value less than the maximum velocity	REAL
S3	Acceleration	Acceleration	No	-	Positive number, less than the maximum acceleration	REAL
S4	Deceleration	Deceleration	Yes	Acceleration	Positive number, less than the maximum acceleration	REAL
S5	PulseWidth	Pulse width (unit: 0.01%)	Yes	5000	1 to 9999	INT
S6	CurveType	Velocity curve type 0: T-shaped velocity curve 1: 5-segment S-curve Others: T-shaped velocity curve	Yes	0	0 to 1	INT
D1	InVelocity	Velocity reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL

D3	CmdA-borted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Function Description

This instruction uses the Cyclic Synchronous Velocity (CSV) mode to control the bus servo axis or local pulse axis to keep the PLCOpen state machine of the axis in ContinuousMotion state. It has similar functions as MC_MoveVelocity.

When the bus servo axis is used, three object dictionaries need to be added to the PDO: 0x6060, 0x6061, and 0x60ff.

This instruction first write 9 into 0x6060 to switch the drive to CSV mode, then converts the target velocity into Int32 data and writes it into 0x60FF. The target velocity increases or decreases based on the specified acceleration or deceleration.

When the local pulse axis is used, no additional mapping parameters need to be configured. This instruction implements PWM waveform output with acceleration and deceleration. PulseWidth specifies the pulse width of the local pulse axis.

During execution of this instruction, you can call MC_Stop, Mc_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.

Note that the MC_MoveSuperImposed instruction cannot be called to perform motion superimposition during execution of this instruction.

3.10.1.26 MC_SyncMoveVelocity

MC_SyncMoveVelocity – CSV-based synchronous velocity control with adjustable pulse width

Graphic Block

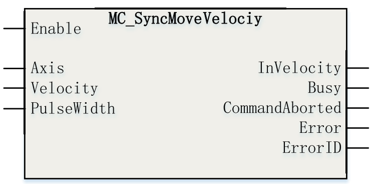
Instruction	Name	LD Expression	LiteST Expression
MC_ SyncMove-Velocity	CSV-based synchronous velocity control of the PWM waveform		MC_SyncMoveVelocity(Enable := ???, Axis := ???, Velocity := ???, PulseWidth := , InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–225 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_SyncMoveVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ Axis ID	No	-	-	_sMCAXIS_ INFO
S2	Velocity	Target velocity	No	-	Positive number/ number/0, absolute value less than the maximum velocity	REAL
S3	PulseWidth	Pulse width (unit: 0.01%)	Yes	5000	1 to 9999	INT
D1	InVelocity	Velocity reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CmdAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT

Function Description

This instruction uses the CSV mode to control the bus servo axis or local pulse axis to keep the axis in ContinuousMotion state.

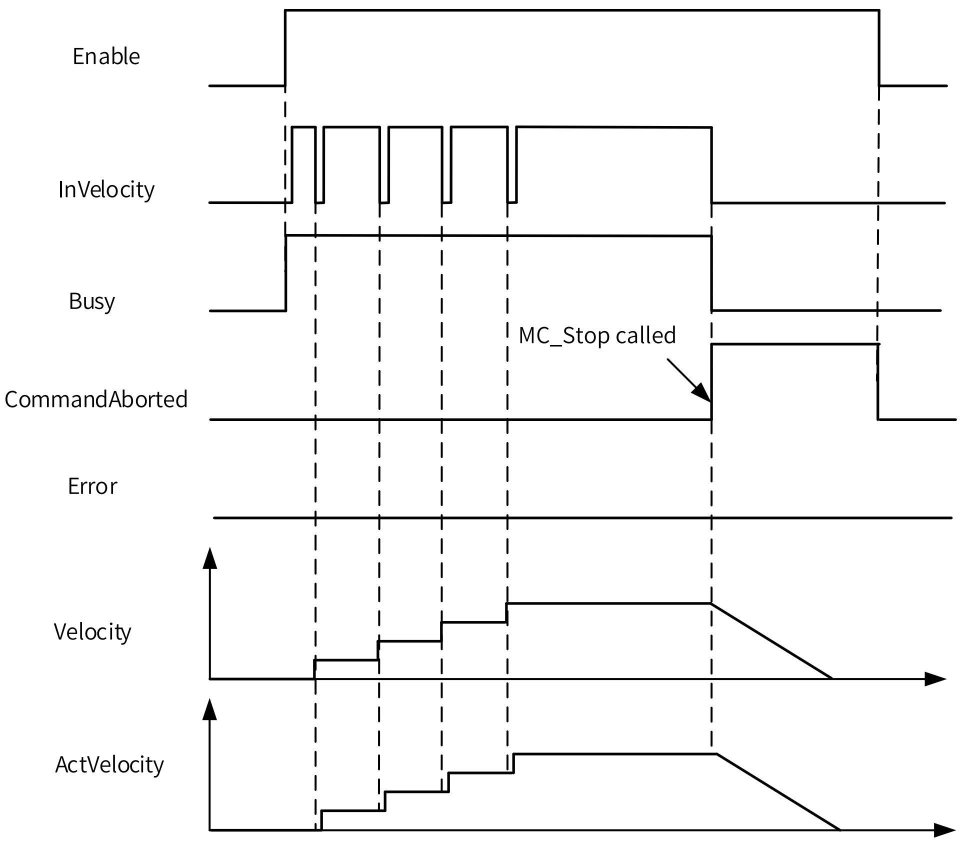
When the bus servo axis is used, three object dictionaries need to be added to the PDO: 0x6060, 0x6061, and 0x60ff.

This instruction first write 9 into 0x6060 to switch the drive to CSV mode, then converts the target velocity into Int32 data and writes it into 0x60FF.

When the local pulse axis is used, no additional mapping parameters need to be configured. This instruction can implement the PWM waveform output function. PulseWidth specifies the pulse width of the local pulse axis.

This instruction can modify the axis velocity and PWM duty cycle in real time in the program without re-triggering. The velocity after modification does not involve acceleration and deceleration and is directly converted into pulse equivalent and then written into 0x60FF.

During execution of this instruction, you can call MC_Stop, Mc_Halt, or MC_ImmediateStop (supported by the drive) to stop the motion of the axis.



Note that the MC_MoveSuperImposed instruction cannot be called to perform motion superimposition during execution of this instruction.

3.10.1.27 MC_SyncTorqueControl

MC_SyncTorqueControl – Synchronous torque control

Graphic Block

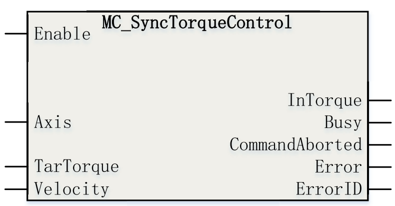
Instruction	Name	LD Expression	LiteST Expression
MC_ SyncTorqueControl	Syn- chro- nous torque con- trol		MC_SyncTorqueControl(Enable := ???, Axis := ???, TarTorque := ???, Velocity := , InTorque => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–226 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_SyncTorqueControl: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	Axis ID	No	-	0 to 32767	INT
S2	TarTorque	Target torque (unit: 1%)	No	-	Positive number, negative number, or 0	REAL
S3	Velocity	Velocity limit (user unit) This parameter is valid when 0x607f is mapped; otherwise, it is invalid.	Yes	0	Positive number or 0	REAL
D1	InTorque	Torque reached The output is active when the set torque reaches the target torque and the absolute value of the difference between the feedback torque and the target torque is less than 5%.	Yes	OFF	-	BOOL
D2	Busy	Busy flag	Yes	OFF	-	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	-	BOOL
D4	Error	Error flag	Yes	OFF	-	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT

Function description

This instruction is used to implement synchronous torque control only for the bus servo axis. It is active on the rising edge and does not support the imaginary axis mode.

This instruction can be executed only after the MC_Power instruction is executed to enable the axis.

The torque instruction can be used only when the following PDOs are configured: 0x6040, 0x6041, 0x6060, 0x6061, 0x6071, and 0x6077. Otherwise, a fault is reported.

This instruction adopts the synchronous torque mode of the drive to implement the torque control function. When Enable is ON, the function block converts the values specified by TarTorque and Velocity from user unit to pulse unit and transfers the data to the servo drive in real time. The axis remains in the ContinuousMotion state and performs synchronous torque motion.

TarTorque: Target torque, in the unit of 1%. Only one decimal place after the decimal point is valid in the program, and the subsequent ones are directly discarded. The actual torque of the drive is limited by the maximum positive and negative torque specified in the configuration parameters.

Specifying axis

Axis is latched on the rising edge of the Enable input.

Modification on Axis is invalid when Enable is ON.

Modification on Axis is valid when Enable is OFF.

Velocity control in torque mode

For servo drives of Inovance, if 0x607f is mapped, this instruction limits the maximum velocity of the servo motor through 0x607f. If 0x607f is not mapped, the velocity limit is invalid.

On the rising edge of Execute, the instruction converts the velocity limit specified by Velocity into pulse unit and writes it into 0x607f through PDO.

If the torque instruction is aborted by another instruction, the maximum velocity of the axis can be limited by specifying Max. Velocity on the configuration interface.

For third-party drives, Velocity can be used as the velocity limit only when the following conditions are met:

1. The maximum velocity of the servo motor can be limited by using 0x607F.
2. 0x607F can be configured in the PDO.
3. The unit of 0x607F is a pulse unit, not a rotation velocity unit.

Stop Control in Torque Mode

In torque mode, the MC_Stop instruction can be executed to stop the drive. Upon receiving the stop instruction, the drive switches to the synchronous position mode and decelerates according to the deceleration specified in the stop instruction.

Abortion

When this instruction is active, the axis is in ContinuousMotion state in PLCOpen. This instruction can be aborted by any other instruction that can make the axis enter DiscreteMotion state or conform to PLCOpen state machine switching. CommandAborted is active when this instruction is aborted.

When Execute is ON and the Done signal is inactive, if deceleration needs to be performed upon a limit signal, CommandAborted is active.

When Execute is ON and the Done signal is inactive, if the axis is disabled, CommandAborted is active.

3.10.1.28 MC_SetAxisConfigPara

MC_SetAxisConfigPara – Axis configuration parameters

Graphic Block

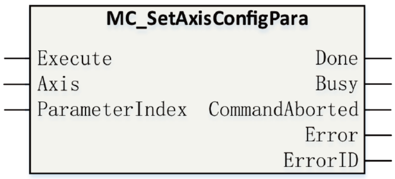
Instruction	Name	LD Expression	LiteST Expression
MC_ SetAxisConfigPara	Axis configu-ration parame-ter		MC_SetAxisConfigPara(Execute := ???, Axis := ???, ParameterIndex := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–227 Instruction format

16-bit Instruction	Consecutive execution of MC_SetAxisConfigPara					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	-	_sMCAXIS_ INFO
S2	ParameterIn-dex	Parameter index –1: All valid 0: All invalid 100: Modify only the gear ratio 200: Modify only the positive and negative software limits 300: Modify only the linearity or rotation mode 400: Modify only the encoder mode 500: Modify only the homing mode 600: Modify only the hard limit and home signal 700: Modify only the pulse output mode 800: Modify only the reverse settings 900: Modify only the virtual axis mode 1000: Modify only the probe signal 1100: Modify only the software limit variable	Yes	–1	-	INT

16-bit Instruction	Consecutive execution of MC_SetAxisConfigPara					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
D1	Done	Execution completed	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CommandA-borted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT16

Function and Instruction Description

This instruction is used to check and modify the configuration parameters of the axis. The axis parameters are reconfigured if they meet requirements. After configuration is completed, the Done signal output becomes active. If the configuration parameters do not meet requirements, the instruction reports an error.

ParameterIndex specifies the range of parameters to be modified. The values are described as follows:

- Parameter index –1: All parameters are updated. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.
- Parameter index 0: No parameter is updated.
- Parameter index 100: Only the gear ratio is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
dPlusePreCycle	DINT	Pulses per revolution of the motor/encoder
fDistancePreCycle	REAL	Distance per revolution of the rotary table
dNumerator	DINT	Gear ratio (numerator)
dDenominator	DINT	Gear ratio (denominator)

- Parameter index 200: Only the positive and negative software limits are modified. Modification is allowed when the axis is in Disabled or StandStill state.

Variable	Unit	Parameter
bSoftLimitEnable	BOOL	Software limit enable OFF: Disabled ON: Enabled
fPLimit	REAL	Positive limit in linear mode
fNLimit	REAL	Negative limit in linear mode

- Parameter index 300: Only the linear/rotary mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
iLineRotateMode	INT	Linear/Rotary mode 0: Linear mode 1: Rotary mode
fRotation	REAL	Rotation period in rotary mode

- Parameter index 400: Only the encoder mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
iEncodeMode	INT	Encoder mode (valid for the bus servo axis) 0: Absolute mode 1: Incremental mode

- Parameter index 500: Only the homing mode is modified. This index is valid for the local pulse axis. Modification is allowed when the axis is in Disabled or StandStill state.

Variable	Unit	Parameter
fHomeMethod	REAL	Homing mode
fHomeVelocity	REAL	Homing velocity
fHomeApproachVelocity	REAL	Homing approach velocity
fHomeAcceleration	REAL	Homing acceleration
dHomeTimeOut	DINT	Homing timeout time
dHomePositionMode	INT	Homing position mode

- Parameter index 600: Only the hardware limit and home signal are modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
bPLimitTerminalPolarity	BOOL	Positive limit polarity OFF: Positive logic ON: Negative logic
bNLimitTerminalPolarity	BOOL	Negative limit polarity OFF: Positive logic ON: Negative logic
bHomeTerminaPolarity	BOOL	Home signal polarity OFF: Positive logic ON: Negative logic
dPLimitTerminalID	DINT	ID of the positive limit signal (Modbus address)
dNLimitTerminalID	DINT	ID of the negative limit signal (Modbus address)
dHomeTerminalID	DINT	ID of the home signal (Modbus address)

- Parameter index 700: Only the pulse output mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
iPluseMethod	INT	Pulse output mode (valid for the local pulse axis)

- Parameter index 800: Only the reverse direction is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
bDirection	BOOL	Direction OFF: Forward ON: Reverse

- Parameter index 900: Only the imaginary axis mode is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
bVirtualMode	BOOL	Imaginary axis mode OFF: Disabled ON: Enabled

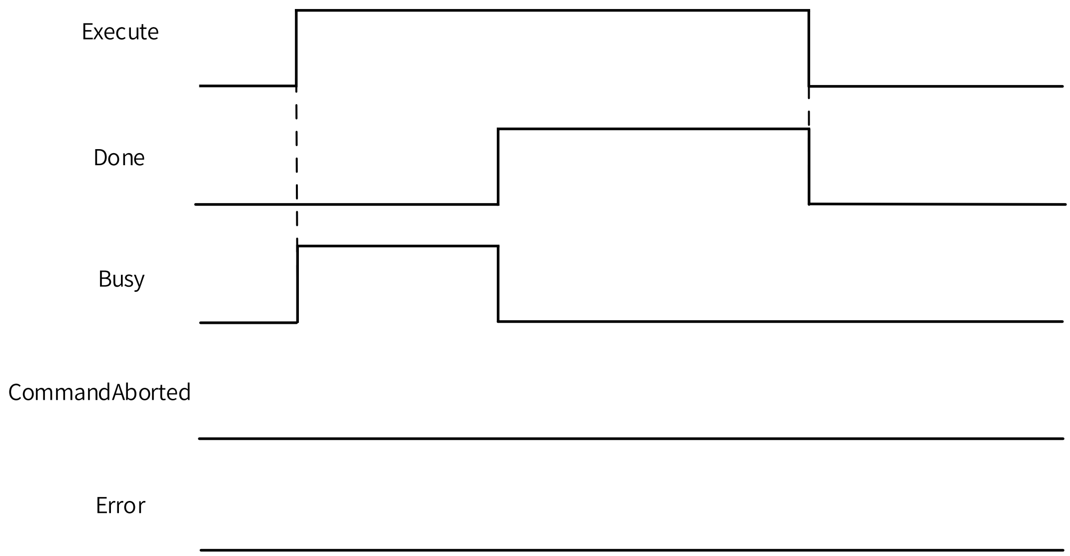
- Parameter index 1000: Only the probe signal is modified. Modification is allowed when the axis is in Disabled state. After the modification is completed, the current position may change greatly and homing needs to be performed.

Variable	Unit	Parameter
dTouchProbeID1	DINT	ID of probe terminal 1
dTouchProbeID2	DINT	ID of probe terminal 2

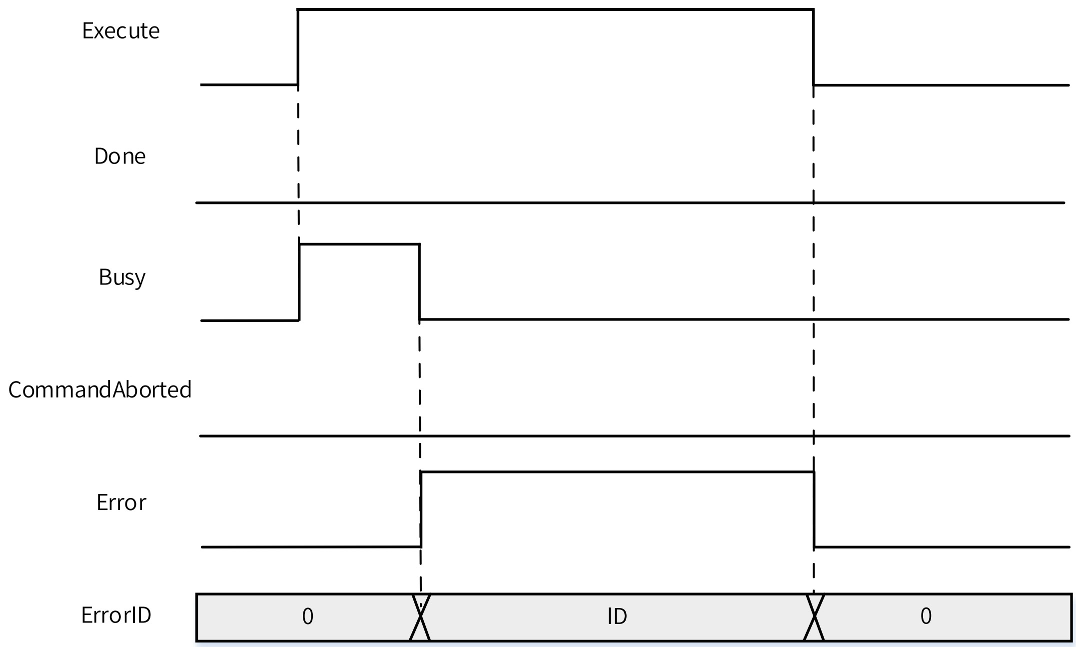
- Parameter index 1100: Only the software limit variables are modified. Modification is allowed when the axis is in Disabled or StandStill state.

Variable	Unit	Parameter
fLimitDeceleraion	REAL	Limit deceleration
fErrorStopDeceleration	REAL	Deceleration upon axis fault
fFollowErrorWindow	REAL	Following error window
fInVelocityWindow	REAL	Speed reach threshold
fMaxVelocity	REAL	Maximum velocity
fMaxJogVelocity	REAL	Maximum jogging velocity
fMaxAcc	REAL	Max acceleration
fMaxPTorque	REAL	Maximum positive torque
fMaxNTorque	REAL	Maximum negative torque
bEnterErrorStop	BOOL	Not entering ErrorStop state upon an axis fault OFF: Disabled ON: Enabled

Timing Diagram

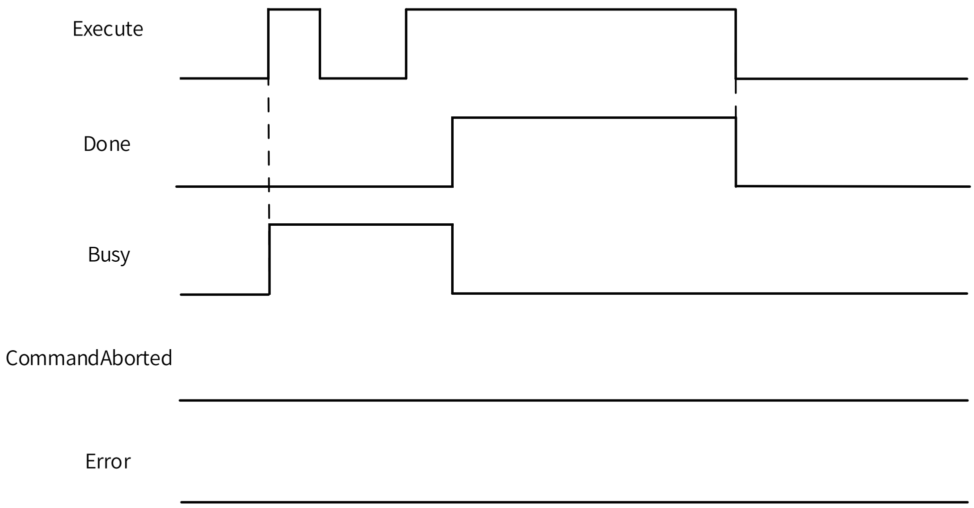


This instruction reports a fault when the parameters are configured improperly.



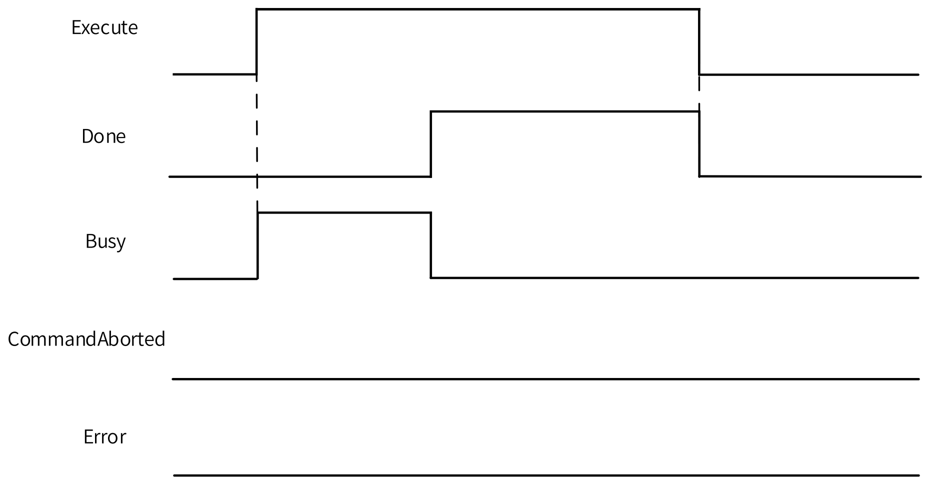
Re-triggering

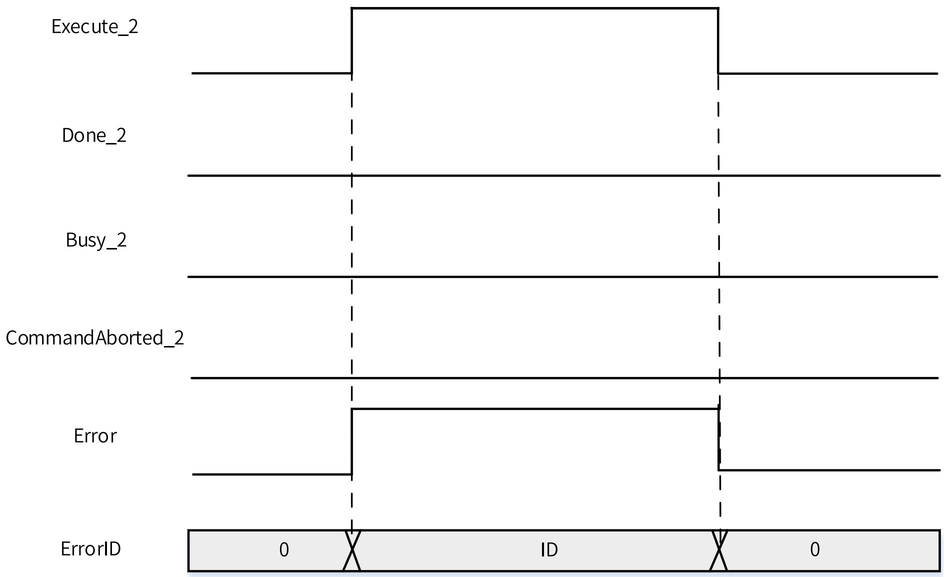
This instruction can be re-triggered while the Busy output is still active.



Multi-execution

It is not allowed to call a second MC_MetAxisConfigPara instruction while the Busy output of this instruction is still active; otherwise, the second instruction reports an error.





3.10.1.29 MC_FollowVelocity

MC_FollowVelocity – CSP-based velocity following

Graphic Block

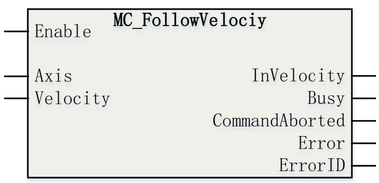
Instruction	Name	LD Expression	LiteST Expression
MC_ FollowVelocity	CSP-based velocity following		MC_FollowVelocity(Enable := ???, Axis := ???, Velocity := ???, InVelocity => , Busy => , CommandAborted => , Error => , ErrorID =>);

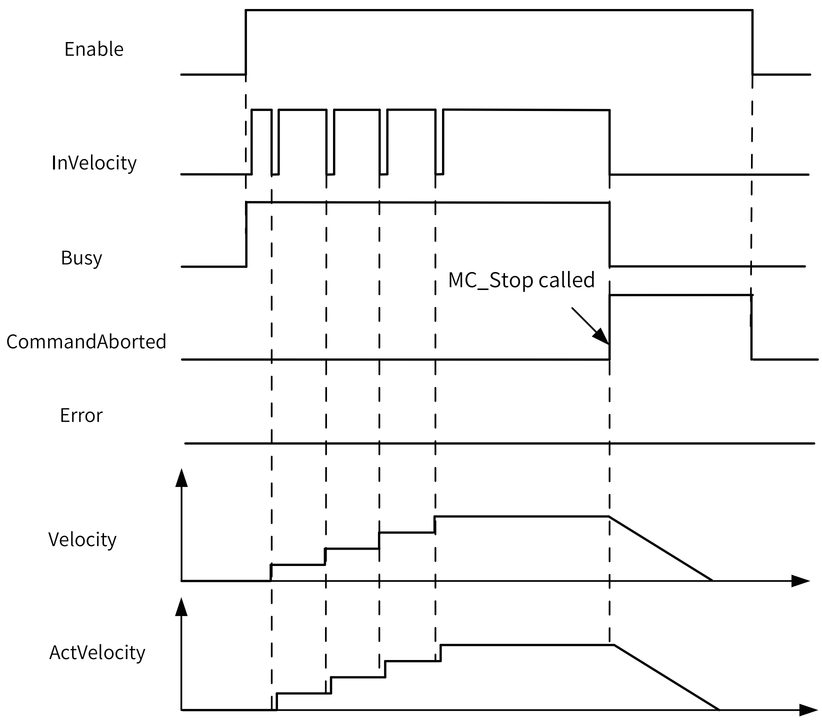
Table 3–228 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_FollowVelocity: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ Axis ID	No	-	-	_sMCAXIS_INFO
S2	Velocity	Target velocity	No	-	Positive number/ number/0, absolute value less than the maximum velocity	REAL32
D1	InVelocity	Velocity reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault Code	Yes	0	*1	INT16

Function Description

This instruction works with the MC_MoveSuperImposed instruction to implement the motion superimposition function. This instruction uses the CSP mode to control the bus servo or local pulse axis to keep the axis in SynchronizedMotion state. It works with MC_MoveSuperImposed to implement motion superimposition.

When Enable is ON, the velocity specified by Velocity takes effect immediately after modification, avoiding the necessity to re-trigger the instruction. The velocity after modification does not involve acceleration and deceleration. The way that the InVelocity signal is triggered is affected by the velocity window in axis configuration. To stop the axis, you need to call the MC_Stop instruction.



This instruction can work with the MC_MoveSuperImposed instruction to implement the motion superimposition function.

3.10.1.30 Axis Fault Codes

Axis fault codes are divided into local pulse axis fault codes and motion control axis fault codes. If an axis instruction reports a fault, see the description of the corresponding fault code.

Local Pulse Axis Fault Codes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9001 (0x2329)	Emergency stop The emergency stop terminal input is triggered.	Disable the emergency stop terminal input and then call the MC_Reset instruction to reset the fault.	Yes
9002 (0x232a)	The following error is too large. (Reserved)	Adjust the target velocity of acceleration (deceleration).	Yes
9003 (0x232b)	Overspeed occurs. The pulse output frequency exceeds 200 kHz.	Ensure that the pulse output frequency does not exceed 200 kHz.	Yes
9020 (0x233c)	A homing error occurs. The negative limit is not mapped.	Map the negative limit on the configuration interface.	Yes
9021 (0x233d)	A homing error occurs. The positive limit is not mapped.	Map the positive limit on the configuration interface.	Yes
9022 (0x233e)	A homing error occurs. The home signal is not mapped.	Map the home switch on the configuration interface.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9023 (0x233f)	A homing error occurs. The output frequency exceeds 200 kHz when the axis runs at the homing velocity. The output frequency exceeds 200 kHz when the axis runs at the homing approach velocity.	Modify the unit conversion setting to ensure that the homing velocity and homing approach velocity do not exceed 200 kHz. Change the homing velocity to ensure that the output frequency does not exceed 200 kHz. Change the homing approach velocity to ensure that the output frequency does not exceed 200 kHz.	Yes
9024 (0x2340)	A homing error occurs. Homing timed out.	Check that the limit signal is conductive. Check whether the homing timeout time is too short.	Yes
9025 (0x2341)	A homing error occurs. The limit signal is disordered during homing.	Check whether the limit signal that is not applicable to the current homing mode is triggered.	Yes
9030 (0x2342)	The limit is active.	Check whether the limit is reached during normal running.	No
9031 (0x2343)	A synchronization error occurs. The target number of transmitted pulses and the actual number of transmitted pulses do not match.	Check whether the limit is reached during normal positioning.	No

Note When a local pulse axis is faulty, please refer to the preceding fault code list for fault information.

Motion Control Axis Fault Codes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9101	The type of the axis specified by AxisID is incorrect. The axis specified by AxisID does not exist.	Check whether the instruction supports the axis specified by AxisID. Check whether the axis specified by AxisID exists.	No
9102	The axis configuration data is lost. The axis configuration parameters are improper.	Check whether the parameters are correct.	No
9103	The MC_Reset instruction is called when the axis is not faulty.	Check whether the MC_Reset instruction is called when the axis is not switched to ErrorStop state.	No
9104	The axis is in unknown state when the MC_ReadStatus instruction is called.	Check whether the current state of the axis is uncontrollable by using the online monitoring function.	No
9105	Setting the current position is not allowed.	Check whether the MC_SetPositon instruction has been called.	No
9106	The axis is stopping upon a fault.	Execute the instruction after stop upon fault is completed and the fault is resolved.	No

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9107	The parameters are improper.	Check whether the parameters on the left of the instruction are set properly.	Yes
9108	The PLCOpen state machine is improper.	Check whether the current PLCOpen state machine satisfies the execution conditions for this instruction. If not, call the relevant instruction to switch the axis to the required state.	No
9109	The axis enters the Disabled state during instruction execution.	Check whether the axis has entered the Disabled state.	No
9110	The MC_Stop instruction is called repeatedly during stop.	Check whether the MC_Stop instruction is called repeatedly in the program.	No
9111	The instruction linked list is lost.	Check whether the background version and board version match.	No
9112	The axis number changes. The axis number changes while the instruction flow is active.	Do not change the axis number while the flow is active for Enable instructions such as MC_Power and MC_Jog.	Yes
9113	Reset by executing the MC_Reset instruction timed out.	Check whether the drive fault can be reset. Check whether the fault type supports reset.	No
9114	The axis fails to write to 0x6060.	Check for interference in network communication.	No
9115	The MC_Halt instruction is called when the axis is in Stopping state.	Do not call the MC_Halt instruction when the axis is in Stopping state.	No
9116	The current axis is in online commissioning mode.	Check whether the current axis is in online commissioning mode.	No
9118	The acceleration (deceleration) of the instruction exceeds the maximum acceleration.	Check whether the acceleration (deceleration) of the instruction exceeds the maximum acceleration.	Yes
9119	The target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	Check whether the target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	Yes
9120	The target velocity exceeds the maximum velocity.	Check whether the target velocity of the instruction exceeds the maximum velocity.	Yes
9121	The forward and reverse motion signals of the jog instruction are both active.	Ensure that the forward and reverse motion signals of the jog instruction are not active at the same time.	Yes
9122	The control word is not mapped to the EtherCAT bus axis.	Add the control word in the PDO and map it to the axis.	No
9123	The target position is not mapped to the EtherCAT bus axis.	Add the target position in the PDO and map it to the axis.	No
9124	The target torque is not mapped to the EtherCAT bus axis.	Add the target torque in the PDO and map it to the axis.	No
9125	The status word is not mapped to the EtherCAT bus axis.	Add the status word in the PDO and map it to the axis.	No
9126	The current position is not mapped to the EtherCAT bus axis.	Add the feedback position in the PDO and map it to the axis.	No
9127	0x60fd is not mapped to the EtherCAT bus axis.	Add 0x60fd in the PDO and map it to the axis.	No

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9128	The current torque is not mapped to the EtherCAT bus axis.	Add the current torque in the PDO and map it to the axis.	No
9129	The probe control word is not mapped to the EtherCAT bus axis.	Add the probe control word in the PDO and map it to the axis.	Yes (interrupt positioning) No (probe)
9130	The probe status word is not mapped to the EtherCAT bus axis.	Add the probe status word in the PDO and map it to the axis.	Yes (interrupt positioning) No (probe)
9131	The probe position is not mapped to the EtherCAT bus axis.	Add the probe position in the PDO and map it to the axis.	Yes (interrupt positioning) No (probe)
9132	An interrupt positioning instruction is being executed and the probe channel is occupied.	The probe instruction and interrupt positioning instruction must not occupy the same probe channel at the same time. When the two instructions are called simultaneously in the program, the interrupt positioning instruction takes priority.	No
9133	The imaginary axis mode is enabled.	The current instruction does not support the imaginary axis mode.	No
9134	Reserved	-	-
9135	The interrupt signal is not triggered in the interrupt positioning instruction.	During execution of the interrupt positioning instruction, no interrupt signal is detected after positioning is completed.	No
9136	The probe channel is occupied by another instruction during the interrupt positioning process.	Ensure that the probe channel is not occupied during the interrupt positioning process.	Yes
9137	The control mode 0x6060 is not mapped to the bus driver.	Add 0x6060 in the PDO and map it to the axis.	No
9138	The control mode 0x6061 is not mapped to the bus driver.	Add 0x6061 in the PDO and map it to the axis.	No
9139	The MC_Home instruction is called repeatedly during homing.	Do not call the MC_Home instruction repeatedly during homing.	No
9140	The target torque of the instruction exceeds the maximum value.	Check whether the target torque of the instruction exceeds the positive and negative torque limits.	Yes
9141	The maximum velocity is not mapped to the bus driver.	Add 0x607f in the PDO and map it to the axis.	No
9142	The immediate stop instruction is active.	Check whether the immediate stop instruction has been called.	No
9143	The immediate stop instruction is called repeatedly.	Check whether the immediate stop instruction is called repeatedly.	No
9144	The limit is reached during jogging.	Check whether the limit is active.	No

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9145	The target position exceeds 9999999. The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target position must not exceed this value.	Check whether the target position is correct. Set the target position again. Change the gear ratio to ensure that the target position is not greater than 9999999.	Yes
9146	The target velocity exceeds 9999999. The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target velocity must not exceed this value.	Check whether the target velocity is correct. Set the target velocity again. Change the gear ratio to ensure that the target velocity is not greater than 9999999.	Yes
9147	The target acceleration exceeds 9999999. The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target acceleration must not exceed this value.	Check whether the target acceleration is correct. Set the target acceleration again. Change the gear ratio to ensure that the target acceleration is not greater than 9999999.	Yes
9148	The target deceleration exceeds 9999999. The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target deceleration must not exceed this value.	Check whether the target deceleration is correct. Set the target deceleration again. Change the gear ratio to ensure that the target deceleration is not greater than 9999999.	Yes
9149	Execution of single-axis motion instructions is not allowed because the axis is in sync control mode.	Check whether the axis is executing the interpolation instruction. Execution of single-axis motion instructions is not allowed during interpolation.	No
9501	The servo drive is faulty.	For the EtherCAT bus axis, check the slave fault type by using 0x603f and then eliminate the fault. For the local axis, check the local axis fault list to troubleshoot the fault.	Yes
9502	The drive is disabled.	Check whether the drive status word 0x6041 switches to the disabled state during motion.	Yes
9503	The drive has reached the limit.	Check whether the limit is configured and whether the limit signal is active.	Yes
9504	Reserved	-	-
9505	Writing to 0x6060 failed.	Check for interference in network communication.	Yes
9506	Reserved	-	-
9507	Reserved	-	-

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9508	Homing fault	Identify the cause of the drive homing failure. Check whether homing timed out. Check whether the limit signal is incorrect.	Yes
9509	Loss of precision occurs.	Check whether the floating-point data of the instruction falls beyond the single-precision floating-point number range.	Yes
9510	The following error is too large. The difference between the set position and the feedback position exceeds the set threshold.	Check whether acceleration is too large. Check whether the set following error is too small.	Yes
9511	Reserved	-	-
9512	Drive communication failed during operation.	Check whether the drive works properly. Check whether the network cable is properly connected. Check for interference in communication.	Yes
9513	Homing failed due to a drive fault.	Check the fault code of the drive to eliminate the fault.	Yes
9514	Homing failed because the homing offset exceeded 32 bits.	Check whether the homing offset multiplied by the gear ratio exceeds 32 bits; if yes, change the gear ratio.	Yes
9515	Homing failed due to loss of the slave.	Contact Inovance for technical support.	Yes
9516	Homing failed because the SDO failed to write to object dictionary 0x607C.	1. Check whether the drive supports 0x607C. 2. Check the network communication quality.	Yes
9517	Homing failed because the SDO failed to write 6 to object dictionary 0x6060.	1. Set 0x6060 in the PDO. 2. Check the network communication quality.	Yes
9518	Homing failed because the SDO failed to read object dictionary 0x6061.	1. Set 0x6061 in the PDO. 2. Check the network communication quality.	Yes
9519	Homing failed because the SDO failed to write 8 to object dictionary 0x6060.	1. Set 0x6060 in the PDO. 2. Check the network communication quality.	Yes
9551	State switching failed.	Check for interference in network communication.	Yes
9552	The target velocity is less than 0.	Check whether the target velocity of position instructions is appropriate.	Yes
9601	The axis stops due to an error of the absolute positioning instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9602	The axis stops due to an error of the relative positioning instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9603	The axis stops due to an error of the velocity control instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9604	The axis stops due to an error of the jogging instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9605	Reserved	-	-
9606	The axis stops due to an error of the buffer control instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9607	The axis stops due to an error of the interrupt positioning instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9608	The axis stops due to an error of the stop instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9609	The axis stops due to an error of the torque control instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9610	The axis stops due to an error of the halt instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.	Yes
9800	Failed to obtain the number of axes.	Change the background version.	Yes
9801	The number of axes is greater than 32.	Reduce the number of axes since the H5U supports at most 32 axes.	Yes
9802	Failed to request the memory.	Check whether the memory runs out.	Yes
9803	Failed to obtain parameters.	Check whether the board and the background version match.	Yes
9804	Failed to obtain the slave.	None	Yes

NoteWhen a motion control axis is faulty, please refer to the preceding fault code list for fault information.

3.10.2 Cam and Gear Instructions

3.10.2.1 Instruction List

The following table lists the electronic cam instructions.

Instruction Category	Instruction	Description
Electronic cam instruction	MC_CamIn	Start cam operation
	MC_CamOut	End cam operation
	MC_GetCamTablePhase	Obtain cam table phase
	MC_GetCamTableDistance	Obtain cam table displacement
	MC_DigitalCamSwitch	Electronic cam tappet control
	MC_GearIn	Start gear operation
	MC_GearOut	End gear operation
	MC_Phasing	Master axis phase shifting
	MC_SaveCamTable	Save cam table
	MC_GenerateCamTable	Update cam table
	MC_GearInPos	Start the gear operation at the specified position

3.10.2.2 MC_CamIn

MC_CamIn – Start cam operation

Graphic Block

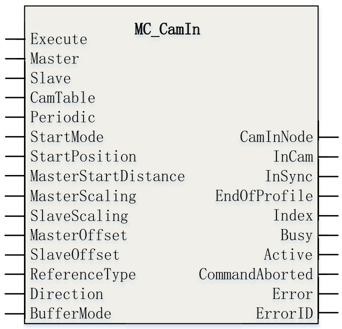
Instruction	Name	LD Expression	LiteST Expression
MC_CamIn	Start cam operation		MC_CamIn(Execute := ???, Master := ???, Slave := ???, CamTable := ???, Periodic := , StartMode := , StartPosition := , MasterStartDistance := , MasterScaling := , SlaveScaling := , MasterOffset := , SlaveOffset := , ReferenceType := , Direction := , BufferMode := , CamInNode => , InCam => , InSync => , EndOfProfile => , Index => , Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

Table 3–229 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_CamIn: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Master	Master axis Bus servo axis, local pulse axis, bus encoder axis, or local encoder axis	No	-	-	_sMCAX-IS_INFO _sENC_AXIS _sENC_EXT_AXIS _sMas-terAxis
S2	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAX-IS_INFO
S3	CamTable	Cam table	No	-	-	_sMC_CAMTA-BLE
S4	Periodic	Periodic mode 0: Periodic Others: Periodic for a specified number of cycles	0	0	0 to 32767	INT
S5	StartMode	Mode for specifying MasterStartDistance 0: Absolute mode 1: Relative mode 2: Start immediately	Yes	0	0 to 2	INT
S6	StartPosi-tion	Start position of the cam table	Yes	0	Positive number 0 Negative number	REAL
S7	Master-StartDis-tance	Master following distance	Yes	0	Positive number 0 Negative number	REAL
S8	Master-Scaling	Master coefficient	Yes	1	Positive number	REAL
S9	SlaveScal-ing	Slave coefficient	Yes	1	Positive number	REAL
S10	MasterOff-set	Master offset	Yes	0	Positive number 0 Negative number	REAL
S11	SlaveOffset	Slave offset	Yes	0	Positive number 0 Negative number	REAL

S12	Reference-Type	Position type 0: Instruction position of the previous cycle 1: Instruction position of the current cycle [1] 2: Feedback position of the current cycle	Yes	0	0 to 2	INT
S13	Direction	Direction 0: Forward 1: Reverse 2: Not specified	Yes	0	0 to 2	INT
S14	Buffer-Mode	Buffer mode 0: Wait until the previous one is completed Others: Reserved	Yes	0	-	INT
D1	CamIn-Node	Cam engagement variable	Yes	-	-	_sMC_CAMIN
D2	InCam	Cam motion	Yes	OFF	ON OFF	BOOL
D3	InSync	Synchronizing	Yes	OFF	ON OFF	BOOL
D4	EndOfPro-file	End of cam cycle	Yes	OFF	ON OFF	BOOL
D5	Index	Index	Yes	0	1 to 360	INT
D6	Busy	Executing	Yes	OFF	ON OFF	BOOL
D7	Active	Controlling	Yes	OFF	ON OFF	BOOL
D8	Comman-dAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D9	Error	Error	Yes	OFF	ON OFF	BOOL
D10	ErrorID	Error code	Yes	0	-	INT

Note

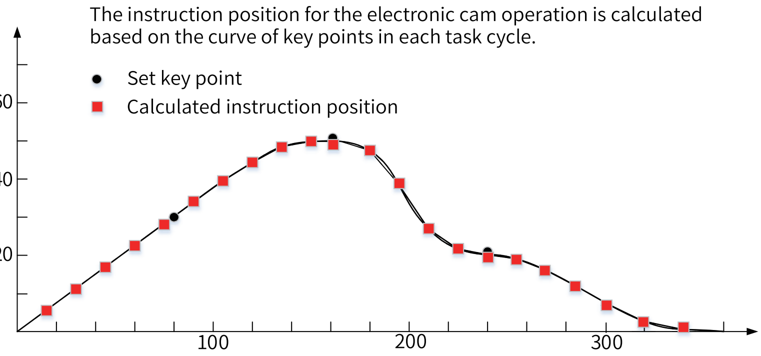
[1]: When selecting the set position under the same task, make sure that the axis ID of the master axis is smaller than that of the slave axis.

Relative Cam Table

The phase and displacement of the cam table are specified as relative quantities from a start point of 0.0. In each EtherCAT cycle, the cam calculation unit calculates the displacement of the slave axis corresponding to the phase of the master axis according to the selected cam curve type.

Cam table

	Phase	Displacement
Start point	0	0
	80	30
	160	50
	240	20
End point	360	0



Instruction Execution Condition

You can execute this instruction while the master axis is stopped, during position control, velocity control, or synchronized control.

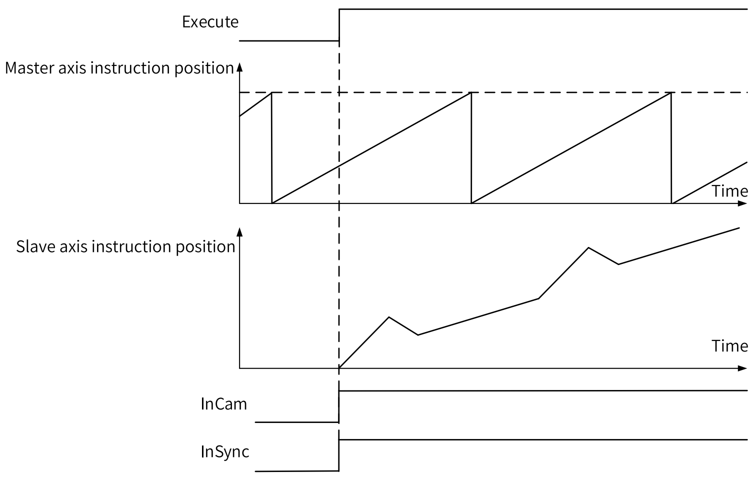
You can execute this instruction while the slave axis is in StandStill, DiscreteMotion, ContinuousMotion, or SynchronizedMotion (non-axis-group motion) state.

Software Limits

If the slave axis exceeds the software limit during cam operation, an error occurs and the axis stops running.

Starting Cam Operation

- StartMode = 2 (Start immediately)
  After the instruction is executed, the cam operation is performed immediately. The current position of the master axis is phase 0 of the cam, and that of the slave axis is displacement 0 of the cam.



- StartMode = 0 or 1 (Start from specified position)
  After the instruction starts, the master axis has to reach the StartPosition (start position of the cam table).

  After the master axis passes the StartPosition (start position of the cam table), the start point in the cam table is executed and the InCam output variable (cam motion) changes to ON.

The phases and displacements in the cam table are specified as relative quantities from zero. The absolute position of each axis at each phase is the relative value from the absolute position of the axis at the start point of the cam table. For example, if the count mode of the master axis is 0° to 360° in rotary mode and the cam table is as shown in the following figure, the StartPosition (start position of the cam table) is 50. The absolute position of the master axis is the phase added to the StartPosition, as shown in the following cam table. The absolute position of the slave axis is the displacement from the cam table added to the absolute position of the slave axis at the start point of the cam table.

Cam table

	Phase	Displacement
Start point	0	0
	80	30
	120	50
	240	20
End point	360	0

StartPosition=50

Absolute positions of axes

Master axis	Slave axis
50	0+Absolute position of slave axis at starting point of cam table
130	30+Absolute position of slave axis at starting point of cam table
170	50+Absolute position of slave axis at starting point of cam table
290	20+Absolute position of slave axis at starting point of cam table
50	0+Absolute position of slave axis at starting point of cam table

When the MasterStartDistance (master following distance) is then passed, the cam operation of the slave axis starts and the output variable InSync (synchronizing) changes to ON.

The MasterStartDistance (master following distance) is specified either as an absolute position (StartMode = 0), or as a relative distance (StartMode = 1) from the StartPosition (start position of the cam table).

The cam table settings are as follows:

Phase	Displacement
0	0
80	120
120	80
360	140

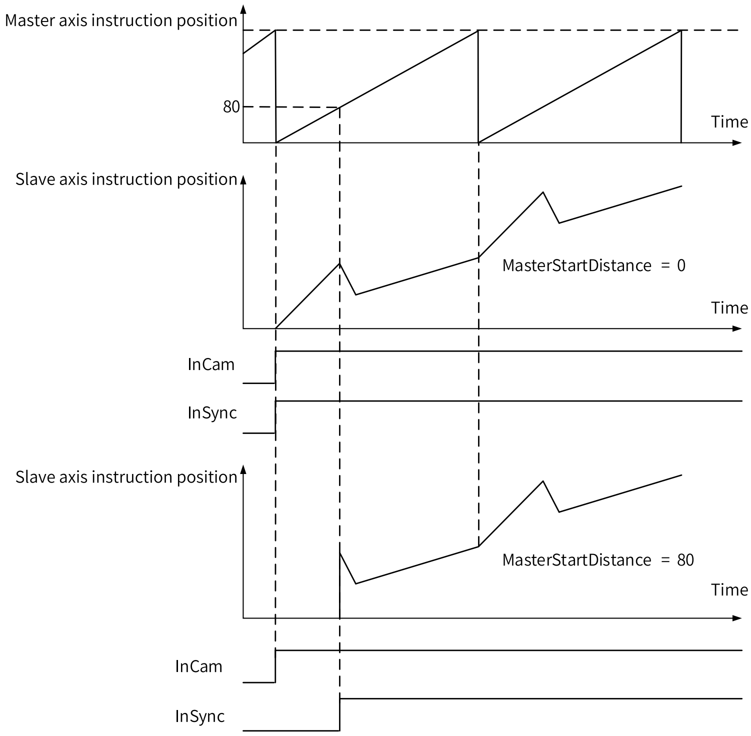
The conditions for starting cam operation are as follows:

Input Variable	Condition 1	Condition 2
Periodic (Periodic mode)	0	0
StartMode (Mode for specifying the start position)	Relative position	Relative position
StartPosition (Start position of the cam table)	0	0
MasterStartDistance (Master following distance)	0	80

For condition 1, the output variables InCam (cam motion) and InSync (synchronizing) both change to ON and the slave axis starts cam operation when the master axis passes 0°.

For condition 2, the output variable InCam (cam motion) changes to ON when the master axis passes 0°. Then, the output variable InSync (synchronizing) changes to ON and the slave axis starts cam operation when the master axis passes 80°.

Note that for condition 2, cam operation starts in the middle of the cam table, so the slave axis will accelerate rapidly.



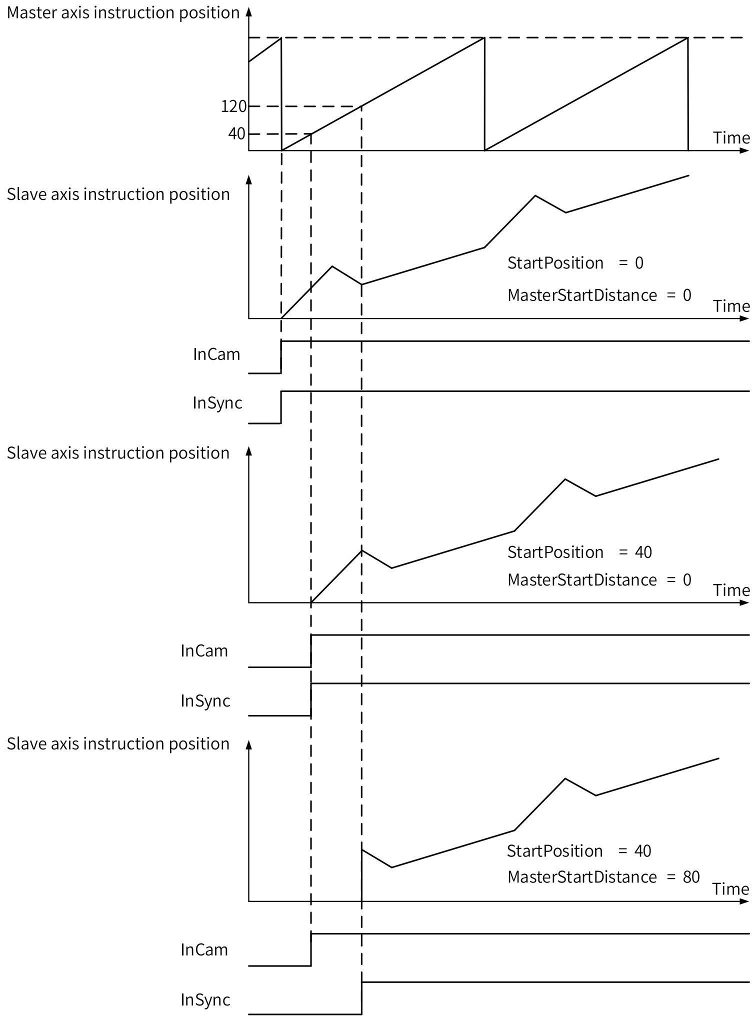
The cam table settings are the same as in the previous example. The conditions for starting cam operation are modified as follows:

Input Variable	Condition 1	Condition 2	Condition 3
Periodic	0	0	0
StartMode	Relative position	Relative position	Relative position
StartPosition	0	40	40
MasterStartDistance	0	0	80

For condition 1, the output variables InCam (cam motion) and InSync (synchronizing) both change to ON and the slave axis starts cam operation when the master axis passes 0°.

For condition 2, the output variables InCam (cam motion) and InSync (synchronizing) both change to ON and the slave axis starts cam operation when the master axis passes 40° specified by StartPosition (start position of the cam table).

For condition 3, the output variable InCam (cam motion) changes to ON when the master axis passes 40°. Then, the output variable InSync (synchronizing) changes to ON and the slave axis starts cam operation when the master axis passes 120°.



You can use StartMode to specify whether the value specified by MasterStartDistance (master following distance) is treated as an absolute position or a relative position.

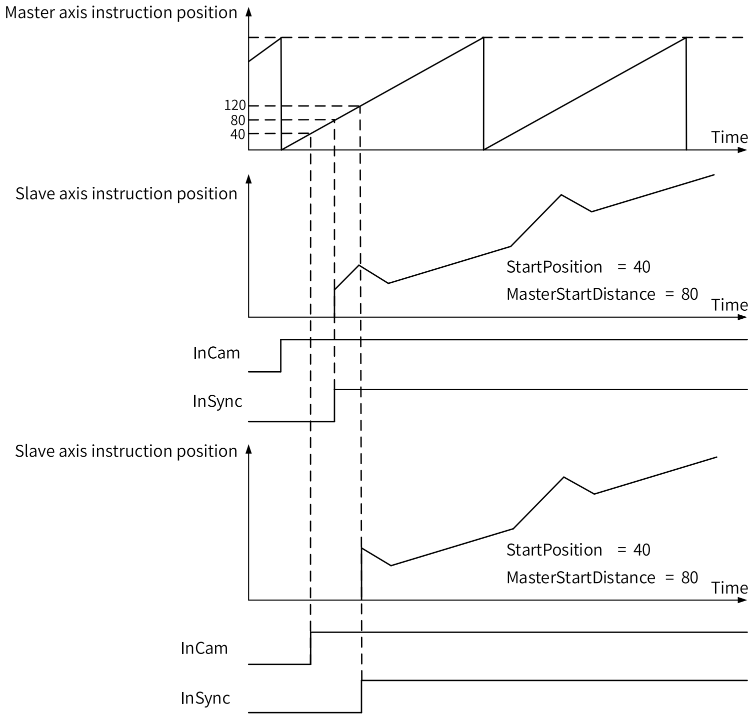
The following describes the differences in starting cam operation of the slave axis based on differences in StartMode. The cam table settings are the same as in the previous example.

The conditions for starting cam operation are as follows:

Input Variable	Condition 1	Condition 2
Periodic	0	0
StartMode	Absolute position	Relative position
StartPosition	40	40
MasterStartDistance	80	80

For both conditions 1 and 2, the output variable InCam (cam motion) changes to ON when the master axis passes 40°. For condition 1, StartMode is set to 0 (absolute position), so the output variable InSync changes to ON and the slave axis starts cam operation when the master axis passes 80°.

For condition 2, StartMode is set to 1 (relative position), so the output variable InSync changes to ON and the slave axis starts cam operation when the master axis passes 120° (= 40° + 80°).



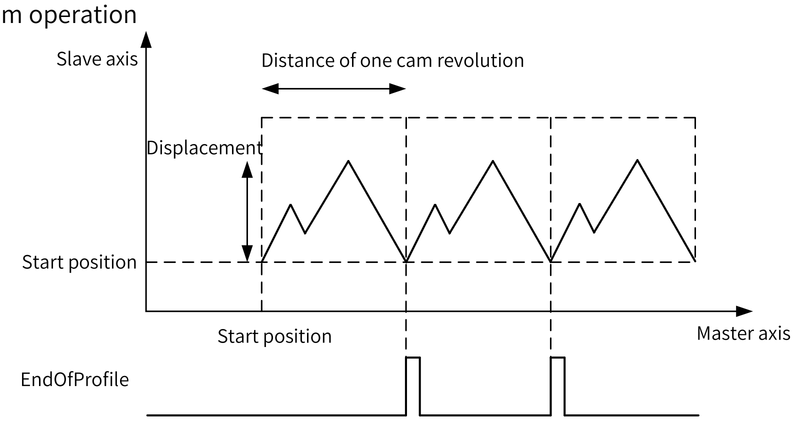
Periodic Mode

When Periodic (periodic mode) is set to 0, the cam motion is repeated from the start to the end point of the cam table. After each cam cycle ends, EndOfProfile is set to TRUE for one PLC scan cycle.

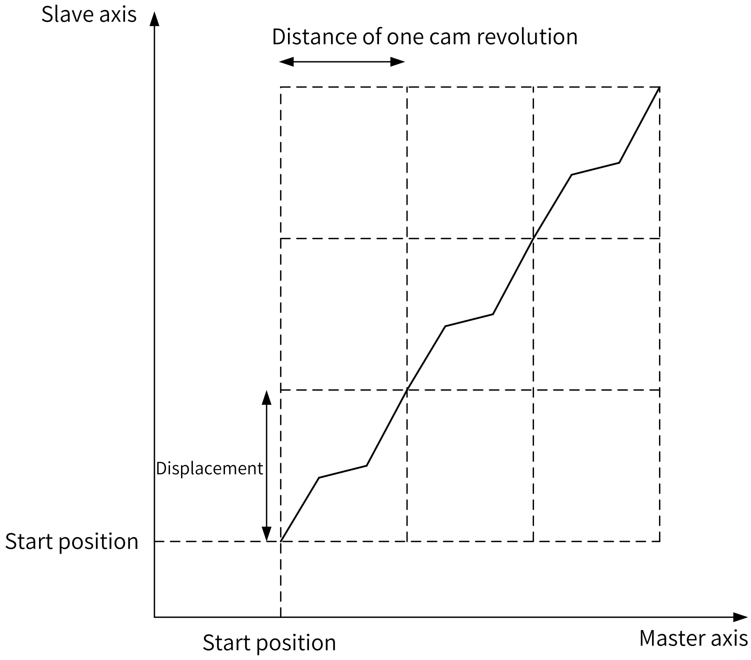
When Periodic (periodic mode) is set to N (N > 0), the cam motion is repeated N times and then ends. After the last cycle, if the Execute input is ON, the EndOfProfile will always be TRUE; if the Execute input is OFF, the EndOfProfile is set to TRUE for one PLC scan cycle.

If the stroke position of the slave axis is the same at the start and end points of the cam table during the repeating process, the cam operates as a reciprocal cam. If the stroke position of the slave axis differs at the start point and end point, the cam operates as a feeding cam.

- Reciprocal cam operation



- Feeding cam operation



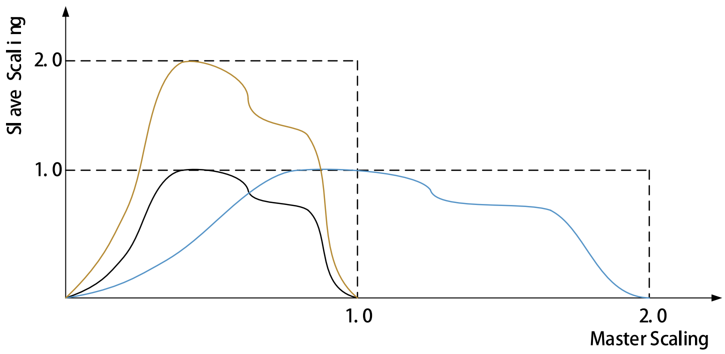
End of Cam Cycle

You can use the MC_CamOut (end cam operation) instruction or MC_Stop instruction to stop cam operation before it is completed.

Scaling Factor

You can specify a scaling factor to scale up or scale down the master axis phase and slave axis displacement of a specified cam table.

You can apply separate factors to the master and slave axes.

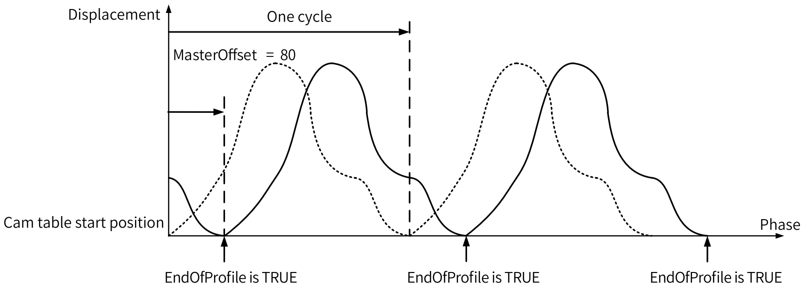


Offset

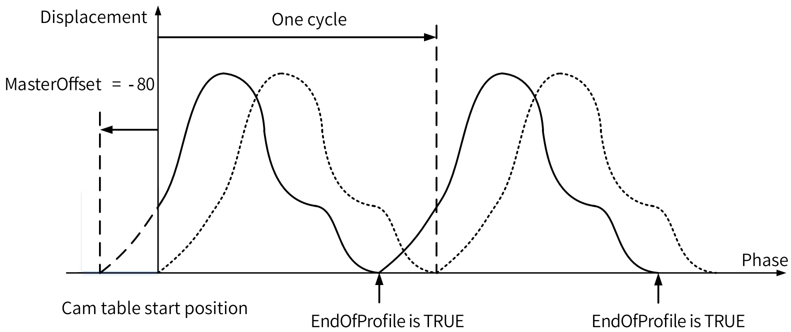
You can shift the phase and displacement by an offset from the specified cam table.

You can specify separate offsets for the master axis phase and slave axis displacement.

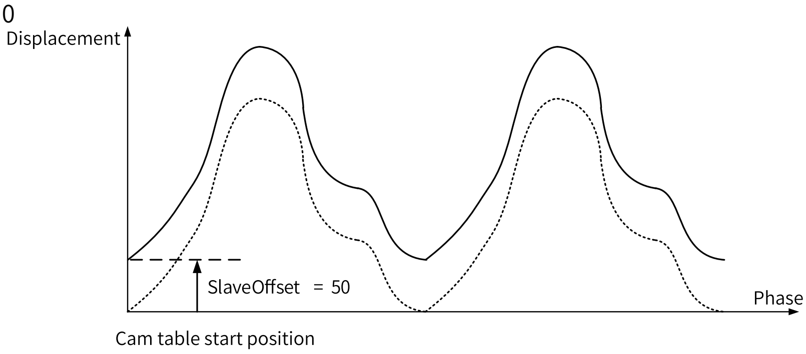
- MasterOffset > 0



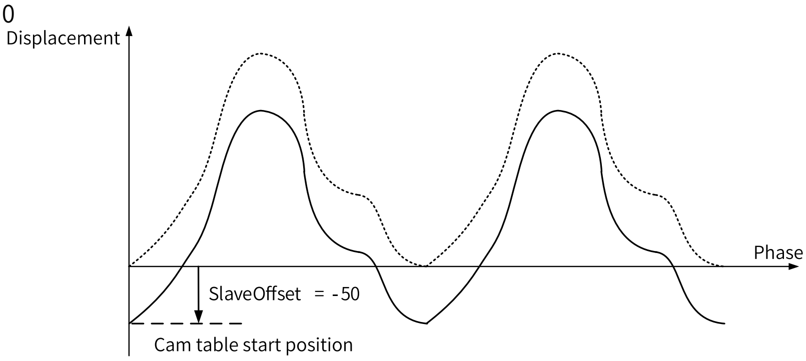
- MasterOffset < 0



- SlaveOffset > 0



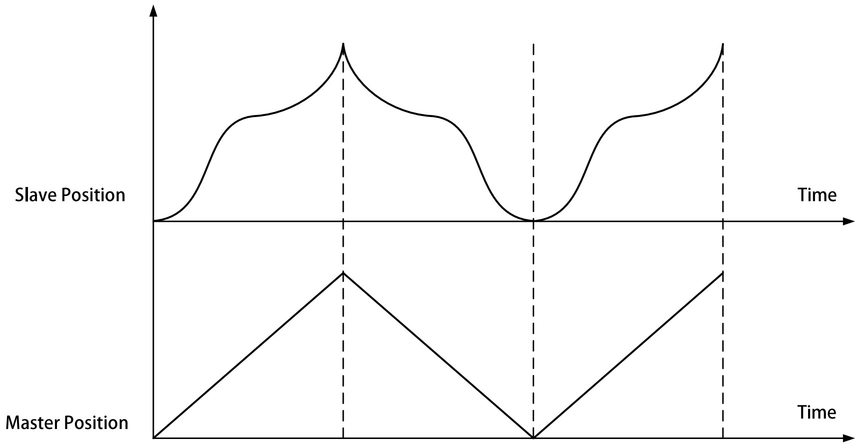
- SlaveOffset < 0



Direction

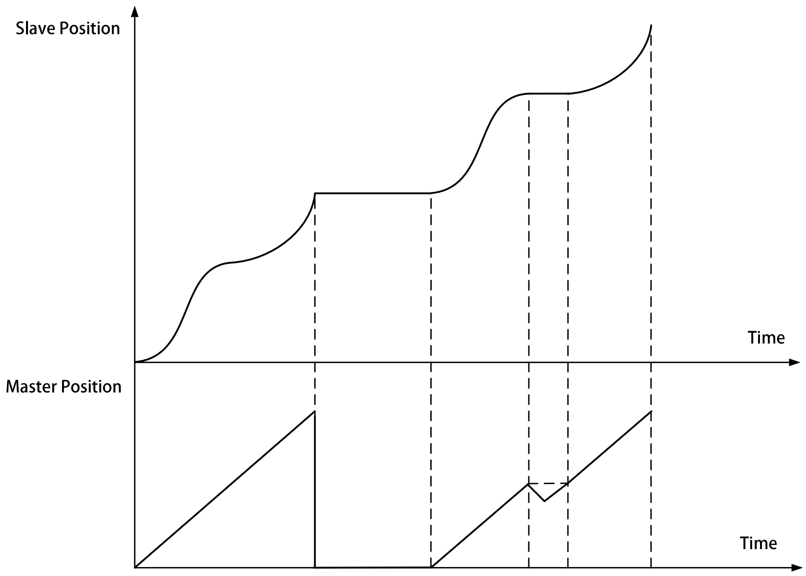
You can start cam operation for the slave axis only if the travel direction of the master axis matches that specified by Direction.

- No direction specified
  Cam operation starts regardless of whether the master axis is traveling in the positive or negative direction.



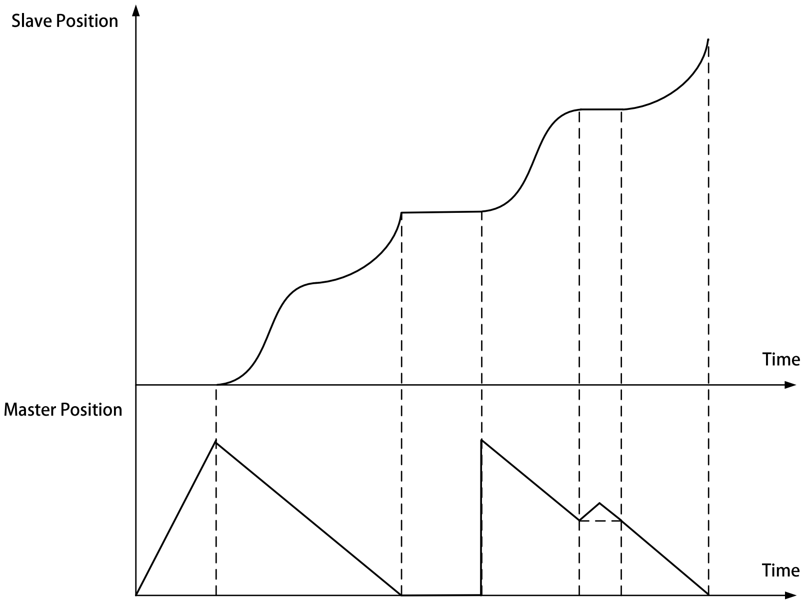
- Positive direction

  Cam operation starts when the master axis is moving in the positive direction. In a cam cycle, if the master axis is reversed, the slave axis remain stationary until the master axis returns to its original position. Then the slave axis continues to follow the master axis.



- Negative direction

  Cam operation starts when the master axis is moving in the negative direction. In a cam cycle, if the master axis is reversed, the slave axis remain stationary until the master axis returns to its original position. Then the slave axis continues to follow the master axis.



Position Type

ReferenceType specifies the data source of the master axis position.

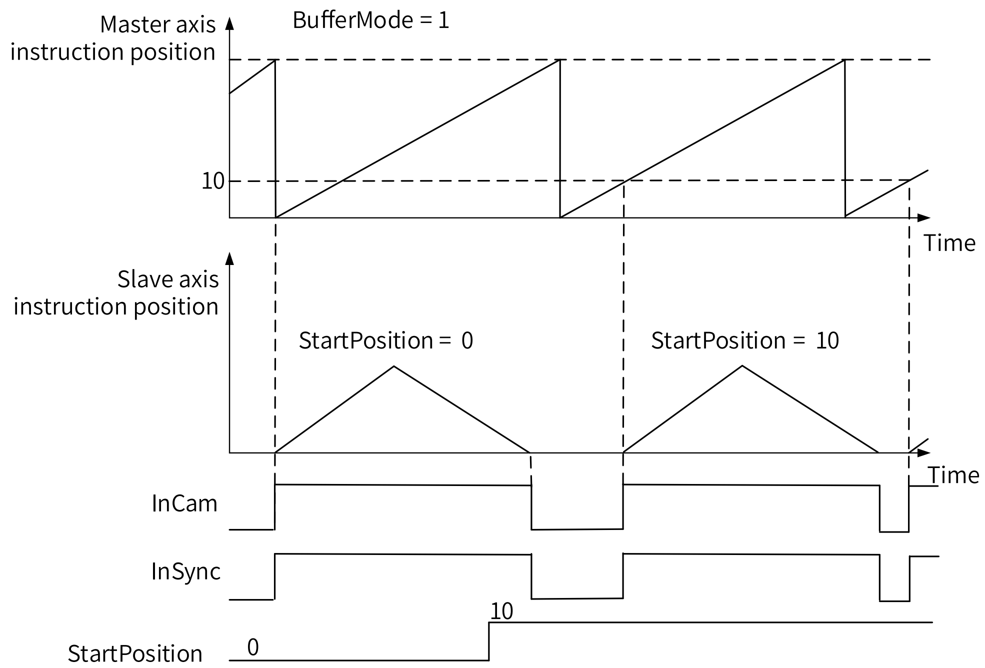
When the master axis is a local encoder axis, this parameter is invalid, and the feedback position of the current cycle is always used.

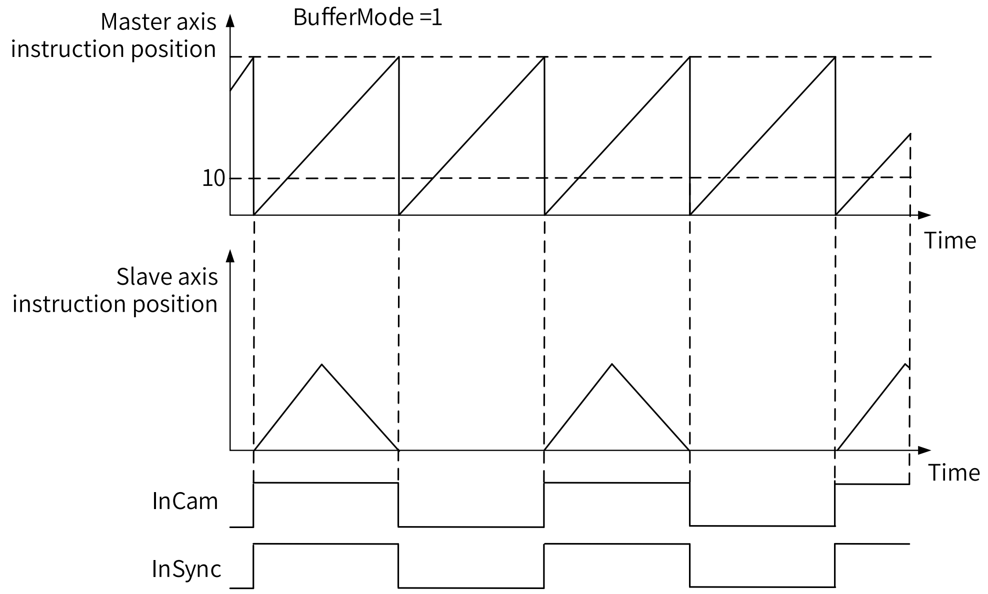
When the master axis is a bus servo axis or local pulse axis, you can use the instruction position of the previous cycle or the current cycle, or the feedback position of the current cycle.

Buffer Mode

When BufferMode is 1, the cam motion enters the data buffer mode.

In data buffer mode, when changes of input parameter values such as StarPosition and MasterStarDistance are received in a cam operation cycle, the changed parameter settings will take effect in the next cam cycle.





Note

In data buffer mode, if the cam cycle is synchronized with the rotation cycle of the master axis, the cam motion will be triggered at regular intervals.

Re-execution

If the MC_CamIn instruction is re-triggered while the Busy signal is still active, parameters including Periodic, MasterScaling, SlaveScaling, RefrenceType, and Direction are buffered and take effect in the next cam cycle.

Multi-execution

If a second MC_CamIn instruction is triggered while the Busy signal of the MC_CamIn instruction is still active, the Busy signal of the second instruction becomes active but the Active signal is inactive. When a cam cycle ends, the first instruction is aborted, the Active output of the second instruction becomes active, and the parameters (Periodic, MasterScaling, SlaveScaling, RefrenceType, and Direction) of the second instruction take effect.

3.10.2.3 MC_CamOut

MC_CamOut – End cam operation

Graphic Block

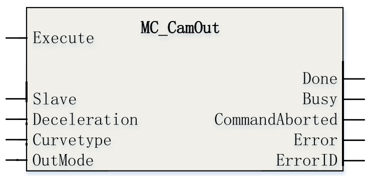
Instruction	Name	LD Expression	LiteST Expression
MC_CamOut	End cam operation		MC_CamOut(Execute := ???, Slave := ???, Deceleration := ???, CurveType := , OutMode := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–230 Instruction format

16-bit Instruc-tion	-					
32-bit Instruc-tion	MC_CamOut: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Slave	Slave axis Bus servo axis or local output axis	No	-		_sMCAXIS_INFO
S2	Decelera-tion	Deceleration Positive number: Stop according to deceleration 0: Stop immediately	No	-	Positive number or 0	REAL
S3	Curvetype	Curve type 0: T-shaped velocity curve	Yes	0		INT

S4	OutMode	Sync end mode 0: Decelerate to stop 1: Stop immediately after executing the current cycle	Yes	0	0 to 1	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	Comman-dAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	ON OFF	INT

Function Description

This instruction ends cam operation of the slave axis.

When Execute is set to ON, the MC_CamIn instruction is aborted, and the abortion flag is active. If OutMode is set to 0, the axis decelerates according to Deceleration. After it decelerates to 0, the Done output is active. The slave axis stays in ContinuousMotion state before it stops moving. If OutMode is set to 1, the axis stops immediately after the cam operation of the current cycle is completed. The slave axis stays in SynchronizedMotion state before the cam operation ends.

An error occurs when this instruction is executed on an axis that is not in cam operation.

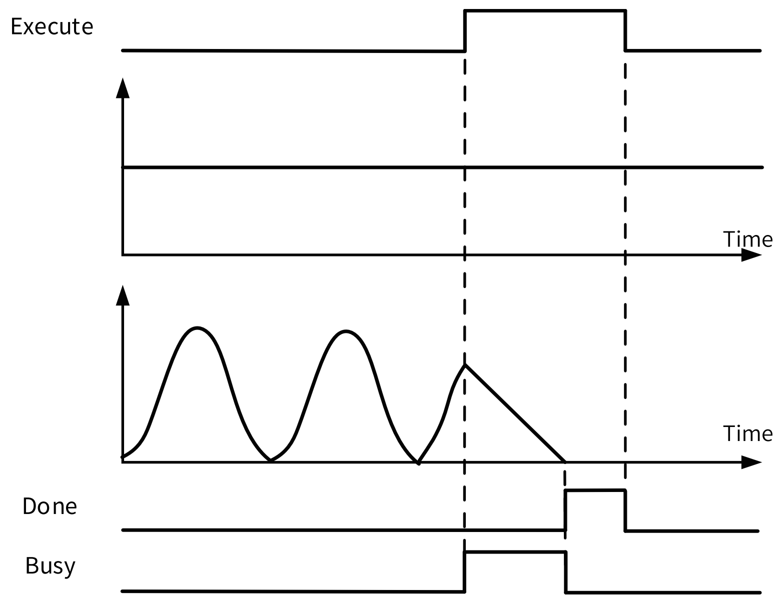
Re-triggering

When the MC_CamOut instruction is re-triggered, the axis stops according to the following rules:

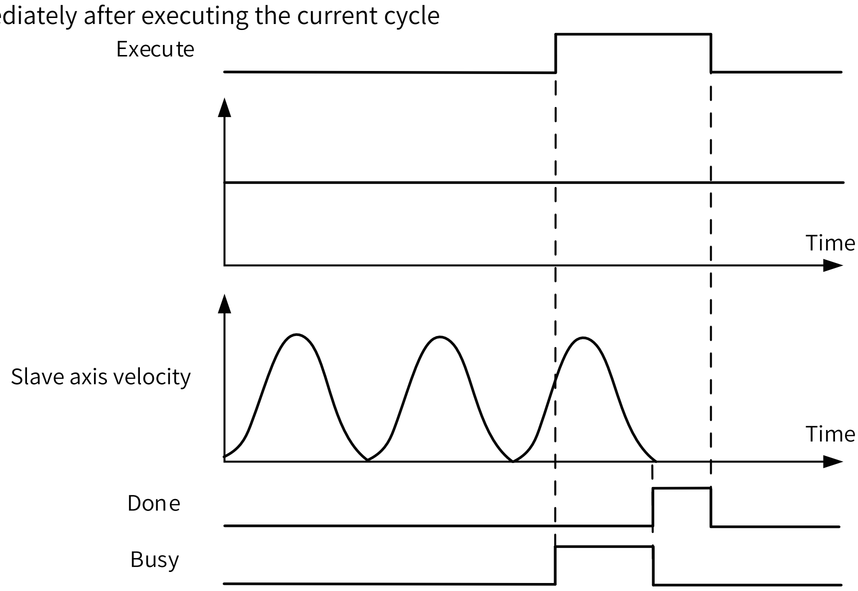
Initial Stop Mode	New Stop Mode	Execution Result
Decelerate to stop	Stop immediately after executing the current cycle	The instruction reports an error, and the axis decelerates to stop and then enters the StandStill state.
Decelerate to stop	Decelerate to stop	The axis stops according to the new deceleration.
Stop immediately after executing the current cycle	Decelerate to stop	The axis decelerates to stop.
Stop immediately after executing the current cycle	Stop immediately after executing the current cycle	The axis stops after executing the current cycle.

Timing Diagram

- Decelerate to stop



- Stop immediately after executing the current cycle



3.10.2.4 MC_GetCamTablePhase

MC_GetCamTablePhase – Obtain cam table phase

Graphic Block

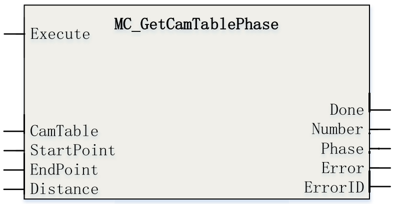
Instruction	Name	LD Expression	LiteST Expression
MC_ GetCamTablePhase	Obtain cam table phase		MC_GetCamTablePhase(Execute := ???, CamTable := , StartPoint := ???, EndPoint := ???, Distance := ???, Done => , Number => , Phase => , Error => , ErrorID =>);

Table 3–231 Instruction format

16-bit Instruction	-
32-bit Instruction	MC_GetCamTablePhase: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table Reserved	Yes	-	-	_sMC_CAMTABLE
S2	StartPoint	Start point	No	-		_sMC_CAM_NODE
S3	EndPoint	End point	No	-		_sMC_CAM_NODE
S4	Distance	Displacement of the slave axis	No	-	Positive number, negative number, or 0	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Number	Number of phases –1: Infinite number of identical solutions 0: None > 0: Actual number of phases	Yes	0	–1, 0 to 5	INT
D3	Phase	Obtained phase	Yes	0	Positive number or 0	REAL[6]
D4	Error	Error	Yes		ON OFF	BOOL
D5	ErrorID	Error code	Yes		ON OFF	INT

Function Description

This instruction is used to obtain the phase (Phase) of the master axis according to the displacement (Distance) of the slave axis between two cam key points.

If the cam curve is a straight line and is parallel to the X axis, the Distance specified in the instruction is on the straight line, the instruction output parameter Number outputs –1, and Phase[0] outputs the abscissa of the start point.

If the cam curve is a quintic curve, there may be multiple solutions. The instruction output parameter Number indicates the number of solutions, and the Phase array stores the specific values obtained.

If there is no solution, the output parameter Number is 0.

3.10.2.5 MC_GetCamTableDistance

MC_GetCamTableDistance – Obtain cam table displacement

Graphic Block

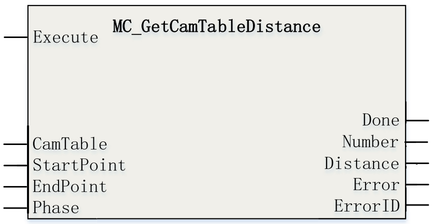
Instruction	Name	LD Expression	LiteST Expression
MC_ GetCamTable-Distance	Ob-tain cam table dis-plac-eme-nt		MC_GetCamTableDistance(Execute := ???, CamTable := , StartPoint := ???, EndPoint := ???, Phase := ???, Done => , Distance => , Error => , ErrorID =>);

Table 3–232 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_GetCamTablePhase: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table Reserved	Yes	-	-	_sMC_CAMTABLE
S2	StartPoint	Start point	No	-		_sMC_CAM_NODE
S3	EndPoint	End point	No	-		_sMC_CAM_NODE
S4	Phase	Phase of the master axis	No	-	Positive number, negative number, or 0	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL

D2	Distance	Obtained displacement of the slave axis	Yes	0	Positive number, negative number, or 0	TRAL
D3	Error	Error	Yes		ON OFF	BOOL
D4	ErrorID	Error code	Yes		ON OFF	INT

Function Description

This instruction is used to obtain the displacement (Distance) of the slave axis according to the phase (Phase) of the master axis between two cam key points.

3.10.2.6 MC_GearIn

MC_CamIn – Start cam operation

Graphic Block

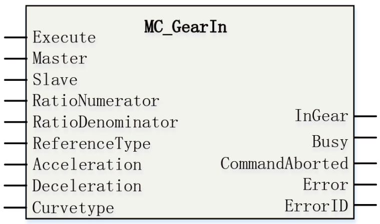
Instruction	Name	LD Expression	LiteST Expression
MC_GearIn	Start gear operation		MC_GearIn(Execute := ???, Master := ???, Slave := ???, RatioNumerator := , RatioDenominator := , ReferenceType := , Acceleration := , Deceleration := , CurveType := , InGear => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–233 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_GearIn: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Master	Master axis Bus servo axis, local pulse axis, bus encoder axis, or local encoder axis	No	-	-	_sMCAXIS_INFO _sENC_AXIS _sENC_EXT_AXIS _sMasterAxis
S2	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S3	RatioNumerator	Gear ratio (numerator)	Yes	1	Positive number Negative number	DINT
S4	RatioDenominator	Gear ratio (denominator)	Yes	1	Positive number	DINT
S5	ReferenceType	Position type 0: Instruction position of the previous task cycle 1: Instruction position of the current task cycle[1] 2: Feedback position of the current task cycle	Yes	0	0 to 2	INT
S6	Acceleration	Acceleration 0: No acceleration	No	-	0 Positive number	REAL
S7	Deceleration	Deceleration 0: No deceleration	Yes	Acceleration	0 Positive number	REAL
S8	Curvetype	Curve type 0: T-shaped acceleration curve	Yes	0	ON OFF	INT
D1	InGear	Gear ratio reached	Yes	-	ON OFF	BOOL
D2	Busy	Executing	Yes	-	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	-	ON OFF	BOOL
D4	Error	Error	Yes	-	ON OFF	BOOL
D5	ErrorID	Error code	Yes	-	ON OFF	INT

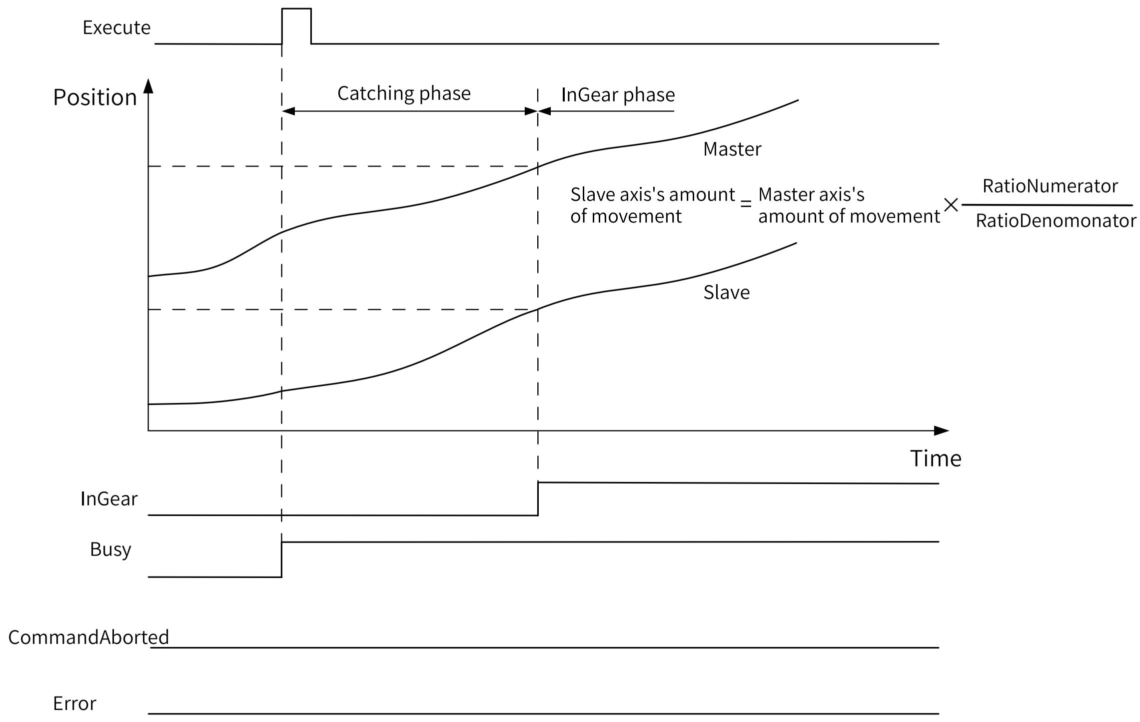
Note

[1]: When selecting the set position under the same task, make sure that the axis ID of the master axis is smaller than that of the slave axis.

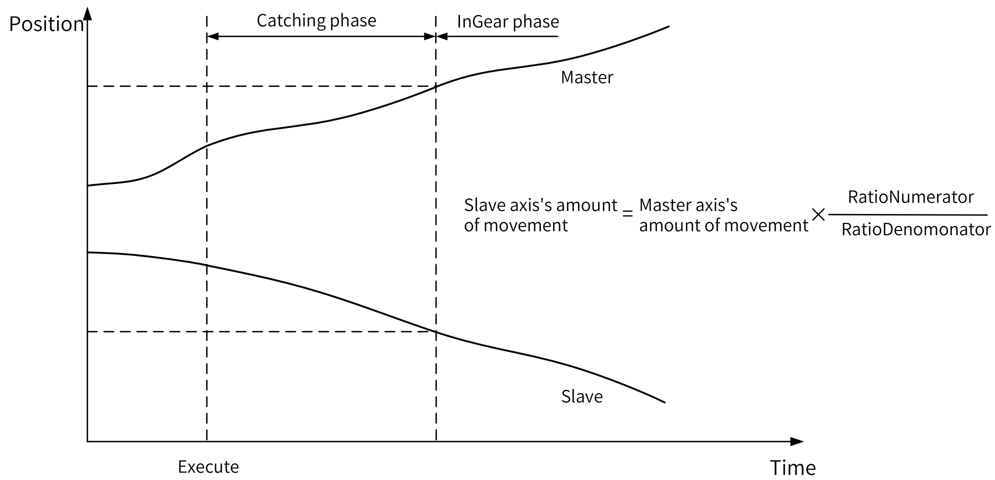
After coming into action, the slave axis uses the velocity obtained by multiplying the master axis velocity by the gear ratio as the target velocity to perform the acceleration and deceleration actions.

The phase is called Catching phase before the axis reaches the target position, and the InGear phase after the axis reaches the target position.

If the gear ratio is positive, the slave axis moves in the same direction as the master axis.

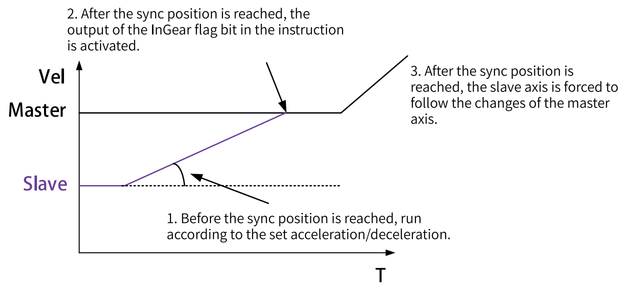


If the gear ratio is negative, the slave axis moves in the opposite direction from the master axis.

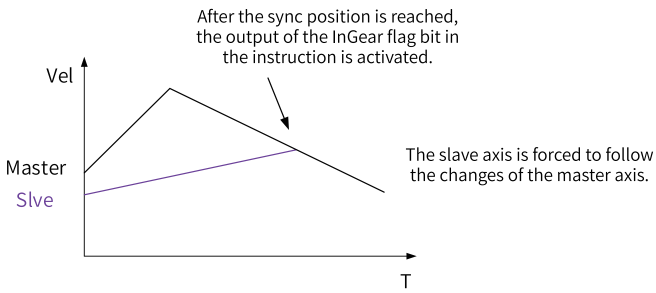


Before reaching synchronization, the slave axis moves at the set acceleration (deceleration). When the slave axis velocity is equal to the master axis velocity multiplied by the gear ratio, the gear is considered to be engaged. After that, the slave axis completely follows the changes of the master axis.

Scenario 1: Before synchronization, the master axis maintains a uniform motion (triggering the MC_GearIn instruction at the gear ratio of 1:1).



Scenario 2: Before synchronization, the master axis performs a variable motion (triggering the MC_GearIn instruction at the gear ratio of 1:1).



Filtering Function

During use, in order to reduce the velocity fluctuation of the master axis caused by the velocity fluctuation of the slave axis, you can adjust the velocity filtering coefficient of the master axis by setting the system variable fFilter parameter of the slave axis. The calculation formula is as follows:

MstVel = fFilter[0]*MstVel[0]+fFilter[1]*MstVel[1]+fFilter[2]*MstVel[2];

Where, MstVel is the synthetic velocity. fFilter is the filtering parameter, and the sum of three must be equal to 1. MstVel indicates the actual master axis velocity in the current cycle, previous cycle, and cycle before last, respectively.

Re-execution

When the MC_GearIn instruction is triggered again while the Busy signal of this instruction is still valid, the master axis velocity will be recalculated based on the gear ratio numerator and denominator. The slave axis will follow the calculation result and determine whether the InGear flag is set.

Multi-execution

When the MC_GearIn instruction is triggered again while the Busy signal of this instruction is still valid, the Busy signal of the second instruction is valid, interrupting the first instruction. At the same time, the master axis velocity will be recalculated based on the gear ratio numerator and denominator. The slave axis will follow the calculation result and determine whether the InGear flag is set.

3.10.2.7 MC_GearOut

MC_GearOut – End gear operation

Graphic Block

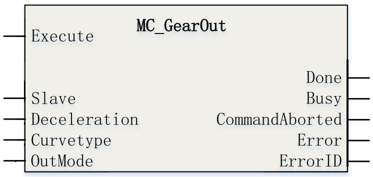
Instruction	Name	LD Expression	LiteST Expression
MC_GearOut	End gear operation		MC_GearOut(Execute := ???, Slave := ???, Deceleration := ???, CurveType := , OutMode := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–234 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_Gearout: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S2	Deceleration	Deceleration	No	-	Positive number or 0	REAL
S3	Curvetype	Curve type 0: T-shaped velocity curve	Yes	0	0	INT
S4	OutMode	Sync end mode 0: Decelerate to stop	Yes	0	0	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAbort-ed	Abortion of execution	Yes	OFF	ON OFF	BOOL

D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

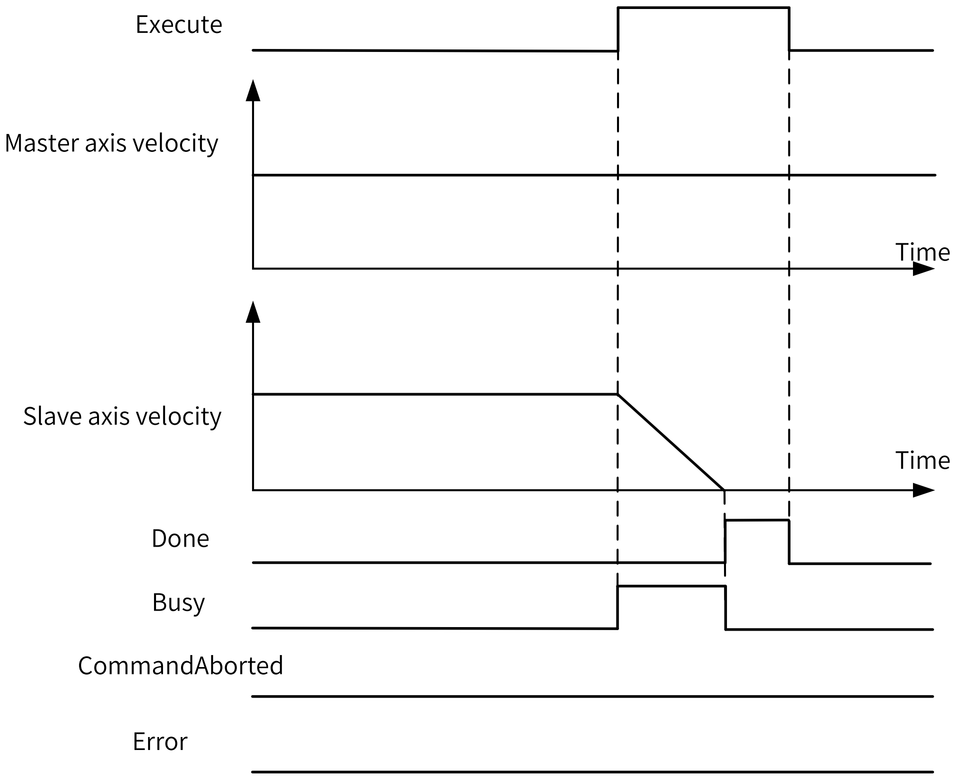
Function Description

The MC_GearOut instruction aborts execution of the MC_GearIn (start gear operation) instruction for the operation axis specified by Slave at the deceleration specified by Deceleration.

This instruction does not affect the MC_GearIn (start gear operation) operation of the master axis.

Timing Diagram

Deceleration to stop



3.10.2.8 MC_Phasing

MC_Phasing – Master axis phase shifting

Graphic Block

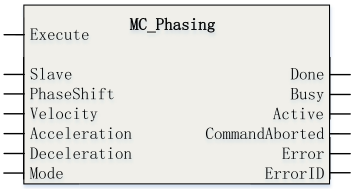
<table>
<tr><th>Instruction</th><th>Name</th><th>LD Expression</th><th>LiteST Expression</th></tr>
<tr><td>MC_Phasing</td><td>Master axis phase shifting</td><td>
</td><td>MC_Phasing(Execute := ???,
Slave := ???,
PhaseShift := ???,
Velocity := ???,
Acceleration := ???,
Deceleration := ,
Mode := ,
Done => ,
Busy => ,
Active => ,
CommandAborted => ,
Error => ,
ErrorID =>);</td></tr>
</table>

Table 3–235 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_Phasing: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_INFO
S2	PhaseShift	Phase compensation	No	-	Positive number 0 Negative number	REAL
S3	Velocity	Target velocity	No	-	Positive number	REAL
S4	Acceleration	Acceleration	No	-	Positive number	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	Positive number	REAL
S6	Mode	Mode 0: Reserved 1: Pause when the velocity of the master axis is 0	Yes	0	0 to 1	INT

D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	Active	Executing instruction	Yes	OFF	ON OFF	BOOL
D4	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	ON OFF	INT

Function Description

If the MC_Phasing instruction is executed when single-axis synchronized control is in progress, the phase of the master axis is shifted according to the settings of PhaseShift (phase shift), Velocity (target velocity), Acceleration (acceleration), and Deceleration (deceleration).

- When working with cam operation, this instruction can be called only after the MC_CamIn instruction is called. When InSync of the MC_CamIn instruction is OFF, the MC_Phasing instruction is in the buffered state, in which the Busy signal is active but the Active signal output is inactive. When InSync of the MC_CamIn instruction becomes ON, the cam is fully engage. At this time, the Active signal output of the MC_Phasing instruction becomes active, and phase shifting starts.
- When working with gear operation, this instruction can be called only after the MC_GearIn instruction is called. The MC_GearIn instruction is triggered first to establish a gear relationship between the master and slave axes. After the slave axis enters the SynchronizedMotion state, the MC_Phasing instruction is triggered and starts to perform the corresponding shifting operation.

During execution, the set position (feedback position) of the master axis does not change, and the relative shift compensated for the set position (feedback position) is taken as the phase of the master axis. The slave axis is synchronized to the shifted master axis phase.

The Done signal changes to ON when the PhaseShift (phase shift) is reached.

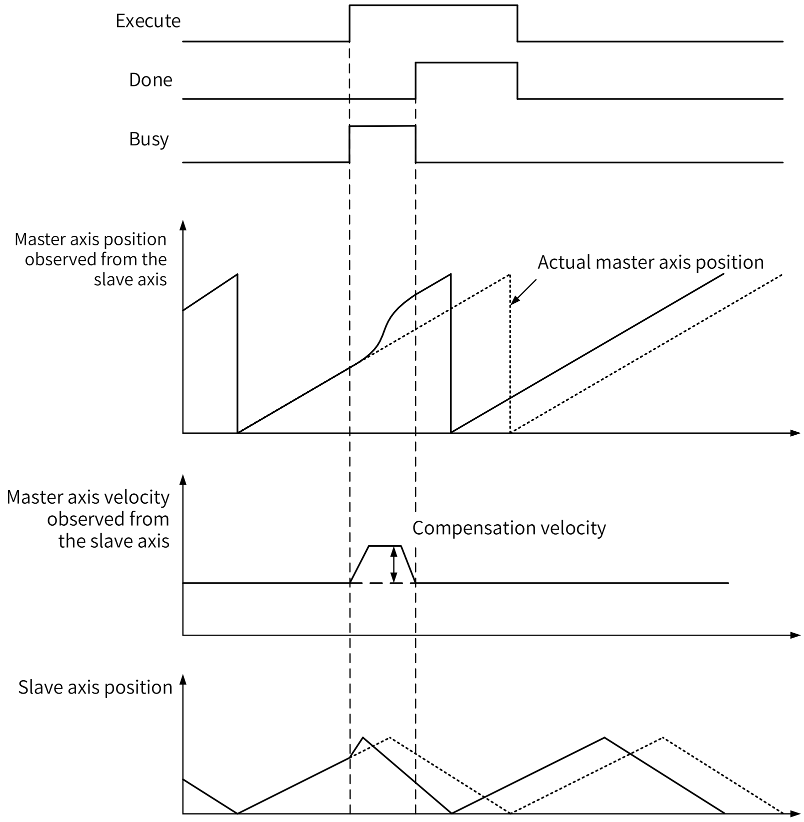
Shifting ends when execution of the synchronized control instruction is completed. If a synchronized control instruction is executed again, the previous amount of shift is not affected.

You can shift the phase of the master axis for the following synchronized control instructions: MC_CamIn (start cam operation) and MC_GearIn (start gear operation).

Control Mode Selection

When Mode is set to 1, if the master axis stops running (the velocity of the master axis is 0), phase shifting automatically stops. When the master axis starts running again, phase shifting continues from the original position where it was suspended.

Timing Diagram



3.10.2.9 MC_SaveCamTable

MC_SaveCamTable – Save cam table

Graphic Block

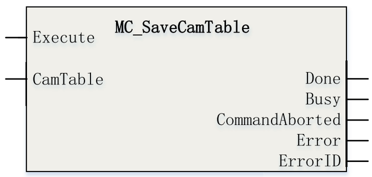
Instruction	Name	LD Expression	LiteST Expression
MC_SaveCamTable	Save cam table		MC_SaveCamTable(Execute := ???, CamTable := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–236 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_SaveCamTable: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table	No	-		_sMC_ CAMTABLE

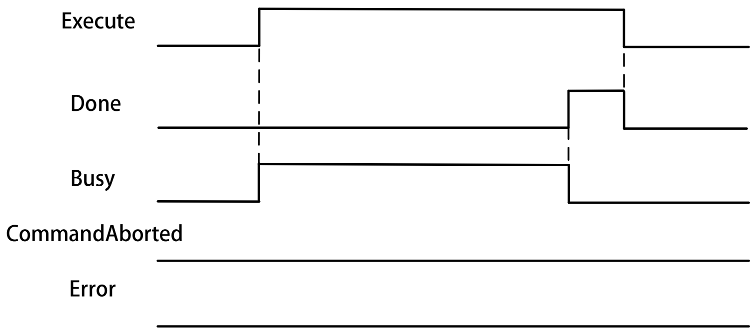
D1	Done	Completion flag	Yes	-	ON OFF	BOOL
D2	Busy	Executing	Yes		ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes		ON OFF	BOOL
D4	Error	Error	Yes		ON OFF	BOOL
D5	ErrorID	Error code	Yes		ON OFF	INT

Function Description

This instruction saves the cam table specified by CamTable to non-volatile memory on the rising edge of the Execute input.

Do not turn off the power supply of the controller during execution of this instruction. Otherwise, data saving may fail, which results in cam data loss.

