Smarter. Greener. Together.

Industrial Automation Headquarters

Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan City,
Taoyuan County 33068, Taiwan
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia

Delta Electronics (Jiangsu) Ltd.
Wujiang Plant 3
1688 Jiangxing East Road,
Wujiang Economic Development Zone
Wujiang City, Jiang Su Province, P.R.C. 215200
TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
238 Min-Xia Road, Pudong District,
ShangHai, P.R.C. 201209
TEL: 86-21-58635678 / FAX: 86-21-58630003

Delta Electronics (Japan), Inc.
Tokyo Office
2-1-14 Minato-ku Shibadaimon,
Tokyo 105-0012, Japan
TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
1511, Byucksan Digital Valley 6-cha, Gasan-dong,
Geumcheon-gu, Seoul, Korea, 153-704
TEL: 82-2-515-5303 / FAX: 82-2-515-5302

Delta Electronics Int'l (S) Pte Ltd.
4 Kaki Bukit Ave 1, #05-05, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No 43 Sector 35, HSIIDC
Gurgaon, PIN 122001, Haryana, India
TEL : 91-124-4874900 / FAX : 91-124-4874945

Americas

Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3800 / FAX: 1-919-767-8080

Delta Greentech (Brasil) S.A.
Sao Paulo Office
Rua Itapeva, 26 - 3° andar Edificio Itapeva One-Bela Vista
01332-000-São Paulo-SP-Brazil
TEL: 55 11 3568-3855 / FAX: 55 11 3568-3865

Europe

Delta Electronics (Netherlands) B.V.
Eindhoven Office
De Witbogt 20, 5652 AG Eindhoven, The Netherlands
TEL : +31 (0)40-8003800 / FAX : +31 (0)40-8003898

AH-0256920-01

2017-09-20

AH Motion Controller - Motion Control Instructions Manual

AH Motion Controller - Motion Control Instructions Manual

www.deltaww.com

Smarter. Greener. Together.

AH Motion Controller
Motion Control Instruction Manual

Revision History

Version	Revision	Date
1st	The first version was published.	2017/09/20

AH Motion - AH Motion – Motion Control Instructions Manual

Contents

Appendices

P

Preface

Table of Contents

P.1 Introduction

Thank you for purchasing the AH series Motion CPU with our advanced motion control system.

This manual introduces the motion control instructions including single-axis, multi-axes instructions and e-cam applications. Please ensure that you understand the configuration and operations of the AH series motion control system, and use the AH series Motion Controller CPU correctly.

To obtain required information for different system configurations, you can navigate between different manuals of AH Motion series manuals and other related manuals.

P.1.1. Applicable Products

This manual relates to the following products

- AHxxEMC-5A (AH08EMC-5A/AH10EMC-5A/AH20EMC-5A)

P.1.2. Related Manuals

The related manuals of the AH Motion Controller series motion controllers are composed of the following.

1. **AH Motion Controller - Hardware Manual**
   It introduces function specifications, electrical specifications, appearances, dimensions, and etc.

2. **ISPSoft User Manual**
   It introduces the use of ISPSoft, the programming languages (ladder diagrams, sequential function charts, function block diagrams, and structured texts), the concept of POUs, the concept of tasks, and the operation of motion control programming.

3. **AH Motion Controller - Standard Instructions Manual**
   It introduces the elements for standard programming including devices, symbols and standard instructions.

4. **AH Motion Controller - Operation Manual**
   It introduces basic knowledge of motion control structure, software/hardware setup, quick start of Software operations, devices to be used, motion control operations and troubleshooting.

5. **AH Motion Controller - Motion Control Instructions Manual**
   It introduces the elements for motion control programming including devices, symbols and single axis/multi-axes motion instructions.

6. **AH500 Motion Control Module Manual**
   It introduces the specifications for the AH500 series motion control modules, the wiring, the instructions, and the functions.

7. **AH500 Module Manual**
   It introduces the use of special I/O modules of AH500 series PLCs. For example, network modules, analog I/O modules, temperature measurement modules, and etc.

P.2 Navigation between Manuals

Before using the products, there are three manuals that should be utilized as fundamental information: ***AH Motion Controller - Hardware Manual***, ***ISPSoft User Manual***, and ***AH Motion Controller - Standard Instructions Manual.*** With the fundamental manuals, you can understand the basic information of hardware configuration, operation procedures of the software, and the basic instructions for using the system.

To obtain required information for different system configurations and applications, refer to other manuals as indicated in the table below. Reading all manuals related to your system configuration helps you make the most use of the AH series motion control system.

<table>
<tr><th colspan="2" rowspan="3">Related manuals
General operation procedures</th><th colspan="5">AH Motion Controller series manuals</th><th rowspan="3">AH500 Motion Control Module Manual</th><th rowspan="3">AH500 Module Manual</th></tr>
<tr><th colspan="3">Fundamental</th><th rowspan="2">AH Motion Controller – Operation Manual</th><th rowspan="2">AH Motion Controller – Motion Control Instructions Manual</th></tr>
<tr><th>AH Motion Controller – Hardware Manual</th><th>ISPSoft User Manual</th><th>AH Motion Controller – Standard Instructions Manual</th></tr>
<tr><td colspan="2">1. Overview of AH Motion series products</td><td>V</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">2. Setting up hardware configuration for the system</td><td rowspan="5">V</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4"></td><td>for motion control applications</td><td></td><td></td><td>V</td><td></td><td></td><td></td></tr>
<tr><td>for communication (e.g. EtherCAT)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>for additional motion control modules</td><td></td><td></td><td></td><td></td><td>V</td><td></td></tr>
<tr><td>for I/O extension using AH500 series modules</td><td></td><td></td><td></td><td></td><td></td><td>V</td></tr>
<tr><td colspan="2">3. Getting started with the software</td><td rowspan="5"></td><td rowspan="5">V</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>for motion control applications</td><td></td><td>V</td><td></td><td></td><td></td></tr>
<tr><td></td><td>for communication (e.g. EtherCAT)</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>for additional motion control modules</td><td></td><td></td><td></td><td>V</td><td></td></tr>
<tr><td></td><td>for I/O extension using AH500 series modules</td><td></td><td></td><td></td><td></td><td>V</td></tr>
<tr><td colspan="2">4. Programming</td><td rowspan="5"></td><td rowspan="5">V</td><td>V</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>for motion control applications</td><td></td><td>V</td><td>V</td><td></td><td></td></tr>
<tr><td></td><td>for communication (e.g. EtherCAT)</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>for additional motion control modules</td><td></td><td></td><td></td><td>V</td><td></td></tr>
<tr><td></td><td>for I/O extension using AH500 series modules</td><td></td><td></td><td></td><td></td><td>V</td></tr>
</table>

Related manuals / General operation procedures		AH Motion Controller series manuals						
		Fundamental						
		AH Motion Controller – Hardware Manual	ISPSoft User Manual	AH Motion Controller – Standard Instructions Manual	AH Motion Controller – Operation Manual	AH Motion Controller – Motion Control Instructions Manual	AH500 Motion Control Module Manual	AH500 Module Manual
5. Testing and troubleshooting								
	for motion control applications					V*		
	for communication (e.g. EtherCAT)		V		V			
	for additional motion control modules			V*			V	
	for I/O extension using AH500 series modules			V*				V
6. Maintenance and Inspection		V						

Note:** Information regarding Error codes and Indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual.

1

Chapter 1 Introduction to Motion Control Instructions

Table of Contents

1.1 Motion Control Instructions

This manual introduces the elements for motion control programming including devices, symbols and motion control instructions.

Motion control instructions are defined as function blocks (FB) and are used in the program for performing a variety of motion control purposes. The motion control instructions are developed based on the specifications of PLCopen* motion control function blocks.

In addition to the PLCopen-based instructions, there are also Delta-defined function blocks for users to achieve complete motion control applications. This section gives an overview of the motion control instructions for both PLCopen-based function blocks and Delta-defined function blocks. PLCopen defines the program and function block interfaces so as to achieve a standardized motion control programming environment for the languages specified in IEC61131-3. Using PLCopen-based instructions together with Delta-defined instructions reduces the costs for training and support. Also, the program can be adjusted and reused in different controllers easier.

Before using the instructions, please be sure that you understand the devices, symbols and the function of instructions sufficiently.

You can also refer to the **Appendices** for a quick reference of the motion control instruction list and error codes.

***Note:**
PLCopen is an organization promoting industrial control based on IEC61131-3, which is an international standard widely adopted for PLC programming. For more information regarding PLCopen, check the official website at: http://www.plcopen.org/

1.1.1 Fundamentals of Motion Control Instructions

Using motion control instructions requires fundamental knowledge defined in the specifications of PLCopen motion control function blocks. This subsection gives an overview of these specifications.

- **Names of Motion Control Instructions**

PLCopen-based motion control instructions begin with “MC_”, and Delta-defined function block instructions begin with”DFB_”.

Types	Description
MC_	PLCopen-based motion control instructions
DFB_	Delta-defined function block instructions*

Note:** Delta-defined function block instructions (DFB) include Delta-defined motion control function blocks and other administrative/non-administrative function blocks which are applicable for AH Motion series CPUs. Therefore you can look up a function block (FB) in this manual and a function (FC) in ***AH Motion Controller – Standard Instructions Manual which includes all applicable functions (FC). For further explanation of Function(FC) and Function Block(FB) and the software interface for using these instructions, refer to ***ISPSoft User Manual***.

Types of Motion Control Related Instructions

Different categories of motion control instructions are divided by functions such as single-axis motion instructions, multi-axis motion instructions, auxiliary motion instructions, and network instructions. Refer to **Ch3 Motion Control Instructions** for more details.

State Transitions

PLCopen specifications also define motion states and their transition behaviors. For details on the state transitions, refer to ***AH Motion Controller - Operation Manual*** for more information.

Execution and Status Indication of a Function Block

Function block instructions generally include two types of inputs for execution: *Execute* and *Enable.* When the instruction is executed or enabled, the outputs of the function block can indicate the status. The basic outputs include *Busy*, *Done*, *CommandAborted*, and *Error*. For detailed information of inputs and outputs of each function block, refer to **Ch3 Motion Control Instructions.**

Error Handling

Information regarding error codes and indicators are attached as Appendices for a quick reference in ***AH Motion Controller - Standard Instructions Manual***, and ***AH Motion Controller - Motion Control Instructions Manual.*** The detailed troubleshooting procedures and error codes are explained in ***AH Motion Controller - Operation Manual***.

Re-execution of a Function Block

Re-execution of a function block refers to triggering *Execute* again after resetting it. If the input values are changed and *Execute* is triggered again while the function block is during operation (in busy status), the action of triggering *Execute* will be ignored and the input values will not be updated. The function block will finish its on-going operation with the original input values.

Buffer Modes

Some motion instructions have an input called *BufferMode*. You can execute a different instruction instance during axis motion when the values for *BufferMode* are specified. This input decides whether the instruction executes immediately (non-buffered mode) or it waits till current motion instruction sets its status outputs (*Done*/*InVelocity*/*InPosition*, etc.)

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in Standstill state.

The following Buffer Modes are supported.