Timing Diagram



3.10.2.10 MC_GenerateCamTable

MC_GenerateCamTable – Update cam table

Graphic Block

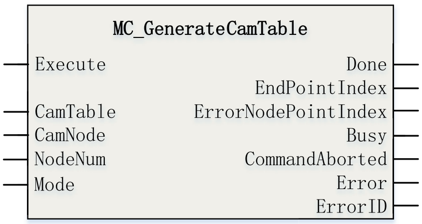
Instruction	Name	LD Expression	LiteST Expression
MC_ Generate-CamTable	Update cam table		MC_GenerateCamTable(Execute := ???, CamTable := ???, CamNode := , NodeNum := , Mode := , Done => , EndPointIndex => , ErrorNodePointIndex => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–237 Instruction format

16-bit Instruc-tion	-					
32-bit Instruc-tion	MC_GenerateCamTable: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	CamTable	Cam table	No	-		_sMC_ CAMTABLE
S2	CamNode	Cam node array If it is left empty, the original cam node array is used.	Yes	-		_sMC_CAM_ NODE
S3	NodeNum	Number of cam nodes If it is left empty, the original cam node quantity is used.	Yes	-	2 to 361	INT
S4	Mode	Effective mode 0: Effective upon the next cam cycle Others: Reserved	Yes	0	0	INT
D1	Done	Completion flag	Yes	-	ON OFF	BOOL
D2	EndPointIndex	End point index	Yes	0	0 to 360	INT
D3	ErrorNodePointIn-dex	Error node number	Yes	0	0 to 360	INT
D4	Busy	Executing	Yes	OFF	ON OFF	BOOL

D5	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	OFF	ON OFF	INT

Function Description

The MC_GenerateCamTable instruction calculates cam data based on input variables CamNode and CamNum on the rising edge of Execute and updates the data to the cam table specified by CamTable. The update takes effect upon the next cam cycle.

Function of CamNode

CamNode specifies whether to use a new cam node array. When it is empty, the original cam table node array specified by CamTable is adopted. When it is not empty, the cam node array specified by CamNode is adopted.

- **CamNode is empty.**
  You can modify the value of the cam node array in the cam table by using the PLC program and make the modification take effect in the next cam cycle by executing the MC_GenerateCamTable instruction.

Cam node array A

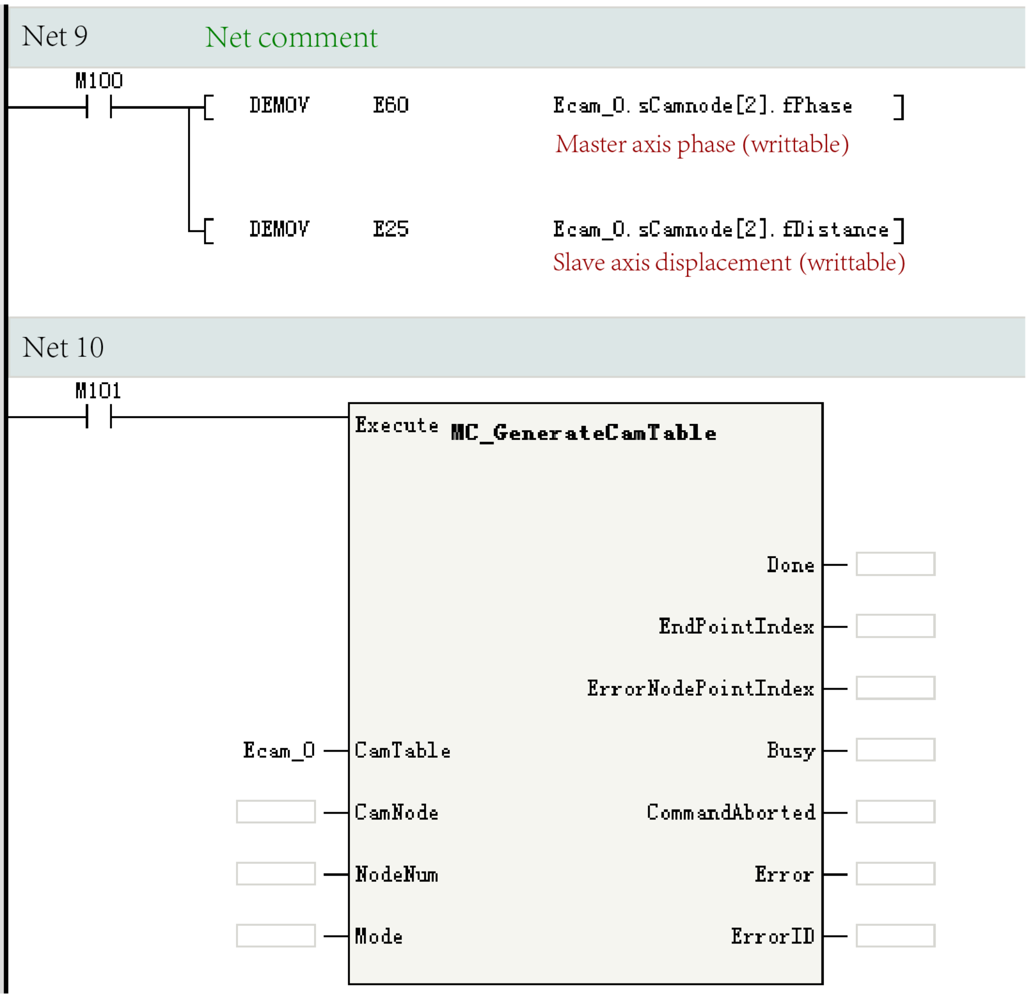
	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
End point	160	0
	0	0
	0	0

Modify key points only.

Cam node array A

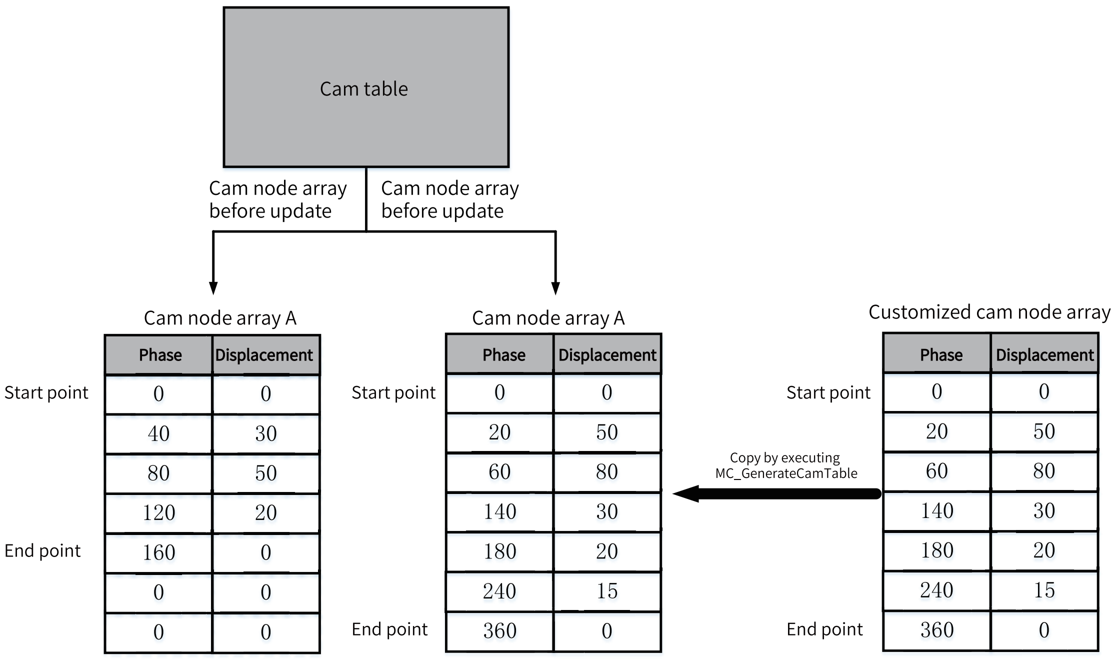
	Phase	Displacement
Start point	0	0
	40	30
	60	25
	120	20
End point	160	0
	0	0
	0	0

The program example is as follows:



- **CamNode is specified.**

  You can create a new cam node array by using the PLC program and copy the value in this array to the cam table by executing the MC_GenerateCamTable instruction so that the value is executed upon the next cam cycle.



Cam node array A

	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
End point	160	0
	0	0
	0	0

Cam node array A

	Phase	Displacement
Start point	0	0
	20	50
	60	80
	140	30
	180	20
	240	15
End point	360	0

Customized cam node array

	Phase	Displacement
Start point	0	0
	20	50
	60	80
	140	30
	180	20
	240	15
End point	360	0

The program example is as follows:

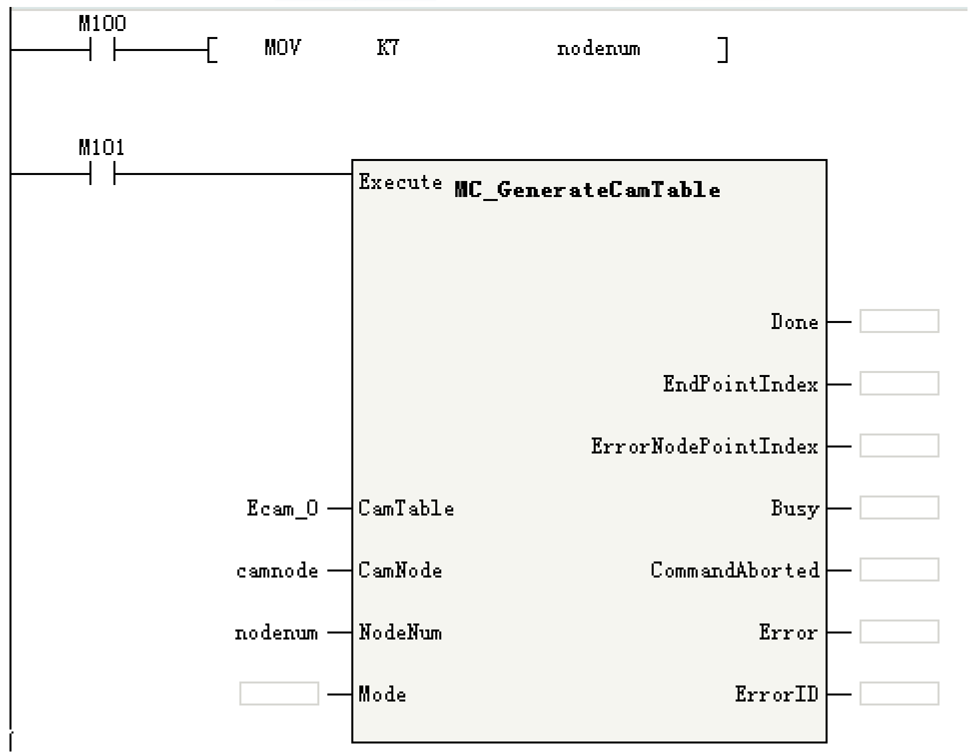
Cam node array A

	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
End point	160	0
	0	0
	0	0

Add key points.

Cam node array A

	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
	160	0
	240	15
End point	360	0



Function of NodeNum

NodeNum specifies the number of nodes in the newly generated cam table. When it is empty, the number of nodes in the cam table remains unchanged. When it is not empty, the node quantity specified by NodeNum is adopted.

You can modify the number of key points in the cam table and make the modification take effect in the next cam cycle by executing the MC_GenerateCamTable instruction.

Cam node array A

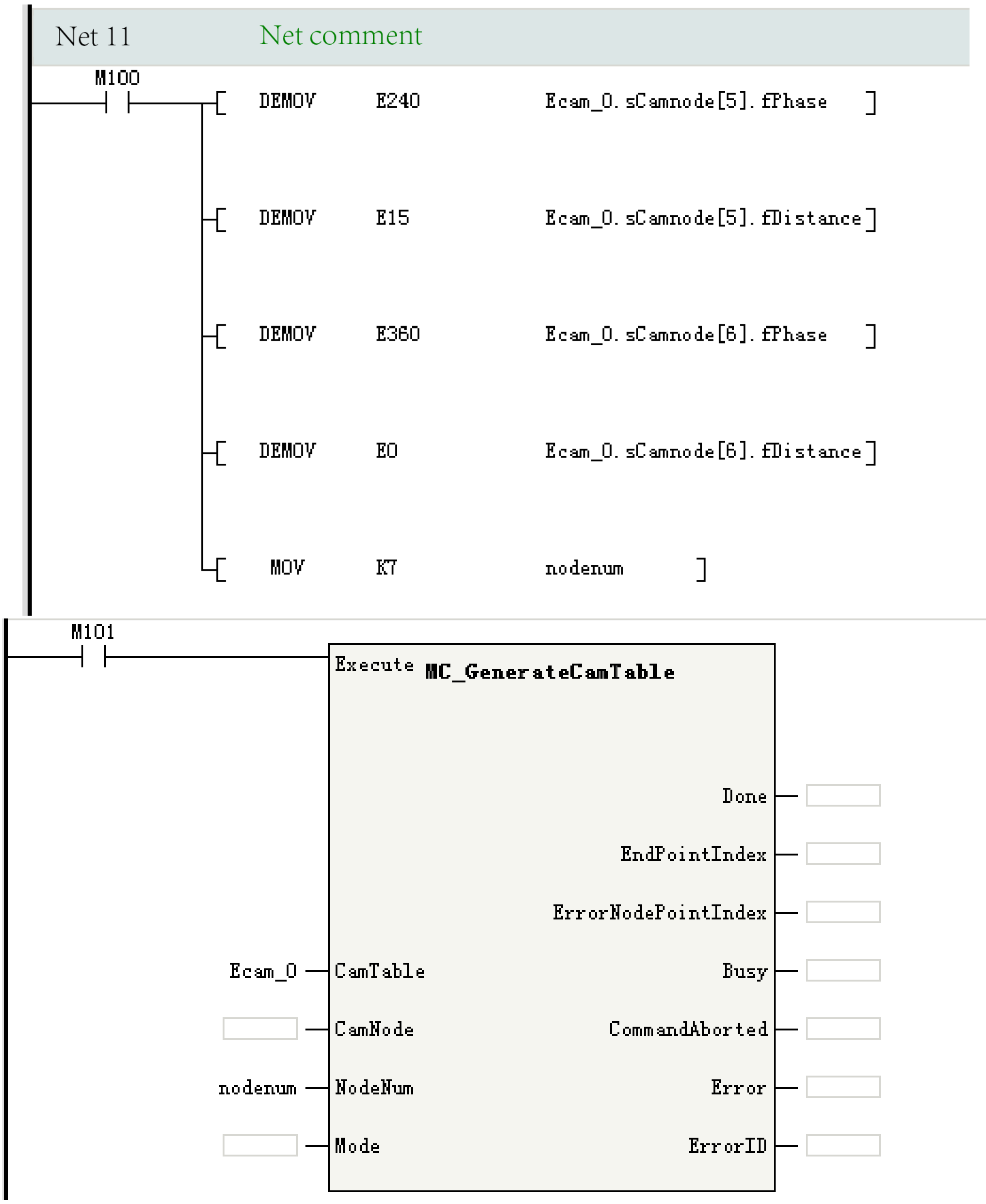
	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
End point	160	0
	0	0
	0	0

Add key points.

Cam node array A

	Phase	Displacement
Start point	0	0
	40	30
	80	50
	120	20
	160	0
	240	15
End point	360	0

The program example is as follows:



Parameter Check

This instruction first checks the cam table data when it is called.

- Both the phase and displacement of the first point must be 0; otherwise, the instruction reports an error.
- The absolute values of the phase, displacement, and velocity ratio cannot be greater than 9999999; otherwise, the instruction reports an error.
- The node quantity cannot be greater than 361; otherwise, the instruction reports an error.
- The node quantity cannot be less than 2; otherwise, the instruction reports an error.
- The phases must be sorted in ascending order; otherwise, the instruction reports an error.
- The difference between two adjacent master axis phases must be greater than 0.0001; otherwise, the instruction reports an error.
- The node curve type is set to linear or quintic curve; otherwise, the instruction reports an error.

Velocity Ratio Adjustment Rules

If the velocity ratio of key points is improper, this instruction will automatically adjust the velocity ratio of cam nodes based on the following rules:

- If the current segment is a straight line, the velocity ratio is automatically adjusted according to the formula.
  For example, if the curve between points A1 and A2 is a straight line, the calculated velocity ratio is written to A2.

  The coordinates of A1 are (x1, y1), the coordinates of A2 are (x2, y2), then the velocity ratio of straight line A1-A2 is as follows: V2 = |y2 – y1|/|x2 – x1|.

- If a quintic curve is followed by a straight line, adjustment is required to ensure the continuity of the velocity ratio at the link point between the quintic curve and the straight line and prevent jumping. Assume that the curve between points A1 and A2 is a quintic curve, and that between points A2 and A3 is a straight line.

  The velocity ratio of the straight line segment is calculated and written to A3. Then the velocity ratio of the end point of the quintic curve is adjusted and written to A2.

  The coordinates of A2 are (x2, y2), the coordinates of A3 are (x3, y3), then the velocity ratio of straight line A2-A3 is as follows: V3 = |y3 – y2|/|x3 – x2|.

  The velocity ratio of point A2 is set to a value same as that of point A3.

- If a quintic curve is followed by a quintic curve, no adjustment is required.
- If a straight line is followed by a straight line, the link velocity of each segment needs to be calculated separately. In this case, sudden change in the link velocity ratio is allowed.
  For example, assume that A1-A2 is straight line segment 1, and A2-A3 is straight line segment 2. The velocity ratio of segment 1 is calculated first and written to A2, and then the link velocity of segment 2 is calculated and written to A3. In this case, there is a sudden change in velocity caused by unequal link velocities between segment 1 and segment 2.

Re-execution

If this instruction is re-triggered while the Busy signal is still active, the cam table is modified according to the new parameters.

Multi-execution

If a new MC_GenerateCamTable instruction is triggered while the Busy signal of the current MC_ GenerateCamTable instruction is still active, the current instruction is aborted, the CommandAborted signal output becomes active, and the cam table is modified according to the parameters of the newly triggered instruction.

3.10.2.11 MC_DigitalCamSwitch

MC_DigitalCamSwitch – Tappet control

Graphic Block

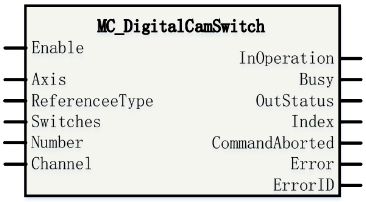
Instruction	Name	LD Expression	LiteST Expression
MC_DigitalCamSwitch	Electronic cam tappet control		MC_DigitalCamSwitch(Enable := ???, Axis := ???, ReferenceType := , Switches := ???, Number := ???, Channel := ???, InOperation => ,

Table 3–238 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_DigitalCamSwitche: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-	-	_sMCAXIS_ INFO _sENC_AXIS
S2	ReferenceType	Position type 0: Set position of the previous cycle 1: Set position of the current cycle 2: Feedback position of the current cycle 3: Phase of the master axis when the axis specified by Axis works as the cam slave axis	Yes	0	0 to 3	INT
S3	Switches	Switch	No	-	-	_sMC_ DigitalSwitch [1-32]
S4	Number	Quantity	No	-	1 to 32	INT
S5	Channel	Tappet terminal 0–13 indicate actual terminal. 1000–1007 indicate virtual tappets.	No	-	-	INT
D1	InOperation	Executing tappet	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D4	OutStatus	Output state	Yes	OFF	ON/OFF	BOOL
D5	Index	Index Comparison point to be executed	Yes	0	0 to 31	INT

16-bit Instruction	-					
32-bit Instruction	MC_DigitalCamSwitche: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
D6	CommandA-borted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D7	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D8	ErrorID	Fault code	Yes	0	-	INT16

Function Description

This instruction works with the cam to implement the tappet function. Switches specifies the tappet output points, and Channel specifies the tappet terminal. The instruction allows a DI terminal of the body but not an expansion module to be used as a tappet terminal.

- The instruction latches the input parameters on the left on the rising edge of Enable to execute the tappet output comparison function.
- When Enable is ON, modifications on the input parameters on the left do not take effect. The Enable signal must remain ON during the entire tappet output process.
- The instruction stops comparison output on the falling edge of Enable and aborts the tappet terminal that outputs ON.

Selecting tappet terminal source

The tappet terminal is specified by the variable Channel. 0–7 correspond to Y00–Y07, 8–13 corresponding to Y10–Y15, and 1000–1007 are virtual tappets, which are only counted as tappets but not output to actual hardware terminals.

Setting tappet points

Tappet comparison points are specified by the variable Switches. This variable is a _sMC_DigitalSwitch structure array.

Variable	Data Type	Description
fPosition	REAL	Absolute position for the output to turn ON
iMode	INT	Switch mode 0: Disabled 1: Position type 2: Time type
iDirection	INT	Direction of the master axis 0: Forward 1: Reverse 2: No direction
fParameter	REAL	Position type: ON end position Time type: ON output time (unit: ms). The decimal part after the decimal point is ignored in time mode, and the value cannot be greater than 10000 ms.

Note

The ON start point of the tappet point array in one direction must remain unique. For example, in the set of points of which iDirection is 0/2, fPosition must be unique.

Start and end of tappet comparison output

After the tappet instruction is executed, it internally sorts the tappet point array, determines the tappet point closest to the current position of the axis, and sets the tappet to ON immediately when the axis moves to this point.

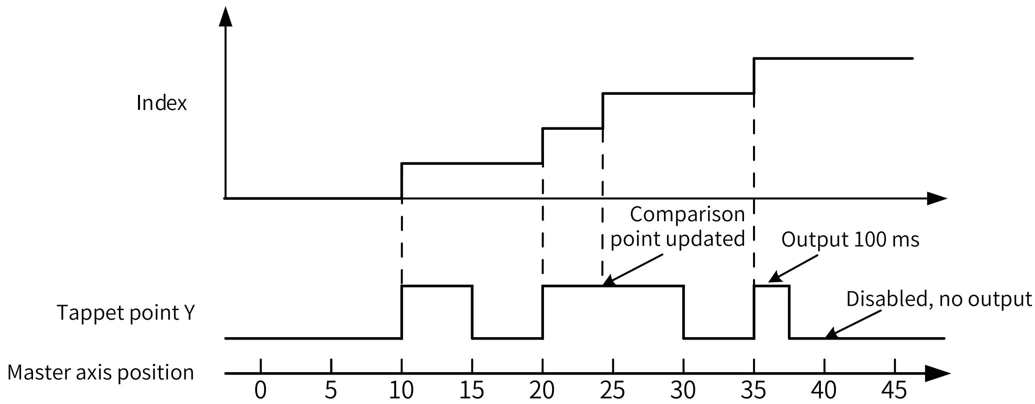
iMode specifies the comparison mode.

- When iMode is set to 0, the comparison point is disabled. When the axis passes this point, the tappet has no output.
- When iMode is set to 1, fParameter specifies the position where the tappet output turns OFF.
- When iMode is set to 2, fParameter specifies the time duration (in ms) during which the tappet output remains ON.

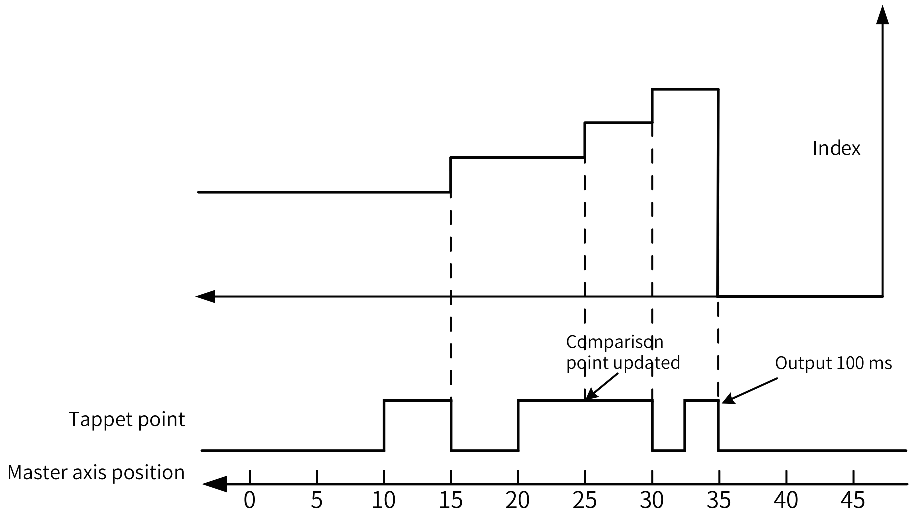
iDirection specifies the running direction of the master axis. It needs to work with iMode.

No.	fPosition	fParameter	iMode	iDirection
0	10	15	1	0
1	20	25	1	0
2	24	30	1	0
3	35	100	2	0
4	40	45	0	0
5	10	15	1	1
6	20	25	1	1
7	24	30	1	1
8	35	100	2	1

When the master axis runs in forward direction:



When the master axis runs in reverse direction:

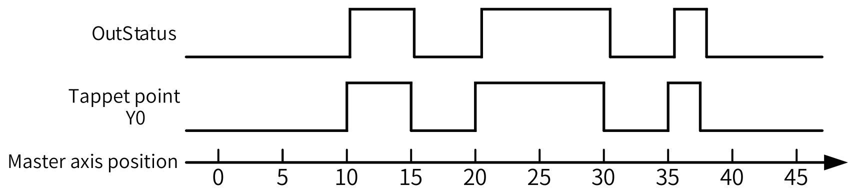


Note

When the width of the tappet output ON is less than one EtherCAT cycle, the actual output terminal will last for one EtherCAT cycle.

Tappet output status monitoring

OutStatus indicates the tappet output state. In the main task, this variable is used to monitor the tappet output state. Assume that the tappet point is set to Y0.



Re-triggering

This instruction is an instruction controlled by Enable and does not involve re-triggering.

Multi-execution

If the values of Channel of two MC_DigitalCamSwitch instructions are the same, and the second instruction is triggered while the Busy signal of the first instruction is still active, the first instruction is aborted and the tappet point is output under the control of the second tappet instruction.

3.10.2.12 MC_GearInPos

MC_GearInPos – Start the gear operation at the specified position

Graphic Block

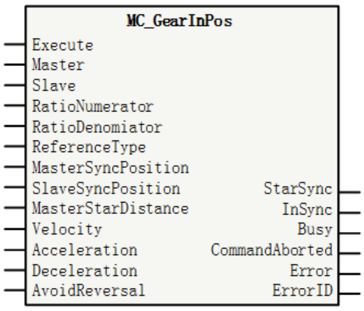
Instruction	Name	LD Expression	LiteST Expression
MC_GearInPos	Start the gear operation at the specified position		MC_GearInPos(Execute := ??? , Master := ???, Slave :=??? , RationNumberator := , RationDenominator := , ReferenceType := , MasterSyncPosition :=??? , SlaveSyncPosition :=??? , MasterStarDistance :=??? , Velocity := , Acceleration := , Deceleration := , AvoidReversal := , StartSync => , InSync =>, Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–239 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_GearInPos: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Master	Master axis Bus servo axis, local pulse axis, bus encoder axis, or local encoder axis	No	-	-	_sMCAXIS_ INFO _sENC_AXIS_ sENC_EXT_ AXIS _sMasterAxis
S2	Slave	Slave axis Bus servo axis or local pulse axis	No	-	-	_sMCAXIS_ INFO
S3	RatioNumerator	Gear ratio (numerator)	Yes	1	Positive or negative number	DINT

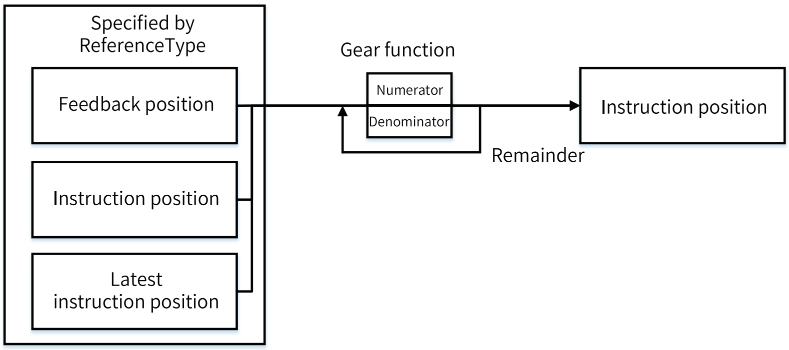
16-bit Instruction	-					
32-bit Instruction	MC_GearInPos: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S4	RatioDenominator	Gear ratio (denominator)	Yes	1	Positive number	DINT
S5	ReferenceType	Position type 0: Set position of the previous cycle 1: Set position of the current cycle 2: Feedback position of the current cycle	Yes	0	0 to 2	INT
S6	MasterSyncPosition	Position of the master axis	No	-	Positive number, negative number, or 0	REAL
S7	SlaveSyncPosition	Sync position of the slave axis	No	-	Positive number, negative number, or 0	REAL
S8	MasterStarDistance	Movement distance of the master axis in the Catching phase	No	-	Positive number or 0	REAL
S9	Velocity	Velocity (Reserved)	Yes	1	Positive number	REAL
S10	Acceleration	Acceleration (Reserved)	Yes	0	Positive number or 0	REAL
S11	Deceleration	Deceleration (Reserved)	Yes	Acceleration	Positive number or 0	REAL
S12	AvoidReversal	Reversal prohibited (Reserved)	Yes	0	0	INT
D1	InSync	Synchronizing	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D4	CommandAborted	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D5	Error	Error	Yes	OFF	ON/OFF	BOOL
D6	ErrorID	Error code	Yes	0	-	INT16

Function Description

This instruction specifies the axis of the operation object through Slave (slave axis). According to the input parameters RatioNumerator, RatioDenominator, ReferenceType, MasterSyncPosition, SlaveSyncPosition, MasterStarDistance, motion planning is performed on the slave axis to finally implement gear operation.

The instruction allows you to select the master axis position (Master) through ReferenceType:

- _mcCommand: Instruction position (value calculated in the latest task cycle). For the current cycle, use the master axis instruction position calculated during the previous task cycle. In the fixed-cycle task before the master axis instruction position is calculated, use the calculated master axis instruction position.
- _mcLatestCommand: Instruction position (value calculated under the same task cycle). Use the master axis instruction position calculated in the same task cycle.
- _mcFeedback: Value obtained during the same task cycle. Use the master axis feedback position obtained in the same task cycle.



The interval from the point that the slave axis starts catching up to the point that the axis reach the sync position is called Catching phase. After the axis reaches the target position, the phase is called InGear phase. After gear synchronization, the slave axis moves synchronously with the master axis at any interval.

The nature of the motion process in the Catching phase is that the slave axis follows an electronic cam of the master axis. At this time, based on the master axis range (MasterCatchPosition, MasterSyncPosition) and the slave axis range (slave axis position when the Catching phase is triggered, SlaveSyncPosition), the instruction will plan a quintic cam curve based on the set gear ratio, velocity ratio of the master and slave axes when the Catching phase is triggered, and the above position parameters, so that the slave axis follow the master axis to complete the cam movement in the Catching phase.

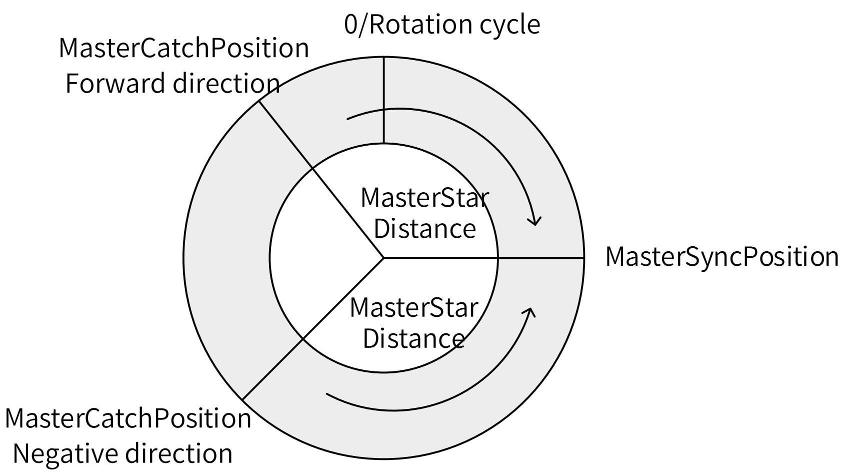
The instruction determines the catch-up start position of the master axis (MasterCatchPosition) based on the movement direction of the master axis at the instruction start time. When the master axis is a linear axis:

The master axis movement direction is positive, and MasterCatchPosition = MasterSyncPosition – MasterStartDistance.

The master axis movement direction is negative, and MasterCatchPosition = MasterSyncPosition + MasterStartDistance.

If the instruction determines that the current position and movement direction of the master axis do not allow the master axis to reach the catch-up start position to trigger the gear Catching phase, it reports the fault code 9304.

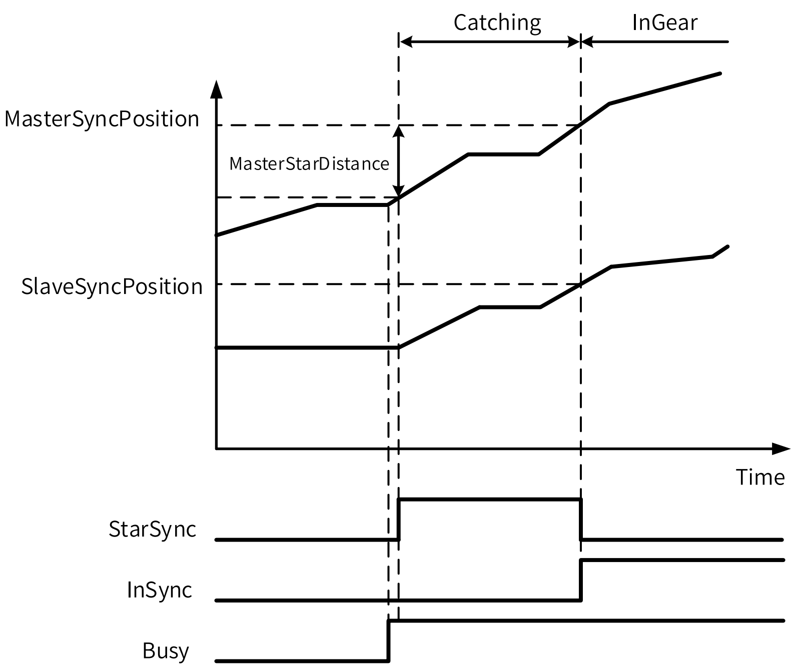
When the master axis is in rotation mode, the calculation principle of its catch-up start position is shown in the following figure.



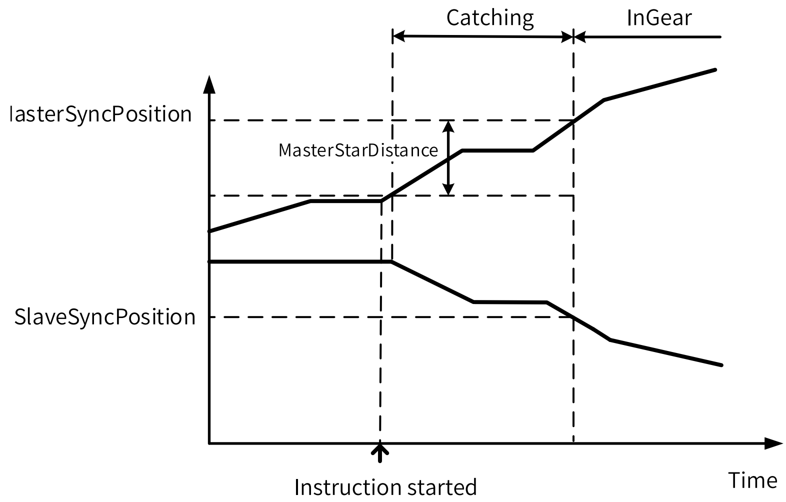
When the master axis velocity is 0 at the instruction start time, the catch-up start position cannot be determined, and the instruction reports the fault code 9301.

After gear synchronization, the slave axis multiplies the master axis velocity by the gear ratio (RatioNumerator/RatioDenominator) to obtain the target velocity, and synchronizes the acceleration and deceleration actions with the master axis.

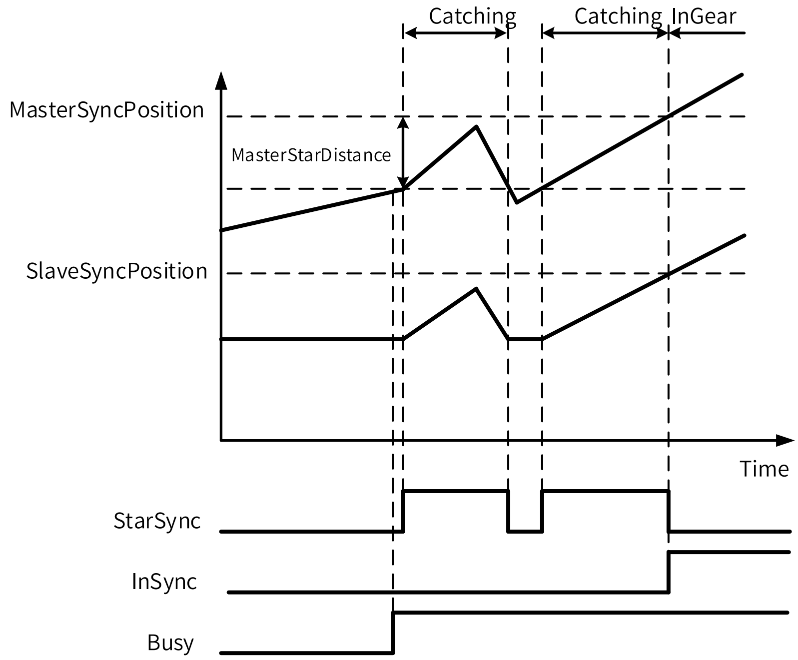
When the gear ratio is positive, the slave axis moves in the same direction as the master axis after it reaches the sync position.



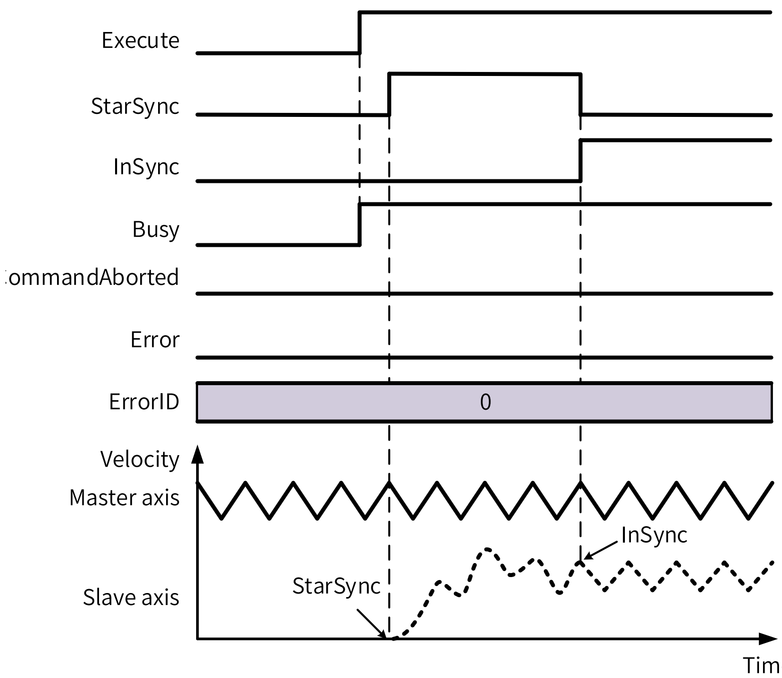
When the gear ratio is negative, the slave axis moves in the opposite direction of the master axis after it reaches the sync position.



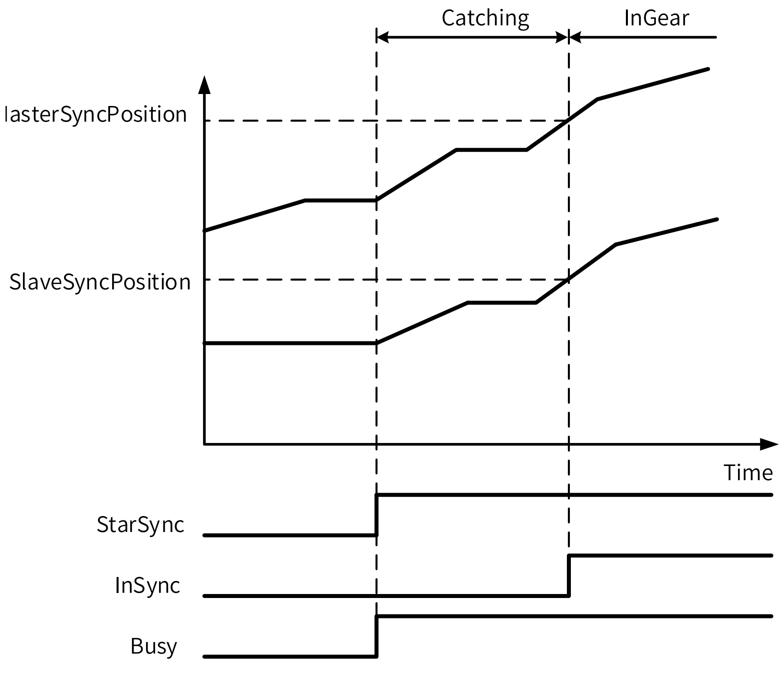
In the catching process, if the master axis position deviates from the catching area due to vibration, the instruction will exit the Catching phase and StarSync will be set to FALSE.



In addition, when the master axis velocity varies greatly in different cycles, the slave axis velocity is also not fixed. Before InSync of gear synchronization is reached, the slave axis also tracks the master axis position. The following figure shows the details.



If MasterStarDistance is set to 0, the instruction immediately enters the Catching phase while starting Execute, StartSync is set to TRUE, and the slave axis starts the gear catching action.



Re-triggering

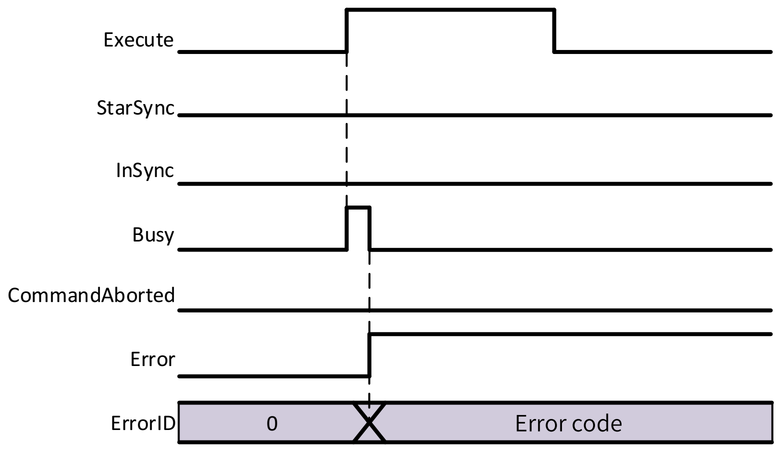
If the MC_GearInPos instruction is triggered again when the Busy signal of the previous MC_GearInPos instruction is valid and StarSync is invalid, the slave axis motion from the Catching phase to the InSync phase will be re-planned based on the input parameters, and the StarSync and InGear flag bits will be reset. If the axis already enters the Catching phase and StarSync is ON, you need to call MC_GearOut to restart the MC_GearInPos instruction, otherwise the instruction reports the fault code 9303.

Multi-execution

If the MC_GearInPos instruction is triggered again when the Busy signal of the previous MC_GearInPos instruction is valid and StarSync is invalid, the Busy signal of the second MC_GearInPos instruction is valid, the first MC_GearInPos instruction is interrupted, and the slave axis motion from the Catching phase to the InSync phase will be re-planned based on the input parameters. If the first MC_GearInPos instruction already enters the Catching phase and StarSync is ON, you need to call MC_GearOut and then trigger the second MC_GearInPos instruction, otherwise the instruction reports the fault code 9303.

Abnormalities

When the instruction encounters an abnormality, the timing diagram is as follows:



It is prohibited to execute the MC_SetPositionon instruction on the master and slave axes during execution of the MC_GearInPos instruction. When the MC_SetPosition instruction is being executed on the master axis, slave axis rapid tracking may occur, which is dangerous. Unbind the master axis from the slave axis before executing the MC_SetPosition instruction on the master or slave axis.

3.10.2.13 Fault Codes

When a fault occurs during use of the electronic cam functions, refer to the fault codes listed in the following table for troubleshooting.

Table 3–240 Fault codes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9200	Failed to obtain the cam table configuration file.	1. Check whether the board software and background software match. 2. Re-download the cam configuration table.	No
9201	Failed to obtain the master axis.	1. Check whether the master axis called in the program exists. 2. Check whether the master axis has reported an error.	Yes
9202	Failed to obtain the slave axis.	1. Check whether the slave axis called in the program exists. 2. Check whether the slave axis has reported an error.	No
9203	Failed to obtain the cam table.	Check whether the cam table called exists.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9204	The number of cams executed simultaneously in the PLC program exceeds the maximum allowable value.	Check whether the number of cams executed simultaneously in the program exceeds the threshold.	Yes
9205	The corresponding cam node is not found.	The specified slave axis is not a cam node. Set the parameters again.	Yes
9206	The master axis is changed during cam engagement.	Do not change the master axis during cam engagement.	Yes
9207	StartMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9208	StartPosition of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9209	MasterStartDistance of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9210	MasterScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9211	SlaveScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9212	MasterOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9213	SlaveOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9214	MasterScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.	Yes
9215	SlaveScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.	Yes
9216	ReferenceType of the MC_CamIn/MC_GearIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9217	Direction of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9218	BufferMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9219	The master axis phases in the node array of the cam table are not sorted in ascending order.	Sort the master axis phases in ascending order when customizing cam table nodes.	Yes
9220	The curve type setting of the node array of the cam table is out of range.	Check whether the curve type of the cam node array is set incorrectly.	Yes
9221	The target deceleration of the MC_CamOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9222	The target deceleration of the MC_CamOut instruction is out of range.	Ensure that the target deceleration is within the specified range.	Yes
9223	The target acceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9224	The target acceleration of the MC_Phasing instruction is out of range.	Ensure that the target acceleration is within the specified range.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9225	The target velocity of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9226	The target velocity of the MC_Phasing instruction is out of range.	Ensure that the target velocity is within the specified range.	Yes
9227	The curve type setting of the MC_CamOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9228	OutMode of the MC_CamOut instruction is out of range.	Ensure that OutMode is within the specified range.	Yes
9229	The MC_GenerateCamTable instruction detects that the cam node array is empty.	Contact Inovance for technical support.	No
9230	The node quantity specified by the MC_GenerateCamTable instruction exceeds the maximum allowable value.	Check whether the target node quantity specified in the instruction is beyond the specified range.	No
9231	Mode of the MC_GenerateCamTable instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9232	The node quantity specified by the MC_GenerateCamTable instruction is too small.	Ensure that the node quantity is 2 or more.	No
9233	RatioNumerator of the gear instruction is 0.	Set this parameter to a non-zero integer.	Yes
9234	RatioDenominator of the gear instruction is not greater than 0.	Set this parameter to an integer greater than 0.	Yes
9235	The MC_GenerateCamTable instruction is being executed when the MC_SaveCamTable instruction is called.	Do not call the MC_SaveCamTable instruction before the cam table data update operation is completed.	No
9236	The MC_SaveCamTable instruction is being executed on the cam table when the MC_GenerateCamTable instruction is called.	Do not call the MC_GenerateCamTable instruction before the cam table is saved.	No
9237	Failed to open the cam table file during execution of the MC_SaveCamTable instruction.	1. Check whether the PLC memory runs out. 2. Replace the PLC.	No
9238	Failed to write the cam point quantity when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.	No
9239	Failed to write data when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.	No
9240	The phase of the first point is not 0.	Ensure that the phase of the first point is 0.	Yes
9241	The displacement of the first point is not 0.	Ensure that the displacement of the first point is 0.	Yes
9242	Mode of the MC_GearOut instruction is out of range.	Ensure that Mode is within the specified range.	Yes
9243	The target deceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9244	The target deceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9245	Periodic of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9246	The phase in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.	Yes
9247	The absolute value of the displacement in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.	Yes
9248	The absolute value of the link velocity in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.	Yes
9249	The gear node is empty.	Contact Inovance for technical support.	Yes
9250	The master axis and slave axis are the same.	Do not use the same axis as both the master axis and slave axis of the cam gear.	Yes
9251	The configuration address of the master axis is greater than or equal to that of the slave axis.	When ReferenceType is set to set position of the current cycle, ensure that the configuration address of the master axis is less than that of the slave axis.	Yes
9252	The master axis filter coefficient fFilter [0] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).	Yes
9253	The master axis filter coefficient 2fFilter [1] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).	Yes
9254	The master axis filter coefficient 3fFilter [2] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).	Yes
9255	The sum of the master axis filter coefficients corresponding to the slave axis is not 1.	Ensure that the sum of the master axis filter coefficients corresponding to the slave axis is 1.	Yes
9256	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to positive, ensure that the cam synchronization point is not less than the cam engagement point.	Yes
9257	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to negative, ensure that the cam synchronization point is not greater than the cam engagement point.	Yes
9258	The target deceleration of the MC_GearOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9259	The target deceleration of the MC_Phasing instruction is out of range.	Ensure that the target deceleration is within the specified range.	Yes
9260	The target deceleration of the MC_GearIn instruction is out of range.	Ensure that the target deceleration is within the specified range.	Yes
9261	The target deceleration of the MC_GearOut instruction is out of range.	Ensure that the target deceleration is within the specified range.	Yes
9262	The target acceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.	Yes
9263	The target acceleration of the MC_GearIn instruction is out of range.	Ensure that the target acceleration is within the specified range.	Yes
9264	The curve type setting of the MC_Phasing instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9265	The curve type setting of the MC_GearIn instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9266	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.	Yes
9267	The slave axis is modified during the cam operation.	Do not modify the slave axis during the cam operation.	Yes
9268	Mode of the MC_Phasing instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9269	The current axis is not in cam control mode when the MC_CamOut instruction is called.	Ensure that the axis works in cam control mode when the MC_CamOut instruction is called.	No
9270	The current axis is not in gear control mode when the MC_GearOut instruction is called.	Ensure that the axis works in gear control mode when the MC_GearOut instruction is called.	No
9271	The position change of the master axis is too large within a single EtherCAT cycle during cam/gear operation.	Ensure that the position change of the master axis is not greater than half a cam cycle within a single EtherCAT cycle.	Yes
9272	The point specified by Phase in the MC_GetCamTableDistance instruction does not fall between the start and end points.	Ensure that the point specified by Phase is within the specified curve.	No
9273	The slave axis is changed during execution of the MC_GearIn instruction.	Do not change the slave axis during execution of the MC_GearIn instruction.	Yes
9274	Channel of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9275	The axis is not found.	Ensure that the axis specified by Axis exists.	No
9276	The number of tappets allowed to be executed at the same time is out of range.	Ensure that the number of tappets allowed to be executed at the same time is within the allowable range.	No
9277	ReferenceType of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9278	Number of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.	No
9279	The Switches array of the MC_DigitalCamSwitch instruction is empty.	Check whether the length of the Switches array meets requirements.	No
9280	fPosition of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9281	iMode of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9282	iDirection of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9283	fParameter of the tappet array is out of range.	Ensure that the parameter value is within the specified range.	No
9284	When the tappet comparison point is set to time mode, the time setting is out of range.	Ensure that the parameter value is within the specified range.	No
9285	The selected axis is not under cam control when ReferenceType of the MC_DigitalCamSwitch instruction is set to 3.	Call the MC_DigitalCamSwitch instruction after cam control takes effect.	No

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9286	Axis communication is interrupted during tappet execution.	Ensure that axis communication is not interrupted during tappet execution.	No
9287	The comparison position start points are the same during tappet execution.	Ensure that the start points are not duplicate.	No
9288	The comparison position start and end point are the same during tappet execution.	Ensure that the start and end points are not duplicate.	No
9289	The selected tappet terminal is being used by another function.	Check whether the terminal is set as the pulse output axis.	No
9290	The MC_DigitalCamSwitch instruction cannot be executed because the state of the motion control axis is improper.	Do not execute the MC_DigitalCamSwitch instruction in homing mode.	No
9291	The MasterSyncPosition setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9292	The SlaveSyncPosition setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9293	The MasterStarDistance setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9294	The Velocity setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.	Yes
9295	The Velocity setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.	Yes
9296	The Acceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.	Yes
9297	The Acceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.	Yes
9298	The Deceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.	Yes
9299	The Deceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.	Yes
9300	The AvoidReversal setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.	Yes
9301	The master axis speed is zero when the MC_GearInPos instruction is started.	Ensure that the master axis speed is not zero when starting this instruction.	Yes
9302	The master axis did not move during the catching phase of the MC_GearInPos instruction.	When MasterStarDistance is set to 0, ensure that the input MasterSyncPosition does not overlap with the current position of the master axis.	Yes
9303	When the MC_GearInPos instruction is started, the speed of the slave axis is not zero before entering the catching phase.	Ensure that the slave axis remains stationary before entering the catching phase.	Yes

Fault Code	Fault Information	Troubleshooting	Stop Triggered
9304	Failed to enter the catching phase when the MC_GearInPos instruction is executed.	Ensure that the master axis can enter the catching phase under the current position and motion direction conditions.	Yes
9305	The velocity of the slave axis exceeds the limit during execution of the MC_ GearInPos instruction.	Ensure that the parameter value is within the specified range.	Yes

3.10.3 Axis Group Control Instructions

3.10.3.1 Instruction List

The following table lists the axis group control instructions.

Instruction Category	Instruction	Function
Axis group control instruction	MC_MoveLinear	Linear interpolation
	MC_MoveCircular	Circular interpolation
	MC_MoveEllipse	Elliptical interpolation
	MC_GroupStop	Stop axis group operation
	MC_GroupPause	Pause axis group operation

3.10.3.2 MC_MoveLinear

MC_MoveLinear – Linear interpolation

Graphic Block

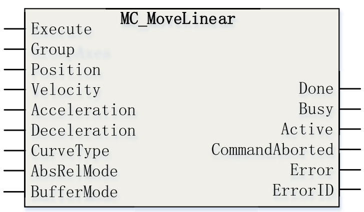
Instruction	Name	LD Expression	LiteST Expression
MC_MoveLinear	Linear interpola-tion		MC_MoveLinear(Execute := ???, Group := ???, Position := ???, Velocity := ???, Acceleration := ???, Deceleration := , CurveType := , AbsRelMode := , BufferMode := , Done => , Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

Table 3–241 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveLinear: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis ID	No	-	-	Axis group, INT
S2	Position	Target position	No	-	Positive number Negative number 0	REAL[0–3] or sMCGROUP_ INFO
S3	Velocity	Target velocity	No	-	Positive number	REAL
S4	Acceleration	Acceleration	No	-	Positive number	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	Positive number	REAL
S6	CurveType	Velocity curve type 0: T-shaped velocity curve Others: T-shaped velocity curve	Yes	0	0	INT
S7	AbsRelMode	Absolute or relative positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
S8	BufferMode	Buffer mode 0: Aborting+No transition 1: Buffered+No transition 2: Previous velocity+No transition 3: Superimpose corners	Yes	0	0 to 3	INT
D1	Done	Target position reached ON after the target position is reached	Yes	OFF	ON OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON OFF	BOOL
D3	Active	Controlling ON when starting to execute the current curve segment	Yes	OFF	ON OFF	BOOL
D4	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error flag	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Fault Code	Yes	0	*1	INT

Note

*1: See *"3.10.3.7 Fault Codes" on page 487*.

Function and Instruction Description

The MC_MoveLinear instruction performs linear interpolation on axis groups. It is active on the rising edge.

- Specifying axis
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

  Modification on Group is valid when Execute is OFF.

- Relationship with single-axis control instructions
  This instruction can be triggered only after all axes in the axis group are switched to the StandStill state by executing the MC_Power instruction.

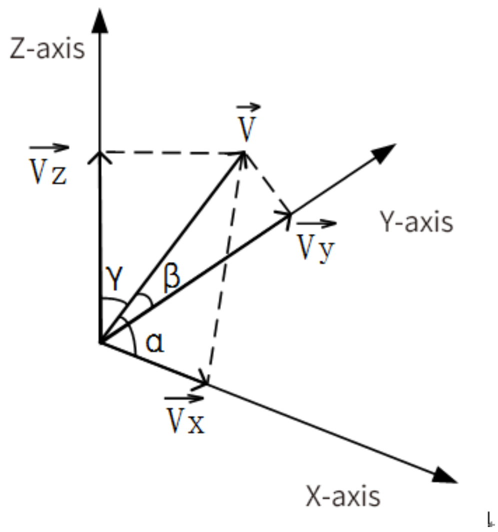
  This instruction is invalid if it is triggered during single-axis operations (such as jogging, torque control, homing, and stop).

  After this instruction is triggered, the single-axis PLCOpen state machine is in SynchronizedMotion state. During operation, this instruction cannot be aborted by single-axis motion instructions. After the interpolation curve is completed, the single-axis PLCOpen state machine enters the StandStill state, and single-axis motion instructions can be executed at this time.

  Velocity specifies the target velocity of the interpolator. The velocity of each coordinate axis is resolved according to formulas (1), (2), and (3).

- Description
  Position specifies the target position or displacement. Position[0] indicates the position displacement component of the x-axis, Position[1] indicates the position displacement component of the y-axis, Position[2] indicates the position displacement component of the z-axis, and Position [3] indicates the position displacement component of the auxiliary axis.

  Velocity specifies the target velocity of the interpolator. The velocity of each coordinate axis is resolved according to formulas (1), (2), and (3).



$$\mathrm{Vx} = \mathrm{V} \times \cos\alpha \qquad (1)$$

$$\mathrm{Vy} = \mathrm{V} \times \cos\beta \qquad (2)$$

$$\mathrm{Vz} = \mathrm{V} \times \cos\gamma \qquad (3)$$

$$\mathrm{V} = \sqrt[2]{Vx^2 + Vy^2 + Vz^2} \qquad (4)$$

The interpolation velocity of the auxiliary axis differs in the following two cases:

1. When the point on the coordinate axes does not move and the auxiliary axis moves independently, the auxiliary axis moves according to the target velocity specified by Velocity.
2. When the point on the coordinate axes moves, the auxiliary axis will reach the target position at the same time as the point on the coordinate axes. Assume that the linear interpolation length is L1, the target displacement of the auxiliary axis is L2, and the linear interpolation rate at a certain moment is V0. The velocity Va of the auxiliary axis is calculated as follows:

$$\mathrm{Va} = \mathrm{V0} \times \frac{L2}{L1} \qquad (5)$$

- Relative/Absolute mode election

  When AbsRelMode is set to 0, the absolute positioning mode is used. After this instruction is triggered, the three coordinate axes will finally move to the position specified by (Position[0], Position[1], Position[2]), and the auxiliary axis will move to the position specified by Position[3].

  When AbsRelMode is set to 1, the relative positioning mode is used. Set the position of the three coordinate axes of the axis group to (Px, Py, Pz) and the current position of the auxiliary axis to Pa. After this instruction is triggered, the three coordinate axes will finally move to (Px + Position[0], Py + Position[1], Pz + Position[2]), and the final position of the auxiliary axis is (Pa + Position[3]).

- Buffer and transition

  There are four optional buffer and transition modes. For details, see the "Interpolation Function" section in the "AutoShop Software Programming and Application Manual".

No.	Buffer Mode	Description
0	Aborting+No transition	Immediately switch to the next function block. There is no transition curve.
1	Buffered+No transition	Execute the buffered function block after the first segment of deceleration is completed. There is no transition curve.
2	Previous velocity+No transition	Move to the end of the first segment at the current velocity and start the second segment at the rate of the first segment.
3	Superimpose corners	Add acceleration of the second segment when deceleration starts in the first segment. There is a transition curve.

When buffer mode 1, 2, or 3 is selected, the interpolation instruction allows up to 8 curves to be buffered. When this instruction enters the buffer state, the Busy signal is active; when it is executed, the Active output becomes active; when the execution is completed, the Done signal output becomes active.

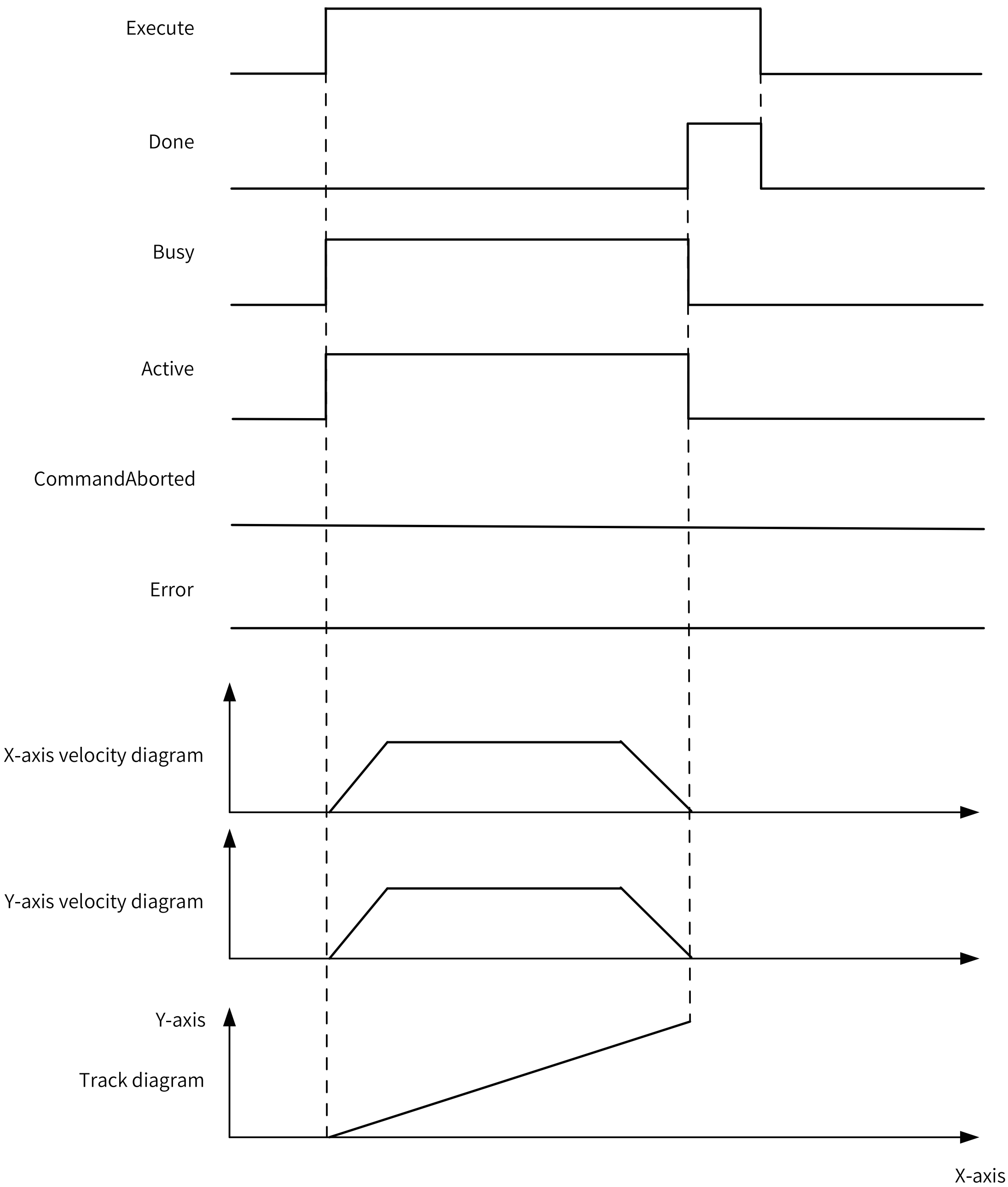
When mode 0 (Aborting+No transition) is selected for a newly added interpolation instruction, it will abort all interpolation instructions that are being executed or buffered. The CommandAborted output of the aborted interpolation instructions becomes active.

- Re-execution
  This instruction cannot be re-executed.

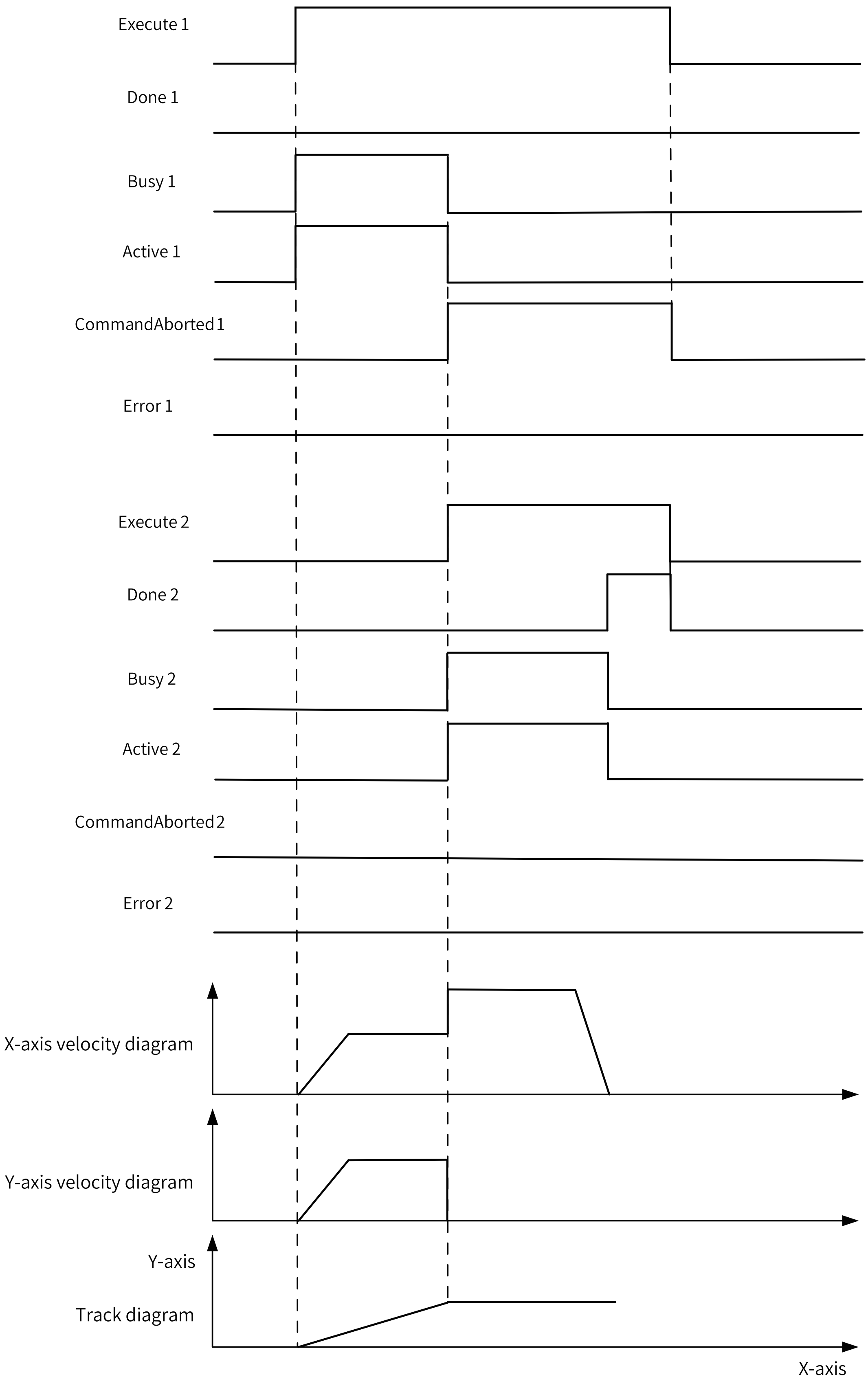
  If this instruction is triggered repeatedly when the Busy output is ON, the axis will report fault 9421 (instruction re-execution error), and all axes will stop operation immediately and enter the ErrorStop state.

Timing Diagram

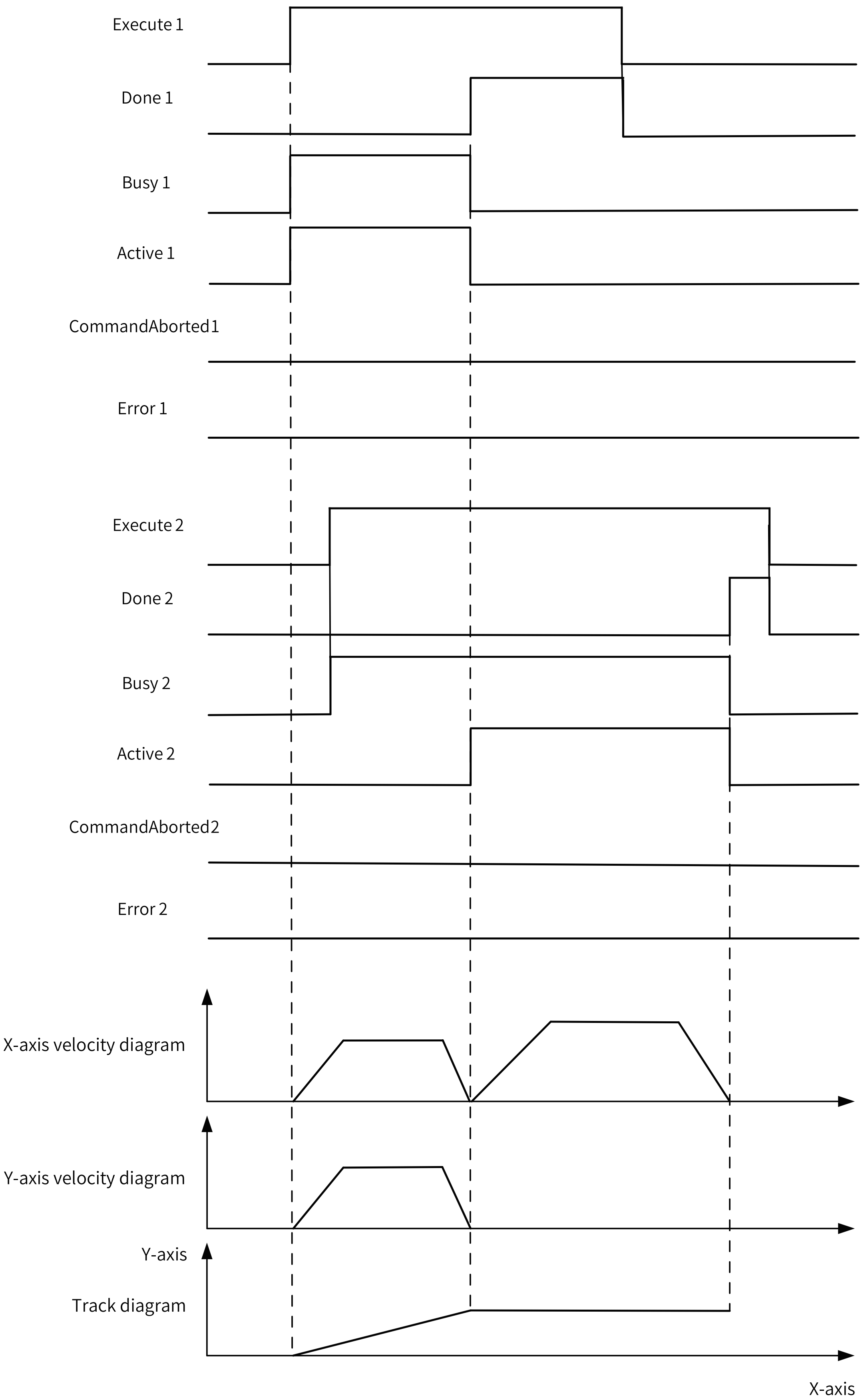
- A linear interpolation instruction is called to perform interpolation along the x-axis and y-axis.



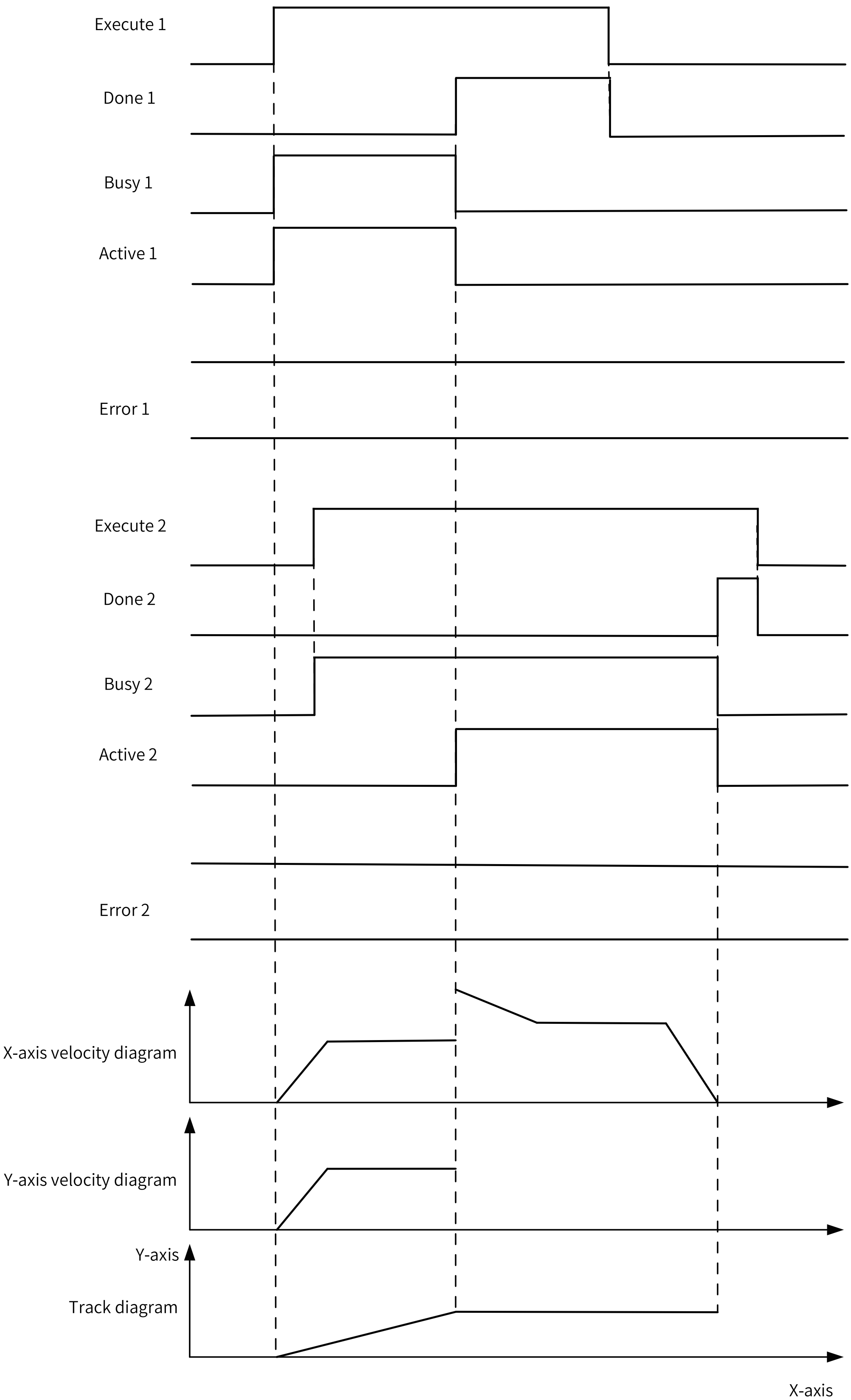
- Two linear interpolation instructions are called, of which the second one is triggered during execution of the first one to abort the first one.



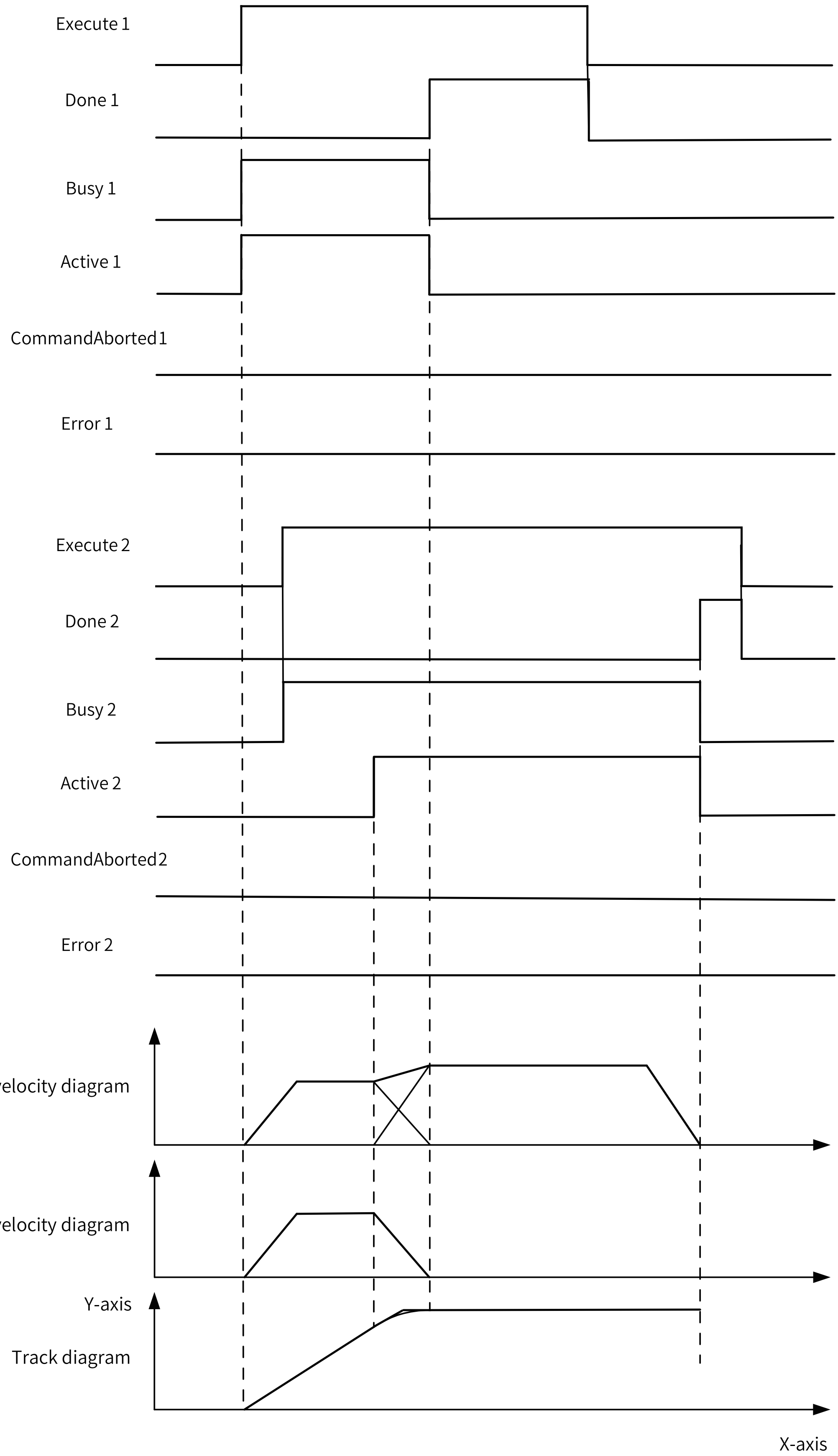
- Two linear interpolation instructions are called, and the second instruction is executed in "buffer +no transition" mode.



- Two linear interpolation instructions are called, and the second instruction is executed in "previous velocity+no transition" mode.


Execute 1
Done 1
Busy 1
Active 1
Error 1
Execute 2
Done 2
Busy 2
Active 2
Error 2
X-axis velocity diagram
Y-axis velocity diagram
Y-axis
Track diagram
X-axis

- Two linear interpolation instructions are called, and the second instruction is executed in "superimpose corners" mode.



3.10.3.3 MC_MoveCircular

MC_MoveCircular – Circular interpolation

Graphic Block

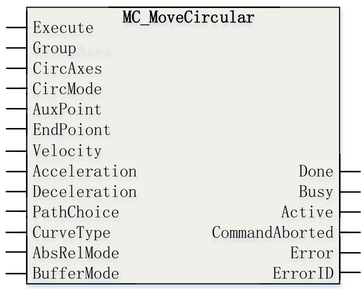
Instruction	Name	LD Expression	LiteST Expression
MC_MoveCircular	Circular interpola-tion		MC_MoveCircular(Execute := ???, Group := ???, CircAxes := , CircMode := , AuxPoint := ???, EndPoint := ???, Velocity := ???, Acceleration := ???, Deceleration := , PathChoice := , CurveType := , AbsRelMode := , BufferMode := , Done => , Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

Table 3–242 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveCircular: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis ID	No	-	-	Axis group, INT
S2	CircAxes	Circular axis 0: x-y axis plane 1: y-z axis plane 2: x-z axis plane	Yes	0	0 to 2	INT

S3	CircMode	Circular interpolation mode 0: Border point 1: Center 2: Radius	Yes	0	0 to 2	INT
S4	AuxPoint	Auxiliary point	No	-	Positive number Negative number 0	REAL[0–3]
S5	EndPoint	End point	No	-	Positive number Negative number 0	REAL[0–3] or _sMC_GROUP_POS
S6	Velocity	Target velocity	No	-	Positive number	REAL
S7	Acceleration	Acceleration	No	-	Positive number	REAL
S8	Deceleration	Deceleration	Yes	Acceleration	Positive number	REAL
S9	PathChoice	Path choice 0: CW 1: CCW	Yes	0	0 to 1	INT
S10	CurveType	Velocity curve type 0: T-shaped velocity curve Others: T-shaped velocity curve	Yes	0	0	INT
S11	AbsRelMode	Absolute or relative positioning mode 0: Absolute positioning 1: Relative positioning	Yes	0	0 to 1	INT
S12	BufferMode	Buffer mode 0: Aborting+No transition 1: Buffered+No transition 2: Previous velocity+No transition 3: Superimpose corners	Yes	0	0 to 3	INT
D1	Done	Target position reached ON after the target position is reached	Yes	OFF	ON OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON OFF	BOOL
D3	Active	Controlling ON when starting to execute the current curve segment	Yes	OFF	ON OFF	BOOL

D4	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error flag	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Fault Code	Yes	0	*1	INT

Note

*1: See *"3.10.3.7 Fault Codes" on page 487*.

Function and Instruction Description

The MC_MoveCircular instruction performs circular interpolation on axis groups. It is active on the rising edge.

- Specifying axis
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

  Modification on Group is valid when Execute is OFF.

- Relationship with single-axis control instructions
  This instruction can be triggered only after the axes are switched to the StandStill state by executing the MC_Power instruction.

  This instruction is invalid if it is triggered during single-axis operations (such as jogging, torque control, homing, and stop).

  After this instruction is triggered, the single-axis PLCOpen state machine is in SynchronizedMotion state. During operation, this instruction cannot be aborted by single-axis motion instructions. After the interpolation curve is completed, the single-axis PLCOpen state machine enters the StandStill state, and single-axis motion instructions can be executed at this time.

- Specifying circular axes
  CircAxes specifies the coordinate plane. To be specific:

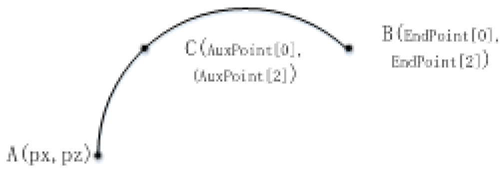
  When CircAxes is set to 0, the x-y coordinate plane is selected. The motion axes specified by AxisID_x and AxisID_y perform circular interpolation, and the axes specified by AxisID_z and AxisID_a are auxiliary axes and perform linear interpolation.

  When CircAxes is set to 1, the y-z coordinate plane is selected. The motion axes specified by AxisID_y and AxisID_z perform circular interpolation, and the axes specified by AxisID_x and AxisID_a are auxiliary axes and perform linear interpolation.

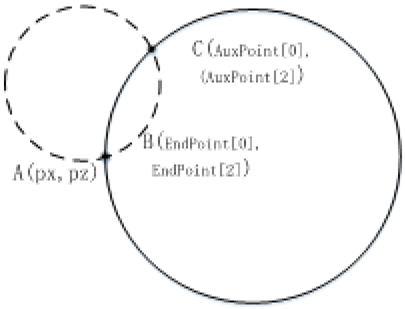
  When CircAxes is set to 2, the x-z coordinate plane is selected. The motion axes specified by AxisID_x and AxisID_z perform circular interpolation, and the axes specified by AxisID_y and AxisID_a are auxiliary axes and perform linear interpolation.

- Selecting interpolation mode

  1. When CircMode is set to 0, circular interpolation is performed based on the border point.



When the starting point and end point are different



When the starting point and end point are the same
The solid line is the path when PathChoice is equal to 0.
The dashed line is the path when PathChoice is equal to 1.

When the x-y plane is selected, the border point is (AuxPoint[0], AuxPoint[1]), and the end point is (EndPoint[0], EndPoint[1]); when the y-z plane is selected, the border point is (AuxPoint[1], AuxPoint[2]), and the end point is (EndPoint[1], EndPoint[2]); when the x-z plane is selected, the border point is (AuxPoint[0], AuxPoint[2]), and the end point is (EndPoint[0], EndPoint[2]).

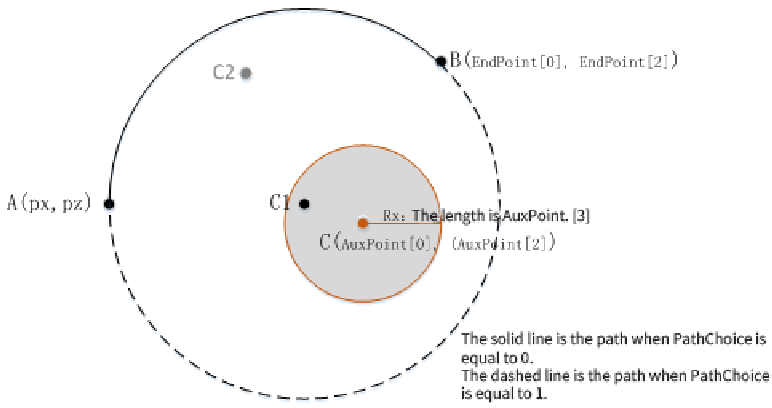
Take the x-y plane as an example. The start position of the x-axis is Px, and that of the y-axis is Py. The instruction performs circular interpolation from the start point (Px, Py) through the border point (AuxPoint[0], (AuxPoint[1]) to the end point (EndPoint[0], EndPoint[1]) after it is triggered.

If the start point and the end point are the same point, a complete circle is drawn with the straight line between the start point and the border point as the diameter. In this case, PathChoice specifies the circular interpolation direction.

If the start point, border point, and end point are along the same line, they cannot form a circle. In this case, an error occurs and execution of the interpolation instruction is aborted.

If the start point and the border point are the same point, or the end point and the border point are the same point, an error occurs and execution of the interpolation instruction is aborted.

2. When CircMode is set to 1, circular interpolation is performed based on the center.



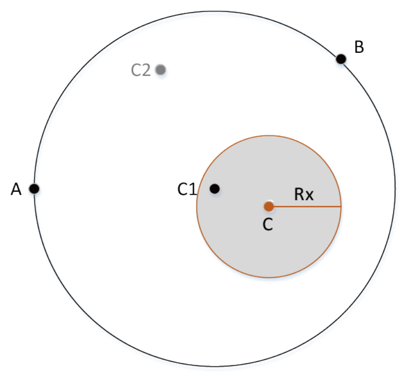
When the x-y plane is selected, the center is (AuxPoint[0], AuxPoint[1]), and the end point is (EndPoint[0], EndPoint[1]); when the y-z plane is selected, the center is (AuxPoint[1], AuxPoint

[2]), and the end point is (EndPoint[1], EndPoint[2]); when the x-z plane is selected, the center is (AuxPoint[0], AuxPoint[2]), and the end point is (EndPoint[0], EndPoint[2]).

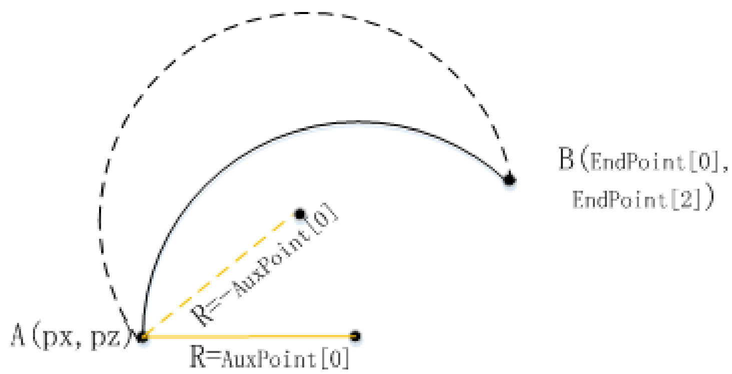
Take the x-z plane as an example. The start position of the x-axis is Px, and that of the z-axis is Pz. The instruction performs circular interpolation from the start point (Px, Pz) to the end point (EndPoint[0], EndPoint[2]) for the circle specified by the center point (AuxPoint[0], (AuxPoint[2]) after it is triggered. PathChoice specifies the circular interpolation direction.

If the distance R1 from the specified center (AuxPoint[0], AuxPoint[2]) to the start point (Px, Pz) differs from the distance R2 to the end point (EndPoint[0], EndPoint[2]) (the difference between R1 and R2 is greater than 1), the average value R of R1 and R2 is calculated as the radius, and the center (Cx, Cy) is calculated in the same way as specifying the radius. Circular interpolation is performed using the calculated radius and center.

Note that when adjusting the center, if two centers are calculated, first calculate the distance between each of the two calculated centers and the set center, and select the one closer to the set center. This point must be located inside the circle with (AuxPoint[0], AuxPoint[2]) as the center and AuxPoint[3] as the radius. As shown in the following figure, C1 is selected as the new center.



3. When CircMode is set to 2, circular interpolation is performed based on the specified radius.



The solid line is the path when R is a positive value.
The dashed line is the path when R is a negative value.

No matter which plane is selected, the radius is always determined by |AuxPoint[0]|. When the x-y plane is selected, the end point is (EndPoint[0], EndPoint[1]); when the y-z plane is selected, the end point is (EndPoint[1], EndPoint[2]); when the x-z plane is selected, the end point is (EndPoint[0], EndPoint[2]).

Take the y-z plane as an example. The start position of the y-axis is Py, and that of the z-axis is Pz. The instruction performs circular interpolation from the start point (Py, Pz) to the end point (EndPoint[1], EndPoint[2]) for the circle specified by the radius |AuxPoint[0]|.

If the sign of the radius is negative, a circle with a long arc will be drawn. If the sign is positive, a circle with a short arc will be drawn. PathChoice specifies the circular interpolation direction.

- Selecting positioning mode

1. Absolute mode

When the border point is selected, the auxiliary point and end point are absolute points in the coordinate system.

When the center is selected, the center point and end point are absolute points in the coordinate system.

When the radius is selected, the end point is the absolute point in the coordinate system.

2. Relative mode

When the border point is selected, the auxiliary point and end point are relative points relative to the start point.

When the center is selected, the center point and end point are relative points relative to the start point.

When the radius is selected, the end point is the relative point relative to the start point.

- Buffer and transition
There are four optional buffer and transition modes. For details, see the "Interpolation Function" section in the "AutoShop Software Programming and Application Manual".

When buffer mode 1, 2, or 3 is selected, the interpolation instruction allows up to 8 curves to be buffered. When this instruction enters the buffer state, the Busy signal is active; when it is executed, the Active output becomes active; when the execution is completed, the Done signal output becomes active.

When mode 0 (Aborting+No transition) is selected for a newly added interpolation instruction, it will abort all interpolation instructions that are being executed or buffered. The CommandAborted output of the aborted interpolation instructions becomes active.

- Re-execution
This instruction cannot be re-executed.

If this instruction is triggered repeatedly when the Busy output is ON, the axis will report fault 9421 (instruction re-execution error), and all axes will stop operation immediately and enter the ErrorStop state.

Timing Diagram

See the timing diagram of the linear interpolation instruction.

3.10.3.4 MC_MoveEllipse

MC_MoveEllipse – Elliptical interpolation

Graphic Block

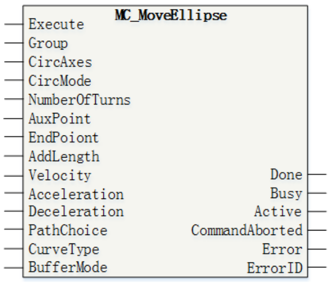
Instruction	Name	LD Expression	LiteST Expression
MC_MoveEllipse	Elliptical interpola-tion		MC_MoveEllipse(Execute := ???, Group := ???, CircAxes := , CircMode := , NumOfTurns := , AuxPoint := ???, EndPoint := ???, AddLength := , Velocity := ???, Acceleration := ???, Deceleration := , PathChoice := , CurveType := , BufferMode := , Done => , Busy => , Active => , CommandAborted => , Error => , ErrorID =>);

Table 3–243 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveEllipse: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis group	No	-	0 to 32767	INT
S2	CircAxes	Ellipse axis 0: x-y axis plane 1: y-z axis plane 2: x-z axis plane	Yes	0	0 to 2	INT

S3	CircMode	Elliptical interpolation mode 0: Complete ellipse 1: Specified arc length	Yes	0	0 to 2	INT
S4	NumOfTurns	Number of turns	Yes	1	1 to 100	INT
S5	AuxPoint	Center point	No	-	Positive number Negative number 0	REAL[0–3]
S6	EndPoint	End point coordinates	No	-	Positive number Negative number 0	REAL[0–3] or _sGROUP-POS_INFO
S7	AddLength	Arc length during running when CircMode is 1	No	-	Negative number 0 Positive number	REAL
S8	Velocity	Target velocity	No	-	Positive number	REAL
S9	Acceleration	Acceleration	No	-	Positive number	REAL
S10	Deceleration	Deceleration	Yes	Acceleration	Positive number	REAL
S11	PathChoice	Path choice 0: CW 1: CCW	Yes	0	0 to 1	INT
S12	CurveType	Velocity curve type 0: T-shaped velocity curve Others: T-shaped velocity curve	Yes	0	0	INT
S13	BufferMode	Buffer mode 0: Aborting+No transition 1: Buffered+No transition 2: Previous velocity+No transition 3: Superimpose corners	Yes	0	0 to 3	INT
D1	Done	Target position reached TRUE after the target position is reached	Yes	FALSE	TRUE FALSE	BOOL
D2	Busy	Busy flag	Yes	FALSE	TRUE FALSE	BOOL

D3	Active	Controlling TRUE when starting to execute the current curve segment	Yes	FALSE	TRUE FALSE	BOOL
D4	CommandA-borted	Abortion of execution	Yes	FALSE	TRUE FALSE	BOOL
D5	Error	Error flag	Yes	FALSE	TRUE FALSE	BOOL
D6	ErrorID	Fault Code	Yes	0	*1	INT

Function and Instruction Description

Determination of Elliptic Plane

By selecting the coordinate plane through CircAxes, you can draw an ellipse on the x-y plane, x-z plane, or y-z plane. After selecting the plane, the other two axes are used as auxiliary axes and linear interpolation is used.

Starting Point and Center Point of Ellipse

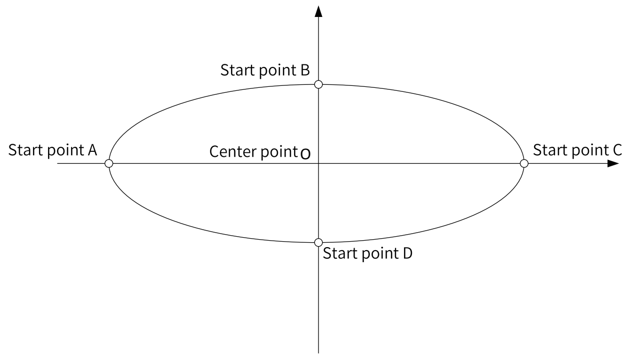
The starting point of the ellipse is the set position of the coordinate axis when the instruction is called, and the center point is (AuxPoint [0], AuxPoint [1]) specified by the instruction.

If you select the x-y plane, AuxPoint [0] is the X-axis coordinate and AuxPoint [1] is the Y-axis coordinate.

If you select the x-z plane, AuxPoint [0] is the X-axis coordinate and AuxPoint [1] is the Z-axis coordinate.

If you select the Y-z plane, AuxPoint [0] is the Y-axis coordinate and AuxPoint [1] is the Z-axis coordinate.

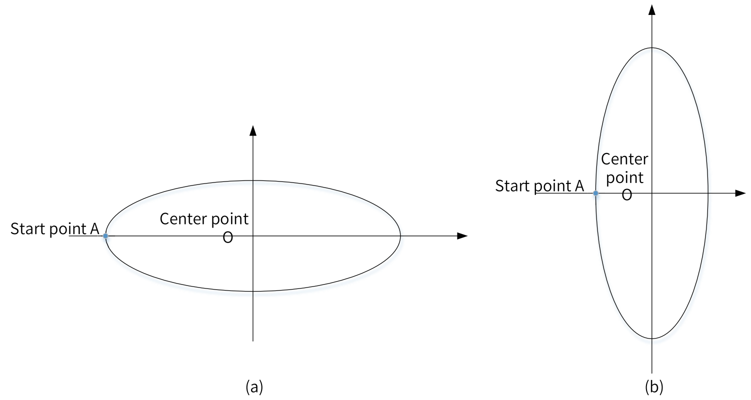
The line connecting the starting point and center point must be parallel to the coordinate axis. Taking the xy plane as an example, set the starting point coordinates to (x0, y0) and the center point coordinates to (x1, y1). When x0 = x1, the line connecting the two points is parallel to the Y-axis. If y0>y1, the starting point is point C in the figure below. If y0<y1, the starting point is point A in the figure below.



Determination of the Major Axis and Minor Axis

The parameters AuxPoint [2] and AuxPoint [3] in the instruction specify the lengths of the major and minor axes of the ellipse. When AuxPoint [2] > AuxPoint [3], AuxPoint [2] is the major axis, otherwise AuxPoint [3] is the major axis.

Assume that AuxPoint [2] = 12 and AuxPoint [3] = 5. If the distance of |AO| is equal to 12, the ellipse on the left is finally selected. If the distance of |AO| is equal to 5, the ellipse on the right is finally selected.



When the major axis equals the minor axis, the ellipse will become a circle.

Running Direction of the Ellipse

PathChoice specifies the running direction of the ellipse. When PathChoice is set to 0, it indicates clockwise running; When PathChoice is set to 1, it indicates counterclockwise running.

Determination of the End Point

CircMode determines the position of the end point.

When CircMode is 0, the running trajectory of the axis is a complete ellipse, where the end point is equal to the starting point.

When CircMode is 1, it is necessary to specify the arc length that is run from the starting point. When NumOfTurns is 0, PathChoice specifies the direction as CW (CCW). If AddLength is positive, the arc length specified by AddLength is run clockwise (counterclockwise). If AddLength is negative, the arc length specified by AddLength is run counterclockwise (clockwise). If NumOfTurns is greater than 0, run for (NumOfTurns – 1) turns in the direction specified by PathChoice, and then run at the arc length specified by AddLength on the last turn.

Processing of the Auxiliary Axis

When you select the x-y plane, the z-axis and a-axis are auxiliary axes. When you select the y-z plane, the x-axis and a-axis are auxiliary axes. When you select the x-z plane, the y-axis and a-axis are auxiliary axes.

In elliptical interpolation, the auxiliary axis will start running together with the elliptic coordinate axis and end the action together. When the elliptic coordinate axis is running, the auxiliary axis performs linear interpolation from the starting point to the end point, and the coordinates of the auxiliary axis at the end point are specified by EndPoint. When you select the x-y plane, the end point coordinate of the Z-axis is EndPoint[2], and the end point coordinate of the A-axis is EndPoint[3].

Fault Code	Cause	Solution	Stop
9466	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9467	AddLength in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes

Fault Code	Cause	Solution	Stop
9468	The axis stops due to an error of the MC_MoveEllipse instruction.	Check the fault code of the MC_MoveEllipse instruction with an exception to locate the fault.	Yes
9469	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9470	CircMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9471	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9472	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9473	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9474	Deceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9475	BufferMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.	Yes
9476	Failed to draw an ellipse because the center and the lengths of the major and minor axes are set incorrectly.	Ensure that the configured values of these parameters can form an ellipse.	Yes

3.10.3.5 MC_GroupStop

MC_GroupStop – Stop axis group operation

Graphic Block

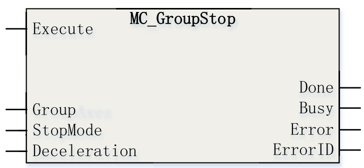
Instruction	Name	LD Expression	LiteST Expression
MC_GroupStop	Stop axis group operation		MC_GroupStop(Execute := ???, Group := ???, StopMode := , Deceleration := , Done => , Busy => , Error => , ErrorID =>);

Table 3–244 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveStop: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis ID	No	-	0 to 32767	INT

S2	StopMode	Stop Modes 0: Decelerate to stop 1: Stop immediately	Yes	1	0 to 1	INT
S3	Deceleration	Deceleration	Yes	1000	Positive number, less than the maximum acceleration	REAL
D1	Done	Stop completed	Yes	OFF	OFF ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF ON	BOOL
D3	Error	Error flag	Yes	OFF	OFF ON	BOOL
D4	ErrorID	Fault Code	Yes	0	*1	INT

Note

*1: For details, see the *“3.10.3.7 Fault Codes” on page 487Fault Codes* section.

Function and Instruction Description

The MC_GroupStop instruction stops all axes in an axis group. It is active on the rising edge.

- Specifying axis
  Group is latched on the rising edge of the Execute input.

  Modification on Group is invalid when Execute is ON.

  Modification on Group is valid when Execute is OFF.

- Effective range
  The MC_GroupStop instruction can only stop interpolation instructions (such as MC_MoveLinear) but not single-axis motion instructions (such as MC_MoveAbsolute).

  The MC_Stop instruction can only stop single-axis motion instructions but not interpolation instructions.

- State transition
  On the rising edge of Execute, the interpolator performs the stop operation according to the stop mode specified by StopMode and aborts all buffered interpolation instructions. After the stop is completed, the Done signal output is active, and the single-axis PLCOpen state machine is still in the SynchronizedMotion state.

  During the period when Execute remains ON, the interpolator is always in the stop state, and a new interpolation instruction is invalid if it is triggered at this time.

  On the falling edge of Execute, the interpolator switches to a non-stop state, and each axis enters the StandStill state. At this time, a new interpolation instruction can be triggered.