Buffer Mode	Function
0: Aborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: Buffered	Automatically executes the next instruction after the ongoing motion is completed.
2: BlendingLow	Takes the lower target velocity as the transit velocity between the current instruction and the buffered instruction. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: BlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.
4: BlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.
5: BlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

■ When a DFB_ Instruction is Followed by a BufferMode Instruction (MC_)

DFB_ instructions do not have the *BufferMode* input. Therefore, care should be taken when an on-going motion driven by a DFB_ instruction (e.g. DFB_TrSeg1) is followed by a buffer-mode instruction (e.g. MC_MoveVelocity) during axis motion status. When the MC_ instruction is executed, the DFB_ instruction will be aborted, and the following MC_ instruction will report an error.

● Structures Applicable for Motion Control

In PLCopen technical standard, the information and parameters required for configuring motion control on axis are defined in a Structure, which is a kind of the DUT(data type unit) provided by the ISPSoft software. A Structure is a data type applicable to group the data elements together, which is easier for users to specify proper parameters

For AH Motion CPUs, the applicable Structure is AXIS_REF as below.

Structures		Definition
PLCopen	Motion Control FBs	
AXIS_REF*	**Applicable for MC_ / DFB_ function blocks**	**Information and parameters required for configuring axis motion is grouped in this Structure.**
AXES_GROUP_REF	N/A	Information and parameters required for configuring axes group motion is included in this Structure.
TRIGGER_REF	**MC_TouchProbe** **MC_AbortTrigger**	**Information on trigger inputs** ● **Specifications of the trigger** ● **Trigger activation patterns (rising-edge, falling edge, and etc.)**
INPUT_REF	N/A	Information relating to the inputs. Virtual data could be included.
OUTPUT_REF	N/A	Information relating to physical outputs.

***Note**: refer to the **2.2.1 Parameters for Motion Axes: Structure** of this manual for the list of parameters grouped in the Structure AXIS_REF which is supported by AH Motion CPU.

1.2 Points to Note about Motion Control Instructions

This section explains important specifications and limitations when applying motion control instructions. For detailed information of each instruction in this manual, refer to **Chapter 3 Motion Control Instructions**.

1

● Programming Languages for Motion Control Instructions

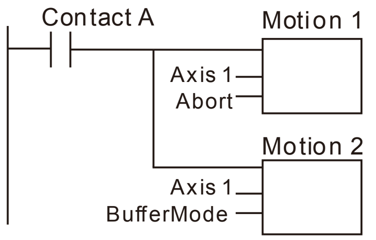
You can use all programming languages provided by ISPSoft to create, edit, or maintain the program. The supported languages include Ladder Diagram (LD), Sequential Function Chart (SFC), Continuous Function Chart (CFC) and Structured Text (ST).

For detailed information about the programming languages, refer to ***ISPSoft User Manual***.

● Multi-execution of Multiple Motion Control Instructions

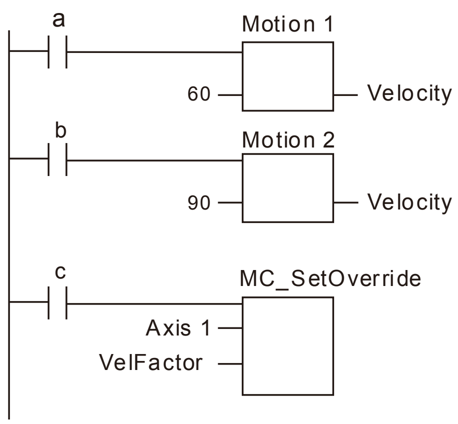
Multi-execution of motion control instructions refers to that multiple instructions on the same axis are executed in the same task execution period.

- In the following program, motion 1 and motion 2 starts in the same task execution period when contact A is ON.
- In the ladder logic, instructions are executed according to the top-down order*, therefore motion 1 is started first and motion 2 started later before motion 1 is completed.
- This situation is considered as multi-execution of multiple motions. The pattern of multi-execution is defined by the input variable *BufferMode* which is specified to blend the two motions. In this program, *BufferMode* of Motion 2 will take effects to blend the two motions.



For descriptions of each buffer mode, refer to **1.1.1 Fundamentals of Motion Control Instructions.**

***Note**: When MC_SetOverride is activated simultaneously in the same condition as above diagram, the values to be apply for MC_SetOverride still take effects first even if it is inserted on the bottom. Therefore, if MC_SetOverride is to be used, the below programming is recommended.



Precautions for Operation Modes

When there are sudden changes of the velocity or the position in synchronized motion of master axis, errors could occur. Refer to below precautions and points to note for avoiding excessive operation for the machine.

Sudden Changes of the Velocity

If the velocity of the master axis is suddenly changed during synchronized motion, the motion on the slave axis could also be changed significantly, and the equipment could be impacted with the excessive operation. Therefore precautions are required in below situations which may cause sudden changes on the master axis.

- If any of the following instructions is executed on master axis:

  MC_Stop

  MC_SetPosition

  To avoid sudden changes on the slave axis, be careful on setting the input parameters and the activation timing for instructions above. You can also execute the above instructions after the synchronized motion is disengaged.
- If an immediate stop signal is triggered for the master axis;
- If the servo is turned off for the master axis;

  If the servo is turned off when the master axis works as a vertical axis, a sudden change on the position of the axis could happen. In this case, proper measures should be applied to prevent the slave axis from sudden displacement. You can design a brake for the master axis or stop the servo only when the synchronized motion is disengaged.

Conditions Causing Errors

When one of the below situations occurs on the master axis which starts a synchronized motion or is during a synchronized motion, a master axis position reading error will occur on the slave axis. In this case, *CommandAborted* output will shifts to True at the same time.

- Process data objects (PDO) are not established for EtherCAT communications.
- EtherCAT slave communication error occurs due to incorrect EtherCAT communications settings.
- Absolute encoder failed to calculate the current position and thus error occurs.
- The slave loses connection with the master.

Note: When MC_Home instruction is started on the master axis, the slave will ignore the sudden position changes on the master axis. This behavior prevents the slave from fluctuating along with the homing process of the master axis.

1.3 Categories of Motion Control Instructions

Categories	Type	Function Group	Description
Single-axis Motion Control Instructions	Motion	Positioning on single axis	"MC_" PLCopen-based motion control instructions "DFB_" Delta-defined motion control instructions
		Velocity control on single axis	
		Torque control on single axis	
		Synchronized control on single axis	
		Manual operation on single axis	
	Administrative	Administrative functions on single axis	
Multi-axis Motion Control Instructions	Motion	G-codes	Numerical Control
		Group Motion	Coordinated Control
	Administrative	Administrative functions on multiple axes	M-codes Administrative functions on G-codes and group motion
Auxiliary Instructions	Administrative	Supporting functions for configuring the system	High speed counters, high speed timers, high speed capture and comparison
Network instructions	Administrative	Supporting functions for setting the networks	Motion network settings

MEMO

2

Chapter 2 Devices, Symbols and Instructions

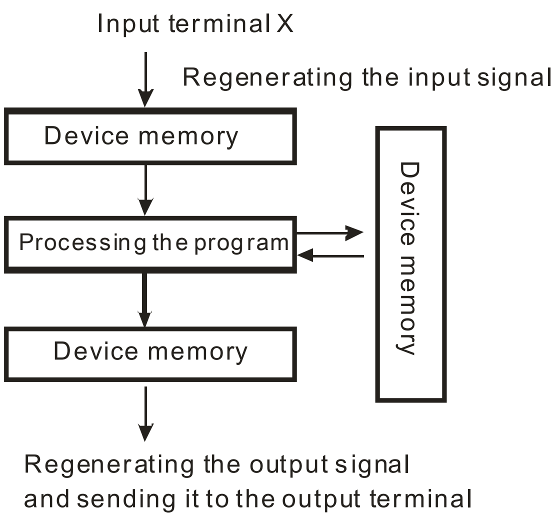
Table of Contents

2.1 Common Devices

This section describes the concept of common devices which include input/output/auxiliary relays, timers, counters, and data registers. For detailed descriptions on functions as well as characteristics of devices, you can refer to **AH Motion Controller – Operation Manual**. For details of motion control devices, refer to **2.2 Motion Control Devices.**

2.1.1. Functions of Common Devices

The procedure for processing the program in the PLC:



- Regenerating the input signal:
  1. Before the program is executed, the state of the external input signal is read into the memory of the input signal.
  2. When program is executed, the state in the memory of the input signal does not change even if the input signal changes from ON to OFF or from OFF to ON. Not until the next scan begins will the input signal be refreshed.
- Processing the program:
  After the input signal is refreshed, the instructions in the program are executed in order from the start address of the program, and the results are stored in the device memories.
- Regenerating the state of the output:
  After the instruction END is executed, the state in the device memory is sent to the specified output terminal.

2.1.2. Common Device List

Type	Device name		Number of devices	Range
Bit device	Input relay	X	8192	X0.0~X511.15 (Supporting Force ON/OFF)
	Output relay	Y	8192	Y0.0~Y511.15 (Supporting Force ON/OFF)
	Data register	D	1048576	D0.0~D65535.15
	Link registers	L	1048576	L0.0~ L65535.15
	Auxiliary relay	M	8192	M0~M8191
	Special auxiliary relay	SM	SM: 2048	SM0~SM2047
	Stepping relay	S	2048	S0~S2047
	Timer	T	2048	T0~T2047
	Counter	C	2048	C0~C2047
	32-bit counter	HC/AC	HC: 64 AC: 56 (AH10EMC)	HC0~HC63 AC0~AC55 (AH10EMC)

Type	Device name		Number of devices	Range
Word device	Input relay	X	512	X0~X511
	Output relay	Y	512	Y0~Y511
	Data register	D	65536	D0~D65535
	Special data register	SR	SR: 2048	SR0~SR2047
	Link registers	L	1048576	L0~ L65535
	Timer	T	2048	T0~T2047
	Counter	C	2048	C0~C2047
	32-bit counter	HC/AC	HC: 64 AC: 56 (AH10EMC)	HC0~HC63 AC0~AC55 (AH10EMC)
	Index register	E	32	E0~E31
Constant*	Decimal system	K	16 bits: -32768~32767 32 bits: -2147483648~2147483647	
	Hexadecimal system	16#	16 bits: 16#0~16#FFFF 32 bits: 16#0~16#FFFFFFFF	
	Single-precision floating-point number	F	32 bits: $\pm1.17549435^{-38}$~$\pm3.40282347^{+38}$	
	Double-precision floating-point number	DF	64 bits: $\pm2.2250738585072014^{-308}$~ $\pm1.7976931348623157^{+308}$	
String*	String	"$"	1~31 characters	
Pointer*	Pointer	PR		

*1: The decimal forms are notated by K in the device table in ***AH Motion - Standard Instructions Manual***, whereas they are entered directly in ISPSoft. For example, entering 50 in ISPSoft indicates the value K50.

*2: The floating-point numbers are notated by F/DF in the device table in ***AH Motion - Standard Instructions Manual***, whereas they are represented by decimal points in ISPSoft. For example, entering 500.0 in ISPSoft indicates the value F500.

*3: The strings are notated by $ in ***AH Motion - Standard Instructions Manual***, whereas they are represented by adding quotes (" ") to the value in ISPSoft. For example, entering "1234" in ISPSoft indicates the string 1234.

2.1.3. Latched Devices

- Latched areas of each type of device

Device	Function	Device range	Latched area
X	Input relay	X devices (bit): X0.0~X511.15 X devices (word): X0~X511	All devices are non-latched.