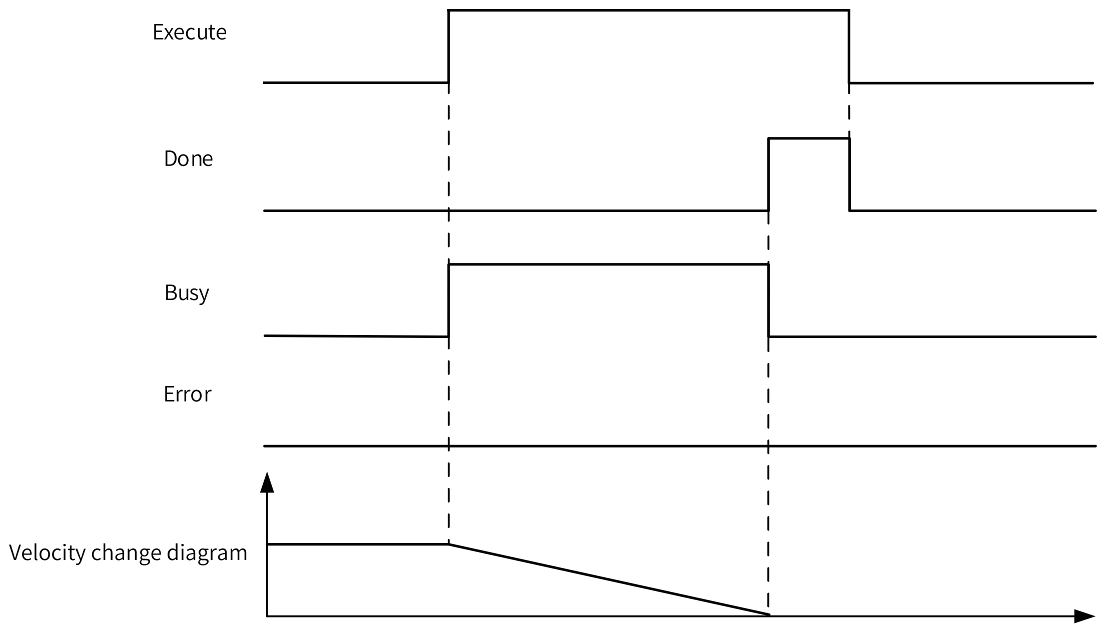
- Stop
  When StopMode is set to 0, the axes decelerate and stop according to the deceleration specified by Deceleration.

When StopMode is set to 1, the axes stop immediately without deceleration.

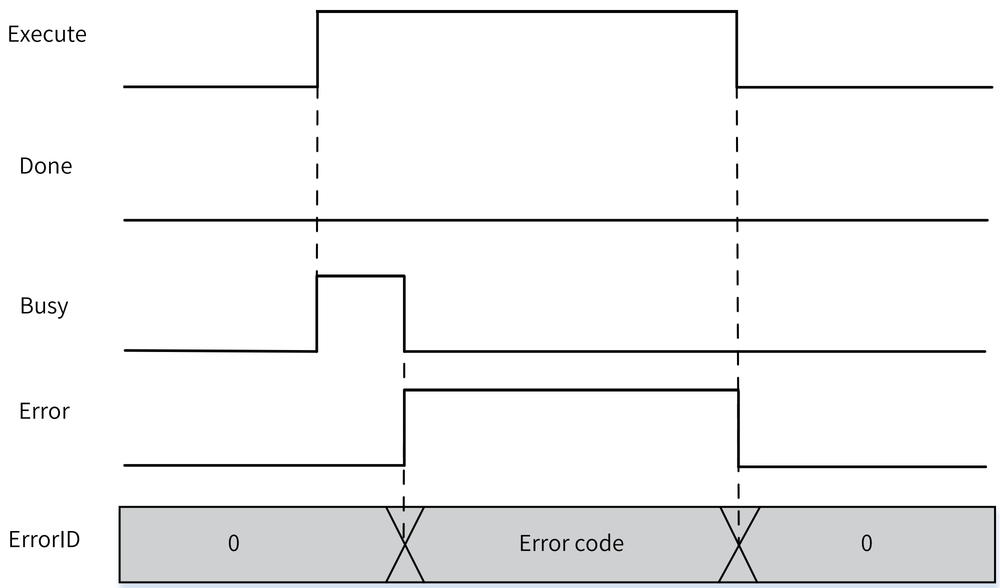
- Re-execution
  When this instruction is re-triggered during axis deceleration, the axes in the axis group will decelerate according to the new deceleration.
- Multi-execution
  This instruction does not support multi-execution. When an MC_GroupStop instruction is being executed and the Execute input is ON, if another MC_GroupStop instruction is triggered, the newly triggered instruction reports error 9441 (MC_GroupStop multi-execution error).

Timing Diagram

- The axes decelerate to stop and can stop normally.



- An axis fails during deceleration.



3.10.3.6 MC_GroupPause

MC_GroupPause – Pause axis group operation

Graphic Block

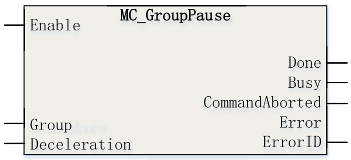
Instruction	Name	LD Expression	LiteST Expression
MC_GroupPause	Pause axis group operation		MC_GroupPause(Enable := ???, Group := ???, Deceleration := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–245 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveCircular: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Group	Axis group	No	-	0 to 32767	INT
S2	Deceleration	Deceleration	Yes	1000	Positive number, less than the maximum acceleration	REAL
D1	Done	Pause completed	Yes	OFF	OFF ON	BOOL
D2	Busy	Busy flag	Yes	OFF	OFF ON	BOOL
D3	CommandA-borted	Abortion flag	Yes	OFF	OFF ON	BOOL
D3	Error	Error flag	Yes	OFF	OFF ON	BOOL
D4	ErrorID	Fault Code	Yes	0	*1	INT

Note

*1: See *"3.10.3.7 Fault Codes" on page 487*.

Function and Instruction Description

The MC_GroupPause pauses all axes in an axis groups. It is active on the rising edge.

- Specifying axis

Group is latched on the rising edge of the Execute input.

Modification on Group is invalid when Execute is ON.

Modification on Group is valid when Execute is OFF.

- Effective range
  The MC_GroupPause instruction can only pause interpolation instructions (such as MC_MoveLinear) but not single-axis motion instructions (such as MC_MoveAbsolute).
- State transition
  All axes in an axis group are in StandStill state:

  When Enable is set to ON, the axes in the axis group are still in the StandStill state. If a linear or circular interpolation instruction is triggered at this time, all axes in the axis group will switch to the SynchronizedMotion state but remain paused and do not perform the interpolation algorithm until the Enable signal of the MC_GroupPause instruction becomes OFF.

  All axes in an axis group are in SynchronizedMotion state:

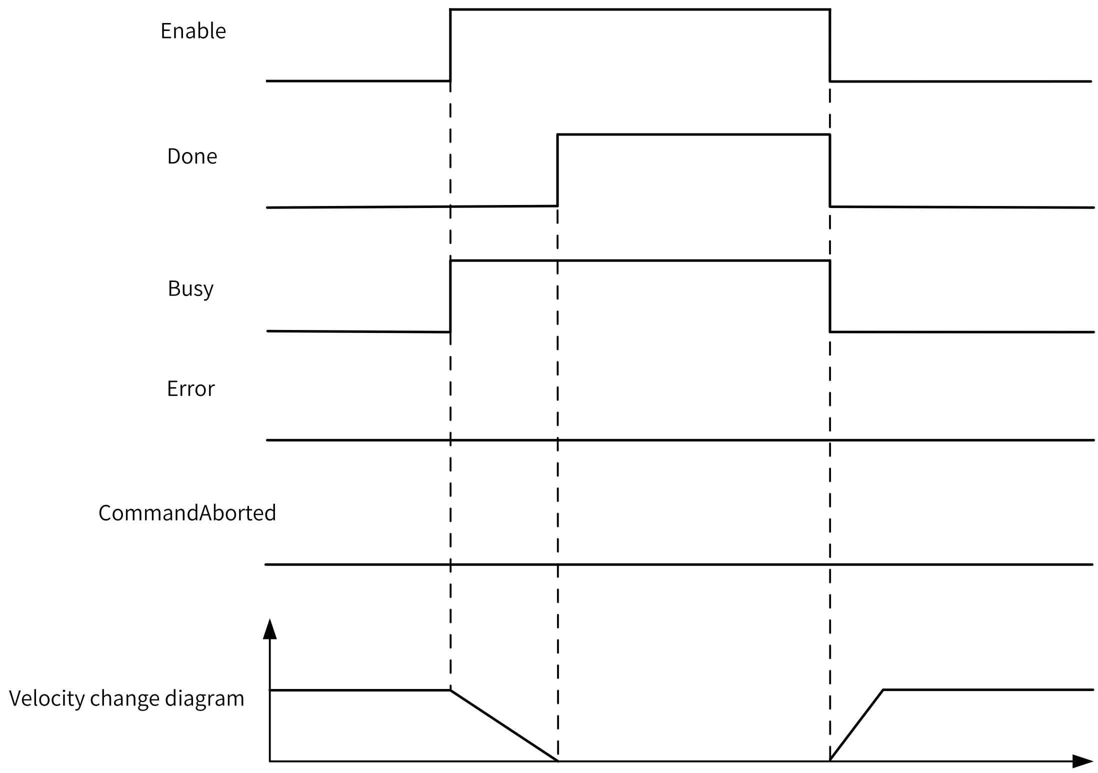
  On the rising edge of Enable, the interpolator performs deceleration according to the deceleration specified by Deceleration. After deceleration is completed, the Done signal output becomes active, the single-axis PLCOpen state machine is still in the SynchronizedMotion state, and the Busy signal and Valid signal of the interpolation instruction being executed remain active during the pause period.

  During the period when the Enable signal is ON, the interpolator remains in paused state, and a new interpolation instruction is buffered if it is triggered at this time.

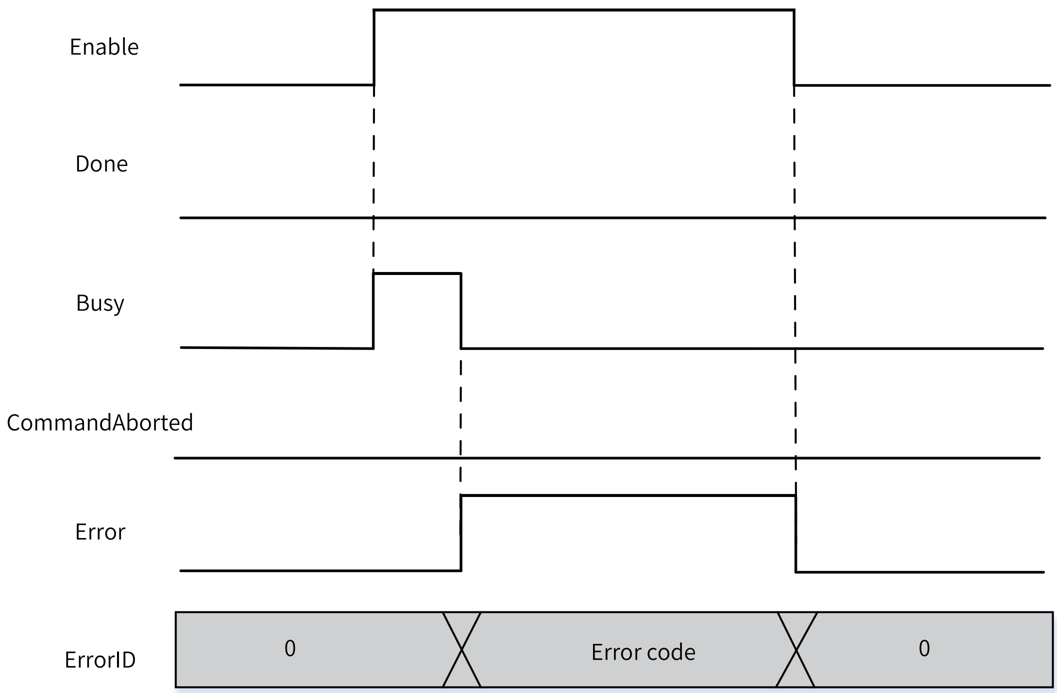
  On the falling edge of Enable, the interpolator resumes execution of the previously paused interpolation instruction.
- Re-execution
  When this instruction is re-triggered during axis deceleration, the axes in the axis group will decelerate according to the new deceleration.
- Multi-execution
  If an MC_GroupPause instruction is triggered during execution of another MC_GroupPause instruction, the first executed pause instruction is aborted, and the interpolator starts to decelerate according to the deceleration of the later triggered instruction.

Timing Diagram

- The axes decelerate to stop and can stop normally.



- An axis fails during deceleration.



3.10.3.7 Fault Codes

When a fault occurs during use of the interpolation functions, refer to the fault codes listed in the following table for troubleshooting.

Fault Code	Fault Information	Troubleshooting
9400	The number of axis groups exceeds the maximum allowable value.	Check whether the number of axis groups is greater than 8.
9401	An axis in the axis group is faulty.	Check whether an axis in the axis group has entered the ErrorStop state. Locate the fault based on the fault code of each axis.
9402	The number of buffered interpolation instructions is greater than 8.	Check whether the number of buffered interpolation instructions is greater than 8.
9403	The axis is reused.	Locate the reused axis and replace it with an unused axis.
9404	Failed to create the axis group.	The x-axis and y-axis cannot be empty. Check whether the x-axis or y-axis does not exist or is not specified.
9405	The specified z-axis does not exist.	Check whether the axis specified by AxisID_z exists.
9406	The specified auxiliary axis does not exist.	Check whether the axis specified by AxisID_a exists.
9407	The axis group ID is duplicate.	Check whether GroupID is duplicate.
9408	Failed to configure the axis.	Check whether any axis in the axis group fails to be configured. If yes, check whether the board software and the background match.
9409	The axis ID is less than 0.	Check whether the ID of an axis in the axis group is less than 0.
9410	The axis group is not released because the same MC_SetAxesGroup instruction is triggered repeatedly in a short time period.	Do not re-trigger the MC_SetAxesGroup instruction while its Busy signal output is still active.
9411	The MC_GroupStop instruction is aborted.	Check whether an instruction with higher priority is called while the MC_GroupStop instruction is still active.
9412	The value of CircAxes of the circular interpolation instruction is out of range.	Check whether the value of CircAxes of the circular interpolation instruction is out of range.
9413	The value of CircMode of the circular interpolation instruction is out of range.	Check whether the value of CircMode of the circular interpolation instruction is out of range.
9414	The value of PathChoice of the circular interpolation instruction is out of range.	Check whether the value of PathChoice of the circular interpolation instruction is out of range.
9415	The value of StopMode of the MC_GroupStop instruction is out of range.	Check whether the value of StopMode of the MC_GroupStop instruction is out of range.
9416	The x-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9417	The y-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9418	The z-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9419	The auxiliary axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9420	The circular interpolation instruction is triggered repeatedly.	Do not re-trigger the same circular interpolation instruction while its Busy signal output is still active.
9421	The linear interpolation instruction is triggered repeatedly.	Do not re-trigger the same linear interpolation instruction while its Busy signal output is still active.
9422	Failed to obtain the axis group.	Check whether the axis group specified by GroupID has been created by calling MC_SetAxesGroup.
9423	Failed to configure the axis.	Check whether an instruction is triggered when axis configuration is not completed. Check whether the communication state of all axes in the axis group is Axis ready.

Fault Code	Fault Information	Troubleshooting
9424	An axis is disabled.	Do not call the interpolation instruction when any axis is in Disabled state.
9425	An axis is executing single-axis motion instructions.	Do not call the interpolation instruction when any axis is executing single-axis motion instructions and not in StandStill state.
9426	An axis is in Stopping state.	Do not call the interpolation instruction when any axis is in Stopping state after executing the MC_Stop instruction.
9427	The axis group is in Stopping state.	Do not call the interpolation instruction while the MC_GroupStop instruction is still active.
9428	An axis is in Homing state.	Do not call the interpolation instruction when any axis is in Homing state after executing the MC_Home instruction.
9429	An axis is executing the position setting instruction.	Do not call the interpolation instruction when any axis is setting the current position by executing the MC_SetPosition instruction.
9430	An axis is in commissioning state.	Do not call the interpolation instruction when any axis is in commissioning state.
9431	An axis enters the commissioning state during interpolation, which aborts instruction execution of other axes.	Check whether any axis enters the commissioning state during interpolation.
9432	Failed to request the memory.	Check whether the memory runs out. Contact the manufacturer.
9433	The target velocity is 0 or less than 0.	Ensure that the target velocity of the instruction is greater than 0.
9434	The target acceleration is 0 or less than 0.	Ensure that the target acceleration of the instruction is greater than 0.
9435	The target deceleration is 0 or less than 0.	Ensure that the target deceleration of the instruction is greater than 0.
9436	The curve type setting is out of range.	Check whether the curve type is set to a value other than the T-shaped curve for the interpolation instruction.
9437	AbsRelMode is set incorrectly.	Check whether the parameter is set to a value other than the absolute positioning and relative positioning modes.
9438	BufferMode is set incorrectly.	Check whether the value of BufferMode is out of range.
9439	InsertMode is set incorrectly.	Check whether the value of InsertMode is proper.
9440	An axis stops due to a fault.	Locate the faulty axis and rectify the fault based on the fault code.
9441	The MC_GroupStop instruction is called repeatedly.	Do not re-trigger an MC_GroupStop instruction or call other MC_GroupStop instructions while an MC_GroupStop instruction is still active.
9442	The data buffer area is not empty.	Contact Inovance for technical support.
9443	No circle can be drawn.	-
9444	The start, end, and border points in the circular interpolation instruction are the same point, and no circle can be drawn.	Check the input parameters of the circular interpolation instruction and ensure that the start, end, and border points can form a circle.
9445	The instruction buffer area is full.	Contact Inovance for technical support.
9446	The velocity of the x-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the x-axis is not greater than the maximum allowable velocity.
9447	The velocity of the y-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the y-axis is not greater than the maximum allowable velocity.
9448	The velocity of the z-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the z-axis is not greater than the maximum allowable velocity.
9449	The velocity of the auxiliary axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the auxiliary axis is not greater than the maximum allowable velocity.
9450	Failed to obtain the number of axis groups.	Update the background software to the latest version.

Fault Code	Fault Information	Troubleshooting
9451	Internal fault	Contact the manufacturer.
9452	The instruction is called when the axis is in StandStill state.	Do not call this instruction when the axis is StandStill state.
9453	The maximum allowable velocity is exceeded.	Ensure that the target velocity of the instruction is not greater than the maximum velocity specified on the axis group configuration interface.
9454	The maximum allowable acceleration (deceleration) is exceeded.	Ensure that the target acceleration (deceleration) of the instruction is not greater than the maximum acceleration (deceleration) specified on the axis group configuration interface.
9455	The axis group becomes faulty due to an error reported by the linear interpolation instruction.	Identify the first linear interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9456	The axis group becomes faulty due to an error reported by the circular interpolation instruction.	Identify the first circular interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9457	The axis group becomes faulty due to an error reported by the axis group stop instruction.	Identify the first axis group stop instruction that reports the error and troubleshoot the fault based on the fault code.
9458	The axis group becomes faulty due to an error reported by the axis group pause instruction.	Identify the first axis group pause instruction that reports the error and troubleshoot the fault based on the fault code.
9459	The x-axis in the axis group is performing the interpolation algorithm for another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9460	The y-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9461	The z-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9462	The auxiliary axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9463	When the MC_GroupStop instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupStop instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupStop instruction when the axes enter the synchronous mode due to other instructions.
9464	When the linear or circular interpolation instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the linear or circular interpolation instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the linear or circular interpolation instruction when the axes enter the synchronous mode due to other non-axis-group instructions.
9465	When the MC_GroupHalt instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupHalt instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupHalt instruction when the axes enter the synchronous mode due to other instructions.
9466	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9467	AddLength in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9468	The MC_MoveEllipse instruction fails and causes shutdown.	Find the MC_MoveEllipse instruction that caused the failure and check the fault code of the instruction to further confirm the fault.

Fault Code	Fault Information	Troubleshooting
9469	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9470	CircMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9471	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9472	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9473	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9474	Deceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9475	BufferMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9476	The set center point, long axis length, and short axis length are improper and cannot form an ellipse.	Ensure that the parameter value is within the allowable range.
9477	The property of the x-axis in the axis group instruction does not support the interpolation motion.	Ensure that the x-axis is not in single-axis mode.
9478	The property of the y-axis in the axis group instruction does not support the interpolation motion.	Ensure that the y-axis is not in single-axis mode.
9479	The property of the z-axis in the axis group instruction does not support the interpolation motion.	Ensure that the z-axis is not in single-axis mode.
9480	The property of the auxiliary axis in the axis group instruction does not support the interpolation motion.	Ensure that the auxiliary axis is not in single-axis mode.

3.11 MC Axis Control Instructions (CANopen)

3.11.1 Instruction List

The following table lists the CANopen axis control instructions.

Instruction Category	Instruction	Function
CANopen axis control instruction	MC_Power_CO	Enable servo axis through communication
	MC_Reset_CO	Reset servo axis fault through communication
	MC_ReadActualPosition_CO	Read current position of axis through communication
	MC_ReadActualVelocity_CO	Read current velocity of axis through communication
	MC_Halt_CO	Halt servo axis through communication
	MC_Stop_CO	Stop servo axis through communication
	MC_MoveAbsolute_CO	Control absolute positioning of axis through communication
	MC_MoveRelative_CO	Control relative positioning of axis through communication
	MC_MoveVelocity_CO	Control axis velocity through communication
	MC_Jog_CO	Control axis jogging through communication
	MC_Home_CO	Control axis homing through communication
	MC_WriteParameter_CO	Write axis parameters through communication
	MC_ReadParameter_CO	Read axis parameters through communication
	MC_SetOverride	Adjust target velocity during motion

3.11.2 MC_Power_CO

This instruction enables or disables a servo axis.

MC_Power_CO – Enable servo axis through communication

Graphic Block

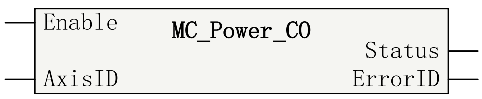


Table 3–246 Instruction format

16-bit Instruction	MC_Power_CO: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Status	Axis state	Yes	OFF	ON/OFF	BOOL
D2	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–247 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	-	√	-	-
D1	√[1]	-	√	-	√	√	-	-	-
D2	-	-	√	√	√	√	-	-	-

Note

- [1] The X element is not supported.
- The MC_Power_CO instruction can be executed only once for each axis.

Function and Instruction Description

"AxisID" specifies the ID of the controlled axis, which ranges from K1 to K16.

"Status" specifies the actual state output of the axis. "ON" indicates that the axis is enabled, and "OFF" indicates that the axis is disabled.

"ErrorID" specifies the error code. For details, see the "Error Codes of CANopen Axis Control Instructions".

The MC_Power_CO instruction writes to the corresponding control word (6040h) based on the read status word (6041h) to enable the axis. The following table describes the correspondence between the status word (6041h) and the control word (6040h).

<table>
<tr><th>Flow State</th><th colspan="2">Status Word (6041h)</th><th colspan="2">Control Word (6040h)</th></tr>
<tr><td rowspan="8">ON</td><td>Not ready to switch on</td><td>xxxx xxxx x0xx 0000 b</td><td rowspan="2">Shutdown</td><td rowspan="2">0000 0000 0000 0110 b</td></tr>
<tr><td>Switch on disabled</td><td>xxxx xxxx x1xx 0000 b</td></tr>
<tr><td>Ready to switch on</td><td>xxxx xxxx x01x 0001 b</td><td>Switch on</td><td>0000 0000 0000 0111 b</td></tr>
<tr><td>Switched on</td><td>xxxx xxxx x01x 0011 b</td><td>Switch on + enable operation</td><td>0000 0000 0000 1111 b</td></tr>
<tr><td>Fault reaction active</td><td>xxxx xxxx x0xx 1111 b</td><td rowspan="2">-</td><td rowspan="2">xxxx xx00 xx00 xxxx b</td></tr>
<tr><td>Fault</td><td>xxxx xxxx x0xx 1000 b</td></tr>
<tr><td colspan="2">Others</td><td>-</td><td>xxxx xxxx xxxx xxxx b</td></tr>
<tr><td rowspan="4">OFF</td><td>Ready to switch on</td><td>xxxx xxxx x01x 0001 b</td><td rowspan="3">Disable voltage</td><td rowspan="3">0000 0000 0000 0000 b</td></tr>
<tr><td>Switched on</td><td>xxxx xxxx x01x 0011 b</td></tr>
<tr><td>Operation enabled</td><td>xxxx xxxx x01x 0111 b</td></tr>
<tr><td colspan="2">Others</td><td>-</td><td>xxxx xx00 xx00 xxxx b</td></tr>
</table>

In this table, x indicates any value (status word) or remains unchanged (control word).

3.11.3 MC_Reset_CO

This instruction resets errors of an axis and makes the axis enter the StandStill or Disabled state.
MC_Reset_CO – Reset servo axis fault through communication

Graphic Block

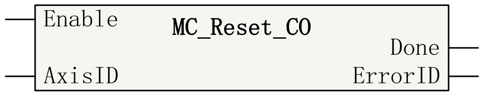


Table 3–248 Instruction format

16-bit Instruction	MC_Reset_CO: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–249 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction resets faults of a CANopen bus axis and makes the axis enter the StandStill or Disabled state.

"AxisID" specifies the ID of the controlled axis, which ranges from K1 to K16.

"Done" is output after the reset operation is completed.

"ErrorID" specifies the error code. For details, see the "Error Codes of CANopen Axis Control Instructions".

The MC_Reset_CO instruction writes to the corresponding control word (6040h) based on the read status word (6041h) to reset the axis fault. The following table describes the correspondence between the status word (6041h) and the control word (6040h).

Flow State	Status Word (6041h)		Control Word (6040h, bit7)
ON	Switch on disabled	xxxx xxxx x1xx 0000b	0
	Operation enabled	xxxx xxxx x01x 0111b	-
	Fault	xxxx xxxx x0xx 1000b	1
	-	Others	x
↑	-	xxxx xxxx xxxx xxxxb	0
OFF	-	xxxx xxxx xxxx xxxxb	x

In this table, x indicates any value (status word) or remains unchanged (control word).

3.11.4 MC_ReadActualVelocity_CO

This instruction reads the current velocity of an axis.

MC_ReadActualVelocity_CO – Read current velocity of axis through communication

Graphic Block

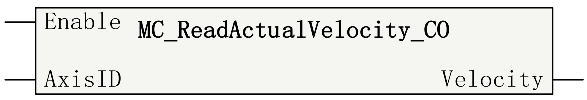


Table 3–250 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualVelocity_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Velocity	Current velocity	Yes	0	-	REAL

Table 3–251 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	√	-	-	-

Function and Instruction Description

This instruction reads the actual velocity of a CANopen bus axis.

"AxisID" specifies the ID of the axis to be read, which ranges from 1 to 16.

"Position" specifies the current position of the axis, which is a 32-bit floating-point number.

3.11.5 MC_ReadActualPosition_CO

This instruction reads the current position of an axis.

MC_ReadActualPosition_CO – Read current position of axis through communication

Graphic Block

Table 3–252 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ReadActualPosition_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Position	Current position	Yes	0	-	REAL

Table 3–253 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	√	-	-	-

Function and Instruction Description

This instruction reads the actual position of a CANopen bus axis.

"AxisID" specifies the ID of the axis to be read, which ranges from 1 to 16.

"Position" specifies the current position of the axis, which is a 32-bit floating-point number.

3.11.6 MC_Halt_CO

This instruction halts the current motion of an axis so that the axis can respond to other motion instructions.

MC_Halt_CO – Halt servo axis through communication

Graphic Block

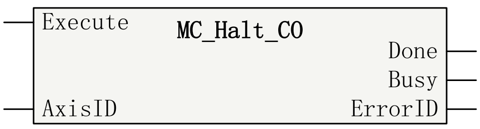


Table 3–254 Instruction format

16-bit Instruction	MC_Halt_CO: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type

S1	Axis	Axis ID	No	-	-	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	-	-	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–255 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

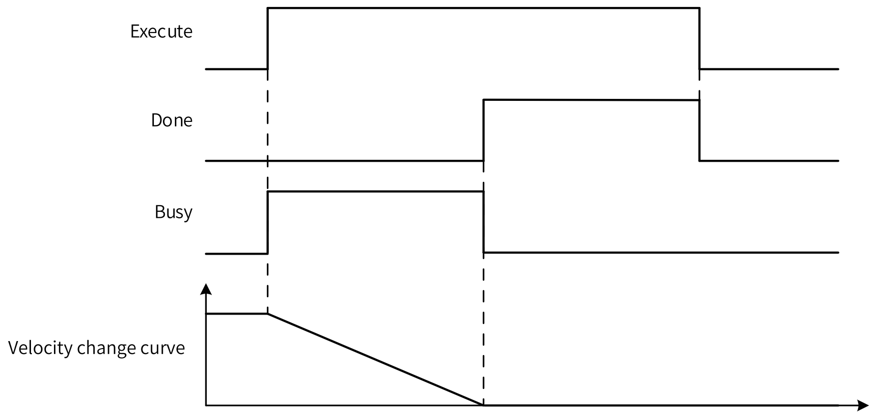
This instruction halts the current motion of a CANopen bus axis so that the axis can respond to other motion instructions.

The MC_Halt_CO instruction can be aborted by instructions including MC_MoveAbsolute_CO, MC_MoveRelative_CO, MC_MoveVelocity_CO, and MC_Jog_CO.

Table 3–256 Operation procedure of the MC_Halt_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6040h.bit4 = 0	Trigger the motion halt operation using the control word. Write 0 to the target velocity.
	6040h.bit5 = 0	
	6040h.bit6 = 0	
	6040h.bit8 = 1	
	60FFh = 0	
2	606Ch = 0	Wait until the halt operation is completed.
	6061h = 3 and 6041h.bit13 = 1	
	6061h != 3 and 6041h.bit10 = 1	
3	6060h = 1	Switch to the position mode.

Timing Diagram



3.11.7 MC_Stop_CO

This instruction stops an axis and makes it enter the Stopping state so that the axis no longer responds to any motion instruction.

MC_Stop_CO – Stop servo axis through communication

Graphic Block

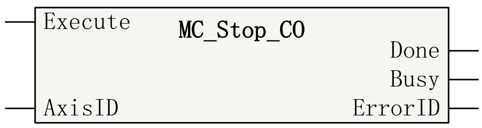


Table 3–257 Instruction format

16-bit Instruction	MC_Stop_CO: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see the *“3.11.16 Error Codes of CANopen Axis Control Instructions” on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–258 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

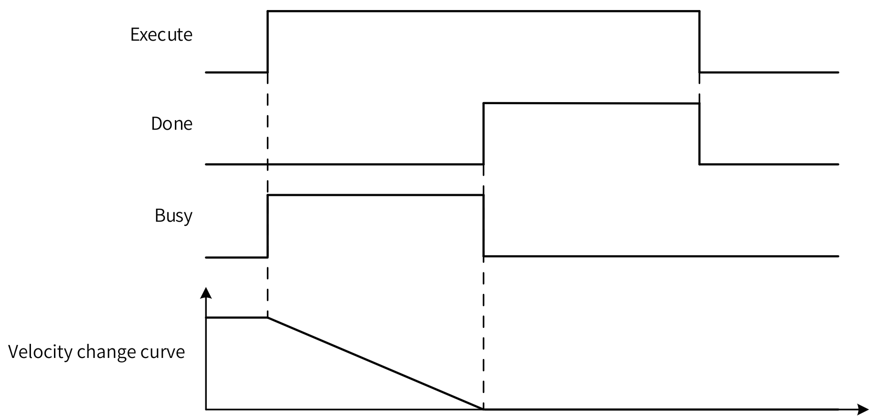
Function and Instruction Description

This instruction stops a CANopen bus axis and makes it enter the Stopping state so that the axis no longer responds to any motion instruction.

Table 3–259 Operation procedure of the MC_Stop_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6040h.bit4 = 0	Trigger the motion halt operation using the control word. Write 0 to the target velocity.
	6040h.bit5 = 0	
	6040h.bit6 = 0	
	6040h.bit8 = 1	
	60FFh = 0	
2	606Ch = 0	Wait until the halt operation is completed.
	6061h = 3 and 6041h.bit13 = 1	
	6061h != 3 and 6041h.bit10 = 1	
3	6060h = 1	Switch to the position mode.

Timing diagram



3.11.8 MC_MoveVelocity_CO

MC_MoveVelocity_CO – Control axis velocity through communication

Graphic Block

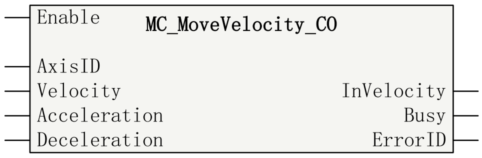


Table 3–260 Instruction format

16-bit Instruction	-					
32-bit Instruction	C_MoveVelocity_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	Velocity	Velocity	No	-	-	REAL
S3	Acceleration	Acceleration	No	-	-	REAL
S4	Deceleration	Deceleration	Yes	Acceleration	-	REAL
D1	InVelocity	Velocity reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–261 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

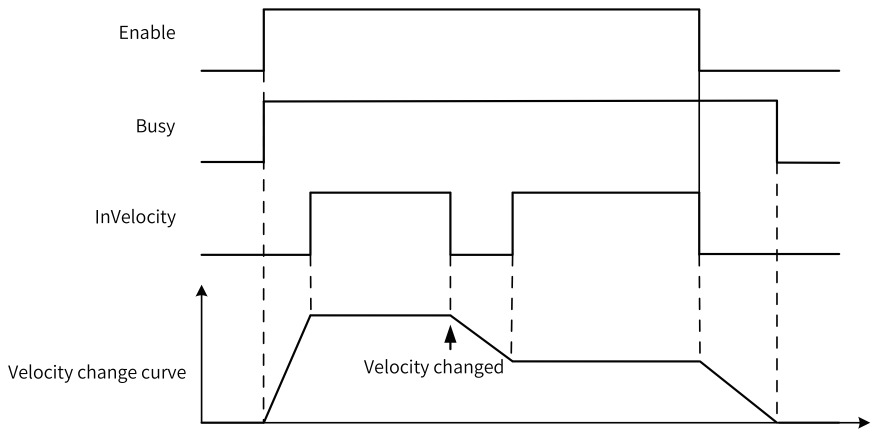
This instruction controls a CANopen bus axis to move at the specified velocity. When the specified velocity ("Velocity") is greater than 0, the axis moves in the forward direction; when it is less than 0, the axis moves in the reverse direction. The velocity can be modified during execution of this instruction, and the modification takes effect in real time.

If "Deceleration" is not specified, that is, it is left empty, the specified acceleration ("Acceleration") is used as the deceleration by default.

Table 3–262 Operation procedure of the MC_MoveVelocity_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6040h.bit8 = 0	Reset the Halt bit of the control word.
2	6083h = Acceleration	Write the acceleration.
3	6084h = Deceleration	Write the deceleration.
4	6060h = 3	Switch to the velocity mode.
5	6061h = 3	Wait for the axis to switch to the velocity mode.
6	60FFh = Target velocity	Set the target velocity.
	6041h.bit10 = 1	The target velocity is reached.
	60FFh < 0, 6041h.bit11 = 1, and 60FDh.bit0 = 1 60FFh = 0	The negative limit is reached, and motion ends.
	607Ah > 0, 6041h.bit11 = 1, and 60FDh.bit1 = 1 60FFh = 0	The positive limit is reached, and motion ends.
	60FFh = 0	The instruction flow becomes inactive, and motion ends.

Timing Diagram



3.11.9 MC_MoveRelative_CO

MC_MoveRelative_CO – Control relative positioning of axis through communication

Graphic Block

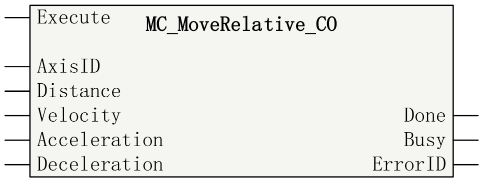


Table 3–263 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveRelative_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	Distance	Target distance	No	-	-	REAL
S3	Velocity	Maximum velocity	No	-	-	REAL
S4	Acceleration	Acceleration	No	-	-	REAL
S5	Deceleration	Deceleration	Yes	Accelera-tion	-	REAL
D1	Done	Completion flag, indicating that the target position is reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–264 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

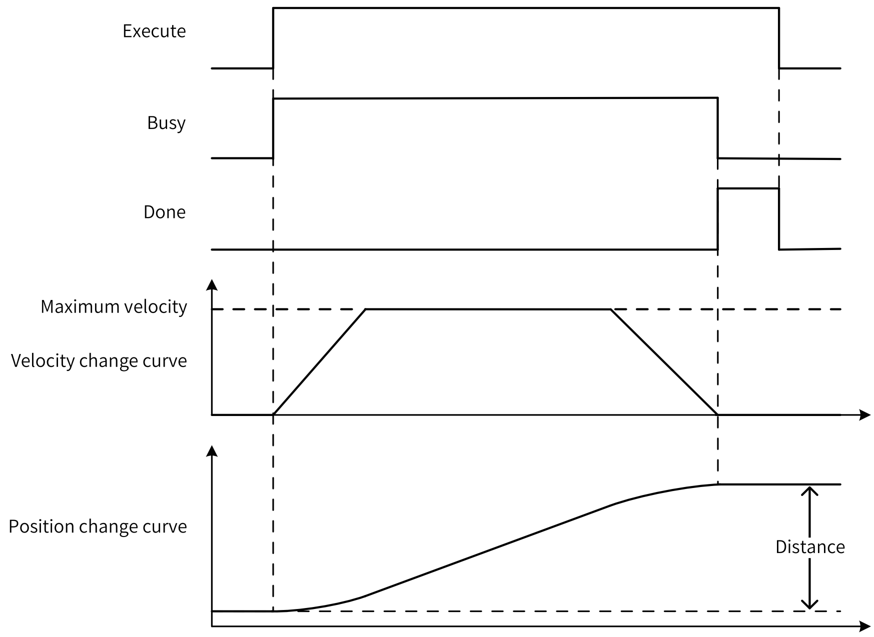
This instruction implements the relative positioning function of a CANopen bus axis. It controls an axis to move from the current position for a specified distance.

If "Deceleration" is not specified, that is, it is left empty, the specified acceleration ("Acceleration") is used as the deceleration by default.

Table 3–265 Operation procedure of the MC_MoveRelative_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6060h = 1	Switch to the position mode.
2	6061h = 1	Wait for the axis to switch to the position mode.
3	6040h.bit5 = m	Write the corresponding mode to the control word. m is 1 when the buffer mode (parameter number: K1000) is set to 0; otherwise, m is 0.
	6040h.bit6 = 1	
	6040h.bit8 = 0	
	6040h.bit9 = 0	
4	607Ah = Position	Write the (relative) target position and positioning velocity.
	6081h = Velocity	
5	6083h = Acceleration	Write the acceleration.
6	6084h = Deceleration	Write the deceleration.
7	6040h.bit4 = 1	Trigger positioning.
8	6041h.bit12 = 1	Wait for positioning to start.
9	6040h.bit4 = 0	Trigger positioning reset.
10	607Ah < 0, 6041h.bit11 = 1, and 60FDh.bit0 = 1	The negative limit is reached, and positioning ends.
	607Ah > 0, 6041h.bit11 = 1, and 60FDh.bit1 = 1	The positive limit is reached, and positioning ends.
	6041h.bit10 = 1 and 6041h.bit12 = 0	The target position is reached, and positioning is completed.

Timing Diagram



3.11.10 MC_MoveAbsolute_CO

MC_MoveAbsolute_CO – Control absolute positioning of axis through communication

Graphic Block

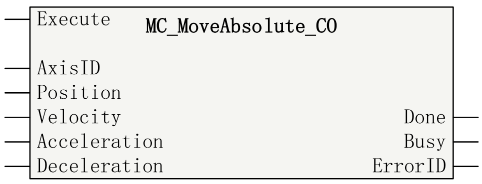


Table 3–266 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_MoveAbsolute_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	Position	Target position	No	-	-	REAL
S3	Velocity	Maximum velocity	No	-	-	REAL
S4	Acceleration	Acceleration	No	-	-	REAL
S5	Deceleration	Deceleration	Yes	Acceleration	-	REAL
D1	Done	Completion flag, indicating that the target position is reached	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–267 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
S3	-	-	-	√	√	√	-	√	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This instruction implements the absolute positioning function of a CANopen bus axis. It controls an axis to move to the specified position.

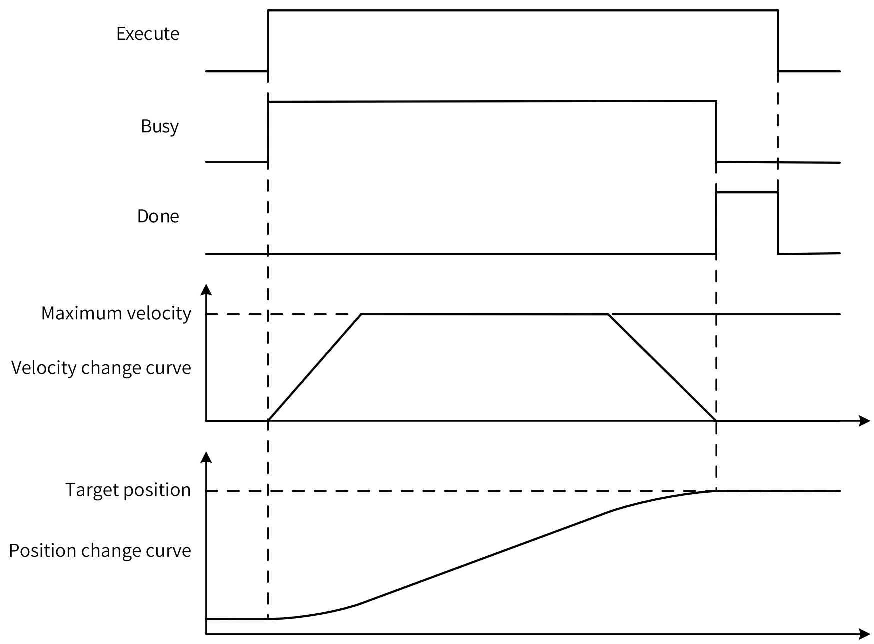
If "Deceleration" is not specified, that is, it is left empty, the specified acceleration ("Acceleration") is used as the deceleration by default.

Table 3–268 Operation procedure of the MC_MoveAbsolute_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6060h = 1	Switch to the position mode.
2	6061h = 1	Wait for the axis to switch to the position mode.
3	6040h.bit5 = m 6040h.bit6 = 0 6040h.bit8 = 0 6040h.bit9 = 0	Write the corresponding mode to the control word. m is 1 when the buffer mode (parameter number: K1000) is set to 0; otherwise, m is 0.
4	607Ah = Position 6081h = Velocity	Write the (absolute) target position and positioning velocity.
5	6083h = Acceleration	Write the acceleration.
6	6084h = Deceleration	Write the deceleration.
7	6040h.bit4 = 1	Trigger positioning.
8	6041h.bit12 = 1	Wait for positioning to start.
9	6040h.bit4 = 0	Trigger positioning reset.

Step	Operation/Condition	Description
10	607Ah < 6064h, 6041h.bit11 = 1, and 60FDh.bit0 = 1	The negative limit is reached, and positioning ends.
	607Ah > 6064h, 6041h.bit11 = 1, and 60FDh.bit1 = 1	The positive limit is reached, and positioning ends.
	6041h.bit10 = 1 and 6041h.bit12 = 0	The target position is reached, and positioning is completed.

Timing Diagram



3.11.11 MC_Home_CO

MC_Home_CO – Control axis homing through communication

Graphic Block

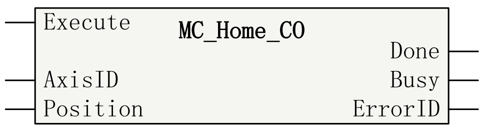


Table 3–269

16-bit Instruction	-					
32-bit Instruction	MC_Home_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	Position	Target position after homing	Yes	0	-	REAL

D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	ErrorID	Fault code	Yes	0	*1	INT

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–270 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

This function implements homing of a CANopen bus axis.

You need to set the homing mode and velocity on the CANopen configuration interface. For details about the homing modes, see the relevant servo/motor drive manual.

Table 3–271 Operation procedure of the MC_Home_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6060h = 6	Switch to homing mode.
2	6061h = 6	Wait for the axis to switch to the homing mode.
3	607Ch = Home offset	Set the home offset.
4	6040h.bit4 = 1	Start homing.
5	6041h.bit10 = 1 and 6041h.bit13 = 1	Homing failed
	6041h.bit10 = 1 and 6041h.bit12 = 1	Homing succeeded.

3.11.12 MC_Jog_CO

MC_Jog_CO – Control axis jogging through communication

Graphic Block

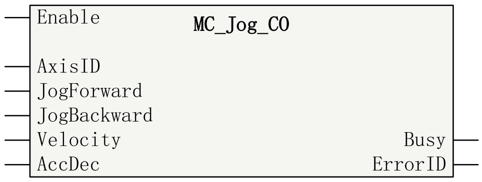


Table 3–272 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_Jog_CO: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	JogForward	Jogging in forward direction, triggered on the rising edge	No	-	ON/OFF	BOOL
S3	JogBackward	Jogging in reverse direction, triggered on the rising edge	No	-	ON/OFF	BOOL
S4	Velocity	Target velocity	No	-	-	REAL
S5	AccDec	Acceleration/Deceleration	No	-	-	REAL
D1	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D2	ErrorID	Fault code	Yes	0	*1	INT

Note

*1: For details, see the *“3.11.16 Error Codes of CANopen Axis Control Instructions” on page 517Error Codes of CANopen Axis Control Instructions* section.

Table 3–273 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	-	√	-	-	√	-	-	-
S3	√	-	√	-	-	√	-	-	-
S4	-	-	-	√	√	√	-	√	-
S5	-	-	-	√	√	√	-	√	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

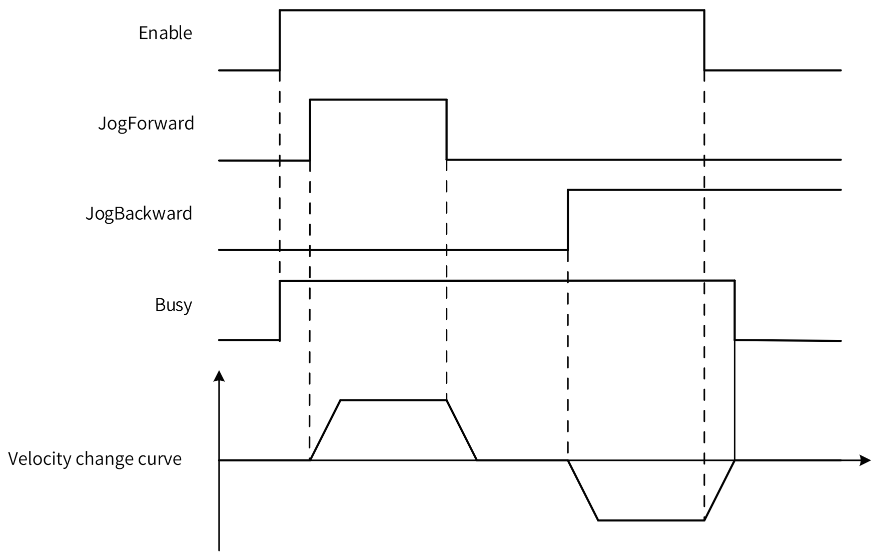
This instruction implements the jogging function of a CANopen bus axis. When JogForward is active, the axis moves forward at the velocity specified by "Velocity"; when JogBackward is active,

the axis moves in reverse direction at the velocity specified by "Velocity". When both JogForward and JogBackward are active, the axis stops motion.

Table 3–274 Operation procedure of the MC_Jog_CO instruction on a CANopen object

Step	Operation/Condition	Description
1	6040h.bit8 = 0	Reset the Halt bit of the control word.
2	6083h = Acceleration/Deceleration	Write the acceleration.
3	6084h = Acceleration/Deceleration	Write the deceleration.
4	6060h = 3	Switch to the velocity mode.
5	6061h = 3	Wait for the axis to switch to the velocity mode.
6	Forward jogging: 60FFh = Target velocity Reverse jogging: 60FFh = –Target velocity Others: 60FFh = 0	Perform forward/reverse jogging.
	60FFh < 0, 6041h.bit11 = 1, and 60FDh.bit0 = 1 60FFh = 0	The negative limit is reached, and jogging ends.
	607Ah > 6040h, 6041h.bit11 = 1, and 60FDh. bit1 = 1 60FFh = 0	The positive limit is reached, and jogging ends.
	60FFh = 0	The instruction flow becomes inactive, and jogging ends.

Timing Diagram



3.11.13 MC_WriteParameter_CO

MC_WriteParameter_CO – Write axis parameters through communication

Graphic Block

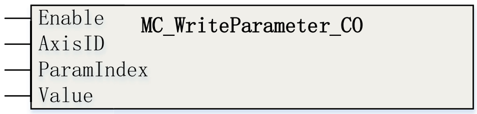


Table 3–275 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_WriteParameter_CO					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	ParamIndex	Parameter number	No	-	*1	INT
S3	Value	Parameter value	No	-	-	DINT

Note

*1: See the following parameter list.

Table 3–276 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-

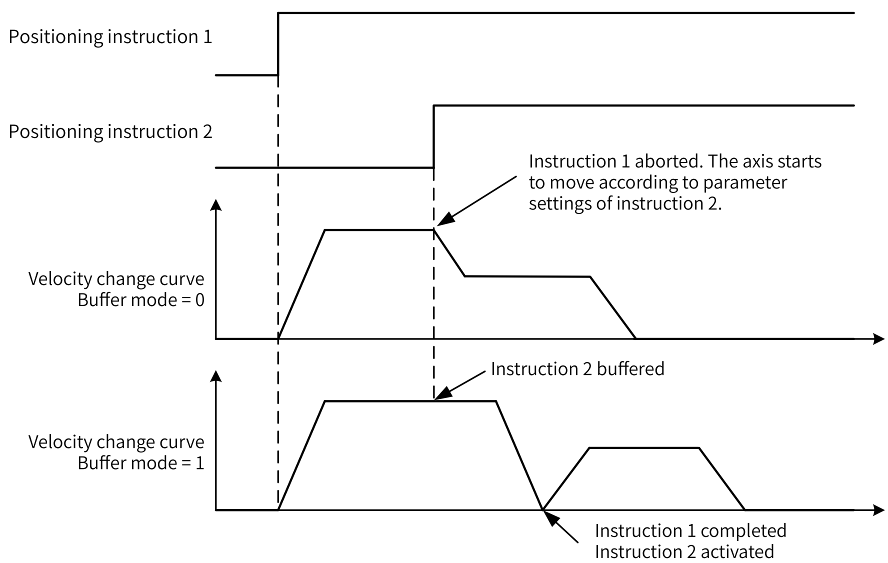
Function and Instruction Description

This instruction is used to set parameters for a CANopen bus axis. The following table lists the parameters.

Table 3–277 Parameter list

Parameter Number	Name	Data Type	Read/Write	Description
K1000	Buffer mode	UINT32	Read-write	Positioning buffer mode 0 (default): Trigger immediately without buffering 1: Wait for the current positioning to complete Reference: Buffer mode timing diagram
K1001	DI state	UINT32	Read	DI state [31:16]: Customized by the manufacturer [15:3]: Reserved [1]: Positive limit 0: Inactive; 1: Active [0]: Negative limit 0: Inactive; 1: Active
K1010	Axis state	INT32	Read	Current state of the axis –1: Not configured 0: Disabled 1: Standstill 2: Stopping 3: Homing 4: ContinuousMotion 5: DiscreteMotion 15: ErrorStop

Timing Diagram



3.11.14 MC_ReadParameter_CO

MC_ReadParameter_CO – Read axis parameters through communication

Graphic Block

Table 3–278 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_ReadParameter_CO					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisID	ID of the CANOpen axis to be operated	No	-	1 to 16	INT
S2	ParamIndex	Parameter number	No	-	*1	INT
D1	Value	Parameter value	Yes	-	-	DINT

Note

*1: See the following parameter list.

Table 3–279 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D1	-	-	-	√	√	√	-	-	-

Function and Instruction Description

This instruction is used to read parameters of a CANopen bus axis. The following table lists the parameters.

Table 3–280 Parameter list

Parameter Number	Name	Data Type	Read/Write	Description
K1000	Buffer mode	UINT32	Read-write	Positioning buffer mode 0 (default): Trigger immediately without buffering 1: Wait for the current positioning to complete Reference: Buffer mode timing diagram
K1001	DI state	UINT32	Read	DI state [31:16]: Customized by the manufacturer [15:3]: Reserved [1]: Positive limit 0: Inactive; 1: Active [0]: Negative limit 0: Inactive; 1: Active
K1010	Axis state	INT32	Read	Current state of the axis –1: Not configured 0: Disabled 1: Standstill 2: Stopping 3: Homing 4: ContinuousMotion 5: DiscreteMotion 15: ErrorStop

3.11.15 MC_SetOverride

This instruction adjusts the target velocity during motion.

MC_SetOverride Adjust target velocity during motion

Graphic Block

Table 3–281 Instruction format

16-bit Instruction	-					
32-bit Instruction	MC_SetOverride: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/Axis ID	No	-	-	_sMCAXIS_INFO
S2	VelFacter	VelFacter	Yes	100	0 to 500	REAL32
S3	AccFacter (Reserved)	Acceleration and deceleration overshoot (reserved)	Yes	100	0 to 500	REAL32
D1	Enabled	Enable flag	Yes	FALSE	TRUE/FALSE	BOOL
D2	Busy	Busy flag	Yes	FALSE	TRUE/FALSE	BOOL
D3	CommandA-borted	Execution abortion flag	Yes	FALSE	TRUE/FALSE	BOOL
D4	Error	Error flag	Yes	FALSE	TRUE/FALSE	BOOL
D5	ErrorID	Fault code	Yes	0	*1	INT16

Note

*1: For details, see the *"3.11.16 Error Codes of CANopen Axis Control Instructions" on page 517Error Codes of CANopen Axis Control Instructions* section.

Function and Instruction Description

This instruction adjusts the target velocity during the motion of the controlled axis by setting the velocity overshoot. This instruction applies only to the master axis. It cannot be used to set the velocity of a slave axis, and it does not affect the status of axes involved in synchronized motion.

Target velocity after adjustment = Current target velocity of the executing instruction x Velocity overshoot

The unit of the overshoot value is [%]. "100" indicates 100%. The overshoot ranges from 0 to 500, with a default value of 100. It cannot be less than 0. If the velocity overshoot is set to 0, the axis decelerates and moves at a velocity of 0. The axis maintains its original motion state. If the target velocity after adjustment exceeds the system's maximum velocity, the target velocity will be limited to the system's maximum velocity.

When the Enable signal transitions from TRUE to FALSE, the overshoot value returns to 100.

By executing the start from stop instruction, re-executing a motion instruction, or executing multiple motion instructions, you can ensure that the overshoot value corresponds to the newly set target velocity.

The following table lists the instructions for which the overshoot can be adjusted.

Table 3–282 Instructions with adjustable overshoot

Instructions with Adjustable Overshoot	
MC_MoveAbsolute (Absolute positioning)	MC_MoveRelative (Relative positioning)
MC_MoveVelocity (Velocity control)	MC_Jog (Jogging)

Instructions with Adjustable Overshoot	
MC_MoveVelocityCSV (Velocity control with adjustable pulse width)	MC_SyncMoveVelocity (Synchronous velocity control that supports PWM waveform)
MC_FollowVelocity (CSP-based synchronous velocity control)	(Reserved)

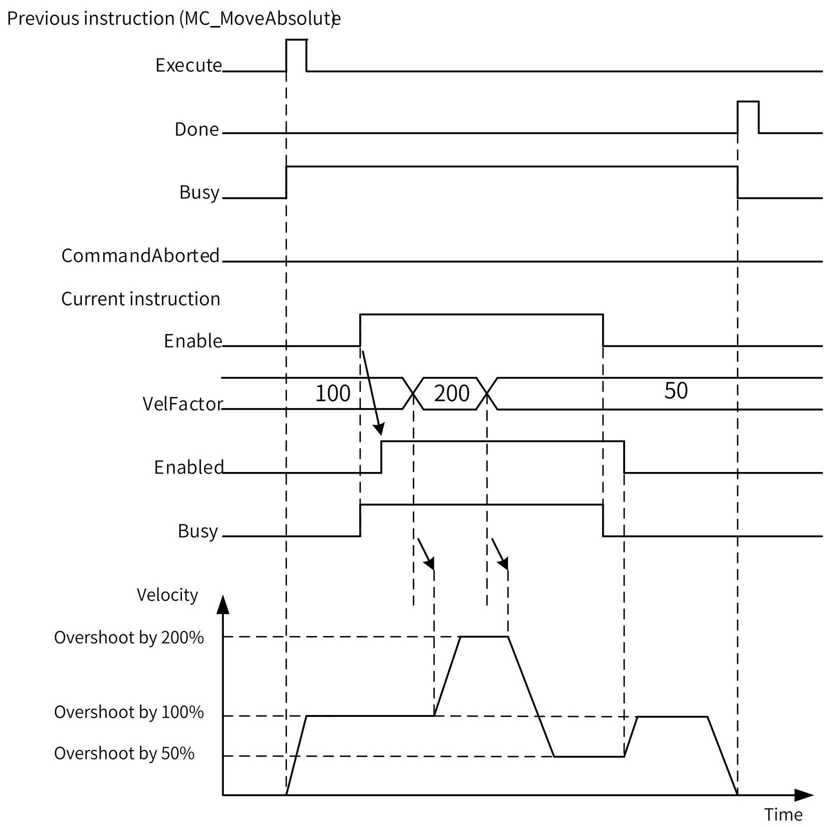
Note

The MC_SetOverRide instruction has no effect on MC_MoveSuperImposed.

Timing Diagram

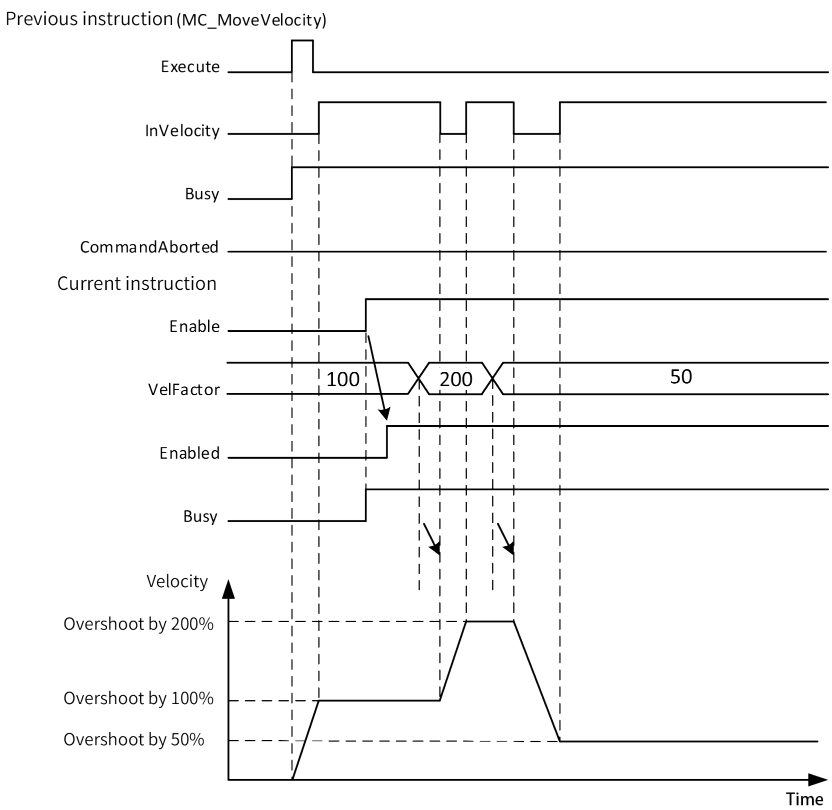
Executing MC_SetOverride in MC_MoveAbsolute (absolute positioning)

The following is a timing diagram illustrating the usage of the MC_SetOverride instruction in the MC_ MoveAbsolute (absolute positioning) instruction.



Executing MC_SetOverride in MC_MoveVelocity (velocity control)

After "InVelocity" becomes "TRUE" (indicating that the target velocity has been reached), it transitions to "FALSE" when the velocity is changed, and it changes back to "TRUE" when the target velocity is reached again.

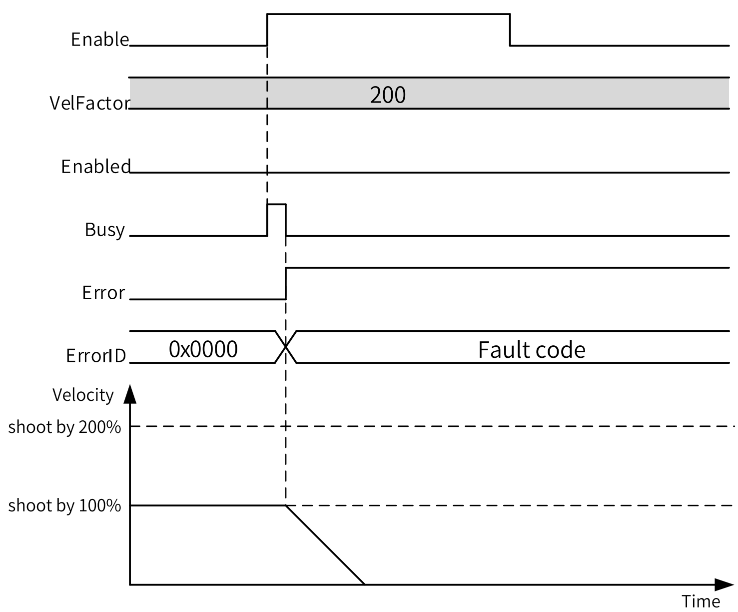


Timing diagram illustrating an exception

When an exception occurs during execution of this instruction, the "Error" flag is set to "TRUE". In the case of a minor fault, the axis stops its motion.

You can check the output of "ErrorID" to find the cause of the exception.

After the exception is resolved, the "Error" flag becomes "FALSE".



3.11.16 Error Codes of CANopen Axis Control Instructions

When an error occurs during use of CANopen axis control instructions, refer to the following error codes.

Table 3–283 Error codes

Code	Description
0	No error occurs.
1	The axis ID is incorrect. a) The axis ID is out of range (1 to 16). b) The axis ID does not exist in the CANopen configuration or a PDO configuration error occurs.
2	The parameters of the instruction are incorrect. a) The acceleration/deceleration in the MC_MoveAbsolute_CO, MC_MoveRelative_CO, MC_MoveVelocity_CO, or MC_Jog_CO instruction is less than or equal to 0. b) The velocity in the MC_MoveAbsolute_CO or MC_MoveRelative_CO instruction is less than or equal to 0.
3	The value of the instruction parameter (position or home offset) is out of range. (Note 1)
4	The value of the instruction parameter (velocity) is out of range. (Note 1)
5	The value of the instruction parameter (acceleration) is out of range. (Note 1)
6	The value of the instruction parameter (deceleration) is out of range. (Note 1)
8	Execution of the current instruction is stopped due to abortion by another instruction, Enable loss, or disconnection.
9	Execution of the current instruction is stopped due to forward overtravel. (Note 2)
10	Execution of the current instruction is stopped due to reverse overtravel. (Note 2)
11	Homing failed.
16	Failed to execute the current instruction because the axis is disabled.
17	Failed to execute the MC_Reset_CO instruction because the axis is not in ErrorStop state.
18	Failed to execute the current instruction because the axis is in Stopping state.
19	Failed to execute the current instruction because the axis is homing.
20	Failed to execute the current instruction because the axis is in ContinuousMotion state.
21	Failed to execute the current instruction because the axis is positioning.
31	Failed to execute the current instruction because the axis is in ErrorStop state.
250	Axis enabling timed out.
251	An error occurs on the servo/motor drive. (Note 3)
255	The servo/motor drive is disconnected. (Note 3)

Note

- Note 1: After conversion into the pulse unit, the value is beyond the range of a 32-bit integer.
- Note 2: The axis enters the ErrorStop state when overtravel occurs during motion. You need to reset the fault by executing MC_Reset_CO before triggering the axis to move in the reverse direction.
- Note 3: This error code applies only to the MC_Power_CO instruction. If this fault occurs during the execution of other instructions, an instruction abortion error is reported; if another instruction is triggered after this fault occurs, an error indicating that the axis is disabled is reported.

3.12 HC Axis Control Instructions (Pulse Input)

3.12.1 Instruction List

The following table lists the high-speed counter instructions.

Instruction Category	Instruction	Function
Bus encoder axis instruction (H5U)	ENC_Counter	Encoder enable
	ENC_Reset	Encoder reset
	ENC_Preset	Encoder preset
	ENC_TouchProbe	Encoder probe
	ENC_ArrayCompare	Encoder one-dimensional array comparison
	ENC_StepCompare	Encoder one-dimensional step comparison
	ENC_GroupArrayCompare	Encoder two-dimensional array comparison
	ENC_ReadStatus	Encoder state read
	ENC_DigitalOutput	Encoder DO control
	ENC_ResetCompare	Encoder comparison output reset
High-speed counter instruction (H5U)	HC_Counter	High-speed counter enable
	HC_Preset	High-speed counter preset
	HC_TouchProbe	High-speed counter probe
	HC_Compare	High-speed counter comparison
	HC_ArrayCompare	High-speed counter array comparison
	HC_StepCompare	High-speed counter step comparison

Instruction Category	Instruction	Function
Encoder axis instruction (Easy)	ENC_Counter	Encoder enable
	ENC_Reset	Encoder reset
	ENC_Preset	Encoder preset
	ENC_TouchProbe	Encoder probe
	ENC_ArrayCompare	Encoder one-dimensional array comparison
	ENC_StepCompare	Encoder one-dimensional step comparison
	ENC_Compare	Single-point comparison output
	ENC_GroupArrayCompare	Encoder two-dimensional array comparison
	ENC_ReadStatus	Encoder state read
	ENC_DigitalOutput	Encoder DO control
	ENC_ResetCompare	Encoder comparison output reset
	ENC_SetUnit	Gear ratio setting
	ENC_SetLineRotationMode	Rotation mode setting

3.12.2 ENC_Counter

This instruction is used to enable counting of the encoder axis.

ENC_Counter – Encoder enable

Graphic Block

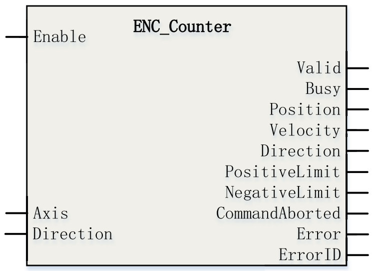
<table>
<tr><th>Instruction</th><th>Name</th><th>LD Expression</th><th>LiteST Expression</th></tr>
<tr><td>ENC_Counter</td><td>Encoder enable</td><td>
</td><td>ENC_Counter(Enable := ???,
Axis := ???,
Invert := ,
Valid => ,
Busy => ,
Position => ,
Velocity => ,
ActDirection => ,
PositiveLimit => ,
NegativeLimit => ,
CommandAborted => ,
Error => ,
ErrorID =>);</td></tr>
</table>

Table 3–284 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_Counter: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	Direction(Invert)	Counting direction reversal (local encoder axis) 0: Forward 1: Reverse	Yes	0	0 to 1	INT
D1	Valid	Active state	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	Position	Current position	Yes	0	Positive number 0 Negative number	REAL
D4	Velocity	Current velocity	Yes	0	Positive number 0 Negative number	REAL

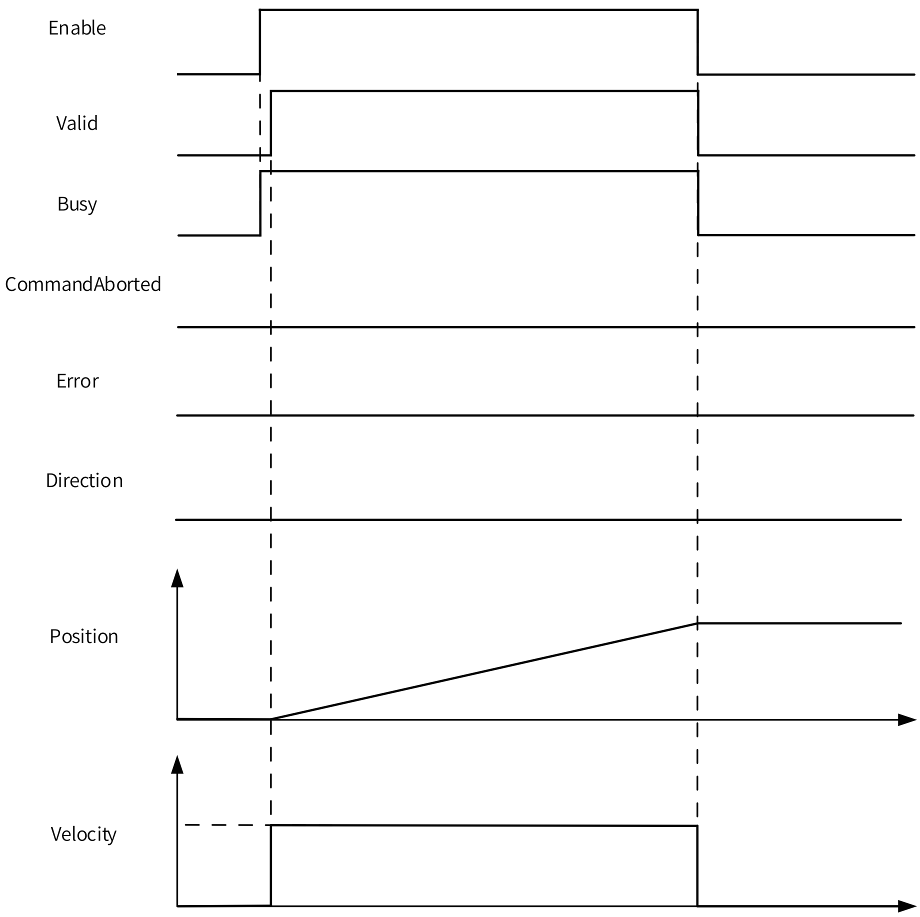
D5	Direction	Counting direction	Yes	0	ON OFF	BOOL
D6	PositiveLimit	Positive limit in linear mode	Yes	OFF	ON OFF	BOOL
D7	NegativeLimit	Negative limit in linear mode	Yes	OFF	ON OFF	BOOL
D8	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D9	Error	Error	Yes	OFF	ON OFF	BOOL
D10	ErrorID	Fault code	Yes	0	*1	INT

Note

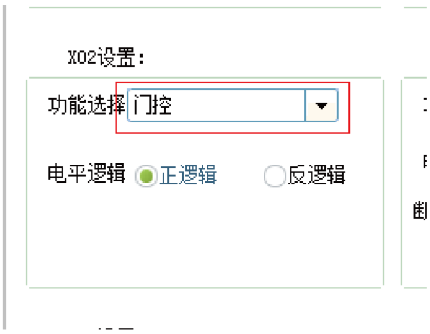
*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

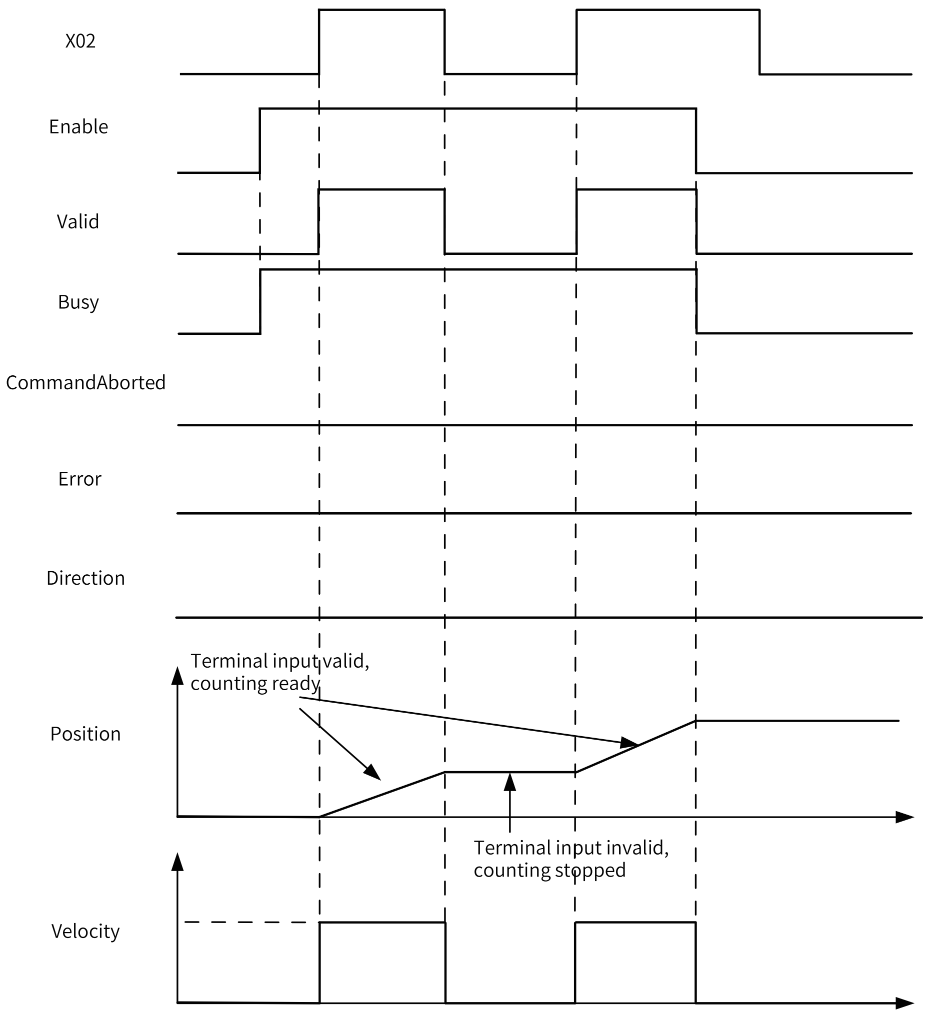
Function Description (Bus Encoder Axis)

- If the gating function of the DI terminal is not used, when the instruction input Enable is ON, the Busy signal and Valid signal are ON and the encoder axis starts counting; when the instruction input Enable is OFF, the Busy signal and Valid signal are OFF and the encoder axis stops counting. For example, when the GR10-2HCE module inputs 10 kHz pulses at constant speed:



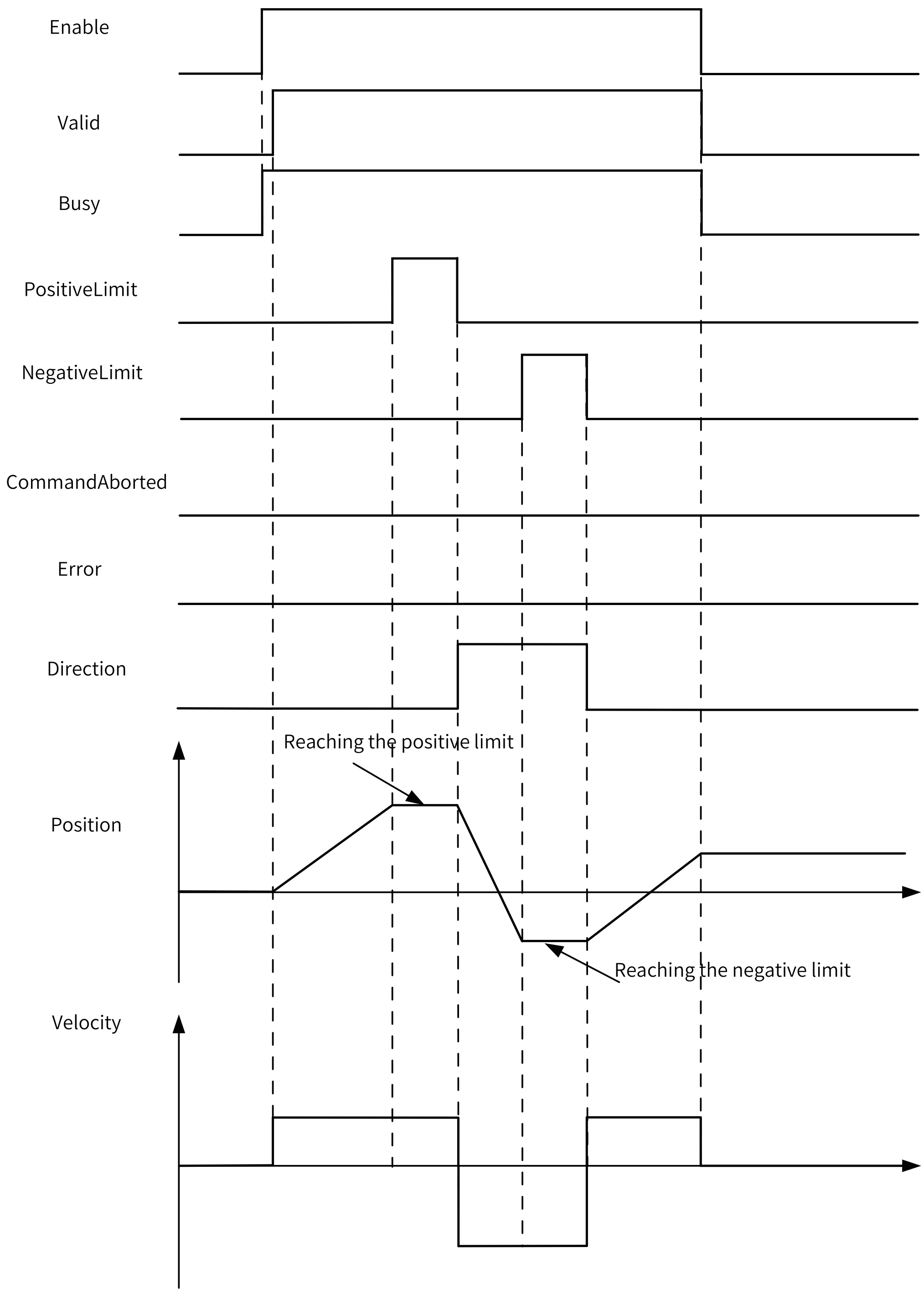
- If the gating function of the DI terminal is used (for example, X02 is assigned with the gating function), when the instruction input Enable is ON and X02 is OFF, the Busy signal is ON, the Valid signal is OFF, and the bus encoder axis suspends counting; when the instruction input Enable is ON and X02 is ON, the Busy signal and Valid signal are ON and the bus encoder axis starts counting; when the instruction input Enable is OFF, the encoder axis stops counting.





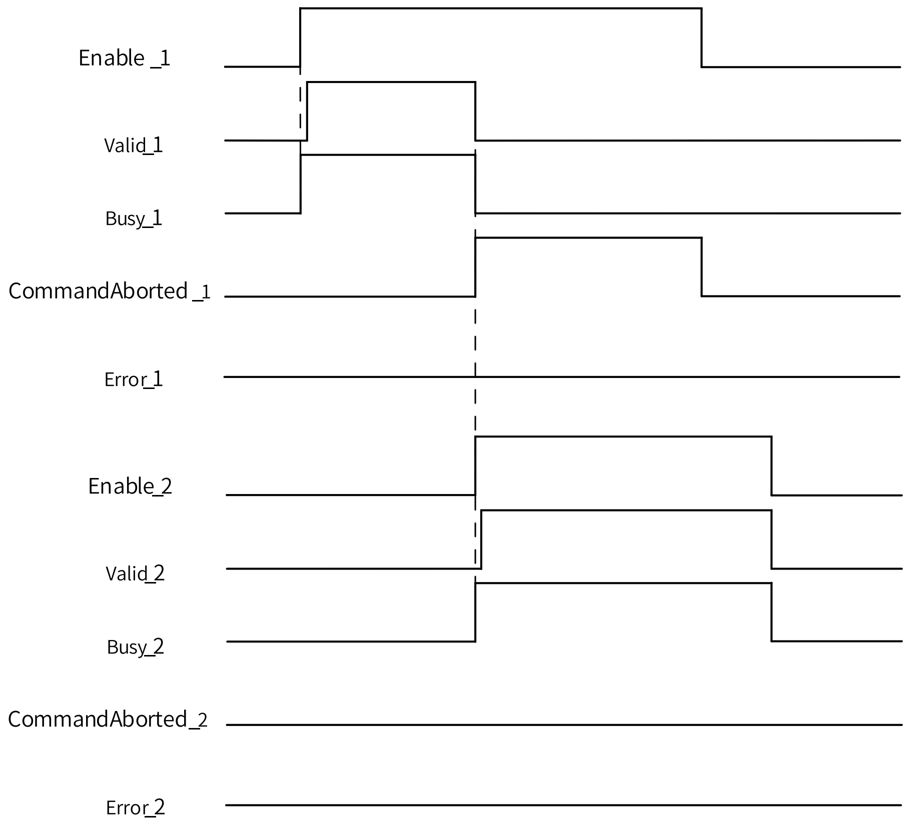
The timing diagram is as follows:

- In linear mode, if software limiting is enabled, when the count value reaches the limit, the counter stops counting and the limit signal output is active, when the pulse input is reversed, the limit signal is reset and the counter counts backwards.



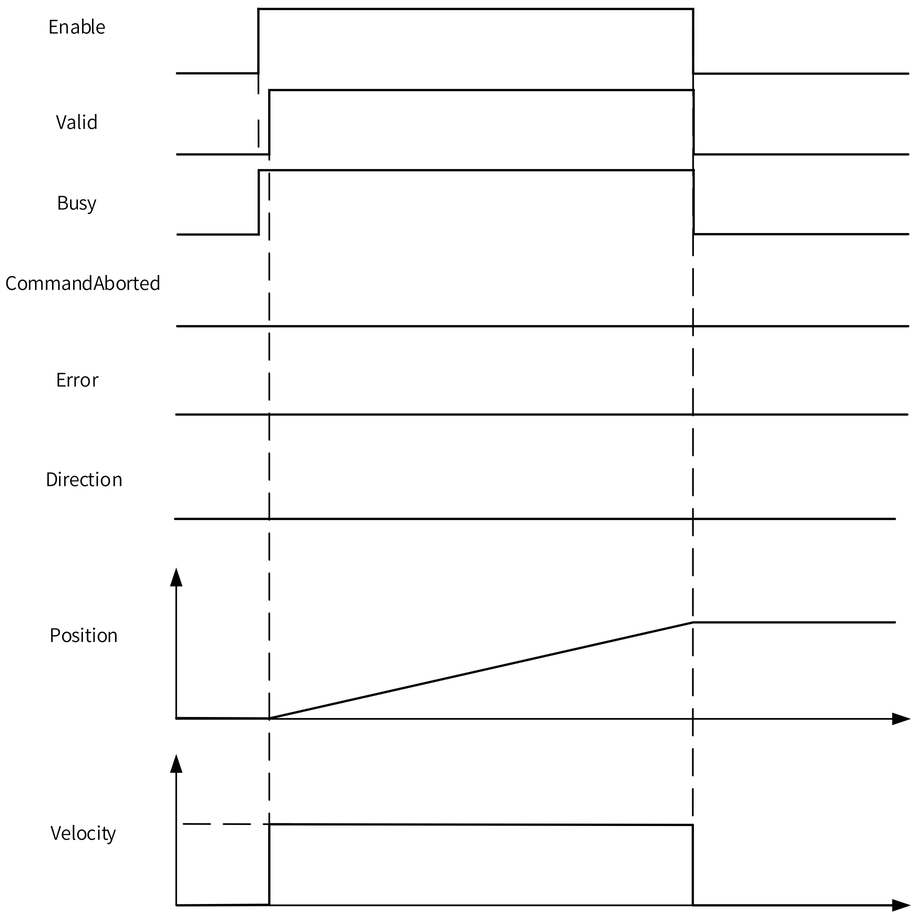
Multi-execution

When multiple ENC_Counter instructions are called to control the same axis, the instruction triggered first will be aborted by the instruction triggered later.

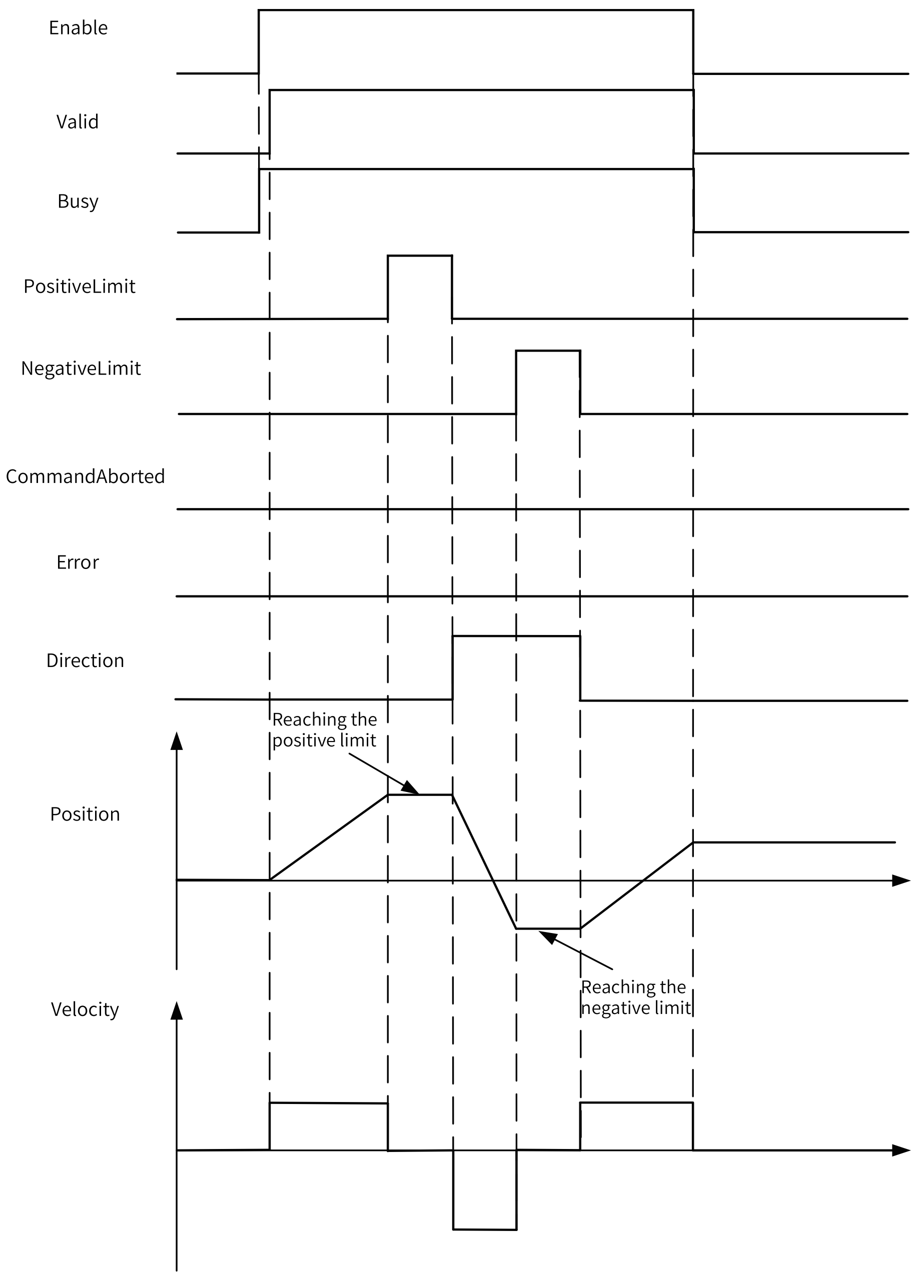


Function Description (Local Encoder Axis)

When the instruction input Enable is ON, the Busy signal and Valid signal are ON and the encoder axis starts counting; when the instruction input Enable is OFF, the Busy signal and Valid signal are OFF and the encoder axis stops counting.

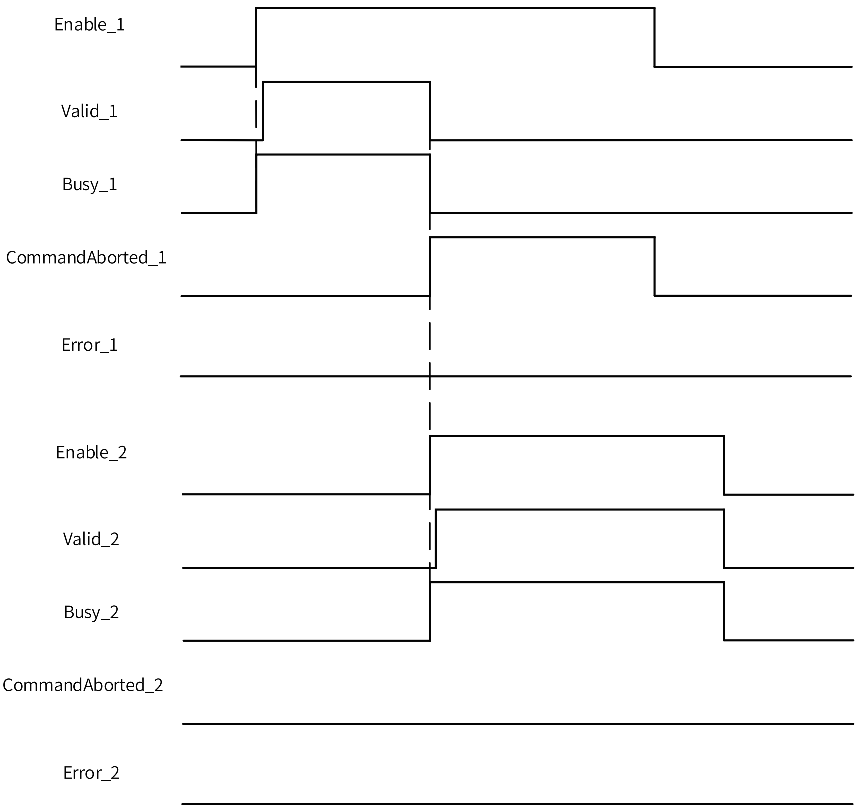


In linear mode, if software limiting is enabled, when the count value reaches the limit, the counter stops counting and the limit signal output is active; when the pulse input is reversed, the limit signal is reset and the counter counts backwards.



Multi-execution

When multiple ENC_Counter instructions are called to control the same axis, the instruction triggered first will be aborted by the instruction triggered later.



3.12.3 ENC_Reset

This instruction resets faults of a bus encoder axis.

ENC_Reset – Encoder reset

Graphic Block

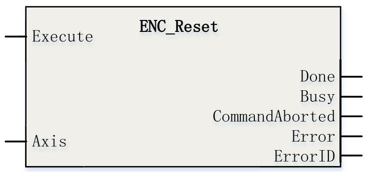
Instruction	Name	LD Expression	LiteST Expression
ENC_Reset	Encoder reset		ENC_Reset(Execute := ???, Axis := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–285 Instruction format

16-bit Instruction	ENC_Reset: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_ EXT_AXIS

D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Function Description

When the bus encoder axis fails and enters the ErrorStop state, this instruction can be used to reset the fault of the axis.

The local encoder axis does not support this instruction.

Timing Diagram – Omitted

3.12.4 ENC_Preset

ENC_Preset – Encoder preset

Graphic Block

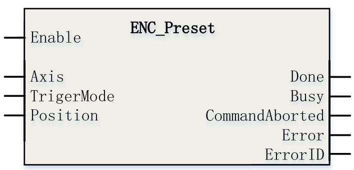
Instruction	Name	LD Expression	LiteST Expression
ENC_Preset	Encoder preset		ENC_Preset(Enable := ???, Axis := ???, TrigerMode := , Position := ???, Done => , Busy => , CommandAborted => ,

Table 3–286 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_Preset: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS

S2	TrigerMode	0: Triggered on the rising edge of the instruction 1: Triggered on the rising edge of the external input X 2: Triggered on the falling edge of the external input X (local encoder axis) 3: Triggered on the rising or falling edge of the external input X (local encoder axis) 4: Triggered by the Z signal (bus encoder axis)	Yes	0	0 to 4	INT
S3	Position	Preset position	Yes	0	Positive number 0 Negative number	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

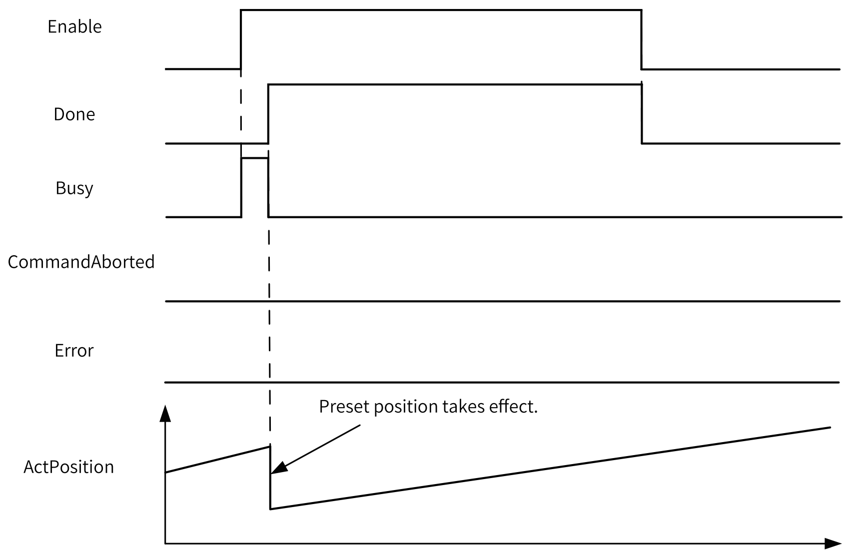
Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

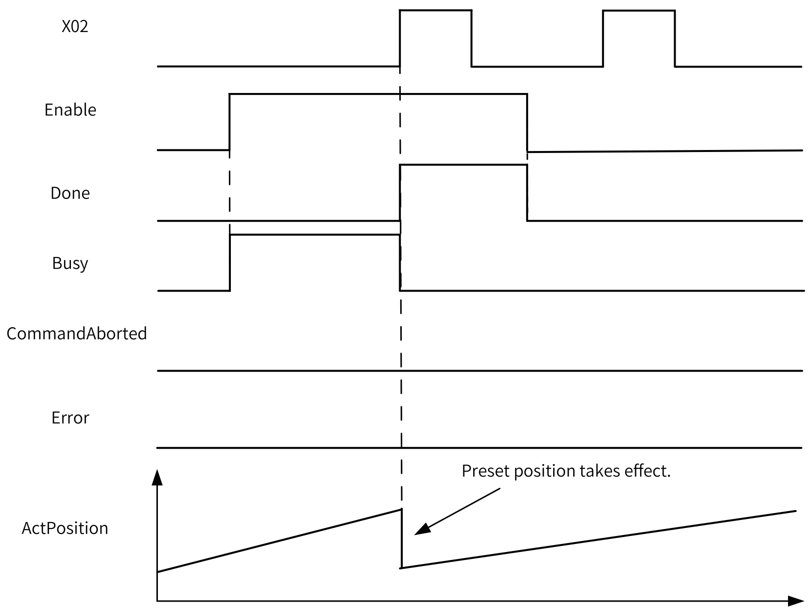
Function Description (Bus Encoder Axis)

This instruction can be used to set the current position of the encoder axis as the value of the input parameter "Position".

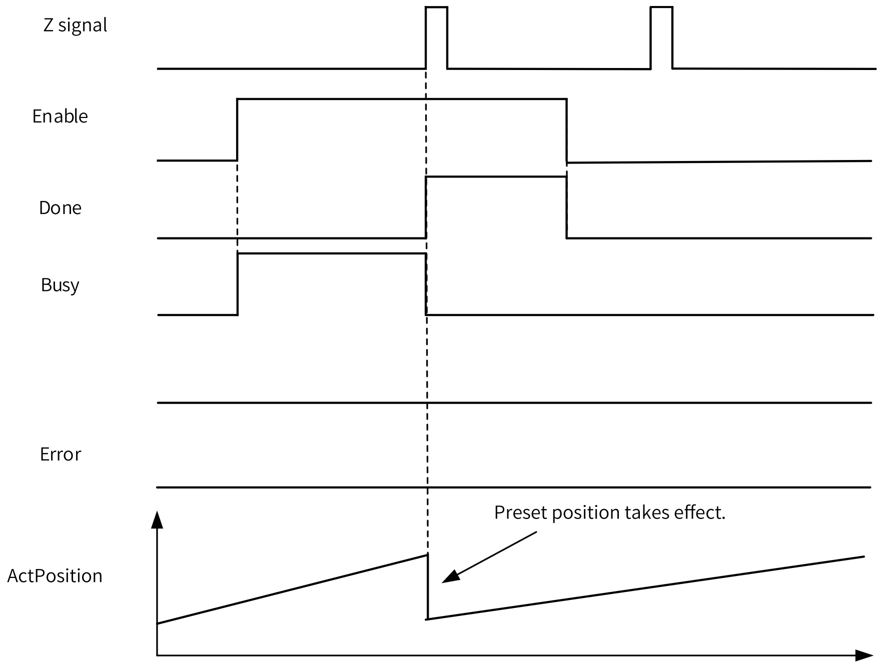
- When "TrigerMode" is set to 0, the encoder position is set when the Enable input is active high.



- When "TrigerMode" is set to 1, the encoder position is set on the rising edge of the DI terminal. To select this mode, you need to assign the DI terminal with the preset function. If multiple DI terminals are assigned with the preset function at the same time, the preset function can be implemented as long as one of the inputs is active.
  The timing diagram is as follows:

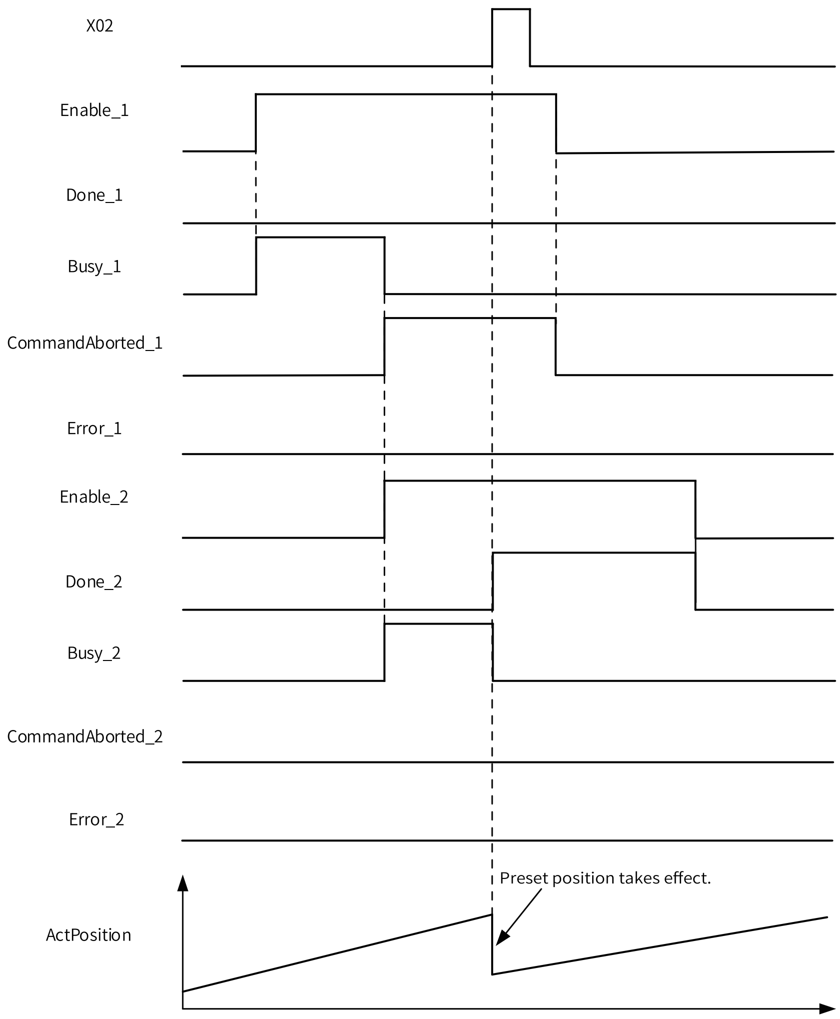


- When "TrigerMode" is set to 4, the preset function is completed when the rising edge of the Z signal is detected.



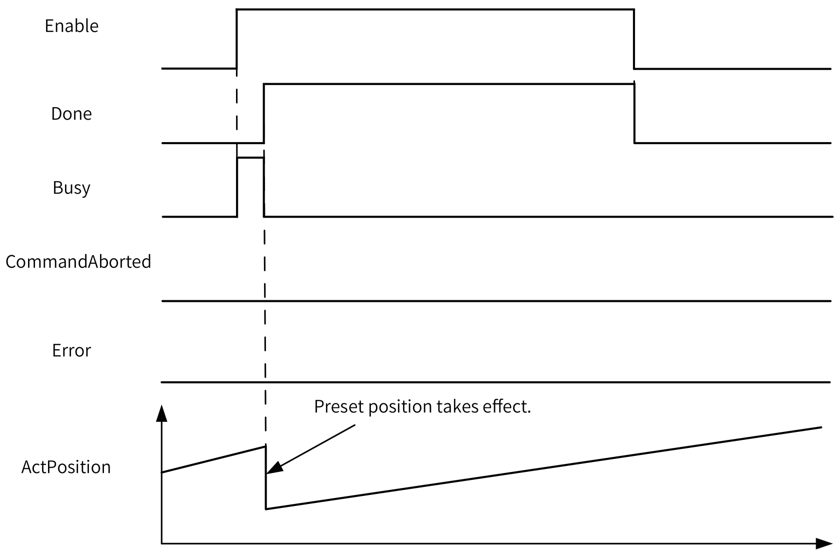
Multi-execution

When multiple preset instructions are called to set the position of the same axis, the instruction triggered first will be aborted by the instruction triggered later.



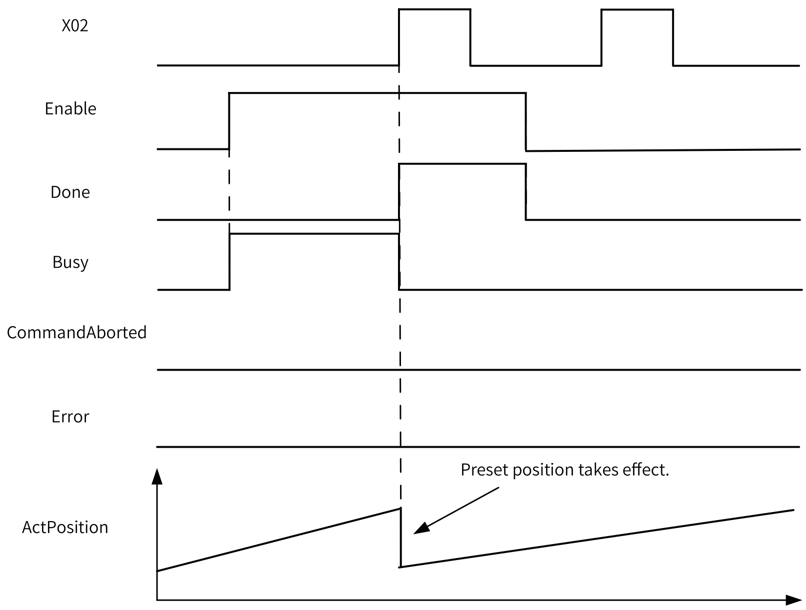
Function Description (Local Encoder Axis)

When "TrigerMode" is set to 0, the encoder position is set on the rising edge of the Enable input.