Device	Function	Device range	Latched area
Y	Output relay	Y devices (bit): Y0.0~Y511.15 Y devices (word): Y0~Y511	All devices are non-latched.
M*	Auxiliary relay	M0~M8191	Default: M0~M8191
SM	Special auxiliary relay	SM: SM0~SM2047	Some devices are latched, and can not be changed. Refer to the table of special auxiliary relays in the appendices for more information.
S	Stepping relay	S0~S2047	All devices are non-latched.
T*	Timer	T0~T2047	Default: T0~T2047.
C*	Counter	C0~C2047	Default: C0~C2047.
HC/AC*	32-bit counter	HC: HC0~HC63 AC: AC0~ AC55 (AH10EMC)	Default: HC0~HC63. AC devices are non-latched.
D*	Data register	D device (bit): D0.0~D65535.15 D device (word): D0~D65535	Default: D0~D32767. At most 32768 devices can be latched devices.
SR	Special data register	SR: SR0~SR2047	Some are latched, and can not be changed. Refer to the list of special data registers for more information.
L	Link register	L0~ L65535	All devices are non-latched.
E	Index register	E0~E31	All devices are non-latched.

***Note**: You can define the range of latched areas for T, C, HC and D and set the devices to be non-latched. Note that the range should not exceed the available device range, and only 32768 D devices at most can be non-latched. For example, you can set D50~D32817 or D32768~D65535 to latched area though the default range of latched areas is D0~D32767.

- Behavior of non-latched and latched devices

PLC action \ Device type		Non-latched	Latched	Output relay
Power: OFF→ON		Cleared	Retained	Cleared
STOP ↓ RUN	**The output relay is cleared.**	Retained	Retained	Cleared
	The state of the output relay is retained.	Retained	Retained	Retained
	The state of the output relay returns to that before the PLC's stopping.	Retained	Retained	The state of the output relay returns to that before the PLC's stopping.
	The non-latched devices are cleared.	Cleared	Retained	Cleared
	The state of latched devices is retained.	Retained	Retained	Retained
RUN→STOP		Retained	Retained	Retained

PLC action \ Device type	Non-latched	Latched	Output relay
SM204 is ON. (All non-latched devices are cleared.)	Cleared	Retained	Cleared
SM205 is ON. (All latched devices are cleared.)	Retained	Cleared	Retained
Default value	0	0	0

2.1.4. Input Relays (X)

- The function of the input

  The input is connected to the input device (e.g. external devices such as button switches, rotary switches, number switches, and etc.), and the input signal is read into the PLC. Besides, contact A or contact B of the input can be used several times in the program, and the ON/OFF state of the input varies with the ON/OFF state of the input device.

- The input number (the decimal number)

  For the PLC, the input numbers start from X0.0. The number of inputs varies with the number of inputs on the digital input/output modules, and the inputs are numbered according to the order in which the digital input/output modules are connected to the CPU module. The maximum number of inputs on the PLC can reach up to 8192, and the range is between X0.0 and X511.15.

- The input type

  The inputs are classified into two types.

  1. Regenerated input: Before the program is executed, the data is fed into the PLC according to the states of the inputs which are regenerated. For example, LD X0.0.
  2. Direct input: During the execution of the instructions, the data is fed into the PLC according to the states of the inputs. For example, LD DX0.0.

2.1.5. Output Relays (Y)

- The function of the output

  The task of the output is sending the ON/OFF signal to drive the load connected to the output. The load can be an external signal lamp, a digital display, or an electromagnetic valve. There are three types of outputs. They are relays, transistors, and TRIACs (AC thyristors). Contact A or contact B of the output can be used several times in the program, but the output should be used only once in the program. Otherwise, according the program-scanning principle of the PLC, the state of the output depends on the circuit connected to the last output in the program.

- The output number (the decimal number)

  For the PLC, the input numbers start from X0.0. The number of outputs varies with the number of outputs on the digital input/output modules, and the outputs are numbered according to the order in which the digital input/output modules are connected to the PLC. The maximum number of outputs on the PLC can reach up to 8192, and the range is between Y0.0 and Y511.15.

  The output which is not practically put to use can be used as a general device.

- The output type

  The outputs are classified into two types.

1. Regenerated output: Not until the program executes the instruction END is the information fed out according to the states of the outputs. For example, OUT Y0.0.
2. Direct output: When the instructions are executed, the information is fed out according to the states of the outputs. For example, OUT DY0.0.

2.1.6. Auxiliary Relays (M)

The auxiliary relay has contact A and contact B. It can be used several times in the program. You can combine the control loops by means of the auxiliary relay, but can not drive the external load by means of the auxiliary relay. The auxiliary relays can be divided into two types according to their attributes.

1. For general use: If an electric power cut occurs when the PLC is running, the auxiliary relay for general use will be reset to OFF. When the power supply is restored, the auxiliary relay for general use is still OFF.
2. For latched use: If an electric power cut occurs when the PLC is running, the state of the auxiliary relay for latched use will be retained. When the power supply is restored, the state remains the same as that before the power electric cut.

2.1.7. Special Auxiliary Relays (SM)

SM: special auxiliary relays

Every special auxiliary relay has its specific function. Please do not use the special auxiliary relays which are not defined.

For function descriptions of each special auxiliary relay (SM), refer to **Appendix 1: Special Auxiliary Relays Table** of ***AH Motion Controller - Standard Instructions Manual.***

2.1.8. Data Registers (D)

The data register stores the 16-bit data. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -32,768 to +32,767. Two 16-bit registers can be combined into a 32-bit register, i.e. (D+1, D) in which the register whose number is smaller represents the low 16 bits. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -2,147,483,648 to +2,147,483,647. Besides, four 16-bit registers can be combined into a 64-bit register, i.e. (D+3, D+2, D+1, D) in which the register whose number is smaller represents the lower 16 bits. The highest bit represents either a positive sign or a negative sign, and the values which can be stored in the data registers range from -9,223,372,036,854,776 to +9,223,372,036,854,775,807. The data registers also can be used to refresh the values in the control registers in the modules other than digital I/O modules. Please refer to ISPSoft User Manual for more information regarding refreshing the values in the control registers.

The registers can be classified into two types according to their properties:

1. General-purpose register: If the PLC begins to run, or is disconnected, the value in the register will be cleared to zero. If you want to retain the data when the PLC begins to RUN, you can refer to ISPSoft User Manual for more information. Please notice that the value will still be cleared to zero if the PLC is disconnected.
2. Latched register: If the PLC is disconnected, the data in the latched register will not be cleared. In other words, the value before the disconnection is still retained. If you want to clear the data in the latched area, you can use RST or ZRST.

2.1.9. Special Data Registers (SR)

SR: special data registers.

Every special data register has its definition and specific function. The system statuses and the error messages are stored in the special data registers. Besides, the special data registers can be used to monitor the system statuses.

For function descriptions of each special data register (SR), refer to **Appendix 2: Special Deta Registers Table** of ***AH Motion Controller - Standard Instructions Manual.***

2.1.10. Link Registers (L)

The link register is mainly used in for automatic data exchange function. When the data exchange occurs between the AH10EMC series PLCs, the link register can be used as the buffer.

The link registers L0~L65535 have 65536 words and can be used as the common auxiliary registers.

2.1.11. Stepping Relays (S)

The function of the stepping relay:

The stepping relay can be easily used in the industrial automation to set the procedure. It is the most basic device in the sequential function chart (SFC). Please refer to ISPSoft User Manual for more information related to sequential function charts.

There are 2048 stepping relays, i.e. S0~S2047. Every stepping relay is like an output relay in that it has an output coil, contact A, and contact B. It can be used several times in the program, but it can not directly drive the external load. Besides, the stepping relay can be used as a general auxiliary relay when it is not used in the sequential function chart.

2.1.12. Timers (T)

1. 100 millisecond timer: The timer specified by the instruction TMR takes 100 milliseconds as the timing unit.
2. 1 millisecond timer: The timer specified by the instruction TMRH takes 1 millisecond as the timing unit.
3. The timers for the subroutine's exclusive use are T1920~T2047.
4. The accumulative timers are ST0~ST2047. If you want to use the device-monitoring function, you can monitor T0~T2047.
5. If the same timer is used repeatedly in the program, including in different instructions TMR and TMRH, the setting value is the one that the value of the timer matches first.
6. If the same timer is used repeatedly in the program, it is OFF when one of the conditional contacts is OFF.
7. If the same timer is used repeatedly in the program as the timer for the subroutine's exclusive use and the accumulative timer in the program, it is OFF when one of the conditional contacts is OFF.
8. When the timer is switched from ON to OFF and the conditional contact is ON, the timer is reset and counts again.
9. When the instruction TMR is executed, the specified timer coil is ON and the timer begins to count. As the value of the timer matches the setting value, the state of the contact is as follows.

Normally open (NO) contact	ON
Normally closed (NC) contact	OFF

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion Controller - Operation Manual*** for more information.

2.1.13. Counters (C)

The function of the counter:

Each time the input switches from OFF to ON, the value of the counter increases by one increment. When the value of the counter matches the setting value, the output coil is ON. You can use either the decimal constant or the value in the data register as the setting value.

The 16-bit counter:

1. Setting range: 0~32,767 (The setting values 0 and 1 mean the same thing in that the output contact is ON when the counter counts for the first time.)
2. For the general-purpose counter, the current value of the counter is cleared when there is a power cut. If the counter is the latched one, the current value of the counter and the state of the contact before the power cut will be retained. The latched counter counts from the current value when the power supply is restored.
3. If you use the instruction MOV or ISPSoft to transmit a value bigger than the setting value to the current value register C0, the contact of the counter C0 will be ON and the current value will become the same as the setting value next time X0.1 is switched from OFF to ON.
4. You can use either the constant or the value in the register as the setting value of the counter.
5. The setting value of the counter can be a positive or a negative. If the counter counts up from 32,767, the next current value becomes -32,768.

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion Controller - Operation Manual*** for more information.

2.1.14. 32-bit Counters (HC/AC)

HC: The 32-bit general-purpose addition/subtraction counter

AC: The 32-bit counters used specifically for motion axis. The function of **AC** is the same as that of **HC**

1. Setting range: -2,147,483,648~2,147,483,647
2. The switch between the 32-bit general-purpose addition counters and the 32-bit general-purpose subtraction counters depends on the states of the special auxiliary relays SM621~SM684. For example, the counter HC0 is the addition counter when SM621 is OFF, whereas HC0 is the subtraction counter when SM621 is ON.
3. You can use either the constant or the value in the data registers as the setting value of the counter, and the setting value can be a positive or a negative. If you use the value in the data registers as the setting value of the counter, the setting value occupies two consecutive registers.
4. For the general-purpose counter, the current value of the counter is cleared when there is a power cut. If the counter is the latched one, the current value of the counter and the state of the contact before the power cut will be retained. The latched counter counts from the current value when the power supply is restored.
5. If the counter counts up from 2,147,483,647, the next current value becomes -2,147,483,648. If the counter counts down from -2,147,483,648, the next current value becomes 2,147,483,647.

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion - Operation Manual*** for more information.

2.1.15. Values and Constants (K, 16#)

The PLC uses four types of values to execute the operation according to different control purposes. The functions of these values are illustrated as follows:

1. Binary number (BIN)

   The PLC adopts the binary system to operate the values.

2. Decimal number (DEC)

   The decimal number in the PLC is used as

   - the setting value of the timer (T) or the setting value of the counter (C/HC). For example, TMR C0 <u>50</u> (**constant K**).

- the device number. For example, M10 and T30 (device number)
- the number before or after the decimal point. For example, X0.0, Y0.11, and D10.0 (device number).
- **the constant K**: It is used as the operand in the applied instruction. For example, MOV 123 D0 **(constant K)**.

3. Binary-coded decimal (BCD)

   A decimal value is represented by a nibble or four bits, and therefore sixteen consecutive bits can represent a four-digit decimal value.

4. Hexadecimal number (HEX)

   The hexadecimal number in the PLC is used as

- **the constant 16#**: It is used as the operand in the applied instruction. For example, MOV 16#1A2B D0 (hexadecimal constant).

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion Controller - Operation Manual*** for more information.

2.1.16. Floating-point Numbers (F, DF)

The floating-point numbers are represented by decimal points in ISPSoft. For example, the floating-point number of 500 is 500.0.

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion Controller - Operation Manual*** for more information.

2.1.17. Strings (“$”)

What strings can process are ASCII codes (*1). A complete string begins with a start character, and ends with an ending character (NULL code). If what you enter is a string, you can enter 31 characters at most, and the ending character 16#00 will be added automatically in ISPSoft.

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion Controller - Operation Manual*** for more information.

2.1.18. Pointers (PR)

- **ISPSoft** supports the function blocks. When the symbol declaration type is VAR_IN_OUT, and the data type is POINTER, the symbol will be assigned as pointer registers (PR). The value in the PR can refer directly to the value of device X, Y, D, or L, and the PR can also refer to the symbols with addresses which are automatically allocated by ISPSoft.
- You can declare 16 pointer registers in every function block: PR0~PR15 or PR0.0~PR15.15.

2.1.18.1 Pointer Registers of Timers (T_Pointer) (TR)

- **ISPSoft** supports the function blocks. If you want to use the timer in the function block, you have to declare a pointer register of the timer (TR) in the function block. The address of the timer is transmitted to TR when the function block is called.
- When the symbol declaration type is VAR_IN_OUT, and the data type is T_POINTER, the symbol will be assigned as TR. The value in the TR can refer directly to the value of device T or the symbols which are assigned as timers by **ISPSoft**.
- You can declare 8 pointer registers of the timers in every function block: TR0~TR7.

- If you want to use an instruction in the function block, and the timer is supported among the operands, you have to use the pointer register of the timer.

2.1.18.2 Pointer Registers of 16-bit Counters (C_Pointer) (CR)

- **ISPSoft** supports the function blocks. If you want to use the 16-bit counter in the function block, you have to declare a pointer register of the 16-bit counter (CR) in the function block. The address of the 16-bit counter is transmitted to the CR when the function block is called.
- When the symbol declaration type is VAR_IN_OUT, and the data type is C_POINTER, the symbol will be assigned as CR. The value in the CR can refer directly to the value of device C or the symbols which are assigned as 16-bit counters by **ISPSoft**.
- You can declare 8 pointer registers of the 16-bit counters in every function block: CR0~CR7.
- If you want to use an instruction in the function block, and the counter is supported among the operands, you have to use the pointer register of the 16-bit counter.

2.1.18.3 Pointer Registers of 32-bit Counters (HC_Pointer) (HCR)

- **ISPSoft** supports the function blocks. If you want to use the 32-bit counter in the function block, you have to declare a pointer register of the 32-bit counter (HCR) in the function block. The address of the 32-bit counter is transmitted to the HCR when the function block is called.
- When the symbol declaration type is VAR_IN_OUT, and the data type is HC_POINTER, the symbol will be assigned as HCR. The value in the HCR can refer directly to the value of device HC or the symbols which are assigned as 32-bit counters by ISPSoft.
- You can declare 8 pointer registers of the 32-bit counters in every function block: HCR0~HCR7.
- If you want to use an instruction in the function block, and the 32-bit counter is supported among the operands, you have to use the pointer register of the 32-bit counter.

Refer to **Chapter 5: Understanding Common Devices** of ***AH Motion Controller - Operation Manual*** for more information.

2.2 Motion Control Devices

Motion control devices are manly used for configuring parameters for motion axis. In most applications, you can set up axis parameters in ISPSoft software, which is a convenient environment for users. Axis parameters required for configuring motion control on axis are defined as Structures. A Structure is a data type applicable to group the data elements together. You can refer to ***ISPSoft User Manual*** for the operation of using structures.

2.2.1 Parameters for Motion Axes: Structure

For AH Motion CPUs, the applicable structure for setting up axis parameters is AXIS_REF. The below table can be used in case there is a need to change axis parameters by modifying specific data registers. The detailed explanation also helps you to have a proper understanding of the axis parameters.

- **The axis parameters for motion axes**

AXIS_REF		Description			
Members		**Data type**	**Function**	**Setting range**	**Factory setting (Default)**
Parm_setting		WORD	Setting the parameters of the axis specified	Bit 0~bit15	0
Pulse_of_1Rev		DINT	Number of pulses it takes for the motor of the axis specified to rotate once	1~99999999 pulses/revolution	10,000
Parm_setting	32	DINT	Distance generated after the motor of the axis specified rotate once	1~1000000 Userunit/ revolution	10,000
Pulse_of_1Rev	64	LREAL			
Maximum_Speed	32	DINT	Maximum speed (V_{MAX}) at which the axis specified rotates	0~2,147,483,647	100,000
Max_Speed_f	64	LREAL			
Start_up_speed	32	DINT	Start-up speed (V_{BIAS}) at which the axis specified rotates	0~100,000	0
Start_up_speed_f	64	LREAL			
JOG_speed	32	DINT	JOG speed (V_{JOG}) at which the axis specified rotates	0~(2^{31}-1)	5,000
Target_JOG_speed_f	64	LREAL		0~1.7976931348*(10^{308})	5,000
Homing_speed_for_switch		DINT	Speed (V_{RT}) at which the axis specified returns home (0.1RPM)	0~2,147,483,647	2,000
Homing_speed_for_zero		DINT	Speed (V_{CR}) to which the speed of the axis specified decreases when the axis returns home (0.1RPM)	0~2,147,483,647	100
Homing_position		DINT	Home position of the axis specified (User Unit)	-(2^{31})~(2^{31}-1)	0

AXIS_REF		Description			
Members		**Data type**	**Function**	**Setting range**	**Factory setting (Default)**
Max_Accelerate_time		INT	Maximum acceleration time (TACC); unit: ms	0~1,000 ms	500
Max_Decelerate_time		INT	Maximum deceleration time (TDEC); unit: ms	0~1,000 ms	500
Target_cmd_position1	32	DINT	Target position of the axis specified (User Unit)	-(2^{31})~(2^{31}-1)	0
Target_cmd_position_f	64	LREAL		(+-)1.7976931348 *(10^{308})	
Target_cmd_speed1	32	DINT	Speed at which the axis specified rotates (User Unit)	0~(2^{31}-1)	0
Target_cmd_speed_f	64	LREAL		0~1.7976931348*(10^{308})	
Current_cmd_position_UU	32	DINT	Present command position of the axis specified (User Unit)	-(2^{31})~(2^{31}-1)	0
Current_cmd_position_UU_f	64	LREAL		（+-）1.7976931348*（10^{308}）	0
Current_actual_speed	32	DINT	Present command speed of the axis specified (User Unit/S)	0~(2^{31}-1) PPS	0
Current_actual_speed_UUperS_f	64	LREAL		0~1.7976931348*(10^{308})	0
Position_Lim_Positive_f		LREAL	Positive dirction position limit (User Unit)	0~1.7976931348*(10^{308})	2,147,483,647
Position_Lim_Negative_f		LREAL	Negative dirction position limit (User Unit)	0~-1.7976931348*(10^{308})	2,147,483,647
Max_Acceleration_f		LREAL	Maximum Acceleration (User Unit /S^2)	0~1.7976931348*(10^{308})	1,000
Target_cmd_Acceleration_F		LREAL	Target acceleration of the axis specified (User Unit /S^2)	0~1.7976931348*(10^{308})	1,000
Max_Deceleration_f		LREAL	Maximum Dcceleration (User Unit /S^2)	0~1.7976931348*(10^{308})	1,000
Target_cmd_Deceleration_f		LREAL	Target deceleration of the axis specified (User Unit /S^2)	0~1.7976931348*(10^{308})	200,000

2

AXIS_REF	Description			
Members	Data type	Function	Setting range	Factory setting (Default)
Target_cmd_Jerk_f	LREAL	Target jerk of the axis specified (User Unit /S^3)	0~1.7976931348*(10^{308})	0
Max_position_of_Rotary_Axis_f	LREAL	Maximum position of rotational axis (User Unit)	0~2,147,483,647	2,147,483,647
Min_position_of_Rotary_Axis_f	LREAL	Minimum position of rotational axis (User Unit)	-2,147,483,647~0	-2,147,483,647
Current_Axis_error_code	WORD	Axis error code	Please refer to the error code tables in appendix A.	0
Egear_ratio_Numerator	INT	Electronic gear ratio of the axis specified (Numerator)	1~99,999,999	128
Egear_ratio_Denominator	INT	Electronic gear ratio of the axis specified (Denominator)	1~99,999,999	1
Electrical_zero	DINT	Electrical zero of the axis specified	Users can set the value according to their needs.	0
StateMachine_AxisStatus*[1]	WORD	State of the designed axis	Refer to the table below	2

***[1]: State of the designed axis**

Axis number	Description
2	Disabled
3	Standstill
4	Homing
5	Stopping
6	ContinuousMotion
7	SynchronizedMotion
8	DiscreteMotion
9	Coordinated
10	CoordinatedHalt
11	CoordinatedStop

2.3 EtherCAT Symbols

The EtherCAT symbols are used to read states of EtherCAT slave. The symbols can be seen from the EtherCAT symbol table in the ISPSoft software.

EtherCAT Symbol	Data type	Description	Range	Default settings
eCAT_Dis.SlaveAddress[…]	WORD[199]	The EtherCAT slave address that disconnected (n=0~198)	0~9999	0
eCAT_Dis.Count	WORD	The number of EtherCAT slave address that disconnected	0~ (2^{16}-1)	0

2.4 Symbols

During the process of developing a traditional program for a PLC, it generally takes much time to manage device addresses. Besides, managing or debugging the program in a big project is a burden on users. As a result, the concept of symbols in a high-level programming language is introduced into IEC 61131-3. A device in a PLC can be represented by a symbol, and a device can be automatically assigned to a symbol. The time of assigning devices is saved, a program is more readable, and the efficiency of developing a program increases. Variables in ISPSoft are called symbols. As a result, variables are the same as symbols in terms of meaning in this manual.

2.4.1 Application of Symbols

A symbol has to be declared before it is used. There are two types of symbols. They are global symbols and local symbols. The global symbols in a project can be used in all the POUs* in the project, and the local symbols in a project can only be used in the POU in which the local symbols are declared. Besides, the identifier of a local symbol in a POU can be the same as the identifier of a local symbol in another POU. However, if the identifier of a local symbol declared in a POU is the same as the identifier of a global symbol, the system will automatically regard the local symbol declared in the POU as a local symbol.