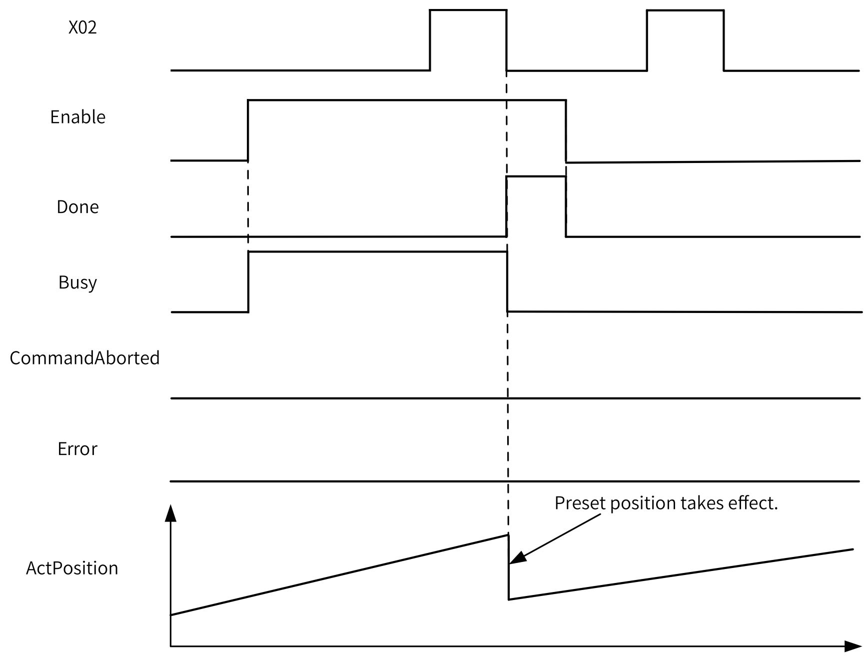


When "TrigerMode" is set to 1, the encoder position is set on the rising edge of the DI terminal. To select this mode, you need to assign the DI terminal with the preset function.

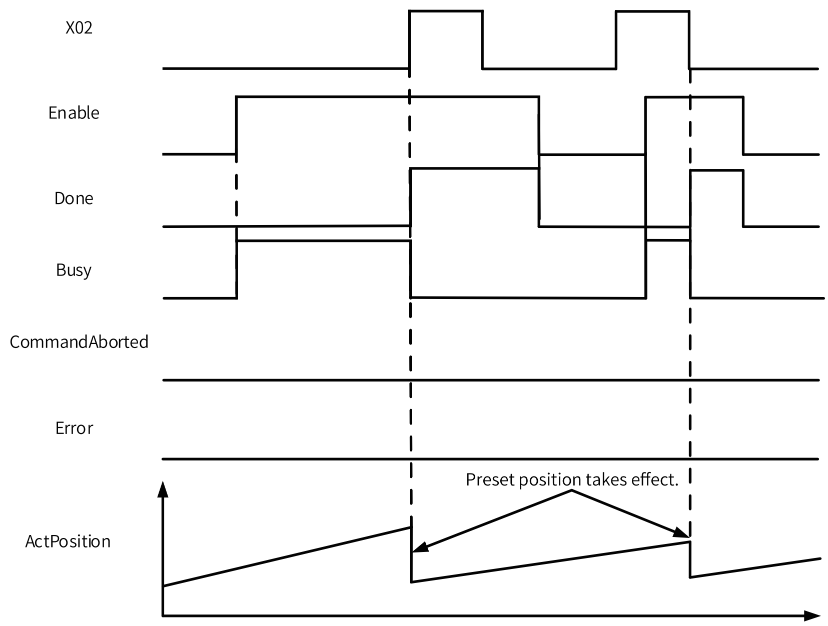
After the instruction is triggered, the position is preset on the rising edge of X02.



When "TrigerMode" is set to 2, the encoder position is set on the falling edge of the DI terminal.



When "TrigerMode" is set to 3, the encoder position is set on the rising or falling edge of the DI terminal. As long as the instruction is active, the position is set on whichever edge that arrives first.



3.12.5 ENC_TouchProbe

ENC_TouchProbe – Encoder probe

Graphic Block

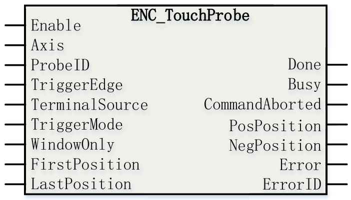
Instruction	Name	LD Expression	LiteST Expression
ENC_TouchProbe	Encoder probe		ENC_TouchProbe(Enable := ???, Axis := ???, ProbeID := ???, TriggerEdge := ???, TerminalSource := , TriggerMode := , WindowOnly := , FirstPosition := , LastPosition := , Done => , Busy => , CommandAborted => , PosPosition => , NegPosition => , Error => , ErrorID =>);

Table 3–287 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_TouchProbe: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-		_sENC_EXT_AXIS
S2	ProbeID	Probe ID 0: Probe 1 1: Probe 2	No	-	0 to 1	INT
S3	TriggerEdge	Trigger edge 0: Only rising edge 1: Only falling edge 2: Both rising edge and falling edge	No	-	0 to 2	INT
S4	TerminalSource	Probe signal source (applicable to only the bus encoder axis) 0: DI terminal 1: Encoder Z signal	Yes	0	0 to 1	INT

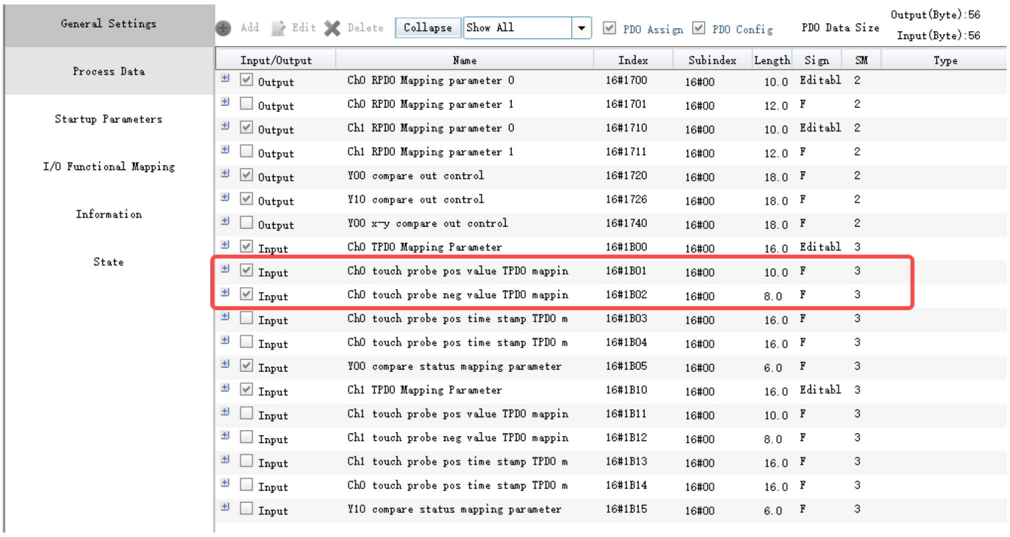
S5	TriggerMode	Trigger mode 0: Single trigger 1: Continuous trigger	Yes	0	0 to 1	INT
S6	WindowOnly	Probe window enable 0: Disabled. Probe signals are detected at all positions. 1: Enabled. Probe signals are detected only when the current position is between FirstPosition (included) and LastPosition.	Yes	OFF	ON OFF	BOOL
S7	FirstPosition	Probe window start position	Yes	0	Positive number, negative number, or 0	REAL
S8	LastPosition	Probe window end position	Yes	0	Not equal to FirstPosition	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON OFF	BOOL
D3	CommandA-borted	Abortion	Yes	OFF	ON OFF	BOOL
D4	PosPosition	Position latched on the rising edge	Yes	0	Positive number, negative number, or 0	REAL
D5	NegPosition	Position latched on the falling edge	Yes	0	Positive number, negative number, or 0	REAL
D6	Error	Error flag	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Fault code	Yes	0	*1	INT

Note

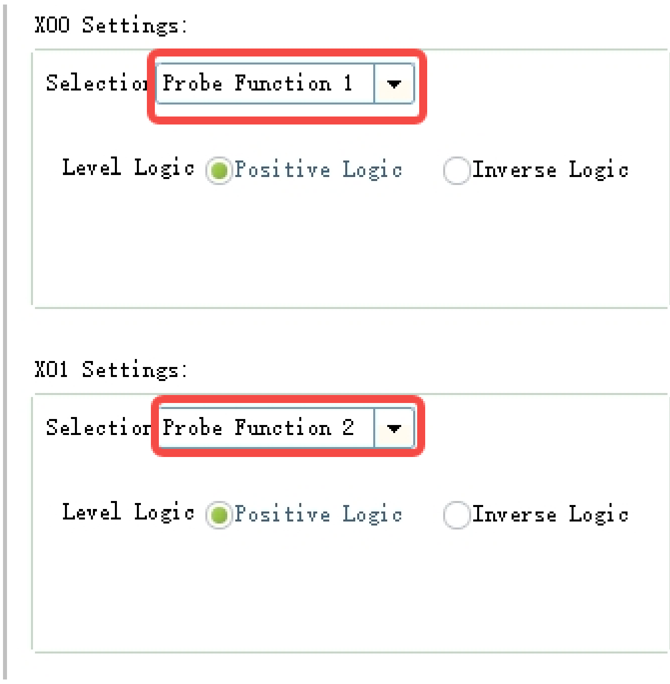
*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Function Description (Bus Encoder Axis)

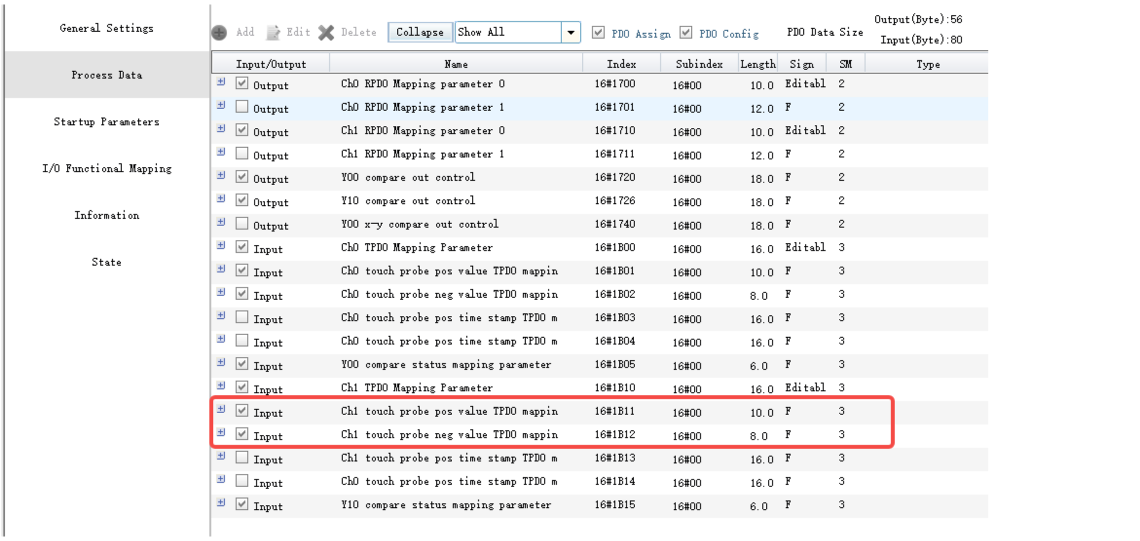
When the bus encoder axis is associated with CH0 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, the terminals X00 and X01 of the GR10-2HCE module can be used as probe terminals.



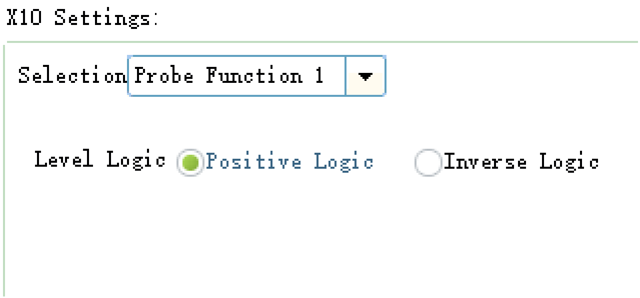
Input/Output	Name	Index	Subindex	Length	Sign	SM	Type
Output	Ch0 RPDO Mapping parameter 0	16#1700	16#00	10.0	Editabl	2	
Output	Ch0 RPDO Mapping parameter 1	16#1701	16#00	12.0	F	2	
Output	Ch1 RPDO Mapping parameter 0	16#1710	16#00	10.0	Editabl	2	
Output	Ch1 RPDO Mapping parameter 1	16#1711	16#00	12.0	F	2	
Output	Y00 compare out control	16#1720	16#00	18.0	F	2	
Output	Y10 compare out control	16#1726	16#00	18.0	F	2	
Output	Y00 x-y compare out control	16#1740	16#00	18.0	F	2	
Input	Ch0 TPDO Mapping Parameter	16#1B00	16#00	16.0	Editabl	3	
Input	Ch0 touch probe pos value TPDO mappin	16#1B01	16#00	10.0	F	3	
Input	Ch0 touch probe neg value TPDO mappin	16#1B02	16#00	8.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B03	16#00	16.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B04	16#00	16.0	F	3	
Input	Y00 compare status mapping parameter	16#1B05	16#00	6.0	F	3	
Input	Ch1 TPDO Mapping Parameter	16#1B10	16#00	16.0	Editabl	3	
Input	Ch1 touch probe pos value TPDO mappin	16#1B11	16#00	10.0	F	3	
Input	Ch1 touch probe neg value TPDO mappin	16#1B12	16#00	8.0	F	3	
Input	Ch1 touch probe pos time stamp TPDO m	16#1B13	16#00	16.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B14	16#00	16.0	F	3	
Input	Y10 compare status mapping parameter	16#1B15	16#00	6.0	F	3	

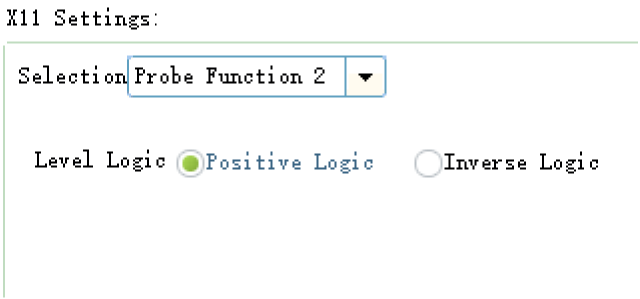


When the bus encoder axis is associated with CH1 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, the terminals X10 and X11 of the GR10-2HCE module can be used as probe terminals.



Input/Output	Name	Index	Subindex	Length	Sign	SM	Type
Output	Ch0 RPDO Mapping parameter 0	16#1700	16#00	10.0	Editabl	2	
Output	Ch0 RPDO Mapping parameter 1	16#1701	16#00	12.0	F	2	
Output	Ch1 RPDO Mapping parameter 0	16#1710	16#00	10.0	Editabl	2	
Output	Ch1 RPDO Mapping parameter 1	16#1711	16#00	12.0	F	2	
Output	Y00 compare out control	16#1720	16#00	18.0	F	2	
Output	Y10 compare out control	16#1726	16#00	18.0	F	2	
Output	Y00 x-y compare out control	16#1740	16#00	18.0	F	2	
Input	Ch0 TPDO Mapping Parameter	16#1B00	16#00	16.0	Editabl	3	
Input	Ch0 touch probe pos value TPDO mappin	16#1B01	16#00	10.0	F	3	
Input	Ch0 touch probe neg value TPDO mappin	16#1B02	16#00	8.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B03	16#00	16.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B04	16#00	16.0	F	3	
Input	Y00 compare status mapping parameter	16#1B05	16#00	6.0	F	3	
Input	Ch1 TPDO Mapping Parameter	16#1B10	16#00	16.0	Editabl	3	
Input	Ch1 touch probe pos value TPDO mappin	16#1B11	16#00	10.0	F	3	
Input	Ch1 touch probe neg value TPDO mappin	16#1B12	16#00	8.0	F	3	
Input	Ch1 touch probe pos time stamp TPDO m	16#1B13	16#00	16.0	F	3	
Input	Ch0 touch probe pos time stamp TPDO m	16#1B14	16#00	16.0	F	3	
Input	Y10 compare status mapping parameter	16#1B15	16#00	6.0	F	3	





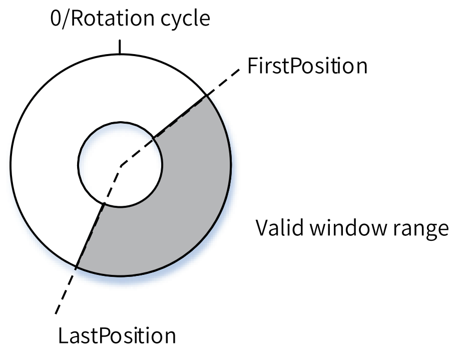
On the rising edge, the instruction latches the input parameters on the left, such as ProbeID and TriggerEdge, and other state update parameters are invalid.

When Enable is ON, the function block latches the current position of the axis when the instruction detects that the input of the probe specified by ProbeID is active and meets the probe detection conditions.

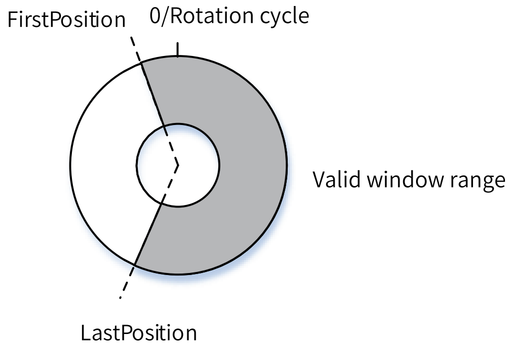
- When WindowOnly is OFF, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.
- When WindowOnly is ON, the window detection function is enabled.

In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.

In ring mode, when FirstPosition is less than LastPosition, the valid window range is as follows:



When FirstPosition is greater than LastPosition, the valid window range is as follows:



This instruction can detect the rising edge or falling edge of the probe signal separately or both the rising edge and the falling edge at the same time. When detecting only the rising edge (falling edge), the instruction writes the value detected on the rising edge (falling edge) into PosPosition (NegPosition). At this time, the Done signal is set to ON when a detection cycle is completed.

If the rising edge and falling edge are detected at the same time, after the Enable signal is active, the instruction immediately writes the position into PosPosition upon detecting the rising edge and writes the position into NegPosition upon detecting the falling edge. After that, the detection cycle is completed and the Done signal is output. There is no requirement on the input sequence of the rising edge and falling edge.

The input TerminalSource of this instruction can be used to set the terminal type to DI or the Z signal.

This instruction supports the single trigger and continuous trigger modes. If the single trigger mode is used, instruction execution ends when the Done signal output is active. If the continuous trigger mode is used, the Done output active signal is reset after one PLC scan cycle, and the instruction automatically starts to detect new probe input signals.

The following is an example.

In linear mode, the window range is 10 to 100, the EtherCAT cycle is set to 8 ms, and the velocity is 100. Then the axis moves 0.8 per EtherCAT cycle. If the current position at the moment when an EtherCAT cycle starts is 9.9, the probe signal is not detected within this EtherCAT cycle. The current position changes to 10.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 10 and 10.7 are lost. If the current position at the moment when an EtherCAT cycle starts is 99.9, the probe signal is detected within this EtherCAT cycle. The current position changes to 100.7 upon start of the next EtherCAT cycle. Therefore, the probe signals between 100 and 100.7 are responded.

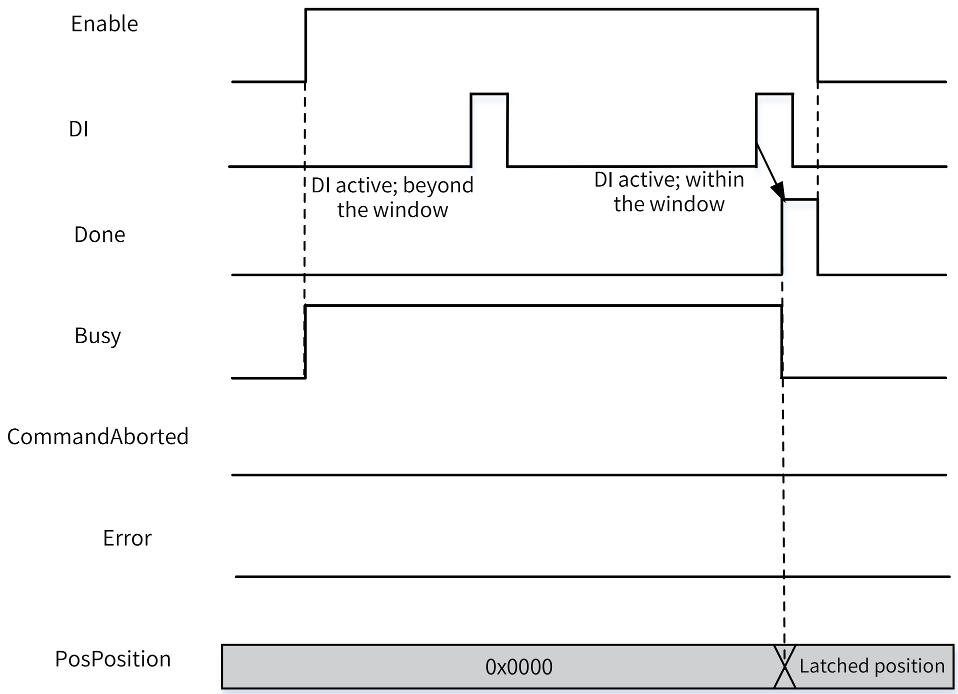
In continuous mode, if the input frequency of the probe signal is greater than the frequency of the PLC scan cycle, some probe signals are lost.

Abortion

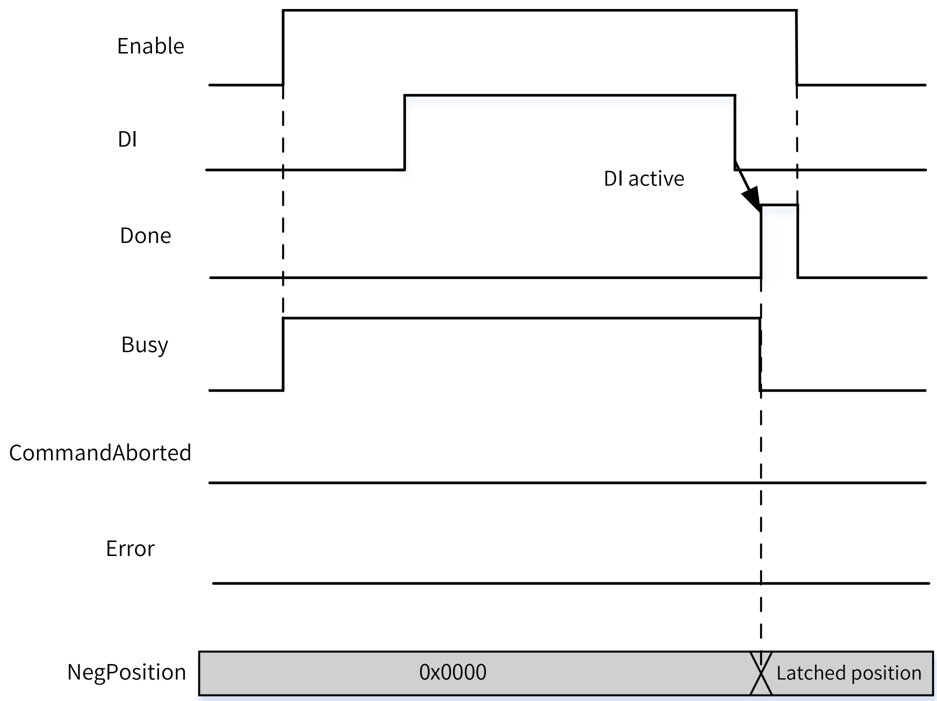
The ENC_TouchProbe instruction supports the detection probe 1 and probe 2. If two probe instructions are defined in the program and the probe IDs of the two instructions are different, the two probe instructions will work independently. If the probe IDs are the same, the probe instruction executed later will abort the previous probe instruction.

Timing Diagram

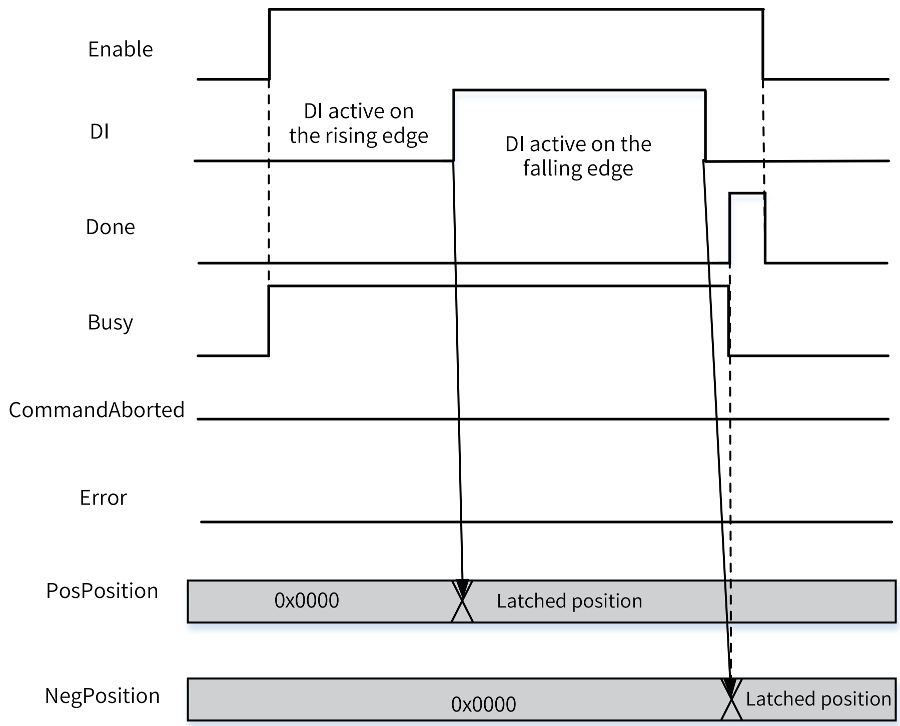
1. Probe 1, active on the rising edge, DI signal trigger source, single trigger mode, window function enabled

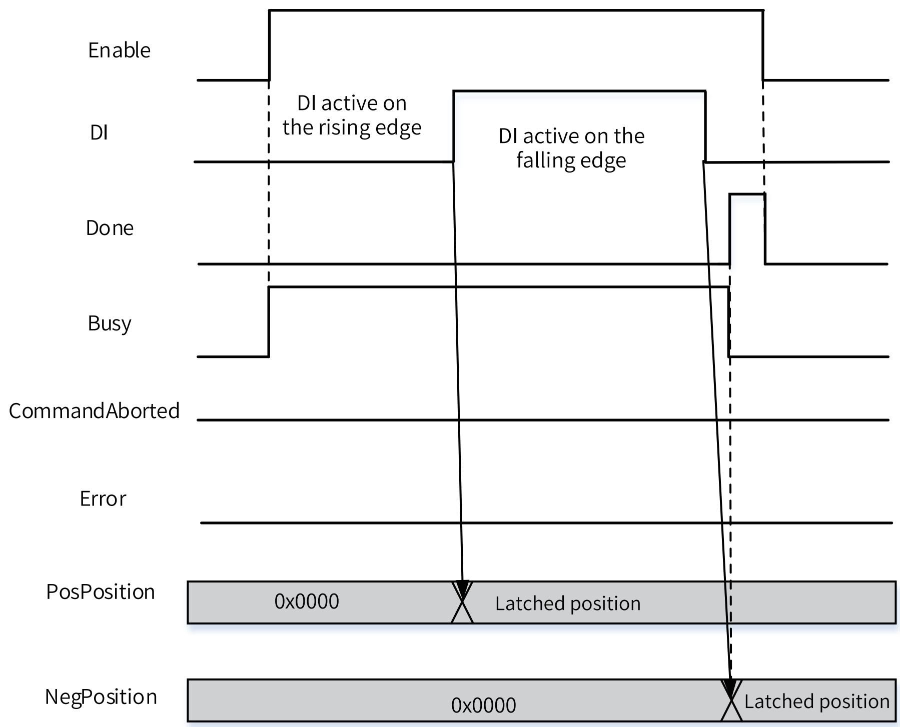


2. Probe 1, active on the falling edge, DI signal trigger source, single trigger mode, window function disabled

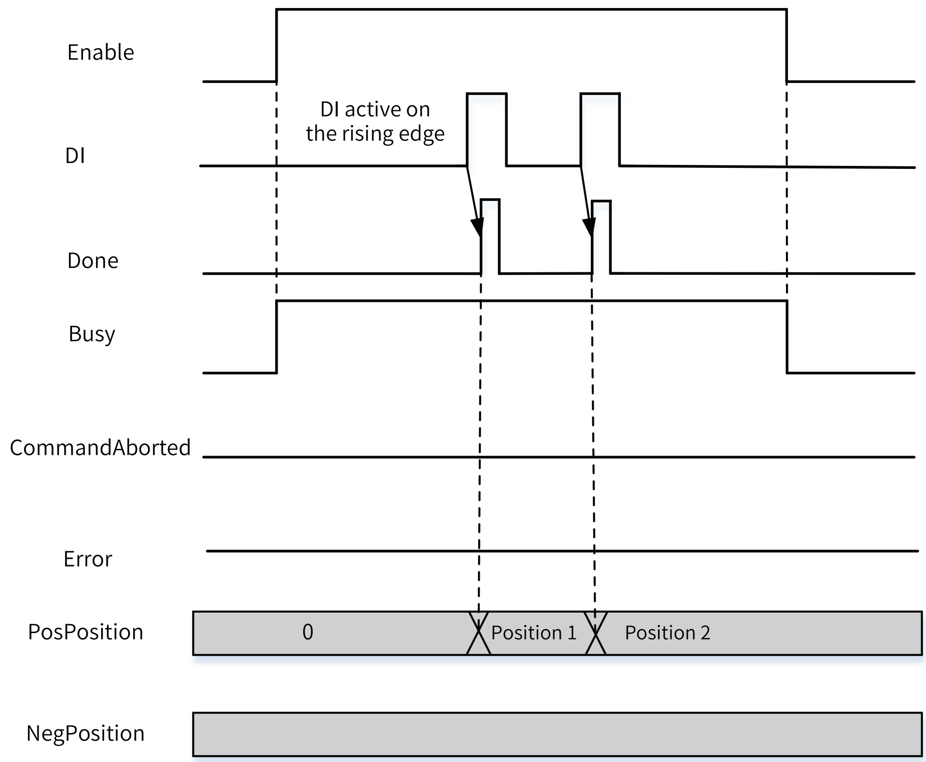


3. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, single trigger mode, window function disabled

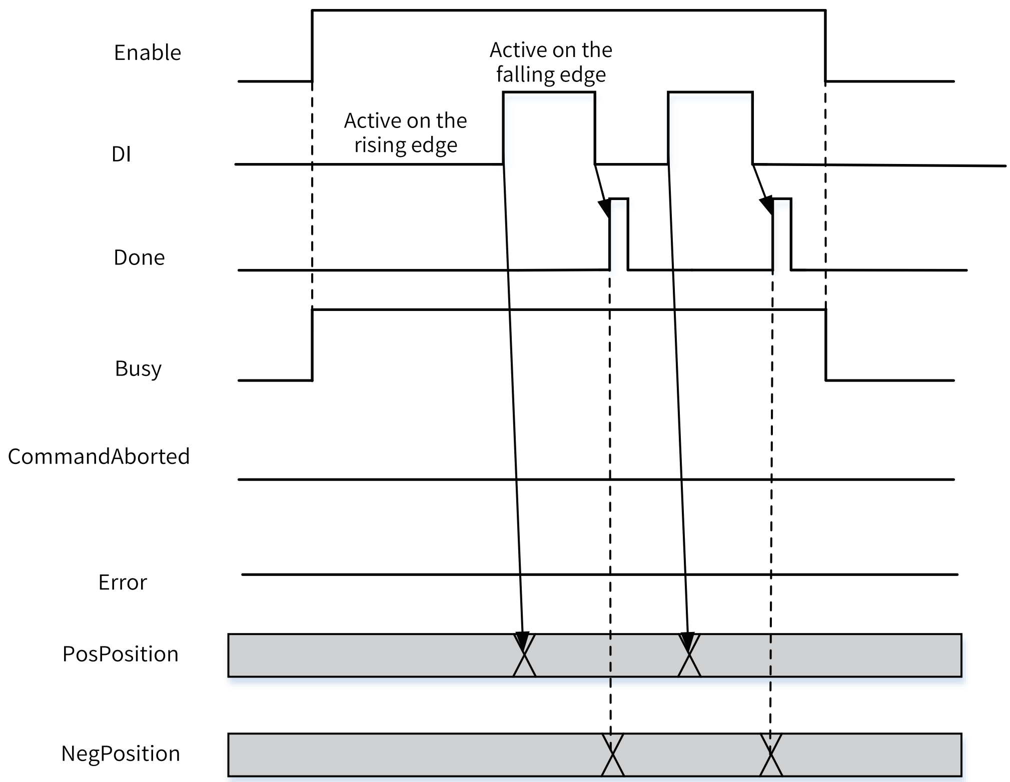




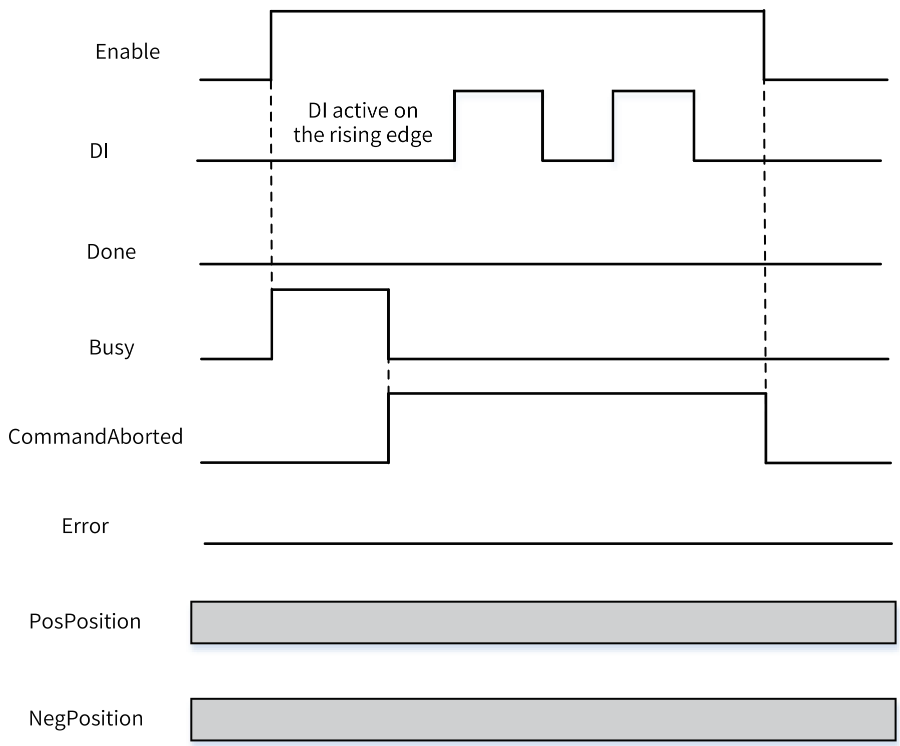
4. Probe 1, active on the rising edge, DI signal trigger source, continuous trigger mode, window function disabled



5. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, continuous trigger mode (the Done signal is active for a cycle after the DI signal is active on both the rising and falling edges), window function disabled



6. Probe 1, aborted by another probe-related instruction, window function disabled



Function Description (Local Encoder Axis)

When the bus encoder axis is associated with CH0 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, the terminals X00 and X01 of the GR10-2HCE module can be used as probe terminals.

Enable probe 1 and probe 2 on the configuration interface of the local encoder axis, and choose the appropriate probe terminals as needed.

On the rising edge, the instruction latches the input parameters on the left, such as ProbeID and TriggerEdge, and other state update parameters are invalid.

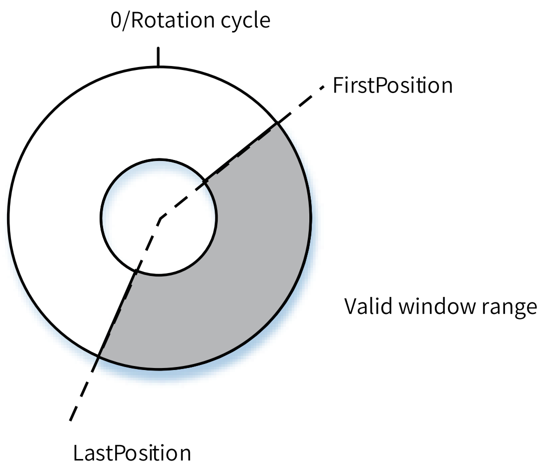
When Enable is TRUE, the function block latches the current position of the axis when the instruction detects that the input of the probe specified by ProbeID is active and meets the probe detection conditions.

When WindowOnly is FALSE, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.

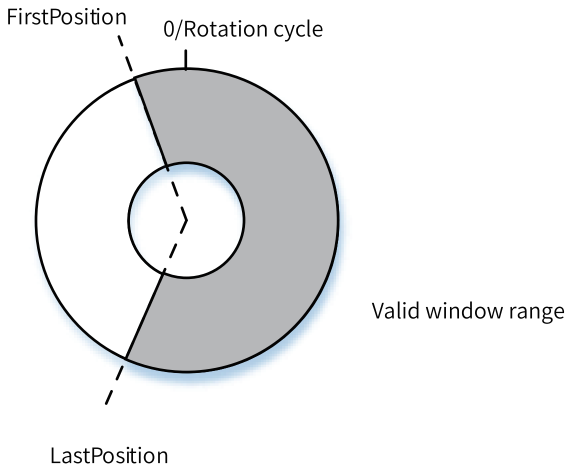
When WindowOnly is TRUE, the window detection function is enabled.

In linear mode, the instruction detects the probe signal only when the current position of the axis falls within the range specified by FirstPosition and LastPosition.

In ring mode, when FirstPosition is less than LastPosition, the valid window range is as follows:



When FirstPosition is greater than LastPosition, the valid window range is as follows:



This instruction can detect the rising edge or falling edge of the probe signal separately or both the rising edge and the falling edge at the same time. When detecting only the rising edge (falling edge), the instruction writes the value detected on the rising edge (falling edge) into PosPosition (NegPosition). At this time, the Done signal is set to ON when a detection cycle is completed.

If the rising edge and falling edge are detected at the same time, after the Enable signal is active, the instruction immediately writes the position into PosPosition upon detecting the rising edge and writes the position into NegPosition upon detecting the falling edge. After that, the detection cycle is completed and the Done signal is output. There is no requirement on the input sequence of the rising edge and falling edge.

This instruction supports the single trigger and continuous trigger modes. If the single trigger mode is used, instruction execution ends when the Done signal output is active. If the continuous trigger mode is used, the Done output active signal is reset after one PLC scan cycle, and the instruction automatically starts to detect new probe input signals.

Description

When the window function is enabled, probe signal loss or detection out-of-range may occur near the window area. The following is an example:

In linear mode, the window range is 10 to 100, the main task cycle is set to 8 ms, and the velocity is 100. Then the axis moves 0.8 per main task cycle. If the current position at the moment when a main task cycle starts is 9.9, the probe signal is not detected within this main task cycle. The current position changes to 10.7 upon start of the next main task cycle. Therefore, the probe signals between 10 and 10.7 are lost. If the current position at the moment when a main task cycle starts is 99.9, the probe signal is detected within this main task cycle. The current position changes to 100.7 upon start of the next main task cycle. Therefore, the probe signals between 100 and 100.7 are responded.

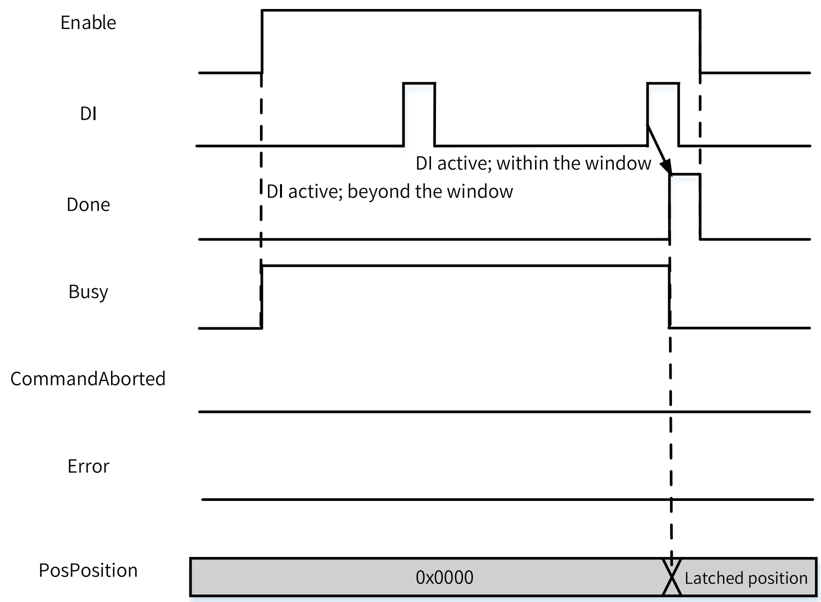
In continuous mode, if the input frequency of the probe signal is greater than the frequency of the PLC scan cycle, some probe signals are lost.

Abortion

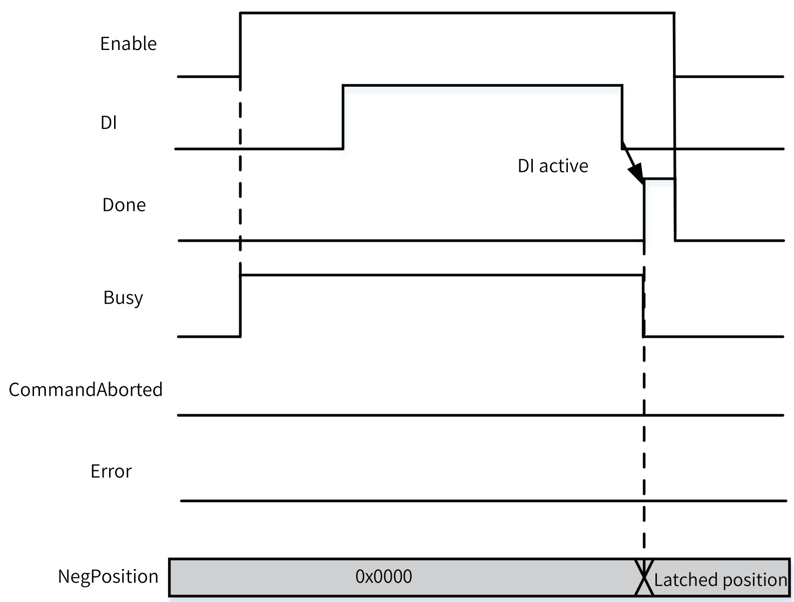
The ENC_TouchProbe instruction supports the detection probe 1 and probe 2. If two probe instructions are defined in the program and the probe IDs of the two instructions are different, the two probe instructions will work independently. If the probe IDs are the same, the probe instruction executed later will abort the previous probe instruction.

Timing Diagram

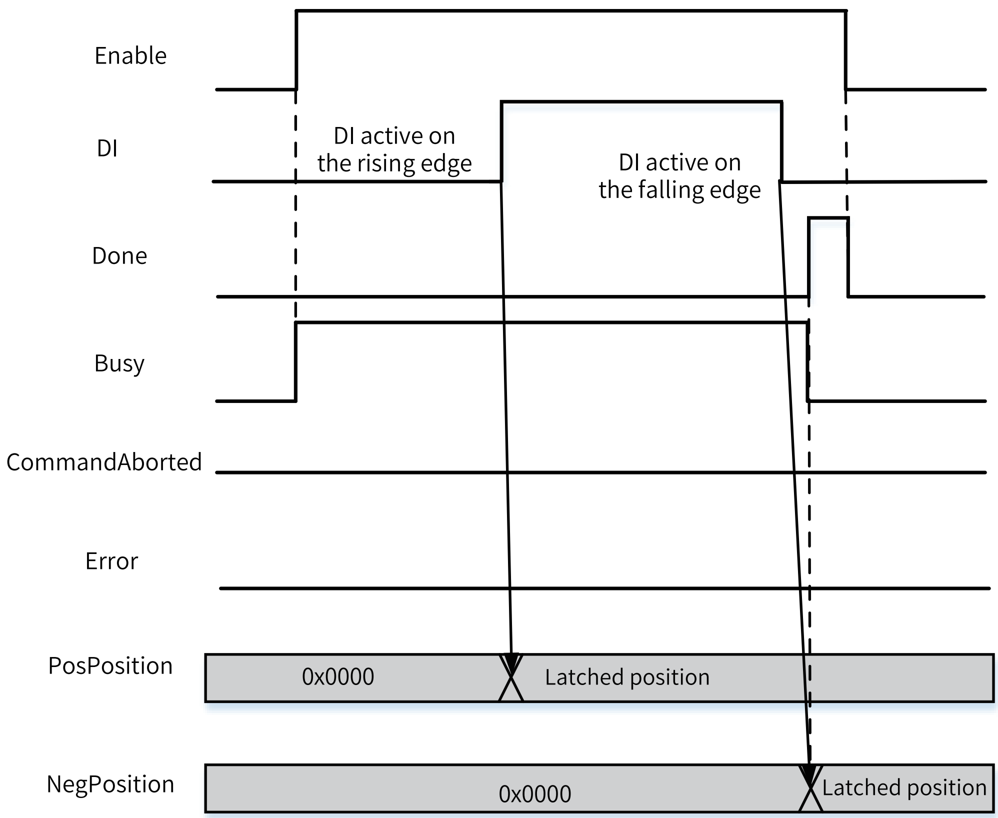
1. Probe 1, active on the rising edge, DI signal trigger source, single trigger mode, window function enabled



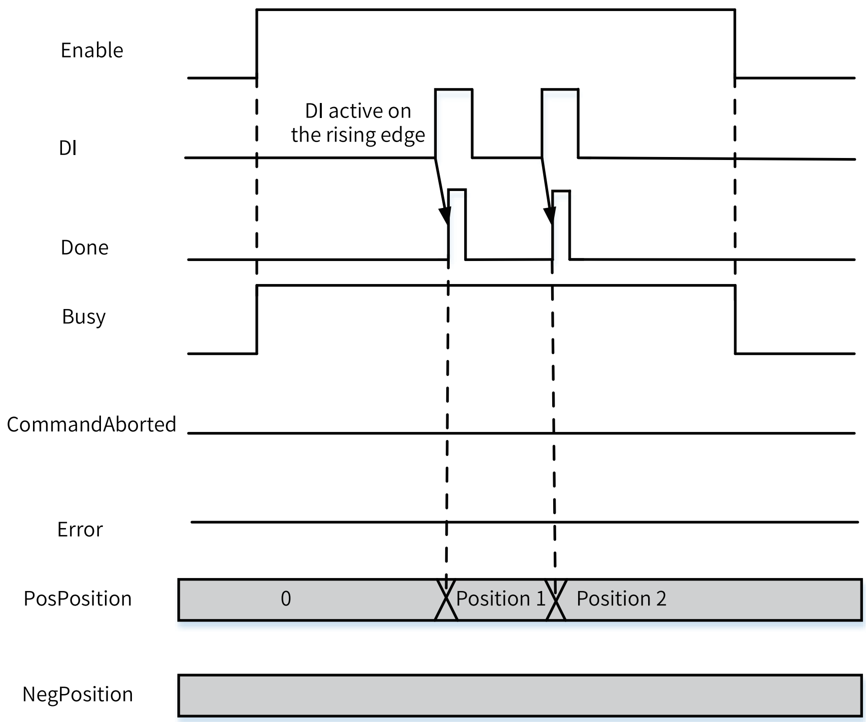
2. Probe 1, active on the falling edge, DI signal trigger source, single trigger mode, window function disabled



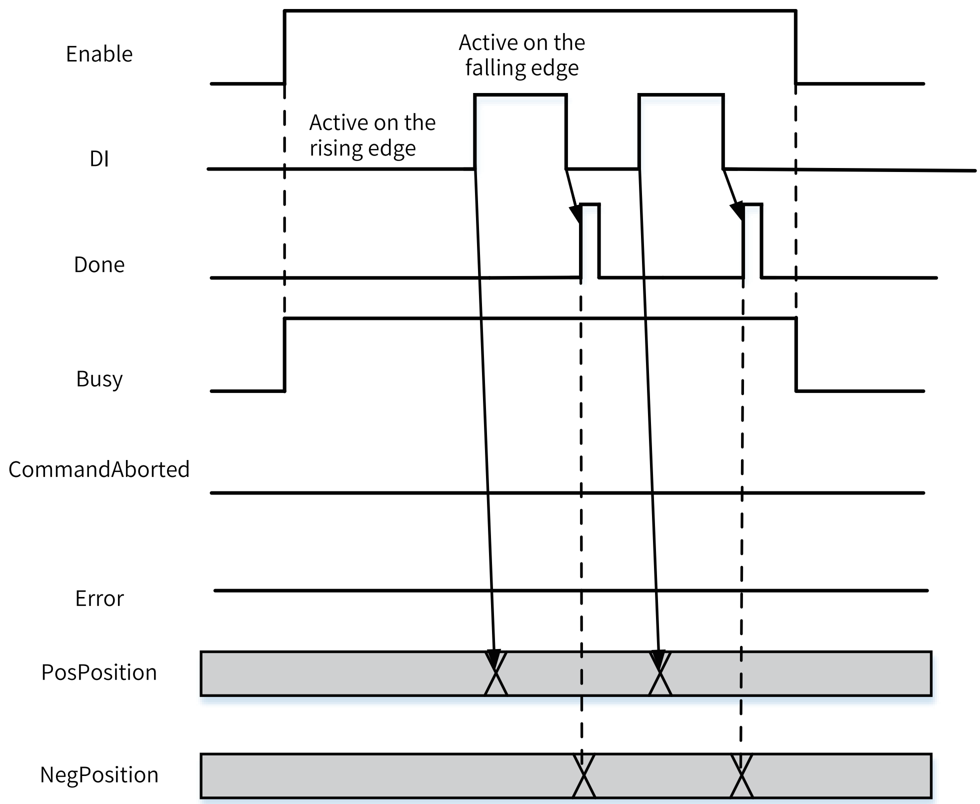
3. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, single trigger mode, window function disabled



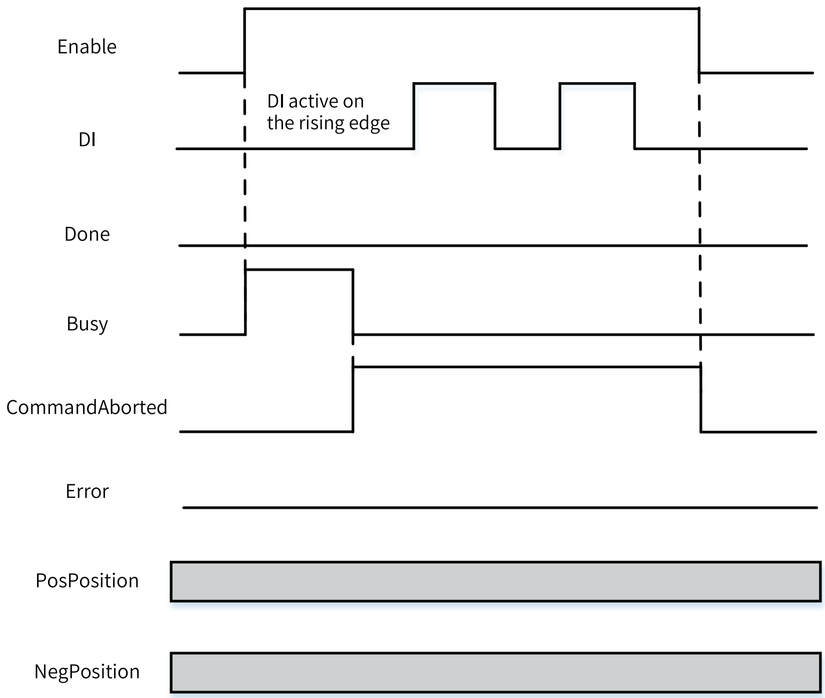
4. Probe 1, active on the rising edge, DI signal trigger source, continuous trigger mode, window function disabled



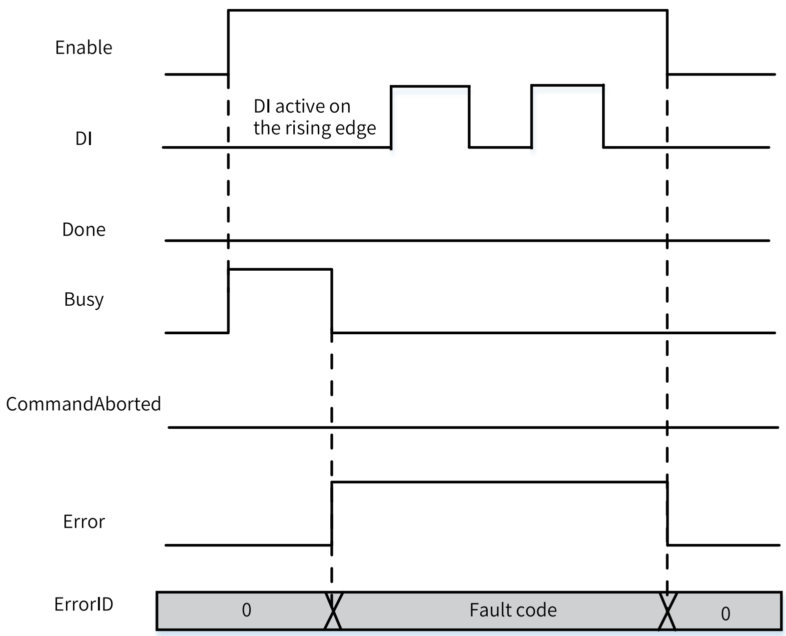
5. Probe 1, active on both the rising edge and falling edge, DI signal trigger source, continuous trigger mode (the Done signal is active for a cycle after the DI signal is active on both the rising and falling edges), window function disabled



6. Probe 1, aborted by another probe-related instruction, window function disabled



7. Probe 1, instruction error



3.12.6 ENC_ArrayCompare

ENC_ArrayCompare – Encoder one-dimensional array comparison

Graphic Block

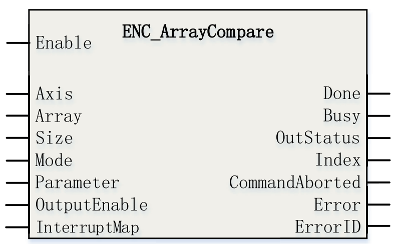
Instruction	Name	LD Expression	LiteST Expression
ENC_ArrayCompare	One-dimensional array comparison		NC_ArrayCompare(Enable := ???, Axis := ???, Array := ???, Size := ???, Mode := ???, Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus => , Index => , CommandAborted => , Error => , ErrorID =>);

Table 3–288 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_ArrayCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	Array	Comparand array	No	-		REAL[0-1000]
S3	Size	Number of comparands	No	-	1 to 1000	INT
S4	Mode	Comparison mode (bus encoder axis) 0: Reserved 1: Time control 2: Pulse control 3: Level control	Yes		0 to 3	INT
S5	Parameter	Control parameters (bus encoder axis) Time control: output active duration, in μs. Pulse control: output active pulse count, in Unit. Level control: initial level; 0 indicates low level and non-zero indicates high level.	Yes		Positive number 0 Negative number	REAL
S6	OutputEnable	Hardware output enable (local encoder axis) 0: Disabled 1: Enabled	Yes	1	0 to 1	INT
S7	InterruptMap	Interrupt ID (local encoder axis) 0: No interrupt is generated. 1: Associate with comparison interrupt 1. … 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	OutStatus	Port output state (bus encoder axis)	Yes	OFF	ON OFF	BOOL
D4	Index	Index of the next comparand	Yes	0	0 to 999	INT
D5	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL

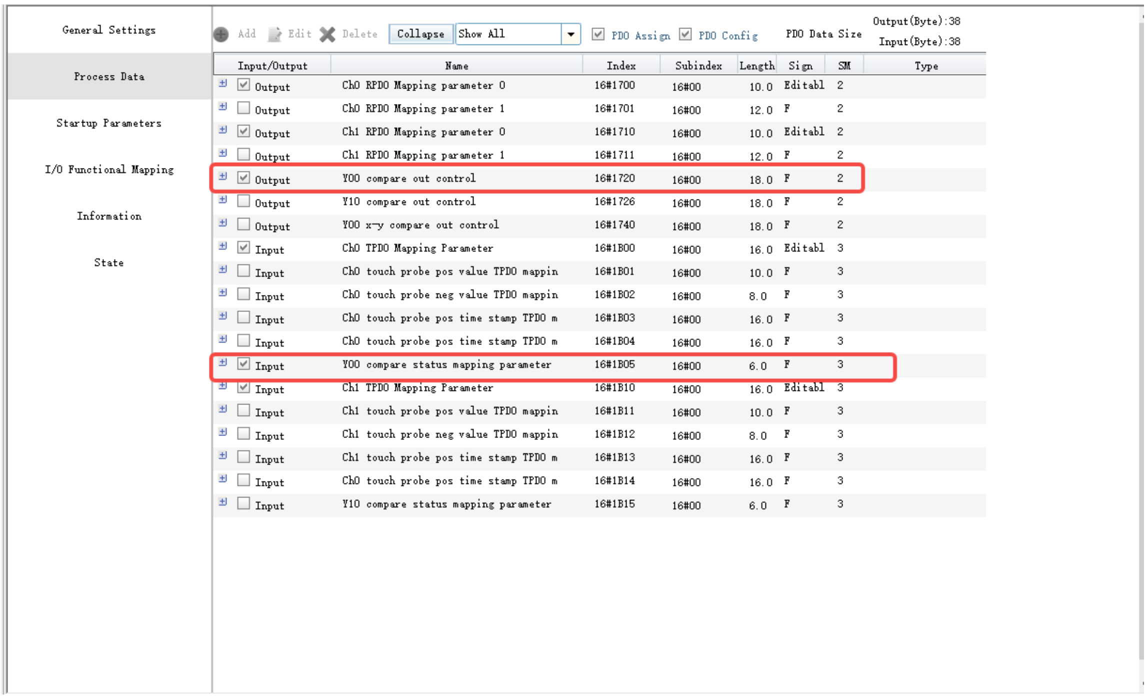
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

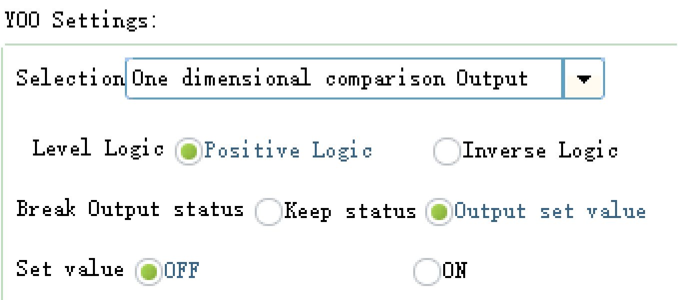
Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

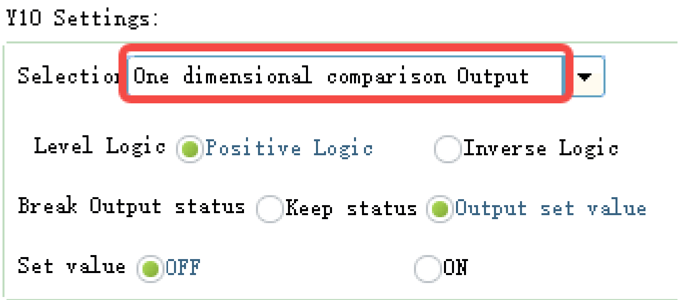
Function Description (Bus Encoder Axis)

When the bus encoder axis is associated with CH0 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y00 of the GR10-2HCE module can be used for comparison output.





When the bus encoder axis is associated with CH1 of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y10 of the GR10-2HCE module can be used for comparison output.



Setting Comparison Points

Array in the instruction specifies the comparand array, and Size indicates the actual number of points to be compared. Ensure that the value of Size is less than or equal to the number of data entries specified by Array in the PLC program. If the value of Size is greater than the number of data entries specified by Array, the instruction will not report an error, but the array index out-of-bounds error and program execution error will occur inside the PLC.

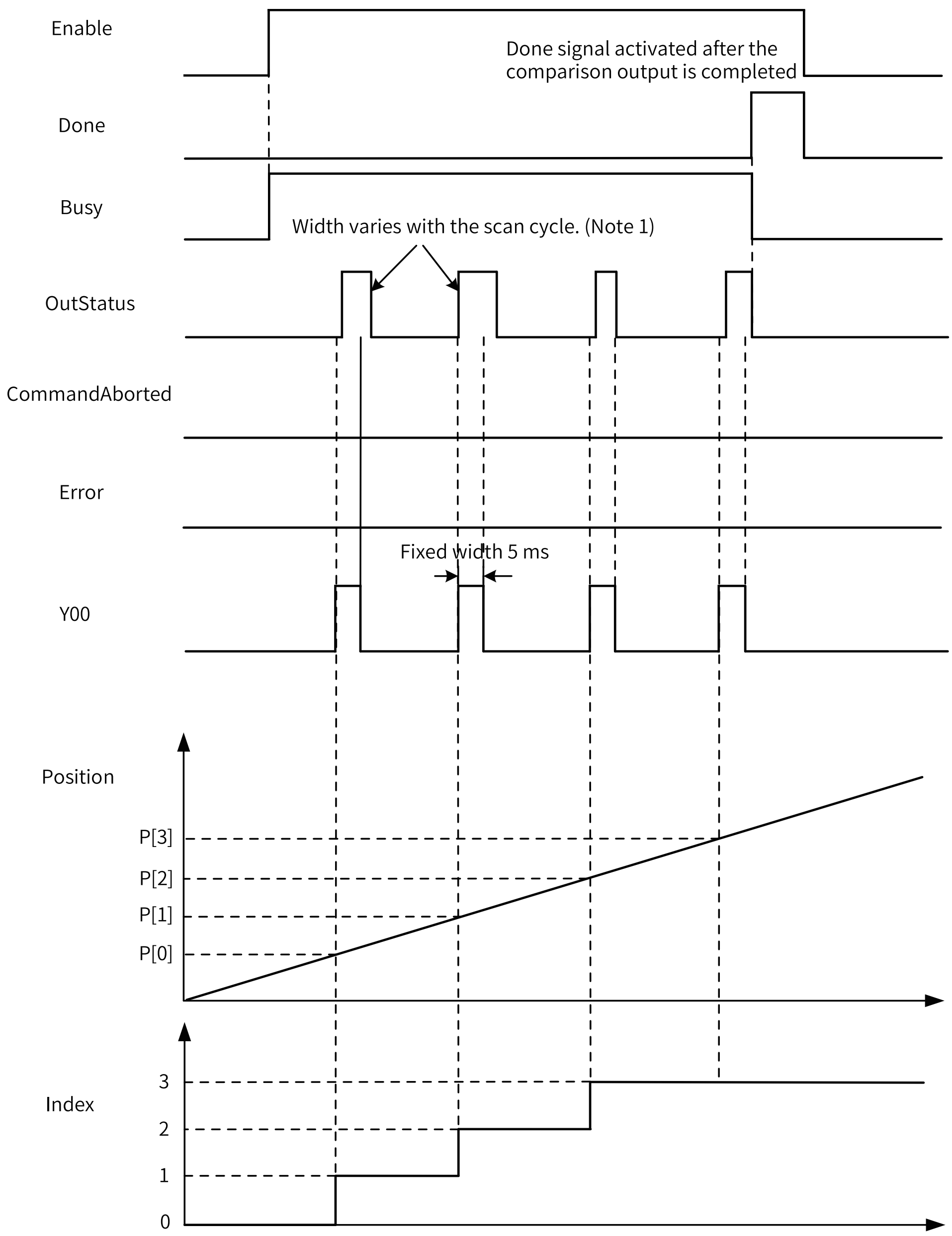
Setting the Comparison Output Mode

The basic principle of the comparison output is to set the DO terminal to high level or reverse the original level after the encoder runs to the specified position. When the output is set to high level, the time duration or the number of consecutive pulses during which the output remains high level can be specified.

1. Time mode

   When Mode is set to 1, the DO terminal outputs high level after the encoder axis reaches the comparison point. Parameter specifies the time duration during which the output remains high level, Size specifies the number of comparison points, and Array specifies the comparison point array. Among the output parameters, OutStatus indicates the output state of the comparison output terminal, and Index indicates the index of the next array coordinate point to be compared.

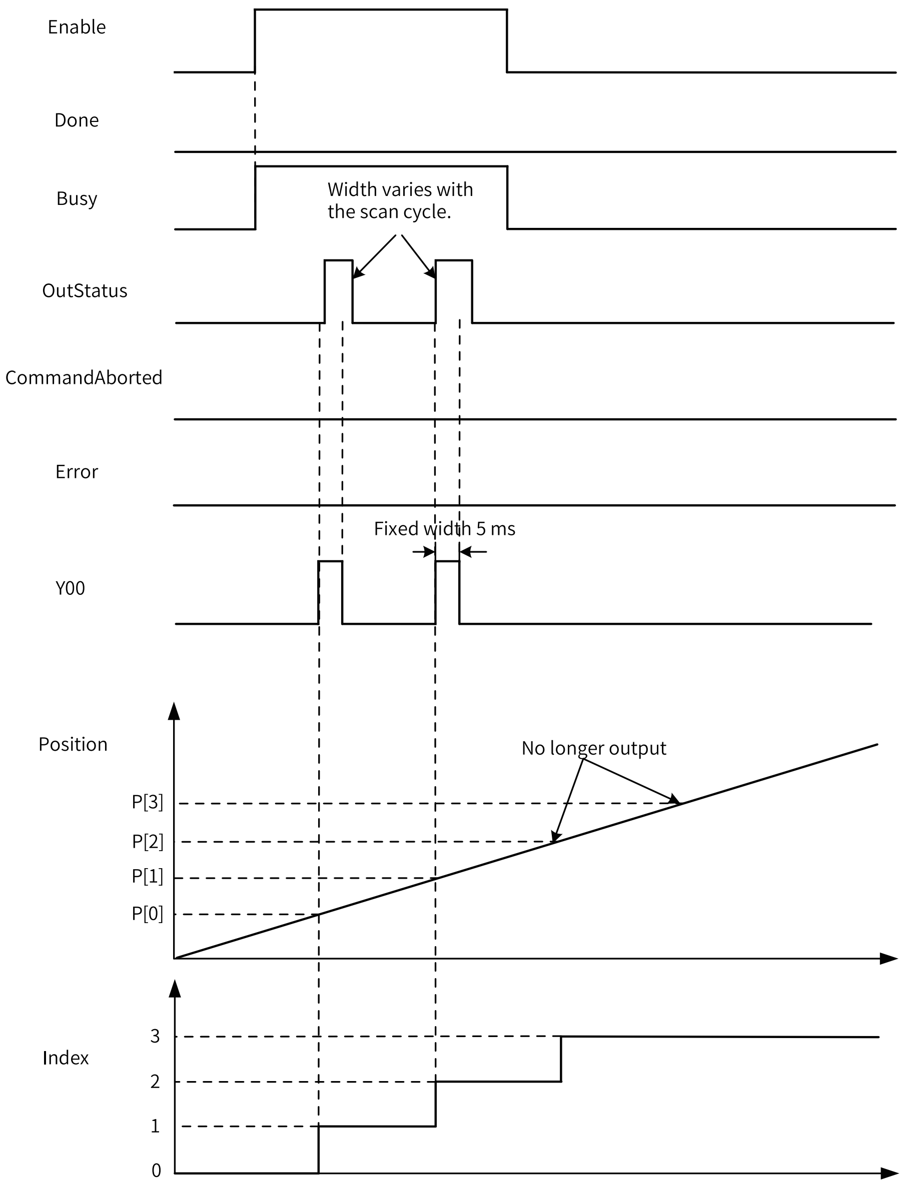
   Assume that the comparison point array is P[4], and the output time at each comparison point is 5 ms. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



Note

If the PLC scan cycle is greater than the specified output time of point Y, for example, if the PLC scan cycle is 10 ms and the output time of point Y is 100 μs, the output change of OutStatus may not be detected in the PLC task, but point Y is output normally.

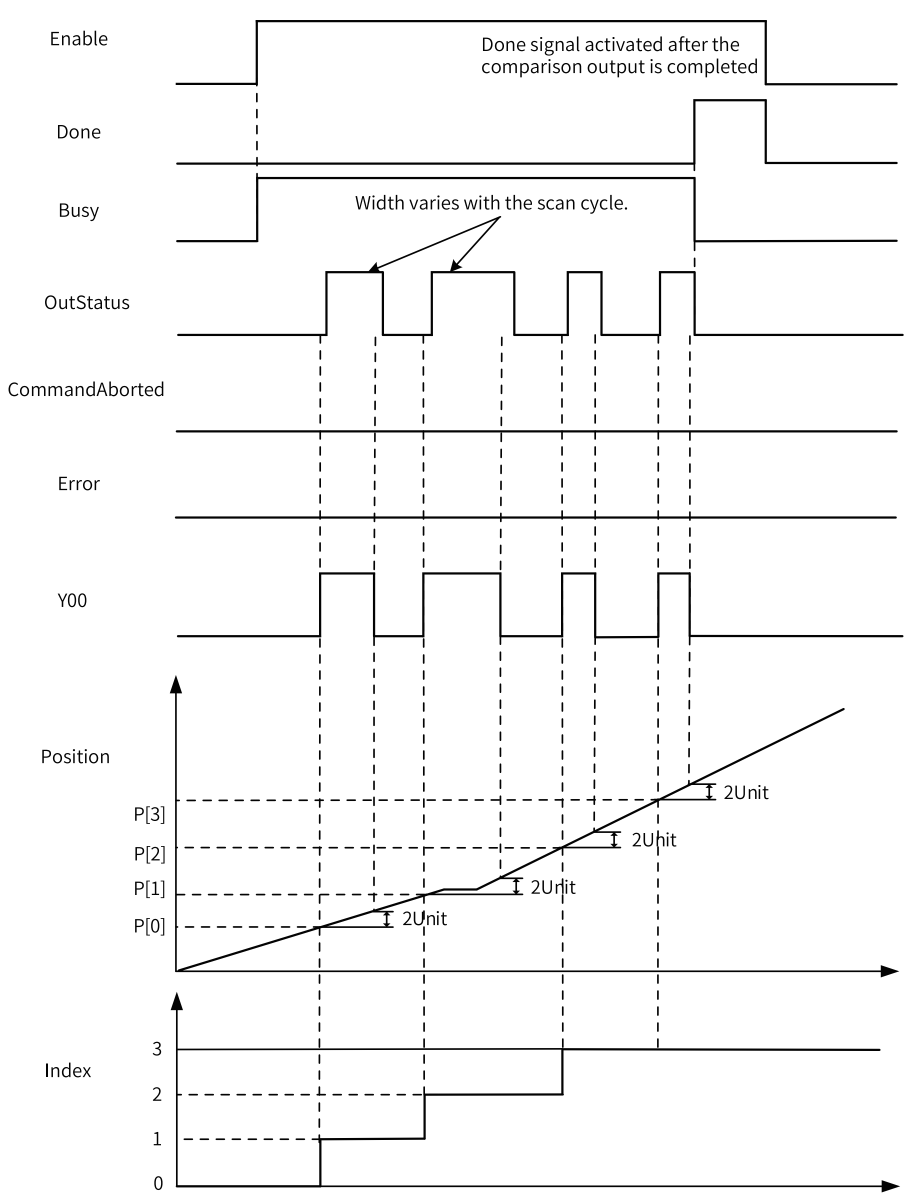
Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more.



2. Pulse mode

When Mode is set to 2, the DO terminal outputs high level after the encoder axis reaches the comparison point. Parameter specifies the number of encoder pulses (unit: Unit) during which the output remains high level, Size specifies the number of comparison points, and Array specifies the comparison point array. Among the output parameters, OutStatus indicates the output state of the comparison output terminal, and Index indicates the index of the next array coordinate point to be compared.

Assume that the comparison point array is P[4], and the output remains high level at each comparison point for 2 Units. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.

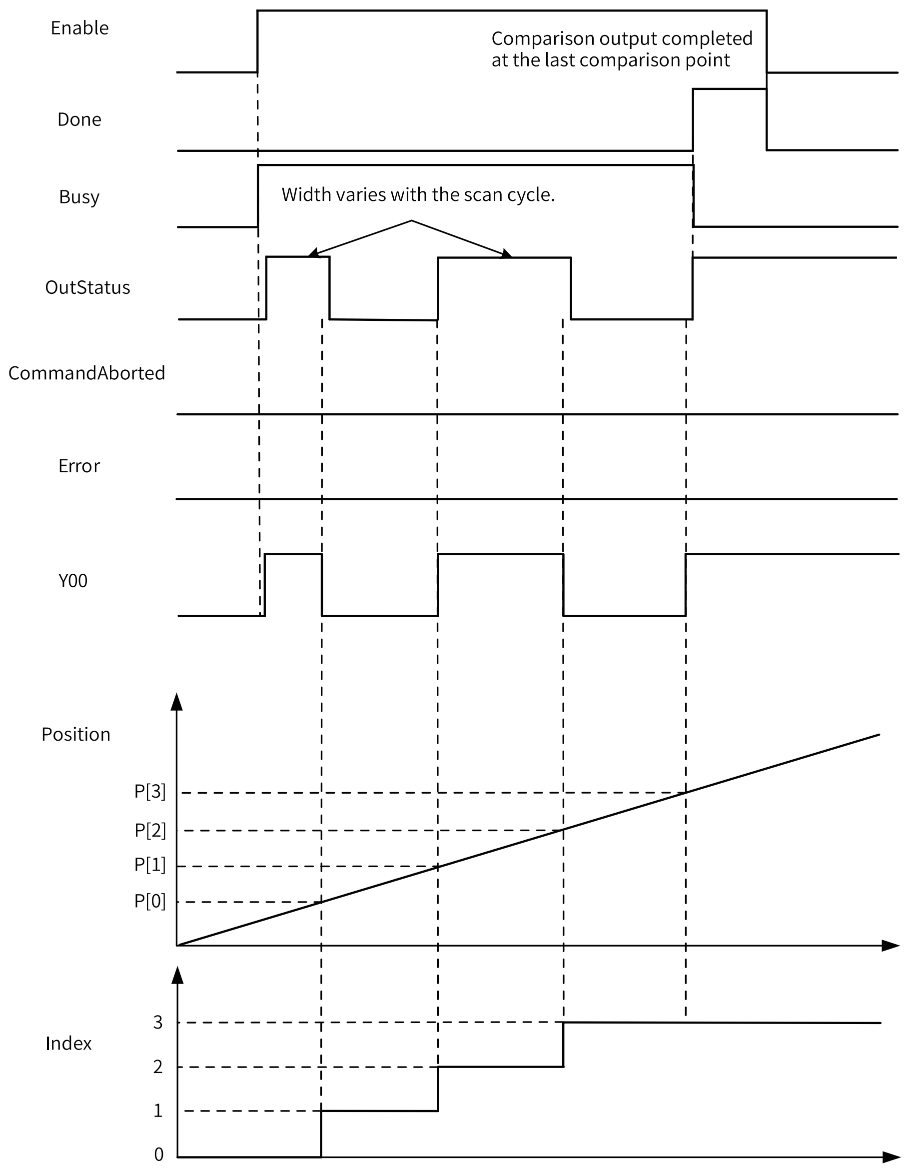


Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more, as in the case of the time mode.

3. Level mode

When Mode is set to 3, the level mode is used. Parameter specifies the initial level of the output terminal when the Enable signal of the instruction is active. If it is 0, the initial level of the Y output terminal is OFF; if it is not 0, the initial level of the Y output terminal is ON. Size specifies the number of comparison points, and Array specifies the comparison point array. Among the output parameters, OutStatus indicates the output state of the comparison output terminal, and Index indicates the index of the next array coordinate point to be compared.

Assume that the comparison point array is P[4], and Parameter is set to a non-zero value. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



As shown in the preceding figure, after the comparison output is completed, the state of the Y00 output terminal is still ON. If you want to set Y00 to OFF at this time, you need to call the ENC_ResetCompare instruction. For details, see the description of the ENC_ResetCompare instruction.

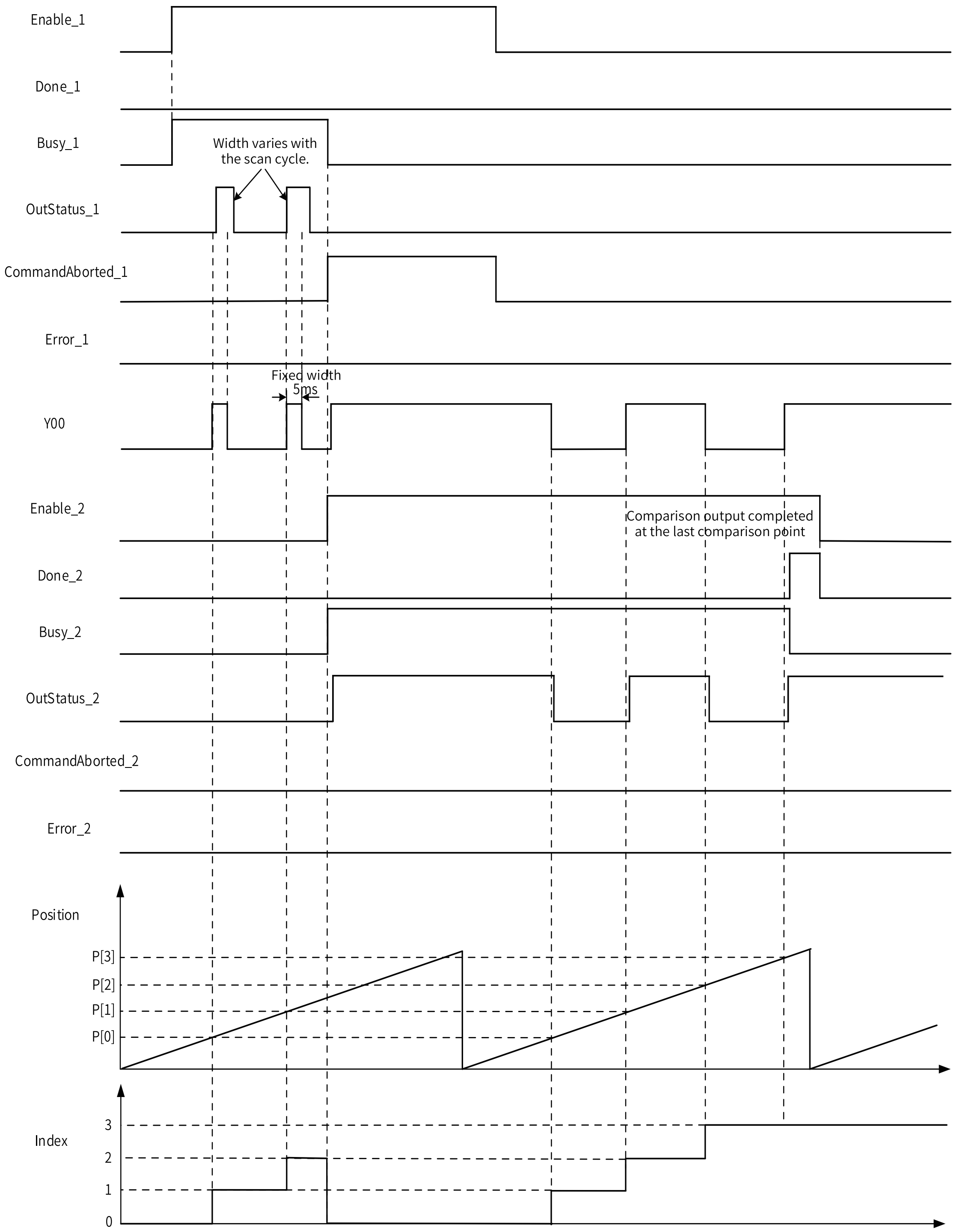
Before the comparison output is completed, if the Enable input is set to OFF, the comparison output stops and the output state of the Y00 terminal remains unchanged (for example, it remains ON). In this case, if you want to forcibly reset Y00 to OFF, you also need to call the ENC_ResetCompare instruction.

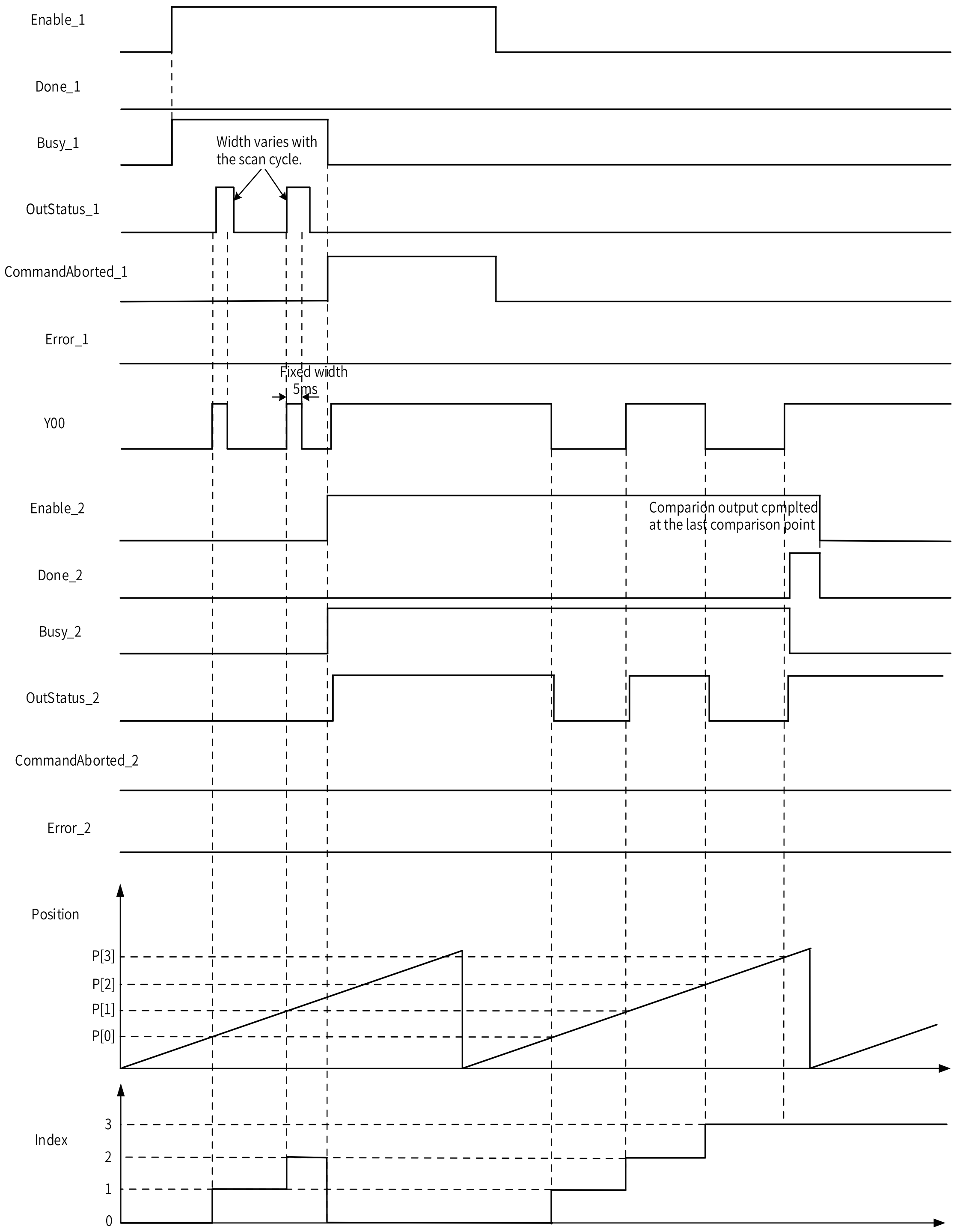
Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

The following is an example.

In the first ENC_ArrayCompare instruction, Mode is set to 1 (time mode), the comparison point array is P[4], and the output time at each comparison point is 5 ms. The instruction starts to run after the Enable input becomes active. After the first instruction runs for a period of time, the second ENC_ ArrayCompare instruction is triggered. In the second instruction, Mode is set to 3 (level mode), the comparison point array is P[4], and Parameter is set to a non-zero value. The timing diagram is as follows.





Function Description (Local Pulse Axis)

Enable comparison output on the comparison output configuration interface of the local pulse axis, select an output terminal, and select the pulse output unit (time or unit).

Setting Comparison Points

Array in the instruction specifies the comparand array, and Size indicates the actual number of points to be compared. Ensure that the value of Size is less than or equal to the number of data entries specified by Array in the PLC program. If the value of Size is greater than the number of data entries specified by Array, the instruction will not report an error, but the array index out-of-bounds error and program execution error will occur inside the PLC.

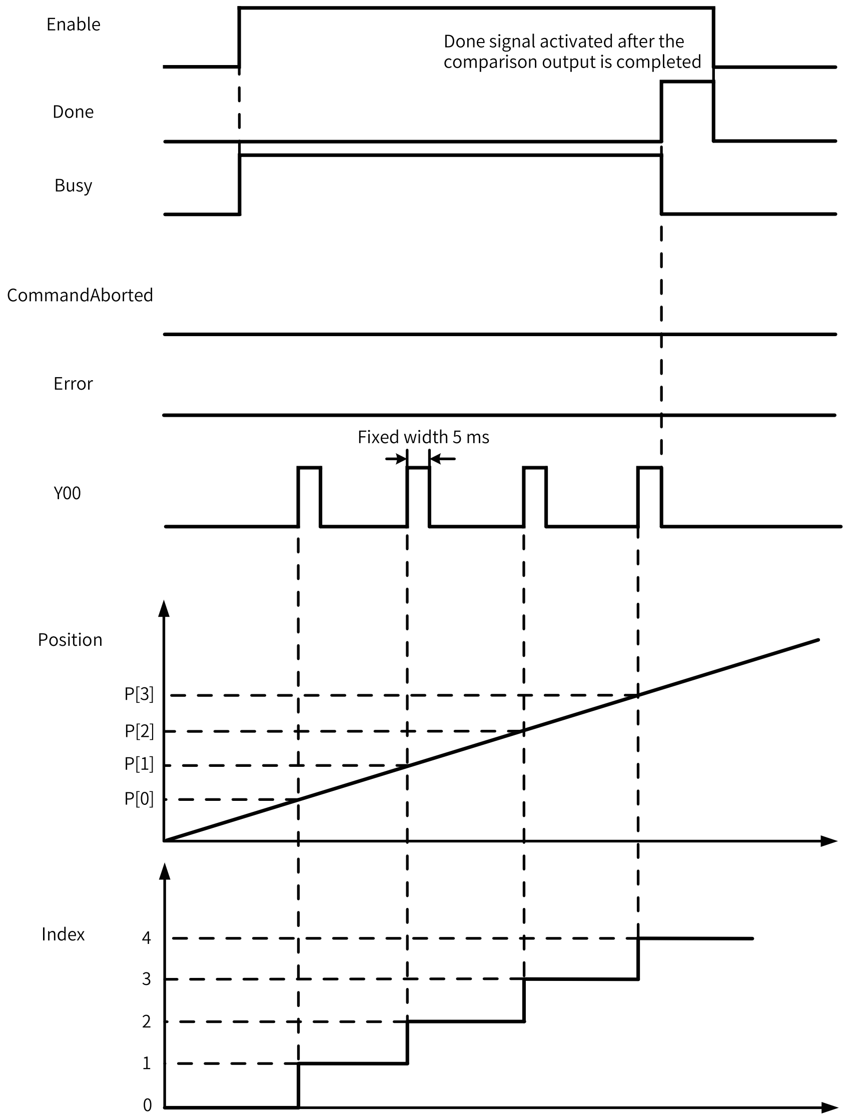
Setting the Comparison Output Mode

The basic principle of the comparison output is to set the DO terminal to high level after the encoder runs to the specified position. When the output is set to high level, the time duration or the number of consecutive pulses during which the output remains high level can be specified.

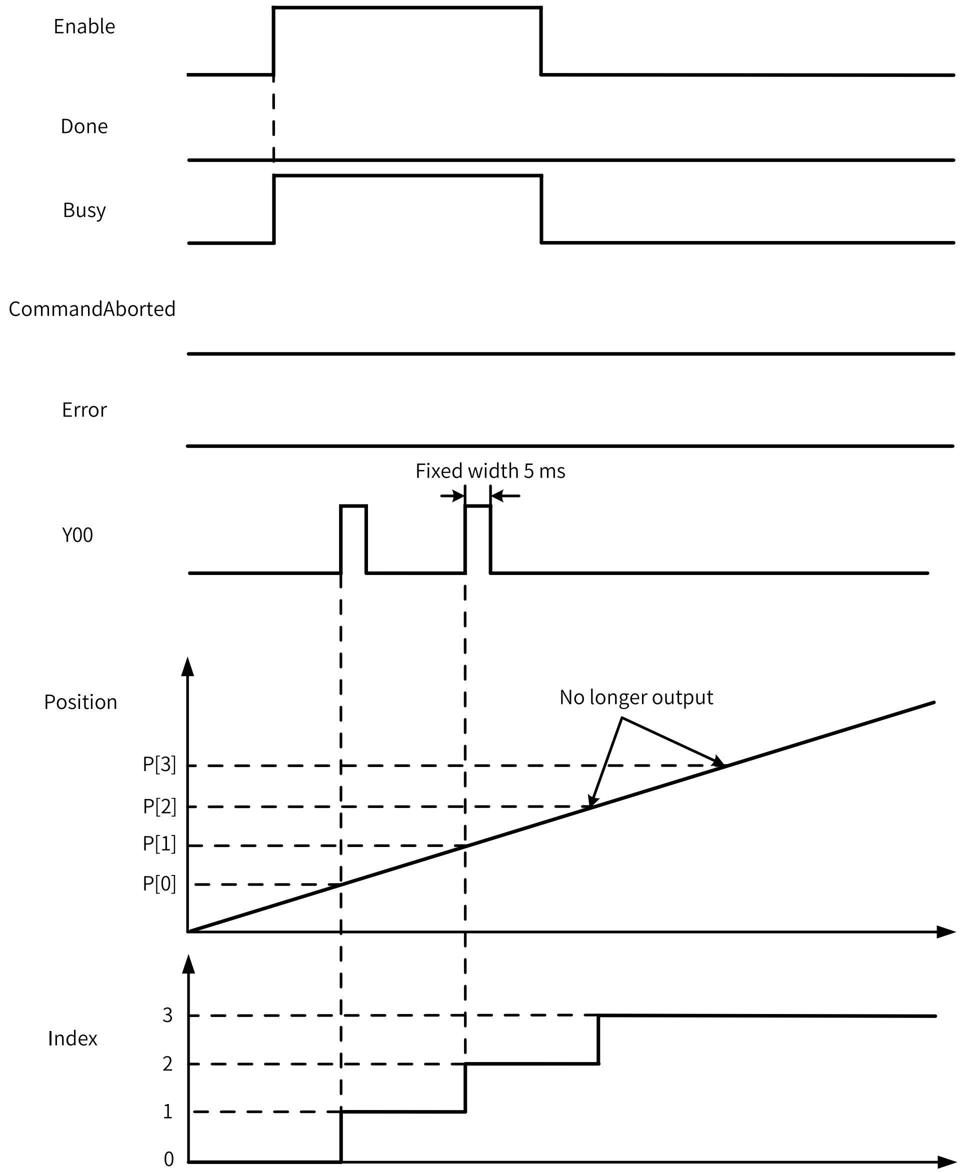
1. Time mode

   In time mode, the DO terminal outputs high level after the encoder axis reaches the comparison point. The time duration during which the output remains high level is configured on the background configuration interface. Size specifies the number of comparison points, and Array specifies the comparison point array. Index indicates the next array coordinate point to be compared.

   Assume that the comparison point array is P[4], and the output time at each comparison point is 5 ms. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



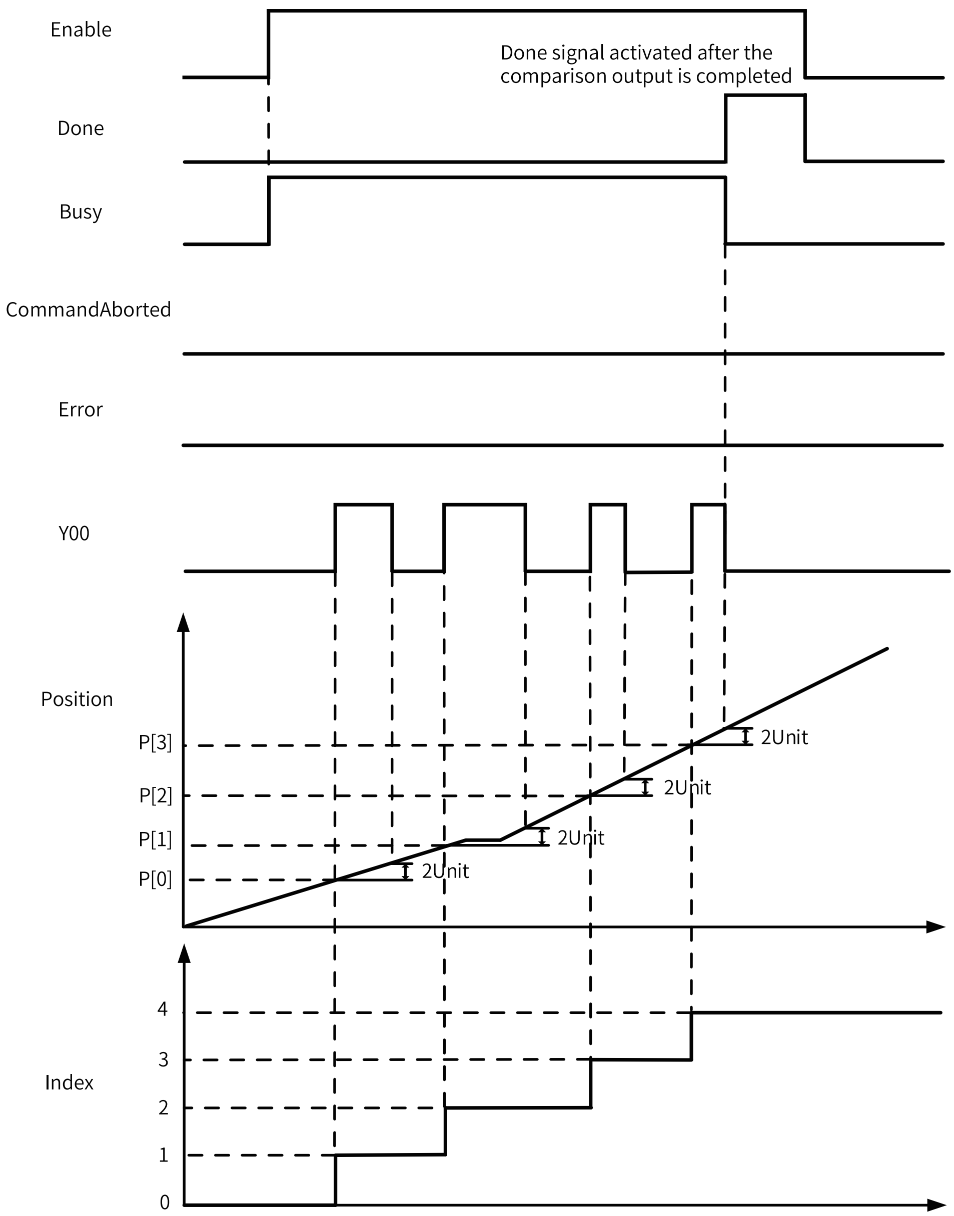
Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more.



2. Pulse mode

When Unit is selected as the unit, the output remains high level for a specified number of encoder pulses. Size specifies the number of comparison points, and Array specifies the comparison point array. Index indicates the next array coordinate point to be compared.

Assume that the comparison point array is P[4], and the output remains high level at each comparison point for 2 Units. The instruction starts to run after the Enable input becomes active. The timing diagram is as follows.



Before the comparison output is completed, if the Enable input is set to OFF, the subsequent comparison points are not compared any more, as in the case of the time mode.

Other Parameters

When OutputEnable is set to 1, the configured comparison output terminal generates comparison output signals. If OutputEnable is set to 0, no comparison output signals are generated.

InterruptMap is used to associate the comparison interrupt subprogram. When it is set to 0, no interrupt subprogram is associated. When it is set to 1 to 16, the specified interrupt subprogram is associated.

3.12.7 ENC_StepCompare

ENC_StepCompare – Encoder one-dimensional step comparison

Graphic Block

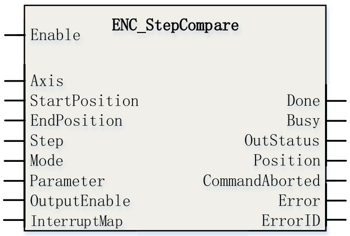
Instruction	Name	LD Expression	LiteST Expression
ENC_ StepCompare	Encoder axis step comparison		ENC_StepCompare(Enable := ???, Axis := ???, StartPosition := ???, EndPosition := ???, Step := ???, Mode := ???, Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus => , Position => , CommandAborted => , Error => , ErrorID =>);

Table 3–289 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_StepCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_EXT_AXIS
S2	StartPosition	Comparison start position	No	-	-	REAL
S3	EndPosition	Comparison end position	No	-	-	REAL
S4	Step	Step	No	-	-	REAL
S5	Mode	Comparison mode 0: Reserved 1: Time control 2: Pulse control 3: Level control	Yes	-	0 to 3	INT

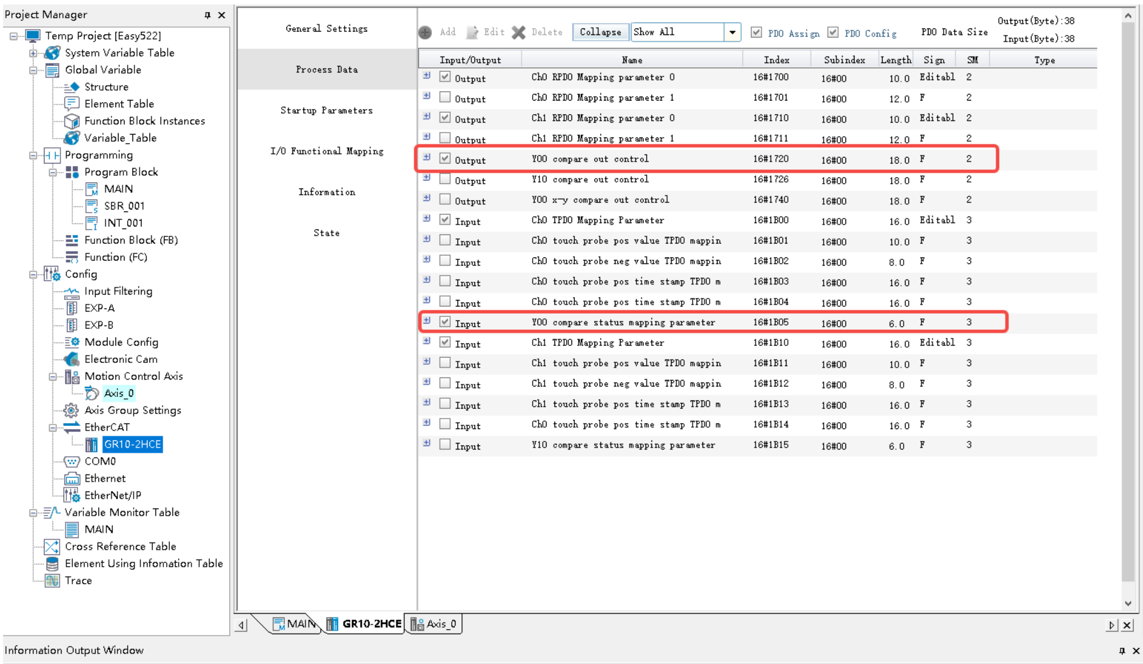
S6	Parameter	Control parameters Time control: output active duration, in μs. Pulse control: output active pulse count, in Unit. Level control: initial level; 0 indicates low level and non-zero indicates high level.	Yes	0	Positive number 0 Negative number	REAL
S7	OutputEnable	Hardware output enable (local encoder axis) 0: Disabled 1: Enabled	Yes	0	0 to 1	INT
S8	InterruptMap	Interrupt ID (local encoder axis) 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	OutStatus	Port output state (bus encoder axis)	Yes	OFF	ON OFF	BOOL
D4	Position	Next comparand	Yes	0		REAL
D5	CommandAbort-ed	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

Note

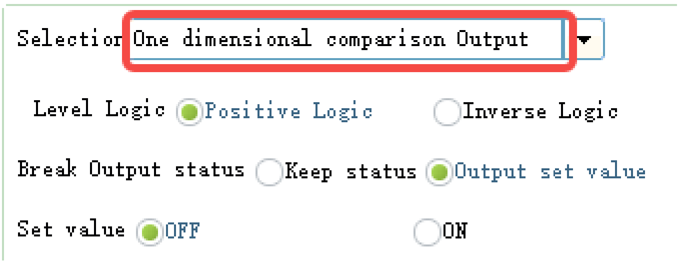
*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes.*

Function Description (Bus Encoder Axis)

When the bus encoder axis is associated with the first channel of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y00 of the GR10-2HCE module can be used for comparison output.

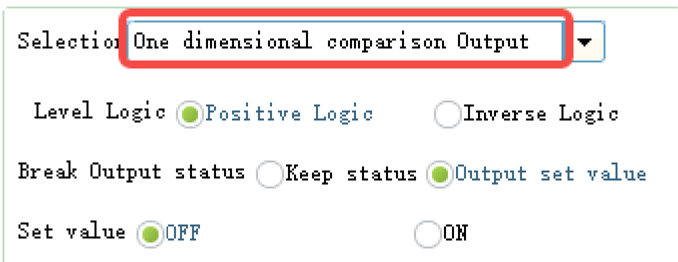


Y00 Settings:



When the bus encoder axis is associated with the second channel of the GR10-2HCE module, the PDO options as shown in the following figure need to be selected on the process data page of the GR10-2HCE module. At this time, Y10 of the GR10-2HCE module can be used for comparison output.

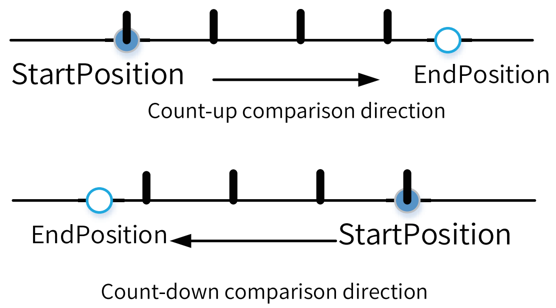
Y10 Settings:



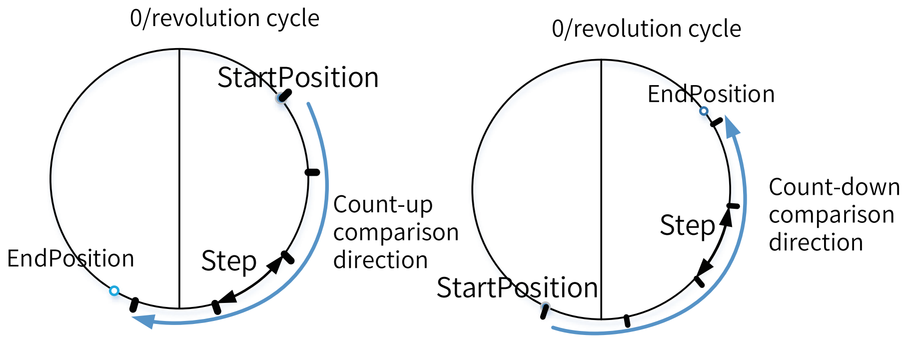
Setting Comparison Points

In this instruction, StartPosition specifies the start position of the comparison points, and EndPosition specifies the end position of the comparison points.