***Note**: For further explanations regarding a POU, refer for **ISPSoft User Manual.**

2.4.2 Classes

Symbols can be classified into four classes. The characteristics of these four classes are described below.

- **VAR** - **General symbol**

  The symbols of this class are for general operations only. The significance of a symbol of this class depends on the data type of the symbol.

- **INPUT** - **Symbol used as an input pin of a function block**

  A symbol of this class is used as an input pin of a function block. It can only be declared in the function block. If a function block is called, the symbols of this class can receive the input values sent by the caller. Besides, in a ladder diagram, the symbols of this class are put at the left sides of function blocks, and the pins which receive the input values sent by the caller are assigned to the symbols of this class.

- **OUTPUT** - **Symbol used as an output pin of a function block**

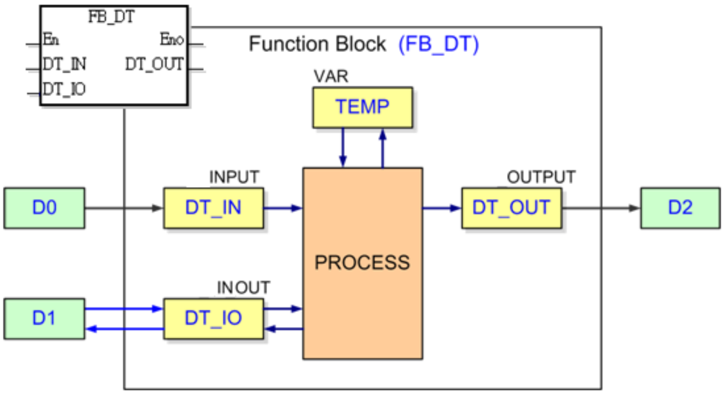
  A symbol of this class is used as an output pin of a function block. It can only be declared in a function block. After the execution of a function block is complete, the operation result will be sent to the caller through the symbols of this class.

2

Besides, in a ladder diagram, the symbols of this class are put at the right sides of function blocks, and the pins which send the operation results to the caller are assigned to the symbols of this class.

- **INOUT - Symbol used as a feedback pin of a function block**

A symbol of this class is used as a feedback pin of a function block. It can only be declared in the function block. Please refer to the following example. When the function block is called, the caller sends the value in D1 to DT_IO, which is a symbol of the INOUT class. After the operation comes to an end, the final value of DT_IO is sent to D1. Besides, in a ladder diagram, the symbols of this class are put at the left sides of function blocks.



2.4.3 Data Types

The data type of a symbol determines the format of the value of the symbol. Suppose there are two symbols VAR_1 and VAR_2. The data type of VAR_1 is BOOL, and the data type of VAR_2 is WORD. If VAR_1 and VAR_2 are used in a program, VAR_1 will represent a contact, and VAR_1 will represent a 16-bit device which can be involved in arithmetic operation or transferring the data.

The data types supported by ISPSoft are listed below.

Data type	Name	Description	Program	Function block
BOOL	Boolean	A Boolean value represents the state of a contact, could be TRUE or FALSE.	✓	✓
WORD	Word	Bit string of length 16.	✓	✓
DWORD	Double Word	Bit string of length 32.	✓	✓
LWORD	Long Word	Bit string of length 64.	✓	✓
UINT	Unsigned integer	16-bit data.	✓	✓
UDINT	Unsigned double integer	32-bit data.	✓	✓
INT	Integer	16-bit data	✓	✓
DINT	Double integer	32-bit data.	✓	✓
LINT	Long Integer	64-bit data.	✓	✓

Data type	Name	Description	Program	Function block
REAL	Real numbers.	32-bit data; applicable to single width floating-point instructions.	✓	✓
LREAL	Long reals	64-bit data; applicable to double width floating-point instructions.	✓	✓
CNT	Counter	16-bit counter value or 32-bit counter value.	✓	✓
TMR	Timer	16-bit timer value.	✓	✓
ARRAY	Array	If a symbol is declared as an array, the size of an array and an array data type must be specified. (An array is composed of 256 elements/members at most.)	✓	✓
String	Character string	Variable-length single-byte data string	✓	✓

2.4.4 Using instructions, Devices and Symbols

A device is assigned to a symbol according to the data type of the symbol. You can set the initial value of a symbol. After the program in a project is downloaded to a motion controller, the initial values of the symbols will be written into the devices assigned to the symbols if the program is scanned for the first time.

The principles of assigning devices to symbols are as follows.

- You can assign devices to the global symbols and the local symbols declared in the POUs which are programs. The system can also automatically assign devices to the global symbols and the local symbols declared in the POUs which are programs.
- If a local symbol declared in a function block is not a symbol of the VAR class, the system will automatically assign a device to the symbol, and you can not assign a device to the symbol.
- The devices assigned by the system are usable devices. (You can set a range of devices which can be assigned automatically.)
- If a symbol is declared, the device assigned to the symbol, the data type of the symbol, and the initial value of the symbol must be compatible with one another.

The relation between the data types and the device types which can be assigned is described below.

Data type	AH Motion Controller CPU	
	Device assigned by users	Device assigned by the system
BOOL	Contact M/SM or bit in the device X/Y (*3)	Contact M/SM
WORD	D	W
DWORD	D	W
LWORD	D	W

Data type	AH Motion Controller CPU	
	Device assigned by users	Device assigned by the system
UINT	D	W
UDINT	D	W
INT	D	W
DINT	D	W
LINT	D	W
REAL	D	W
LREAL	D	W
CNT	C	C
TMR	T	T
ARRAY	`The devices assigned to a symbol whose data type is ARRAY depend on the array type specified. An array is composed of the devices starting from the device assigned by users or the system, and the number of devices in an array conforms to the size of the array.	
String	N/A	

*1. Please refer to ***ISPSoft User manual*** for more information about setting a range of devices which can be assigned automatically.

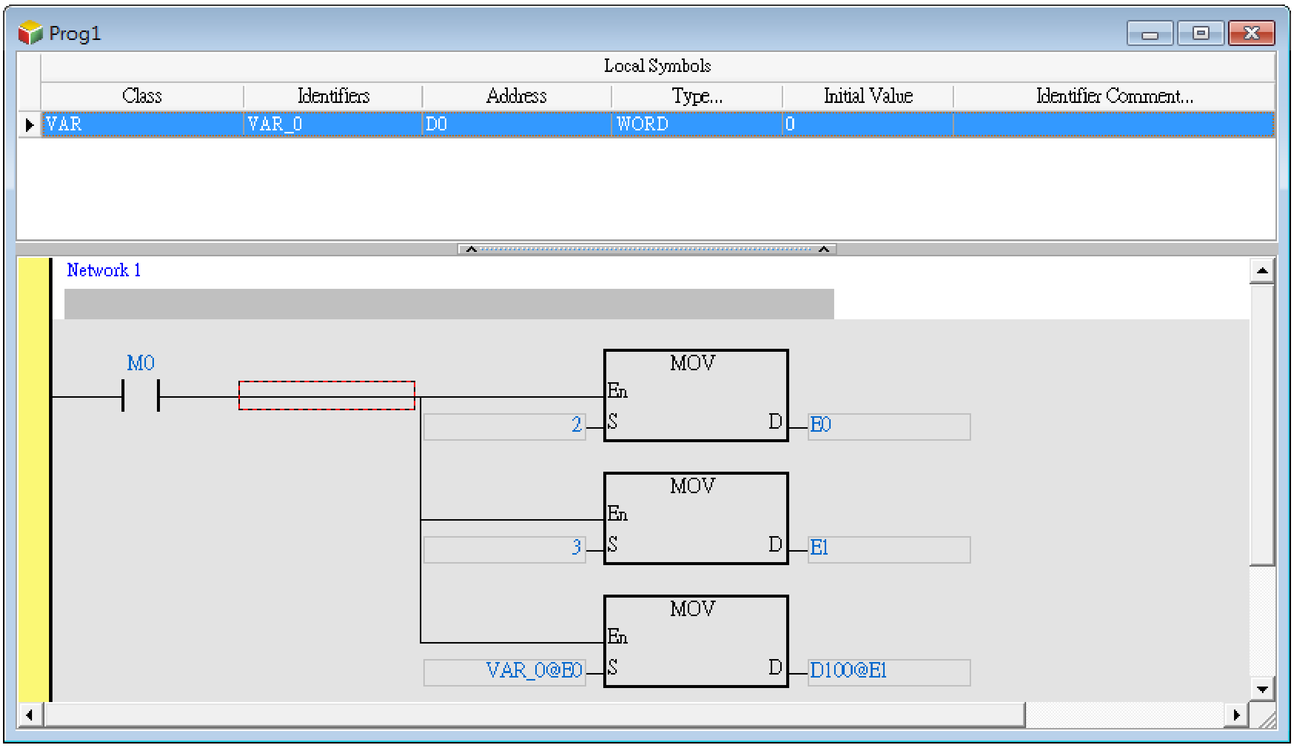
*2. A symbol representing a function block definition has a special significance. Please refer to ***ISPSoft User manual*** for more information.

*3. X0.0 and Y0.1 are bits in the word devices X and Y. Please refer to ***ISPSoft User manual*** for more information.

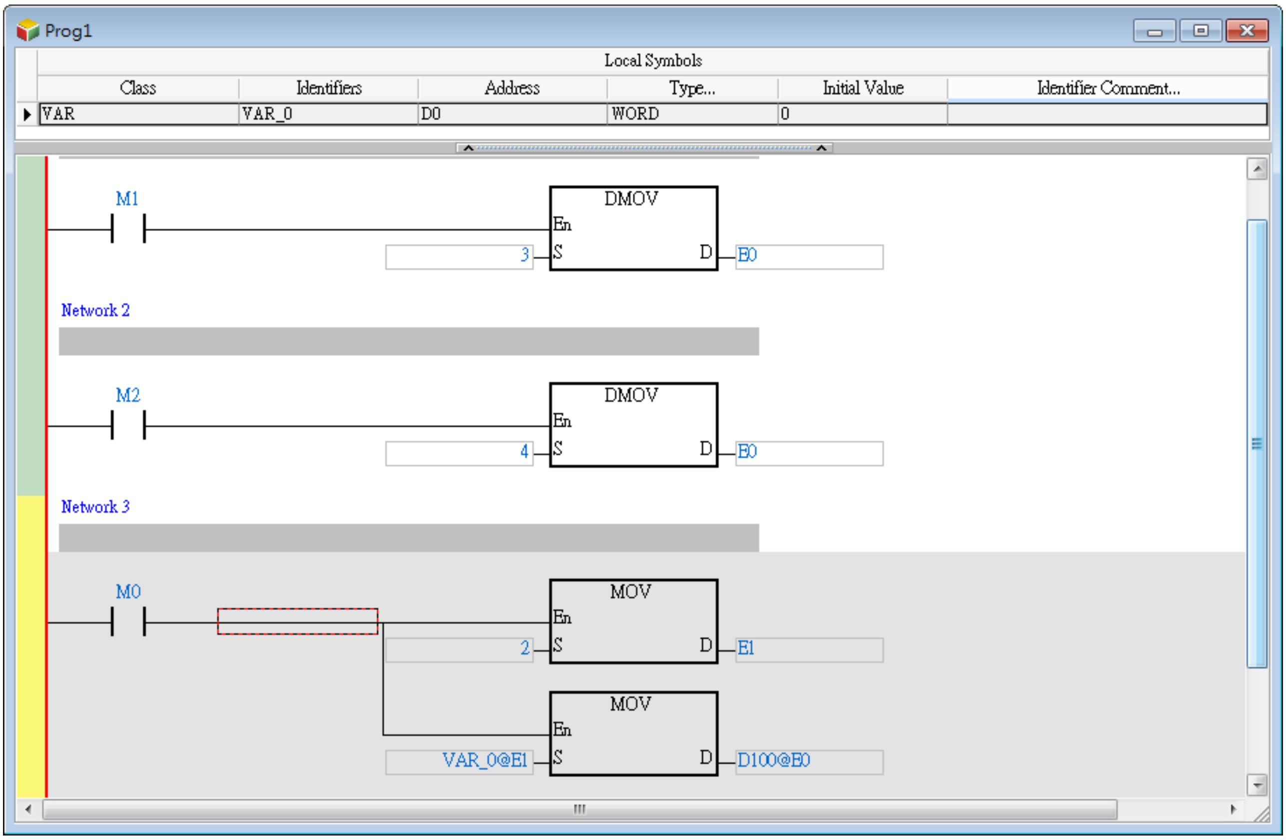
2.4.5 Modifying a Symbol with an Index Register