- In linear mode, when StartPosition is less than EndPosttion, set Step to a positive value, which indicates the count-up comparison mode; when StartPostion is greater than EndPosition, set Step to a negative value, which indicates the count-down comparison mode.



- In ring mode, when StartPosition is less than EndPosttion, set Step to a positive value, which indicates the up-counting comparison mode; when StartPostion is greater than EndPosition, set Step to a negative value, which indicates the down-counting comparison mode.



Note

The first comparison point must be the point specified by StartPosition, but the last comparison point is not necessarily the point specified by EndPosition. For example, assume that the start point is 10, the end point is 25, and the step is 10. In this case, the comparison output signal is generated only at position 10 and position 20.

Comparison Modes

The step comparison instruction supports the time control, pulse control, and level control modes, and the mode is specified by the parameter Mode of the instruction. For details about the three modes, see the description of the ENC_ArrayCompare instruction.

Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

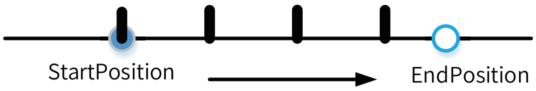
Function Description (Local Encoder Axis)

Enable comparison output on the comparison output configuration interface of the local pulse axis, select an output terminal, and select the pulse output unit (time or unit).

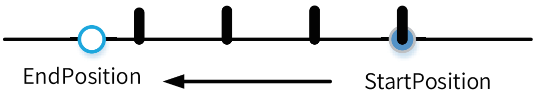
Setting Comparison Points

In this instruction, StartPosition specifies the start position of the comparison points, and EndPosition specifies the end position of the comparison points.

In linear mode, when StartPosition is less than EndPosttion, set Step to a positive value, which indicates the up-counting comparison mode; when StartPostion is greater than EndPosition, set Step to a negative value, which indicates the down-counting comparison mode.

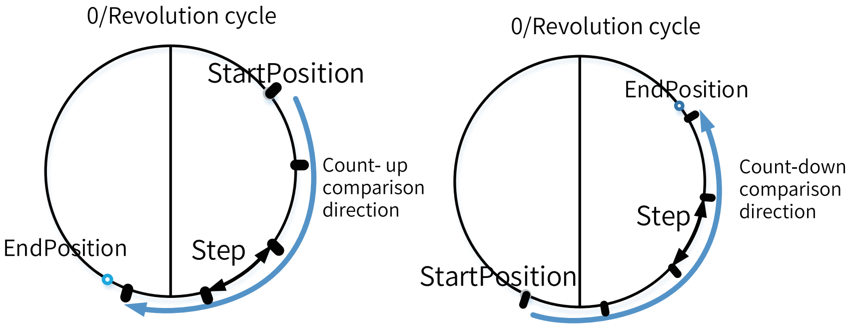


Counting up comparison direction



Counting down comparison direction

In ring mode, when StartPosition is less than EndPosttion, set Step to a positive value, which indicates the up-counting comparison mode; when StartPostion is greater than EndPosition, set Step to a negative value, which indicates the down-counting comparison mode.



Note

The first comparison point must be the point specified by StartPosition, but the last comparison point is not necessarily the point specified by EndPosition. For example, assume that the start point is 10, the end point is 25, and the step is 10. In this case, the comparison output signal is generated only at position 10 and position 20.

Comparison Modes

The step comparison instruction supports the time control and pulse control modes, which is configured on the background configuration interface. For details about the two modes, see the description of the ENC_StepCompare instruction.

Other Parameters

When OutputEnable is set to 1, the configured comparison output terminal generates comparison output signals. If OutputEnable is set to 0, no comparison output signals are generated.

InterruptMap is used to associate the comparison interrupt subprogram. When it is set to 0, no interrupt subprogram is associated. When it is set to 1 to 16, the specified interrupt subprogram is associated.

Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

3.12.8 ENC_Compare

ENC_Compare – Single-point comparison output

Graphic Block

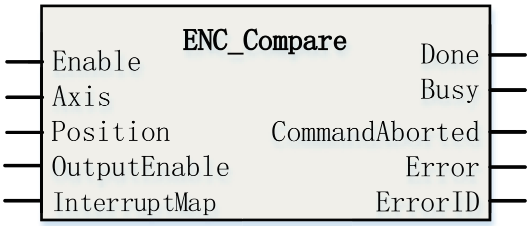


Table 3–290 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_Compare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Encoder axis	No	-	-	_sENC_ EXT_AXIS
S2	Position	Comparison position	No	-	-	REAL
S3	OutputEnable	Hardware output enable (local encoder axis) 0: Disabled 1: Enabled	Yes	0	0 to 1	INT16
S4	InterruptMap	Interrupt ID (local encoder axis) 0: No interrupt is generated. 1: Associate with comparison interrupt 1. … 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT16
D1	Done	Completion flag	Yes	FALSE	TRUE FALSE	BOOL
D2	Busy	Executing	Yes	FALSE	TRUE FALSE	BOOL
D3	CommandA-borted	Abortion of execution	Yes	FALSE	TRUE FALSE	BOOL
D4	Error	Error	Yes	FALSE	TRUE FALSE	BOOL
D5	ErrorID	Error code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

Enable comparison output on the comparison output configuration interface of the local pulse axis, select an output terminal, and select the pulse output unit (time or unit).

When OutputEnable is set to 1, the configured comparison output terminal generates comparison output signals. If OutputEnable is set to 0, no comparison output signals are generated.

InterruptMap is used to associate the comparison interrupt subprogram. When it is set to 0, no interrupt subprogram is associated. When it is set to 1 to 16, the specified interrupt subprogram is associated.

3.12.9 ENC_GroupArrayCompare

ENC_GroupArrayCompare – Encoder two-dimensional array comparison

Graphic Block

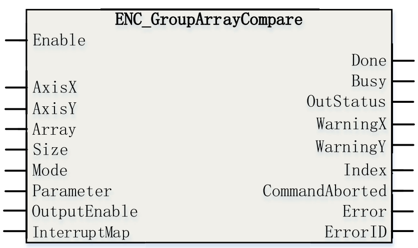
Instruction	Name	LD Expression	LiteST Expression
ENC_ GroupArray- Compare	Encoder axis array compari- son (bus encoder axis)		ENC_GroupArrayCompare(Enable := ???, AxisX := ???, AxisY := ???, Array := ???, Size := ???, Mode := ???, Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus => , WarningX => , WarningY => , Index => , CommandAborted => , Error => , ErrorID =>);

Table 3–291 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_GroupArrayCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	AxisX	Encoder axis (x-axis)	No	-	-	_sENC_EXT_AXIS
S2	AxisY	Encoder axis (y-axis)	No			_sENC_EXT_AXIS
S3	Array	Comparand array	No	-	-	_sPoint2D [0..1000]
S4	Size	Number of comparands	No	-	1 to 1000	INT
S5	Mode	Comparison mode 0: Configuration mode (supported by the local encoder axis) 1: Time control 2: Reserved 3: Level control	No		0 to 3	INT
S6	Parameter	Control parameters Time control: output active duration, in μs. Level control: initial level; 0 indicates low level and non-zero indicates high level.	Yes	0	Positive number 0 Negative number	REAL
S7	OutputEnable	Reserved	Yes	0	0 to 1	INT
S8	InterruptMap	Reserved	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	OutStatus	Port output state	Yes	OFF	ON OFF	BOOL
D4	WarningX	Warning output of x-axis	Yes	OFF	ON OFF	BOOL
D5	WarningY	Warning output of y-axis	Yes	OFF	ON OFF	BOOL
D6	Index	Index of the next comparand	Yes	0	0 to 999	INT
D7	CommandAbort-ed	Abortion of execution	Yes	OFF	ON OFF	BOOL
D8	Error	Error	Yes	OFF	ON OFF	BOOL
D9	ErrorID	Error code	Yes	0	*1	INT

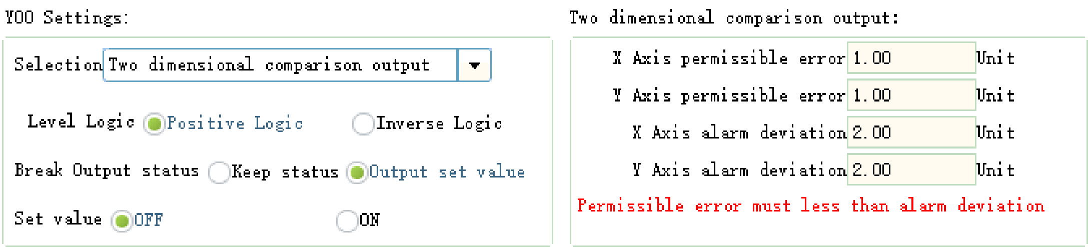
Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

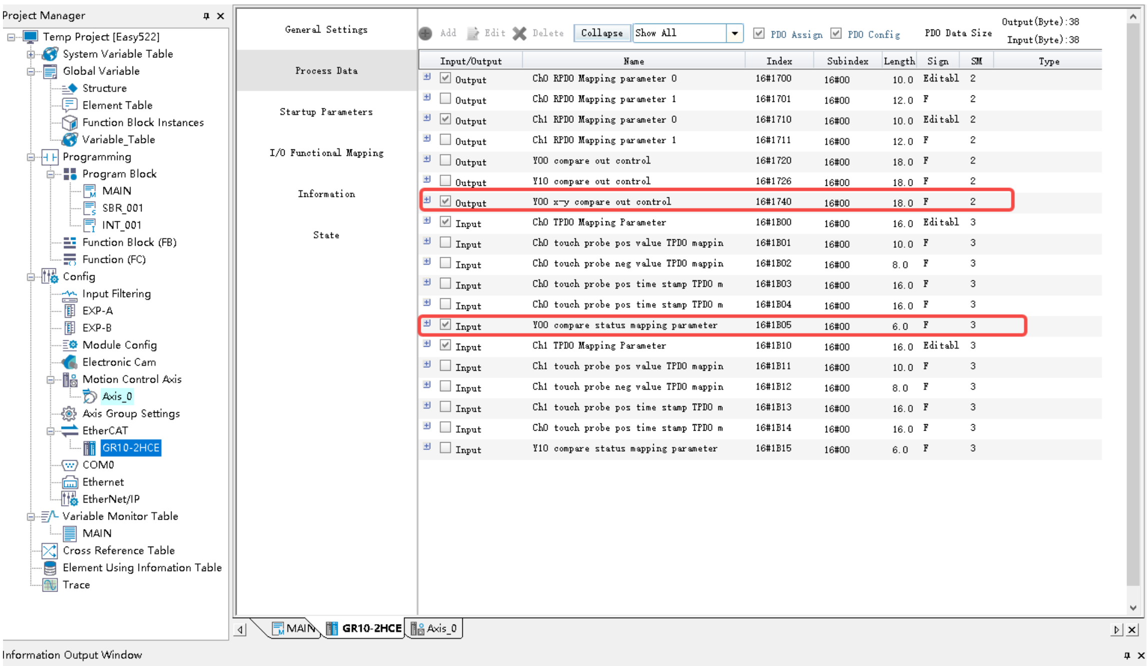
Function Description (Bus Encoder Axis)

The two channels of the GR10-2HCE module can work together to implement the two-dimensional comparison output function. The first channel is used as the x-axis input, and the second channel is used as the y-axis input. Therefore, when calling this instruction, ensure that the encoder axes specified by AxisX and AxisY are bound to the same GR10-2HCE module. Moreover, ensure that AxisX is bound to the first channel and AxisY is bound to the second channel.

You need to assign the two-dimensional comparison output function to the Y00 output terminal on the AxisX mode/parameter configuration interface, and set the allowable error and warning deviation for each axis of the two-dimensional comparison output.



The PDO options as shown in the following figure need be selected on the process data page of the GR10-2HCE module. In this case, Y00 of the GR10-2HCE module can be used as a two-dimensional comparison output terminal.



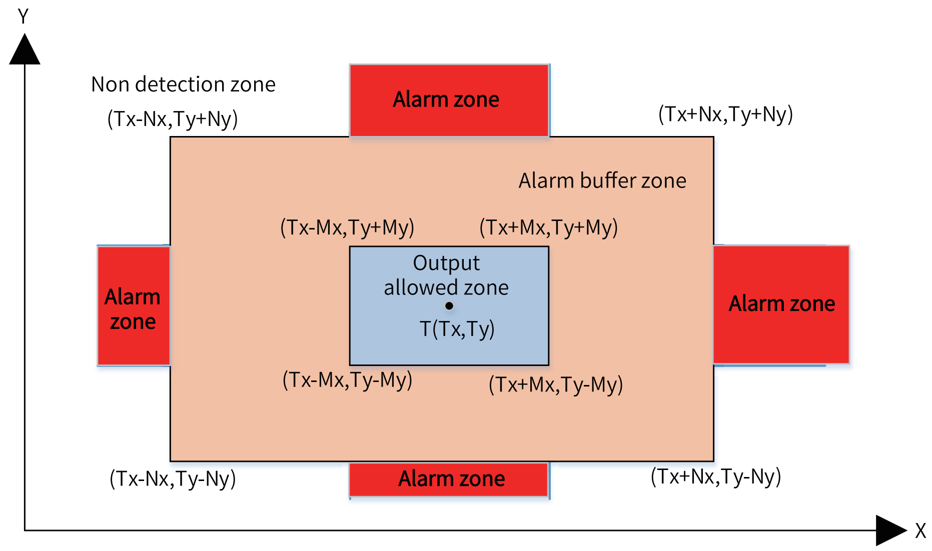
Two-dimensional Comparison Output Principles

Set T (Tx, Ty) as the target point on the plane, M (Mx, My) as the maximum allowable position error, and N (Nx, Ny) as the warning buffer detection range.

As shown in the following figure, when the system runs to the allowable output area, it is considered to have entered the target area. After the system enters the target area, the controller will select the output based on the motion profile of the encoder axis to achieve optimal control (Note 1).

When the x-axis (y-axis) has entered the allowable output area, but the y-axis (x-axis) has not entered the warning buffer area, that is, it is still in the areas in red in the following figure, the WarningY (WarningX) output of the instruction becomes active and remains active until the y-axis (x-axis) enters the warning buffer area.

In the non-detection area and warning buffer area, since the comparison output conditions are not satisfied, there is no comparison output signal or warning signal.



Note

After reaching the range of the comparison point, if the encoder input does not change any more, the comparison output will be generated after 1 second.

Comparison Output Modes

The two-dimensional comparison output instruction supports the time control and level control modes. For details, see the description of the ENC_ArrayCompare instruction.

Multi-execution

If a new comparison output instruction is triggered during the comparison output process, the previous instruction being executed is aborted, its CommandAborted signal output becomes active, and the state of the Y output terminal is determined by the new instruction.

Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

3.12.10 ENC_ReadStatus

ENC_ReadStatus – Encoder state read

Graphic Block

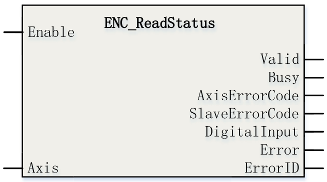
Instruction	Name	LD Expression	LiteST Expression
ENC_ReadStatus	Encoder axis state read (bus encoder axis)		ENC_ReadStatus(Enable := ???, Axis := ???, Valid => , Busy => , AxisErrorCode => , SlaveErrorCode => , DigitalInput => , Error => , ErrorID =>);

Table 3–292 Instruction format

16-bit Instruction	ENC_ReadStatus: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Bus encoder axis	No	-	-	_sENC_EXT_AXIS
D1	Valid	Active	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	AxisErrorCode	Axis fault code	Yes	0	-	INT
D4	SlaveErrorCode	Drive fault code	Yes	0	-	INT
D5	DigitalInput	DI terminal state Bit0: CHn-X0 terminal state Bit1: CHn-X1 terminal state Bit2: CHn-X2 terminal state Bit3: CHn-X3 terminal state Others: Reserved	Yes	0	-	INT
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Function Description (Bus Encoder Axis)

This instruction reads the state of a bus encoder axis.

AxisErrorCode is used to read the fault code of the bus encoder axis. For details about the fault code, see the list of fault codes. SlaveErrorCode displays the fault code of the GR10-2HCE module. For details about the fault code list, see the application manual of the GR10-2HCE module.

DigitalInput displays the DI terminal state of the GR10-2HCE module. When the axis is bound to the CH0 channel, it displays the state of X00, X01, X02, and X03; when the axis is bound to the CH1 channel, it displays the states of X10, X11, X12, and X13.

Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

3.12.11 ENC_DigitalOutput

ENC_DigitalOutput – Encoder DO control

Graphic Block

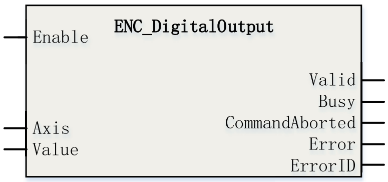
Instruction	Name	LD Expression	LiteST Expression
ENC_ DigitalOutput	Encoder axis DO control (bus encoder axis)		ENC_DigitalOutput(Enable := ???, Axis := ???, Value := , Valid => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–293 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_DigitalOutput: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Bus encoder axis instruction	No	-	-	_sENC_EXT_ AXIS
S2	Value	DO terminal set value Bit0: CHn-Y0 output state Bit1: CHn-Y1 output state Bit2: CHn-Y2 output state Others: Reserved	Yes	0	-	INT
D1	Valid	Output active	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL

D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Function Description (Bus Encoder Axis)

This instruction is used to set the DO terminal state of the GR10-2HCE module.

When the bus encoder axis is bound to the CH0 channel of the GR10-2HCE module, the instruction controls the states of Y00, Y01, and Y02; when the axis is bound to the CH1 channel of the GR10-2HCE module, the instruction controls the states of Y10, Y11, and Y12.

Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

3.12.12 ENC_ResetCompare

ENC_ResetCompare – Encoder comparison output reset

Graphic Block

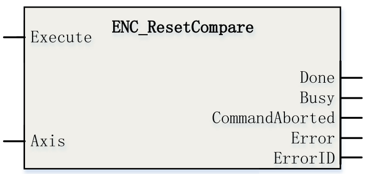
Instruction	Name	LD Expression	LiteST Expression
ENC_ResetCompare	Encoder axis comparison output reset (bus encoder axis)		ENC_ResetCompare(Execute := ???, Axis := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–294 Instruction format

16-bit Instruction	ENC_ResetCompare: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Bus encoder axis	No	-	-	_sENC_EXT_AXIS
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL

D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

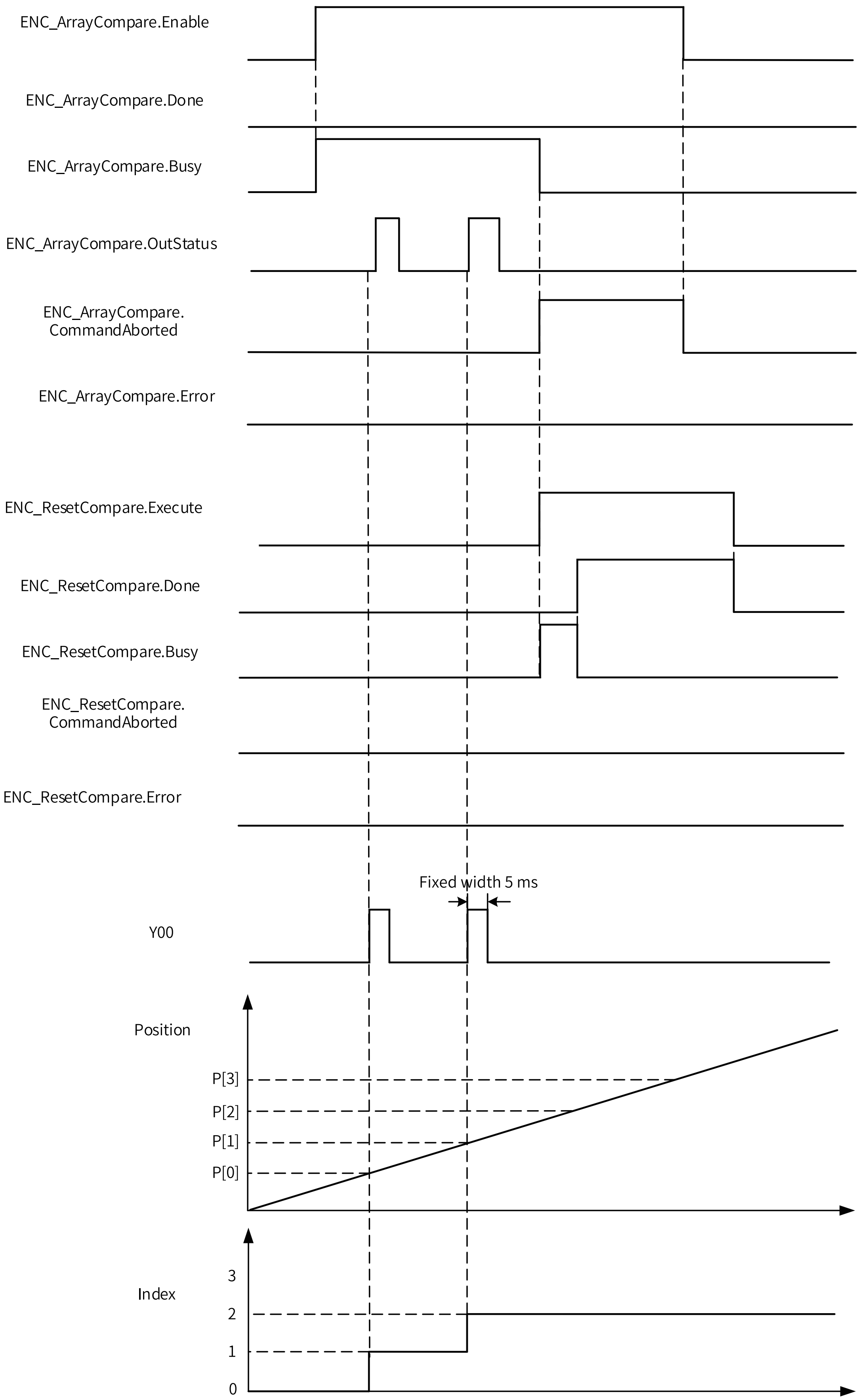
Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Function Description (Bus Encoder Axis)

This instruction aborts the execution of three comparison output instructions (ENC_StepCompare, ENC_GroupArrayCompare, and ENC_ArrayCompare) and sets the comparison output terminal to OFF.

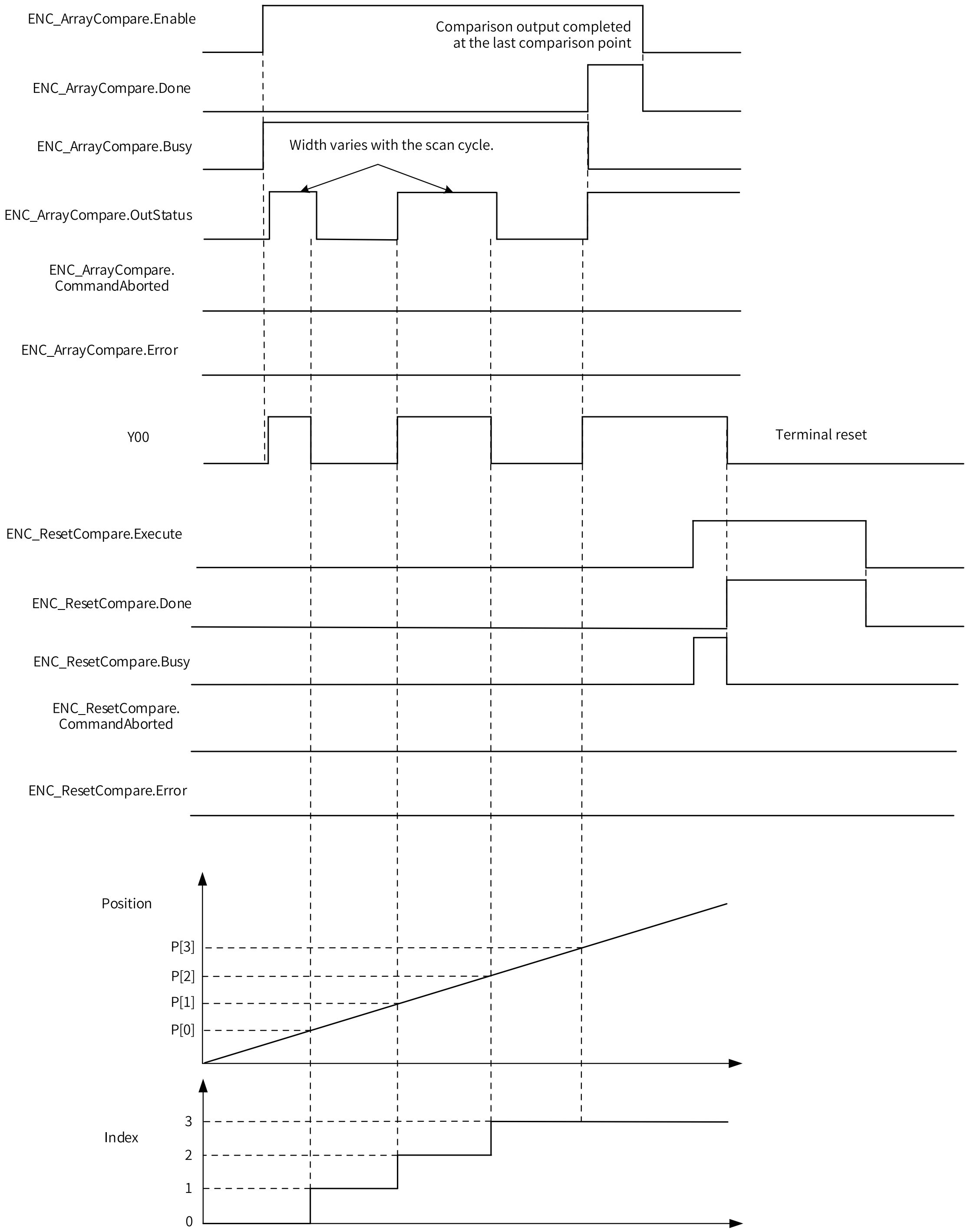
1. Abortion rules in time or pulse mode
   When the comparison output instruction is set to time or pulse mode, if this instruction is called when the Done signal of the comparison output instruction is OFF, the comparison output instruction is aborted, its CommandAborted signal output becomes active, and the comparison output terminal is forced to OFF. After that, the Done signal output of ENC_ResetCompare instruction becomes active.



2. Abortion rules in level mode

When the comparison output instruction is set to level mode, if this instruction is called when the Done signal of the comparison output instruction is OFF, the instruction processing is the same as that in time or pulse mode mentioned above.

If the ENC_ResetCompare instruction is called when the Done signal of the comparison output instruction is ON, that is, when the comparison output is completed, the output signals (Done, Busy, CommandAborted) of the comparison output instruction remain unchanged, the controlled comparison output terminal is forced to OFF, and then the Done signal output of the ENC_ResetCompare instruction becomes ON.



Multi-execution

If a new comparison output reset instruction is triggered during the comparison reset process, the previous instruction being executed is aborted, and its CommandAborted signal output becomes active.

Function Description (Local Encoder Axis)

The local encoder axis does not support this instruction.

3.12.13 ENC_SetUnit

ENC_SetUnit – Gear ratio setting

Graphic Block

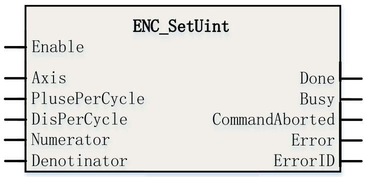
Instruction	Name	LD Expression	LiteST Expression
ENC_SetUnit	Gear ratio setting		ENC_SetUnit(Enable := ???, Axis := ???, PlusePerCycle := ???, DisPerCycle := ???, Numerator := ???, Denotinator := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–295 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_SetUnit: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-		_sENC_ EXT_AXIS
S2	PlusePerCycle	Number of pulses per revolution of the encoder	No	-	Positive number	DINT
S3	DisPerCycle	Distance per revolution of the rotary table	No	-	0.01 to 9999999	REAL
S4	Numerator	Gear ratio (numerator)	No	1	Positive number	DINT
S5	Denotinator	Gear ratio (denominator)	No	1	Positive number	DINT
D1	Done	Completion flag	Yes	FALSE	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON/OFF	BOOL

D3	CommandA-borted	Abortion	Yes	FALSE	ON/OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

This instruction is used for the PLC to reconfigure the gear ratio of the local encoder axis before enabling counting after power-on, program download, or execution of a RUN/STOP operation.

The gear ratio of a local encoder axis is calculated as follows:

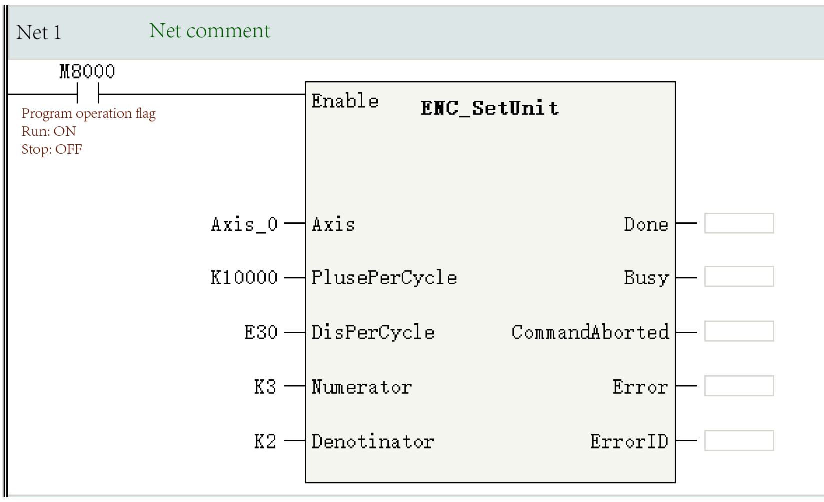
$$\text{Gear ratio} = \frac{PlusePreCycle * \ Numerator}{DisPerCycle * \ Denotinator}$$

Example

This example describes how to set the local encoder axis parameters as follows automatically after power-on of the PLC.

Parameter	Value
Number of pulses per revolution of the encoder	10000
Distance per revolution of the rotary table	30
Gear ratio (numerator)	3
Gear ratio (denominator)	2

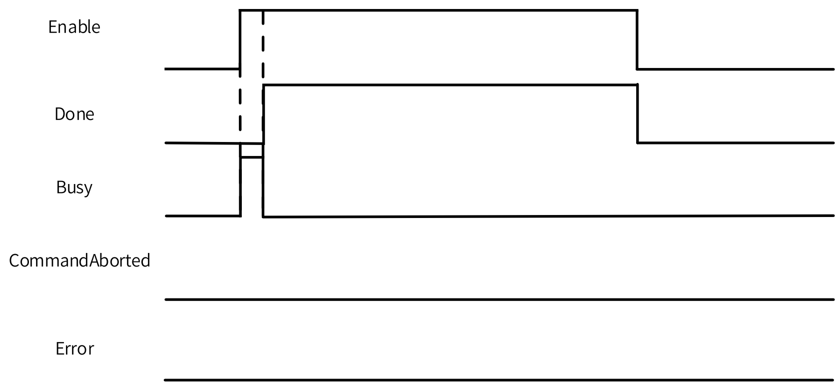
Add the following program in the PLC. Use M8000 to trigger the instruction.



Note

If this instruction is called during program execution, the local encoder axis will be re-initialized, causing a sudden change in its feedback position.

Timing Diagram



3.12.14 ENC_SetLineRotationMode

ENC_SetLineRotationMode – Rotation mode setting

Graphic Block

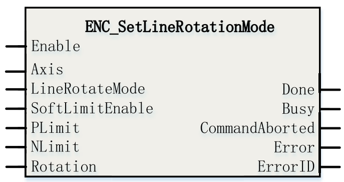
Instruction	Name	LD Expression	LiteST Expression
ENC_ SetLineRotationMode	Rotation mode setting		ENC_SetLineRotationMode(Enable := ???, Axis := ???, LineRotateMode := ???, SoftLimitEnable := ???, Plimit := ???, Nlimit := ???, Rotation := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–296 Instruction format

16-bit Instruction	-					
32-bit Instruction	ENC_SetLineRotationMode: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name	No	-		_sENC_ EXT_AXIS
S2	LineRotate- Mode	Linear/Rotary mode 0: Linear mode 1: Rotary mode	No	-	0 to 1	INT

S3	SoftLimitEna-ble	Limiting enable in linear mode ON: Enabled OFF: Disabled	No	-	Positive number	BOOL
S4	PLimit	Positive limit in linear mode	No	-		REAL
S5	NLimit	Negative limit in linear mode	No	-		REAL
S6	Rotation	Rotation period in rotary mode	No	-		REAL
D1	Done	Completion flag	Yes	FALSE	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	FALSE	ON/OFF	BOOL
D3	CommandA-borted	Abortion	Yes	FALSE	ON/OFF	BOOL
D4	Error	Error flag	Yes	FALSE	ON/OFF	BOOL
D5	ErrorID	Fault code	Yes	0	-	INT

Function Description (Bus Encoder Axis)

The bus encoder axis does not support this instruction.

Function Description (Local Encoder Axis)

This instruction is used for the PLC to reconfigure the linear/rotary mode of the local encoder axis before enabling counting after power-on, program download, or execution of a RUN/STOP operation.

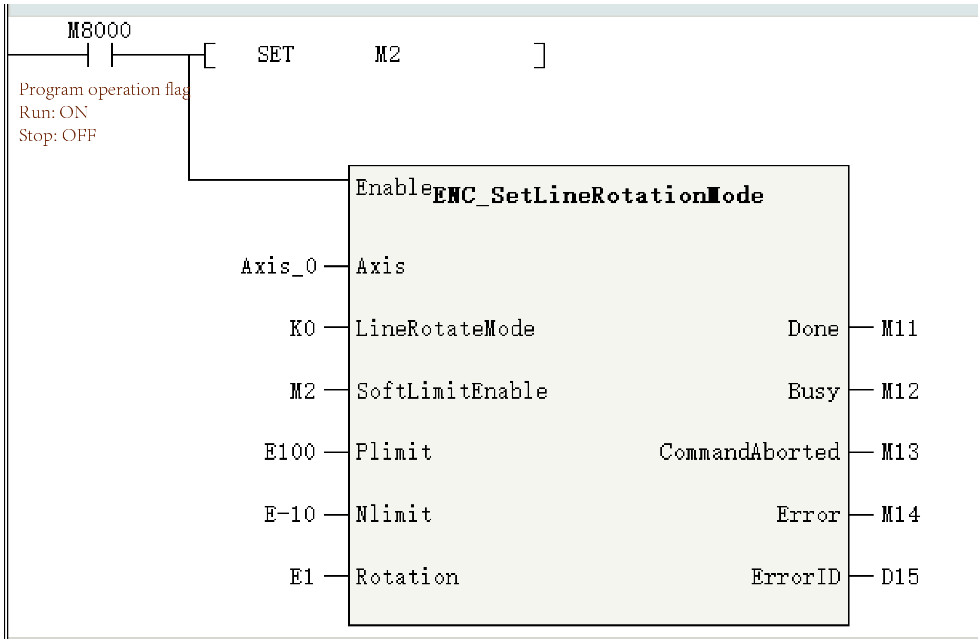
When LineRotateMode is set to 0, the local encoder axis works in linear mode.

In linear mode, limiting is disabled when SoftLimitEnable is OFF. When SoftLimitEnable is ON, limiting is enabled. At this time, PLimit indicates the positive limit, and NLimit indicates the positive limit.

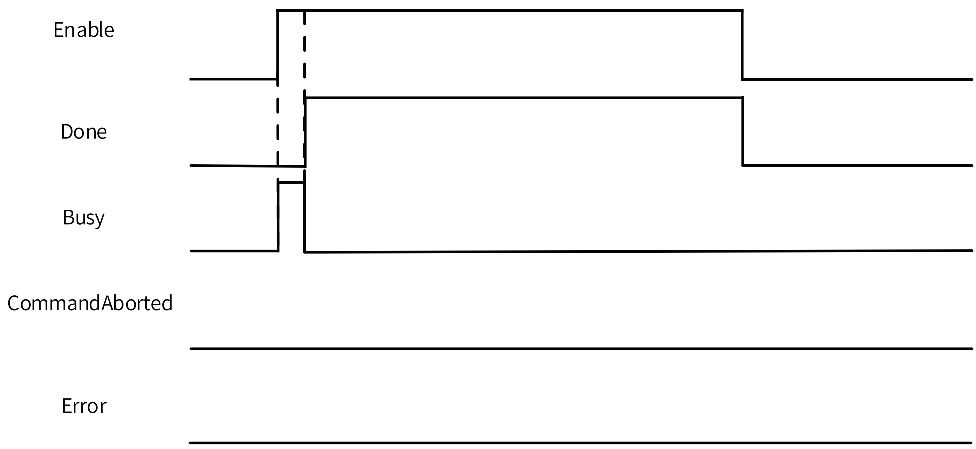
When LineRotateMode is set to 1, the local encoder axis works in rotary mode. Rotation indicates the rotation cycle.

Example

In this example, the local encoder axis switches to linear mode automatically after power-on of the PLC. Limiting is enabled. The positive limit is 100, and the negative limit is –10. The program is as follows.



Timing Diagram



3.12.15 HC_Preset

This instruction sets the counter value as the preset value according to the trigger signal.
HC_Preset – High-speed counter preset

Graphic Block

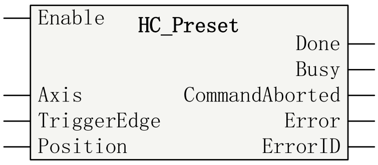
Instruction	Name	LD Expression	LiteST Expression
HC_Preset	High-speed counter preset		HC_Preset(Enable := ???, Axis := ???, TriggerEdge := , Position := ???, Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–297 Instruction format

16-bit Instruction	-					
32-bit Instruction	HC_Preset: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	TriggerEdge	Trigger edge 0: Triggered on the rising edge of the instruction 1: Triggered on the rising edge of the input terminal X 2: Triggered on the falling edge of the input terminal X 3: Triggered on the rising or falling edge of the input terminal X	Yes	0	0 to 3	INT
S3	Position	Preset position (unit: Unit)	No	-	-	REAL
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see "3.12.21 Error Codes" on page 602Error Codes.

Table 3–298 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	√	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	√[1]	-	√	-	√	-	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

The HC_Preset instruction assigns the counter axis position according to the preset condition.

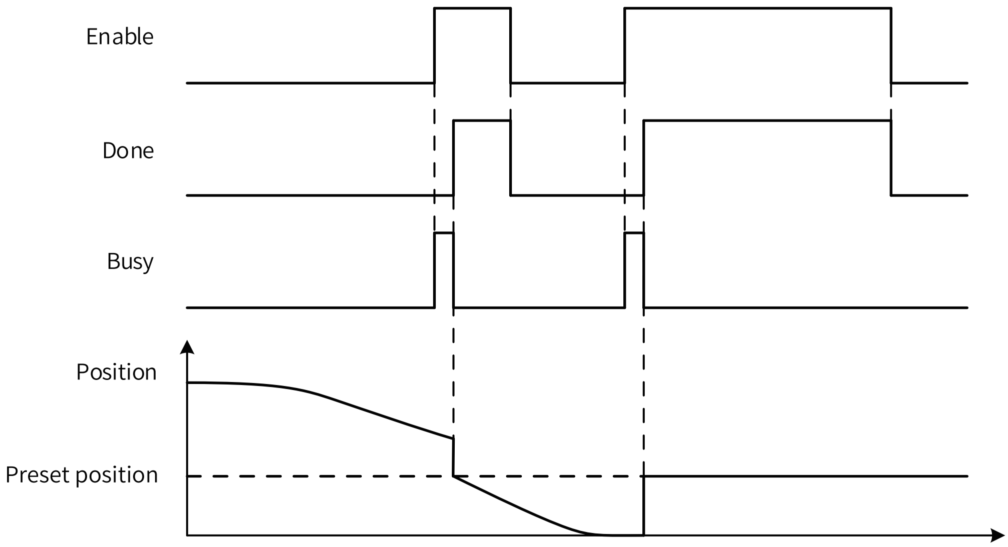
TriggerType specifies the preset condition, including the rising edge of the instruction or external terminal input.

Item	Setting	Definition
TriggerType	0	Triggered on the rising edge of the instruction flow
	1	Triggered on the rising edge of the external input X
	2	Triggered on the falling edge of the external input X
	3	Triggered on the rising or falling edge of the external input X

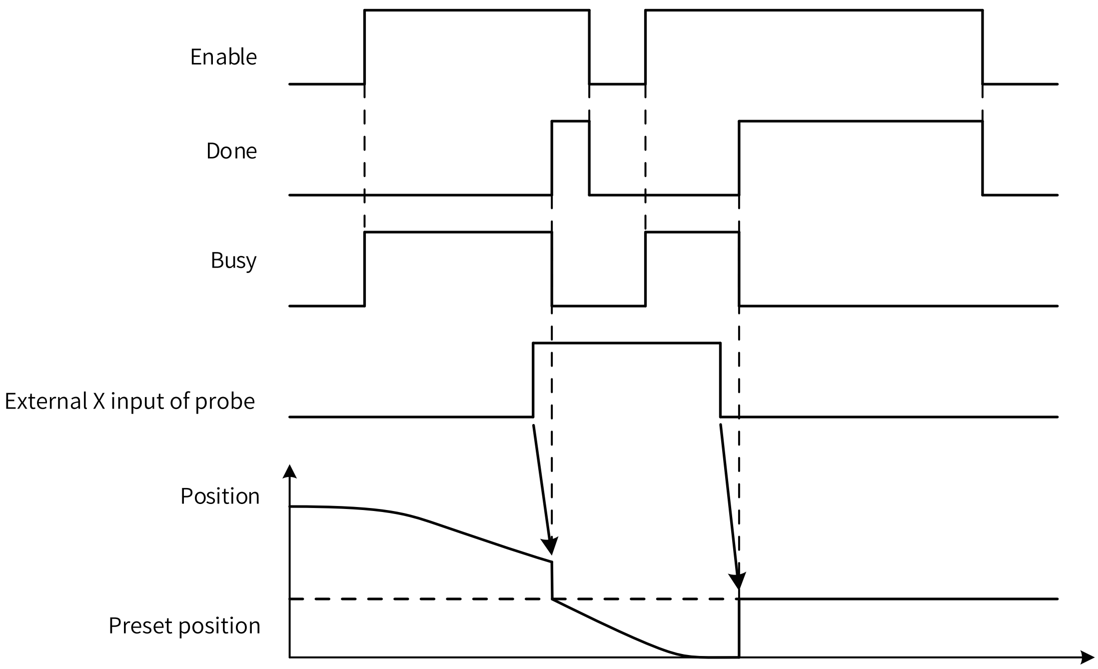
When the preset condition is set to the external input X, you need to select the preset function in counter parameter configuration and select the input terminal and trigger condition. The input terminal can be set to any one of X0 to X7, and the trigger condition can be the rising edge or falling edge.

Timing Diagram

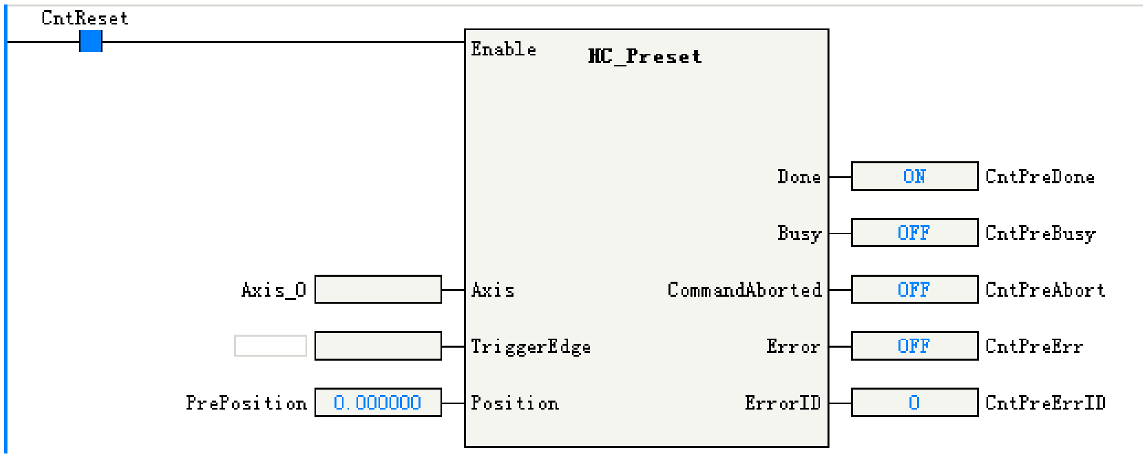
- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising edge of the instruction (TriggerEdge = 0).



- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising or falling edge of the external input X (TriggerEdge = 3).



Instruction Example



3.12.16 HC_Counter

This instruction controls the high-speed counter to start or stop counting.

HC_Counter – High-speed counter enable

Graphic Block

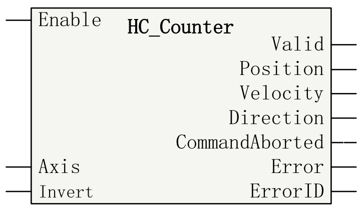
Instruction	Name	LD Expression	LiteST Expression
HC_Counter	High-speed counter enable		HC_Counter(Enable := ???, Axis := ???, Invert := , Valid => , Position => , Velocity => , Direction => , CommandAborted => , Error => , ErrorID =>);

Table 3–299 Instruction format

16-bit Instruction	-					
32-bit Instruction	HC_Counter: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	Invert	Counting inversion	Yes	0	0 to 1	INT

D1	Valid	Active state, which is ON when the counter enters the counting state	Yes	OFF	ON OFF	BOOL
D2	Position	Current position (unit: Unit)	Yes	0	-	REAL
D3	Velocity	Current velocity (unit: Unit/s)	Yes	0	-	REAL
D4	Direction	Counting direction	Yes	OFF	ON OFF	BOOL
D5	CommandAbort-ed	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see "3.12.21 Error Codes" on page 602Error Codes.

Table 3–300 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	-	-	-	√	√	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	√[1]	-	√	-	√	-	-	-	-
D7	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

The HC_Counter instruction implements position counting and velocity measurement of the counter axis.

The counter axis position (unit: Unit) varies within a certain range based on the mode setting.

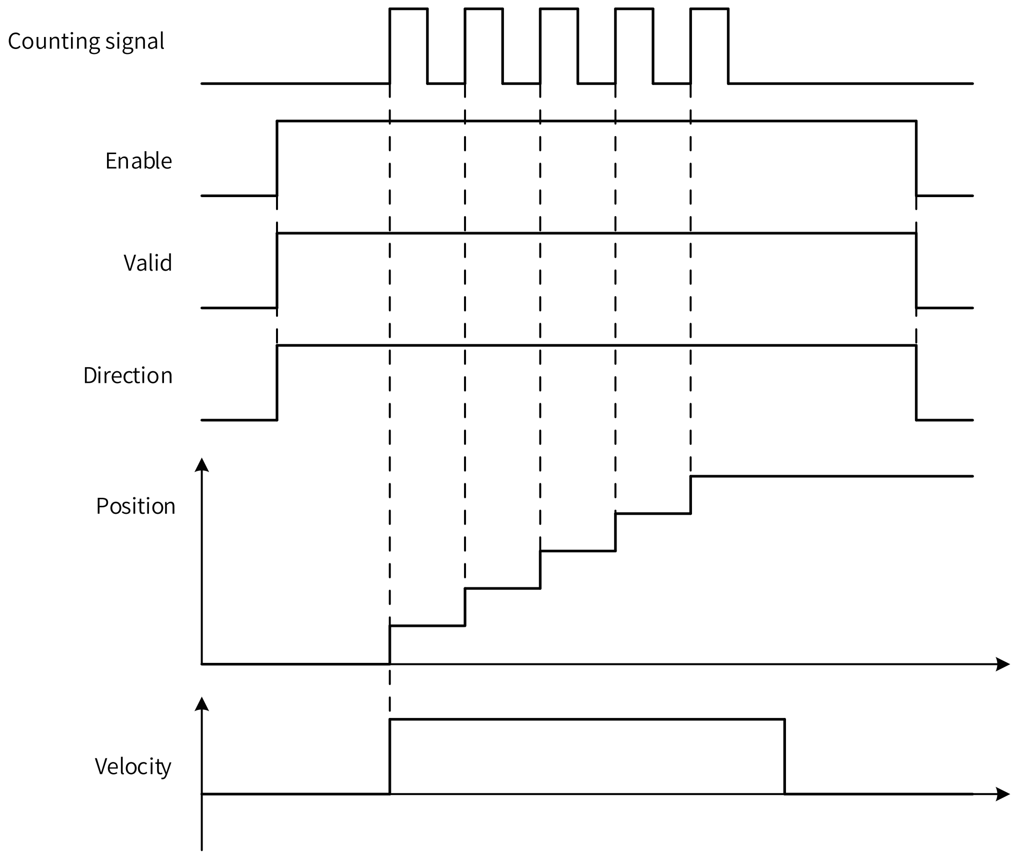
Invert (counting inversion)

Invert specifies the counting direction of the counter. The following table lists the counting directions of different counting modes. Modification on Invert takes effect only after this function block instruction is enabled again.

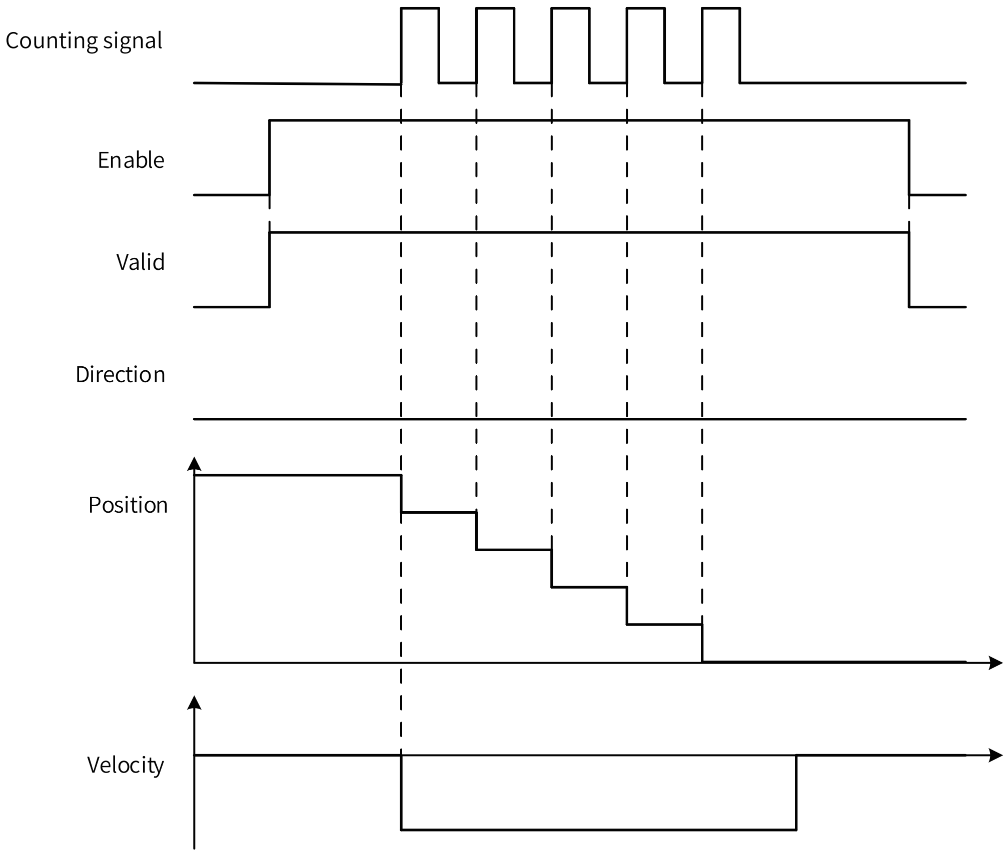
Invert	A/B Phase	Pulse+Direction	CW/CCW	Single-phase Counter
0	Phase A leading phase B, counting up Phase B leading phase A, counting down	Direction signal = OFF, counting down Direction signal = ON, counting up	Phase A, counting up Phase B, counting down	Counting up
1	Phase A leading phase B, counting down Phase B leading phase A, counting up	Direction signal = OFF, counting up Direction signal = ON, counting down	Phase A, counting down Phase B, counting up	Counting down

Timing Diagram

- In pulse+direction mode, if the direction signal is ON and Invert is set to 0, or the direction signal is OFF and Invert is set to 1, the counter counts up, as shown in the following figure.

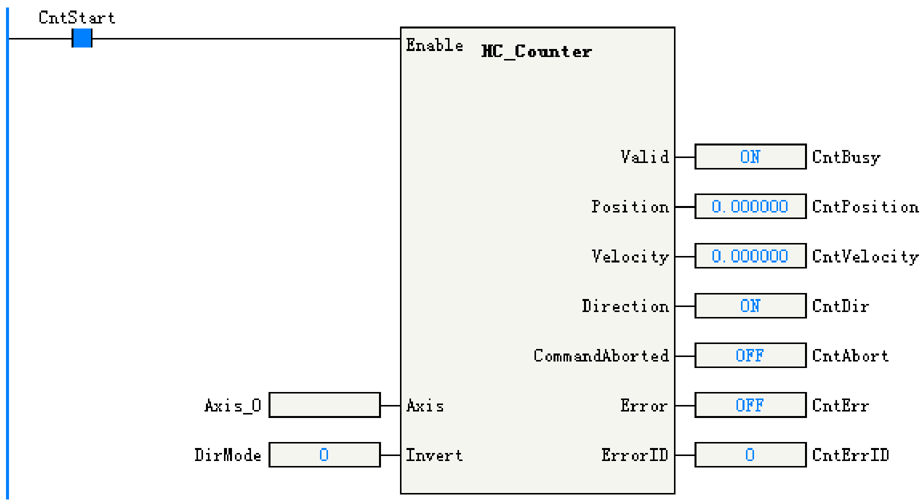


- In pulse+direction mode, if the direction signal is ON and Invert is set to 1, or the direction signal is OFF and Invert is set to 0, the counter counts down, as shown in the following figure.



Instruction Example

The counter axis velocity is the current real-time velocity (unit: Unit/s). The minimum velocity that can be measured by the counter axis is the velocity corresponding to 1 pulse of the counter within 1s. If 1 pulse of the counter corresponds to 0.1 Unit, the minimum velocity that can be measured is 0.1 Unit/s.



3.12.17 HC_TouchProbe

This instruction records the counter value based on the trigger signal.

HC_TouchProbe – High-speed counter probe

Graphic Block

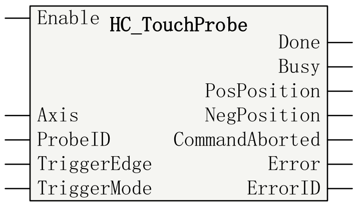
Instruction	Name	LD Expression	LiteST Expression
HC_TouchProbe	High-speed counter probe		HC_TouchProbe(Enable := ???, Axis := ???, ProbeID := ???, TriggerEdge := , TriggerMode := , Done => , Busy => , PosPosition => , NegPosition => , CommandAborted => , Error => , ErrorID =>);

Table 3–301 Instruction format

16-bit Instruction	-					
32-bit Instruction	HC_TouchProbe: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT16
S2	ProbeID	Probe ID 0: Probe 1 1: Probe 2	No	0	0 to 1	INT16
S3	TriggerEdge	Trigger edge 1: Triggered on the rising edge of the external input X 2: Triggered on the falling edge of the external input X 3: Triggered on the rising and falling edges of the external input X	Yes	1	1 to 3	INT16
S4	TriggerMode	Trigger mode 0: Single trigger 1: Continuous trigger	Yes	0	0 to 1	INT16
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	PosPosition	Position latched on the rising edge (unit: Unit)	Yes	0	-	REAL32
D4	NegPosition	Position latched on the falling edge (unit: Unit)	Yes	0	-	REAL32

D5	CommandA-borted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D6	Error	Error	Yes	OFF	ON OFF	BOOL
D7	ErrorID	Error code	Yes	0	*1	INT16

Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Table 3–302 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	√[1]	-	√	-	√	-	-	-	-
D7	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

The HC_TouchProbe instruction can latch the counter axis position value when the external input trigger condition is active.

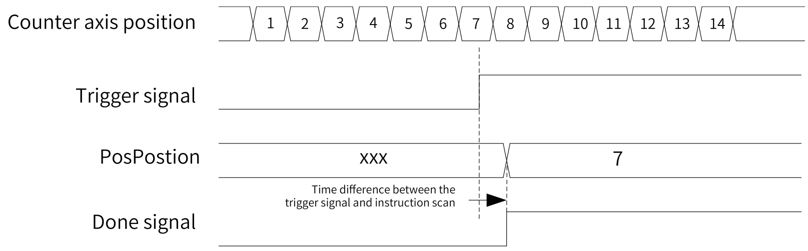
Each counter axis supports two probes. During use, you need to select the corresponding probe function in counter parameter configuration as well as the input terminal and trigger condition. The input terminal can be set to any one of X1 to X7.

TriggerEdge specifies the probe trigger edge. The rising edge trigger position is latched in the output parameter PosPosition, and the falling edge trigger position is latched in the output parameter NegPosition.

Item	Setting	Definition
TriggerEdge	1	Triggered on the rising edge of the external input X
	2	Triggered on the falling edge of the external input X
	3	Triggered on the rising or falling edge of the external input X

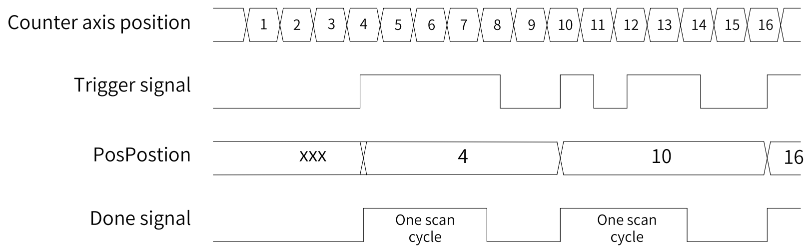
TriggerMode in the instruction can be set to the single trigger or continuous trigger mode.

- If the single trigger mode is used, when the function block instruction flow and the external input trigger condition are active, the counter axis position is latched once, and the Done signal is output. The counter axis position is latched in real time based on the trigger edge, which is not affected by program execution. During instruction execution, affected by the scan cycle, when the program scans and executes to the latched instruction, it updates the latched position to the output parameter of the instruction.



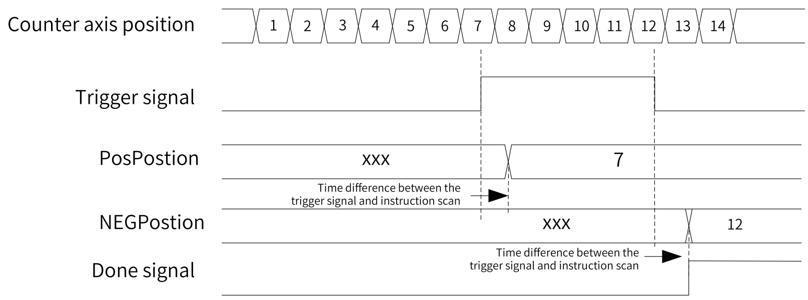
Single triggering on the rising edge

- If the continuous trigger mode is used, when the function block instruction flow and the external input trigger condition are active, the counter axis position is latched, and the Done signal that is active for one scan cycle is output. When the Done signal becomes OFF and the external input trigger condition is active, the counter axis position continues to be latched and the Done signal that is active for one scan cycle is output. During the scan cycle in which the Done signal is active, if the external input trigger condition is active, the counter axis position is not latched at this time.

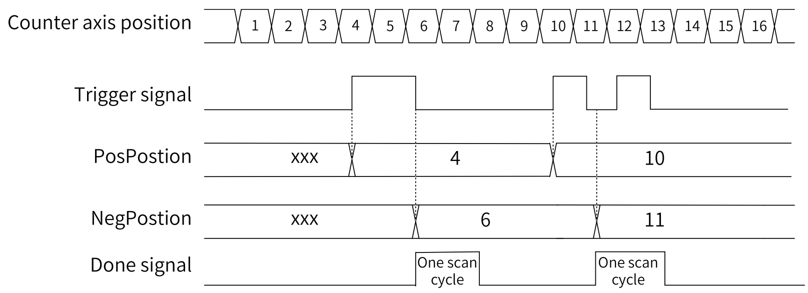


Continuous triggering on the rising edge

- When the dual-edge trigger mode is used, the Done signal is output after the instruction is triggered on both the rising and falling edges to complete the latch. In single trigger mode, the Done signal remains active until the instruction execution is completed; in continuous trigger mode, the Done signal is active for one scan cycle, and the latch signal is not responded within the scan cycle when the Done signal is active.



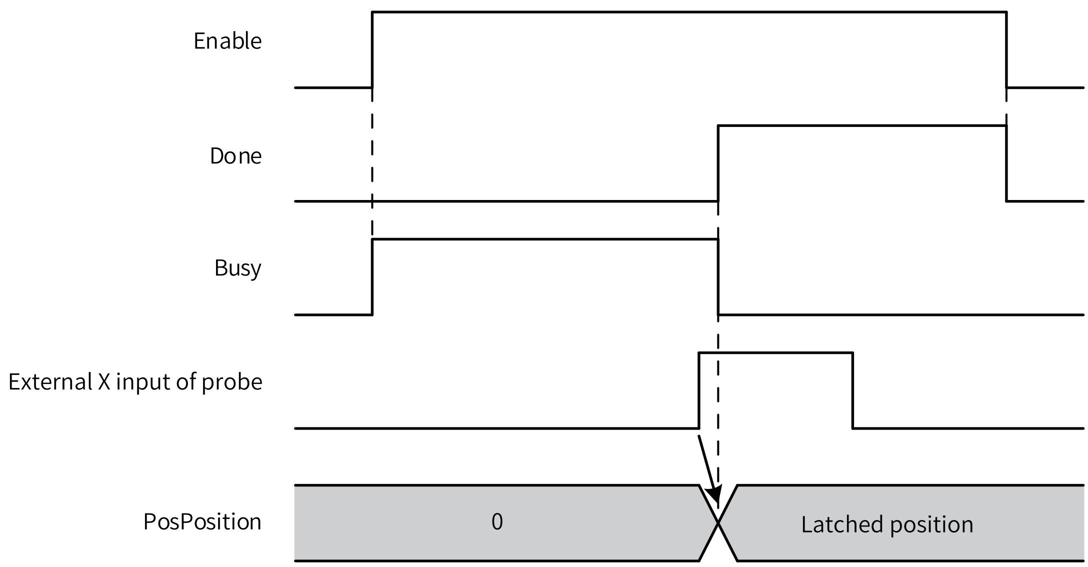
Single triggering on both the rising and falling edges



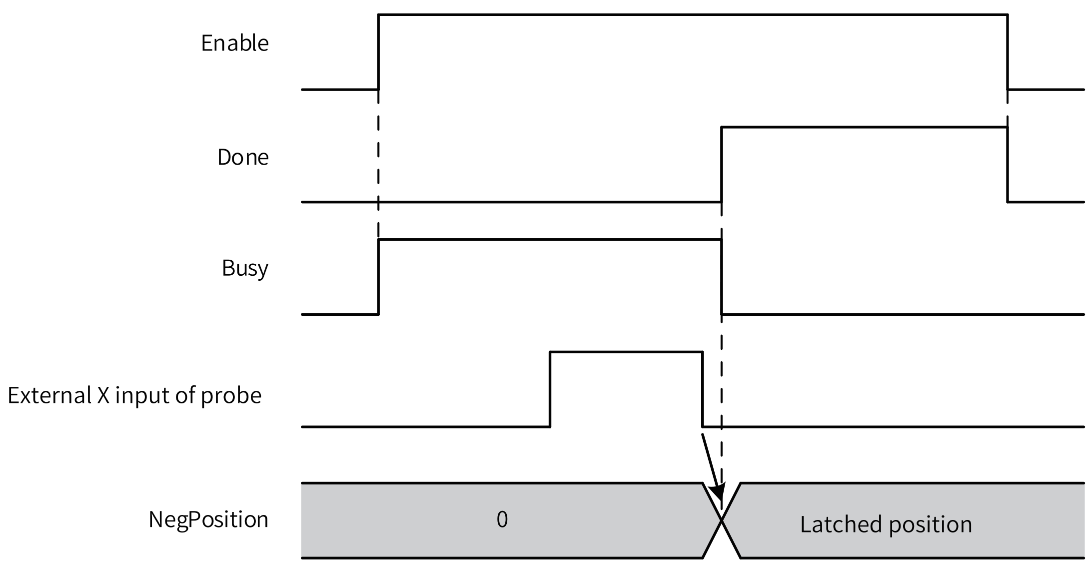
Continuous triggering on both the rising and falling edges

Timing Diagram

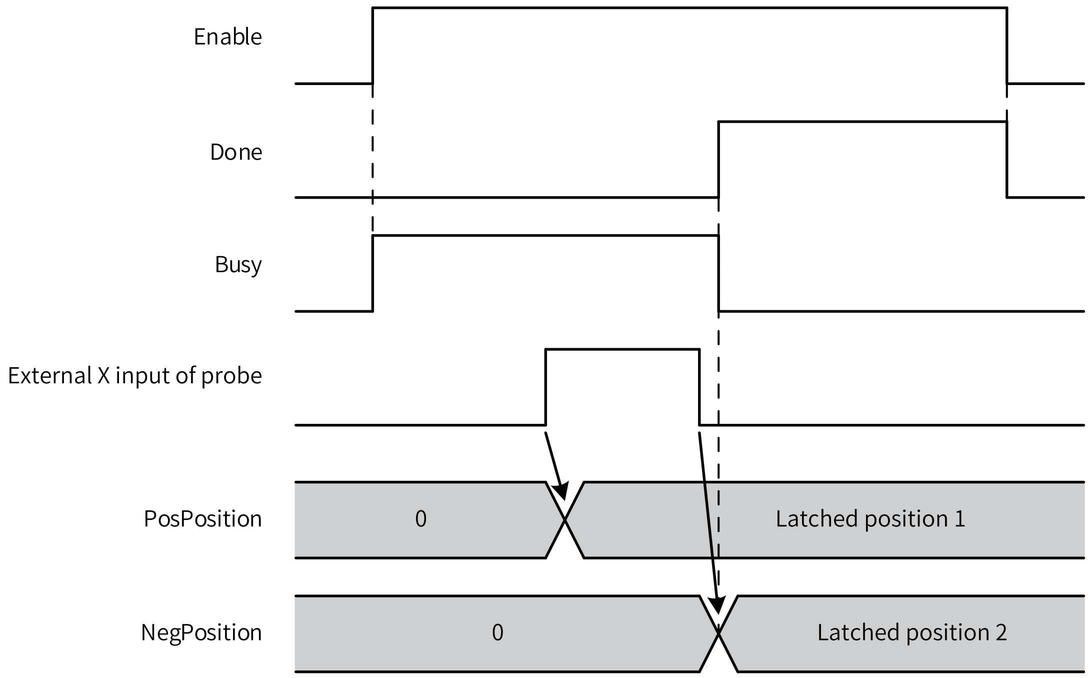
- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising edge of the external input X (TriggerEdge = 1) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



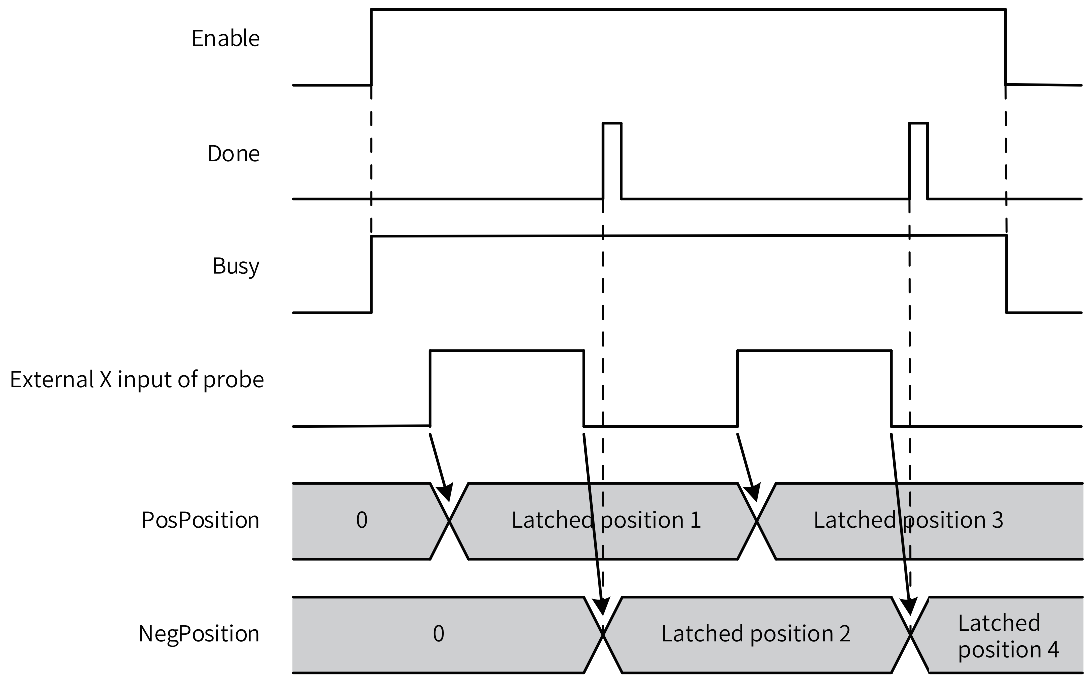
- The timing diagram of the instruction is as follows when TriggerEdge is set to the falling edge of the external input X (TriggerEdge = 2) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



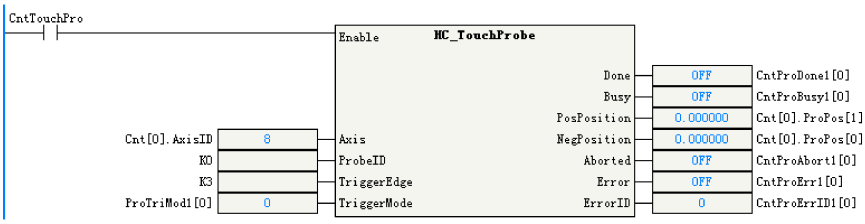
- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising and falling edges of the external input X (TriggerEdge = 3) and TriggerMode is set to the single trigger mode (TriggerMode = 0).



- The timing diagram of the instruction is as follows when TriggerEdge is set to the rising and falling edges of the external input X (TriggerEdge = 3) and TriggerMode is set to the continuous trigger mode (TriggerMode = 1).



Instruction Example



3.12.18 HC_Compare

This instruction detects whether the counter count value reaches the specified value.

HC_Compare – High-speed counter comparison

Graphic Block

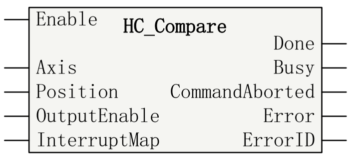
Instruction	Name	LD Expression	LiteST Expression
HC_Compare	High-speed counter comparison		HC_Compare(Enable := ???, Axis := ???, Position := ???, OutputEnable := , InterruptMap := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

Table 3–303 Instruction format

16-bit Instruction	-					
32-bit Instruction	HC_Compare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	Position	Comparison position (unit: Unit)	No	-	-	REAL
S3	OutputEnable	Hardware output enable 0: Disabled 1: Enabled	Yes	0	0 to 1	INT

S4	InterruptMap	Interrupt generation and association when the comparand and count value match 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL
D3	CommandAbort-ed	Abortion of execution	Yes	OFF	ON OFF	BOOL
D4	Error	Error	Yes	OFF	ON OFF	BOOL
D5	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see *"3.12.21 Error Codes" on page 602Error Codes*.

Table 3–304 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	√	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	√[1]	-	√	-	√	-	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

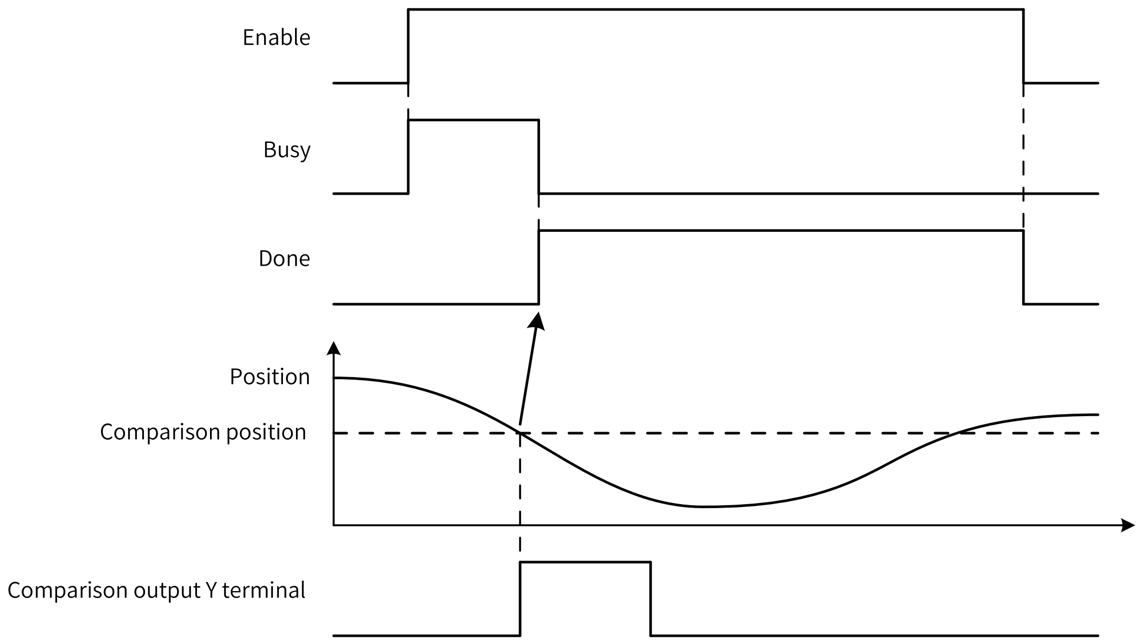
[1] The X element is not supported.

Function and Instruction Description

The HC_Compare instruction compares the counter axis position with a single position.

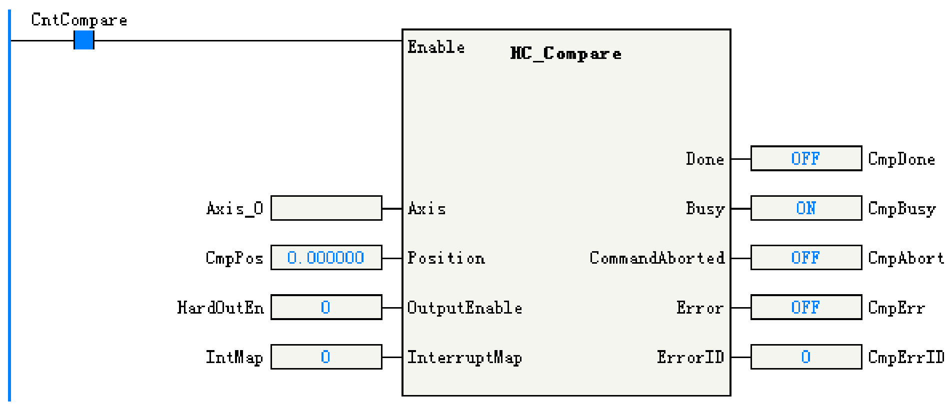
Timing Diagram

The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1).



Instruction Example

The instruction compares the counter axis with a single position. When the instruction flow is active, the Done signal is output after the counter axis position reaches the comparison position.



3.12.19 HC_ArrayCompare

This instruction continuously detects whether the counter count value reaches the specified array sequence.

HC_ArrayCompare – High-speed counter array comparison

Graphic Block

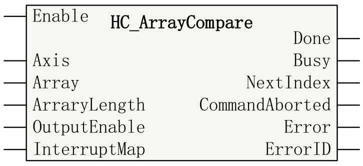
Instruction	Name	LD Expression	LiteST Expression
HC_ArrayCompare	High-speed counter array comparison		HC_ArrayCompare(Enable := ???, Axis := ???, Array := ???, ArrayLength := ???, OutputEnable := , InterruptMap := , Done => , Busy => , NextIndex => , CommandAborted => , Error => , ErrorID =>);

Table 3–305 Instruction format

16-bit Instruction	-					
32-bit Instruction	HC_ArrayCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	Array	Comparison position array (unit: Unit)	No	-	-	REAL
S3	ArrayLength	Array length	No	-	0 to 100	INT
S4	OutputEnable	Hardware output enable 0: Disabled 1: Enabled	Yes	0	0 to 1000	INT
S5	InterruptMap	Interrupt generation and association when the comparand and count value match 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	Completion flag	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON OFF	BOOL

D3	NextIndex	Index of the next comparand	Yes	0	0 to 100	INT
D4	Aborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see "3.12.21 Error Codes" on page 602Error Codes.

Table 3–306 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

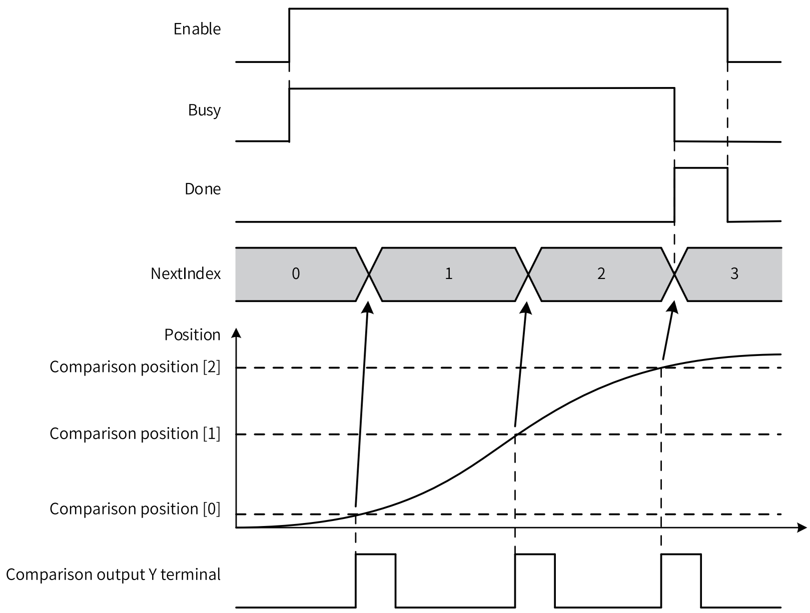
The HC_ArrayCompare instruction compares the counter axis position with multiple positions continuously.

When the instruction flow is active, the counter axis position is compared with the position value in array 0. If they are equal, the counter axis position is compared with the next position value in the array. ArrayLength in the instruction specifies the array length. After all the positions are compared, the Done signal is output.

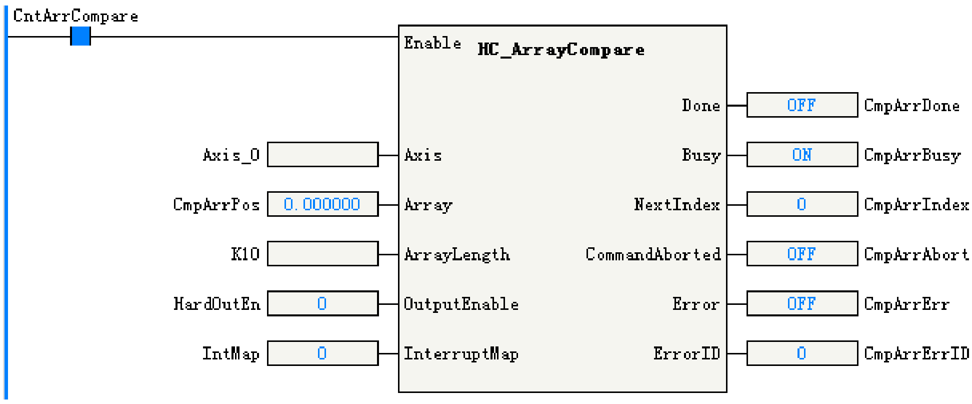
The output parameter NextIndex indicates the index of the next comparison point, that is, the number of completed comparison points.

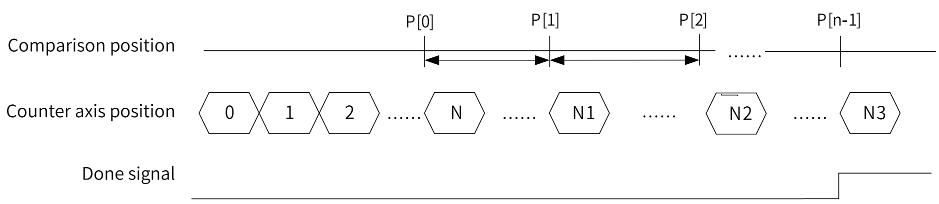
Timing Diagram

The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1) and three positions are compared (ArrayLength = 3).



Instruction Example





3.12.20 HC_StepCompare

This instruction continuously detects whether the counter count value reaches the continuous sequence with specified range and step.

HC_StepCompare – High-speed counter step comparison

Graphic Block

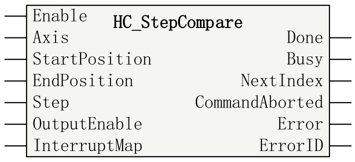
Instruction	Name	LD Expression	LiteST Expression
HC_StepCompare	High-speed counter step comparison		HC_StepCompare(Enable := ???, Axis := ???, StartPosition := ???, EndPosition := ???, Step := ???, OutputEnable := , InterruptMap := , Done => , Busy => , NextIndex => ,

Table 3–307 Instruction format

16-bit Instruction	-					
32-bit Instruction	HC_StepCompare: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Axis	Axis name/ID, which specifies the local encoder axis to be operated	No	-	0 to 32767	INT
S2	StartPosition	Start position (unit: Unit)	No	-	-	REAL
S3	EndPosition	End position (unit: Unit)	No	-	-	REAL
S4	Step	Step (unit: Unit)	No	-	Positive number	REAL
S5	OutputEnable	Hardware output enable 0: Disabled 1: Enabled	Yes	0	0 to 1	INT
S6	InterruptMap	Interrupt generation and association when the comparand and count value match 0: No interrupt is generated. 1: Associate with comparison interrupt 1. ... 16: Associate with comparison interrupt 16.	Yes	0	0 to 16	INT
D1	Done	0: Not completed 1: Completed	Yes	OFF	ON OFF	BOOL
D2	Busy	Executing	Yes	FLASE	ON OFF	BOOL
D3	NextIndex	Index of the next comparand	Yes	0	0 to 100	INT

D4	CommandAborted	Abortion of execution	Yes	OFF	ON OFF	BOOL
D5	Error	Error	Yes	OFF	ON OFF	BOOL
D6	ErrorID	Error code	Yes	0	*1	INT

Note

*1: For details, see "3.12.21 Error Codes" on page 602Error Codes.

Table 3–308 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	√	-
S3	-	-	-	√	√	√	√	√	-
S4	-	-	-	√	√	√	√	√	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	-	√	-	√	-	-	-	-
D2	√[1]	-	√	-	√	-	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	-	√	-	√	-	-	-	-
D5	√[1]	-	√	-	√	-	-	-	-
D6	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description

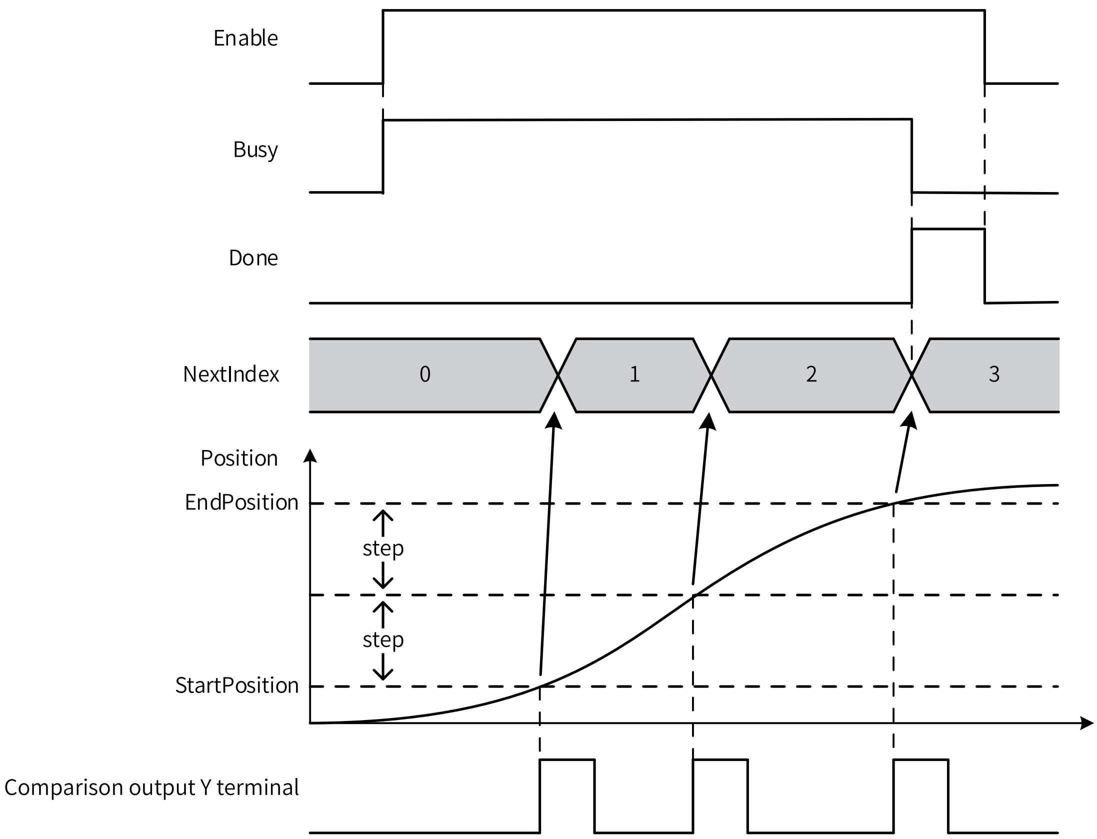
The HC_StepCompare instruction compares the counter axis position with consecutive positions with equal steps.

When the instruction flow is active, the counter axis position is compared with the position specified by StartPosition. When they are equal, the comparison position increases or decreases by a value specified by Step and then is compared with the counter axis position. After the last comparison position is compared, the Done signal is output.

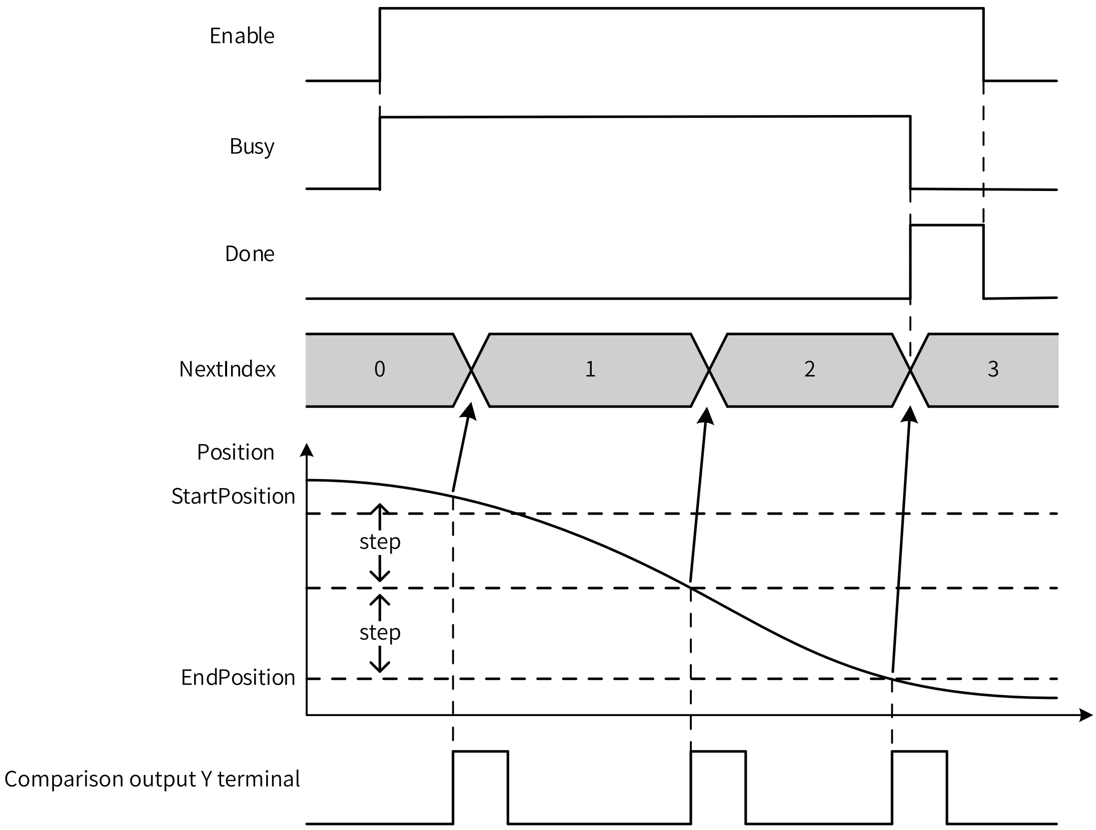
The output parameter NextIndex indicates the index of the next comparison point. The index starts from 0, that is, 0 indicates the first comparison point. Therefore, this index number is equal to the number of comparison points that have been compared.

Timing Diagram

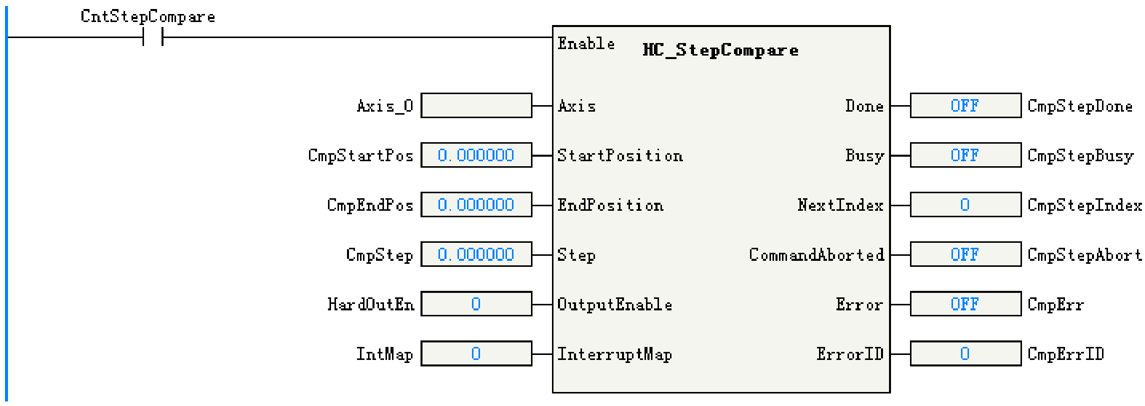
- The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1) and StartPosition is less than EndPosition.



- The timing diagram of the instruction is as follows when the hardware output is enabled (OutputEnable = 1) and StartPosition is greater than EndPosition.



Instruction Example



3.12.21 Error Codes

The following table lists the error codes of the high-speed counter function blocks.

Error Code	Description
0	No error occurs.
100	The axis ID is invalid./It's not the local encoder axis.
101	The imaginary axis mode is not supported.
102	The ENC_SetUnit instruction is configured after encoder counting is enabled. Ensure that the instruction is enabled before the encoder is enabled, and that PlusePerCycle is set properly.
103	DisPerCycle is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.

Error Code	Description
104	Numerator is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
105	Denotinator is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
106	Failed to set the gear ratio.
107	The ENC_SetLineRotationMode instruction is configured after encoder counting is enabled. Ensure that the instruction is enabled before the encoder is enabled, and that LineRotateMode is set properly.
108	PLimit is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
109	NLimit is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
110	PLimit and NLimit are set incorrectly in the instruction. Ensure that the parameter values are within the permissible range.
111	Rotation is set incorrectly in the instruction. Ensure that the parameter value is within the permissible range.
113	The current parameters do not meet the conditions for enabling software limiting.
114	Failed to set the linear/rotary mode.
200	An invert parameter input error occurs.
201	A trigger mode parameter input error occurs.
202	A trigger edge parameter input error occurs. The trigger edge is invalid or the X input is not configured.
203	FirstPosition is set incorrectly in the instruction.
204	LastPosition is set incorrectly in the instruction.
205	Failed to execute the instruction because the parameters in the probe instruction are set incorrectly.
206	Failed to set the preset position due to a parameter exception.
300	A probe ID parameter input error occurs.
301	An output enable parameter input error occurs. The output enable signal is invalid or the Y output is not configured.
400	An interrupt mapping parameter input error occurs.
500	A preset position parameter input error occurs.
501	A comparison position parameter input error occurs.
502	A start position parameter input error occurs.
503	An end position parameter input error occurs.
504	A step parameter input error occurs.
600	An array length parameter input error occurs or the number of positions for step comparison exceeds 100.
1000	Counting exceeded the lower limit.
1001	Counting exceeded the upper limit.

The following table lists the error codes of the bus encoder axis instructions.

Error Code	Cause	Solution
9701	The encoder axis instruction failed to request the memory.	1. Check whether the PLC memory runs out. 2. Contact the manufacturer.
9702	1. The encoder axis type is incorrect. 2. The requested encoder axis does not exist. 3. The instruction is not supported in offline commissioning.	This instruction does not support the set axis type. Check whether the axis type setting is incorrect. 2. The instruction is not supported during offline commissioning.
9703	Failed to configure the axis.	Check whether the board software and the software tool match.
9704	Counter operation command is not configured in I/O mapping of the encoder axis.	Configure Counter operation command in I/O mapping of the encoder axis.
9705	Counter status is not configured in I/O mapping of the encoder axis.	Configure Counter status in I/O mapping of the encoder axis.
9706	Encoder present position is not configured in I/O mapping of the encoder axis.	Configure Encoder present position in I/O mapping of the encoder axis.
9707	Pulse rate is not configured in I/O mapping of the encoder axis.	Configure Pulse rate in I/O mapping of the encoder axis.
9708	The positive limit of the encoder axis is not greater than the negative limit.	Ensure that the positive limit of the encoder axis is greater than the negative limit.
9709	The positive limit of the encoder axis is greater than 2147483647 after being converted into the pulse unit.	Ensure that the positive limit of the encoder axis is less than or equal to 2147483647 after being converted into the pulse unit.
9710	The negative limit of the encoder axis is less than –2147483648 after being converted into the pulse unit.	Ensure that the negative limit of the encoder axis is greater than or equal to –2147483648 after being converted into the pulse unit.
9711	The revolution cycle of the encoder axis in ring mode is greater than 2147483647 after being converted into the pulse unit.	Ensure that the revolution cycle of the encoder axis in ring mode is less than or equal to 2147483647 after being converted into the pulse unit.
9712	The encoder axis is changed while the ENC_Counter instruction is still active.	Do not change the encoder axis while the ENC_Counter instruction is still active.
9713	The GR10-2HCE module is faulty.	Check the fault code object dictionary of the GR10-2HCE module and troubleshoot the fault according to the fault code.
9714	Failed to reset the encoder axis fault.	1. The current fault of the encoder axis does not support reset. 2. The encoder shaft enters the faulty state immediately after the fault is reset. Check the axis fault codes and slave fault codes to further determine the fault type.
9715	The ENC_Reset instruction is called when the encoder axis is not faulty.	Do not call the ENC_Reset instruction when the encoder axis is not faulty.
9716	The value of TriggerMode of the ENC_Preset instruction is out of range.	Ensure that the parameter value is within the allowable range.
9717	The value of Position of the ENC_Preset instruction is greater than 9999999.	Set Position of the ENC_Preset instruction to a value less than or equal to 9999999.
9718	Physical output command is not configured in I/O mapping of the encoder axis.	Configure Physical output command in I/O mapping of the encoder axis.
9719	The preset position or comparison output position of the encoder axis instruction is greater than the positive limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than or equal to the positive limit.

Error Code	Cause	Solution
9720	The preset position or comparison output position of the encoder axis instruction is less than the negative limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is greater than or equal to the negative limit.
9721	The preset position or comparison output position of the encoder axis instruction is greater than 2147483647 or less than –2147483648 after being converted into the pulse unit.	Ensure that the preset position or comparison output position of the encoder axis instruction is between –2147483648 and +2147483647 after being converted into the pulse unit.
9722	The preset position or comparison output position of the encoder axis instruction is greater than or equal to the revolution cycle in ring mode.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than the revolution cycle in ring mode.
9723	The value of ProbeID of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9724	The value of TriggerEdge of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9725	The value of TerminalSource of the ENC_ TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9726	The value of TriggerMode of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9727	The probe status word is not associated in I/O mapping of the encoder axis.	Ensure that the probe status word is associated in I/O mapping of the encoder axis.
9728	The probe feedback position is not associated in I/ O mapping of the encoder axis.	Ensure that the probe feedback position is associated in I/O mapping of the encoder axis.
9729	The control word is not associated in I/O mapping of the encoder axis.	Ensure that the control word is associated in I/O mapping of the encoder axis.
9730	The probe window function of the encoder axis is enabled, but the start position of the window is not less than the end position.	Ensure that the start position of the probe window is less than the end position.
9731	The Xn0 terminal is not assigned with the probe function.	Assign the probe function to the Xn0 terminal.
9732	The Xn1 terminal is not assigned with the probe function.	Assign the probe function to the Xn1 terminal.
9733	The instruction is not supported by the local encoder axis.	Note that this instruction applies only to the bus encoder axis.
9734	The instruction is not supported by the bus encoder axis.	Note that this instruction applies only to the local encoder axis.
9742	Compare mode is not configured in I/O mapping of the encoder axis.	Configure Compare mode in I/O mapping of the encoder axis.
9743	Compare pulse/time is not configured in I/O mapping of the encoder axis.	Configure Compare pulse/time in I/O mapping of the encoder axis.
9744	Compare size/step is not configured in I/O mapping of the encoder axis.	Configure Compare size/step in I/O mapping of the encoder axis.
9745	Compare point value 1 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 1 in I/O mapping of the encoder axis.
9746	Compare point value 2 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 2 in I/O mapping of the encoder axis.
9747	Physical output status' is not configured in I/O mapping of the encoder axis.	Configure Physical output status in I/O mapping of the encoder axis.
9748	Compare error code is not configured in I/O mapping of the encoder axis.	Configure Compare error code in I/O mapping of the encoder axis.
9749	Current compare number/position is not configured in I/O mapping of the encoder axis.	Configure Current compare number/position in I/O mapping of the encoder axis.

Error Code	Cause	Solution
9750	Failed to obtain the start address of the array of the single-axis array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the background and board software match. 3. Check whether the array of the instruction is out of bounds.
9751	Failed to obtain the start address of the axis group of the axis group array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the background and board software match. 3. Check whether the array of the instruction is out of bounds.
9752	The bus encoder axis is not associated with any slave.	Associate the bus encoder axis with a slave.
9753	The x-axis and y-axis of the axis group array comparison instruction are not associated with the same slave.	Associate the x-axis and y-axis of the axis group comparison output instruction with the same slave.
9754	The x-axis of the axis group array comparison instruction is not associated with the first channel of the slave.	Associate the x-axis of the axis group comparison output instruction with the first channel of the slave.
9755	The y-axis of the axis group array comparison instruction is not associated with the second channel of the slave.	Associate the y-axis of the axis group comparison output instruction with the second channel of the slave.
9756	The Yn0 terminal is not assigned with the one-dimensional comparison output function.	Assign the one-dimensional comparison output function to the Yn0 output terminal corresponding to the channel.
9757	The absolute value of the start value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9758	The absolute value of the end value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9759	The absolute value of the step of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9760	The absolute value of Parameter of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9761	The value of Mode of the encoder axis comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9762	The time for time control of the encoder axis comparison output is out of range.	Ensure that the parameter value is within the allowable range.
9763	The step of the encoder axis step comparison output instruction is 0.	Set the step of the step comparison output instruction to a value other than 0.
9764	The start position of the step comparison output instruction of the encoder axis is equal to the end position.	Ensure that the start position of the step comparison output instruction is not equal to the end position.
9765	The start position of the step comparison output instruction of the encoder axis is greater than the end position, but the step is positive.	Set the step to a negative value.
9766	The start position of the step comparison output instruction of the encoder axis is less than the end position, but the step is negative.	Set the step to a positive value.

Error Code	Cause	Solution
9767	The value of Size of the encoder axis array comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9768	The absolute value of the target position in the encoder axis array comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9769	The axis is performing one-dimensional comparison output and must not be aborted by a two-dimensional comparison output instruction.	Wait for the one-dimensional comparison output to complete or stop the one-dimensional comparison output before executing the two-dimensional comparison output instruction.
9770	The EtherCAT slave is disconnected during operation.	Check whether the EtherCAT slave is disconnected during operation.
9771	The bus encoder axis is in offline commissioning mode.	The bus encoder axis does not support the offline commissioning mode.
9772	The DI terminal is not assigned with the preset position function.	Assign the preset position function to the DI terminal before calling the preset position instruction.
9773	The value of Parameter in the comparison instruction is out of range when the pulse output mode is selected.	Do not set Parameter to 0 or a negative value when the pulse output mode is selected in the comparison instruction.
9774	The 2HCE module fails when the comparison output instruction is called.	1. Ensure that the input parameters are within the allowable range. 2. Check whether I/O mapping of the encoder axis is manually modified and whether it meets the I/O mapping configuration requirements of the comparison output instruction.
9775	The set position in ring mode is less than 0.	Set the position in ring mode to a value greater than or equal to 0.
9776	The Yn0 terminal is not assigned with the two-dimensional comparison output function.	Assign the two-dimensional comparison output function to the Yn0 output terminal corresponding to the channel.
9777	The axis is performing two-dimensional comparison output and cannot be aborted by a one-dimensional comparison output instruction.	Wait for the two-dimensional comparison output to complete or stop the two-dimensional comparison output before calling the one-dimensional comparison output instruction.

3.13 Timer Instructions

3.13.1 Timer Instruction Parameters

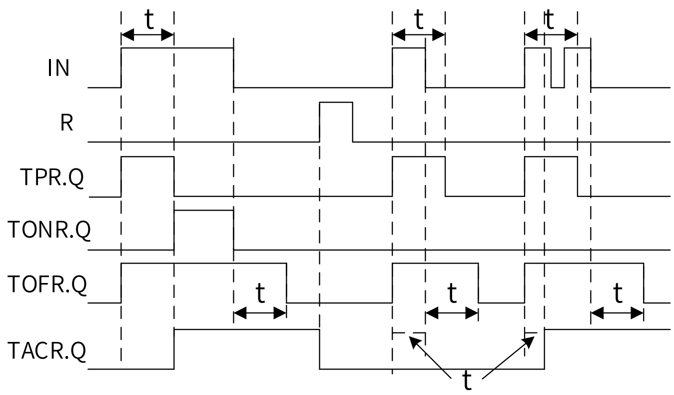
The PLC supports four types of timers: pulse timer (TPR), on-delay timer (TONR), off-delay timer (TOFR), and accumulating timer.

The time base of the timers is 1 ms, and the timer count value and state are updated when the timer instruction is executed. The program supports a maximum of 4096 timer instructions. The instruction parameters of these four types of timers are the same, which are listed as follows:

Table 3–309 Timer instruction parameters

Parameter	Definition	Data Type	Description
IN	Instruction execution input	/	Start input
PT	Input variable	DINT	Delay time
R	Input variable	BOOL	Reset input
Q	Output variable	BOOL	Timer output
ET	Output variable	DINT	Current timing time

Timer timing



3.13.2 Instruction List

The following table lists the timer instructions.

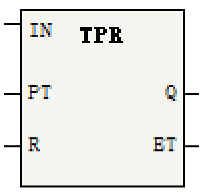
Table 3–310 Timer instruction list

Instruction Category	Instruction	Function
Timer instruction	TPR	Pulse timer
	TONR	On-delay timer
	TOFR	Off-delay timer
	TACR	Accumulating timer

3.13.3 TPR

TPR – Pulse timer

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TPR	Pulse timer		TPR(IN := ???,PT := ???,R := ,Q => ,ET =>);

16-bit Instruction	-			
32-bit Instruction	TPR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration (in ms)	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1 (in ms)	-	DINT

Note

*1: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

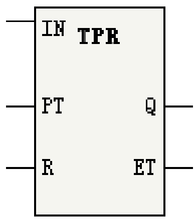
Table 3–311 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Parameter Description



When the IN input flow of the timer instruction changes from OFF to ON, the timer starts timing and the output Q turns ON. At this time, no matter how the IN input flow changes, Q remains ON for the time period specified by PT. When the timing duration reaches the time period specified by PT, Q changes to OFF.

During timing of the timer, ET outputs the current timing duration. After the timing duration reaches the value specified by PT, if the IN input flow is ON, the ET value is retained; if the IN input flow is OFF, the ET value becomes 0.

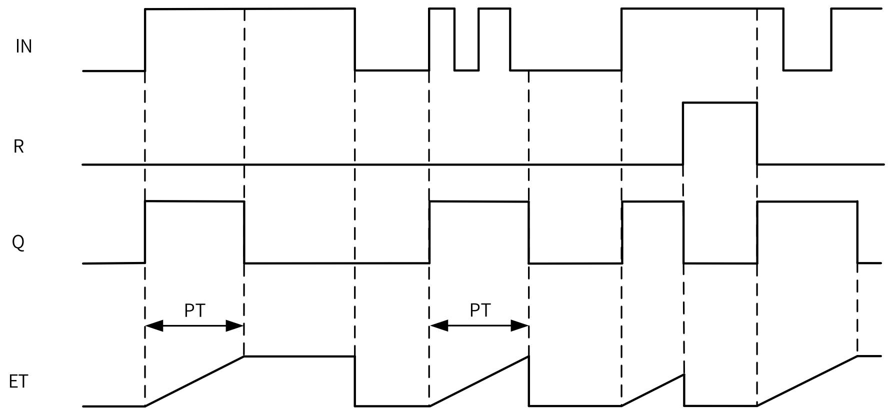
During timing, if the reset input R changes from OFF to ON, the timing duration of the TPR timer is reset to 0, and the output Q turns OFF. After the reset input R turns OFF, if the IN input flow is active, the timer resumes timing.

Description of parameters:

PT ranges from 0 to 2147483647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The timing diagram of the parameters IN, R, Q, and ET is as follows:



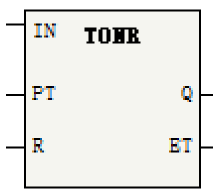
Note

The output parameter Q is updated in the PLC main task. Therefore, affected by the PLC scan cycle, it may not be output immediately when the time specified by PT elapses. The output may be delayed in varying degrees, with a maximum delay of one PLC scan cycle.

3.13.4 TONR

TONR – On-delay timer

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TONR	On-delay timer		TONR(IN := ???,PT := ???,R := ,Q => ,ET =>);

16-bit Instruction	-			
32-bit Instruction	TONR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration (in ms)	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1 (in ms)	-	DINT

Note

*1: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

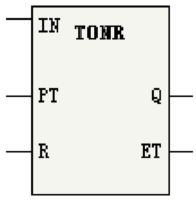
Table 3–312 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description



When the IN input flow of the timer instruction changes from OFF to ON, the timer starts timing and the output Q turns ON. During the period when the IN input flow remains ON, the running time of the timer is the time specified by PT. After the timing duration reaches the time period specified by PT, Q turns ON. During the timing process or after timing is completed, when the IN input flow changes to OFF, timing ends and Q turns OFF. During the period when the IN input flow remains OFF, Q remains OFF.

When the IN input flow is ON, ET outputs the current timing duration during timing of the timer, and the ET value is retained after the timing duration reaches the value specified by PT. When the IN input flow is OFF, the ET value becomes 0.

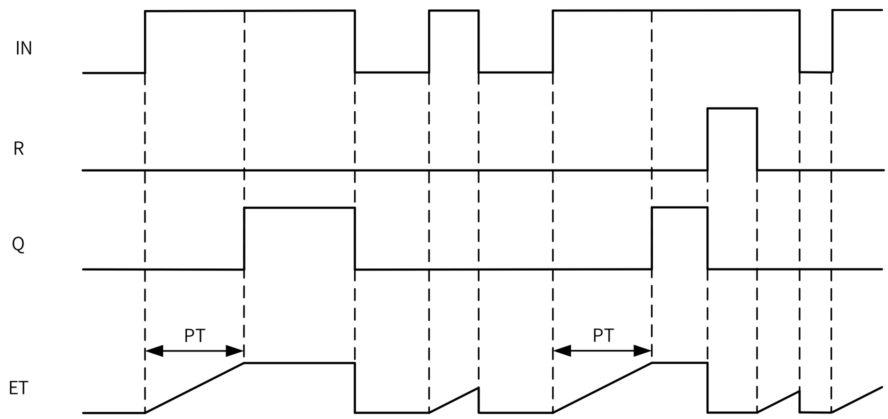
During timing, if the reset input R changes from OFF to ON, the timing duration of the TONR timer is reset to 0, and the output Q turns OFF. After the reset input R turns OFF, to resume timing, you need to set the IN input flow to ON again.

Description of parameters:

PT ranges from 0 to 2147483647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The timing diagram of the parameters IN, R, Q, and ET is as follows:



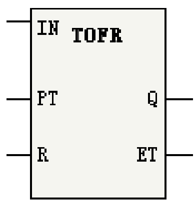
Note

The output parameter Q is updated in the PLC main task. Therefore, affected by the PLC scan cycle, it may not be output immediately when the time specified by PT elapses. The output may be delayed in varying degrees, with a maximum delay of one PLC scan cycle.

3.13.5 TOFR

TOFR – Off-delay timer

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TOFR	Off-delay timer		TOFR(IN := ???, PT := ???, R := , Q => , ET =>);

16-bit Instruction	-			
32-bit Instruction	TOFR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration (in ms)	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1 (in ms)	-	DINT

Note

*1: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

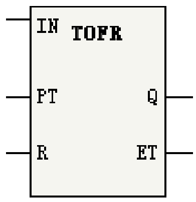
Table 3–313 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description



When the IN input of the timer instruction changes from OFF to ON, the timer starts timing and the output Q turns ON. When the IN input flow changes from ON to OFF, during the period when the IN input flow remains ON, the running time of the timer is the time specified by PT. After the timing duration reaches the time period specified by PT, Q turns OFF. During the period when the IN input flow remains OFF, Q remains OFF.

When the IN input flow is ON, the ET output becomes 0. When the IN input changes from ON to OFF, ET outputs the current timing duration during timing of the timer, and the ET value is retained after the timing duration reaches the value specified by PT.

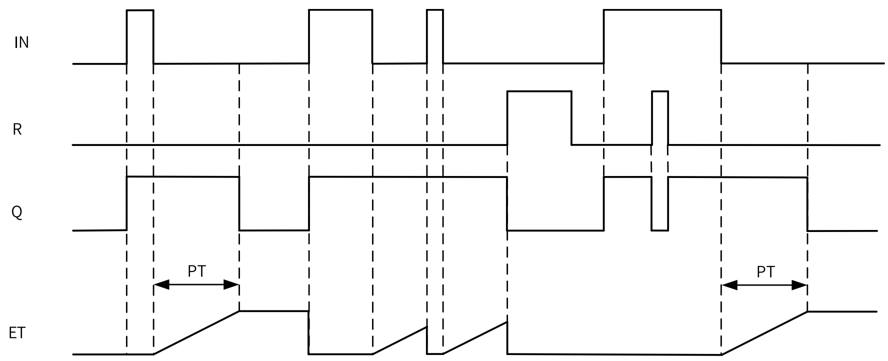
When the IN input flow is ON, if the reset input R changes from OFF to ON, the output Q turns OFF; if R resumes OFF, the output Q resumes ON. When the IN input flow changes from ON to OFF, if the reset input R changes from OFF to ON during the timing process or after timing is completed, the output Q turns OFF, and ET is reset to 0. After the reset input R turns OFF, to resume timing, you need to set the IN input flow OFF again.

Description of parameters:

PT ranges from 0 to 2147483647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The timing diagram of the parameters IN, R, Q, and ET is as follows:



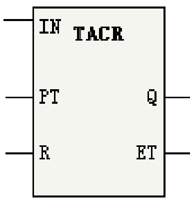
Note

The output parameter Q is updated in the PLC main task. Therefore, affected by the PLC scan cycle, it may not be output immediately when the time specified by PT elapses. The output may be delayed in varying degrees, with a maximum delay of one PLC scan cycle.

3.13.6 TACR

TACR – Accumulating timer

Graphic Block

Instruction	Name	LD Expression	LiteST Expression
TACR	Accumulating timer		TACR(IN := ???,PT := ???,R := ,Q => ,ET =>);

16-bit Instruction	-			
32-bit Instruction	TACR: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	PT	Preset timing duration (in ms)	-	DINT
S2	R	Reset*1	-	BOOL
D1	Q	Output result*1	-	BOOL
D2	ET	Elapsed time*1 (in ms)	-	DINT

Note

*1: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

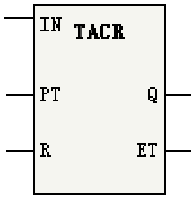
Table 3–314 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	√	√	√	-	-	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Function and Instruction Description



When the IN input flow of the timer instruction is ON, if the timer value has not reached the time period specified by PT, the timer continues to count, and the output Q is OFF; when the timing duration reaches the time period specified by PT, Q turns ON. During the timing process, if IN changes from ON to OFF, the timing duration is retained. When IN turns ON again, the timer starts counting from the current retained value. After the time specified by PT is reached, Q becomes ON.

When the IN input flow is ON, ET outputs the current timing value. After the timing duration reaches the time period specified by PT, the ET value is retained. When the IN input flow turns OFF, ET remains unchanged.

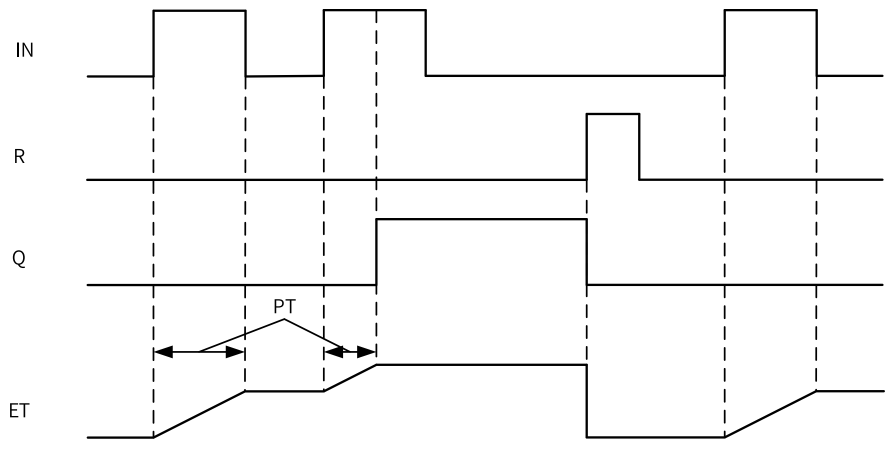
During the timing process or after timing is completed, if the reset input R changes from OFF to ON, the output Q turns OFF, and ET is reset to 0. After the reset input R turns OFF, to resume timing, you need to set the IN input flow OFF again.

Description of parameters:

PT ranges from 0 to 2147483647 ms (about 24 days). If the value of PT is less than or equal to 0, it is considered 0.

Timing Diagram

The timing diagram of the parameters IN, R, Q, and ET is as follows:



Note

The output parameter Q is updated in the PLC main task. Therefore, affected by the PLC scan cycle, it may not be output immediately when the time specified by PT elapses. The output may be delayed in varying degrees, with a maximum delay of one PLC scan cycle.

3.14 Pointer instruction

3.14.1 Instruction List

The following table lists the pointer instructions.

Instruction Category	Instruction	Function
Pointer instruction	PTGET	Pointer variable assignment
	PTINC	Pointer variable address incremented by 1
	PTDEC	Pointer variable address decremented by 1
	PTADD	Pointer variable address addition
	PTSUB	Pointer variable address subtraction
	PTSET	Pointer variable assignment
	PTMOV	Pointer variable mutual assignment
	PTLD>	Pointer variable contact comparison greater than
	PTLD>=	Pointer variable contact comparison greater than or equal to
	PTLD<=	Pointer variable contact comparison less than or equal to
	PTLD=	Pointer variable contact comparison equal to
	PTLD<>	Pointer variable contact comparison not equal to
	PTAND>	Pointer variable AND contact comparison greater than
	PTAND>=	Pointer variable AND contact comparison greater than or equal to
	PTAND<	Pointer variable AND contact comparison less than
	PTAND<=	Pointer variable AND contact comparison less than or equal to
	PTAND=	Pointer variable AND contact comparison equal to
	PTAND<>	Pointer variable AND contact comparison not equal to
	PTOR>	Pointer variable OR contact comparison greater than
	PTOR>=	Pointer variable OR contact comparison greater than or equal to
	PTOR<	Pointer variable OR contact comparison less than
	PTOR<=	Pointer variable OR contact comparison less than or equal to
	PTOR=	Pointer variable OR contact comparison equal to
	PTOR<>	Pointer variable OR contact comparison not equal to

3.14.2 PTGET

PTGET – Pointer variable assignment

16-bit instruction	PTGET (bit): Continuous execution/PTGETP: Pulse execution			
32-bit instruction	-			
16-bit instruction	PTGET (word): Continuous execution/PTGETP: Pulse execution			
32-bit instruction	PTGET (dword): Continuous execution/PTGETP: Pulse execution			
Operand	Name	Description	Range	Data Type
S1	Pointer Variable	Start address of the target	-	DINT
S2	Target variable	Start address of the target pointed to by the pointer variable	-	INT, DINT

Table 3–315 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	✓[1]	-	-	-
S2	√	√	√	√	√	-	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The PTGET instruction can obtain the address of a bit, word, or dword element or variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

PTGET PT5 D10: Point the pointer variable PT5 to the D10 element.

3.14.3 PTINC

PTINC – Pointer variable address incremented by 1

16-bit instruction	-			
32-bit instruction	PTINC: Continuous execution/PTINCP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Pointer Variable	Pointer Variable	-	DINT

Table 3–316 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	√[1]	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The increment is based on the unit of the variable pointed to by the pointer variable. The pointer points to the next element of the same type of the current variable. For example, the pointer points to the next bit element if the current variable is a bit element, it points to the next word variable if the current variable is a word variable, and it points to the next dword variable if the current variable is a dword variable.

- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. PTGET PT5 D10: Point the pointer PT5 to the D10 element.
2. PTINC PT5: Point the pointer PT5 to the next element, that is, D11.

3.14.4 PTDEC

PTDEC – Pointer variable address decremented by 1

16-bit instruction	-			
32-bit instruction	PTDEC: Continuous execution/PTDECP: Pulse execution			
Operand	Name	Description	Range	Data Type
D	Pointer Variable	Pointer Variable	-	DINT

Table 3–317 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D	-	-	-	-	-	√[1]	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The decrement is based on the unit of the variable pointed to by the pointer variable. The pointer points to the previous element of the same type of the current variable. For example, the pointer points to the previous bit element if the current variable is a bit element, it points to the previous word variable if the current variable is a word variable, and it points to the previous dword variable if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. PTGET PT5 D10: Point the pointer PT5 to the D10 element.

2. PTDEC PT5: Point the pointer PT5 to the previous element, that is, D9.

3.14.5 PTADD

PTADD – Pointer variable address addition

16-bit instruction	PTADD: Continuous execution/PTADDP: Pulse execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S1	Source pointer	Source pointer	-	DINT
S2	Offset address	Offset address	0 to 32767	INT
D	Target pointer	Target pointer	-	DINT

Table 3–318 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	-	-	-	√	√	-	√	-	-
D	-	-	-	-	-	√[1]	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The addition is based on the unit of the variable pointed to by the POINTER variable. The pointer points to the next n element of the current variable. For example, the pointer points to the next n bit element if the current variable is a bit element, it points to the next n word variable if the current variable is a word variable, and it points to the next n dword variable if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. PTGET PT5 D10: Point the pointer PT5 to the D10 element.
2. PTADD PT5 K4 PT5: When the pointer PT5 points to the D10 element, executing the PTADD instruction points PT5 to the position of the element pointed to by PT5 plus 4 elements, that is, D14.
3. PTADD PT5 K5 PT6: When the pointer PT5 points to the D10 element, executing the PTADD instruction points PT6 to the position of the element pointed to by PT5 plus 5 elements, that is, D15, while the element pointed to by PT5 remains unchanged, that is, D10.

3.14.6 PTSUB

PTSUB – Pointer variable address subtraction

16-bit instruction	PTSUB: Continuous execution/PTSUBP: Pulse execution			
32-bit instruction	-			
Operand	Name	Description	Range	Data Type
S1	Source pointer	Source pointer	-	DINT
S2	Offset address	Offset address	0 to 32767	INT
D	Target pointer	Target pointer	-	DINT

Table 3–319 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	-	-	-	√	√	-	√	-	-
D	-	-	-	-	-	√[1]	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The subtraction is based on the unit of the variable pointed to by the pointer variable. The pointer points to the previous n element of the current variable. For example, the pointer points to the previous n bit element if the current variable is a bit element, it points to the previous n word variable if the current variable is a word variable, and it points to the previous n dword variable if the current variable is a dword variable.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. PTGET PT5 D10: Point the pointer PT5 to the D10 element.
2. PTSUB PT5 K4 PT5: When the pointer PT5 points to the D10 element, executing the PTSUB instruction points PT5 to the position of the element pointed to by PT5 minus 4 elements, that is, D6.
3. PTSUB PT5 K5 PT6: When the pointer PT5 points to the D10 element, executing the PTSUB instruction points PT6 to the position of the element pointed to by PT5 minus 5 elements, that is, D5, while the element pointed to by PT5 remains unchanged, that is, D10.

3.14.7 PTSET

This instruction points the pointer variable to the target variable with specified variable length.

PTSET – Pointer variable assignment

16-bit Instruction	-			
32-bit Instruction	PTSET: Continuous execution/PTSETP: Pulse execution, 13 steps			
Operand	Name	Description	Range	Data Type
S1	Pointer element	-	-	-
S2	Target variable	Start address of the target pointed to by the pointer variable	-	BOOL, word, dword, FLT32
S3	Variable length	-	-	-

Table 3–320 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	√	√	√	√	√	-	-	-	-
S3	-	-	-	√	√	-	√	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

This instruction is a higher-order application and should be used with caution.

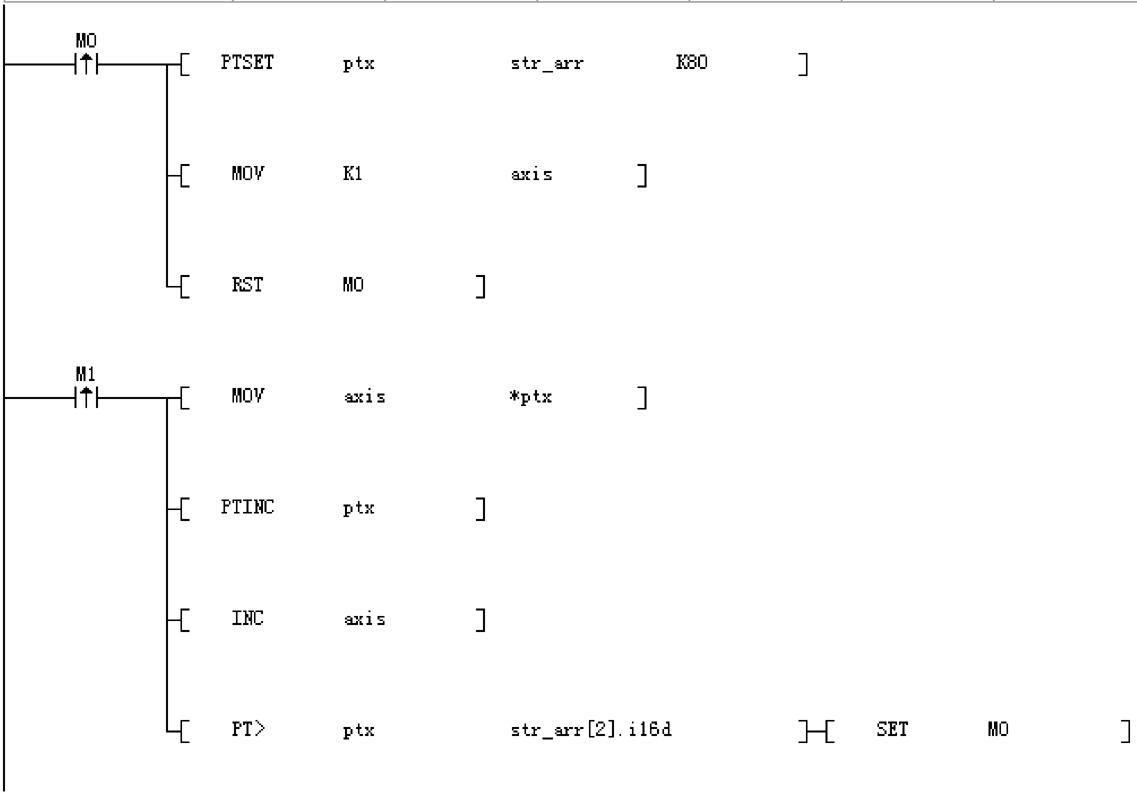
This instruction defines pointer variables of no specified type in the unit of bit. It can point pointer variables to various basic types, arrays, and structures.

This instruction can be used as transit for forced type conversion.

Instruction Example

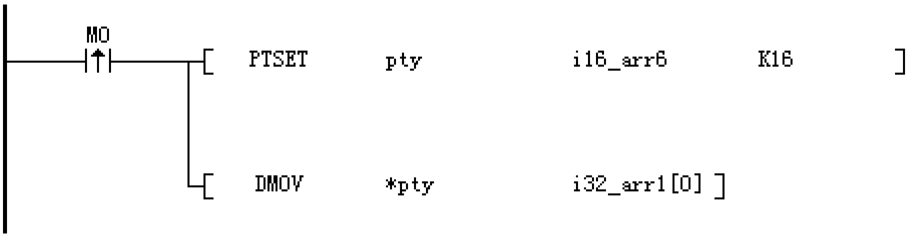
Pointers are used to assign the structure array. The pointer ptx points to the first element of the first structure of the structure array to assign values to the first element of each structure. The length of the variable is the length of each structure Stru, that is, 80 bits. See the following figure.

str_arr	Stru[3]	...	Not retained			nBitLen:240
str_arr[0]	Stru	...				
i16d	INT	0				
bd	BOOL	OFF				
f32_d	REAL	0.000000				
b2d	BOOL	OFF				
str_arr[1]	Stru	...				
i16d	INT	0				
bd	BOOL	OFF				
f32_d	REAL	0.000000				
b2d	BOOL	OFF				
str_arr[2]	Stru	...				
i16d	INT	0				
bd	BOOL	OFF				
f32_d	REAL	0.000000				
b2d	BOOL	OFF				
ptx	POINTER	NULL	Not retained			nBitLen:32
axis	INT	0	Not retained			nBitLen:16



After execution, the first element of each structure becomes 1, 2, and 3 respectively.

The following is an example of forced type conversion:



The execution result is equivalent to combining two i16_arr6 into a 32-bit number, that is:

The 16 bits of i16_arr6[0] is converted into the low-order 16 bits of i32arr1[0].

The 16 bits of i16_arr6[1] is converted into the high-order 16 bits of i32arr1[0].

3.14.8 PTMOV

PTMOV – Pointer variable mutual assignment

16-bit instruction	-			
32-bit instruction	PTMOV: Continuous execution/PTMOVP: Pulse execution			
Operand	Name	Description	Range	Data Type
S	Source pointer	Source pointer	-	DINT
D	Target pointer	Target pointer	-	DINT

Table 3–321 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S	-	-	-	-	-	✓[1]	-	-	-
D	-	-	-	-	-	✓[1]	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The PTMOV instruction is used to back up the address of the pointer variable, that is, it makes two pointer variables to point to the same address.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.
- The pulse-type instruction is recommended for level execution.

Instruction Example

1. PTGET PT5 D10: Point the pointer PT5 to the D10 element.
2. PTMOV PT5 PT6: Point the pointer PT6 to the position pointed to by PT5, that is, D10.

3.14.9 PT#

PT# – Pointer variable contact comparison

Pointer variable contact comparison instructions include the PTLD, PTAND, and PTOR instructions, and # indicates >, >=, <, <=, =, or <>.

16-bit instruction	-
32-bit instruction	PTLD>: Continuous execution
16-bit instruction	-
32-bit instruction	PTLD>=: Continuous execution

16-bit instruction	-			
32-bit instruction	PTLD<: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTLD<=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTLD=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTLD<>: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTAND>: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTAND>=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTAND<: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTAND<=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTAND=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTAND<>: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTOR>: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTOR>=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTOR<: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTOR<=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTOR=: Continuous execution			
16-bit instruction	-			
32-bit instruction	PTOR<>: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	Current pointer	Current pointer	-	-
S2	Compare object	Compare object	-	-

Note

For the PTLD*, PTAND*, and PTOR* instructions, the input is PT*, and the corresponding instructions are automatically generated at the background.

Table 3–322 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√[1]	-	-	-
S2	√	√	√	√	√	-	-	-	-

Note

[1] Only the pointer variable POINTER is supported.

Function and Instruction Description

- The PT# instruction compares the address of the element pointed to by the pointer with the address of the compare object.
- To use the pointer variable POINTER, you need to call the PTGET instruction for value assignment first. Otherwise, the pointer may point to an incorrect position, resulting in a system execution exception.

Instruction Example

1. PTGET PT5 D10: Point the pointer PT5 to the D10 element.
2. PT> PT5 D5: The output flow is ON. PT> PT5 D20: The output flow is OFF.

3.15 Communication Instructions

3.15.1 Instruction List

The following table lists the communication protocol instructions.

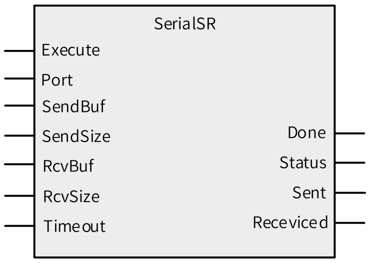
Instruction Category	Communication protocol	Instruction	Function
Communication protocol instruction	Serial port free protocol	SerialSR	Serial port free protocol transmission and reception
		SerialSend	Serial port free protocol transmission
		SerialRcv	Serial port free protocol reception
	Modbus protocol	MB_Master	Transmission and reception of serial Modbus protocol
		MB_Client	Transmission and reception of the Modbus TCP protocol
	TCP/ IP free protocol	TCP_Listen	TCP listening
		TCP_Accept	TCP connection request accept
		TCP_Connect	TCP connection request initiation
		TCP_Close	TCP connection close
		TCP_Send	TCP data transmission
		TCP_Receive	TCP data reception
	UDP/IP free protocol	UDP_Bind	UDP socket binding
		UDP_Send	UDP data transmission
		UDP_Receive	UDP data reception
	EtherCAT protocol	ETC_ReadParameter_CoE	Reading SDO parameters of EtherCAT slave
		ETC_WriteParameter_CoE	Writing SDO parameters of EtherCAT slave
		ETC_RestartMaster	Restarting EtherCAT master
	EtherNet/IP protocol	EIP_Generic_Service	Calling the "Generic" service
		EIP_Get_Attributes_All	Calling the "Get_Attributes_All" service
		EIP_Get_Attribute_Single	Calling the "Get_Attribute_Single" service
		EIP_Set_Attributes_All	Calling the "Set_Attributes_All" service
		EIP_Set_Attribute_Single	Calling the "Set_Attribute_Single" service
		EIP_Apply_Attributes	Calling the "Apply_Attributes" service
		EIP_NOP	Calling the "NOP" service
		EIP_Reset	Calling the "Reset" service
		EIP_Start	Calling the "Start" service
		EIP_Stop	Calling the "Stop" service

3.15.2 SerialSR

SerialSR – Serial port free protocol transmission and reception and free protocol cancellation

This instruction is used to implement free protocol communication through the serial port.

Graphic Block



16-bit Instruction	SerialSR: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Port	Port number	-	INT
S2	SendBuf	TX buffer	-	BYTE[]/INT[]
S3	SendSize	Number of bytes to transmit	0 to 256	INT
S4	RcvBuf	RX buffer	-	BYTE[]/INT[]
S5	RcvSize	Number of bytes to receive	0 to 256	INT
S6	Timeout	Reception timeout time (unit: ms[1])	-	INT
D1	Done	Completion flag[1]	-	BOOL
D2	Status	Instruction operation state[1]	-	INT
D3	Sent	Size of transmitted data[1]	-	INT
D4	Received	Size of received data[1]	-	INT

Note

[1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.

Table 3–323 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	-	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[2]	√	√	-	-	√	-	-	-
D2	-	-	-	√	√	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

Note

[2] The X element is not supported.

Function and Instruction Description

- Function description

  This instruction implements data transmission and reception of free protocols. After the instruction is triggered, data of the specified length is sent through the specified port, and after the transmission is completed, data of the specified length is received. The corresponding output is updated during the transmission and reception process.

  When the SerialSR instruction is in a receive waiting state, you can set the system variable _SerialSR.abort to a non-zero value to terminate the current receiving process. The parameter takes effect immediately after modification.

  You can set the system variables _SerialSR.startchar_en and _SerialSR.startchar[4] to set the receive start character function of the free protocol. The modified parameters take effect next time data receiving is triggered. When _SerialSR.startchar_en is set to a value that falls within the range of 1 to 4 and equals to or smaller than RcvSize, the receive start character function is enabled. The start character length is specified by _SerialSR.startchar_en, and the start character content is specified by _SerialSR.startchar[4].

  You can set the system variables _SerialSR.endchar_en and _SerialSR.endchar[4] to set the receive end character function of the free protocol. The modified parameters take effect next time data receiving is triggered. When _SerialSR.endchar_en is set to a value that falls within the range of 1 to 4, the receive end character function is enabled. The end character length is specified by _SerialSR. endchar_en, and the start character content is specified by _SerialSR.endchar[4].

  You can set the system variables_SerialSR.Bytetimeout_en and _SerialSR.Bytetimeout to set the receive byte timeout function of the free protocol. The modified parameters take effect next time data receiving is triggered. When _SerialSR.Bytetimeout_en is set to ON, the receive byte timeout function is enabled. The byte timer is specified by _SerialSR.Bytetimeout, and the minimum value is 1 (unit: ms*1). After the byte timeout function is enable and takes effect, the timer starts when the start byte/frame start character is received. If the idle time between received bytes is greater than the set time, the current receiving process is terminated and the Done signal is set.

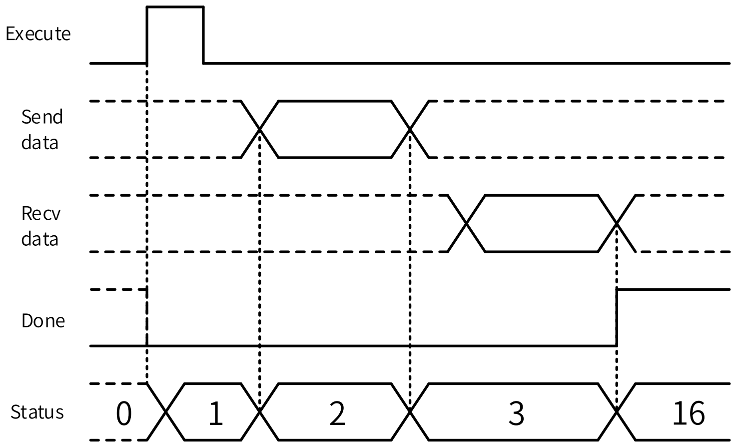
  Each serial port corresponds to an independent system variable: _ SerialSR corresponds to COM0, and _SerialSR1 to _SerialSR15 correspond to COM1 to COM15.

- Description

  - S1: Port number (corresponding to the serial port. You need to set it to an actual serial port number for communication.)
  - S2: TX buffer. It is recommended that you set the size of this buffer to a value greater than 128 word elements or 256 bytes.
  - S3: Number of bytes to transmit. The data length ranges from 1 byte to 256 bytes. When this parameter is set to 0, no data is sent.
  - S4: RX buffer. It is recommended that you set the size of this buffer to a value greater than 128 word elements or 256 bytes.
  - S5: Number of bytes to receive. The data length ranges from 1 byte to 256 bytes. When this parameter is set to 0, no data is received.
  - S6: Timeout time. If the specified time does not fall between 20 and 30000 (unit: ms[1]), it is automatically adjusted to the allowable range. If the specified time is -1, the receive status remains and never times out.

- D2: Operation state. 0: Empty; 1: Triggering; 2: Transmitting; 3: Receiving; 16: Completed; 32: Transmission error; 48: Reception error; 64: Other error.

Timing Diagram



Note

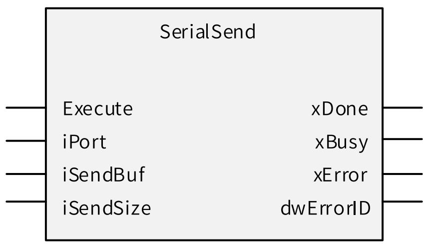
- The instruction is triggered and executed on the rising edge.
- The parameter D2 displays the operating state machine of the serial port, including normal and abnormal state values. The error codes of the instruction are not displayed in D2, but in the error table as errors of standard instructions.
- The timeout time refers to the total timeout duration for both transmission and reception.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.

3.15.3 SerialSend

SerialSend – Serial port free protocol transmission

This instruction is used to implement free protocol communication through the serial port.

Graphic Block



16-bit Instruction	SerialSend: Triggered execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	iPort	Port number	0 to 15	INT
S2	iSendBuf	TX buffer	-	BYTE[]/INT[]
S3	iSendSize	Number of bytes to transmit	0 to 256	INT
D1	xDone	Completion flag[1]	ON/OFF	BOOL

D2	xBusy	Executing[1]	ON/OFF	BOOL
D3	xError	Error flag[1]	ON/OFF	BOOL
D4	dwErrorID	Error code[1]	-[2]	INT

Note

- [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.
- [2]: See "3.15.5 Error Codes of Serial Port Free Protocol Communication Instructions" on page 635.

Table 3–324 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[3]	√	√	-	-	√	-	-	-
D2	√[3]	√	√	-	-	√	-	-	-
D3	√[3]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

Note

[3] The X element is not supported.

Function and Instruction Description

- Function description

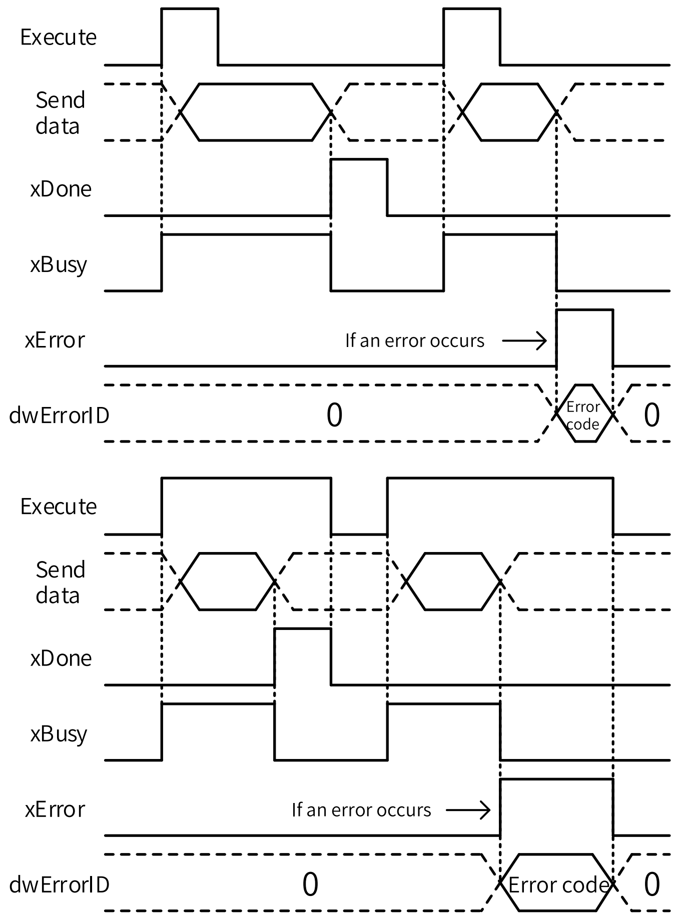
  This instruction implements data transmission of free protocols. After the instruction is triggered, data of the specified length is sent through the specified port. The corresponding output is updated during the transmission process.

- Description

  S1: Port number (corresponding to the serial port. You need to set it to an actual serial port number for communication.)

  S2: TX buffer. It is recommended that you set the size of this buffer to a value greater than 128 word elements or 256 bytes.

Timing Diagram



Note

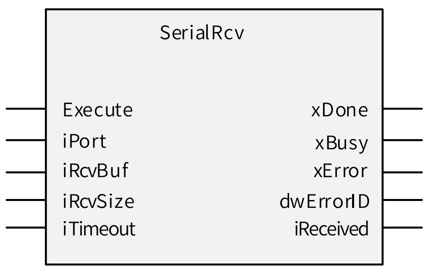
- The instruction is triggered and executed on the rising edge.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.

3.15.4 SerialRcv

SerialRcv – Serial port free protocol reception and free protocol cancellation

This instruction is used to implement free protocol communication through the serial port.

Graphic Block



16-bit Instruction	SerialRcv: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	iPort	Port number	0 to 15	INT
S2	iRcvBuf	Data receiving area	-	BYTE[]/INT[]
S3	iRcvSize	Maximum site of received data, in bytes	1 to 256	INT
S4	iTimeout	Reception timeout time[1]	-	INT
D1	xDone	Completion flag[1]	ON/OFF	BOOL
D2	xBusy	Executing[1]	ON/OFF	BOOL
D3	xError	Error flag[1]	ON/OFF	BOOL
D4	dwErrorID	Error code[1]	-[2]	INT
D5	iReceived	Size of received data[1]	0 to 256	INT

Note

- [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.
- [2]: See *"3.15.5 Error Codes of Serial Port Free Protocol Communication Instructions" on page 635*.

Table 3–325 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√ [3]	√	√	-	-	√	-	-	-
D2	√[3]	√	√	-	-	√	-	-	-
D3	√[3]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	√	-	-
D5	-	-	-	√	√	√	√	-	-

Note

[3] The X element is not supported.

Function and Instruction Description

- Function description

  This instruction implements data reception of free protocols. After the instruction is triggered, data of the specified length is received through the specified port. The corresponding output is updated during the reception process.

  When the SerialRcv instruction is in a receive waiting state, you can set the system variable _SerialSR.abort to a non-zero value to terminate the current receiving process. One cycle after the xDone signal is reset, the next receiving process starts. Parameter takes effect immediately after modification.

You can set the system variables _SerialSR.startchar_en and _SerialSR.startchar[4] to set the receive start character function of the free protocol. The modified parameters take effect next time data receiving is triggered. When _SerialSR.startchar_en is set to a value that falls within the range of 1 to 4 and equals to or smaller than iRcvSize, the receive start character function is enabled. The start character length is specified by _SerialSR.startchar_en, and the start character content is specified by _SerialSR.startchar[4].

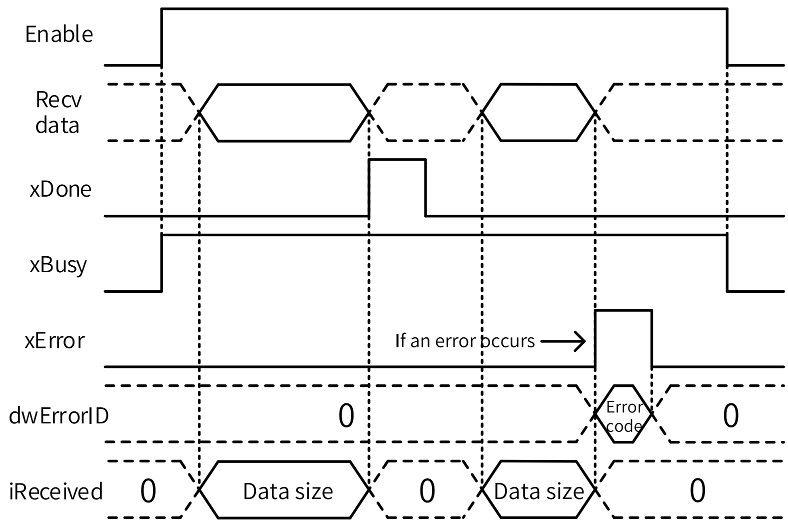
You can set the system variables _SerialSR.endchar_en and _SerialSR.endchar[4] to set the receive end character function of the free protocol. The modified parameters take effect next time data receiving is triggered. When _SerialSR.endchar_en is set to a value that falls within the range of 1 to 4, the receive end character function is enabled. The end character length is specified by _SerialSR. endchar_en, and the start character content is specified by _SerialSR.endchar[4].

You can set the system variables _SerialSR.Bytetimeout_en and _SerialSR.Bytetimeout to set the receive byte timeout function of the free protocol. The modified parameters take effect next time data receiving is triggered. When _SerialSR.Bytetimeout_en is set to ON, the receive byte timeout function is enabled. The byte timer is specified by _SerialSR.Bytetimeout, and the minimum value is 1 (unit: ms*1). After the byte timeout function is enable and takes effect, the timer starts when the start byte/frame start character is received. If the idle time between received bytes is greater than the set time, the current receiving process is terminated. One cycle after the xDone signal is reset, the next receiving process starts.

Each serial port corresponds to an independent system variable: _ SerialSR corresponds to COM0, and _SerialSR1 to _SerialSR15 correspond to COM1 to COM15.

- Description
  - S1: Port number (corresponding to the serial port. You need to set it to an actual serial port number for communication.)
  - S2: RX buffer. It is recommended that you set the size of this buffer to a value greater than 128 word elements or 256 bytes.
  - S4: Timeout time. If the specified time does not fall between 20 and 30000 (unit: ms[1]), it is automatically adjusted to the allowable range. If the specified time is -1, the receive status remains and never times out.
  - D1: Completion flag. When data of a specified length, a specified end character, or byte timeout signal is received, the current reception will end, and the completion flag will be set to one scan cycle.

Timing Diagram



Note

- The instruction is executed when the enable signal is at a high level.
- The reception timeout time refers to the frame timeout time, which is the total reception time.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.

3.15.5 Error Codes of Serial Port Free Protocol Communication Instructions

The following table lists the error codes of serial port free protocol communication instructions.

Table 3–326 Error codes of socket communication instructions

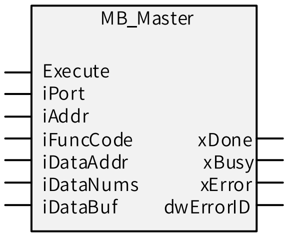
Error code	Description
5601	The port number is out of range.
5602	Protocol error
5603	Port conflict
5604	The sent data length is out of range or smaller than 0.
5605	TX buffer error
5606	The received data length is out of range, or is equal to or smaller than 0.
5607	RX buffer error
5620	Port number changed
5621	Reception timeout

3.15.6 MB_Master

MB_Master – Transmission and reception of serial Modbus protocol

This instruction is used to implement Modbus communication through the serial port.

Graphic Block



16-bit Instruction	-			
32-bit Instruction	MB_Master: Triggered execution			
Operand	Name	Description	Range	Data Type
S1	iPort	Port number	0 to 15	INT
S2	iAddr	Slave station ID	0 to 255	INT
S3	iFuncCode	Function code	1 to 6, 15, or 16	INT
S4	diDataAddr	Address of the slave station to be accessed	0 to 65535	DINT
S5	iDataNums	Bits or words to be accessed	-	INT
S6	iDataBuf	TX or RX buffer	-	BYTE[]/INT[]
D1	xDone	Completion flag[1]	ON/OFF	BOOL
D2	xBusy	Executing[1]	ON/OFF	BOOL
D3	xError	Error flag[1]	ON/OFF	BOOL
D4	dwErrorID	Error code[1]	-[2]	INT

Note

- [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.
- [2]: See "3.15.8 Fault Codes of Modbus Communication Instructions" on page 641.

Table 3–327 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	-	-	-
D1	√[3]	√	√	-	-	√	-	-	-
D2	√[3]	√	√	-	-	√	-	-	-
D3	√[3]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

Note

[3] The X element is not supported.

Function and Instruction Description

- Function description

  This instruction is used to implement Modbus communication through the serial port. After the instruction is triggered, the Modbus instruction is sent through the specified port. After the sending is completed, the master station waits for the response instruction from the slave station.

  Before using this instruction, you must set the communication protocol of the corresponding serial port to Modbus RTU or Modbus-ASC master station.

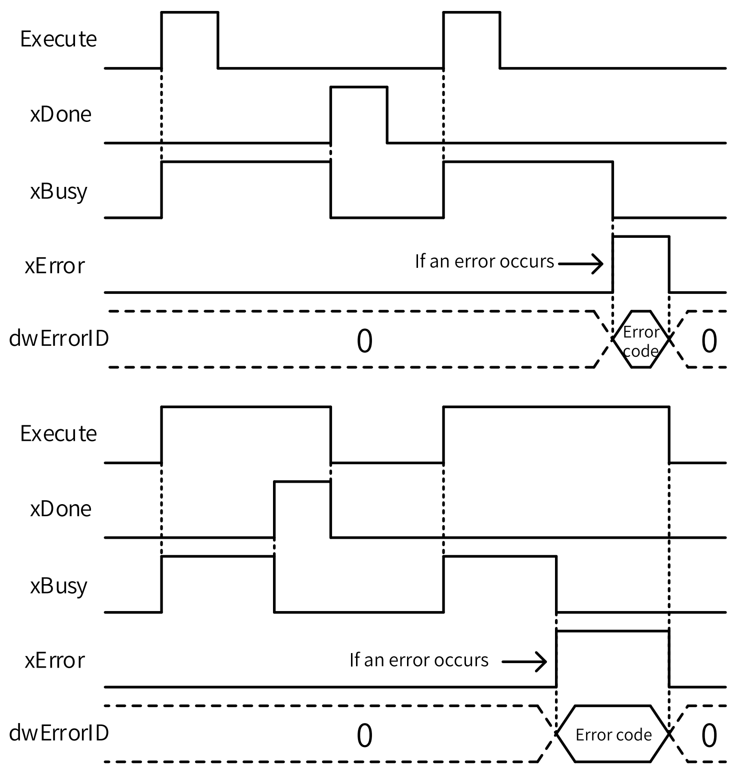
  The system variable _MbMstEx.RetryTimes specifies the number of retransmissions. Its value range is 1 to 15. If the set value is out of this valid range, it will automatically be adjusted to the allowed range.

  Each serial port corresponds to an independent system variable: _MbMstEx corresponds to COM0, and _MbMstEx1 to _MbMstEx15 correspond to COM1 to COM15.

  Before using the MB_Master instruction, you must set the communication protocol of the corresponding COM port to Modbus RTU or Modbus-ASC master station. The MC_Master instruction and Modbus serial port configuration table can be used simultaneously, and both share the timeout configuration (default: 500 ms).

- Description
  - S1: Port number (corresponding to the serial port. You need to set it to an actual serial port number for communication.)
  - S6: Buffer. You must set the buffer size to a value greater than 125 word elements or 250 bytes.

Timing Diagram



Note

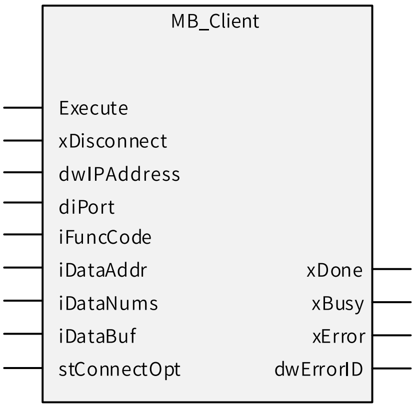
- The instruction is triggered and executed on the rising edge.
- The MB_Master instruction and the Modbus configuration table for the same serial port share the same timeout configuration, which is 500 ms by default.
- This instruction is executed only when the port is available, and only one instruction can be executed at the same time on the same port. This instruction is not executed if a port conflict or protocol setting error occurs. For details about the relevant error codes, see standard instruction errors.

3.15.7 MB_Client

MB_Client – Transmission and reception of the Modbus TCP protocol

This instruction is used to implement Modbus TCP communication.

Graphic Block



16-bit Instruction	-			
32-bit Instruction	MB_Client: Triggered execution			
Operand	Name	Description	Range	Data Type
S1	xDisconnect	Disconnect[1]	ON/OFF	BOOL
S2	dwIPAddress	IP address of the server	-	IP
S3	diPort	Port number	1 to 65535	DINT
S4	iFuncCode	Function code	1 to 6, 15, or 16	INT
S5	diDataAddr	Address of data to be accessed	0 to 65535	DINT
S6	iDataNums	Bits or words to be accessed	-	INT
S7	iDataBuf	TX or RX buffer	-	BYTE[]/INT[]
S8	stConnectOpt	Connection parameters and attributes	-	INT
D1	xDone	Completion flag[1]	ON/OFF	BOOL
D2	xBusy	Executing[1]	ON/OFF	BOOL
D3	xError	Error flag[1]	ON/OFF	BOOL
D4	dwErrorID	Error code[1]	-[2]	INT

Note

- [1]: The parameters of the instruction are not mandatory. If they are not specified, the default values are used or there is no output.
- [2]: See *"3.15.8 Fault Codes of Modbus Communication Instructions" on page 641*.

Table 3–328 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	√	√	√	-	-	√	-	-	-
S2	-	-	-	-	-	√	-	-	√
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	√	√	-	-
S7	-	-	-	√	√	√	-	-	-
S8	-	-	-	√	√	√	-	-	-
D1	√[3]	√	√	-	-	√	-	-	-
D2	√[3]	√	√	-	-	√	-	-	-
D3	√[3]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

Note

[3] The X element is not supported.

Function and Instruction Description

- Function description

  This instruction is used to implement client (master station) communication over Modbus TCP. After the instruction is triggered, an attempt is made to connect to the slave station based on the IP address and port number specified in the instruction. After the connection is set up, the Modbus instruction is sent. After the sending is completed, the master station waits for the response instruction from the slave station.

  The MB_Client instruction supports up to 31 connections. When the number of connections exceeds 31, the instruction will report an error and the original connections need to be released.

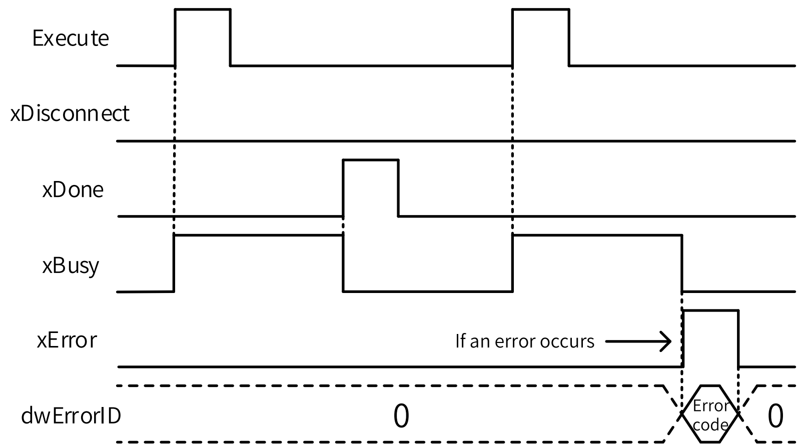
  The MB_Client instruction and Modbus TCP configuration table can be used simultaneously without affecting each other.

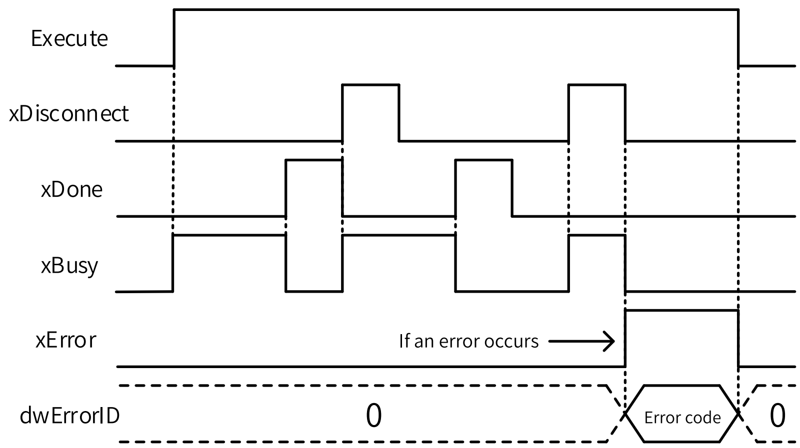
- Description

  - S1: Release the connection (When this parameter is set and the flow is enabled, the current connection is released.)
  - S7: Buffer. You must set the buffer size to a value greater than 125 word elements or 250 bytes.
  - S8: Connection parameters and attributes, such as the unit identifier, number of retransmissions, receive timeout. For details, see the table below.

Address	Data Type	Range	Default Value	Unit	Description
stConnectOpt	INT	0 to 1	-	-	Configuration enabling. 0: Use the default configuration, 1: Use the parameter configuration.
stConnectOpt+1	INT	0 to 255	255	-	Unit identifier
stConnectOpt+2	INT	1 to 15	1		Number of retransmissions
stConnectOpt+3	INT	100 to 10000	500	ms	Reception timeout time

Timing Diagram





Note

- The instruction is triggered and executed on the rising edge.
- The MB_Client instruction has a default receive timeout of 500 ms and a retry count of 1.
- This instruction is only executed when the connection is available, and only one MB_Client instruction can be executed for a connection at any given time. This instruction is not executed if a connection conflict occurs. For details about the relevant error codes, see standard instruction errors.

3.15.8 Fault Codes of Modbus Communication Instructions

The following table lists the fault codes of Modbus communication instructions.

Table 3–329 Fault codes of socket communication instructions

Fault code	Description
6000	Network connection failed
6001	Function code not supported
6002	Register/coil address out of range
6003	Improper data range
6004	Slave device fault
6128	Response station number and requested station number mismatch
6129	Response function code and requested function code mismatch
6130	Mismatched register/coil address in the response and request
6131	Mismatched response and request data
6240	Invalid mapping address in the configuration
6255	Slave station response timeout
6261	The port number is out of range.
6262	Protocol error
6263	Port conflict
6264	Incorrect slave station node
6280	Slave station disabled
6290	Network connection pool already full
6291	Network connection already occupied
6292	Incorrect response ID
6293	Incorrect received data length
6294	Non existing network connection

3.15.9 Connection-oriented Socket TCP Communication

The socket is a two-way communication interface. Hosts in the network transmits data through the interface provided by the socket.

The Ethernet socket interface is provided. By using sockets, users can easily implement communication between different devices with the TCP/IP network. The following table defines the socket structure type (_sSOCKET).

Member	Card Type	Read/Write	Parameter Function:
ID	DINT	Read-only	ID
Type	INT	Read-only	Socket type 1: TCP 2: UDP
LocalPort	DINT	Read-only	Local port
RemoteIP	DINT	Read-only	Remote IP address
RemotePort	DINT	Read-only	Remote port
Active	BOOL	Read-only	Active state
Connected	BOOL	Read-only	Connected state

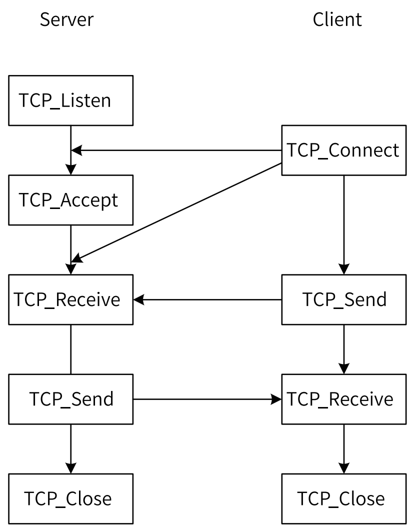
Member	Card Type	Read/Write	Parameter Function:
Listening	BOOL	Read-only	Listening state
Reserved0	BOOL[13]	Read-only	Reserved
Connections	INT	Read-only	Number of current connections
Reserved1	INT	Read-only	Reserved
Descriptor	DINT	Read-only	Reserved
ListeningSocket	DINT	Read-only	Reserved
Reserved2	DINT[2]	Read-only	Reserved

At present, AutoShop does not support system type custom variables, and structures are not supported as instruction input parameters. Therefore, the _sSOCKET variable is temporarily replaced by an INT[20] array.

The Transmission Control Protocol (TCP) is a connection-oriented, reliable, byte stream-based transport layer communication protocol.

An internetwork is very different from a single network because different parts of an internetwork can have vastly different topologies, bandwidths, latency, packet sizes, and other parameters. TCP is designed to dynamically adapt to these characteristics of the Internet and to demonstrate robustness in the face of various failures.

The process of a connection-oriented socket TCP communication interface is shown in the following figure.



3.15.10 TCP_Listen

TCP_Listen – TCP listening

The server must wait for the client's connection request. When working as a server, the local machine uses the TCP_Listen instruction to listen to connection requests from clients.

Graphic Block

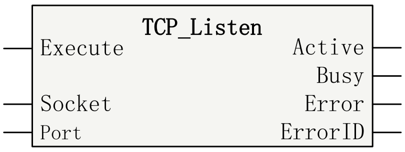


Table 3–330 Instruction format

16-bit Instruction	-					
32-bit Instruction	TCP_Listen: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Port	Local port to listen on*2	No	-	1 to 65535	DINT
D1	Active	Active state	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Error code	Yes	0	*3	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: System internal ports (23, 12939, and 12940) and the Modbus-TCP server port (502) cannot be used.
- *3: See *"3.15.23 Error Codes of Socket Communication Instructions" on page 662*.

Table 3–331 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

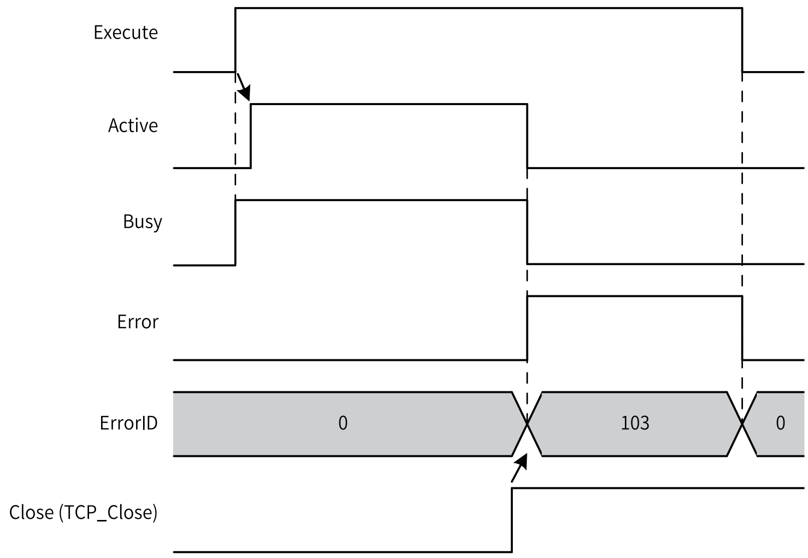
Note

[1] The X element is not supported.

Function and Instruction Description

The server must wait for the client's connection request. The TCP_Listen instruction is used to listen on the specified local port to wait for client requests. Upon receiving a connection request from the client, the server needs to use the TCP_Accept instruction to establish communication with the client.

Timing Diagram



3.15.11 TCP_Accept

TCP_Accept – TCP connection request accept

Upon receiving a connection request from the client, a server in listening state will put the client in the waiting queue. When working as a server, the local machine uses the TCP_Accept instruction to receive connection request from clients.

Graphic Block

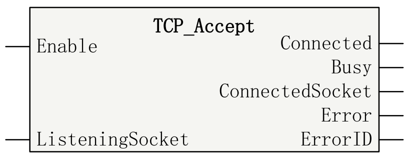


Table 3–332 Instruction format

16-bit Instruction	-					
32-bit Instruction	TCP_Accept: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	ListeningSocket	Listening socket*1	No	-	-	_sSocket
D1	Connected	Connected state	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	ConnectedSocket	Connection socket*1	No	-	-	_sSocket
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see *"3.15.23 Error Codes of Socket Communication Instructions" on page 662*.

Table 3–333 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	-	-	-	-
D1	√[1]	-	√	-	-	√	-	-	-
D2	√[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	-	-	-	-
D4	√[1]	-	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

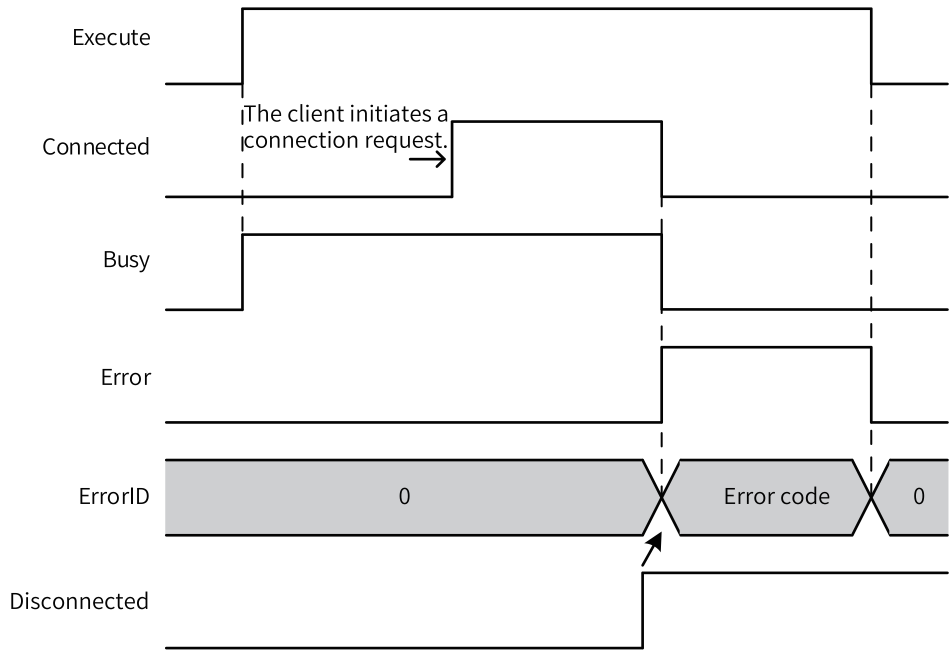
[1] The X element is not supported.

Function and Instruction Description

Upon receiving a connection request from the client, a server in listening state needs to use the TCP_Accept instruction to establish communication with the client. After communication is successfully established, the server can transmit or receive data by using TCP_Send or TCP_Receive.

The server can establish communication with multiple clients through the same local port by executing multiple TCP_Accept instructions.

Timing Diagram



3.15.12 TCP_Connect

TCP_Connect – TCP connection request initiation

To communication with the server, a client needs to initiate a connection request to the server. When working as a client, the local machine uses the TCP_Connect instruction to initiate a connection request.

Graphic Block

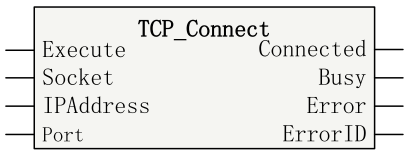


Table 3–334 Instruction format

16-bit Instruction	-					
32-bit Instruction	TCP_Connect: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	IPAddress	IP Address	No	-	-	DINT
S3	Port	port	No	-	1 to 65535	DINT
D1	Connected	Connected state	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Error code	Yes	0	*2	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see "3.15.23 Error Codes of Socket Communication Instructions" on page 662.

Table 3–335 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

Note

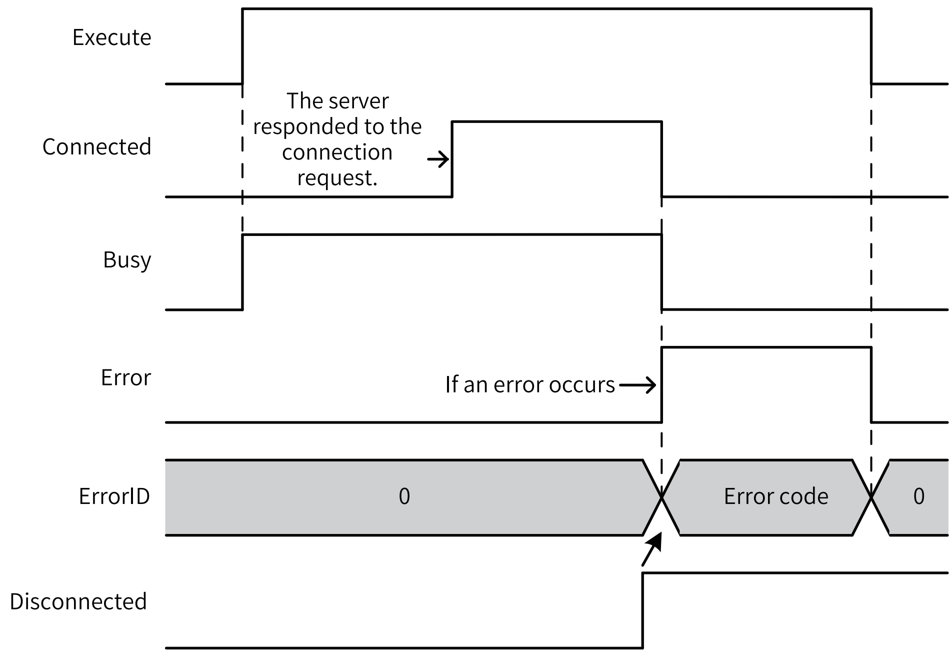
[1] The X element is not supported.

Function and Instruction Description

When the local machine works as a client and needs to communicate with the server, it executes the TCP_Connect instruction to connect to the specified port of the server. After the server accepts the connection request, the client can transmit or receive data by using TCP_Send or TCP_Receive.

After sending a connection request to the server by using the TCP_Connect instruction, the client waits at most 127 seconds. If the server does not respond, the connection fails.

Timing Diagram



3.15.13 TCP_Close

TCP_Close – TCP connection close

The TCP_Close instruction can be used to close the connection or listening after communication is completed.

Graphic Block

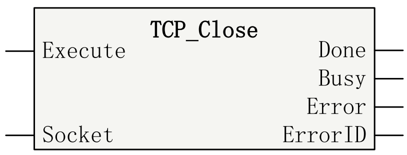


Table 3–336 Instruction format

16-bit Instruction	-					
32-bit Instruction	TCP_Close: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorID	Error code	Yes	0	*2	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see "3.15.23 Error Codes of Socket Communication Instructions" on page 662.

Table 3–337 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

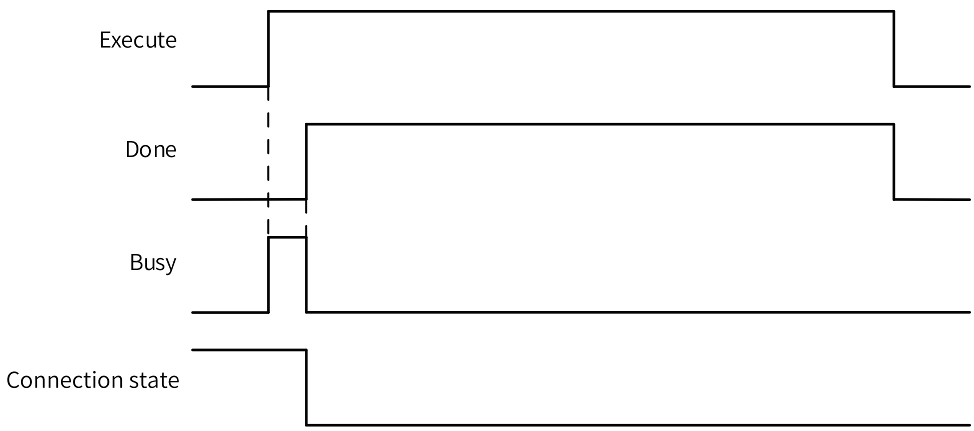
Note

[1] The X element is not supported.

Function and Instruction Description

The TCP_Close instruction can be used to close the connection, stop listening, or terminate the connecting socket after communication is completed.

Timing Diagram



3.15.14 TCP_Send

TCP_Send – TCP data transmission

After the connection between the server and the client is successfully established, data can be transmitted to the remote host by using the TCP_Send instruction.

Graphic Block

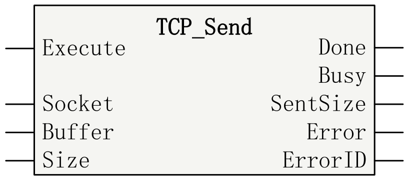


Table 3–338 Instruction format

16-bit Instruction	-					
32-bit Instruction	TCP_Send: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S3	Size	Data size	Yes	0	0 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	SentSize	Size of transmitted data	Yes	0	ON/OFF	INT
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see "3.15.23 Error Codes of Socket Communication Instructions" on page 662.

Table 3–339 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

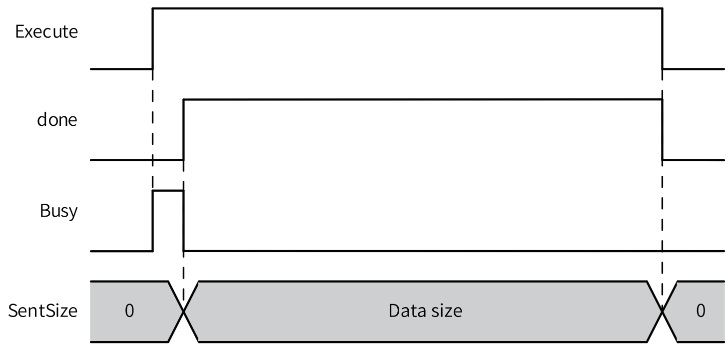
[1] The X element is not supported.

Function and Instruction Description

After the connection between the server and the client is successfully established, the local machine uses the TCP_Send instruction to send the data in the buffer with the specified length to the remote host.

The data size (size) must be less than or equal to the actual size of the data buffer (Buffer); otherwise, there is a risk of out-of-bounds data access.

Timing Diagram



NoteIf this instruction is triggered again when the data frame of the instruction is still being sent, an error is reported due to abnormal timing. To avoid this problem, it is recommended that the next transmission or instruction be triggered upon the state change of the Done or Error signal.

3.15.15 TCP_Receive

TCP_Receive – TCP data reception

After the connection between the server and the client is successfully established, message data transmitted by the remote host can be obtained from the specified socket by using the TCP_Receive instruction.

Graphic Block

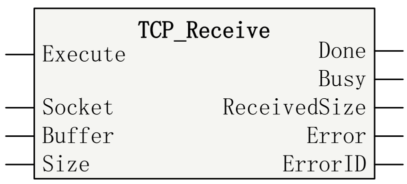


Table 3–340 Instruction format

16-bit Instruction	-					
32-bit Instruction	TCP_Receive: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket

S2	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S3	Size	Data size	No	-	1 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	ReceivedSize	Size of received data	Yes	0	0 to 32767	INT
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see "3.15.23 Error Codes of Socket Communication Instructions" on page 662.

Table 3–341 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

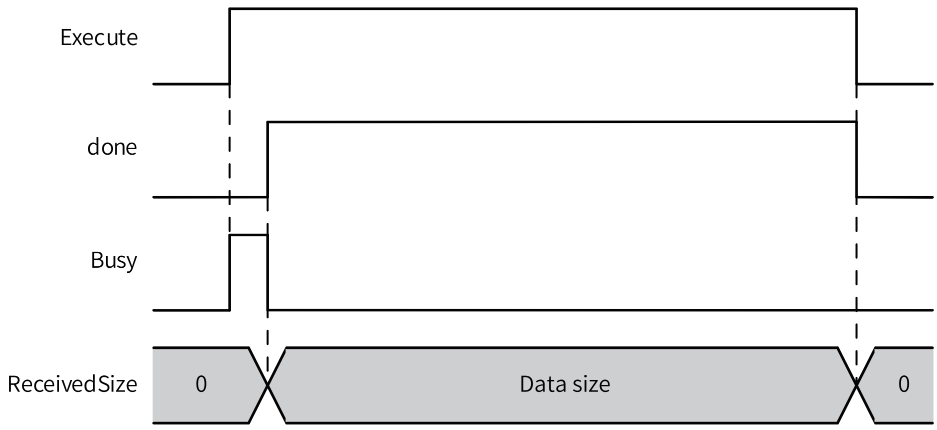
Function and Instruction Description

When the data size parameter (size) is 0, data is transmitted as character strings. That is, the data between the first byte and the terminator (excluding the terminator) in the data buffer (Buffer) is sent. The ASCII code of the terminator is 0.

The data size (size) must be less than or equal to the actual size of the data buffer (Buffer); otherwise, there is a risk of out-of-bounds data access.

After the connection between the server and the client is successfully established, message data transmitted by the remote host to the local machine will be stored in the socket buffer area. The TCP_Receive instruction is used to obtain the received message data from the specified socket buffer area.

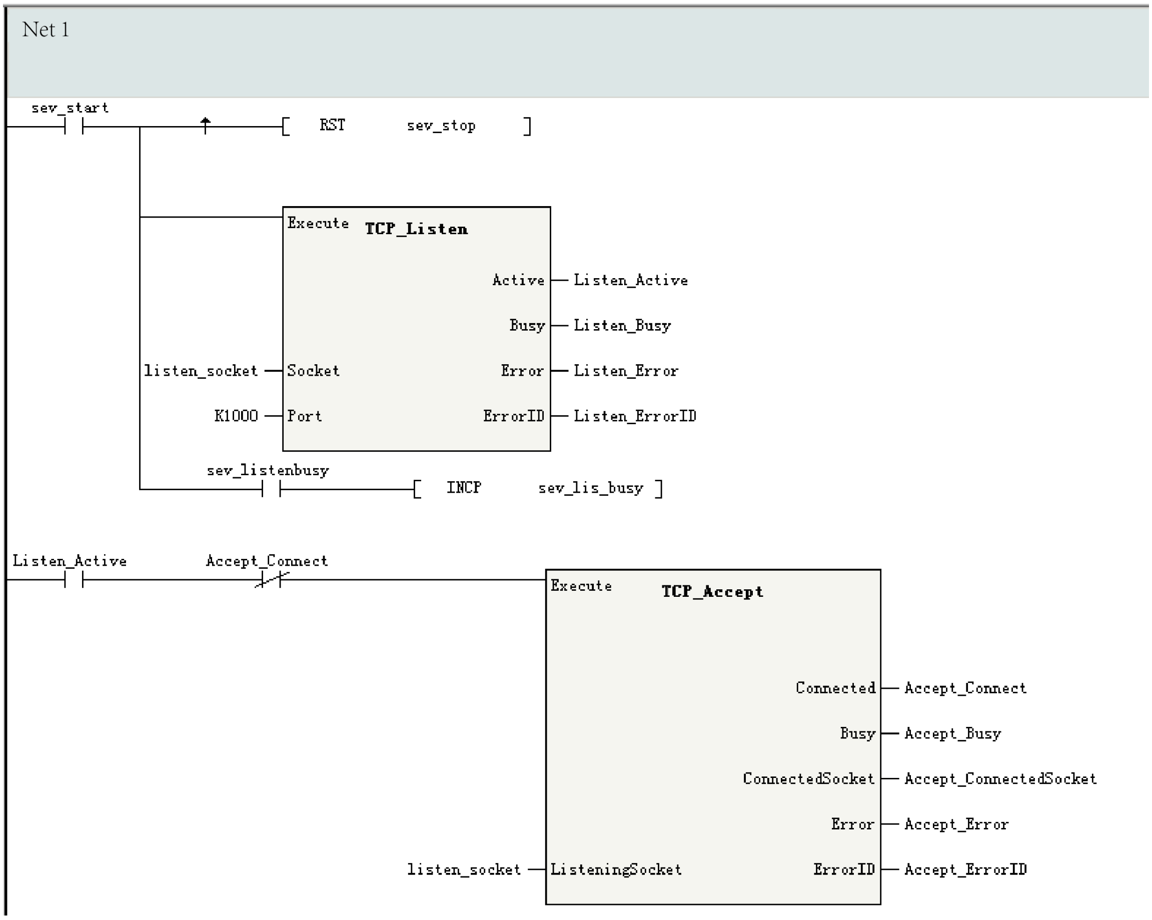
Timing Diagram

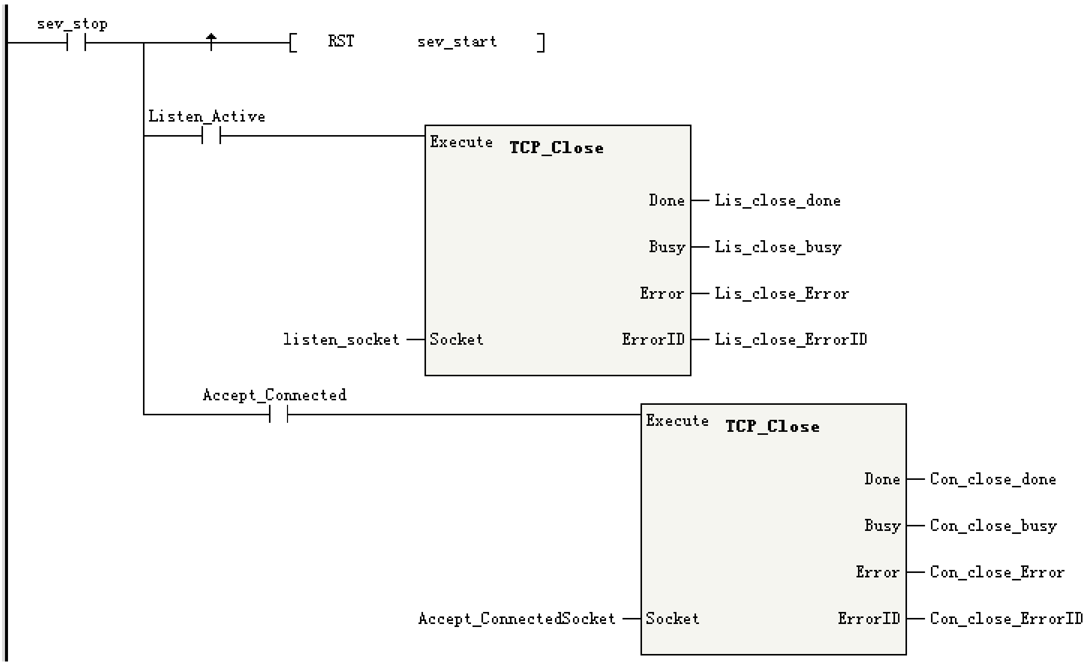


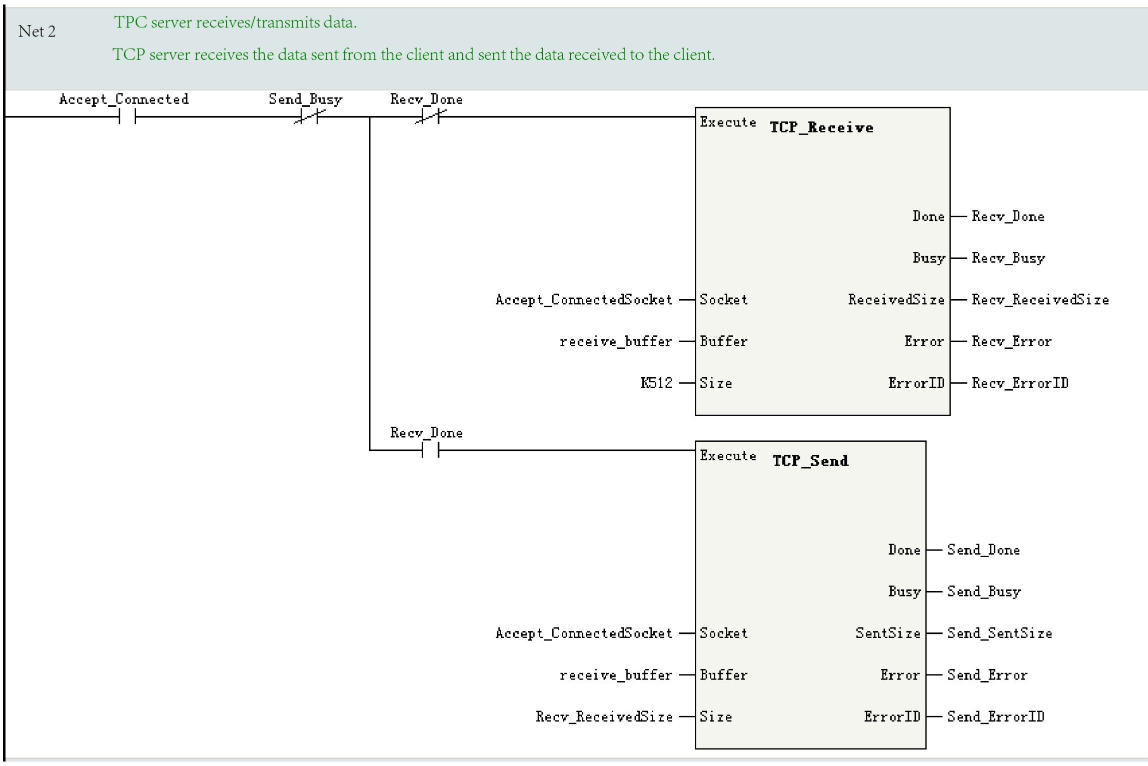
3.15.16 TCP Server Communication Instance

Working as a TCP server, the H5U listens on TCP port 1000 by using the TCP_Listen instruction and sends the data received from the client back to the client after accepting the connection request from the client.

The instance is programmed as follows:





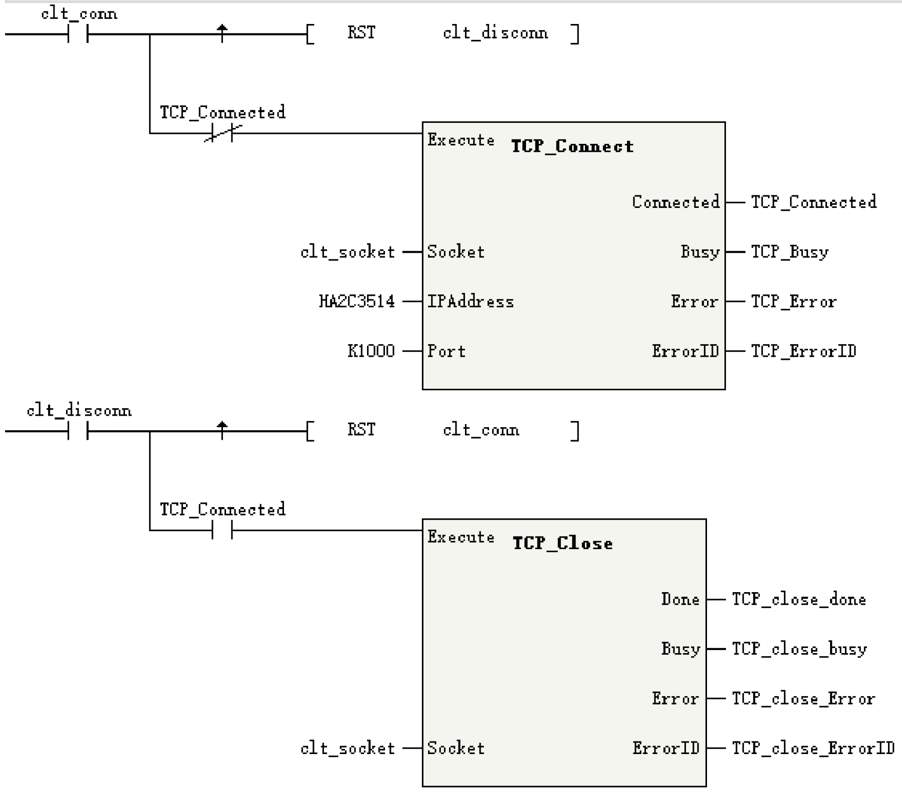


3.15.17 TCP Client Communication Instance

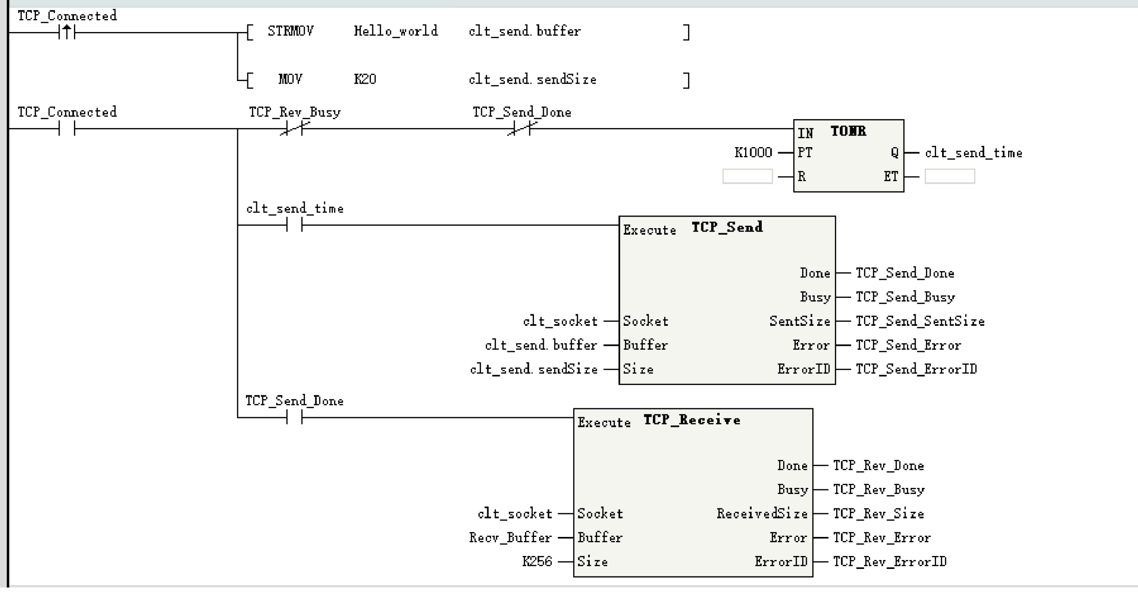
Working as a client, the H5U sends a connection request to the server at 10.44.53.20:1000. After the connection is established successfully, it sends "Hello!" to the server periodically (at an interval of 1 second) and stops sending "Hello!" after receiving any data sent by the server.
The instance is programmed as follows:

Net 1 TCP client connected/disconnected

TCP server address: 10.44.53.20 (HA2C3514 as hex equivalent)
Port No.: 1000



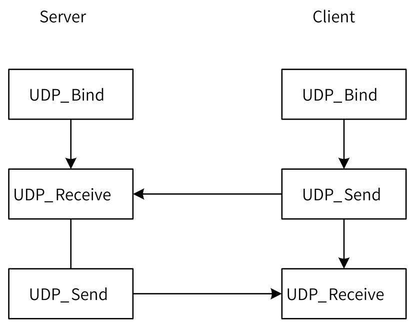
Net 2 TCP client transmits/receives data.
TCP client sends data to the server and receives data thereafter.



3.15.18 Connectionless Socket UDP Communication

User Datagram Protocol (UDP) is a connectionless transport layer protocol, which is mainly used in transmissions that do not require sequential arrival of packets. The check and sorting of packet transmission sequence is completed by the application layer. UDP provides simple and unreliable transaction oriented information transmission services.
UDP packets lack the QoS, sequence assurance, and flow control fields, resulting in poor reliability. However, because UDP protocol has fewer control options, it has low latency and high data transmission efficiency during data transmission. Therefore, UDP is suitable for applications with low reliability requirements or applications that can ensure reliability.

The following figure shows the UDP communication socket process.



3.15.19 UDP_Bind

UDP_Bind – UDP socket binding

Before data transmission or reception through UDP, you need to bind the socket to a local port. The UDP_Bind instruction is used to bind the socket to the specified UDP port.

Graphic Block

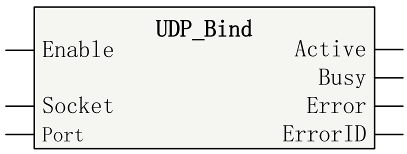


Table 3–342 Instruction format

16-bit Instruction	-					
32-bit Instruction	UDP_BIND: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Port	Port*2	Yes	0	0 to 65535	DINT
D1	Active	Active state	Yes	FASLE	ON/OFF	BOOL
D2	Busy	Executing	Yes	FASLE	ON/OFF	BOOL
D3	Error	Function block error flag	Yes	FASLE	ON/OFF	BOOL
D4	ErrorID	Error code*3	Yes	0	-	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: System internal ports (12939 and 12940) cannot be used.
- *3: For details, see *"3.15.23 Error Codes of Socket Communication Instructions" on page 662*.

Table 3–343 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

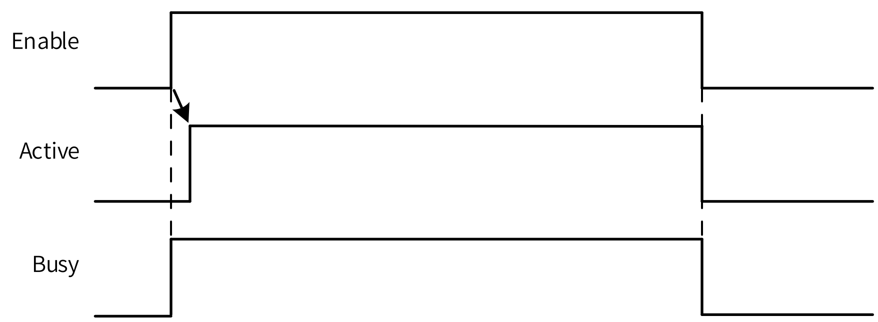
Note

[1] The X element is not supported.

Function and Instruction Description

Before data transmission or reception through UDP, you need to bind the socket to a local port. The UDP_Bind instruction is used to bind the socket to the specified UDP port. When the specified port ID is 0 or empty, UDP_Bind will automatically assign a random port ID, which is then indicated by LocalPort of the socket.

Timing Diagram



3.15.20 UDP_Receive

UDP_Receive – UDP data reception

The UDP_Receive instruction is used to receive data sent by the remote host.

Graphic Block

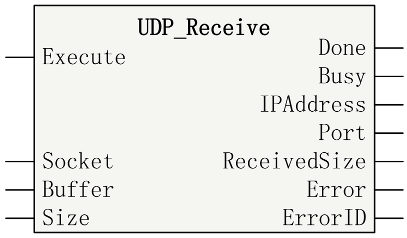


Table 3–344 Instruction format

16-bit Instruction	-					
32-bit Instruction	UDP_Receive: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S3	Size	Data size	No	-	1 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	IPAddress	IP Address	Yes	0	-	DINT
D4	Port	Port	Yes	0	1 to 65535	DINT
D5	ReceivedSize	Size of received data	Yes	0	0 to 32767	INT
D6	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D7	ErrorID	Error code*2	Yes	0	-	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see *"3.15.23 Error Codes of Socket Communication Instructions" on page 662*.

Table 3–345 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	√[1]	√	√	-	-	√	-	-	-
D7	-	-	-	√	√	√	-	-	-

Note

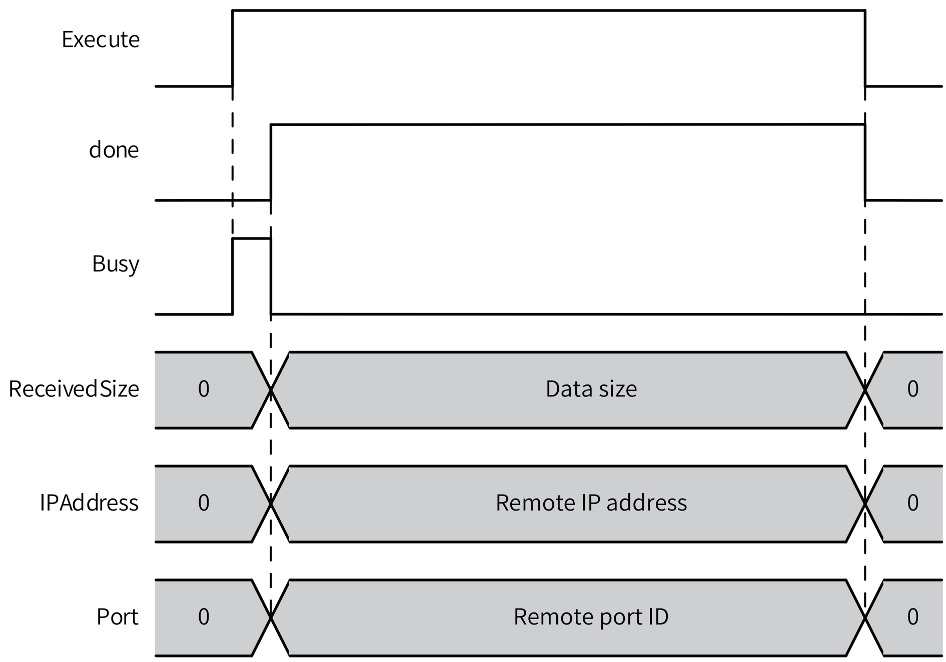
[1] The X element is not supported.

Function and Instruction Description

Message data transmitted by the remote UDP host to the local machine will be stored in the socket buffer area. The UDP_Receive instruction is used to obtain the received message data from the specified socket buffer area.

The data size (size) must be less than or equal to the actual size of the data buffer (Buffer); otherwise, there is a risk of out-of-bounds data access.

Timing Diagram



3.15.21 UDP_Send

UDP_Send – UDP data transmission

The UDP_Send instruction is used to send data to the specified remote host.

Graphic Block

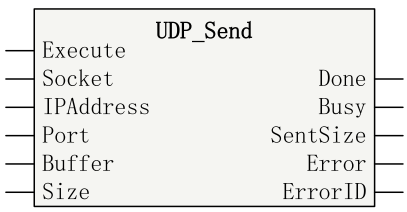


Table 3–346 Instruction format

16-bit Instruction	-					
32-bit Instruction	UDP_Send: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Socket	Socket*1	No	-	-	_sSocket
S2	IPAddress	IP Address	No	-	-	DINT
S3	Port	port	No	-	1 to 65535	DINT
S4	Buffer	Data buffer	No	-	-	BYTE[]/INT[]
S5	Size	Data size	Yes	0	0 to 32767	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Executing	Yes	OFF	ON/OFF	BOOL
D3	SentSize	Size of transmitted data	Yes	0	0 to 32767	INT
D4	Error	Function block error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Error code	Yes	0	*2	INT

Note

- *1: The parameters corresponding to the _sSocket data type are all input and output data types. At present, definition of the _sSocket type variables is not supported, and an INT[20] array can be used instead.
- *2: For details, see *"3.15.23 Error Codes of Socket Communication Instructions" on page 662*.

Table 3–347 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	-	-	-
S2	-	-	-	√	√	√	-	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

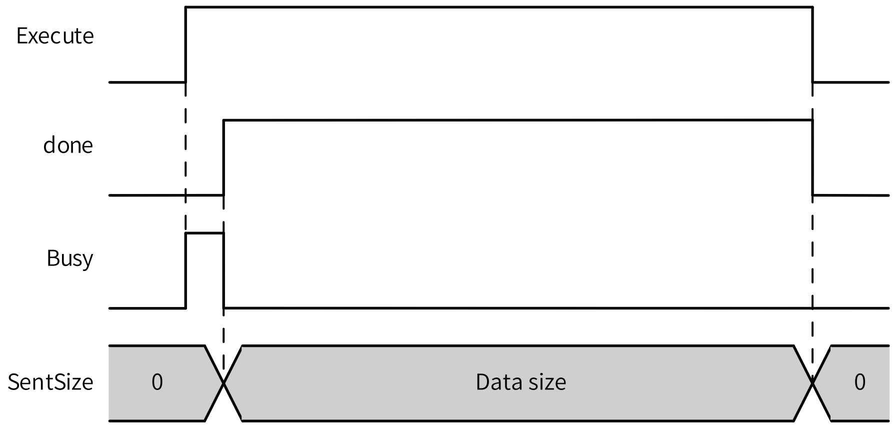
Note

[1] The X element is not supported.

Function and Instruction Description

When the data size parameter (size) is 0, data is transmitted as character strings. That is, the data between the first byte and the terminator (excluding the terminator) in the data buffer (Buffer) is sent. The ASCII code of the terminator is 0.

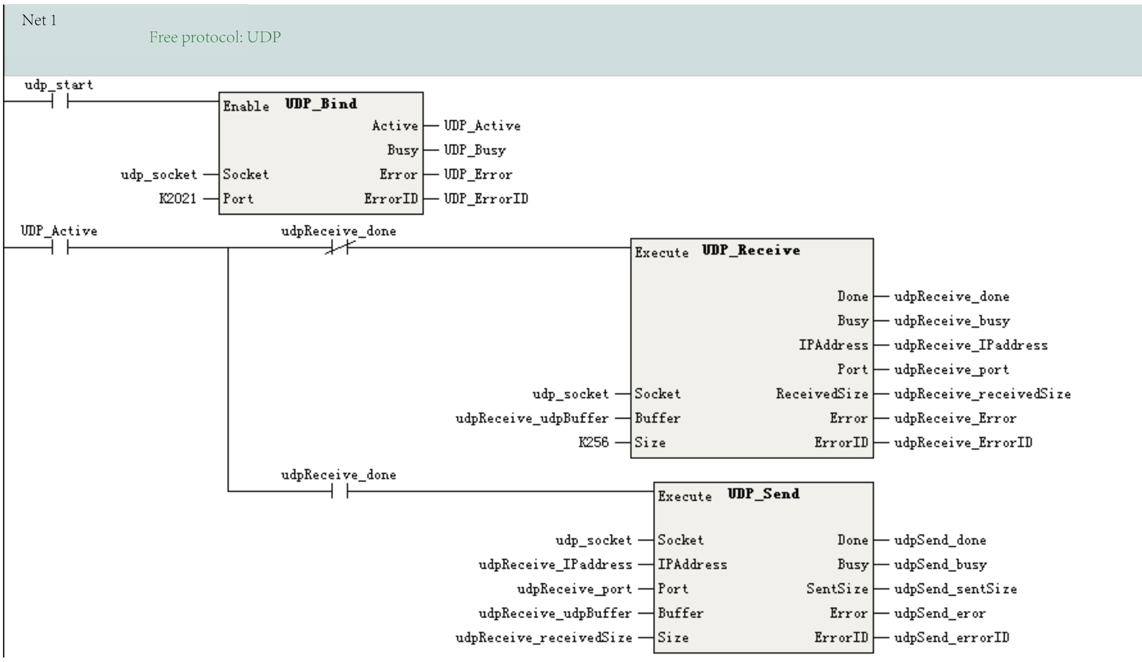
Timing Diagram



Note If this instruction is triggered again when the data frame of the instruction is still being sent, an error is reported due to abnormal timing. To avoid this problem, it is recommended that the next transmission or instruction be triggered upon the state change of the Done or Error signal.

3.15.22 UDP Communication Instance

When communicating with a remote host through the UDP free protocol, the H5U uses the UDP_Bind instruction to bind UDP to port K2021, and after receiving data sent by the remote host, it sends the received data back to the remote host.
The following figure shows the program.



3.15.23 Error Codes of Socket Communication Instructions

The following table lists the error codes of socket communication instructions.

Table 3–348 Error codes of socket communication instructions

Error Code	Description
1	Operation not permitted
2	No such file or directory
3	No such process
4	Interrupted system call
5	I/O error
6	No such device or address
7	Arg list too long
8	Exec format error
9	Bad file number
10	No child subprocesses
11	Try again
12	Out of memory
13	Permission denied
14	Bad address
15	Block device required
16	Device or resource busy
17	File exists
18	Cross-device link
19	No such device
20	Not a directory
21	Is a directory
22	Invalid argument
23	File table overflow

Error Code	Description
24	Too many open files
25	Not a typewriter
26	Text file busy
27	File too large
28	No space left on device
29	Illegal seek
30	Read-only file system
31	Too many links
32	Broken pipe
33	Math argument out of domain of func
34	Math result not representable
35	Resource deadlock would occur
36	File name too long
37	No record locks available
38	Function not implemented
39	Directory not empty
40	Too many symbolic links encountered
41	Operation would block
42	No message of desired type
43	Identifier removed
44	Channel number out of range
45	Level 2 not synchronized
46	Level 3 halted
47	Level 3 reset
48	Link number out of range
49	Protocol driver not attached
50	No CSI structure available
51	Level 2 halted
52	Invalid exchange
53	Invalid request descriptor
54	Exchange full
55	No anode
56	Invalid request code
57	Invalid slot
58	File locking deadlock error
59	Bad font file format
60	Device not a stream
61	No data available
62	Timer expired
63	Out of streams resources
64	Machine is not on the network
65	Package not installed
66	Object is remote
67	Link has been severed
68	Advertise error
69	Srmount error
70	Communication error on send

Error Code	Description
71	Protocol error
72	Multihop attempted
73	RFS specific error
74	Not a data message
75	Value too large for defined data type
76	Name not unique on network
77	File descriptor in bad state
78	Remote address changed
79	Cannot access a needed shared library
80	Accessing a corrupted shared library
81	.lib section in a.out corrupted
82	Attempting to link in too many shared libraries
83	Cannot exec a shared library directly
84	Illegal byte sequence
85	Interrupted system call should be restarted
86	Streams pipe error
87	Too many users
88	Socket operation on non-socket
89	Destination address required
90	Message too long
91	Protocol wrong type for socket
92	Protocol not available
93	Protocol not supported
94	Socket type not supported
95	Operation not supported on transport endpoint
96	Protocol family not supported
97	Address family not supported by protocol
98	Address already in use
99	Cannot assign requested address
100	Network is down
101	Network is unreachable
102	Network dropped connection because of reset
103	Software caused connection abort
104	Connection reset by peer
105	No buffer space available
106	Transport endpoint is already connected
107	Transport endpoint is not connected
108	Cannot send after transport endpoint shutdown
109	Too many references: cannot splice
110	Connection times out.
111	Connection refused
112	Host is down
113	No route to host
114	Operation already in progress
115	Operation now in progress
500	Transmission timing triggering error. The previous data frame is still being sent

3.15.24 ETC_ReadParameter_CoE

ETC_ReadParameter_CoE – Reading SDO parameters of the slave

Graphic Block

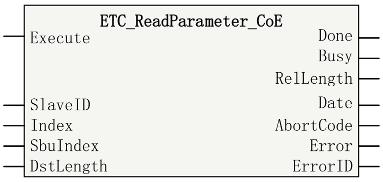
Instruction	Name	LD Expression	LiteST Expression
ETC_ ReadParameter_ CoE	Reading SDO parame- ters of Ether- CAT slave		ETC_ReadParameter_CoE(Execute := ???, SlaveID := ???, Index := ???, SubIndex := ???, DstLength := ???, Done => , Busy => , RelLength => , Data => , AbortCode => , Error => , ErrorID =>);

Table 3–349 Instruction format

16-bit Instruction	-			
32-bit Instruction	ETC_ReadParameter_CoE: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	SlaveID	Slave address	0 to 71	INT
S2	Index	Index	-	INT
S3	SubIndex	Sub-index	-	INT
S4	DstLength	Target length	1 to 4	INT
D1	Done	Completion flag	-	BOOL
D2	Busy	Busy flag	-	BOOL
D3	RelLength	Actual length of the read data, in byte	1 to 4	INT
D4	data	Read data	-	DINT
D5	AbortCode	Abortion code when reading slave object dictionary fails*1	-	DINT
D6	Error	Error flag	-	BOOL
D7	ErrorID	Fault code*2	-	INT

Note

- *1: See *“ SDO AbortCode” on page 673*.
- *2: See *“ Instruction Fault Codes” on page 674*.