You are allowed to use index registers (E device) in ISPSoft to modify a symbol. The E devices are like general 16-bit data registers.You can write data into the E devices and read data from the E devices freely. If a E device is used as a general register, it can only be used in a 16-bit instruction. The modification of a symbol by an index register is represented by the format: **Identifier@Index register**. If an E device is used to modify an operand, it can be used in a 16-bit instruction or a 32-bit instruction.

Please refer to the program below. The device assigned to VAR_0 is D0. The data stored in an index register indicates the offset for the symbol which the index register modifies. If the value in the index register E0 is 2, VAR_0@ E0 indicates that 2 is added to the device address (D0) assigned to VAR_0, that is, VAR_0@E0 represents D2. If M0 is ON, the value in E0 will be 2, the value in E1 will be 3, and the value in D2 will be moved to D103.



Besides, if the value in an index register is changed, the device which actually operates differs from the original device. As a result, if the original device is not used in the program, the final value in the original device is retained. In the figure below, if the value in E0 is 3, the value in D2 will be moved to D103. When the value in E0 is changed from 3 to 4, the value in D2 will be moved to D104, and the value in D103 will remain unchanged.



*1. The data stored in an index register indicates the offset for the device which the index register modifies. If the system automatically assigns a device to a symbol, the use of an index register to modify the symbol will cause the program to be executed incorrectly because users do not know which device is assigned to the symbol.

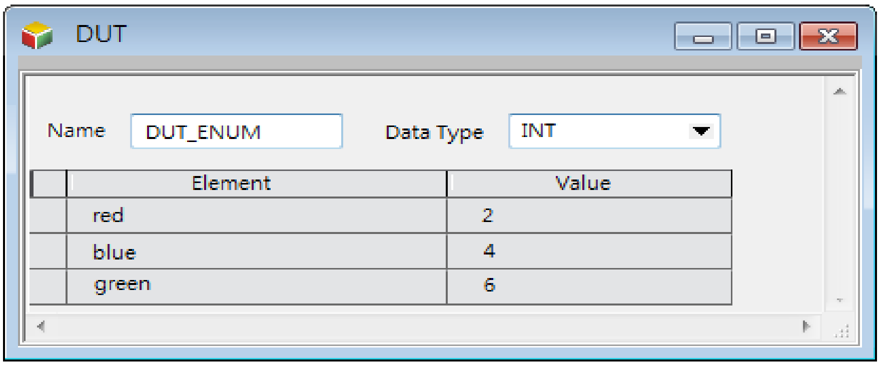
*2. If you want to assign index registers to symbols, you have to specify device addresses and data types.

2.5 Data Type Unit (DUT): ENUM

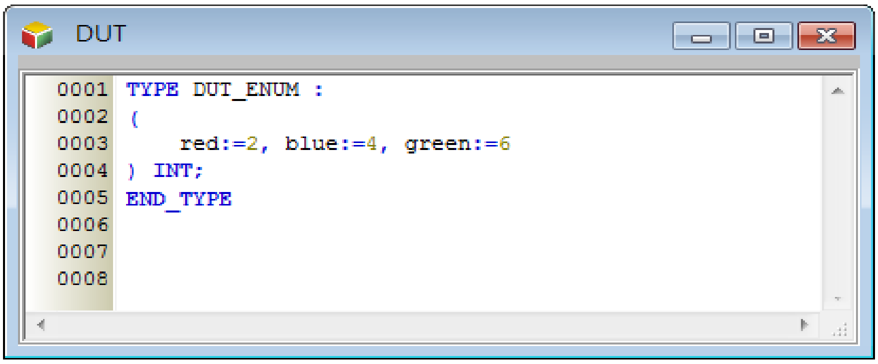
You can also use the data type unit (DUT) for enumeration (ENUM). ENUM is a derivative data type which defines the ENUM symbol with its elements and the associated values. You can specify the initial value to an element, and use one of the enumerated values in the associated elements list. The list defines an ordered set of values in series, starting with the first element and ending with the last one. You can use the same element in different ENUM symbols.

The rules for specifying the elements and the values for an ENUM symbol:

- If elements are not specified, the initial value will be 0, and following 1, and etc.
- When elements are specified with initial values, the initial value of the element will be changed. The enumerated values before the element is defined with an initial value remain unchanged. For example, 0, 1, 2 (not defined), 35(the defined initial value), 36, 37, and etc.
- Another example when the element is defined twice: 0, 1, 2 (not defined), 35(first defined value), 36, 70(second defined value), 71, and etc.
- The two setting methods:



Via table



Via texts

You can refer to **Appendices** of this manual for a list of Enumerations. For more information about the software operation, refer to ***ISPSoft User Manual***.

2.6 Instructions

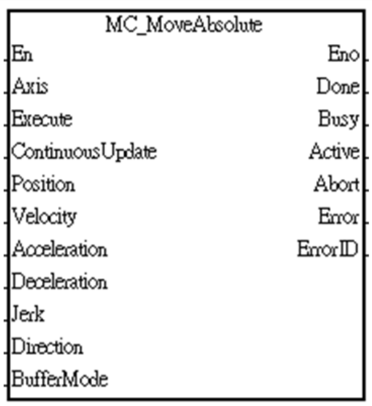
Instructions used for the AH Motion products include standard instructions and motion control instructions.
For information of standard instructions, refer to ***AH Motion Controller – Standard Instructions Manuals***.

2.6.1 Categories of Motion Control Instructions

Categories	Type	Function Group	Description
Single-axis Motion Control Instructions	Motion	Positioning on single axis	"MC_" PLCopen-based motion control instructions "DFB_" Delta-defined motion control instructions
		Velocity control on single axis	
		Torque control on single axis	
		Synchronized control on single axis	
		Manual operation on single axis	
	Administrative	Administrative functions on single axis	
Multi-axis Motion Control Instructions	Motion	G-codes	Numerical Control
		Group Motion	Coordinated Control
	Administrative	Administrative functions on multiple axes	M-codes Administrative functions on G-codes and group motion
Auxiliary Instructions	Administrative	Supporting functions for configuring the system	High speed counters, high speed timers, high speed capture and comparison and
Network instructions	Administrative	Supporting functions for setting the networks	Motion network settings

2.6.2 List of Motion Control Instructions

■ **Columns**

1	2	3	4
FB/FC	**Name**	**Graphic expression**	**Description**
FB	MC_MoveAbsolute		MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.

Items provided in the table		
1	**FB/FC**	**FB**: Function block; **FC**: Function For further explanations of Program(PROG), Function block(FB), and Function (FC), refer to ***ISPSoft User Manual.***
2	**Name**	The name of the instruction “MC_”: PLCopen-based motion control instructions “DFB_”: Delta-defined motion control instructions
3	**Graphic expression**	The graphic expression used in the ladder diagram in the software
4	**Description**	The function description of the instruction

● Single-axis Motion Control Instructions

For instruction details, refer to **3.2 PLCopen-based Motion Control Instructions** and **3.3 Delta-defined Motion Control Instructions**

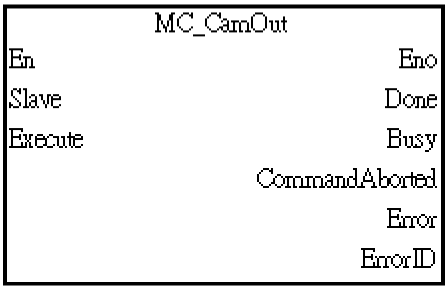
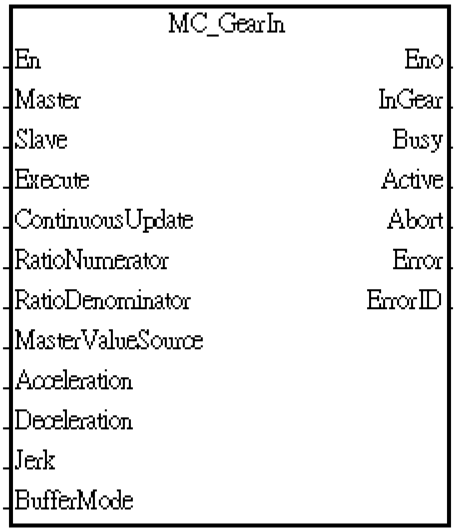
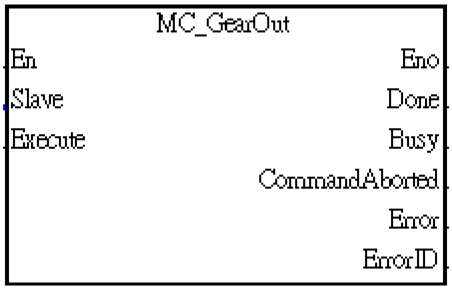
■ Motion

FB/FC	Name	Graphic expression	Description
FB	MC_Home	MC_Home En Eno Axis Done Execute Busy Position Active HomeMode CommandAborted BufferMode Error ErrorID	MC_Home controls the axis to perform the homing operation.
FB	MC_Stop	MC_Stop En Eno Axis Done Execute Busy Deceleration Active Jerk CommandAborted Error ErrorID	MC_Stop decelerates an axis to a stop.
FB	MC_Halt	MC_Halt En Eno Axis Done Execute Busy Deceleration Active Jerk CommandAborted BufferMode Error ErrorID	MC_Halt halts an axis.

FB/FC	Name	Graphic expression	Description
FB	MC_MoveAbsolute	MC_MoveAbsolute En Eno Axis Done Execute Busy ContinuousUpdate Active Position CommandAborted Velocity Error Acceleration ErrorID Deceleration Jerk Direction BufferMode	MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.
FB	MC_MoveRelative	MC_MoveRelative En Eno Axis Done Execute Busy ContinuousUpdate Active Distance CommandAborted Velocity Error Acceleration ErrorID Deceleration Jerk BufferMode	MC_MoveRelative controls the axis to move a specified relative distance with a specified behavior.
FB	MC_MoveAdditive	MC_MoveAdditive En Eno Axis Done Execute Busy ContinuousUpdate Active Distance CommandAborted Velocity Error Acceleration ErrorID Deceleration Jerk BufferMode	MC_MoveAdditive controls the axis to move an additional distance at a given speed and acceleration.