Table 3–350 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-
D6	√[1]	√	√	-	-	√	-	-	-
D7	-	-	-	√	√	√	-	-	-

Note

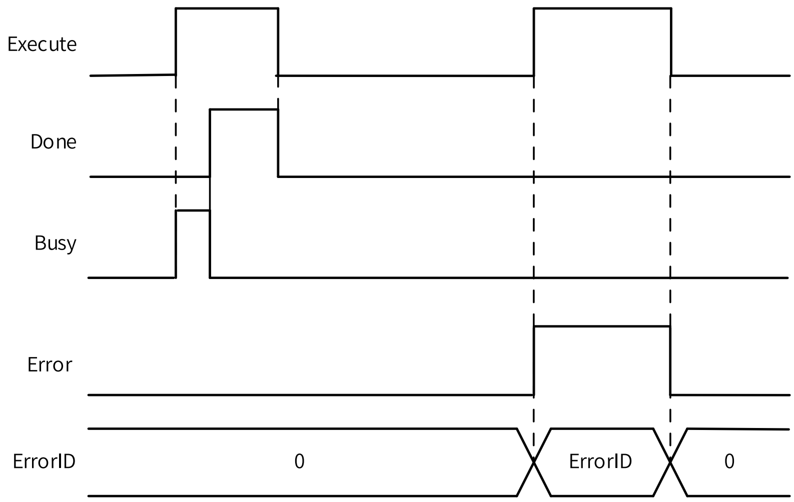
[1] The X element is not supported.

Function and Instruction Description

This instruction is used to read the object dictionary of an EtherCAT slave. It is active on the rising edge.

- SlaveID specifies the configuration address of the EtherCAT slave.
- On the rising edge of Execute, the instruction latches the input parameters on the left and triggers reading the object dictionary specified by Index and SubIndex.
- DstLength specifies the length (in byte) of the object dictionary to read.
- When reading is successful, the Done signal becomes active. Dtate displays the read value, and RelLength displays the actual length of the read object dictionary. When reading fails, the Error output becomes active, and you can determine the cause of the failure by checking AbortCode and ErrorID.
- The parameter Data in this instruction is a DINT type parameter that occupies 4 bytes. When the read object dictionary is SINT or INT data, the reading result is stored in the low-order 8 bits or 16 bits of Data, and the unused high-order 24 bits or 16 bits are filled with 0s. For example, when the read data is the SINT-type or INT-type –8, the data actually stored in Data is 0x000000f8 or 0x0000fff8.

Timing Diagram



Fault Codes

Fault Code	Cause	Solution
8001	Failed to configure the master.	Check whether the master configuration parameters are appropriate.
8002	Failed to configure the slave.	Check whether the slave configuration parameters are appropriate.
8003	Reserved	-
8200	Failed to write the slave startup parameters to the SDO.	Check whether the SDO of the startup parameter list is appropriate.

3.15.25 ETC_WriteParameter_CoE

ETC_WriteParameter_CoE – Writing SDO parameters of the slave

Graphic Block

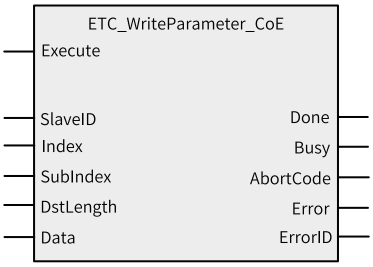
Instruction	Name	LD Expression	LiteST Expression
ETC_ WriteParameter_ CoE	Writing SDO parame- ters of EtherCAT slave		ETC_WriteParameter_CoE(Execute := ???, SlaveID := ???, Index := ???, SubIndex := ???, DstLength := ???, Data := ???, Done => , Busy => , AbortCode => , Error => , ErrorID =>);

Table 3–351 Instruction format

16-bit Instruction	-			
32-bit Instruction	ETC_WriteParameter_CoE: Continuous execution			
Operand	Name	Description	Range	Data Type
S1	SlaveID	Slave ID (Only the configuration address of the slave is allowed.)	0 to 71	INT
S2	Index	Index	-	INT
S3	SubIndex	Sub-index	-	INT
S4	DstLength	Target length	1 to 4	INT
S5	Data	Target data	-	DINT
D1	Done	Completion flag	-	BOOL
D2	Busy	Busy flag	-	BOOL
D3	AbortCode	Abortion code	-	DINT
D4	Error	Error flag	-	BOOL
D5	ErrorID	Fault code*1	-	INT

Note

*1: See *"Instruction Fault Codes" on page 674*.

Table 3–352 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

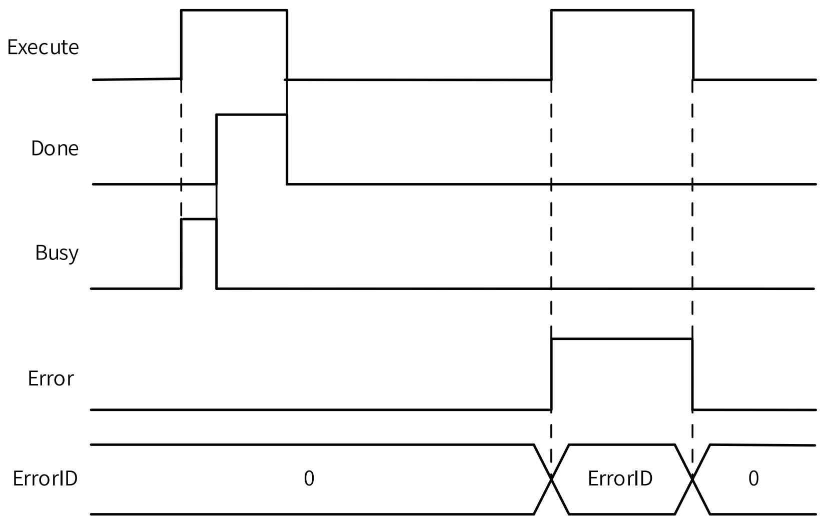
[1] The X element is not supported.

Function and Instruction Description

This instruction is used to write to the object dictionary of an EtherCAT slave. It is active on the rising edge.

- SlaveID specifies the configuration address of the EtherCAT slave.
- On the rising edge of Execute, the instruction latches the input parameters on the left and writes the data in Data to the object dictionary specified by Index and SubIndex.
- DstLength specifies the length (in byte) of the object dictionary to write to.
- When writing is successful, the Done signal becomes active. When writing fails, the Error output becomes active, and you can determine the cause of the failure by checking AbortCode and ErrorID.

Timing Diagram

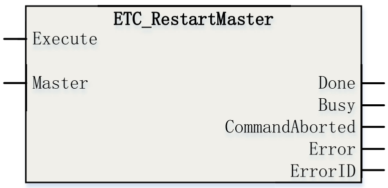


Fault Codes

Fault Code	Cause	Solution
8001	Failed to configure the master.	Check whether the master configuration parameters are appropriate.
8002	Failed to configure the slave.	Check whether the slave configuration parameters are appropriate.
8003	Reserved	-
8200	Failed to write the slave startup parameters to the SDO.	Check whether the SDO of the startup parameter list is appropriate.

3.15.26 ETC_RestartMaster

ETC_RestartMaster – Restarting EtherCAT master

Instruction	Name	LD Expression	LiteST Expression
ETC_ RestartMaster	Restarting EtherCAT master		ETC_RestartMaster(Execute := ???, Master := , Done => , Busy => , CommandAborted => , Error => , ErrorID =>);

16-bit Instruction	ETC_RestartMaster: Continuous execution					
32-bit Instruction	-					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	Master	EtherCAT master	Yes	-	-	-
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	CommandAbort-ed	Abortion of execution	Yes	OFF	ON/OFF	BOOL
D4	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D5	ErrorID	Fault code*1	Yes	0	-	INT16

Note

*1: See *“ Instruction Fault Codes” on page 674*.

Table 3–353 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	√[1]	√	√	-	-	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

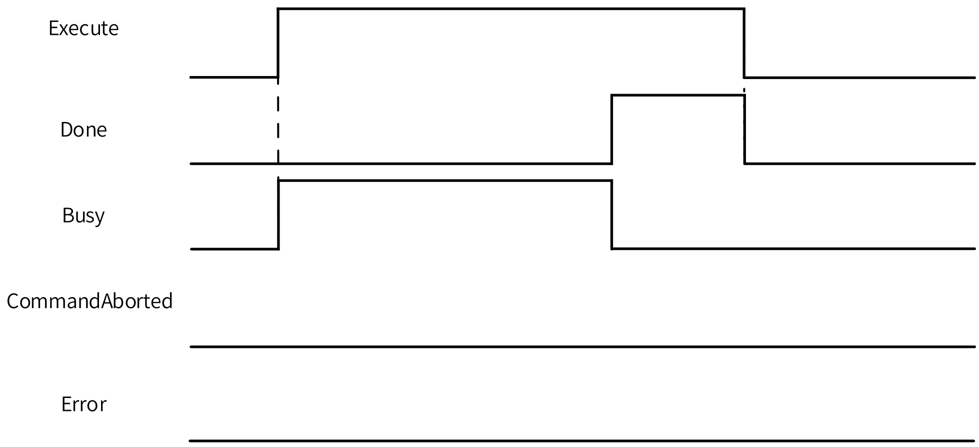
[1] The X element is not supported.

Function and Instruction Description

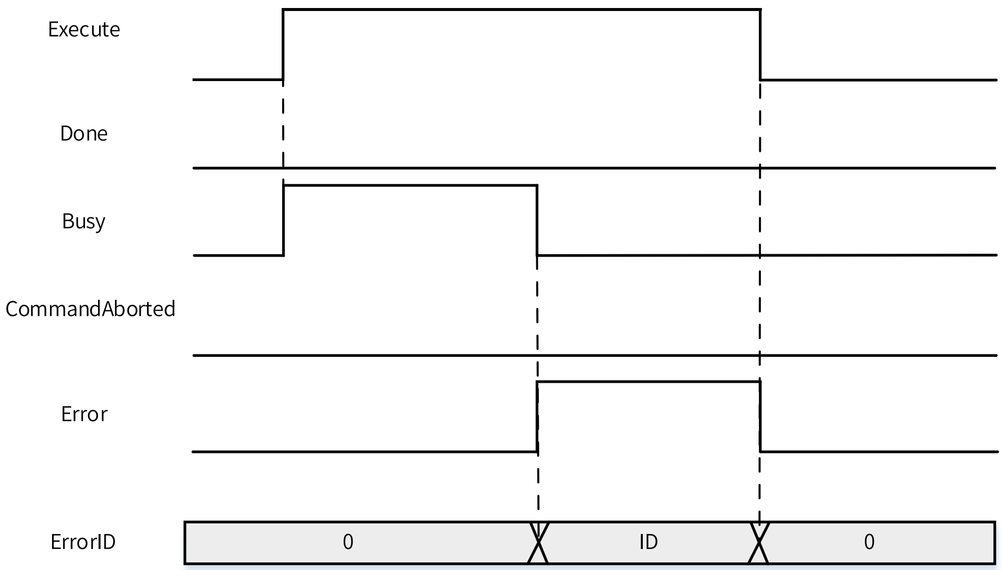
This instruction is used to restart the EtherCAT bus.

Timing Diagram

- If the bus is restarted successfully, the Done output becomes active.

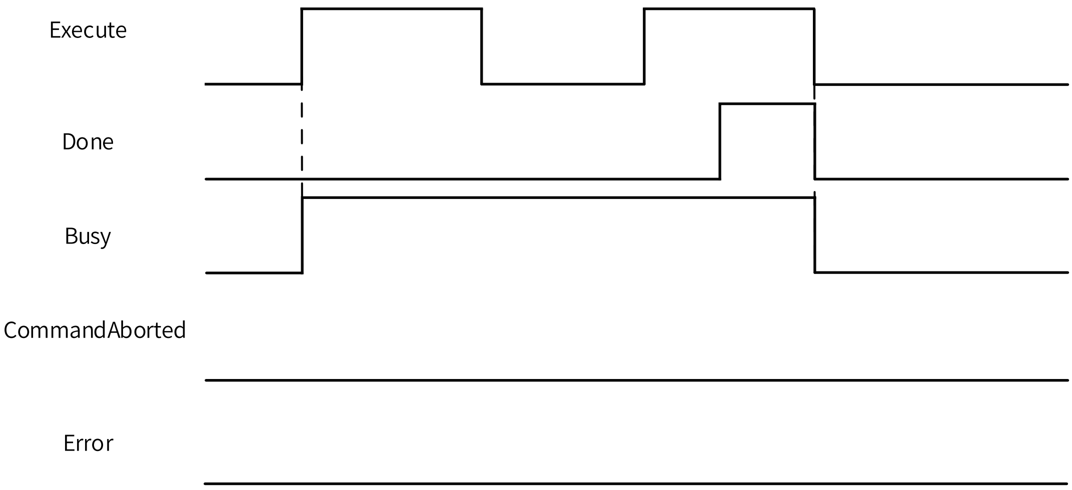


- If the bus fails to restart, the Error output becomes active, and ErrorID indicates the fault code.



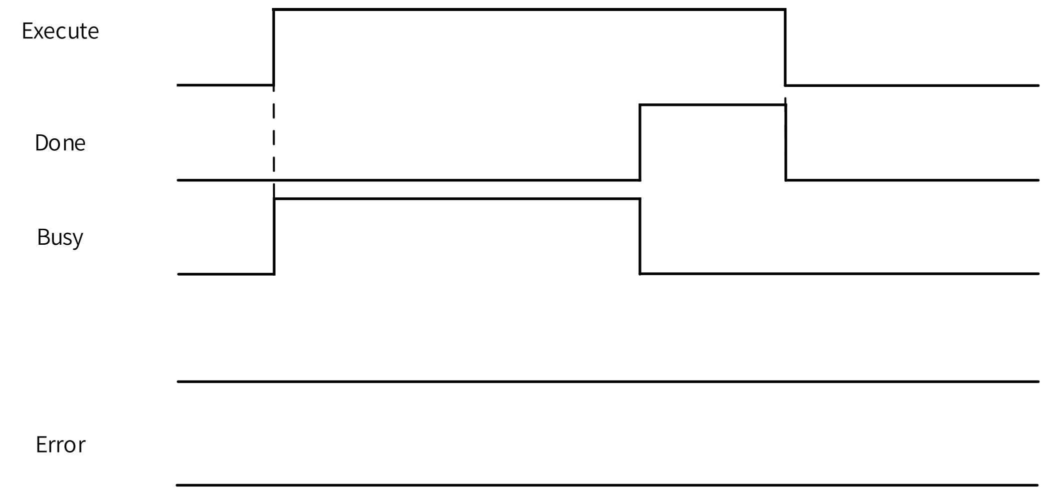
- Re-execution

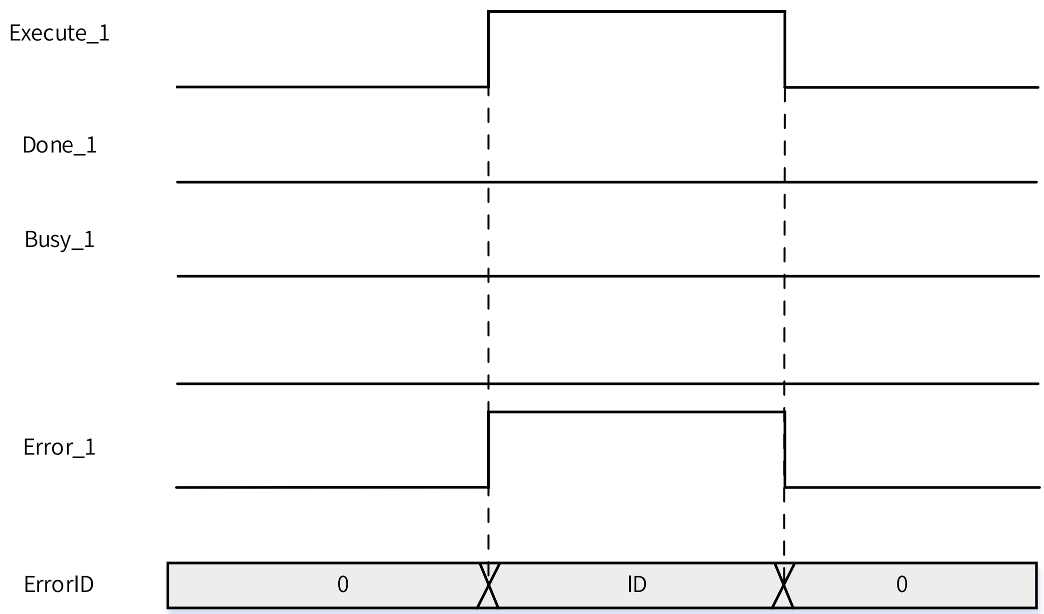
  The same restart instruction can be re-triggered.



- Multi-execution

  This instruction does not support multi-execution. If a second instruction is triggered during execution of an instruction, the second instruction reports a fault, and the first instruction continues execution.





3.15.27 Instruction Codes

SDO AbortCode

Value	Description
0x05 03 00 00	Toggle bit not changed
0x05 04 00 00	SDO protocol timed out
0x05 04 00 01	Client/server instruction qualifier invalid or unknown
0x05 04 00 05	Memory overflow
0x06 01 00 00	Unsupported access object
0x06 01 00 01	Attempting to read a write-only object
0x06 01 00 02	Attempting to write to a read-only object
0x06 02 00 00	No such object in the object directory
0x06 04 00 41	The object cannot be mapped to the PDO.

Value	Description
0x06 04 00 42	The quantity and length of objects to be mapped exceed the allowable range of the PDO.
0x06 04 00 43	General parameter incompatibility
0x06 04 00 47	General internal incompatibility in the device
0x06 06 00 00	Access failed due to a hardware error.
0x06 07 00 10	Data type mismatch: service parameter length mismatch
0x06 07 00 12	Data type mismatch: service parameter too long
0x06 07 00 13	Data type mismatch: service parameter too short
0x06 09 00 11	Sub-index does not exist
0x06 09 00 30	Parameter value out of range (write access)
0x06 09 00 31	Written parameter value too large
0x06 09 00 32	Written parameter value too small
0x06 09 00 36	Maximum value is smaller than minimum value
0x08 00 00 00	General error
0x08 00 00 20	Data cannot be transmitted or stored to the application.
0x08 00 00 21	Data cannot be transmitted or stored to the application due to local control.
0x08 00 00 22	Data cannot be transmitted or stored to the application due to the current device state.
0x08 00 00 23	Failed to generate the object dictionary dynamically to there is currently no object dictionary.

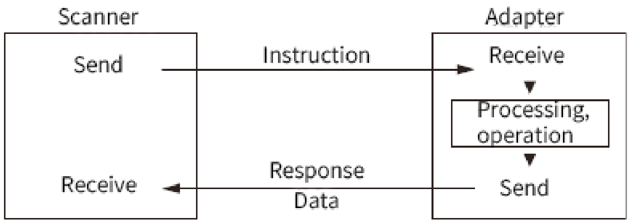
Instruction Fault Codes

Fault Code	Cause	Solution
0x0500	The master is not found.	Check whether EtherCAT bus communication is enabled.
0x0501	The slave is not found.	Check whether the slave exists in configuration.
0x0502	The length of the SDO to read or write to is 0 or greater than 4.	Check whether the length specified in the SDO read or write function is correct.
0x0503	The master is not found.	Check whether the master configuration parameters are correct.
0x0504	Reading or writing fails. 1. The SDO read or write operation times out. 2. The SDO does not exist. 3. Reading or writing to the SDO is not allowed by the slave state. 4. The length of the SDO to read or write to is incorrect.	Check whether the SDO operation is allowed by the slave state machine. Check whether the SDO to read or write to exists. Check whether the length of the SDO to read or write to is correct.
0x0505	Failed to request the memory.	1. Check whether the PLC memory runs out. 2. Contact the manufacturer.
0x0506	The master is in Stopping state.	Do not call this instruction when the master is in Stopping state.

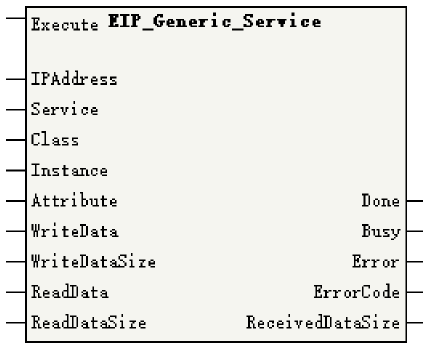
3.15.28 EIP_Generic_Service

EIP_Generic_Service – Calling the "Generic" service of a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Generic_Service: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Service	Service code	No	-	0 to 255	INT
S3	Class	Class code	No	-	-	DINT
S4	Instance	Instance code	No	-	-	DINT
S5	Attribute	Attribute code	No	-	-	DINT
S6	WriteData	Written data buffer area	No	-	-	BYTE[]/INT[]
S7	WriteDataSize	Written data size (in bytes)	No	-	0 to 1502	INT
S8	ReadData	Read data buffer area	No	-	-	BYTE[]/INT[]
S9	ReadDataSize	Read data size (in bytes)	No	-	0 to 1502	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL

D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT
D5	ReceivedData-Size	Received data size	Yes	0	0 to 1502	INT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–354 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	√	√	-	-
S6	-	-	-	√	√	-	-	-	-
S7	-	-	-	√	√	√	√	-	-
S8	-	-	-	√	√	-	-	-	-
S9	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

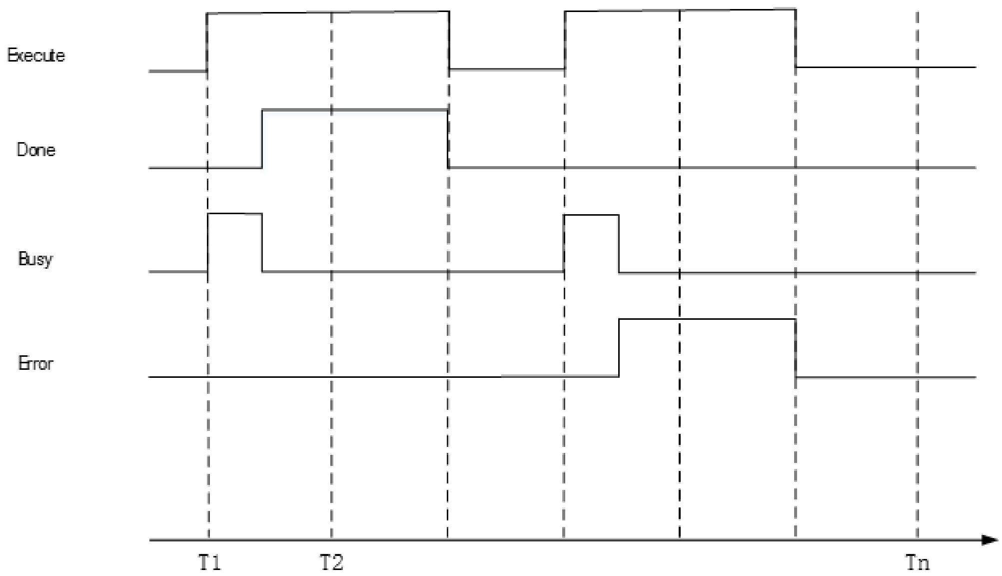
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values, and the ReadData buffer area is cleared.
- When execution of the instruction is successful, Done is set to ON, the size of the service response data is saved to ReceivedDataSize, and the service response data is written to the ReadData buffer area.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

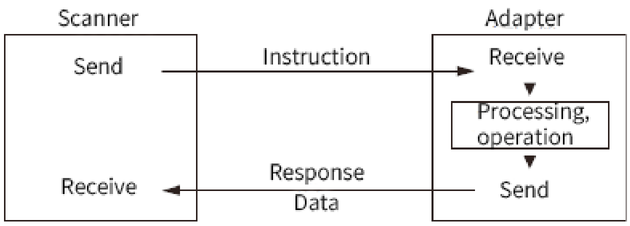
Timing Diagram



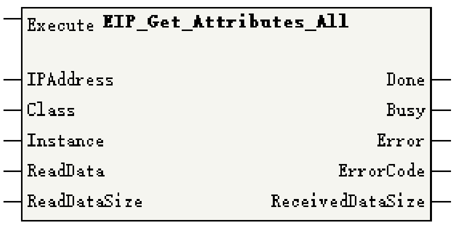
3.15.29 EIP_Get_Attributes_All

EIP_Get_Attributes_All – Calling the "Get_Attributes_All" service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Get_Attributes_All: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP

S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	ReadData	Read data buffer area	No	-	-	BYTE[]/INT[]
S5	ReadDataSize	Read data size (in bytes)	No	-	0 to 1502	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT
D5	ReceivedData-Size	Received data size	Yes	0	0 to 1502	INT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–355 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-
S5	-	-	-	√	√	√	√	-	-
D1	✓[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

Note

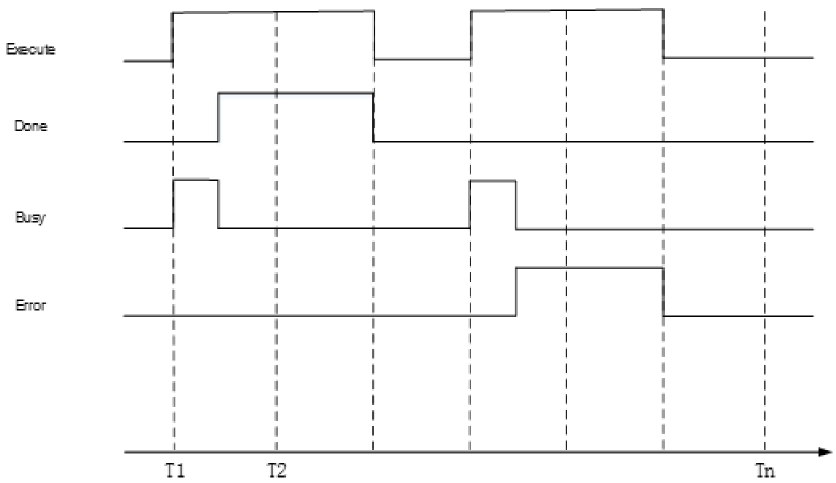
[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values, and the ReadData buffer area is cleared.

- When execution of the instruction is successful, Done is set to ON, the size of the service response data is saved to ReceivedDataSize, and the service response data is written to the ReadData buffer area.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

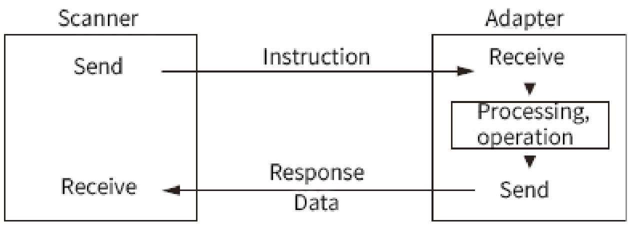
Timing Diagram



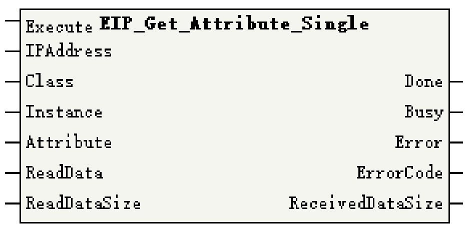
3.15.30 EIP_Get_Attribute_Single

EIP_Get_Attribute_Single – Calling the "Get_Attribute_Single" service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-
32-bit Instruction	EIP_Get_Attribute_Single: Continuous execution

Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	Attribute	Attribute code	No	-	-	DINT
S5	ReadData	Read data buffer area	No	-	-	BYTE[]/INT[]
S6	ReadDataSize	Read data size (in bytes)	No	-	0 to 1502	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT
D5	ReceivedData-Size	Received data size	Yes	0	0 to 1502	INT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–356 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	-	-	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-
D5	-	-	-	√	√	√	-	-	-

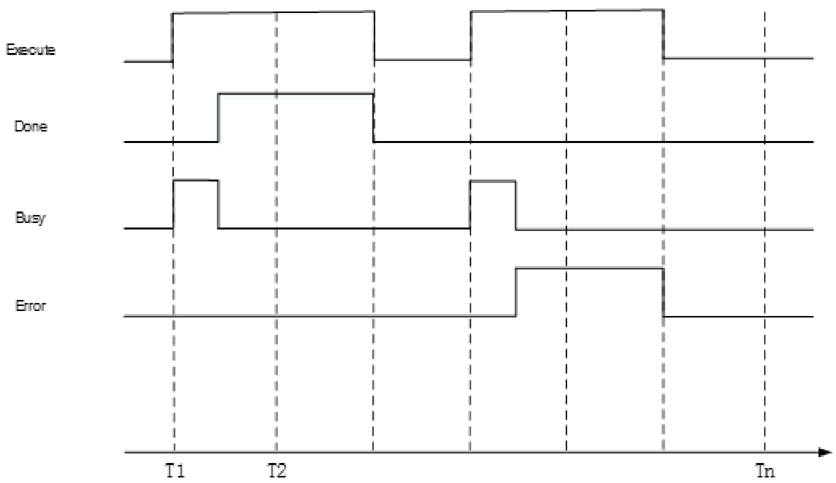
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values, and the ReadData buffer area is cleared.
- When execution of the instruction is successful, Done is set to ON, the size of the service response data is saved to ReceivedDataSize, and the service response data is written to the ReadData buffer area.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

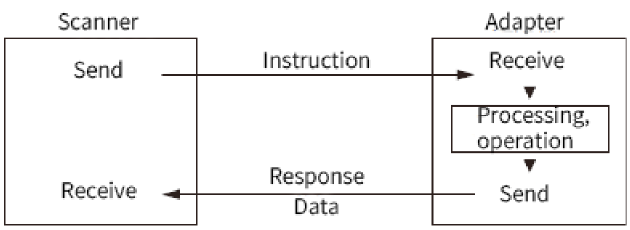
Timing Diagram



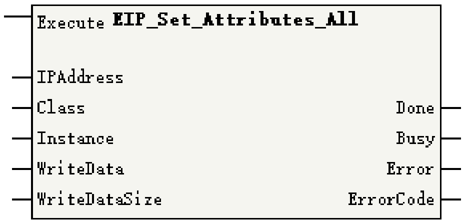
3.15.31 EIP_Set_Attributes_All

EIP_Set_Attributes_All – Calling the "Set_Attributes_All" service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Set_Attributes_All: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	WriteData	Written data buffer area	No	-	-	BYTE[]/INT[]
S5	WriteDataSize	Written data size (in bytes)	No	-	0 to 1502	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–357 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-
S5	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

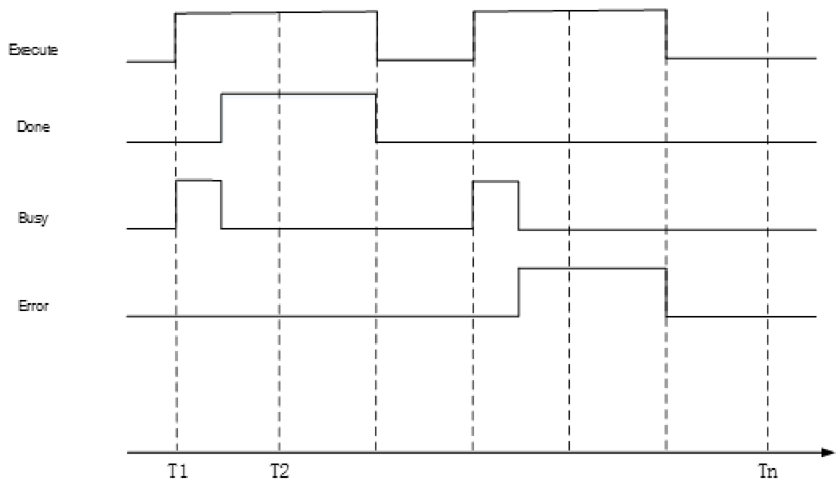
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

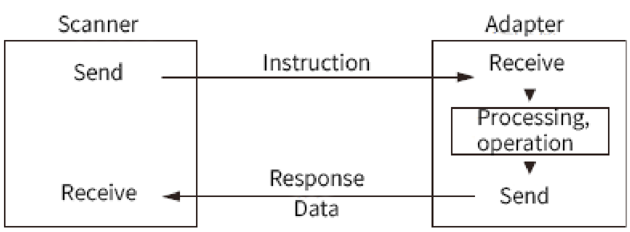
Timing Diagram



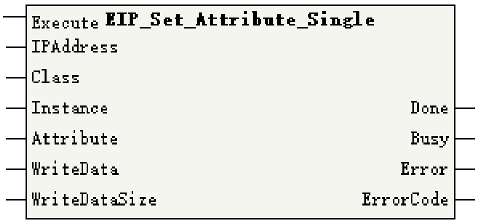
3.15.32 EIP_Set_Attribute_Single

EIP_Set_Attribute_Single – Calling the "Set_Attribute_Single" service for a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Set_Attribute_Single: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
S4	Attribute	Attribute code	No	-	-	DINT
S5	WriteData	Written data buffer area	No	-	-	BYTE[]/INT[]
S6	WriteDataSize	Written data size (in bytes)	No	-	0 to 1502	INT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–358 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	√	√	-	-
S5	-	-	-	√	√	-	-	-	-
S6	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

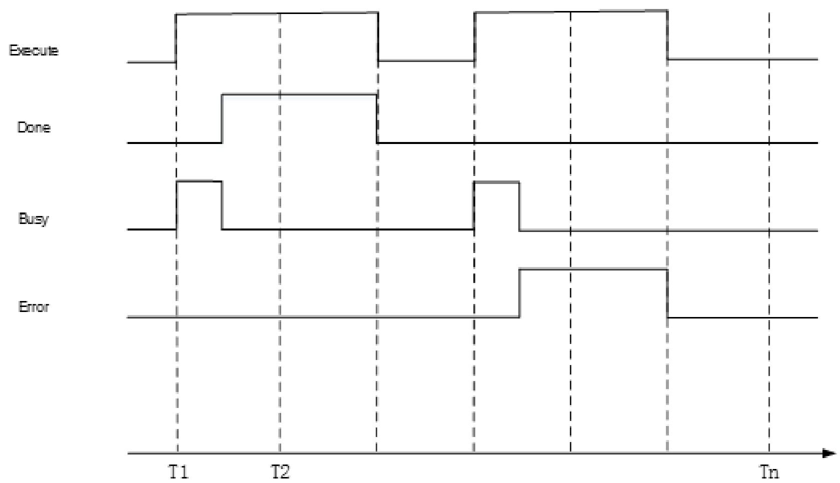
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Timing Diagram

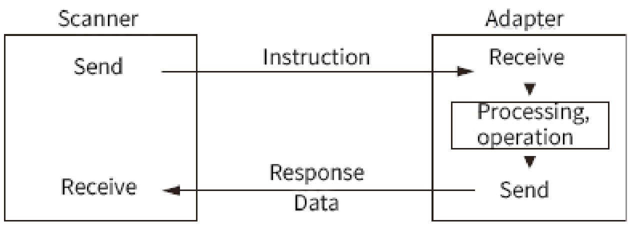


3.15.33 EIP_Apply_Attributes

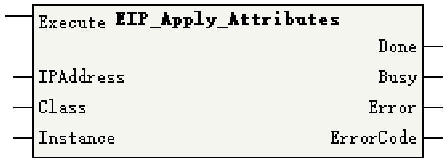
EIP_Apply_Attributes – Calling the "Apply_Attributes" service for a specific instance of the EtherNet/IP object

The adapter adopts and saves attributes set by "Get_Attribute_Single" or "Get_Attribute_All".

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Apply_Attributes: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–359 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

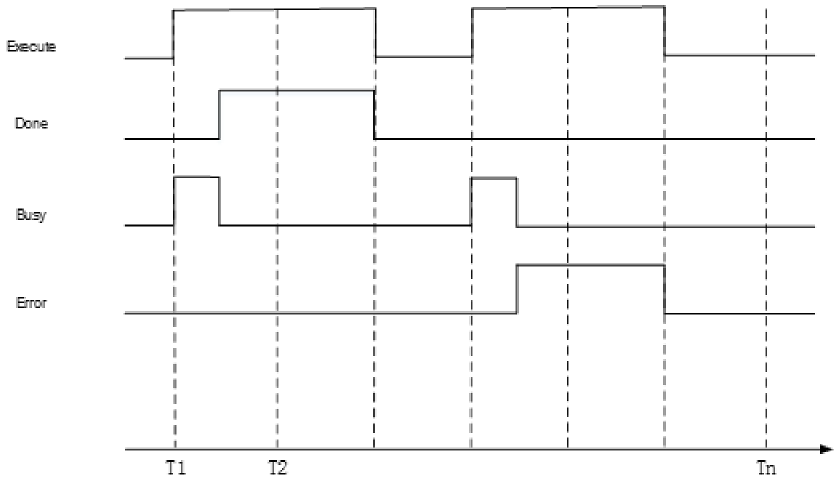
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

Timing Diagram

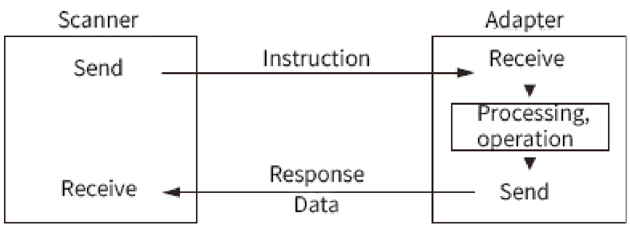


3.15.34 EIP_NOP

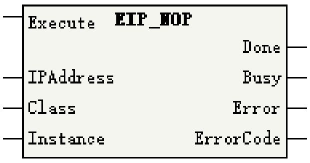
EIP_NOP – Calling the "NOP" (No Operation) service for a specific instance of the EtherNet/IP object

It is often used to check whether the adapter is still available in the network.

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_NOP: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–360 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

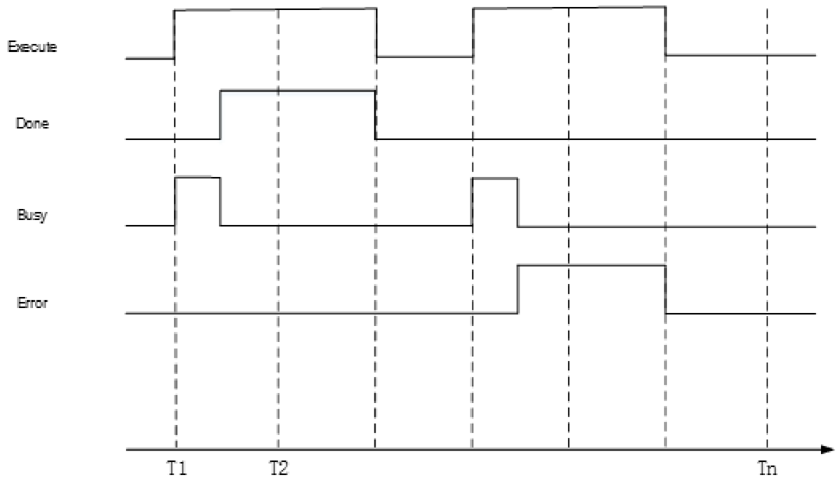
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

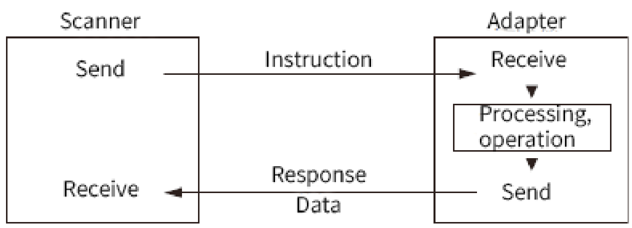
Timing Diagram



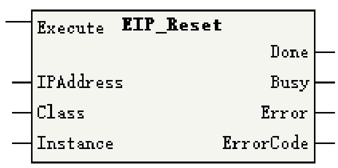
3.15.35 EIP_Reset

EIP_Generic_Service – Calling the "Reset" service of a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Reset: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL
D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–361 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

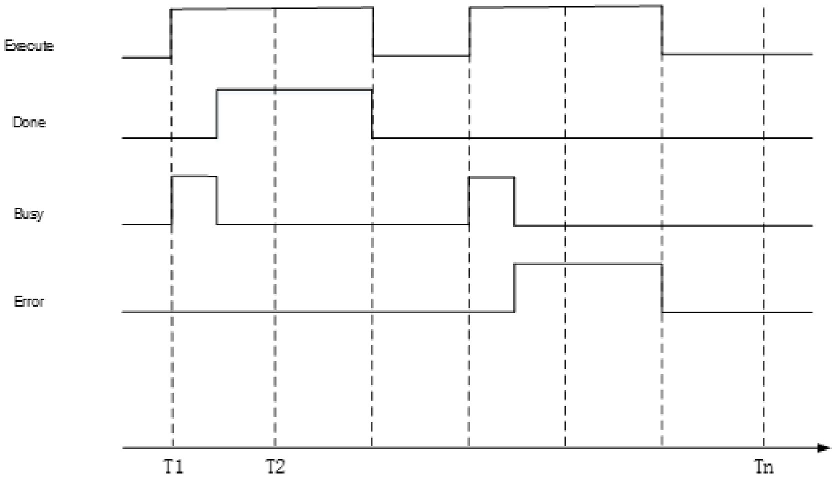
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

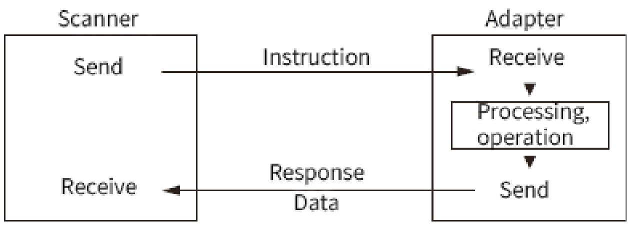
Timing Diagram



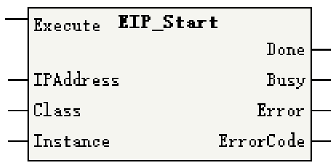
3.15.36 EIP_Start

EIP_Start – Calling the "Start" service of a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-Bit Instruction	-					
32-bit Instruction	EIP_Start: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL

D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–362 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

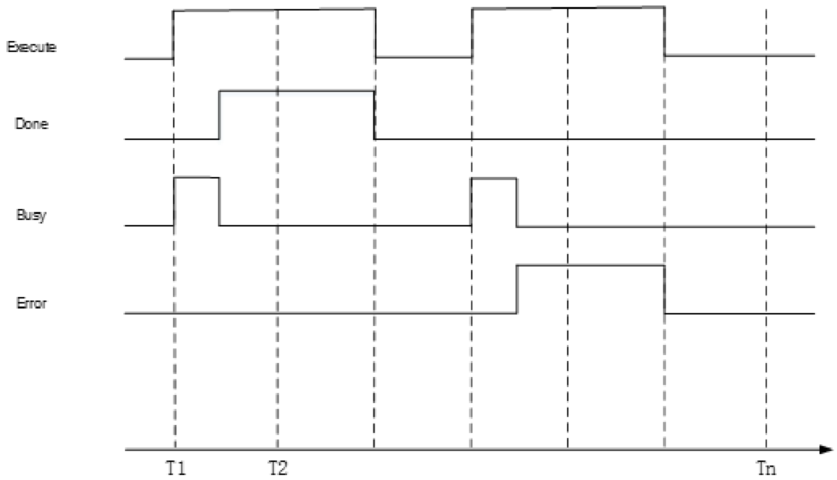
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

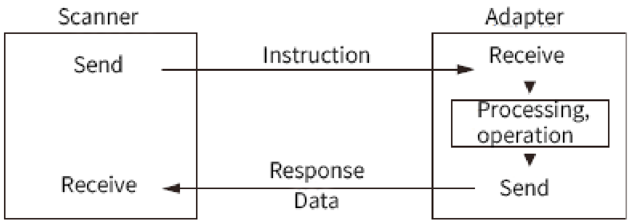
Timing Diagram



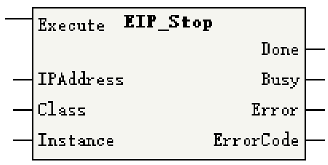
3.15.37 EIP_Stop

EIP_Stop – Calling the "Stop" service of a specific instance of the EtherNet/IP object

This function is programmed on the EtherNet/IP scanner. The EtherNet/IP scanner sends an "Unconnected Explicit Message" service request to the EtherNet/IP adapter. The EtherNet/IP adapter accepts and processes the request, and sends a service response to the EtherNet/IP scanner.



Graphic Block



16-bit Instruction	-					
32-bit Instruction	EIP_Stop: Continuous execution					
Operand	Name	Description	Empty Allowed	Default	Range	Data Type
S1	IPAddress	IP address*1	No	-	-	IP
S2	Class	Class code	No	-	-	DINT
S3	Instance	Instance code	No	-	-	DINT
D1	Done	Completion flag	Yes	OFF	ON/OFF	BOOL
D2	Busy	Busy flag	Yes	OFF	ON/OFF	BOOL

D3	Error	Error flag	Yes	OFF	ON/OFF	BOOL
D4	ErrorCode	Fault code	Yes	0	-	DINT

Note

*1: A parameter of which the data type is IP is a dotted decimal IP address. For example, if the adapter IP address is 192.168.1.88, the operand IPAddress should be set to 192.168.1.88.

Table 3–363 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	-	-	√	-	-	√
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
D1	√[1]	√	√	-	-	√	-	-	-
D2	√[1]	√	√	-	-	√	-	-	-
D3	√[1]	√	√	-	-	√	-	-	-
D4	-	-	-	√	√	√	-	-	-

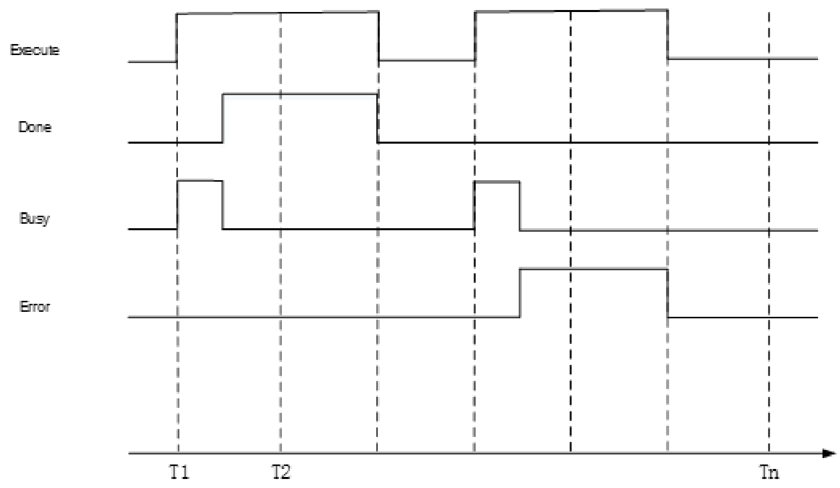
Note

[1] The X element is not supported.

Function and Instruction Description

- Obtain the EtherNet/IP service provided by the specified manufacturer by setting parameters such as the adapter IPAddress, Class, and Instance.
- On the rising edge of the flow, Busy is set to ON, indicating that the instruction is being executed.
- On the falling edge of the flow, if Busy is ON, execution of the instruction continues. Otherwise, Done, Error, and ErrorCode are set to default values.
- When execution of the instruction is successful, Done is set to ON.
- When an error occurs during execution, Error is set to ON, and the error code is saved to ErrorCode.

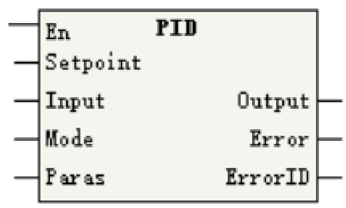
Timing Diagram



3.16 Other Instructions

3.16.1 PID

The PID instruction performs PID calculation to control the parameters of a close-loop control system.
PID – PID calculation



16-bit Instruction	PID: Continuous execution			
32-bit Instruction	-			
Operand	Name	Description	Range	Data Type
S1	Setpoint	Set target control value. Unit: 0.1°C	-	INT16
S2	Input	Measured feedback value. The user program needs to read the actual value of the device and update this parameter. Unit: 0.1°C.	-	INT16
S3	Mode	PID working mode, that is, algorithm selection It is recommended that the variable retentive at power failure be used.	-	INT16
S4	Paras	Settings of parameters required for PID calculation or buffer of intermediate results	-	INT16, VOID*n
D1	Output	PID analog output percentage in 0.1%. For example, 1000 represents 100%.	-	INT16

D2	Error	Error flag	-	BOOL
D3	ErrorID	Error code	-	INT16

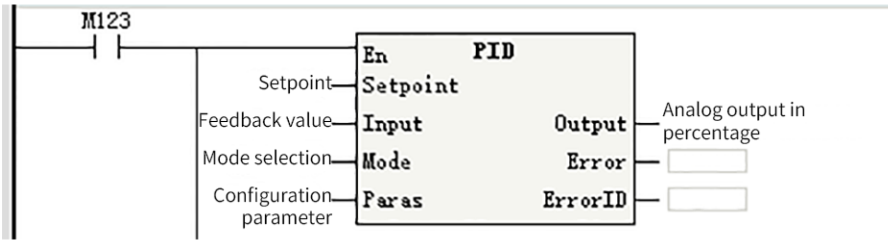
Table 3–364 List of elements

Operand	Bit			Word		Pointer	Constant		Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	K, H	E	
S1	-	-	-	√	√	√	√	-	-
S2	-	-	-	√	√	√	√	-	-
S3	-	-	-	√	√	√	√	-	-
S4	-	-	-	√	√	-	-	-	-
D1	-	-	-	√	√	√	-	-	-
D2	✓[1]	-	√	-	-	√	-	-	-
D3	-	-	-	√	√	√	-	-	-

Note

[1] The X element is not supported.

Instruction Description



S3 specifies the PID mode, which is described as follows:

Para.	Mode	Description
S3	0	Incremental PID
	1	Position PID
	2	Special PID
	3	Temperature control PID
	4	MPC control PID (used for process control in the air compressor industry)
	5	Large-inertia temperature control PID (used for process control in the injection molding machine industry)
	6	Auto-tuning PID

Mode 0: Incremental PID Instruction

Unit	Parameter	Description
S4	Sampling time (TS)	The maximum sampling time is 132767 ms and the sampling time must be greater than the PLC's scan cycle.
S4+1	Action direction (ACT)	Bit0 = 0: forward action; bit0 = 1: reverse action Bit1 = 0: input variable alarm disabled; bit1 = 1: input variable alarm enabled Bit2 = 0: output variable alarm disabled; bit2 = 1: output variable alarm enabled Bit3: unavailable Bit4 = 0: auto-tuning not executed; bit4 = 1: auto-tuning executed (The current version does not provide auto-tuning for the moment.) Bit5 = 0: upper/lower output limits invalid; bit5 = 1: upper/lower output limits valid Bit6 to bit15: unavailable Do not set both bit5 and bit2 to ON.
S4+2	Input filter constant (α)	Value range: 0 to 99, in percent. When it is set to 0, no input filter is processed.
S4+3	Proportional gain (Kp)	Value range: 1 to 32767, in percent.
S4+4	Integral time (T1)	Value range: 0 to 32767 (x 100 ms). When it is set to 0, it is processed as ∞ (no integral).
S4+5	Differential gain (KD)	Value range: 0 to 100, in percent. When it is set to 0, no differential gain is processed.
S4+6	Differential time (TD)	Value range: 0 to 32767 (x 10 ms). When it is set to 0, no differential is processed.
S4+(7–19)	Occupied by internal processing of PID calculation. Clear these units before initial running.	
When bit1 is 1 and bit2 or bit5 is 1 in <ACT>, S4+(20–24) are occupied and defined as follows:		
S4+20	Input variable (incremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit1 is 1 in <ACT>.
S4+21	Input variable (decremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit1 is 1 in <ACT>.
S4+22	Output variable (incremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit2 is 1 and bit5 is 0 in <ACT>.
		The upper output limit ranges from -32768 to +32767. This parameter is valid when bit1 is 0 and bit5 is 1 in <ACT>.
S4+23	Output variable (decremental) alarm value	Value range: 0 to 32767. This parameter is valid when bit2 is 1 and bit5 is 0 in <ACT> of S4+1.
		The lower output limit ranges from -32768 to +32767. This parameter is valid when bit1 is 0 and bit5 is 1 in <ACT>.
S4+24	Alarm output	Bit0 input variable (incremental) overflow Bit1 input variable (decremental) overflow Bit2 output variable (incremental) overflow Bit3 output variable (decremental) overflow This parameter is valid when bit1 is 1 or bit2 is 1 in <ACT>.
S4+25	Occupied by internal processing of PID calculation	

Mode 1: Position PID Instruction

The following table lists the functions and setting methods of the parameters in each unit.

Address	Name	Value Range	Description
S4 + 0	Sampling cycle	1 to 32767, in ms	PID calculation cycle, which is 10 by default
S4 + 1	Control mode	-	0: Forward (default) 1: Reverse
S4 + 2	Proportional gain Kp1	0 to 32767, in percent	Proportional gain (Default value: 0)
S4 + 3	Integral gain Ki1	0 to 32767, in percent	Integral gain (Default value: 0)
S4 + 4	Differential gain Kp1	0 to 32767, in percent	Differential gain (Default value: 0)
S4 + 5	Deviation dead zone	0 to 32767	0: Disabled Non-0: Deviation is zero if the deviation value is less than the specified value. (Default value: 0)
S4 + 6	Upper output limit	-32768 to 32767	Maximum output value
S4 + 7	Lower output limit	-32768 to 32767	Minimum output value
S4 + 8	Upper integral limit	-32768 to 32767	Maximum cumulative integral value ※1
S4 + 9	Lower integral limit	-32768 to 32767	Minimum cumulative integral value ※1
S4 + 10 S4 + 11	Cumulative integral	-	32-bit floating-point number
S4 + 12	Last output	-32768 to 32767	Used for differential calculation
S4 + 13	Kp2	0 to 32767, in percent	(Default value: 0)
S4 + 14	Ki2	0 to 32767, in percent	(Default value: 0)
S4 + 15	Kd2	0 to 32767, in percent	(Default value: 0)
S4 + 16	Parameter switching condition	-	0: No switching 1: Switching based on deviation 2: User-defined ※2
S4 + 17	Lower deviation limit E1	-32768 to 32767	Deviation start point or user-defined switching start point
S4 + 18	Upper deviation limit E2	-32768 to 32767	Deviation end point or user-defined switching end point
S4 + 19	User-defined switching reference	-32768 to 32767	Switching reference when the parameter switching condition is set to 2
S4 + 20 S4 + 21 S4 + 22 S4 + 23 S4 + 24 S4 + 25 S4 + 26	Occupied by internal operation	-	-

- ※1: When both the upper and lower integral limits are set to 0, the upper limit +32,737 and lower limit –32,768 take effect.
- ※2: When (S4+16) is 0, (S4+17) to (S4+19) are invalid.

Principles of position PID calculation

The PID calculation formula is as follows:

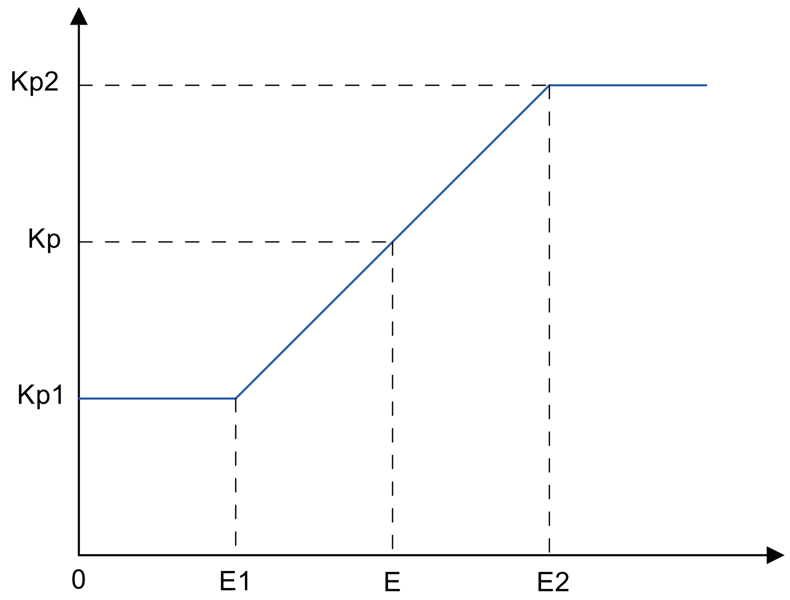
u(k) = Kp x e(k) + Ki x T x Σe(i) + (Kd/T) x [Pv(k) – Pv(k-1)]

u(k)	Current output value	Pv(k-1)	Feedback value at the last time point
e(k)	Current deviation	T	Sampling time
Σe(i)	Current cumulative integral	Kp	Proportional gain
Sv(k)	Current setpoint	Ki	Integral gain
Pv(k)	Current feedback value	Kd	Differential gain

Forward direction: e(k) = Sv(k) – Pv(k)

Reverse direction: e(k) = Pv(k) – Sv(k)

Principles of parameter switching (proportional gain Kp used an as example)



Kp1	(S4 + 2)
Kp2	(S4 + 13)
E1	(S4 + 17)
E2	(S4 + 18)
E	Switching reference

E ⩽ E1: Kp = Kp1

E1 < E < E2: Kp = (Kp2 – Kp1) x E/(E2 – E1)

E ⩾ E2: Kp = Kp2

S4 + 16	0	No switching
	1	E = |Sv – Pv|
	2	E = S4+19

Mode 2: Special PID Instruction (operation principles same as those of customized Inovance 307 series AC drive)

The following table lists the functions and setting methods of the parameters in each unit.

Address	Name	Value Range	Description	AC Drive Function Code	Winding Parameter	Unwinding Parameter	Wire Drawing Machine Parameter
S4 + 0	Sampling time	1 to 32767, in ms	PID calculation cycle	-	10	10	10
S4 + 1	Mode setting	-	0: Forward 1: Reverse	-	-	-	-
S4 + 2	Default parameter selection	-	0: No initialization 1: Winding parameter 2: Unwinding parameter 3: Wire drawing machine parameter	-	1	2	3
S4 + 3	Feedback range setting	0 to 32767	AND feedback range setting	FA-04	1000	1000	1000
S4 + 4	Output range	0 to 32767	Output range	-	10000	10000	10000
S4 + 5	Maximum reverse output	0 to 32767	Maximum reverse output ※1	-	10000	10000	10000
S4 + 6	Output range selection	-	0: Relative to the maximum range 1: Relative to the main output (D+1)	F0-05	0	0	1
S4 + 7	Auxiliary output range	0 to 32767, in percent	Valid when (S4+6) is 1	F0-06	-	-	70
S4 + 8	Proportional gain Kp1	0 to 32767, in 0.1%	Proportional gain (Default value: 0)	FA-05	100	150	45
S4 + 9	Integral time Ti1	0 to 32767, in 0.01s	Integral gain (Default value: 0)	FA-06	120	130	200
S4 + 10	Differential time Td1	0 to 32767, in 0.001s	Differential gain (Default value: 0)	FA-07	150	0	0
S4 + 11	Deviation limit	0 to 32767, in 0.1%	Maximum calculation deviation	FA-09	0	0	0
S4 + 12	Differential limit	0 to 32767, in 0.01%	Maximum differential limit	FA-10	50	-	-
S4 + 13	PID reference change time	0 to 32767, in ms	After startup, the reference value reaches the setpoint after the specified time elapses.	FA-11	5000	0	0
S4 + 14	Proportional gain Kp2	0 to 32767, in 0.1%	(Default value: 0)	FA-15	-	-	-
S4 + 15	Integral time Ti2	0 to 32767, in 0.01s	(Default value: 0)	FA-16	-	-	-
S4 + 16	Differential time Td2	0 to 32767, in 0.001s	(Default value: 0)	FA-17	-	-	-

Address	Name	Value Range	Description	AC Drive Function Code	Winding Parameter	Unwinding Parameter	Wire Drawing Machine Parameter
S4 + 17	Parameter switching condition	-	0: No switching 1: Switching based on deviation 2: User-defined ※2	FA-18	-	-	-
S4 + 18	Lower deviation limit	0 to 32767, in 0.1%	Deviation start point or user-defined switching start point	FA-19	-	-	-
S4 + 19	Upper deviation limit	0 to 32767, in 0.1%	Deviation end point or user-defined switching end point	FA-20	-	-	-
S4 + 20	User-defined switching reference	0 to 32767, in 0.1%	Switching reference when the parameter switching condition is set to 2	-	-	-	-
S4 + 21	Initial output	0 to 32767, in 0.1%	Initial value after PID startup	FA-21	0	0	0
S4 + 22	Initial output hold time	0 to 32767, in ms	Time during which the initial value remains unchanged	FA-22	0	0	0
S4 + 23	Output deviation limit	0 to 32767, in 0.1%	Range of every deviation change	-	0	0	0
S4 + 24 ... S4 + 30	Internal operation	-	-	-	-	-	-

Address	Name	Description
D1 + 0	Total output	PID calculation component + (D1+1)
D1 + 1	Main output	User-designated main output (AC drive dominant frequency) This value is set to 0 for pure PID.

- ※1: Maximum negative value of PID output. The following are two examples. If this parameter is set to 100, the maximum negative output is –100.
- ※2: See the parameter switching principle of the position-type PID instruction.

PID calculation formula

$u(k) = Kp \{e(k) + T/Ti \times \Sigma e(i) + Td/T \times [e(k) - e(k-1)]\}$

u(k)	Current output value	Σe(i)	Current cumulative integral
Kp	Proportional gain	T	Sampling time
e(k)	Current deviation	Ti	Integral time
e(k-1)	Deviation at the last time point	Td	Differential time
Sv(k)	Current setpoint	Ki	Integral gain
Pv(k)	Current feedback value	Kd	Differential gain

Forward direction: e(k) = Sv(k) – Pv(k); reverse direction: e(k) = Pv(k) – Sv(k)

For details about parameter switching, see the position PID description.

Main output application

When (S4+6) is 0, (D1+1) is forcibly set to 0.

When (S4+6) is 1, (S4+7) is enabled. The maximum PID component is equal to (S4+7) percent of (D1+1).

Final (D1+0) = PID component + Main output (D1+1)

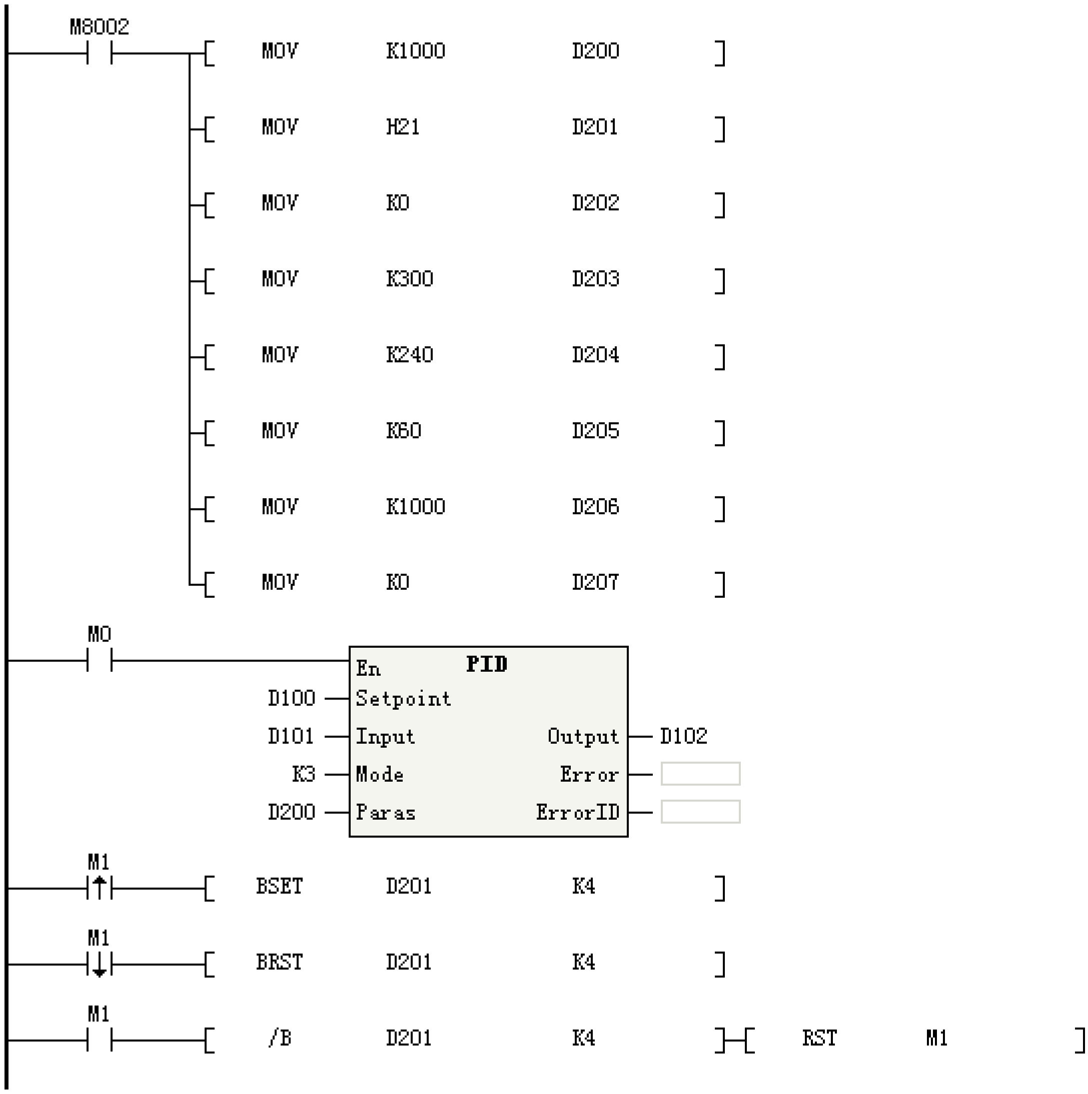
Mode 3: Temperature Control PID Instruction

The following table lists the functions and setting methods of the parameters in each unit.

Unit	Parameter	Description
S4	Sampling cycle	The sampling cycle ranges from 1 ms to 32767 ms and must be greater than the PLC's scan cycle.
S4+1	Mode	Bit8 to bit15: unavailable Bit5 to bit7: unavailable Bit4 = 0: auto-tuning not executed; bit4 = 1: auto-tuning executed, auto reset after auto-tuning is completed Bit1 to bit3: unavailable Bit0 = 0: forward action; bit0 = 1: reverse action
S4+2	Auto-tuning rule	0: common mode, moderate overshoot 1: Slow mode, small overshoot but slow temperature rise 2: Fast mode, fast temperature rise but large overshoot
S4+3	Scaling band	Auto-tuning result scaling band output. The value range is 1 to 32767. The smaller the value of the scaling band, the stronger the scaling effect. After self-tuning, the scaling band will be automatically adjusted to the value after self-tuning.
S4+4	Integral time	Auto-tuning result integral band output. The value range is 0 to 32767, in second. 0 indicates no integral processing.
S4+5	Differential time	Auto-tuning result differential band output. The value range is 0 to 32767, in second. 0 indicates no differential processing.
S4+6	Upper output limit	The upper output limit ranges from -32768 to +32767 and must be greater than the lower output limit.
S4+7	Lower output limit	The lower output limit ranges from –32768 to +32767 and must be less than sign than the upper output limit.
S4+8	Reserved	Occupied by internal processing of PID calculation
S4+9	Scaling output	Current scaling output
S4+10	Integral output	Current integral output
S4+11	Differential output	Current differential output
S4+(12 to 29)	Reserved	Occupied by internal processing of auto-tuning calculation

Instruction example:

M0: Temperature control enable/disable; M1: Auto-tuning start/stop (auto reset after auto-tuning is completed)



Mode 4: Special PID Instruction for Air Compressors

Note the following:

- Only one special PID is supported.
- You need to control the relevant logic and process of the air compressor by using the user program, such as determining whether to enable certain functions and controlling parameter modification (the instruction flow needs to be re-controlled) and air compressor feedback.
- Before calling the instruction, you must preset the parameters related to the special PID, including the default values of PID parameters. For example, you can use M8002 to assign values for PID parameters, AC drive parameters, and air compressor parameters. Basically, the control effect is good when the default values are used.

Description of parameters:

S1 specifies the set target value of the PID (in 0.01 Mpa).

S2 is the measured feedback value (in 0.01 Mpa). The user program needs to read the actual value of the device and update this parameter.

S4 is the start unit for storing the operation result. It occupies 60 consecutive D elements or 16-bit word variable arrays. S4 is used for user parameter setting and control.

D1 specifies the PID output value (in 0.01 Hz).

The following table lists the functions and setting methods of the parameters in each unit.

Category	Address	Range	Default Value	Unit	Description
PID parameter setting 30 Ds	S4+0	-	-	-	For system use, not writable!
	S4+1	-	-	-	For system use, not writable!
	S4+2	0 and 1	1	-	Whether there is an air tank 0: No 1: Yes
	S4+3	0 and 1	0	-	Model coefficient calculation enable 0: Hold 1: Re-calculate
	S4+4	0 and 1	0	-	PID direction 0: Forward 1: Reverse
	S4+5	0 and 10000	3000	ms	Control pressure filter time, setting not required
	S4+6	20 and 60	45	%	Percentage of model switching frequency
	S4+7	100 and 30000	1000	ms	PID control calculation interval
	S4+8	0 and 3000	6	s	Motor acceleration/ deceleration and pipeline delay time constant
	S4+9	-	-	-	For system use, not writable!
	S4+10	-32767 and 32767	150	0.01	Model scale factor
	S4+11	-32767 and 32767	300	s	Model time constant
	S4+12	0 and 50	25	1	Fast prediction step 1
	S4+13	0 and 30000	1	1	Fast output weighting factor 1
	S4+14	0 and 50	25	1	Fast prediction step 2
	S4+15	0 and 30000	2	1	Fast output weighting factor 2
	S4+16	0 and 50	35	1	Fast prediction step 3
	S4+17	0 and 30000	5	1	Fast output weighting factor 3
	S4+18	0 and 50	40	1	Fast prediction step 4
	S4+19	0 and 30000	15	1	Fast output weighting factor 4
	S4+20	-32767 and 32767	150	0.01	Non-air tank model scale factor
	S4+21	-32767 and 32767	30	s	Air tank model time constant
	S4+22	0 and 50	9	1	Non-air tank control step
	S4+23	0 and 30000	5	1	Non-air tank output weighting factor
	S4+24	0 and 50	9	1	Slow non-air tank control step
	S4+25	0 and 30000	5	1	Slow non-air tank output weighting factor
	S4+26	-	-	-	For system use, not writable!
	S4+27	-	-	-	For system use, not writable!
	S4+28	-	-	-	For system use, not writable!
	S4+29	-	-	-	For system use, not writable!

Category	Address	Range	Default Value	Unit	Description
Air compressor and AC drive parameter setting (30 D elements)	S4+30	1 and 30000	-	0.01 Hz	Maximum frequency of the AC drive
	S4+31	1 and 30000	-	0.01 Hz	Upper output frequency of the AC drive
	S4+32	0 and 30000	-	0.01 Hz	Lower output frequency of the AC drive
	S4+33	0 and 30000	-	0.01 Hz	Pre-run frequency of the AC drive
	S4+34	-	-	0.01 Hz	Current running frequency of the AC drive, updated in real time. The user program needs to read the AC drive and update this parameter.
	S4+35	0 and 1000	160	0.01Mpa	Maximum pressure of the air compressor, used for calibration. The output frequency must be less than the value of this parameter.
	S4+36	-	-	-	For system use, not writable!
	S4+37	-	-	-	For system use, not writable!
	S4+38	-	-	-	For system use, not writable!
	S4+39	-	-	-	For system use, not writable!
	S4+40	-	-	-	For system use, not writable!
	S4+41	-	-	1	Target calibration value
	S4+42	-	-	-	For system use, not writable!
	S4+43	-	-	1	Feedback calibration value
	S4+44	-	-	-	For system use, not writable!
	S4+45	-	-	-	For system use, not writable!
	S4+46	0 to the maximum value	-	0.01Mpa	Pressure feedback value
	S4+47	0 to the maximum value	-	0.01Mpa	Pressure protection value
	S4+48	0 to the maximum value	-	0.01 Hz	Actual value of the output result
	S4+49	-	-	1	Calibration value of the output result
	S4+50	-	-	-	For system use, not writable!
	S4+51	-	-	-	For system use, not writable!
	S4+52	-	-	-	For system use, not writable!
	S4+53	-	-	-	For system use, not writable!
	S4+54	-	-	-	For system use, not writable!
	S4+55	-	-	-	For system use, not writable!
	S4+56	-	-	-	For system use, not writable!
	S4+57	-	-	-	For system use, not writable!
	S4+58	-	-	-	For system use, not writable!
	S4+59	-	-	-	For system use, not writable!

Do not occupy 60 consecutive elements in the program.

Mode 5: Large-inertia Temperature Control PID Instruction

Note the following:

- This instruction is applicable to long-term temperature control applications with large inertia, in which the control cycle is long and the heating time is more than 100s (typically 200s to 500s).
- You are advised to enable this instruction all the time. Before the first startup, ensure that the temperature difference is 100 degrees. You are advised to clear the user program and re-download and start the target program.
- Before calling this instruction, you need to preset relevant parameters. The M8002 is recommended for value assignment. The parameters to configure include the ambient temperature (S4+13) (which is 0 if it is not specified) and output control parameters such as the output status word (S4+7) (which needs to be implemented by the program). As some states (such as the auto-tuning state) and parameters need to be saved and take effect all the time, the S4 parameter area will not be completely cleared.
- Note in the program that the S4 parameters occupy 90 word elements. Do not reuse these 90 word elements in the program.
- Bit0 of the parameter in S4+6 is used by the user to enable and disable temperature control. Pay attention to it in the program.
- S4+7 stores the output bits. You need to associate the output bits with the external temperature control I/O by programming.
- Retentive registers or variables are strongly recommended to store the S4 parameters.

Description of parameters:

- S1 specifies the set target value of the PID (in 0.1°C).
- S2 is the measured feedback value (in 0.1°C). The user program needs to read the actual value of the device and update this parameter.
- S4 is the start unit for storing the operation result. It occupies 90 consecutive D elements or 16-bit word variable arrays. S4 is used for user parameter setting and control.
- D1 specifies the PID output value (in ms).

The following table lists the functions and setting methods of the parameters in each unit.

Category	Address	Range	Default Value	Unit	Description
PID parameter setting	S4+0	-	-	-	For system use, not writable!
	S4+1	-	-	-	For system use, not writable!
	S4+2	-	5000	ms	Sampling time
	S4+3	0 and 4	0	-	Sampling cycle
	S4+4	-	-	-	Updated target value
	S4+5	-	-	-	Running phase
	S4+6	-	-	-	Status word: Bit0: Temperature control mode enable (user control required) Bit1 to bit4: State control Bit8: Heat charging area (no control required) Bit9: Linear area (no control required) Bit10: Heat discharging area (no control required) Bit12: Auto-tuning completed Bit13: Auto-tuning in progress Bit14: Heating auto-tuning in progress Bit15: Heat dissipation auto-tuning in progress Other bits: For system use, not writable!
	S4+7	-	-	-	Output control word: Bit0: Heating output Bit1: Natural heat dissipation Bit2: Heat dissipation by ventilation Bit8 to bit12: System flag Other bits: For system use, not writable!
	S4+8	-	-	-	Reserved
	S4+9	-	-	-	Reserved
	S4+10	-	-	0.1°C	Target value
	S4+11	-	-	0.1°C	Sampling value
	S4+12	-	-	0.1°C	Difference between the sampling value and target value
	S4+13	-	-	0.1°C	Ambient temperature (**user control required**)
	S4+14	-	-	0.1°C	Auto-tuning start temperature
	S4+15	-	-	0.1°C	Heat dissipation self-check start temperature
	S4+16	-	-	-	Heat dissipation self-check procedure
	S4+17	-	-	-	Reserved
	S4+18	-	-	-	Previous backup value of the average of sampling differences
	S4+19	-	-	-	Average of sampling differences
	S4+20 to S4+29	-	-	-	For system use, not writable!
	S4+30 to S4+39	-	-	-	Sampling data, sampling value, sampling change value, and so on

Category	Address	Range	Default Value	Unit	Description
PID parameter setting	S4+40	-	-	0.1°C	Total temperature rise of the heat charging area
	S4+41	-	-	0.1°C	Start temperature of the heat charging area
	S4+42	-	-	0.1°C/time unit	Slope of the linear area, temperature rise change rate
	S4+43	-	-	0.1°C	Total temperature rise of the heat discharging area
	S4+44	-	-	0.1°C	Start temperature of the heat discharging area
	S4+45	-	-	0.1°C	Predicted temperature
	S4+46	-	-	0.1°C	Heat discharging temperature rise caused by this heat charging
	S4+47	-	-	0.1°C	Heat discharging temperature rise caused by previous (before the current one) heat charging
	S4+48	-	-	0.1°C	Previously accumulated heat discharging temperature rise margin
	S4+49	-	-	0.1°C	Reserved
	S4+50	32-bit data	-	ms	Total time spent in the heat charging area
	S4+52	32-bit data	-	ms	Start time of the heat charging area
	S4+54	32-bit data	-	ms	Total time spent in the heat discharging area
	S4+56	32-bit data	-	ms	Start time of the heat discharging area
	S4+58	32-bit data	-	ms	Total time for PID heat dissipation compensation auto-tuning
	S4+60	32-bit data	-	ms	Start time of PID heat dissipation compensation auto-tuning
	S4+62	32-bit data	-	ms	Heating start time during temperature prediction
	S4+64	32-bit data	-	ms	Parameter self-test cycle start time
	S4+66	32-bit data	-	ms	Heat dissipation start time during parameter self-test
	S4+68	32-bit data	-	ms	Reserved
	S4+70 to S4+79	-	-	-	PID calculation
PID parameter output	S4+80	-	-	ms	Switch base time
	S4+81	-	-	ms	Scaling output
	S4+82	-	-	ms	Integral output
	S4+83	-	-	ms	Differential output
	S4+84	-	-	ms	Total switch ON duration
	S4+85	-	-	ms	Timer for digital control
	S4+86 to S4+89	-	-	-	Reserved

The following table lists the PID error codes in various modes.

PID Error Codes

Error Code	Description
10	Sampling time (TS) less than 0 or out of range
11	Auto-tuning failed
12	Input filter constant object illegal
13	Input scaling coefficient illegal
14	Integral time illegal
15	Differential gain illegal
16	Differential time illegal
17	Range setting error
20	PID result illegal
21	Offset illegal
22	Scaling item illegal
23	Integral item illegal
24	Differential item illegal
100	Current PID mode not supported

Mode 6: Auto-tuning PID instruction

Note the following:

- Before calling the instruction, you must pre-set the relevant parameters. It is recommended that you use values assigned by M8002. Main parameters include:
  - PID mode selection
  - Output period
  - Sampling time
  - Max./Min. output percentage
  - Auto-tuning coefficient
  - Scaling coefficient
  - Target temperature
- Paras occupies 90 word elements. Avoid reusing these word elements in the program.
- Bit0 of Paras+2 is used to start and stop temperature control, which requires user control. Be careful not to operate other bits in the program.
- For Paras+25, compile a program to associate the output bit with the external temperature control output I/O.
- Paras+26 is the percentage of PID digital output, that is, the ratio of output ON time to total output period in percentage.
- For the Setpoint, Mode, and Paras parameters, it is strongly recommended that you use registers retentive at power failure, such as D components after D200 and R components.

The following table lists the functions and setting methods of the parameters in each unit.

Category	Address	Range	Default	Unit	Description
PID parameter setting	PARAS+0	-	-	-	For system use, not writable!
	PARAS+1	-	-	-	Not in use
	PARAS+2	0 to 1	0		Used to start or stop PID. 1: Start, 0: Stop.
	PARAS+3	0 and 1	0	-	Used to start or stop auto-tuning. 1: Start, 0: Stop.
	PARAS+4	32-bit data 1 to 1000000	1000	ms	Output period in ms
	PARAS+6	32-bit data 1 to 1000000	100	ms	Sampling time in ms
	PARAS+8	32-bit data 1 to 1000000	10	-	Scaling coefficient Kp, with one decimal place
	PARAS+10	32-bit data 0 to 1000000	-	0.1s	Integral time Ti, with one decimal place
	PARAS+12	32-bit data 0 to 1000000	-	0.1s	Derivative time Td, with one decimal place
	PARAS+14	0 to 1000	100	0.1%	PID maximum output percentage, with one decimal place. If there are no special requirements, it is recommended that you set this parameter to 1000, that is, 100.0%.
	PARAS+15	0 to 1000	0	0.1%	PID minimum output percentage, with one decimal place. If there are no special requirements, it is recommended that you set this parameter to 0, that is, 0.0%.
	PARAS+16	1-10	5	-	PID auto-tuning coefficient, with one decimal place. The value range is 0.1 to 1.0. The larger the value, the faster the tuning, and the overshoot may be large. If there are no special requirements, it is recommended that you set this parameter to 5, that is, 0.5.
	PARAS+17	0 to 1	0	-	Used to enable or disable the manual mode. TRUE: Manual mode, FALSE: Auto mode.
	PARAS+18	0 to 1000	0	0.1%	Output percentage in manual mode, with one decimal place.
	PARAS+19	0 to 1	0	-	The value 1 indicates that integral separation is canceled.
	PARAS+20	-	-	-	Reserved
	PARAS+21	-	-	-	Reserved
	PARAS+22	-	-	-	Reserved
	PARAS+23	-	-	-	Reserved
	PARAS+24	-	-	-	Reserved

Category	Address	Range	Default	Unit	Description
PID parameter output	PARAS+25	0 to 1	-	-	DO output status. 1: Heating enabled, 0: Heating disabled.
	PARAS+26	0 to 1000	-	0.1%	DO output percentage
	PARAS+27	-	-	0.1°C	Measured value
	PARAS+28	-	-	0.1°C	Setpoint
	PARAS+29	0 to 1000		0.1%	Analog AO output percentage
	PARAS+30	32-bit data	-	0.1	Integral coefficient Ki
	PARAS+32	32-bit data	-	0.1	Differential coefficient Kd
	PARAS+34 to PARAS+35	-	-	-	Reserved
	PARAS+36	32-bit data	-	0.1	Scaling calculation result
	PARAS+38	32-bit data	-	0.1	Integral calculation result
	PARAS+40	32-bit data	-	0.1	Differential calculation result
	PARAS+42	-	-	0.1	Deviation
	PARAS+43	-	-	0.1	Previous deviation
	PARAS+44	-	-	-	Reserved.
	PARAS+45	-	-	-	Max. Pv
	PARAS+46	32-bit data	-	-	Ti time
	PARAS+48	32-bit data	-	-	Kd_AO
	PARAS+50 to PARAS+89	-	-	-	Reserved

Program Example

For the default of Kp, you can first set a value greater than 0. After parameter auto-tuning is completed, set appropriate defaults for Kp, Ti and Td according to the setting results.

Function and Instruction Description

- Before running the PID, check the defaults to ensure that they are reasonable, otherwise the program may encounter a running error.
- Enable the auto-tuning function at room temperature. When auto-tuning is completed, you can obtain the set Kp, Ti and Td values. Let the program continue to run and check whether the controlled object can reach steady state.
- If the current usage needs are met after the controlled object reaches steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults.
- You can manually adjust the Kp, Ti, and Td values to meet your usage needs.
- Note that if you use the DO output, associate the value of PARAS+25 with the DO output.
- If you use the AO analog output, associate the output percentage with the AO output and perform numerical conversion.
- The larger the auto-tuning coefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached, and the temperature curve will be very smooth, without overshoot or with very small overshoot. Unless otherwise required, use the default 5.
  If the customer requires to reach the set temperature faster, increase the auto-tuning coefficient Paras+16 and reset it.

  If the customer needs a smooth curve and small overshoot, decrease the auto-tuning coefficient Paras+16 and reset it.

4 Instruction Description (LiteST)

4.1 Data Operation Instructions

4.1.1 Trigonometric Function Instructions

4.1.1.1 Instruction List

The following table lists the trigonometric function instructions.

Instruction Category	Instruction	Function
Trigonometric function	SIN	Sine operation instruction
	COS	Cosine operation instruction
	TAN	Tangent operation instruction
	ASIN	Arcsine operation instruction
	ACOS	Arccosine operation instruction
	ATAN	Arctangent operation instruction

4.1.1.2 SIN

Return Value Type	Sine operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the sine value needs to be evaluated	-	BYTE/INT/REAL

Table 4–1 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the sine value of the specified angle (in RAD), where:

- S is the angle variable (in RAD), of which the sine value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the sine calculation result after conversion.

Instruction Example

```
real0 := SIN(real1);
```

Evaluate the sine value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

4.1.1.3 COS

Return Value Type	Cosine operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the cosine value needs to be evaluated	-	BYTE/INT/REAL

Table 4–2 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the cosine value of the specified angle (in RAD), where:

- S is the angle variable (in RAD), of which the cosine value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the cosine calculation result after conversion.

Instruction Example

```
real0 := COS(real1);
```

Evaluate the cosine value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

4.1.1.4 TAN

Return Value Type	Tangent operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the tangent value needs to be evaluated	-	BYTE/INT/REAL

Table 4–3 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the tangent value of the specified angle (in RAD), where:

- S is the angle variable (in RAD), of which the tangent value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the tangent calculation result after conversion.

Instruction Example

```
real0 := TAN(real1);
```

Evaluate the tangent value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

4.1.1.5 ASIN

Return Value Type	Arcsine operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the arsine value needs to be evaluated	-	BYTE/INT/REAL

Table 4–4 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the arcsine value of the specified angle (in RAD), where:

- S is the angle variable (in RAD), of which the arcsine value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the arcsine calculation result after conversion.

Note

An operation error will occur if the value in S falls beyond the range of –1.0 to +1.0.

Instruction Example

```
real0 := ASIN(real1);
```

Evaluate the arcsine value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

4.1.1.6 ACOS

Return Value Type	Arccosine operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the arccosine value needs to be evaluated	-	BYTE/INT/REAL

Table 4–5 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the arccosine value of the specified angle (in RAD), where:

- S is the angle variable (in RAD), of which the arccosine value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the arccosine calculation result after conversion.

Note

An operation error will occur if the value in S falls beyond the range of –1.0 to +1.0.

Instruction Example

```
real0 := ACOS(real1);
```

Evaluate the arccosine value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

4.1.1.7 ATAN

Return Value Type	Arctangent operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Angle variable (in RAD), of which the arctangent value needs to be evaluated	-	BYTE/INT/REAL

Table 4–6 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate the arctangent value of the specified angle (in RAD), where:

- S is the angle variable (in RAD), of which the arctangent value needs to be evaluated. It is expressed as an integer or floating number.
- The return value is the arctangent calculation result after conversion.

Instruction Example

```
real0 := ATAN(real1);
```

Evaluate the arctangent value of real1 and store the result in real0.

According to the equation Angle in radians = Angle in degrees x π/180°, an angle of 360° is converted to radians as follows: 360° x π/180° = 2π.

4.1.2 Exponent Operation Instructions

4.1.2.1 Instruction List

The exponent function instructions are listed below.

Instruction Category	Instruction	Function
Exponent operation instruction	LOG	Base-10 logarithm
	LN	Base-e (2.71828) logarithm
	SQRT	Square root operation instruction
	EXPT	Power operation instruction

4.1.2.2 LOG

Return Value Type	Base-10 logarithm		
REAL			
Operand	Description	Range	Data Type
S	Variable to be logarithmic	-	BYTE/INT/REAL

Table 4–7 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to perform the common logarithm operation on data with the base 10. where:

- S is the variable to be logarithmic.
- The return value is the result of the logarithm operation.

Note

The value in S must be positive. If it is 0 or negative, an operation error will occur.

Example

```
real0 := LOG(real1);
```

Perform the base-10 logarithm operation on real1 and store the result in real0.

4.1.2.3 LN

Return Value Type	Base e (2.71828) logarithm		
REAL			
Operand	Description	Range	Data Type
S	Variable to be logarithmic	-	BYTE/INT/REAL

Table 4–8 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to perform the natural logarithm operation on data with the base e (2.71828), where:

- S is the variable to be logarithmic.
- The return value is the result of the logarithm operation.

Note

The value in S must be positive. If it is 0 or negative, an operation error will occur.

Example

```
real0 := LN(real1);
```

Perform the base-e logarithm operation on real1 and store the result in real0.

4.1.2.4 SQRT

Return Value Type	Square root operation instruction		
REAL			
Operand	Description	Range	Data Type
S	Data to be square rooted	-	BYTE/INT/REAL

Table 4–9 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to take the square root of S, where:

- S is the data to be square rooted.
- The return value is the calculation result of the square root operation.

Note

The value in S1 must be positive. If it is negative, an operation error occurs.

Example

```
real0 := SQRT(real1);
```

Take the square root of real1 and store the result in real0.

4.1.2.5 EXPT

Return Value Type	Power operation instruction		
REAL			
Operand	Description	Range	Data Type
S1	Data to be powered	-	BYTE/INT/REAL
S2	Power	-	BYTE/INT/DINT/REAL

Table 4–10 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to evaluate S1 to the power S2, where:

- S1 is the data to be powered.
- S2 is the power.
- The return value is the calculation result of the power operation.

Example

```
real0 := EXPT(real1, real2);
```

Evaluate real1 to the power real2 and store the result in real0.

4.1.3 Explicit Conversion Instructions

4.1.3.1 Instruction List

The explicit conversion function instructions are listed below.

Instruction Category	Instruction	Function
Explicit conversion	INT_TO_<TYPE>	Convert the INT type into the type specified by <TYPE>.
	DINT_TO_<TYPE>	Convert the DINT type into the type specified by <TYPE>.
	BOOL_TO_<TYPE>	Convert the BOOL type into the type specified by <TYPE>.
	REAL_TO_<TYPE>	Convert the REAL type into the type specified by <TYPE>.
	BYTE_TO_<TYPE>	Convert the BYTE type into the type specified by <TYPE>.
	TO_<TYPE>	Convert the variable into the type specified by <TYPE>.

4.1.3.2 INT_TO_<TYPE>

Return Value Type	Convert the INT type into the type specified by <TYPE>.		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	BYTE/INT

The following <TYPE> options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	-	√	√	√	-	-	-

Table 4–11 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	-	-	-	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the INT type into the variable of the type specified by <TYPE>, where:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := INT_TO_BOOL(10)	TRUE
byte0 := INT_TO_BYTE(10)	10

LiteST Code	Result
dint0 := INT_TO_DINT(10)	10
real0 := INT_TO_REAL(10)	10.0

Note

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- When S is an expression, its result will be implicitly converted into the INT type and then converted again by using INT_TO_<TYPE>.

4.1.3.3 DINT_TO_<TYPE>

Return Value Type	Convert the DINT type into the type specified by <TYPE>.		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	BYTE/INT/DINT

The following <TYPE> options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	-	√	√	-	-	-

Table 4–12 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	-	-	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the DINT type into the variable of the type specified by <TYPE>, where:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := DINT_TO_BOOL(10)	TRUE
byte0 := DINT_TO_BYTE(10)	10
int0 := DINT_TO_INT(10)	10
real0 := DINT_TO_REAL(10)	10.0

Note

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- When S is an expression, its result will be implicitly converted into the DINT type and then converted again by using DINT_TO_<TYPE>.

4.1.3.4 BOOL_TO_<TYPE>

Return Value Type	Convert the BOOL type into the type specified by *<TYPE>*.		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	BOOL

The following *<TYPE>* options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	√	-	-	-	-

Table 4–13 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	√	√	√	-	-	-	-	-	-	-	√	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the BOOL type into the variable of the type specified by *<TYPE>*, where:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
byte0 := BOOL_TO_BYTE(TRUE)	1
int0 := BOOL_TO_INT(TRUE)	1
dint0 := BOOL_TO_DINT(TRUE)	1
real0 := BOOL_TO_REAL(TRUE)	1.0

Note

- For the numeric type, if the operand is TRUE, the conversion result is 1. If the operand is FALSE, the result is 0.
- When S is an expression, its result will be implicitly converted into the BOOL type and then converted again by using BOOL_TO_*<TYPE>*.

4.1.3.5 REAL_TO_<TYPE>

Return Value Type	Convert the REAL type into the type specified by *<TYPE>*.		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	BYTE/INT/REAL

The following *<TYPE>* options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	-	√	-	-	-

Table 4–14 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the REAL type into the variable of the type specified by *<TYPE>*, where:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := REAL_TO_BOOL(11.2)	TRUE
byte0 := REAL_TO_BYTE(11.2)	11
int0 := REAL_TO_INT(11.2)	11
dint0 := REAL_TO_DINT(11.2)	11

Note

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- For the numeric type, the general rule for rounding applies during conversion.
- When S is an expression, its result will be implicitly converted into the REAL type and then converted again by using REAL_TO_*<TYPE>*.

4.1.3.6 BYTE_TO_TYPE

Return Value Type	Convert the BYTE type into the type specified by *<TYPE>*.		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	BYTE

The following *<TYPE>* options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	-	√	√	√	√	√	-	-

Table 4–15 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	-	√	-	√	-	-	-	-	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the BYTE type into the variable of the type specified by *<TYPE>*, where:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := BYTE_TO_BOOL(10)	TRUE
int0 := BYTE_TO_INT(10)	10
dint0 := BYTE_TO_DINT(10)	10
real0 := BYTE_TO_REAL(10)	10.0
string0 := BYTE_TO_STRING(10)	'10'

Note

- For the boolean type, if the operand is 0, the conversion result is FALSE. If the operand is any other value, the result is TRUE.
- When S is an expression, its result will be implicitly converted into the BYTE type and then converted again by using BYTE_TO_*<TYPE>*.

4.1.3.7 TO_<TYPE>

Return Value Type	Convert the INT, DINT, REAL, or BOOL type into the type specified by *<TYPE>*.		
<TYPE>			
Operand	Description	Range	Data Type
S	Variable to be converted	-	BYTE/INT/DINT/REAL/BOOL

The following *<TYPE>* options are supported:

ARRAY	BYTE	INT	DINT	REAL	BOOL	STRING	IP	POINTER
-	√	√	√	√	√	-	-	-

Table 4–16 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	√	√	√	√	√	-	√	√	√	√	√	-

Function and Instruction Description

This instruction is used to explicitly convert the variable of the INT, DINT, REAL, or BOOL type into the variable of the type specified by *<TYPE>*, where:

- S is the variable to be converted.
- The return value is the conversion result.

Example

LiteST Code	Result
bool0 := TO_BOOL(11.2)	TRUE
byte0 := TO_BYTE(15)	15
int0 := TO_INT(11.2)	11
dint0 := TO_DINT(TRUE)	1
real0 := TO_REAL(11)	11.0

4.1.4 Comparison Instructions

4.1.4.1 Instruction List

The comparison instructions are listed below.

Instruction Category	Instruction	Function
Comparison instruction	MAX	Max operation
	MIN	Min operation

4.1.4.2 MAX

Return Value Type	Max operation		
BYTE/INT/DINT/REAL			
Operand	Description	Range	Data Type
S1	Variable 1 for comparison	-	BYTE/INT/DINT/REAL
S2	Variable 2 for comparison	-	BYTE/INT/DINT/REAL

Table 4–17 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to compare the values of two BYTE, INT, DINT, or REAL variables and return the larger value, where:

- S1 is variable 1 for comparison.
- S2 is variable 2 for comparison.
- The return value is the larger one of two values.
- The return value type is consistent with the input type.

Note

1. The data types of S1 and S2 must be consistent, or the type of one variable in S1 and S2 can be implicitly converted into the type of the other variable.
2. When the type of one variable in S1 and S2 is implicitly converted into the type of another variable, the return value is the type after the implicit conversion.
3. When the type of one variable in S1 or S2 is the result of a bit operation, the other variable must also be the result of a bit operation, or be a single variable.

Example

```
real0 := MAX(real1, real2);
```

Obtain the larger value of real1 and real2 and store the result in real0.

4.1.4.3 MIN

Return Value Type	Min operation		
BYTE/INT/DINT/REAL			
Operand	Description	Range	Data Type
S1	Variable 1 for comparison	-	BYTE/INT/DINT/REAL
S2	Variable 2 for comparison	-	BYTE/INT/DINT/REAL

Table 4–18 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	√	√	√	-	-
S2	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to compare the values of two BYTE, INT, DINT, or REAL variables and return the smaller value, where:

- S1 is variable 1 for comparison.
- S2 is variable 2 for comparison.
- The return value is the smaller one of two values.
- The return value type is consistent with the input type.

Note

1. The data types of S1 and S2 must be consistent, or the type of one variable in S1 and S2 can be implicitly converted into the type of the other variable.
2. When the type of one variable in S1 and S2 is implicitly converted into the type of another variable, the return value is the type after the implicit conversion.
3. When the type of one variable in S1 or S2 is the result of a bit operation, the other variable must also be the result of a bit operation, or be a single variable.

Example

```
real0 := MIN(real1, real2);
```

Obtain the smaller value of real1 and real2 and store the result in real0.

4.1.5 Shift Instructions

4.1.5.1 Instruction List

The shift instructions are listed below.

Instruction Category	Instruction	Function
Shift instruction	SHL	Shift left operation
	SHR	Shift right operation

4.1.5.2 SHL

Return Value Type	Shift left operation		
BYTE/INT/DINT			
Operand	Description	Range	Data Type
S1	Shift operation variable	-	BYTE/INT/DINT
S2	Number of shifted bits	-	BYTE/INT/DINT

Table 4–19 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	-	-	-	-	-
S2	-	-	-	√	√	-	√	√	√	-	-	-

Function and Instruction Description

This instruction is used to shift the variable S1 in binary form to the left by S2 bits and return the shift result, where:

- S1 is the variable to be shifted.
- S2 is the number of shifted bits.
- The return value is the shift result.
- The type and length of the return value are consistent with those of S1.

Note

1. SHL is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
2. The result returned by SHL can be used in an explicit conversion operation TO_XXX, such as TO_REAL.
3. When the value of S2 is greater than the number of bits in the data type of S1, no exception will occur, and the return value is 16#0.
4. When S1 is an expression, it will be implicitly converted into the type with the maximum length in the expression before the shift operation is performed.
5. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
dint0 := SHL(dint1, 2);
```

Shift dint1 to the left by 2 bits and store the result in dint0.

4.1.5.3 SHR

Return Value Type	Shift right operation		
BYTE/INT/DINT			
Operand	Description	Range	Data Type
S1	Shift operation variable	-	BYTE/INT/DINT
S2	Number of shifted bits	-	BYTE/INT/DINT

Table 4–20 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	-	-	-	√	√	-	√	-	-	-	-	-
S2	-	-	-	√	√	-	√	√	√	-	-	-

Function and Instruction Description

This instruction is used to shift the variable S1 in binary form to the right by S2 bits and return the shift result, where:

- S1 is the variable to be shifted.
- S2 is the number of shifted bits.
- The return value is the shift result.
- The type and length of the return value are consistent with those of S1.

Note

1. SHR is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
2. The result returned by SHR can be used in an explicit conversion operation TO_XXX, such as TO_REAL.
3. When the value of S2 is greater than the number of bits in the data type of S1, no exception will occur, and the return value is 16#0.
4. When S1 is an expression, it will be implicitly converted into the type with the maximum length in the expression before the shift operation is performed.
5. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
dint0 := SHR(dint1, 2);
```

Shift dint1 to the right by 2 bits and store the result in dint0.

4.1.6 Absolute Value Operation Instruction

4.1.6.1 ABS

Return Value Type	Obtain the absolute value of a variable.		
BYTE/INT/DINT/REAL			
Operand	Description	Range	Data Type
S	Variable, of which the absolute value is obtained	-	BYTE/INT/DINT/REAL

Table 4–21 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	-	-	-	√	√	-	√	√	√	√	-	-

Function and Instruction Description

This instruction is used to obtain the absolute value of a variable, where:

- S is the variable, of which the absolute value is obtained.
- The return value is the absolute value of S.
- The return value type is consistent with the input type.

Example

```
int0 := ABS(int1);
```

Obtain the absolute value of int1 and store the result in int0.

4.1.7 Bit Operators

4.1.7.1 Instruction List

The bit operators are listed below.

Instruction Category	Instruction	Function
Bit operator	AND	AND operation
	OR	OR operation
	XOR	XOR operation
	NOT	NOT operation

4.1.7.2 AND

Return Value Type	AND operation		
BYTE/INT/DINT/BOOL			
Operand	Description	Range	Data Type
S1	AND operation variable 1	-	BYTE/INT/DINT/BOOL
S2	AND operation variable 2	-	BYTE/INT/DINT/BOOL

Table 4–22 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	√	√	-	√	√	√	-	√	-
S2	√	√	√	√	√	-	√	√	√	-	√	-

Function and Instruction Description

This instruction is used to perform an AND operation on two variables and return the result of the AND operation, where:

- S1 is operation variable 1 and expressed as an integer.
- S2 is operation variable 2 and expressed as an integer.
- The return value type is consistent with the input type.

Note

1. The data types of S1 and S2 must be consistent.
2. S1 and S2 can only be a single variable, the result of a bit operation, a binary constant, an octal constant, or a hexadecimal constant.
3. AND is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
4. The data type of the return value of the AND operation is the same as that of the input value.
5. When both S1 and S2 are either constant 0 or constant 1, the return value is of type INT.
6. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
int0 := int1 AND int2;
```

Perform an AND operation on int1 and int2 and store the result in int0.

4.1.7.3 OR

Return Value Type	OR operation		
BYTE/IINT/DINT/BOOL			
Operand	Description	Range	Data Type
S1	OR operation variable 1	-	BYTE/IINT/DINT/BOOL
S2	OR operation variable 2	-	BYTE/IINT/DINT/BOOL

Table 4–23 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	√	√	-	√	√	√	-	√	-
S2	√	√	√	√	√	-	√	√	√	-	√	-

Function and Instruction Description

This instruction is used to perform an OR operation on two variables and return the result of the OR operation, where:

- S1 is operation variable 1 and expressed as an integer.
- S2 is operation variable 2 and expressed as an integer.
- The return value type is consistent with the input type.

Note

1. The data types of S1 and S2 must be consistent.
2. S1 and S2 can only be a single variable, the result of a bit operation, a binary constant, an octal constant, or a hexadecimal constant.
3. OR is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
4. The data type of the return value of the OR operation is the same as that of the input value.
5. When both S1 and S2 are either constant 0 or constant 1, the return value is of type INT.
6. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
int0 := int1 OR int2;
```

Perform an OR operation on int1 and int2 and store the result in int0.