FB/FC	Name	Graphic expression	Description
FB	MC_MoveSuperImposed	MC_MoveSuperimposed En Eno Axis Done Execute Busy ContinuousUpdate Active Distance CommandAborted Velocity Error Acceleration ErrorID Deceleration CoveredDistance Jerk	MC_Superimposed controls the axis move a relative superimposed distance at a specified behavior while the axis is moving.
FB	MC_HalfSuperimposed	MC_HaltSuperimposed En Eno Axis Done Execute Busy Deceleration Active Jerk CommandAborted Error ErrorID	MC_HaltSuperimpose-d halts all superimposed motions of the axis without aborting the previous superimposed motion.
FB	MC_MoveVelocity	MC_MoveVelocity En Eno Axis InVelocity Execute Busy ContinuousUpdate Active Velocity CommandAborted Acceleration Error Deceleration ErrorID Jerk Direction BufferMode	MC_MoveVelocity performs velocity control on an axis in the position mode with a specified behavior and an average velocity.
FB	MC_VelocityControl	MC_VelocityControl En Eno Axis InVelocity Execute Busy ContinuousUpdate Active Velocity CommandAborted Acceleration Error Deceleration ErrorID Jerk Direction BufferMode	MC_VelocityControl performs velocity control on an axis in the velocity mode with a specified behavior and an average velocity.

FB/FC	Name	Graphic expression	Description
FB	MC_TorqueControl	MC_TorqueControl En Eno Axis InTorque Execute Busy ContinuousUpdate Active Torque CommandAborted TorqueRamp Error Velocity ErrorID Acceleration Deceleration Jerk Direction BufferMode	MC_TorqueControl controls the torque by using the Torque Control Mode of the applied servo drive.
FB	MC_CamIn	MC_CamIn En Eno Master InSync Slave EndOfProfile Execute Busy ContinuousUpdate Active CamTable CommandAborted Periodic Error MasterAbsolute ErrorID SlaveAbsolute MasterOffset SlaveOffset MasterScaling SlaveScaling MasterStartDistance MasterSyncPosition ActivationPosition ActivationMode StartMode Velocity Acceleration Deceleration Jerk MasterValueSource BufferMode	MC_CamIn performs cam operation by engaging the cam.

FB/FC	Name	Graphic expression	Description
FB	MC_CamOut		Cam operation is ended for the axis specified with the input parameter.
FB	MC_GearIn		MC_GearIn establishes the gear relation (velocity) between master and slave axis.
FB	MC_GearOut		MC_GearOut disconnects the gear relation (velocity) between master and slave axis.
FB	MC_PhasingAbsolute		MC_PhasingAbsolute shifts the phase of the master axis virtually by a specified absolute phase shift value.

FB/FC	Name	Graphic expression	Description
FB	MC_PhasingRelative	MC_PhasingRelative En Eno Master Done Slave Busy Execute Active PhaseShift Abort Velocity Error Acceleration ErrorID Deceleration CoveredPhaseShift Jerk BufferMode	MC_PhasingRelative shifts the phase of the master axis virtually by a specified relative phase shift value.

2

■ **Administrative**

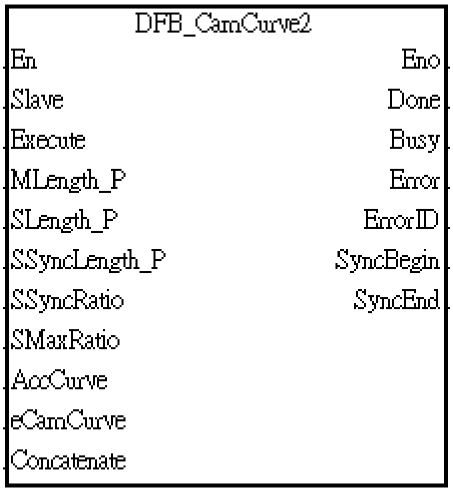
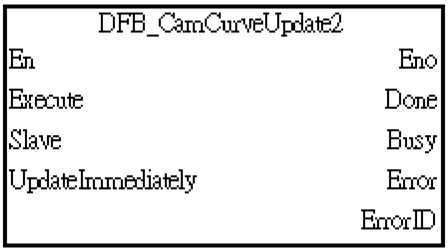
FB/FC	Name	Graphic expression	Description
FB	MC_Power	MC_Power En Eno Axis Status Enable Busy EnablePositive Active EnableNegative Error Mode ErrorID	MC_Power enables or disables the corresponding servo axis.
FB	MC_SetTorqueLimit	MC_SetTorqueLimit En Eno Axis Status Enable Busy PositiveEnable Error PositiveValue ErrorID NegativeEnable NagativeValue	MC_SetTorqueLimit instruction limits the torque output from the Servo Drive through the torque limit function of the Servo Drive.
FB	MC_SetPosition	MC_SetPosition En Eno Axis Done Execute Busy Position Error Relative ErrorID ExecutionMode	MC_SetPosition changes the current position by shifting the coordinate system of an axis.

FB/FC	Name	Graphic expression	Description
FB	MC_SetOverride	MC_SetOverride En Eno Axis Enabled Enable Busy VelFactor Error AccFactor ErrorID JerkFactor	MC_SeOverride changes the velocity override factor so as to change the target velocity of a motion axis.
FB	MC_ReadActualPosition	MC_ReadActualPosition En Eno Axis Valid Enable Busy Error ErrorID Position	This instruction reports the actual axis position continuously when *Enable* is set.
FB	MC_ReadActualVelocity	MC_ReadActualVelocity En Eno Axis Valid Enable Busy Error ErrorID Velocity	This instruction reports the actual axis velocity continuously when *Enable* is set
FB	MC_ReadActualTorque	MC_ReadActualTorque En Eno Axis Valid Enable Busy Error ErrorID Torque	This instruction reports the axis torque continuously when *Enable* is set.

FB/FC	Name	Graphic expression	Description
FB	MC_ReadStatus	MC_ReadStatus En Eno Axis Valid Enable Busy Error ErrorID ErrorStop Disabled Stopping Homing Standstill DiscreteMotion ContinousMotion SyncMotion Coordinated CoordinatedStop CoordinatedHalt	MC_ReadStatus reads the state of the axis and indicates it at the outputs.
FB	MC_ReadMotionState	MC_ReadMotionState En Eno Axis Valid Enable Busy Source Error ErrorID ConstantVelocity Accelerating Decelerating DirectionPositive DirectionNegative	This instruction reports details of the axis status relating the on-going motion behavior
FB	MC_ReadAxisError	MC_ReadAxisError En Eno Axis Valid Enable Busy Error ErrorID AxisErrorID	MC_ReadStatus reads the state of the axis and indicates it at the outputs.
FB	MC_Reset	MC_Reset En Eno Axis Done Execute Busy Error ErrorID	MC_Reset clears axis-related errors

FB/FC	Name	Graphic expression	Description
FB	MC_TouchProbe	MC_TouchProbe En — Eno Axis — Done TriggerInput — Busy Execute — CommandAborted WindowOnly — Error FirstPosition — ErrorID LastPosition — RecordedPosition	MC_TouchProbe captures and records an axis position when a trigger event occurs.
FB	MC_AbortTrigger	MC_AbortTrigger En — Eno Axis — Done TriggerInput — Busy Execute — Error ErrorID	MC_AbortTrigger aborts instructions which are intended to capture trigger events (e.g. MC_TouchProbe).
FB	DFB_AxisSetting1	DFB_AxisSetting1 En — Eno Axis — Done Execute — Busy Vmax — Error Vbias — ErrorID Tacc Tdec	DFB_AxisSetting1 sets motion parameters for the specified axis.
FB	DFB_AxisSetting2	DFB_AxisSetting2 En — Eno Axis — Done Execute — Busy Vcurve — Error PulseRev — ErrorID DistanceRev	DFB_AxisSetting2 sets motion parameters for the specified axis.

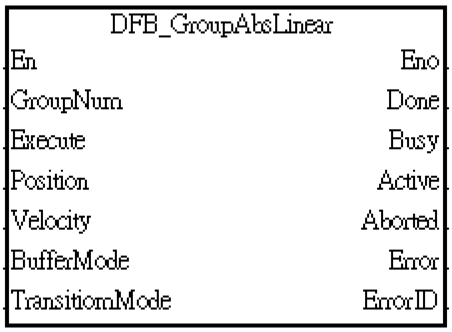
FB/FC	Name	Graphic expression	Description
FB	DFB_InputPolarity	DFB_InputPolarity En — Eno Enable — Valid X0_00_Pg0 — Pg0_X0_00 X0_01_Pg1 — Pg1_X0_01 X0_02_Pg2 — Pg2_X0_02 X0_03_Pg3 — Pg3_X0_03 X0_08_mpgA — mpgA_X0_08 X0_09_mpgB — mpgB_X0_09 X0_10_Dog4 — Dog4_X0_10 X0_11_Dog5 — Dog5_X0_11 X0_12_Dog0 — Dog0_X0_12 X0_13_Dog1 — Dog1_X0_13 X0_14_Dog2 — Dog2_X0_14 X0_15_Dog3 — Dog3_X0_15 X1_00 — Nor_X1_00 X1_01 — Nor_X1_01 X1_02 — Nor_X1_02 X1_03 — Nor_X1_03 X1_04 — Nor_X1_04 X1_05 — Nor_X1_05 Busy	DFB_InputPolarity sets the polarity of inputs and reads the states of these input terminals.
FB	DFB_CamMultiRead	DFB_CamMultiRead En — Eno CamTableId — Valid Enable — Error ReadStartPointNo — ErrorID ReadAmount — MasterPosition SlavePosition	DFB_CamMultiRead reads cam points from the specified motion axis.
FB	DFB_CamMultiWrite	DFB_CamMultiWrite En — Eno CamTableId — Done Execute — Busy WriteStartPointNo — Error WriteAmount — ErrorID MasterPosition SlavePosition	DFB_CamMultiWrite writes cam poinst to the specified cam curve

FB/FC	Name	Graphic expression	Description
FB	DFB_CamCurve2		DFB_CamCurve2 creates cam curves which are mainly used in rotary cut and flying saw applications.
FB	DFB_CamCurveUpdate2		DFB_CamCurveUpdate2 updates the cam operation with the modified cam profile in the next cycle.

- ## Multi-axis Motion Control Instructions

For instruction details, refer to **3.3 Delta-defined Motion Control Instructions**

■ Motion

FB/FC	Name	Graphic expression	Description
FB	DFB_GroupGcodeRun		DFB_GroupGcodeRun performs G-code motion on the axis group specified by *GcoupNum*.
FB	DFB_GroupAbsLinear		DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified absolute target position.