4.1.7.4 XOR

Return Value Type	XOR operation		
BYTE/INT/DINT/BOOL			
Operand	Description	Range	Data Type
S1	XOR operation variable 1	-	BYTE/INT/DINT/BOOL
S2	XOR operation variable 2	-	BYTE/INT/DINT/BOOL

Table 4–24 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	√	√	-	√	√	√	-	√	-
S2	√	√	√	√	√	-	√	√	√	-	√	-

Function and Instruction Description

This instruction is used to perform an XOR operation on two variables and return the result of the XOR operation, where:

- S1 is operation variable 1 and expressed as an integer.
- S2 is operation variable 2 and expressed as an integer.
- The return value type is consistent with the input type.

Note

1. The data types of S1 and S2 must be consistent.
2. S1 and S2 can only be a single variable, the result of a bit operation, a binary constant, an octal constant, or a hexadecimal constant.
3. XOR is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
4. The data type of the return value of the XOR operation is the same as that of the input value.
5. When both S1 and S2 are either constant 0 or constant 1, the return value is of type INT.
6. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
int0 := int1 XOR int2;
```

Perform an XOR operation on int1 and int2 and store the result in int0.

4.1.7.5 NOT

Return Value Type	NOT operation		
BYTE/INT/DINT/BOOL			
Operand	Description	Range	Data Type
S	NOT operation variable 1	-	BYTE/INT/DINT/BOOL

Table 4–25 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S	√	√	√	√	√	-	√	√	√	-	√	-

Function and Instruction Description

This instruction is used to perform a NOT operation on a variable and return the result of the NOT operation, where:

- S1 is operation variable 1 and expressed as an integer.
- The return value type is consistent with the input type.

1. S1 can only be a single variable, the result of a bit operation, a binary constant, an octal constant, or a hexadecimal constant.
2. NOT is a bit operation instruction, and the result it returns can only be used with bit operation results, binary constants, octal constants, or hexadecimal constants in an operation.
3. The data type of the return value of the NOT operation is the same as that of the input value.
4. When S is either constant 0 or constant 1, the return value is of type INT.
5. After the returned result is assigned to a variable, the variable can be used for the operations allowed by the variable.

Example

```
int0 := NOT int1;
```

Perform a NOT operation on int1 and store the result in int0.

4.2 Program Logic Instructions

4.2.1 Binary Operation Instruction

4.2.1.1 SEL

Return Value Type	Binary operation instruction		
BYTE/INT/DINT/REAL/BOOL			
Operand	Description	Range	Data Type
S1	Conditional variable	-	BOOL
S2	Selection variable 1	-	BYTE/INT/DINT/REAL/BOOL
S3	Selection variable 2	-	BYTE/INT/DINT/REAL/BOOL

Table 4–26 List of elements

Operand	Bit			Word		Pointer	Constant					Others
	X, Y, M, S, B	Bits of Word Element	Custom Bit Variable	D, R, W	Custom Word Variable	Pointer Variable	BYTE	INT	DINT	REAL	BOOL	
S1	√	√	√	-	-	-	-	-	-	-	√	-
S2	√	√	√	√	√	-	√	√	√	√	√	-
S3	√	√	√	√	√	-	√	√	√	√	√	-

Function and Instruction Description

This instruction is used to select one of the two INT, DINT, REAL, or BOOL variables. When S1 is TRUE, return S3. When S1 is FALSE, return S2. Where:

- S1 is a conditional variable.
- S2 is selection variable 1.
- S3 is selection variable 2.
- The return value is S2 or S3.
- The return value type is consistent with the type of S2 or S3.

Note

1. The data types of S2 and S3 must be consistent, or the type of one variable in S2 and S3 can be implicitly converted into the type of the other variable.
2. When the type of one variable in S2 and S3 is implicitly converted into the type of another variable, the return value is the type after the implicit conversion.
3. When the type of one variable in S1 or S2 is the result of a bit operation, the other variable must also be the result of a bit operation, or be a single variable.

Example

```
dint0 := SEL(bool0,dint1,dint2);
```

When bool0 is TRUE, obtain the value of dint2 and store it in dint0.

When bool0 is FALSE, obtain the value of dint1 and store it in dint0.

5 Appendix

5.1 ASCII Code Conversion

The ASCII code conversion table is as follows:

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Description
0000 0000	00	0	0x00	NUL (null)	Null character
0000 0001	01	1	0x01	SOH (start of headline)	Start of headline
0000 0010	02	2	0x02	STX (start of text)	Start of text
0000 0011	03	3	0x03	ETX (end of text)	End of text
0000 0100	04	4	0x04	EOT (end of transmission)	End of transmission
0000 0101	05	5	0x05	ENQ (enquiry)	Enquiry
0000 0110	06	6	0x06	ACK (acknowledge)	Acknowledgement
0000 0111	07	7	0x07	BEL (bell)	Bell
0000 1000	010	8	0x08	BS (backspace)	Backspace
0000 1001	011	9	0x09	HT (horizontal tab)	Horizontal tab
0000 1010	012	10	0x0A	LF (NL line feed, new line)	Line feed
0000 1011	013	11	0x0B	VT (vertical tab)	Vertical tab
0000 1100	014	12	0x0C	FF (NP form feed, new page)	Form feed
0000 1101	015	13	0x0D	CR (carriage return)	Carriage return
0000 1110	016	14	0x0E	SO (shift out)	Shift out
0000 1111	017	15	0x0F	SI (shift in)	Shift in
0001 0000	020	16	0x10	DLE (data link escape)	Data link escape
0001 0001	021	17	0x11	DC1 (device control 1)	Device control 1
0001 0010	022	18	0x12	DC2 (device control 2)	Device control 2
0001 0011	023	19	0x13	DC3 (device control 3)	Device control 3
0001 0100	024	20	0x14	DC4 (device control 4)	Device control 4
0001 0101	025	21	0x15	NAK (negative acknowledge)	Negative acknowledgement
0001 0110	026	22	0x16	SYN (synchronous idle)	Synchronous idle
0001 0111	027	23	0x17	ETB (end-of-transmission block)	End-of-transmission block
0001 1000	030	24	0x18	CAN (cancel)	Cancel
0001 1001	031	25	0x19	EM (end of medium)	End of medium
0001 1010	032	26	0x1A	SUB (substitute)	Substitute
0001 1011	033	27	0x1B	ESC (escape)	Escape (overflow)
0001 1100	034	28	0x1C	FS (file separator)	File separator
0001 1101	035	29	0x1D	GS (group separator)	Group separator
0001 1110	036	30	0x1E	RS (record separator)	Record separator
0001 1111	037	31	0x1F	US (unit separator)	Unit separator
0010 0000	040	32	0x20	(space)	Space
0010 0001	041	33	0x21	!	Exclamation mark

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Description
0010 0010	042	34	0x22	"	Double quotes
0010 0011	043	35	0x23	#	Hashtag
0010 0100	044	36	0x24	$	Dollar sign
0010 0101	045	37	0x25	%	Percent sign
0010 0110	046	38	0x26	&	Ampersand
0010 0111	047	39	0x27	'	Closed single quote
0010 1000	050	40	0x28	(	Open bracket
0010 1001	051	41	0x29	)	Closing bracket
0010 1010	052	42	0x2A	*	Asterisk
0010 1011	053	43	0x2B	+	Plus
0010 1100	054	44	0x2C	,	Comma
0010 1101	055	45	0x2D	-	Minus/Dash
0010 1110	056	46	0x2E	.	Period
0010 1111	057	47	0x2F	/	Slash
0011 0000	060	48	0x30	0	Character 0
0011 0001	061	49	0x31	1	Character 1
0011 0010	062	50	0x32	2	Character 2
0011 0011	063	51	0x33	3	Character 3
0011 0100	064	52	0x34	4	Character 4
0011 0101	065	53	0x35	5	Character 5
0011 0110	066	54	0x36	6	Character 6
0011 0111	067	55	0x37	7	Character 7
0011 1000	070	56	0x38	8	Character 8
0011 1001	071	57	0x39	9	Character 9
0011 1010	072	58	0x3A	:	Colon
0011 1011	073	59	0x3B		Semicolon
0011 1100	074	60	0x3C	<	Less than
0011 1101	075	61	0x3D	=	Equal sign
0011 1110	076	62	0x3E		Greater than
0011 1111	077	63	0x3F	?	Question mark
0100 0000	0100	64	0x40	@	Email symbol
0100 0001	0101	65	0x41	A	Uppercase A
0100 0010	0102	66	0x42	B	Uppercase B
0100 0011	0103	67	0x43	C	Uppercase C
0100 0100	0104	68	0x44	D	Uppercase D
0100 0101	0105	69	0x45	E	Uppercase E
0100 0110	0106	70	0x46	F	Uppercase F
0100 0111	0107	71	0x47	G	Uppercase G
0100 1000	0110	72	0x48	H	Uppercase H
0100 1001	0111	73	0x49	I	Uppercase I
01001010	0112	74	0x4A	J	Uppercase J
0100 1011	0113	75	0x4B	K	Uppercase K
0100 1100	0114	76	0x4C	L	Uppercase L
0100 1101	0115	77	0x4D	M	Uppercase M
0100 1110	0116	78	0x4E	N	Uppercase N
0100 1111	0117	79	0x4F	O	Uppercase O
0101 0000	0120	80	0x50	P	Uppercase P

Bin (Binary)	Oct (Octal)	Dec (Decimal)	Hex (Hexadecimal)	Abbreviation/Character	Description
0101 0001	0121	81	0x51	Q	Uppercase Q
0101 0010	0122	82	0x52	R	Uppercase R
0101 0011	0123	83	0x53	S	Uppercase S
0101 0100	0124	84	0x54	T	Uppercase T
0101 0101	0125	85	0x55	U	Uppercase U
0101 0110	0126	86	0x56	V	Uppercase V
0101 0111	0127	87	0x57	W	Uppercase W
0101 1000	0130	88	0x58	X	Uppercase X
0101 1001	0131	89	0x59	Y	Uppercase Y
0101 1010	0132	90	0x5A	Z	Uppercase Z
0101 1011	0133	91	0x5B	[	Open square bracket
0101 1100	0134	92	0x5C	\	Backslash
0101 1101	0135	93	0x5D	]	Closing square bracket
0101 1110	0136	94	0x5E	^	Caret
0101 1111	0137	95	0x5F	_	Underscore
0110 0000	0140	96	0x60	`	Opening single quote
0110 0001	0141	97	0x61	a	Lowercase a
0110 0010	0142	98	0x62	b	Lowercase b
0110 0011	0143	99	0x63	c	Lowercase c
0110 0100	0144	100	0x64	d	Lowercase d
0110 0101	0145	101	0x65	e	Lowercase e
0110 0110	0146	102	0x66	f	Lowercase f
0110 0111	0147	103	0x67	g	Lowercase g
0110 1000	0150	104	0x68	h	Lowercase h
0110 1001	0151	105	0x69	i	Lowercase i
0110 1010	0152	106	0x6A	j	Lowercase j
0110 1011	0153	107	0x6B	k	Lowercase k
0110 1100	0154	108	0x6C	l	Lowercase l
0110 1101	0155	109	0x6D	m	Lowercase m
0110 1110	0156	110	0x6E	n	Lowercase n
0110 1111	0157	111	0x6F	o	Lowercase o
0111 0000	0160	112	0x70	p	Lowercase p
0111 0001	0161	113	0x71	q	Lowercase q
0111 0010	0162	114	0x72	r	Lowercase r
0111 0011	0163	115	0x73	s	Lowercase s
0111 0100	0164	116	0x74	t	Lowercase t
0111 0101	0165	117	0x75	u	Lowercase u
0111 0110	0166	118	0x76	v	Lowercase v
0111 0111	0167	119	0x77	w	Lowercase w
0111 1000	0170	120	0x78	x	Lowercase x
0111 1001	0171	121	0x79	y	Lowercase y
0111 1010	0172	122	0x7A	z	Lowercase z
0111 1011	0173	123	0x7B	{	Opening curly bracket
0111 1100	0174	124	0x7C	|	Perpendicular
0111 1101	0175	125	0x7D	}	Closing curly bracket
0111 1110	0176	126	0x7E	~	Tilde
0111 1111	0177	127	0x7F	DEL (delete)	Delete

5.2 Fault Codes

The software tool prompts various categories of fault codes when faults occur in user programming. The following table lists the fault codes and corresponding solutions.

Table 5–1 Fault codes

Fault Code	Message	Description	Troubleshooting
Program			
1500	User program watchdog timed out	The user program execution time is too long and has exceeded the set program watchdog time.	Increase the watchdog time as appropriate, or check whether there is a program block with unexpectedly long execution time in the user program.
1501	Undefined instruction	The instruction is not supported.	Upgrade the PLC firmware to the version that supports the instruction.
1502	Incomplete user program, length error	The user program is incomplete, and the length is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1503	Program authorization protection identifier error. Check whether the identifier matches.	The program authorization protection identifier is incorrect. Check whether the authorization protection identifier of the device is set correctly.	Contact the equipment provider.
1504	User program empty	The user program is empty. There is no valid program.	Re-download the user program.
1505	Block POU identifier error	The block POU identifier is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1510	Subprogram identifier error	The subprogram identifier is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1511	Subprogram type error	The subprogram type is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1512	Subprogram serial number error or out of range	The subprogram serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1513	Incorrect, duplicate, or conflicting subprogram address	The subprogram address is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1514	Interrupt subprogram serial number error or out of range	The interrupt subprogram serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1515	Incorrect, duplicate, or conflicting interrupt subprogram address	The interrupt subprogram address is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1516	Interrupt subprogram edge error (not rising edge or falling edge)	The interrupt subprogram edge is incorrect (not rising edge or falling edge).	Recompile and download the user program, or recompile and download the user program after replacing the software tool.

Fault Code	Message	Description	Troubleshooting
1517	Interrupt timing duration range error in the interrupt subprogram timer	The interrupt timing duration range of the interrupt subprogram timer is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1520	OBprog program identifier error	The OBprog program identifier is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1521	OBprog program type error	The OBprog program type is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1522	OBprog program serial number error or out of range	The OBprog program serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1523	Incorrect, duplicate, or conflicting OBprog program address	The OBprog program address is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1524	OBprog program variable quantity error	The variable quantity of the OBprog program is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1525	OBprog program variable length error	The variable length of the OBprog program is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1526	OBprog program header data error	The header data of the OBprog program is incorrect.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1530	CJ-LBL instruction LBL serial number error or out of range	The LBL serial number of the CJ-LBL instruction is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
1531	Incorrect, duplicate, or conflicting LBL address of CJ-LBL instruction	The LBL address of the CJ-LBL instruction is incorrect, duplicated, or conflicting.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5001	Exception in user program execution or instruction return value error, some instructions not executed	Execution of the user program is abnormal or the return value of the instruction is incorrect, and some instructions are not executed, causing program execution to end abnormally.	Check the logic of the user program for any exception in execution process or execution logic.
5010	CALL instruction subprogram serial number error or out of range	The subprogram serial number of the CALL instruction is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5011	CALL instruction subprogram non-existent or not initialized	The subprogram of the CALL instruction does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5012	CALL instruction subprogram nesting levels out of range or less than or equal to 0	The number of subprogram nesting levels of the CALL instruction is out of range.	Modify the program logic to reduce the subprogram nesting levels.

Fault Code	Message	Description	Troubleshooting
5013	Relationship error returned by the subprogram of the CALL instruction	The subprogram of the CALL instruction returns a relationship error.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5014	Mismatch between subprogram call and subprogram return	Subprogram execution is abnormal. The subprogram call and subprogram return do not match.	Check whether the subprogram call and return are disordered due to the abnormal end of the user program.
5015	Interrupt subprogram undefined	The interrupt subprogram is undefined or does not exist.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5016	Interrupt queue full and interrupt lost in the interrupt subprogram timer	The interrupt queue of the interrupt subprogram timer is full and the interrupt is lost.	Modify the interrupt subprogram attributes or logic, and reduce the number of interrupts as appropriate.
5020	FBFC program serial number error or out of range	The FBFC program serial number is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5021	FBFC program non-existent or not initialized	The FBFC program does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5022	FBFC program variable non-existent or not initialized	The variable of the FBFC program does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5023	FBFC program nesting levels out of range or less than or equal to 0	The number of FBFC program nesting levels is out of range.	Modify the program logic to reduce the FBFC program nesting levels.
5024	Relationship error returned by FBFC program	The FBFC program returns a relationship error.	Check whether the FBFC special instruction is used in the wrong position, or recompile and download the user program.
5025	Mismatch between OBprog program call and program return	OBprog program execution is abnormal. The program call and program return do not match.	Check whether the program call and return are disordered due to the abnormal end of the user program.
5030	CJ-LBL instruction LBL serial number error or out of range	The LBL serial number of the CJ-LBL instruction is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5031	CJ-LBL instruction LBL non-existent or not initialized	The LBL of the CJ-LBL instruction does not exist or is not initialized.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5032	FOR-NEXT instruction nesting levels out of range or less than or equal to 0	The number of nesting levels of the FOR-NEXT instruction is out of range.	Modify the program logic to reduce the FOR-NEXT instruction nesting levels.
5033	FOR-NEXT instruction loops out of range or less than or equal to 0	The number of FOR-NEXT instruction loops is out of range or less than or equal to 0.	Modify the program logic to change the number of FOR-NEXT instruction loops.
5034	FOR-NEXT instruction loops equal to 0	The number of FOR-NEXT instruction loops is 0.	Modify the program logic to change the number of FOR-NEXT instruction loops.
5035	FOR and NEXT not paired	The FOR and NEXT instructions are not paired.	Check whether the disorder is caused by abnormal stop of the user program.

Fault Code	Message	Description	Troubleshooting
5080	Array subscript access out of bounds	The array access subscript is greater than the maximum array subscript value, and the subscript value in use has been changed to the maximum array subscript value.	Double-click the fault code to go to the corresponding program position to modify the subscript value.
5081	Division-by-zero protection, divisor 0 replaced by 1	The division-by-zero protection is triggered and the divisor 0 is replaced by 1 automatically.	Double-click the fault code to go to the corresponding program position to modify the divisor.
5082	Long-time no response from program loop	The program loop has no response for a long time.	Double-click the fault code to go to the corresponding program position to modify the loop statement.
5083	Array subscript access out of bounds	The array access subscript is less than 0, and the subscript value in use has been changed to 0.	Double-click the fault code to go to the corresponding program position to modify the subscript value.
5084	Invalid data	The floating-point data is invalid.	Check whether the input values of functions such as LN, LOG, SQRT are legal.
5101	Instruction parameter variable address error, or variable non-existent	The address of the parameter variable of the instruction is incorrect, or the variable does not exist.	Check whether the address of the parameter variable of the instruction is normal and whether the variable exists.
5102	Instruction parameter variable size error, or variable non-existent or out of range	The size of the parameter variable of the instruction is incorrect. The variable does not exist or is out of range.	Check whether the data length of the parameter variable of the instruction is out of range.
5103	xxxx0001 error	xxxx0001 error occurs.	Recompile and download the user program by using the software tool.
5104	Instruction parameter sequence error or relationship error	The instruction parameter sequence or relationship is incorrect.	Check whether the parameter sequence or relationship of the instruction is correct.
5105	String data error or length error in string instruction	The character string data or length of the string instruction is incorrect.	Check whether the character string data of the string instruction is illegal.
5110	Pointer serial number error or out of range	The serial number of the Pointer is incorrect or out of range.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
5111	Pointer not initialized or not pointing to a valid data variable	The Pointer is not initialized or does not point to a valid data variable.	Check whether the Pointer is initialized and whether it points to a valid variable address.
5112	Variable pointed to by the Pointer non-existent or out of range	The variable pointed to by the Pointer does not exist or is out of range.	Check the variable address pointed to by the Pointer or initialize the Pointer again.
5113	Pointer offset out of range	The offset of the Pointer is out of range.	Check whether the offset of the Pointer is too large. If yes, reduce the offset.
5114	Variable pointed to by the Pointer execution result non-existent or out of range	The variable pointed to by the execution result of the Pointer does not exist or is out of range.	Check whether the variable address pointed to by the execution result of the Pointer exists and whether it is out of range.
5120	Counter instruction instantiation failed	Failed to instantiate the counter instruction.	Recompile and download the user program.

Fault Code	Message	Description	Troubleshooting
5121	Counter instruction comparand error or out of range	The comparand of the counter instruction is incorrect or out of range.	Check whether the comparand of the counter instruction is incorrect or out of range.
5130	Timer instruction instantiation failed	Failed to instantiate the timer instruction.	Recompile and download the user program.
5131	Timer instruction comparand error or out of range	The comparand of the timer instruction is incorrect or out of range.	Check whether the comparand of the timer instruction is incorrect or out of range.
5140	Number of SFC STL parallel branch/parallel recombination/selective branch/selective recombination lines out of range	The number of SFC STL parallel branch/parallel recombination/selective branch/selective recombination lines is out of range.	Ensure that the number of SFC STL parallel branch/ parallel recombination/selective branch/selective recombination lines is within the specified range.
5150	Function block instruction instantiation failed	Failed to instantiate the function block instruction.	Recompile and download the user program.
5160	Array subscript variable code error or non-existent	The subscript variable code of the array is incorrect or does not exist.	Recompile and download the user program.
5161	Array subscript variable data error or out of range	The subscript variable of the array is incorrect or out of range.	Modify the value of the subscript variable so that the array falls within the allowable range.
5600	SerialSR instruction instantiation failed	Failed to instantiate the SerialSR instruction.	Recompile and download the user program.
5601	SerialSR instruction port ID out of range	The port ID of the SerialSR instruction is out of range.	Modify the port ID of the SerialSR instruction.
5602	SerialSR instruction protocol error	The protocol of the SerialSR instruction is incorrect.	Set the free protocol for the serial port by using the software tool.
5603	SerialSR instruction port conflict	Multiple instructions call the SerialSR instruction at the same time, and the instruction that fails to preempt the port reports an error.	Modify the instruction scheduling timing to implement time division multiplexing.
5604	SerialSR instruction TX data length out of range or less than 0	The TX data length of the SerialSR instruction is out of range or less than 0.	Check whether the TX data length of the SerialSR instruction is out of range or less than 0.
5605	SerialSR instruction TX data buffer error	Failed to obtain the TX data buffer of the SerialSR instruction.	Enable this instruction again.
5606	SerialSR instruction RX data length out of range or less than 0	The RX data length of the SerialSR instruction is out of range or less than 0.	Check whether the RX data length of the SerialSR instruction is out of range or less than 0.
5607	SerialSR instruction RX data buffer error	Failed to obtain the RX data buffer of the SerialSR instruction.	Enable this instruction again.
6580	Invalid axis ID in the CANopen axis instruction	The axis ID specified in the CANopen axis instruction is invalid.	Modify the axis ID.

Fault Code	Message	Description	Troubleshooting
6701	Invalid memory address: element or variable non-existent	The memory address is invalid. The element or variable to access does not exist.	Modify the instruction parameter to use a valid element or variable.
6705	Invalid memory size: memory non-existent or out of range	The memory size is invalid. The number of elements or variables to access is too large or out of range.	Modify the instruction parameter to adjust the number of elements or variables.
6706	Improper data or data out of range	The instruction parameter is improper or out of the allowable range.	Refer to the instructions guide to modify the instruction parameter value.
6711	Invalid variable address: variable non-existent	The variable address is invalid. The element or variable to access does not exist.	Modify the instruction parameter to use a valid element or variable.
6712	Invalid variable size: variable out of range	The variable size is invalid. The number of elements or variables to access is too large or out of range.	Modify the instruction parameter to adjust the number of elements or variables.
6713	Invalid variable encoding	The variable encoding is invalid.	Recompile and download the user program, or recompile and download the user program after replacing the software tool.
CPU			
1011	FPGA initialization failed	FPGA initialization failed.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
1012	Interrupt initialization failed	Interrupt initialization failed.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
1013	Timer interrupt initialization failed	Failed to initialize the timer interrupt of the user program.	Restart the device and try again, or replace the device and return the faulty device to the factory for repair.
5200	Error in data retention upon power failure	An error occurs to data retention upon power failure.	Check whether the function of data retention upon power failure works properly.
5238	2038 problem imminence warning	The device will not work normally after 11:14:07 on January 19, 2038 (UTC+8).	Change the device time.
5250	Low RTC battery voltage	The battery voltage of the RTC clock is low. If the device is powered off at this time, the system time will be restored to the initial value.	Replace the battery of the RTC clock while keeping the device powered on.
5900	Network down: Ethernet IP address conflict	When the device connects to the network or starts running after stop, or when its IP address is modified, it detects whether its IP address is used by other devices in the current network. If yes, the device automatically shuts down the network to avoid conflict.	Change the device IP address.

Fault Code	Message	Description	Troubleshooting
		Local I/O	
5300	Initialization failed	Initialization failed.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
5301	Invalid DI filter parameter configuration	The DI filter parameter configuration data is invalid.	Modify the DI filter parameter configuration data.
		Extension Module	
5400	Failed to initialize extension module interface hardware	The hardware of the extension module interface is faulty, which causes the initialization to fail.	The device hardware is faulty. Replace the device and return the faulty device to the factory for repair.
5401	Failed to parse extension module configuration data	The configuration data of the extension module cannot be parsed correctly because its format does not meet requirements.	Clear the compilation information using AutoShop and recompile and download the program. If the problem persists, delete the module configurations and add modules and configurations again one by one.
5402	Failed to initialize extension module interface slot	The slot of the extension module interface is faulty, which causes the initialization to fail.	1. Check whether the extension module interface slot is short-circuited. If yes, eliminate the short circuit. 2. Check whether the installed module hardware works properly. If not, replace the module.
5403	Extension module not installed	The extension module is configured but not installed.	Install the extension module as required, or modify the configuration of the extension module.
5404	Module installed inconsistent with module configured	The module installed in the slot must be inconsistent with the configured module; otherwise, it cannot work properly.	Install the extension module as required and modify module configuration accordingly to ensure consistency.
5405	Extension module interface hardware exception	The extension module interface is abnormal.	1. Check whether the extension module interface slot is short-circuited. If yes, eliminate the short circuit. 2. Check whether the installed module hardware works properly. If not, replace the module.
5406	Extension module interface software error	The extension module interface software is abnormal.	1. Upgrade the PLC firmware. 2. If the problem persists after the firmware upgrade, replace the device and return the faulty device to the factory for repair.
5411	Module in the slot not powered	The module requires external power supply to function properly, but the external power supply is not on.	Connect the external power supply correctly according to the module specifications.
5412	Slot module hardware fault	The module has an internal fault and cannot work properly.	Replace the module and return the faulty module to the factory for repair.
5413	Slot module over-temperature	The module detected a high internal temperature that may lead to malfunction.	1. Do not install the module in an environment that does not meet the relevant temperature requirements. 2. Replace the module and return the faulty module to the factory for repair.

Fault Code	Message	Description	Troubleshooting
5419	Slot module channel input or output overflow	For the input channel, the input signal has exceeded the upper sampling threshold. Sampling cannot be performed properly, and there is a possibility that the input port may be burned. For the output channel, the output value of the corresponding channel has exceeded the set upper threshold, and signals cannot be output properly.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Output channel: Check the set output value and ensure that the set output is within the set range. If the set range cannot meet requirements, modify it as appropriate.
5420	Slot module channel input or output underflow	For the input channel, the input signal has fallen below the lower sampling threshold, and sampling cannot be performed properly. For the output channel, the output value of the corresponding channel has fallen below the set lower threshold, and signals cannot be output properly.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Output channel: Check the set output value and ensure that the set output is within the set range. If the set range cannot meet requirements, modify it as appropriate.
5421	Slot module channel input upper limit exceeded or current output disconnected	For the input channel, the input signal has exceeded the upper sampling threshold. At this time, the signal can be sampled normally but the accuracy cannot be guaranteed. For the current output channel, the output port is not connected to the load or the impedance of the connected load is too large, so that the current cannot be output normally.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Current output channel: Ensure that the load of the output port is connected properly and reliably, and that the load impedance is within the range specified in the module specifications.
5422	Slot module channel input lower limit exceeded or voltage output short-circuited	For the input channel, the input signal has fallen below the lower sampling threshold. At this time, the signal can be sampled normally but the accuracy cannot be guaranteed. For the voltage output channel, the output port is possibly short-circuited or the impedance of the connected load is too small, so that the voltage cannot be output normally.	Input channel: Check the actual input signal value. If the signal input to this channel has exceeded the set sampling range under normal working conditions, modify the sampling range as appropriate. If the signal is abnormal, check the output device or instrument of the signal. Voltage output channel: Ensure that the load of the output port is connected properly and reliably, and that the load impedance is within the range specified in the module specifications.

Fault Code	Message	Description	Troubleshooting
5423	Slot module channel input disconnected or output hardware faulty	For the input channel, no input signal is connected to the input port or the input signal is too weak and cannot be detected or sampled. For the output channel, the channel hardware is faulty and may have burned out.	Input channel: Ensure that the signal of the input port is normal and valid and is connected properly and reliably. Output channel: Replace the module and return the faulty module to the factory for repair.
Local Encoder Axis			
6300	Input device not assigned or assigned input device invalid	The local encoder axis must be assigned with a high-speed counter, and each high-speed counter can only be assigned to one axis, otherwise the axis cannot work properly.	Assign a high-speed counter that has not been assigned yet in "Input Device" on the "Basic Settings" page of the axis.
6301	Axis unit conversion configuration invalid	After a high-speed counter is assigned to an axis, its count value (pulse unit) is converted into the equivalent in user unit (Unit) according to the unit conversion setting parameter. If the number of pulses per revolution of the encoder, the displacement of the encoder per revolution, or the gear ratio of the transmission device is set incorrectly, the axis cannot work properly.	Check the settings on the "Unit Conversion Settings" page of the axis and correct the parameter values.
6302	Axis software limit or revolution cycle configuration invalid	In linear mode, the negative limit must be less than 0, and the positive limit must be greater than 0. In rotary mode, the revolution cycle must be greater than 0. Since the high-speed counter is a 32-bit counter, the negative limit, positive limit, and revolution cycle must be 32-bit integers in the range of [–2147483648, +2147483647] after being converted into pulse units.	Linear mode: Modify the positive and negative limits to ensure that the negative limit is less than 0, the positive limit is greater than 0, and they are 32-bit integers in the range of [–2147483648, +2147483647] after being converted into pulse units. Rotary mode: Modify the revolution cycle to ensure that it is greater than 0 and is a 32-bit integer in the range of [–2147483648, +2147483647] after being converted into pulse units.

Fault Code	Message	Description	Troubleshooting
6303	Axis counting mode or signal source configuration invalid	The high-speed counter supports the following counting modes and signal sources: A/B phase frequency multiplication by 1: X0-A phase, X1-B phase, X2-A phase, X3-B phase A/B phase frequency multiplication by 2: X0-A phase, X1-B phase, X2-A phase, X3-B phase A/B phase frequency multiplication by 4: X0-A phase, X1-B phase, X2-A phase, X3-B phase CW/ CCW: X0-CW, X1-CCW, X2-CW, X3-CCW Pulse +direction: X0-pulse, X1-direction, X2-pulse, X3-direction	Select a supported counting mode and signal source.
6304	Axis preset function: input terminal invalid	The preset function supports the input terminals X0, X1, X2, X3, X4, X5, X6, and X7.	Select an input terminal supported by the preset function.
6305	Axis probe 1: input terminal invalid	Probe 1 supports the input terminals X0, X1, X2, X3, X4, X5, X6, and X7.	Select an input terminal supported by probe 1.
6306	Axis probe 2: input terminal invalid	Probe 2 supports the input terminals X0, X1, X2, X3, X4, X5, X6, and X7.	Select an input terminal supported by probe 2.
6307	Axis comparison output: terminal invalid	The comparative output supports the output terminals Y0, Y1, Y2, and Y3.	Select an output terminal supported by the comparison output.
6308	Axis comparison output: pulse width invalid	When the unit is ms, the time range is 0.1 ms to 6553.5 ms. When the unit is Unit, the set value must fall between 1 and 65535 after being converted into pulse units.	Modify the pulse width to ensure that it is within the allowable range.
CANlink			
6400	Station address conflict: Station address already exists in the network.	In CANlink communication, the addresses of all stations connected to the network must be unique. Address conflict detection is performed after a device node is powered on and initialized or the station address is modified. If the address is duplicated, a fault is reported and all CANlink bus activities of the node are stopped.	Change the station address to ensure that there are no duplicate addresses in the network.

Fault Code	Message	Description	Troubleshooting
6401	Slave offline	Failed to communicate with the slave because it is offline.	Check whether the CAN network connection works properly. Ensure that the connection is reliable without short circuit or open circuit, CANH and CANL are not reversely connected, and the terminal resistance is normal.
6411	Slave configuration exception response (1) "Undefined encoding used"	During configuration of a slave, the slave returns exception response (1) "Undefined encoding used".	Check whether the type/model of the connected device is consistent with the configuration.
6412	Slave configuration exception response (2) "Configured index exceeds the maximum value supported by the node"	During configuration of a slave, the slave returns exception response (2) "Configured index exceeds the maximum value supported by the node".	Check whether the type/model of the connected device is consistent with the configuration.
6413	Slave configuration exception response (3) "Register address non-existent or inaccessible"	During configuration of a slave, the slave returns exception response (3) "Register address non-existent or inaccessible".	Check whether the type/model of the connected device is consistent with the configuration.
6415	Slave configuration exception response (5) "Register data length invalid"	During configuration of a slave, the slave returns exception response (5) "Register data length invalid".	Check whether the type/model of the connected device is consistent with the configuration.
6416	Waiting for slave configuration command response timed out	During configuration of a slave, waiting for slave response timed out.	Check whether the type/model of the connected device is consistent with the configuration.
6421	Slave synchronization exception response (1) "Illegal command code"	When a synchronization command is sent to a slave, the slave returns exception response (1) "Illegal command code".	Check whether the type/model of the connected device is consistent with the configuration.
6422	Slave synchronization exception response (2) "Register address non-existent or inaccessible"	When synchronization data is sent to a slave, the slave returns exception response (2) "Register address non-existent or inaccessible".	Check whether the type/model of the connected device is consistent with the configuration.
6423	Slave synchronization exception response (3) "Value beyond allowable range"	When synchronization data is sent to a slave, the slave returns exception response (3) "Value beyond allowable range".	1. Check whether the set value in the corresponding register address has exceeded the allowed range. 2. Check whether the type/model of the connected device is consistent with the configuration.
6424	Slave synchronization exception response (4) "Operation unreachable or not allowed in the current state"	When synchronization data is sent to a slave, the slave returns exception response (4) "Operation unreachable or not allowed in the current state".	Check whether the type/model of the connected device is consistent with the configuration.
6425	Slave synchronization exception response (5) "Data length invalid"	When synchronization data is sent to a slave, the slave returns exception response (5) "Data length invalid".	Check whether the type/model of the connected device is consistent with the configuration.

Fault Code	Message	Description	Troubleshooting
6426	Waiting for slave synchronization command response timed out	Waiting for slave response to a synchronization command timed out.	Check whether the type/model of the connected device is consistent with the configuration.
CANopen			
6401	Node offline	Failed to communicate with the node because it is offline.	Check whether the CAN network connection works properly. Ensure that the connection is reliable without short circuit or open circuit, CANH and CANL are not reversely connected, and the terminal resistance is normal.
Modbus Master			
5500	8-bit data required for Modbus RTU serial port	The Modbus RTU serial port only supports 8-bit data.	Use 8-bit data for Modbus RTU serial port.
6001	Slave returned exception response (01) "Illegal function code"	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to new devices, and is not implementable in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example, because it is unconfigured and is being asked to return register values.	Check whether the server (or slave) supports the function code.
6002	Slave returned exception response (02) "Illegal data address"	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with the offset 96 and the length 4 will succeed, but a request with the offset 96 and the length 5 will result in exception code 02.	Check whether the corresponding function code of the server (or slave) supports all the addresses accessed by this configuration.

Fault Code	Message	Description	Troubleshooting
6003	Slave returned exception response (03) "Illegal data"	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.	Check whether the value is within the allowed range.
6004	Slave returned exception response (04) "Slave device fault"	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.	Check whether slave is abnormal or faulty.
6128	Response station number and requested station number mismatch	After the master sends a request frame, the station number in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6129	Response function code and requested function code mismatch	After the master sends a request frame, the function code in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6130	Response data address and requested data address mismatch	After the master sends a request frame, the data address in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6131	Response data value and requested data value mismatch	After the master sends a request frame, the data value in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.

Fault Code	Message	Description	Troubleshooting
6240	Cache address mapping in configuration invalid	The cache address mapping in the configuration is invalid and the configuration cannot be executed correctly.	Modify the cache address mapping in the configuration to a valid variable or element address.
6255	Request timed out	After sending a request frame, if the master does not receive a response from the slave within the specified timeout period, it retries according to the set number of retries. When the retry attempts exceed the set number, the master considers the slave abnormal and reports a request timeout error.	1. Ensure that the communication network cable is connected reliably. 2. Ensure that the slave station number is consistent with the configured slave station number. 3. Modify the timeout period to ensure that the master can receive the response frame within the timeout period. 4. Check whether the connected slave is a normal Modbus slave.
Modbus TCP Master			
6000	Configuration disconnected	The Modbus TCP client fails to establish a TCP connection with the server.	1. Ensure that the communication network cable is connected reliably. 2. Check whether the slave IP address and port ID are consistent with the configuration. 3. If the client and server are connected through a network bridge, router, or gateway, make sure that the client and server gateways are set correctly.
6001	Slave returned exception response (01) "Illegal function code"	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to new devices, and is not implementable in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example, because it is unconfigured and is being asked to return register values.	Check whether the server (or slave) supports the function code.

Fault Code	Message	Description	Troubleshooting
6002	Slave returned exception response (02) "Illegal data address"	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with the offset 96 and the length 4 will succeed, but a request with the offset 96 and the length 5 will result in exception code 02.	Check whether the corresponding function code of the server (or slave) supports all the addresses accessed by this configuration.
6003	Slave returned exception response (03) "Illegal data"	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does not mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.	Check whether the value is within the allowed range.
6004	Slave returned exception response (04) "Slave device fault"	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.	Check whether slave is abnormal or faulty.
6128	Response station number and requested station number mismatch	After the master sends a request frame, the station number in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6129	Response function code and requested function code mismatch	After the master sends a request frame, the function code in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.

Fault Code	Message	Description	Troubleshooting
6130	Response data address and requested data address mismatch	After the master sends a request frame, the data address in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6131	Response data value and requested data value mismatch	After the master sends a request frame, the data value in the received response frame is inconsistent with that in the transmitted request frame.	Check whether the connected slave is a normal Modbus slave.
6240	Cache address mapping in configuration invalid	The cache address mapping in the configuration is invalid and the configuration cannot be executed correctly.	Modify the cache address mapping in the configuration to a valid variable or element address.
6255	Request timed out	After sending a request frame, if the master does not receive a response from the slave within the specified timeout period, it retries according to the set number of retries. When the retry attempts exceed the set number, the master considers the slave abnormal and reports a request timeout error.	1. Ensure that the communication network cable is connected reliably. 2. Ensure that the slave station number is consistent with the configured slave station number. 3. Modify the timeout period to ensure that the master can receive the response frame within the timeout period. 4. Check whether the connected slave is a normal Modbus slave.
EtherCAT			
8001	Failed to read master configuration	Failed to read the master configuration information.	Check whether the board software and software tool versions match.
8002	Failed to obtain slave configuration parameters	Failed to obtain slave configuration parameters.	Check whether the board software and software tool versions match.
8003	EtherCAT startup timed out	EtherCAT startup timed out.	1. Check whether the network is properly connected. 2. Check whether the connected slave is consistent with the configuration. 3. Check whether the slave type matches.
8004	Failed to request the master	Failed to request the master.	Restart the PLC.
8200	Failed to write slave startup parameters to SDO	Failed to write the slave startup parameters to the SDO.	1. Check whether there is an object dictionary that is not supported by the slave in the startup parameter list. 2. Check whether the value of the object dictionary is out of range.
8201	Slave lost during operation	The slave is lost during operation.	1. Check whether the network with the slave is disconnected. 2. Check whether the slave is powered off.

Fault Code	Message	Description	Troubleshooting
8202	Slave state machine switched to non-OP mode	The slave state machine is switched to non-OP mode.	1. Check whether the network with the slave is disconnected. 2. Check whether the slave is powered off.
8203	Slave state machine switching failed	Slave state machine switching failed.	-
8204	Slave type mismatch	The slave type is incorrect.	1. Check whether the network cable is reversely connected. 2. Check whether the connected device matches the configuration.
8205	PDO address error	The PDO address is incorrect.	1. Check whether the memory runs out. 2. Check whether the background and board software versions match. 3. Power off and restart the PLC.
8206	PDO length error	The PDO length is incorrect.	Check whether the background and board software versions match.
8301	Failed to switch to INIT state	Failed to switch to INIT state.	Check whether the slave station machine supports state transition.
8302	Failed to switch to PerOP state	Failed to switch to PerOP state.	Check whether the slave supports the CoE protocol.
8304	Failed to switch to SafeOP state	Failed to switch to SafeOP state.	Check whether the PDO communication configuration is correct.
8308	Failed to switch to OP state	Failed to switch to OP state.	1. Check the network communication quality. 2. Check whether the EtherCAT task cycle is appropriate.
8310	FMMU unit configuration error	An FMMU unit configuration error occurs.	Check whether the slave supports the FMMU unit.
8311	Email configuration error	An email configuration error occurs.	Check whether the slave supports the SM unit.
8400	ECTA configuration error	The ECTA configuration is incorrect.	Check whether the configured extension module is consistent with the actually connected extension module.
8401	ECTA hardware error	An ECTA hardware error occurs.	1. Check whether the connection between the ECTA and the extension module is loose. 2. Replace the ECTA.
8402	ECTA extension module error	An ECTA extension module error occurs.	1. Locate the extension module with the ERR indicator on. 2. Read the diagnosis object dictionary of the faulty module by using ETC_ReadParameter_CoE. 3. Determine the fault type based on description of the diagnosis object dictionary of the extension module in the ECTA application guide and eliminate the fault.
Motion Control Axis			
9001	Local axis emergency stop active	The emergency stop terminal input is active, and the pulse output is stopped.	Disable the emergency stop terminal input and then call the MC_Reset instruction to reset the fault.
9003	Overspeed	The pulse output frequency exceeds 200 kHz.	Check whether the pulse frequency obtained by multiplying the target velocity by the gear ratio exceeds 200 kHz.

Fault Code	Message	Description	Troubleshooting
9020	Homing error	The negative limit is not mapped.	Map the negative limit on the configuration interface.
9021	Homing error	The positive limit is not mapped.	Map the positive limit on the configuration interface.
9022	Homing error	The home signal is not mapped.	Map the home switch on the configuration interface.
9023	Homing error	1. The output frequency exceeds 200 kHz when the axis runs at the homing velocity. 2. The output frequency exceeds 200 kHz when the axis runs at the homing approach velocity.	1. Modify the unit conversion setting to ensure that the homing velocity and homing approach velocity do not exceed 200 kHz. 2. Change the homing velocity to ensure that the output frequency does not exceed 200 kHz. 3. Change the homing approach velocity to ensure that the output frequency does not exceed 200 kHz.
9024	Homing error	Homing timed out.	1. Check whether the limit signal and home signal can be connected normally. 2. Check whether the homing timeout time is too short.
9025	Homing error	The limit signal is incorrect during homing.	Check whether the limit signal that is not applicable to the current homing mode is triggered.
9030	Limiting active	The limit signal input is active during positioning.	Check whether the limit is reached during normal running.
9031	Synchronization error	The target number of transmitted pulses and the actual number of transmitted pulses do not match.	Check whether the limit is reached during normal positioning.
9101	Axis type error or non-existent	1. The type of the axis specified by AxisID is incorrect. 2. The axis specified by AxisID does not exist.	1. Check whether the instruction supports the axis specified by AxisID. 2. Check whether the axis specified by AxisID exists.
9102	Axis configuration failed	1. The axis configuration data is lost. 2. The axis configuration parameters are improper.	Check whether the parameters are correct.
9103	MC_Reset called when the axis is not faulty	The MC_Reset instruction is called when the axis is not faulty.	Check whether the MC_Reset instruction is called when the axis is not switched to ErrorStop state.
9104	Axis state unknown when MC_ReadStatus is called	The axis is in unknown state when the MC_ ReadStatus instruction is called.	Check whether the current state of the axis is uncontrollable by using the online monitoring function.
9105	Current position setting not allowed	The MC_SetPosition instruction is called during running or stop.	Set the current position when the axis is in StandStill, Poweroff, or ErrorStop state.
9106	Stopping upon fault	The axis is stopping upon a fault.	Execute the instruction after stop upon fault is completed, the fault is resolved, and the reset instruction is executed.
9107	Improper parameter	The parameters are improper.	Check whether the parameters on the left of the instruction are set properly.

Fault Code	Message	Description	Troubleshooting
9108	Improper PLCOpen state machine	The PLCOpen state machine is improper.	Check whether the current PLCOpen state machine satisfies the execution conditions for this instruction. If not, call the relevant instruction to switch the axis to the required state.
9110	MC_Stop called repeatedly during stop	The MC_Stop instruction is called repeatedly during stop.	Trigger only one MC_Stop instruction at a time.
9111	Instruction linked list lost	The instruction linked list is lost.	1. Check whether the background version and board version match. 2. Contact the manufacturer.
9112	AxisID changed	The value of AxisID is changed while the instruction flow is active.	Do not change the axis number while the flow is active for Enable instructions such as MC_Power and MC_Jog.
9113	Reset by MC_Reset timed out	Reset by executing the MC_Reset instruction timed out.	1. Check whether the drive fault can be reset. 2. Check whether the axis fault type supports reset.
9114	Failed to write to 0x6060	The axis fails to write to 0x6060.	1. Check for interference in network communication. 2. Check whether the slave supports the object dictionary 0x6060.
9115	MC_Halt called when the axis is in Stopping state	The MC_Halt instruction is called when the axis is in Stopping state.	Do not call the MC_Halt instruction when the axis is in Stopping state.
9116	Axis in online commissioning mode	The current axis is in online commissioning mode.	Check whether the current axis is in online commissioning mode. PLC motion control instructions are invalid in online commissioning mode.
9118	Maximum acceleration (deceleration) exceeded	The acceleration (deceleration) of the instruction exceeds the maximum acceleration.	Check whether the acceleration (deceleration) of the instruction exceeds the maximum acceleration.
9119	MC_Jog target velocity exceeded maximum jogging velocity	The target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.	Check whether the target velocity of the MC_Jog instruction exceeds the maximum jogging velocity.
9120	Target velocity exceeded maximum velocity	The target velocity exceeds the maximum velocity.	Check whether the target velocity of the instruction exceeds the maximum velocity.
9121	Jog forward and reverse motion signals both active	The forward and reverse motion signals of the jog instruction are both active.	Ensure that the forward and reverse motion signals of the jog instruction are not active at the same time.
9122	Control word not mapped to EtherCAT bus axis	The control word is not mapped to the EtherCAT bus axis.	Add the control word in the PDO and map it to the axis.
9123	Target position not mapped to EtherCAT bus axis	The target position is not mapped to the EtherCAT bus axis.	Add the target position in the PDO and map it to the axis.
9124	Target torque not mapped to EtherCAT bus axis	The target torque is not mapped to the EtherCAT bus axis.	Add the target torque in the PDO and map it to the axis.
9125	Status word not mapped to EtherCAT bus axis	The status word is not mapped to the EtherCAT bus axis.	Add the status word in the PDO and map it to the axis.

Fault Code	Message	Description	Troubleshooting
9126	Current position not mapped to EtherCAT bus axis	The current position is not mapped to the EtherCAT bus axis.	Add the feedback position in the PDO and map it to the axis.
9127	0x60fd not mapped to EtherCAT bus axis	0x60fd is not mapped to the EtherCAT bus axis.	Add 0x60fd in the PDO and map it to the axis.
9128	Current torque not mapped to EtherCAT bus axis	The current torque is not mapped to the EtherCAT bus axis.	Add the current torque in the PDO and map it to the axis.
9129	Probe control word not mapped to EtherCAT bus axis	The probe control word is not mapped to the EtherCAT bus axis.	Add the probe control word in the PDO and map it to the axis.
9130	Probe status word not mapped to EtherCAT bus axis	The probe status word is not mapped to the EtherCAT bus axis.	Add the probe status word in the PDO and map it to the axis.
9131	Probe position not mapped to EtherCAT bus axis	The probe position is not mapped to the EtherCAT bus axis.	Add the probe position in the PDO and map it to the axis.
9132	Probe channel occupied by interrupt positioning instruction	An interrupt positioning instruction is being executed and the probe channel is occupied.	The probe instruction and interrupt positioning instruction must not occupy the same probe channel at the same time. When the two instructions are called simultaneously in the program, the interrupt positioning instruction takes priority.
9133	Imaginary axis mode enabled	The imaginary axis mode is enabled.	The current instruction does not support the imaginary axis mode.
9134	Imaginary axis probe in use	The imaginary axis probe is being used.	Two imaginary axis probes are supported. Check whether the current probe is out of range.
9135	Interrupt signal not triggered in interrupt positioning	The interrupt signal is not triggered in the interrupt positioning instruction.	During execution of the interrupt positioning instruction, no interrupt signal is detected after positioning is completed.
9136	Probe channel occupied by another instruction during interrupt positioning	The probe channel is occupied by another instruction during the interrupt positioning process.	Ensure that the probe channel is not occupied during the interrupt positioning process.
9137	Control mode 0x6060 not mapped to bus driver	The control mode 0x6060 is not mapped to the bus driver.	Add 0x6060 in the PDO and map it to the axis.
9138	Control mode 0x6061 not mapped to bus driver	The control mode 0x6061 is not mapped to the bus driver.	Add 0x6061 in the PDO and map it to the axis.
9139	MC_Home called repeatedly during homing	The MC_Home instruction is called repeatedly during homing.	Do not call the MC_Home instruction repeatedly during homing.
9140	Target torque exceeded maximum value	The target torque of the instruction exceeds the maximum value.	Check whether the target torque of the instruction exceeds the positive and negative torque limits.
9141	Maximum velocity not mapped to bus driver	The maximum velocity is not mapped to the bus driver.	Add 0x607f in the PDO and map it to the axis.
9142	Immediate stop instruction active	The immediate stop instruction is active.	Check whether the immediate stop instruction has been called.

Fault Code	Message	Description	Troubleshooting
9143	Immediate stop instruction called repeatedly	The immediate stop instruction is called repeatedly.	Check whether the immediate stop instruction is called repeatedly.
9144	Limit reached during jogging	The limit is reached during jogging.	Check whether the limit is active.
9145	Target position exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target position must not exceed this value.	1. Check whether the target position is correct. Set the target position again. 2. Change the gear ratio to ensure that the target position is not greater than 9999999.
9146	Target velocity exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target velocity must not exceed this value.	1. Check whether the target velocity is correct. Set the target velocity again. 2. Change the gear ratio to ensure that the target velocity is not greater than 9999999.
9147	Target acceleration exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target acceleration must not exceed this value.	1. Check whether the target acceleration is correct. Set the target acceleration again. 2. Change the gear ratio to ensure that the target acceleration is not greater than 9999999.
9148	Target deceleration exceeded 9999999	The precision is reduced if a single-precision floating-point number exceeds 9999999. Therefore, the target deceleration must not exceed this value.	1. Check whether the target deceleration is correct. Set the target deceleration again. 2. Change the gear ratio to ensure that the target deceleration is not greater than 9999999.
9149	Axis in sync control mode, abortion not allowed	1. A single-axis motion instruction is called when the axis is performing interpolation in sync control mode. The single-axis motion instruction reports an error.	Do not call single-axis motion instructions during interpolation.
9150	MC_Halt in execution, abortion not allowed	The MC_MoveSuperImposer instruction is called while the MC_Halt instruction is still active.	Do not call the MC_MoveSuperImposer instruction while the MC_Halt instruction is still active.
9151	MC_MoveVelocityCSV PulseWidth out of range	The variable PulseWidth of the MC_MoveVelocityCSV instruction is out of range.	Ensure that the parameter value is within the allowable range.
9152	Object dictionary 60FFh not associated in I/O mapping of bus servo axis when MC_MoveVelocityCSV is called	The object dictionary 60FFh is not associated in I/O mapping of the bus servo axis when the MC_MoveVelocityCSV instruction is called.	Ensure that the object dictionary 60FFh is associated in I/O mapping of the bus servo axis when the MC_MoveVelocityCSV instruction is called.
9153	Probe terminal not configured	The probe terminal is not configured.	Check whether the software tool version supports configuration of the probe terminal ID.

Fault Code	Message	Description	Troubleshooting
9154	MC_SetAxisConfigPara ParameterIndex out of range	The value of ParameterIndex of the MC_ SetAxisConfigPara instruction is out of range.	Ensure that the parameter value is within the allowable range.
9155	Instruction execution not allowed when axis configuration parameters are being modified	The configuration parameters of the axis are being modified, and execution of this instruction is not allowed before the modification is completed.	Perform the enable operation after axis initialization is completed.
9156	Multi-execution of MC_ SetAxisConfigPara not allowed	MC_SetAxisConfigPara does not support multi-execution.	Note that this instruction does not support re-execution or multi-execution.
9157	Gear/cam motion instruction not supported by axis	The gear/cam motion instruction is not supported by the axis due to axis properties.	Ensure that the axis is not in single-axis mode or that the PLC supports the motion instruction.
9200	Failed to obtain cam table configuration file	Failed to obtain the cam table configuration file.	1. Check whether the board software and software tool match. 2. Re-download the cam configuration table.
9201	Failed to obtain master axis	Failed to obtain the master axis.	1. Check whether the master axis called in the program exists. 2. Check whether the master axis has reported an error.
9202	Failed to obtain slave axis	Failed to obtain the slave axis.	1. Check whether the slave axis called in the program exists. 2. Check whether the slave axis has reported an error.
9203	Failed to obtain cam table	Failed to obtain the cam table.	Check whether the cam table called exists.
9204	Number of cams executed simultaneously in the PLC program exceeded maximum value	The number of cams executed simultaneously in the PLC program exceeds the maximum allowable value.	Check whether the number of cams executed simultaneously in the program exceeds the threshold.
9205	No cam node found	The corresponding cam node is not found.	This instruction can be called only when the slave axis is in cam engagement state.
9206	Master axis changed during cam engagement	The master axis is changed during cam engagement.	Do not change the master axis during cam engagement.
9207	MC_CamIn StartMode out of range	StartMode of the MC_ CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9208	MC_CamIn StartPosition exceeded maximum value	StartPosition of the MC_ CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9209	MC_CamIn MasterStartDistance exceeded maximum value	MasterStartDistance of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.

Fault Code	Message	Description	Troubleshooting
9210	MC_CamIn MasterScaling exceeded maximum value	MasterScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9211	MC_CamIn SlaveScaling exceeded maximum value	SlaveScaling of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9212	MC_CamIn MasterOffset exceeded maximum value	MasterOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9213	MC_CamIn SlaveOffset exceeded maximum value	SlaveOffset of the MC_CamIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9214	MC_CamIn MasterScaling not positive	MasterScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.
9215	MC_CamIn SlaveScaling not positive	SlaveScaling of the MC_CamIn instruction is not a positive number.	Set this parameter to a positive number.
9216	MC_CamIn/MC_GearIn ReferenceType out of range	ReferenceType of the MC_CamIn/MC_GearIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9217	MC_CamIn Direction out of range	Direction of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9218	MC_CamIn BufferMode out of range	BufferMode of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9219	Master axis phases in cam table node array not in ascending order	The master axis phases in the node array of the cam table are not sorted in ascending order.	Sort the master axis phases in ascending order when customizing cam table nodes.
9220	Curve type setting of cam table node array out of range	The curve type setting of the node array of the cam table is out of range.	Check whether the curve type of the cam node array is set incorrectly.
9221	MC_CamOut target deceleration exceeded maximum value	The target deceleration of the MC_CamOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9222	MC_CamOut target deceleration out of range	The target deceleration of the MC_CamOut instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9223	MC_Phasing target acceleration exceeded maximum value	The target acceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9224	MC_Phasing target acceleration out of range	The target acceleration of the MC_Phasing instruction is out of range.	Ensure that the target acceleration is within the specified range.

Fault Code	Message	Description	Troubleshooting
9225	MC_Phasing target velocity exceeded maximum value	The target velocity of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9226	MC_Phasing target velocity out of range	The target velocity of the MC_Phasing instruction is out of range.	Ensure that the target deceleration is within the specified range.
9227	MC_CamOut curve type setting out of range	The curve type setting of the MC_CamOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9228	MC_GearOut Mode out of range	The value of Mode of the MC_CamOut instruction is out of range.	Ensure that the value of Mode is within the specified range.
9229	Cam node array empty detected by MC_ GenerateCamTable	The MC_GenerateCamTable instruction detects that the cam node array is empty.	Contact Inovance for technical support.
9230	MC_GenerateCamTable node quantity input exceeded maximum value	The node quantity specified by the MC_ GenerateCamTable instruction exceeds the maximum allowable value.	Check whether the target node quantity specified in the instruction is beyond the specified range.
9231	MC_GenerateCamTable Mode out of range	The value of Mode of the MC_GenerateCamTable instruction is out of range.	Ensure that the parameter value is within the specified range.
9232	MC_GenerateCamTable node quantity input too small	The node quantity specified by the MC_ GenerateCamTable instruction is too small.	Ensure that the node quantity is 2 or more.
9233	RatioNumerator in gear instruction set to 0	The RatioNumerator parameter in the gear instruction is set to 0.	Set this parameter to a non-zero integer.
9234	RatioDenominator in gear instruction not greater than 0	The RatioDenominator parameter in the gear instruction is not greater than 0.	Set this parameter to an integer greater than 0.
9235	MC_GenerateCamTable in execution when MC_ SaveCamTable is called	The MC_GenerateCamTable instruction is being executed when the MC_ SaveCamTable instruction is called.	Do not call the MC_SaveCamTable instruction before the cam table data update operation is completed.
9236	MC_SaveCamTable in execution when MC_ GenerateCamTable is called	The MC_SaveCamTable instruction is being executed on the cam table when the MC_ GenerateCamTable instruction is called.	Do not call the MC_GenerateCamTable instruction before the cam table is saved.
9237	Failed to open cam table during execution of MC_ SaveCamTable	Failed to open the cam table file during execution of the MC_SaveCamTable instruction.	1. Check whether the PLC memory runs out. 2. Replace the PLC.
9238	Failed to write cam point quantity when saving the cam table	Failed to write the cam point quantity when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.

Fault Code	Message	Description	Troubleshooting
9239	Failed to write data when saving cam table	Failed to write data when the cam table is being saved.	1. Check whether the PLC memory runs out. 2. Replace the PLC.
9240	Phase of the first point not 0	The phase of the first point is not 0.	Ensure that the phase of the first point is 0.
9241	Displacement of the first point not 0	The displacement of the first point is not 0.	Ensure that the displacement of the first point is 0.
9242	MC_GearOut Mode out of range	The value of Mode of the MC_GearOut instruction is out of range.	Ensure that the value of Mode is within the specified range.
9243	MC_Phasing target deceleration exceeded maximum value	The target deceleration of the MC_Phasing instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9244	MC_GearIn target deceleration exceeded maximum value	The target deceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9245	MC_CamIn Periodic out of range	The value of Periodic of the MC_CamIn instruction is out of range.	Ensure that the parameter value is within the specified range.
9246	Cam table phase exceeded maximum value	The phase in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.
9247	Absolute value of cam table displacement exceeded maximum value	The absolute value of the displacement in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.
9248	Absolute value of cam table link velocity exceeded maximum value	The absolute value of the link velocity in the cam table exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number does not exceed 9999999.
9249	Gear node empty	The gear node is empty.	Contact Inovance for technical support.
9250	Master axis same as slave axis	The master axis and slave axis are the same.	Do not use the same axis as both the master axis and slave axis of the cam gear.
9251	Master axis configuration address greater than or equal to slave axis address	The configuration address of the master axis is greater than or equal to that of the slave axis.	When ReferenceType is set to set position of the current cycle, ensure that the configuration address of the master axis is less than that of the slave axis.
9252	Master axis filter coefficient fFilter[0] corresponding to the slave axis out of range	The master axis filter coefficient fFilter[0] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).
9253	Master axis filter coefficient fFilter[1] corresponding to the slave axis out of range	The master axis filter coefficient fFilter[1] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).
9254	Master axis filter coefficient fFilter[2] corresponding to the slave axis out of range	The master axis filter coefficient fFilter[2] corresponding to the slave axis is out of range.	Ensure that the value of this variable is between 0 and 1 (0 and 1 included).

Fault Code	Message	Description	Troubleshooting
9255	Sum of master axis filter coefficients corresponding to the slave axis not 1	The sum of the master axis filter coefficients corresponding to the slave axis is not 1.	Ensure that the sum of the master axis filter coefficients corresponding to the slave axis is 1.
9256	Improper StartPosition and MasterStartDistance in MC_CamIn	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to positive, ensure that the cam synchronization point is not less than the cam engagement point.
9257	Improper StartPosition and MasterStartDistance in MC_CamIn	The start position and start distance of the master axis in the MC_CamIn instruction are improper.	If the master axis works in linear mode and Direction in the instruction is set to negative, ensure that the cam synchronization point is not greater than the cam engagement point.
9258	MC_GearOut target deceleration exceeded maximum value	The target deceleration of the MC_GearOut instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9259	MC_Phasing target deceleration out of range	The target deceleration of the MC_Phasing instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9260	MC_GearIn target deceleration out of range	The target deceleration of the MC_GearIn instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9261	MC_GearOut target deceleration out of range	The target deceleration of the MC_GearOut instruction is out of range and causes stop upon a fault.	Ensure that the target deceleration is within the specified range.
9262	MC_GearIn target acceleration exceeded maximum value	The target acceleration of the MC_GearIn instruction exceeds the maximum allowable value.	Ensure that the absolute value of the floating-point number in the motion control instruction does not exceed 9999999.
9263	MC_GearIn target acceleration out of range	The target acceleration of the MC_GearIn instruction is out of range.	Ensure that the target acceleration is within the specified range.
9264	MC_Phasing curve type setting out of range	The curve type setting of the MC_Phasing instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9265	MC_GearIn curve type setting out of range	The curve type setting of the MC_GearIn instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9266	MC_GearOut curve type setting out of range	The curve type setting of the MC_GearOut instruction is out of range.	Ensure that the curve type setpoint in the instruction is within the specified range.
9267	Slave axis changed during cam operation	The slave axis is modified during the cam operation.	Do not modify the slave axis during the cam operation.
9268	MC_Phasing PhasingMode out of range	The value of PhasingMode of the MC_Phasing instruction is out of range.	Ensure that the value of the parameter is within the specified range.
9269	Axis not in cam control mode when MC_CamOut is called	The current axis is not in cam control mode when the MC_CamOut instruction is called.	Ensure that the axis works in cam control mode when the MC_CamOut instruction is called.

Fault Code	Message	Description	Troubleshooting
9270	Axis not in gear control mode when MC_GearOut is called	The current axis is not in gear control mode when the MC_GearOut instruction is called.	Ensure that the axis works in gear control mode when the MC_GearOut instruction is called.
9271	Master axis position change too large within a single EtherCAT cycle during cam/gear operation	The position change of the master axis is too large within a single EtherCAT cycle during cam/gear operation.	Ensure that the position change of the master axis is not greater than half a cam cycle within a single EtherCAT cycle.
9272	MC_ GetCamTableDistance Phase out of range	The point specified by Phase in the MC_ GetCamTableDistance instruction does not fall between the start and end points.	Ensure that the point specified by Phase is within the specified curve.
9273	Slave axis changed during execution of MC_ GearIn	The slave axis is changed during execution of the MC_GearIn instruction.	Do not change the slave axis during execution of the MC_GearIn instruction.
9274	MC_DigitalCamSwitch Channel out of range	The value of Channel of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.
9275	No axis found	The axis is not found.	Ensure that the axis specified by Axis exists.
9276	Number of tappets allowed to be executed at the same time out of range	The number of tappets allowed to be executed at the same time is out of range.	Ensure that the number of tappets allowed to be executed at the same time is within the allowable range.
9277	MC_DigitalCamSwitch ReferenceType out of range	The value of ReferenceType of the MC_ DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.
9278	MC_DigitalCamSwitch Number out of range	The value of Number of the MC_DigitalCamSwitch instruction is out of range.	Ensure that the parameter value is within the specified range.
9279	MC_DigitalCamSwitch Switches array empty	The Switches array of the MC_DigitalCamSwitch instruction is empty.	Check whether the length of the Switches array meets requirements.
9280	Tappet array fPosition out of range	The value of fPosition of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9281	Tappet array iMode out of range	The value of iMode of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9282	Tappet array iDirection out of range	The value of iDirection of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9283	Tappet array fParameter out of range	The value of fParameter of the tappet array is out of range.	Ensure that the parameter value is within the specified range.
9284	Time setting out of range in time mode	When the tappet comparison point is set to time mode, the time setting is out of range.	Ensure that the parameter value is within the specified range.

Fault Code	Message	Description	Troubleshooting
9285	Selected axis not under cam control when MC_ DigitalCamSwitch ReferenceType is set to 3	The selected axis is not under cam control when ReferenceType of the MC_ DigitalCamSwitch instruction is set to 3.	Call the MC_DigitalCamSwitch instruction after cam control takes effect.
9286	Axis communication interrupted during tappet execution	Axis communication is interrupted during tappet execution.	Ensure that axis communication is not interrupted during tappet execution.
9287	Duplicate comparison position start points	The comparison position start points are the same during tappet execution.	Ensure that the start points are not duplicate.
9288	Comparison position start point same as end point	The comparison position start and end point are the same during tappet execution.	Ensure that the start and end points are not duplicate.
9289	Selected tappet terminal in use	The selected tappet terminal is being used by another function.	Check whether the terminal is set as the pulse output axis.
9290	Failed to execute MC_ DigitalCamSwitch due to improper motion control axis state	The MC_DigitalCamSwitch instruction cannot be executed because the state of the motion control axis is improper.	Do not execute the MC_DigitalCamSwitch instruction in homing mode.
9291	MasterSyncPosition setting in MC_GearInPos out of range	The MasterSyncPosition setting in the MC_ GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9292	SlaveSyncPosition setting in MC_GearInPos out of range	The SlaveSyncPosition setting in the MC_ GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9293	MasterStarDistance in MC_GearInPos out of range	The MasterStarDistance setting in the MC_ GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9294	Velocity setting in MC_ GearInPos over system limit	The Velocity setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.
9295	Velocity setting in MC_ GearInPos over setting limit	The Velocity setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.
9296	Acceleration setting in MC_GearInPos over system limit	The Acceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.
9297	Acceleration setting in MC_GearInPos over setting limit	The Acceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.
9298	Deceleration setting in MC_GearInPos over system limit	The Deceleration setting in the MC_GearInPos instruction exceeds the system limit.	Ensure that the parameter value is within the specified range.

Fault Code	Message	Description	Troubleshooting
9299	Deceleration setting in MC_GearInPos over setting limit	The Deceleration setting in the MC_GearInPos instruction exceeds the setting limit.	Ensure that the parameter value is within the specified range.
9300	AvoidReversal in MC_ GearInPos out of range	The AvoidReversal setting in the MC_GearInPos instruction is out of range.	Ensure that the parameter value is within the specified range.
9301	Zero master axis speed when MC_GearInPos instruction is started	The master axis speed is zero when the MC_ GearInPos instruction is started.	Ensure that the master axis speed is not zero when starting this instruction.
9302	Zero master axis displacement in catching phase of MC_ GearInPos	The master axis did not move during the catching phase of the MC_GearInPos instruction.	When MasterStarDistance is set to 0, ensure that the input MasterSyncPosition does not overlap with the current position of the master axis.
9303	Slave axis speed not zero before entering chasing phase after MC_ GearInPos is started	When the MC_GearInPos instruction is started, the speed of the slave axis is not zero before entering the catching phase.	Ensure that the slave axis remains stationary before entering the catching phase.
9304	Failed to enter catching phase of MC_GearInPos	Failed to enter the catching phase when the MC_ GearInPos instruction is executed.	Ensure that the master axis can enter the catching phase under the current position and motion direction conditions.
9305	Slave axis over-speed during MC_GearInPos operation	The velocity of the slave axis exceeds the limit during execution of the MC_GearInPos instruction.	Ensure that the parameter value is within the specified range.
9400	Maximum axis group quantity exceeded	The number of axis groups exceeds the maximum allowable value.	Reduce the number of axis groups in the project so that it does not exceed the maximum value.
9401	Faulty axis in axis group	An axis in the axis group is faulty.	Locate the faulty axis, view the fault codes of the axis, and rectify the fault.
9402	Number of buffered interpolation instructions exceeded 8	The number of buffered interpolation instructions is greater than 8.	Check whether the number of buffered interpolation instructions is greater than 8.
9403	Axis reused	An axis in the axis group is reused.	Each axis can be used in only one axis group. Check whether there is a reused axis in the axis group and replace it with an unused axis.
9404	Failed to create axis group	The x-axis or y-axis does not exist.	Check whether the x-axis and y-axis exist. An axis group consists of at least the x-axis and y-axis.
9405	Specified z-axis non-existent	The z-axis is specified in the instruction but does not exist in the configuration.	Check whether the z-axis specified in the instruction exists.
9406	Specified auxiliary axis non-existent	The auxiliary axis is specified in the instruction but does not exist in the configuration.	Check whether the auxiliary axis specified in the instruction exists.
9407	Axis group ID duplicated	The specified axis group ID has been used.	Change the axis group ID because the axis group ID must be unique.
9408	Axis configuration failed	Failed to configure the axis.	Check whether any axis in the axis group fails to be configured. If yes, check whether the board software and the background match.

Fault Code	Message	Description	Troubleshooting
9409	Axis ID less than 0	The axis ID is less than 0.	Check whether the ID of an axis in the axis group is less than 0.
9410	Axis group not released	The axis group is not released because the same MC_SetAxesGroup instruction is triggered repeatedly in a short time period.	Do not re-trigger the MC_SetAxesGroup instruction while its Busy signal output is still active.
9411	MC_GroupStop aborted	The MC_GroupStop instruction is aborted.	Check whether an instruction with higher priority is called while the MC_GroupStop instruction is still active.
9412	Circular interpolation instruction CircAxes out of range	The value of CircAxes of the circular interpolation instruction is out of range.	Check whether the value of CircAxes of the circular interpolation instruction is out of range.
9413	Circular interpolation instruction CircMode out of range	The value of CircMode of the circular interpolation instruction is out of range.	Check whether the value of CircMode of the circular interpolation instruction is out of range.
9414	Circular interpolation instruction PathChoice out of range	The value of PathChoice of the circular interpolation instruction is out of range.	Check whether the value of PathChoice of the circular interpolation instruction is out of range.
9415	Stop instruction StopMode out of range	The value of StopMode of the stop instruction is out of range.	Check whether the value of StopMode of the stop instruction is out of range.
9416	X-axis set to ring mode	The x-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9417	Y-axis set to ring mode	The y-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9418	Z-axis set to ring mode	The z-axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9419	Auxiliary axis set to ring mode	The auxiliary axis is set to ring mode.	Do not set the motion control axis to the ring mode in an interpolation instruction.
9420	Circular interpolation instruction triggered repeatedly	The circular interpolation instruction is triggered repeatedly.	Do not re-trigger the same circular interpolation instruction while its Busy signal output is still active.
9421	Linear interpolation instruction triggered repeatedly	The linear interpolation instruction is triggered repeatedly.	Do not re-trigger the same linear interpolation instruction while its Busy signal output is still active.
9422	Failed to obtain the axis group	Failed to obtain the axis group.	Check whether the axis group specified by GroupID has been created by calling MC_SetAxesGroup.
9423	Axis configuration failed	Failed to configure the axis.	Check whether an instruction is triggered when axis configuration is not completed. Check whether the communication state of all axes in the axis group is Axis ready.
9424	Axis disabled	An axis is disabled.	Do not call the interpolation instruction when any axis is in Disabled state.
9425	Axis in execution of single-axis motion instruction	The interpolation instruction is triggered when an axis is executing a single-axis motion instruction.	Do not call the interpolation instruction when any axis is executing single-axis motion instructions and not in StandStill state.

Fault Code	Message	Description	Troubleshooting
9426	Axis in Stopping state	An axis is in Stopping state.	Do not call the interpolation instruction when any axis is in Stopping state after executing the MC_Stop instruction.
9427	Axis group in Stopping state	The axis group is in Stopping state.	Do not call the interpolation instruction while the MC_GroupStop instruction is still active.
9428	Axis in Homing state	An axis is in Homing state.	Do not call the interpolation instruction when any axis is in Homing state after executing the MC_Home instruction.
9429	Axis in execution of the position setting instruction	An axis is executing the position setting instruction.	Do not call the interpolation instruction when any axis is setting the current position by executing the MC_SetPosition instruction.
9430	Axis in commissioning state	An axis is in commissioning state.	Do not call the interpolation instruction when any axis is in commissioning state.
9431	Axis in commissioning state during interpolation, aborted instruction execution of other axes	An axis enters the commissioning state during interpolation, which aborts instruction execution of other axes.	Check whether any axis enters the commissioning state during interpolation and aborts instruction execution of other axes.
9432	Failed to request memory	Failed to request the memory.	Check whether the memory runs out. Contact the manufacturer.
9433	Target velocity less than or equal to 0	The target velocity is 0 or less than 0.	Ensure that the target velocity of the instruction is greater than 0.
9434	Target acceleration less than or equal to 0	The target acceleration is 0 or less than 0.	Ensure that the target acceleration of the instruction is greater than 0.
9435	Target deceleration less than or equal to 0	The target deceleration is 0 or less than 0.	Ensure that the target deceleration of the instruction is greater than 0.
9436	Curve type setting out of range	The curve type setting is out of range.	Check whether the curve type is set to a value other than the T-shaped curve for the interpolation instruction.
9437	Improper AbsRelMode	AbsRelMode is set incorrectly.	Check whether the parameter is set to a value other than the absolute positioning and relative positioning modes.
9438	Improper BufferMode	BufferMode is set incorrectly.	Check whether the value of BufferMode is proper.
9439	Improper InsertMode	InsertMode is set incorrectly.	Check whether the value of InsertMode is proper.
9440	Axis stopped due to a fault	An axis stops due to a fault.	Locate the faulty axis and rectify the fault based on the fault code.
9441	MC_GroupStop called repeatedly	The MC_GroupStop instruction is called repeatedly.	Do not re-trigger an MC_GroupStop instruction or call other MC_GroupStop instructions while an MC_GroupStop instruction is still active.
9442	Data buffer area not empty	The data buffer area is not empty. It is an internal fault.	Contact the manufacturer.
9443	Not a circle	No circle can be drawn due to improper parameter settings.	Update the parameter settings.
9444	Not a circle	The start, end, and border points in the circular interpolation instruction are the same point, and no circle can be drawn.	Check the input parameters of the circular interpolation instruction and ensure that the start, end, and border points can form a circle.

Fault Code	Message	Description	Troubleshooting
9445	Instruction buffer area full	The instruction buffer area is full.	Contact Inovance for technical support.
9446	X-axis exceeded maximum velocity	The velocity of the x-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the x-axis is not greater than the maximum allowable velocity.
9447	Y-axis exceeded maximum velocity	The velocity of the y-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the y-axis is not greater than the maximum allowable velocity.
9448	Z-axis exceeded maximum velocity	The velocity of the z-axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the z-axis is not greater than the maximum allowable velocity.
9449	Auxiliary axis exceeded maximum velocity	The velocity of the auxiliary axis exceeds the maximum allowable velocity.	Ensure that the target velocity of the auxiliary axis is not greater than the maximum allowable velocity.
9450	Failed to obtain the number of axis groups	Failed to obtain the number of axis groups.	Update the software tool to the latest version.
9451	Internal fault	An internal fault occurs.	Contact the manufacturer.
9452	Instruction called when the axis is in StandStill state	The instruction is called when the axis is in StandStill state.	Do not call this instruction when the axis is StandStill state.
9453	Maximum velocity exceeded	The maximum velocity specified on the axis group configuration interface is exceeded.	Check whether the target velocity of the instruction is greater than the maximum velocity specified on the axis group configuration interface.
9454	Maximum acceleration (deceleration) exceeded	The maximum allowable acceleration (deceleration) is exceeded.	Check whether the target acceleration (deceleration) of the instruction is greater than the maximum acceleration (deceleration) specified on the axis group configuration interface.
9455	Axis group fault due to linear interpolation instruction error	The axis group becomes faulty due to an error reported by the linear interpolation instruction.	Identify the first linear interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9456	Axis group fault due to circular interpolation instruction error	The axis group becomes faulty due to an error reported by the circular interpolation instruction.	Identify the first circular interpolation instruction that reports the error and troubleshoot the fault based on the fault code.
9457	Axis group fault due to axis group stop instruction error	The axis group becomes faulty due to an error reported by the axis group stop instruction.	Identify the first axis group stop instruction that reports the error and troubleshoot the fault based on the fault code.
9458	Axis group fault due to axis group pause instruction error	The axis group becomes faulty due to an error reported by the axis group pause instruction.	Identify the first axis group pause instruction that reports the error and troubleshoot the fault based on the fault code.
9459	X-axis performing the interpolation algorithm of another axis group	The x-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9460	Y-axis performing the interpolation algorithm of another axis group	The y-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.

Fault Code	Message	Description	Troubleshooting
9461	Z-axis performing the interpolation algorithm of another axis group	The z-axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9462	Auxiliary axis performing the interpolation algorithm of another axis group	The auxiliary axis in the axis group is performing the interpolation algorithm of another axis group.	An axis can be configured in different axis groups at the same time. However, ensure that it executes the interpolation instruction of only one axis group at the same time.
9463	Axes in synchronous mode but not under axis group control when the MC_GroupStop instruction is called	When the MC_GroupStop instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupStop instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupStop instruction when the axes enter the synchronous mode due to other instructions.
9464	Axes in synchronous mode but not under axis group control when the linear or circular interpolation instruction is called	When the linear or circular interpolation instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the linear or circular interpolation instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the linear or circular interpolation instruction when the axes enter the synchronous mode due to other non-axis-group instructions.
9465	Axes in synchronous mode but not under axis group control when the MC_GroupHalt instruction is called	When the MC_GroupHalt instruction is called, the axes are in synchronous mode but not under axis group control, such as interpolation control or cam control.	Note that the MC_GroupHalt instruction can be called only when the axes in the axis group are in synchronous mode under axis group control. Do not call the MC_GroupHalt instruction when the axes enter the synchronous mode due to other instructions.
9466	Improper NumOfTurns in MC_MoveEllipse	NumOfTurns in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9467	Improper AddLength in MC_MoveEllipse	AddLength in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9468	Shutdown due to MC_MoveEllipse failure	The MC_MoveEllipse instruction fails and causes shutdown.	Find the MC_MoveEllipse instruction that caused the failure and check the fault code of the instruction to further confirm the fault.
9469	Improper CircAxes in MC_MoveEllipse	CircAxes in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9470	Improper CircMode in MC_MoveEllipse	CircMode in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9471	Improper PathChoice in MC_MoveEllipse	PathChoice in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9472	Improper Velocity in MC_MoveEllipse	Velocity in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9473	Improper Acceleration in MC_MoveEllipse	Acceleration in the MC_MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.

Fault Code	Message	Description	Troubleshooting
9474	Improper Deceleration in MC_MoveEllipse	Deceleration in the MC_ MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9475	Improper BufferMode in MC_MoveEllipse	BufferMode in the MC_ MoveEllipse instruction is set incorrectly.	Ensure that the parameter value is within the allowable range.
9476	Cannot form ellipse due to unreasonable center point, long axis length, and short axis length	The set center point, long axis length, and short axis length are improper and cannot form an ellipse.	Ensure that the parameter value is within the allowable range.
9477	Interpolation not supported by x-axis	The property of the x-axis in the axis group instruction does not support the interpolation motion.	Ensure that the x-axis is not in single-axis mode.
9478	Interpolation not supported by y-axis	The property of the y-axis in the axis group instruction does not support the interpolation motion.	Ensure that the y-axis is not in single-axis mode.
9479	Interpolation not supported by z-axis	The property of the z-axis in the axis group instruction does not support the interpolation motion.	Ensure that the z-axis is not in single-axis mode.
9480	Interpolation not supported by auxiliary axis	The property of the auxiliary axis in the axis group instruction does not support the interpolation motion.	Ensure that the auxiliary axis is not in single-axis mode.
9501	EtherCAT bus drive error	A drive error occurs. The fault code in the object dictionary 0x603F of the drive is 0x%x{16:16}.	1. Determine the drive fault type according to the bus drive guide and rectify the fault.
9502	Drive disabled	The drive is disabled.	1. Check whether the drive status word 0x6041 switches to the disabled state during motion. 2. Check whether communication is disconnected during motion.
9503	Limit reached	The limit is reached.	1. Check whether the software limit is configured and reached. 2. Check whether the hardware limit is reached.
9505	Failed to modify the control mode	Failed to modify the control mode.	1. Check for interference in network communication. 2. Check whether the drive supports the object dictionary 0x6060.
9508	Homing failed	Homing failed.	1. Identify the cause of the drive homing failure based on the fault code. 2. Check whether homing timed out.
9509	Axis internal calculation precision error	An axis internal calculation precision error occurs.	Check whether the floating-point data of the instruction falls beyond the single-precision floating-point number range.
9510	Following error out of range	The following error is out of range.	1. Check whether the acceleration is too large. 2. Check whether the set following error is too small.

Fault Code	Message	Description	Troubleshooting
9512	Servo drive disconnected during operation	The servo drive is disconnected during operation.	1. Check whether the drive works properly. 2. Check whether the network cable is properly connected. 3. Check for strong interference in communication.
9513	Homing failed due to a drive fault	Homing failed due to a drive fault.	Check the fault code of the drive to eliminate the fault.
9514	Homing failed because the homing offset exceeded 32 bits	Homing failed because the homing offset exceeded 32 bits.	Check whether the homing offset multiplied by the gear ratio exceeds 32 bits; if yes, change the gear ratio.
9515	Homing failed due to loss of the slave	Homing failed because the EtherCAT drive is lost.	Contact Inovance for technical support.
9516	Homing failed because the SDO failed to write to object dictionary 0x607C	Homing failed because the SDO failed to write to object dictionary 0x607C.	1. Check whether the drive supports 0x607C. 2. Check the network communication quality.
9517	Homing failed because the SDO failed to write 6 to object dictionary 0x6060	Homing failed because the SDO failed to write 6 to object dictionary 0x6060.	1. Set 0x6060 in the PDO. 2. Check the network communication quality.
9518	Homing failed because the SDO failed to read object dictionary 0x6061	Homing failed because the SDO failed to read object dictionary 0x6061.	1. Set 0x6061 in the PDO. 2. Check the network communication quality.
9519	Homing failed because the SDO failed to write 8 to object dictionary 0x6060	Homing failed because the SDO failed to write 8 into object dictionary 0x6060.	1. Set 0x6060 in the PDO. 2. Check the network communication quality.
9551	Failed to switch the control mode	Failed to switch the control mode.	Check for interference in network communication.
9552	Target velocity equal to 0	The target velocity is 0.	Check whether the target velocity of position instructions is appropriate.
9601	Axis stopped due to MC_MoveAbsolute parameter exception	The axis stops due to parameter exception of the MC_MoveAbsolute instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9602	Axis stopped due to MC_MoveRelative parameter exception	The axis stops due to parameter exception of the MC_MoveRelative instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9603	Axis stopped due to MC_MoveVelocity exception	The axis stops due to exception of the MC_MoveVelocity instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9604	Axis stopped due to MC_Jog exception	The axis stops due to exception of the MC_Jog instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9605	Axis stopped due to MC_MoveVelocityCSV exception	The axis stops due to exception of the MC_MoveVelocityCSV instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9606	Axis stopped due to MC_MoveBuffer exception	The axis stops due to exception of the MC_MoveBuffer instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.

Fault Code	Message	Description	Troubleshooting
9607	Axis stopped due to MC_ MoveFeed parameter exception	The axis stops due to parameter exception of the MC_MoveFeed instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9608	Axis stopped due to MC_ Stop parameter exception	The axis stops due to parameter exception of the MC_Stop instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9609	Axis stopped due to MC_ MoveTorque parameter exception	The axis stops due to parameter exception of the MC_MoveTorque instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9610	Axis stopped due to MC_ Halt parameter exception	The axis stops due to parameter exception of the MC_Halt instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9611	Axis stopped due to MC_ MoveSuperImposed parameter exception	The axis stops due to parameter exception of the MC_MoveSuperImposed instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9612	Axis stopped due to MC_ SyncMoveVelocity error	The axis stops due to an error reported by the MC_ SyncMoveVelocity instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9613	Axis stopped due to MC_ SyncTorqueControl error	The axis stops due to an error reported by the MC_ SyncTorqueControl instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9614	Axis stopped due to MC_ FollowVelocity error	The axis stops due to an error reported by the MC_ FollowVelocity instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9615	Axis stopped due to MC_ SetOverRide parameter exception	The axis stops due to parameter exception of the MC_SetOverRide instruction.	Check the instruction that reports the error, and further determine the fault based on the fault code of the instruction.
9701	Failed to request memory for the encoder axis instruction	The encoder axis instruction failed to request the memory.	1. Check whether the PLC memory runs out. 2. Contact the manufacturer.
9702	1. Encoder axis type error 2. Requested encoder axis non-existent 3. Instruction not supported in offline commissioning	1. The encoder axis type is incorrect. 2. The requested encoder axis does not exist. 3. The instruction is not supported in offline commissioning.	This instruction does not support the set axis type. Check whether the axis type setting is incorrect.
9703	Axis configuration failed	Failed to configure the axis.	Check whether the board software and the software tool match.
9704	Counter operation command not configured in I/O mapping of encoder axis	Counter operation command is not configured in I/O mapping of the encoder axis.	Configure Counter operation command in I/O mapping of the encoder axis.
9705	Counter status not configured in I/O mapping of encoder axis	Counter status is not configured in I/O mapping of the encoder axis.	Configure Counter status in I/O mapping of the encoder axis.

Fault Code	Message	Description	Troubleshooting
9706	Encoder present position not configured in I/O mapping of encoder axis	Encoder present position is not configured in I/O mapping of the encoder axis.	Configure Encoder present position in I/O mapping of the encoder axis.
9707	Pulse rate not configured in I/O mapping of encoder axis	Pulse rate is not configured in I/O mapping of the encoder axis.	Configure Pulse rate in I/O mapping of the encoder axis.
9708	Positive limit not greater than negative limit	The positive limit of the encoder axis is not greater than the negative limit.	Ensure that the positive limit of the encoder axis is greater than the negative limit.
9709	Positive limit greater than 2147483647 after being converted into the pulse unit	The positive limit of the encoder axis is greater than 2147483647 after being converted into the pulse unit.	Ensure that the positive limit of the encoder axis is less than or equal to 2147483647 after being converted into the pulse unit.
9710	Negative limit less than –2147483648 after being converted into the pulse unit	The negative limit of the encoder axis is less than –2147483648 after being converted into the pulse unit.	Ensure that the negative limit of the encoder axis is greater than or equal to –2147483648 after being converted into the pulse unit.
9711	Revolution cycle in ring mode greater than 2147483647 after being converted into the pulse unit	The revolution cycle of the encoder axis in ring mode is greater than 2147483647 after being converted into the pulse unit.	Ensure that the revolution cycle of the encoder axis in ring mode is less than or equal to 2147483647 after being converted into the pulse unit.
9712	Encoder axis changed while ENC_Counter is active	The encoder axis is changed while the ENC_Counter instruction is still active.	Do not change the encoder axis while the ENC_Counter instruction is still active.
9713	GR10-2HCE module faulty	The GR10-2HCE module is faulty.	Check the fault code object dictionary of the GR10-2HCE module and troubleshoot the fault according to the fault code.
9714	Failed to reset the encoder axis fault	Failed to reset the encoder axis fault.	1. The current fault of the encoder axis does not support reset. 2. The encoder shaft enters the faulty state immediately after the fault is reset. Check the axis fault codes and slave fault codes to further determine the fault type.
9715	ENC_Reset called when the encoder axis is not faulty	The ENC_Reset instruction is called when the encoder axis is not faulty.	Do not call the ENC_Reset instruction when the encoder axis is not faulty.
9716	ENC_Preset TriggerMode out of range	The value of TriggerMode of the ENC_Preset instruction is out of range.	Ensure that the parameter value is within the allowable range.
9717	ENC_Preset Position greater than 9999999	The value of Position of the ENC_Preset instruction is greater than 9999999.	Set Position of the ENC_Preset instruction to a value less than or equal to 9999999.
9718	Physical output command not configured in I/O mapping of encoder axis	Physical output command is not configured in I/O mapping of the encoder axis.	Configure Physical output command in I/O mapping of the encoder axis.

Fault Code	Message	Description	Troubleshooting
9719	Preset position or comparison output position greater than positive limit	The preset position or comparison output position of the encoder axis instruction is greater than the positive limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than or equal to the positive limit.
9720	Preset position or comparison output position less than negative limit	The preset position or comparison output position of the encoder axis instruction is less than the negative limit.	Ensure that the preset position or comparison output position of the encoder axis instruction is greater than or equal to the negative limit.
9721	Preset position or comparison output position greater than 2147483647 or less than –2147483648 after being converted into the pulse unit	The preset position or comparison output position of the encoder axis instruction is greater than 2147483647 or less than –2147483648 after being converted into the pulse unit.	Ensure that the preset position or comparison output position of the encoder axis instruction is between –2147483648 and +2147483647 after being converted into the pulse unit.
9722	Preset position or comparison output position greater than or equal to revolution cycle in ring mode	The preset position or comparison output position of the encoder axis instruction is greater than or equal to the revolution cycle in ring mode.	Ensure that the preset position or comparison output position of the encoder axis instruction is less than the revolution cycle in ring mode.
9723	ENC_TouchProbe ProbeID out of range	The value of ProbeID of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9724	ENC_TouchProbe TriggerEdge out of range	The value of TriggerEdge of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9725	ENC_TouchProbe TerminalSource out of range	The value of TerminalSource of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9726	ENC_TouchProbe TriggerMode out of range	The value of TriggerMode of the ENC_TouchProbe instruction is out of range.	Ensure that the parameter value is within the allowable range.
9727	Probe status word not associated in I/O mapping of the encoder axis	The probe status word is not associated in I/O mapping of the encoder axis.	Ensure that the probe status word is associated in I/O mapping of the encoder axis.
9728	Probe feedback position not associated in I/O mapping of the encoder axis	The probe feedback position is not associated in I/O mapping of the encoder axis.	Ensure that the probe feedback position is associated in I/O mapping of the encoder axis.
9729	Control word not associated in I/O mapping of the encoder axis	The control word is not associated in I/O mapping of the encoder axis.	Ensure that the control word is associated in I/O mapping of the encoder axis.

Fault Code	Message	Description	Troubleshooting
9730	Window start position not less than end position	The probe window function of the encoder axis is enabled, but the start position of the window is not less than the end position.	Ensure that the start position of the probe window is less than the end position.
9731	Xn0 not assigned with probe function	The Xn0 terminal is not assigned with the probe function.	Assign the probe function to the Xn0 terminal.
9732	Xn1 not assigned with probe function	The Xn1 terminal is not assigned with the probe function.	Assign the probe function to the Xn1 terminal.
9742	Compare mode not configured in I/O mapping of encoder axis	Compare mode is not configured in I/O mapping of the encoder axis.	Configure Compare mode in I/O mapping of the encoder axis.
9743	Compare pulse/time not configured in I/O mapping of encoder axis	Compare pulse/time is not configured in I/O mapping of the encoder axis.	Configure Compare pulse/time in I/O mapping of the encoder axis.
9744	Compare size/step not configured in I/O mapping of encoder axis	Compare size/step is not configured in I/O mapping of the encoder axis.	Configure Compare size/step in I/O mapping of the encoder axis.
9745	Compare point value 1 not configured in I/O mapping of encoder axis	Compare point value 1 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 1 in I/O mapping of the encoder axis.
9746	Compare point value 2 not configured in I/O mapping of encoder axis	Compare point value 2 is not configured in I/O mapping of the encoder axis.	Configure Compare point value 2 in I/O mapping of the encoder axis.
9747	Physical output status not configured in I/O mapping of encoder axis	Physical output status is not configured in I/O mapping of the encoder axis.	Configure Physical output status in I/O mapping of the encoder axis.
9748	Compare error code not configured in I/O mapping of encoder axis	Compare error code is not configured in I/O mapping of the encoder axis.	Configure Compare error code in I/O mapping of the encoder axis.
9749	Current compare number/position not configured in I/O mapping of encoder axis	Current compare number/ position is not configured in I/O mapping of the encoder axis.	Configure Current compare number/position in I/O mapping of the encoder axis.
9750	Failed to obtain the array start address of the single-axis array comparison output instruction	Failed to obtain the start address of the array of the single-axis array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the background and board software match. 3. Check whether the array of the instruction is out of bounds.
9751	Failed to obtain the axis group start address of the axis group array comparison output instruction	Failed to obtain the start address of the axis group of the axis group array comparison output instruction.	1. Check whether the PLC memory is sufficient. 2. Check whether the background and board software match. 3. Check whether the array of the instruction is out of bounds.
9752	Bus encoder axis not associated with slave	The bus encoder axis is not associated with any slave.	Associate the bus encoder axis with a slave.

Fault Code	Message	Description	Troubleshooting
9753	X-axis and y-axis of the axis group array comparison instruction not associated with the same slave	The x-axis and y-axis of the axis group array comparison instruction are not associated with the same slave.	Associate the x-axis and y-axis of the axis group comparison output instruction with the same slave.
9754	X-axis of the axis group array comparison instruction not associated with the first channel of the slave	The x-axis of the axis group array comparison instruction is not associated with the first channel of the slave.	Associate the x-axis of the axis group comparison output instruction with the first channel of the slave.
9755	Y-axis of the axis group array comparison instruction not associated with the second channel of the slave	The y-axis of the axis group array comparison instruction is not associated with the second channel of the slave.	Associate the y-axis of the axis group comparison output instruction with the second channel of the slave.
9756	Yn0 not assigned with the one-dimensional comparison output function	The Yn0 terminal is not assigned with the one-dimensional comparison output function.	Assign the one-dimensional comparison output function to the Yn0 output terminal corresponding to the channel.
9757	Absolute value of start value of encoder axis step comparison output instruction greater than 9999999	The absolute value of the start value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9758	Absolute value of end value of encoder axis step comparison output instruction greater than 9999999	The absolute value of the end value of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9759	Absolute value of the step of the encoder axis step comparison output instruction greater than 9999999	The absolute value of the step of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9760	Absolute value of Parameter of the encoder axis step comparison output instruction greater than 9999999	The absolute value of Parameter of the encoder axis step comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9761	Mode of the encoder axis comparison output instruction out of range	The value of Mode of the encoder axis comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9762	Time for time control of the encoder axis comparison output out of range	The time for time control of the encoder axis comparison output is out of range.	Ensure that the parameter value is within the allowable range.
9763	Step of the encoder axis step comparison output instruction equal to 0	The step of the encoder axis step comparison output instruction is 0.	Set the step of the step comparison output instruction to a value other than 0.

Fault Code	Message	Description	Troubleshooting
9764	Start position of the step comparison output instruction equal to end position	The start position of the step comparison output instruction of the encoder axis is equal to the end position.	Ensure that the start position of the step comparison output instruction is not equal to the end position.
9765	Start position of the step comparison output instruction less than end position, but step negative	The start position of the step comparison output instruction of the encoder axis is less than the end position, but the step is negative.	Set the step to a positive value.
9766	Start position of the step comparison output instruction greater than end position, but step positive	The start position of the step comparison output instruction of the encoder axis is greater than the end position, but the step is positive.	Set the step to a negative value.
9767	Size of the encoder axis array comparison output instruction out of range	The value of Size of the encoder axis array comparison output instruction is out of range.	Ensure that the parameter value is within the allowable range.
9768	Absolute value of the target position of the encoder axis array comparison output instruction greater than 9999999	The absolute value of the target position of the encoder axis array comparison output instruction is greater than 9999999.	Ensure that the absolute value of the floating-point number in the motion instruction does not exceed 9999999.
9769	Axis performing one-dimensional comparison output, must not be aborted by a two-dimensional comparison output instruction	The axis is performing one-dimensional comparison output and must not be aborted by a two-dimensional comparison output instruction.	Wait for the one-dimensional comparison output to complete or stop the one-dimensional comparison output before executing the two-dimensional comparison output instruction.
9770	EtherCAT slave disconnected during operation	The EtherCAT slave is disconnected during operation.	Check whether the EtherCAT slave is disconnected during operation.
9771	Bus encoder axis in offline commissioning mode	The bus encoder axis is in offline commissioning mode.	The bus encoder axis does not support the offline commissioning mode.
9772	DI terminal not assigned with the preset position function	The DI terminal is not assigned with the preset position function.	Assign the preset position function to the DI terminal before calling the preset position instruction.
9773	Parameter in comparison instruction out of range when the pulse output mode is selected	The value of Parameter in the comparison instruction is out of range when the pulse output mode is selected.	Do not set Parameter to 0 or a negative value when the pulse output mode is selected in the comparison instruction.
9774	2HCE module failed when the comparison output instruction is called	The 2HCE module fails when the comparison output instruction is called.	1. Ensure that the input parameters are within the allowable range. 2. Check whether I/O mapping of the encoder axis is manually modified and whether it meets the I/O mapping configuration requirements of the comparison output instruction.

Fault Code	Message	Description	Troubleshooting
9775	Set position in ring mode less than 0	The set position in ring mode is less than 0.	Set the position in ring mode to a value greater than or equal to 0.
9776	Y00 not assigned with the two-dimensional comparison output function	The Y00 terminal is not assigned with the two-dimensional comparison output function.	Assign the two-dimensional comparison output function to the Y00 output terminal corresponding to the channel.
9777	Axis performing two-dimensional comparison output, cannot be aborted by a one-dimensional comparison output instruction	The axis is performing two-dimensional comparison output and cannot be aborted by a one-dimensional comparison output instruction.	Wait for the two-dimensional comparison output to complete or stop the two-dimensional comparison output before calling the one-dimensional comparison output instruction.
9800	Failed to read the number of motion control axes	Failed to read the number of motion control axes.	Change the background version.
9801	Motion control axis quantity out of range	The number of motion control axes is out of range.	Reduce the number of axes since the H5U supports at most 32 axes.
9802	Axis failed to request internal space	The axis failed to request internal storage space.	1. Check whether the memory runs out. 2. Contact the manufacturer.
9803	Failed to obtain axis parameters	Failed to obtain axis parameters.	Change the background version.
9804	Failed to obtain the slave	Failed to obtain the slave.	Change the background version.
9805	Failed to obtain the system variable	Failed to obtain the system variable.	1. Check whether the memory runs out. 2. Return the machine to the manufacturer for analysis.
9806	Improper gear ratio settings	Parameters related to the gear ratio are set improperly.	1. Ensure that the numerator and denominator of the gear ratio are greater than 0. 2. Ensure that the number of pulses per revolution of the motor/encoder is greater than 0. 3. Ensure that the displacement per revolution of the rotary table is between 0.000001 and 9999999.
9807	Improper software limiting parameters	The software limiting parameters are set improperly.	1. Ensure that the positive limit is not greater than 9999999. 2. Ensure that the negative limit is not greater than 9999999. 3. Ensure that the negative limit is not greater than the positive limit.
9808	Improper linear/rotary mode	The linear/rotary mode parameter is set improperly.	Note that only the linear mode and rotary mode are supported.
9809	Improper revolution cycle	The revolution cycle is set improperly.	Ensure that the revolution cycle is between 0.01 and 9999999.
9810	Improper encoder mode	The encoder mode is set improperly.	Ensure that the encoder mode is set properly. Note that only the incremental mode and absolute value mode are supported.
9811	Improper homing parameter setting	The homing parameter is set improperly.	1. Do not modify the homing mode of the bus servo axis. If you want to modify the homing mode of the bus servo axis, write to the SDO. 2. Check whether the homing mode is set properly. Note that only the values 17 to 30 and 35 are supported.

Fault Code	Message	Description	Troubleshooting
9812	Limit, home, or probe terminal Modbus address out of range	The Modbus address setting of the limit, home, or probe terminal is out of range.	1. Check whether the set address is out of the range of Modbus addresses. 2. Select an address among X0 to X7 for the home signal. 3. Select an address among X0 to X7 for the probe signal.
9813	Improper pulse output mode setting of the local pulse axis	The pulse output mode of the local pulse axis is set improperly.	Check whether the pulse output mode of the local pulse axis is set improperly.
9814	Improper limiting deceleration	The limiting deceleration is set improperly.	Ensure that the limiting deceleration is between 0.0001 and 9999999.
9815	Improper deceleration upon axis fault	The deceleration upon axis fault is set improperly.	Ensure that the deceleration upon axis fault is between 0.0001 and 999999.
9816	Improper maximum velocity	The maximum velocity is set improperly.	Ensure that the maximum velocity is between 0.0001 and 999999.
9817	Improper maximum positive torque	The maximum positive torque is set improperly.	Ensure that the maximum positive torque is between 1 and 65534.
9818	Improper maximum negative torque	The maximum negative torque is set improperly.	Ensure that the maximum negative torque is between 1 and 65534.
9819	Improper maximum jogging velocity	The maximum jogging velocity is set improperly.	Ensure that the maximum jogging velocity is between 0.0001 and the maximum velocity.
9820	Improper maximum acceleration	The maximum acceleration is set improperly.	Ensure that the maximum acceleration is between 0.0001 and 9999999.
9821	Improper following error threshold	The following error threshold is set improperly.	Ensure that the following error threshold is between 0.0001 and 9999999.
9822	Improper velocity reach threshold	The velocity reach threshold is set improperly.	Ensure that the velocity reach threshold is between 0.0001 and 9999999.
9823	Improper homing velocity	The homing velocity is set improperly.	1. Ensure that the homing velocity is between 0.0001 and 9999999. 2. Ensure that the homing velocity is not greater than the maximum velocity. 3. Ensure that the value obtained by multiplying the homing velocity by the gear ratio is between 1 and 2148483647.
9824	Improper homing approach velocity	The homing approach velocity is set improperly.	1. Ensure that the homing approach velocity is between 0.0001 and 9999999. 2. Ensure that the homing approach velocity is not greater than the maximum velocity. 3. Ensure that the value obtained by multiplying the homing approach velocity by the gear ratio is between 1 and 2148483647. 4. Ensure that the homing approach velocity is less than the homing velocity.
9825	Homing position mode setting out of range	The homing position mode setting is out of range.	Ensure that the parameter value is within the allowable range.
9826	Improper homing acceleration	The homing acceleration setting is improper.	1. Ensure that the homing acceleration is between 0.0001 and 9999999. 2. Ensure that the homing acceleration is not greater than the maximum acceleration.
9827	Homing timeout time out of range	The homing timeout time is out of range.	Ensure that the homing timeout time is greater than or equal to 1.

A

B

C

D

E

F

H

N

O

P

R

S

T

U

W

X

Z

19012250A12

Shenzhen Inovance Technology Co., Ltd.

www.inovance.com

Add.: Inovance Headquarters Tower, High-tech Industrial Park, Guanlan Street, Longhua New District, Shenzhen

Tel: (0755) 2979 9595 Fax: (0755) 2961 9897

Suzhou Inovance Technology Co., Ltd.

www.inovance.com

Add.: No. 16 Youxiang Road, Yuexi Town, Wuzhong District, Suzhou 215104, P.R. China

Tel: (0512) 6637 6666 Fax: (0512) 6285 6720