FB/FC	Name	Graphic expression	Descriptio n
FB	DFB_GroupRelLinear	DFB_GroupRelLinear En Eno GroupNum Done Execute Busy Distance Active Velocity Aborted BufferMode Error TransitiomMode ErrorID	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified relative distance.
FB	DFB_GroupAbsCircular	DFB_GroupAbsCircular En Eno GroupNum Done Execute Busy DirectionCCW Active IPMode Aborted Position Error AuxPosition ErrorID Velocity SpiralTurns BufferMode TransitiomMode	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified relative distance.
FB	DFB_GroupRelCircular	DFB_GroupRelCircular En Eno GroupNum Done Execute Busy DirectionCCW Active IPMode Aborted Position Error AuxPosition ErrorID Velocity SpiralTurns BufferMode TransitiomMode	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move a specified relative distance.
FB	DFB_GroupStop	DFB_GroupStop En Eno GroupNum Done Execute Busy StopMode Error ErrorID	DFB_GroupStop decelerates the group axes to stop or pause to the current position.

■ Administrative

FB/FC	Name	Graphic expression	Description
FB	DFB_GroupMcode	DFB_GroupMcode En Eno GroupNum Valid Enable Busy CLRMcode Error ErrorID Value	In the G-code motion of the specified axis group, DFB_GroupMcode reads the M-code in use.
FB	DFB_GroupGcodeSetting	DFB_GroupGcodeSetting En Eno GroupNum Done Execute Busy VelPercentage Error ErrorID	DFB_GroupGcodeSetting specifies the behavior for continuous interpolation in G-code motion by the velocity percentage setting (*VelPercentage*)
FB	DFB_GroupEnable	DFB_GroupEnable En Eno GroupNum Done Execute Busy AxisNumOrder_1 Error AxisNumOrder_2 ErrorID AxisNumOrder_3 AxisNumOrder_4 AxisNumOrder_5 AxisNumOrder_6	DFB_GroupEnable enables a group of axes for group motion.
FB	DFB_GroupDisable	DFB_GroupDisable En Eno GroupNum Done Execute Busy Error ErrorID	DFB_GroupDisable disables the axis group with the specidied group number.
FB	DFB_GroupReset	DFB_GroupReset En Eno GroupNum Done Execute Busy Error ErrorID	DFB_GroupReset resets the axis group which is in the state of “Errorstop”.

FB/FC	Name	Graphic expression	Description
FB	DFB_ReadGroupStatus	DFB_ReadGroupStatus En Eno GroupNum Valid Enable Error ErrorID AxisNumOrder_1 AxisNumOrder_2 AxisNumOrder_3 AxisNumOrder_4 AxisNumOrder_5 AxisNumOrder_6 GroupStatus	DFB_ReadGroupStatus reads the axis numbers in an axis group, and indicates the status of the axis group at *GroupStatus*.

Auxiliary instructions

For instruction details, refer to **3.3 Delta-defined Motion Control Instructions**

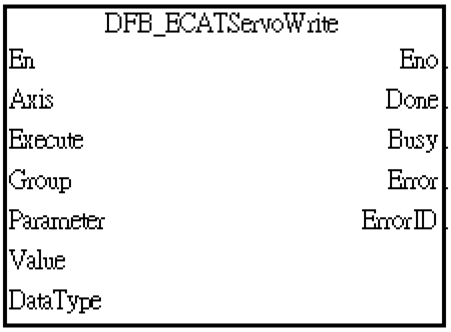
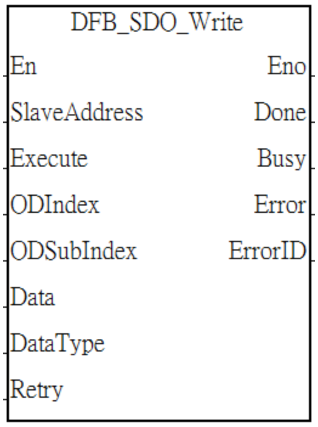
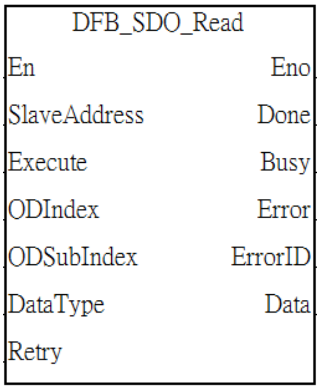
FB/FC	Name	Graphic expression	Description
FB	DFB_HCnt	DFB_HCnt En Eno Channel Valid Enable Busy ExtRstEN Error InputType ErrorID InitialValue CountValue	DFB_HCnt enables the specified high speed counter according to the specified parameters and monitors the count value.
FB	DFB_HTmr	DFB_HTmr En Eno Channel Valid Enable Busy TriggerMode Error ErrorID TimerValue	DFB_HCnt enables the specified high speed counter according to the specified parameters and monitors the count value.
FB	DFB_Compare	DFB_Compare En Eno Channel Valid Enable Busy Source Error CmpMode ErrorID OutputDevice OutputMode CmpValue	DFB_Compare compares the designated source with a specified value and outputs the specified results on a desired device when the comparison result is True.

FB/FC	Name	Graphic expression	Description
FB	DFB_CmpOutRst	DFB_CmpOutRst En / Eno Enable / Valid CLR_Y08 / CMP_Y08 CLR_Y09 / CMP_Y09 CLR_Y10 / CMP_Y10 CLR_Y11 / CMP_Y11 CLR_AC0Rst / CMP_AC0Rst CLR_AC4Rst / CMP_AC4Rst CLR_AC8Rst / CMP_AC8Rst CLR_AC12Rst / CMP_AC12Rst Busy	DFB_CmpOutRst monitors the output results and clears the output states triggered by the comparators.
FB	DFB_Capture	DFB_Compare En / Eno Channel / Valid Enable / Busy Source / Error CmpMode / ErrorID OutputDevice OutputMode CmpValue	DFB_Capture base on user selected trigger device to capture the command pulse of the user assign axis.

Network Instructions

For instruction details, refer to **3.3 Delta-defined Motion Control Instructions**

FB/FC	Name	Graphic expression	Description
FB	DFB_ECATReset	DFB_ECATReset En / Eno Execute / Done Busy Error ErrorID	DFB_ECATReset resets an abnormal EtherCAT network.
FB	DFB_ECATServoRead	DFB_ECATServoRead En / Eno Axis / Done Execute / Busy Group / Error Parameter / ErrorID Retry / Value	DFB_ECATServoRead reads the values of parameters from the Delta servo drive specified on an EtherCAT network.

FB/FC	Name	Graphic expression	Description
FB	DFB_ECATServoWrite		DFB_ECATServoWrite writes the values of parameters into the Delta servo drive specified on an EtherCAT network.
FB	DFB_SDO_Write		DFB_SDO_Write writes the values of parameters into the specified OD of the EtherCAT Slave via SDO.
FB	DFB_SDO_Read		DFB_SDO_Read reads the values of parameters from the specified OD of the EtherCAT Slave via SDO.

MEMO

3

Chapter 3 Motion Control Instructions

Table of Contents

3.1 Applying This Chapter

The interface of a motion control instruction includes inputs, outputs and in-outs. The definitions and the general behaviors of the common interface are explained in this section.

3.1.1 The Interface of a Motion Control Function Block

- **Definitions of Inputs and Outputs**

Common inputs and outputs in motion control function blocks are listed below. A function block has one or a part of the inputs/outputs listed below. For example, a function block has either the *Execute* input or the *Enable* input, based on the properties of the motion control function block.

Inputs			
Name	Description	Date Type	Setting value (Default)
En	Receiving the logic state in front of this instruction.	BOOL	True/False (False)
Enable	Enabling the motion control function block	BOOL	True/False (False)
Execute	Starting the motion control function block	BOOL	True/False (False)
Outputs			
Name	Description	Date Type	Setting value(Default)
Eno	Transfering the logic state of the *En* input to the following instruction which is connected in series	BOOL	True/False (False)
Done	The execution of the function block is complete.	BOOL	True/False (False)
Valid	An output value is valid.	BOOL	True/False (False)
Busy	The motion control function block is being executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
Command Aborted(Aborted)	The execution of the motion control function block is interrupted by another instruction.	BOOL	True/False (False)
Error	An error occurs in a function block.	BOOL	True/False (False)

A motion control function block has either the *Execute* input or the *Enable* input. The *Execute*/*Enable* input in a motion control function block is used to start the motion control function block. A motion control function block generally has the *Busy* output and the *Done* output. The *Busy* output and the *Done* output indicate the state of the motion control function block. If the execution of motion control function block is to be interrupted by another motion control function block, the *CommandAborted/Aborted* output will be added to the motion control function block. Besides, the *Error* output is used to indicate that an error occurs when the motion control function block is executed.

A motion control function block has not only the *Execute*/*Enable* input, but also value/state inputs. The characteristics of the value/state inputs are described below.

- **Use of input values:**
  - If a function block has an *Execute* input, values are used when *Execute* input changes from False to True. If a new value is created, it becomes valid when the *Execute* input is triggered again while the instruction is not in busy status.
  - If a function block has an *Enable* input, values are used when *Enable* input changes from False to True. Compared with the *Execute* input, the *Enable* input is used more often when an input value is updated repeatedly.
- **An input value exceeds a range.**

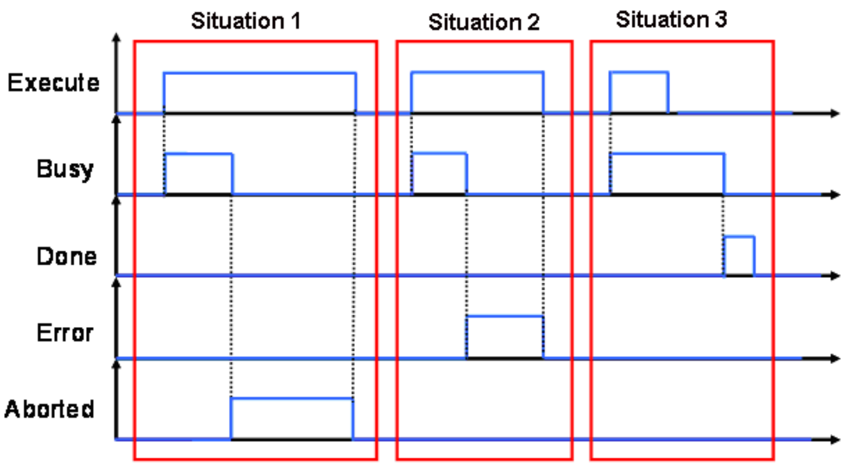
  After a function block is started, the input values which are not in ranges allowed will be limited, or result in an error occurring in the motion control function block. If an error occurring in a function block results in an error occurring in an axis, the function block is applied incorrectly. Users should prevent incorrect values from being generated in an applied program.
- **Outputs are mutually exclusive**.
  - If the a function block has an *Execute* input which is set to True, either the *Busy* output, the *Done* output, the *CommandAborted* output, or the *Error* output can be set to True.
  - If the a function block has an *Enable* input, the *Valid* output and the *Error* output are mutually exclusive, and only the *Valid* output or the *Error* output can be set to True.
- **Time when output data/states are valid**
  - If a function block has an *Execute* input, the *Done* output, the *Error* output, the *CommandAborted* output, and data output are reset when the *Execute* input changes from False to True. If the *Execute* input is reset before the execution of the function block is complete, i.e. *Busy* is True, output states of *Done*, *Error*, *CommandAborted* will still be generated and retained for one cycle.
  - If a function block has an *Enable* input, the *Valid* output, the *Busy* output, and the *Error* output are reset when *Enable* input changes from True to False. (MC_Power has different behaviors on the inputs and outputs, refer to MC_Power for details.)
- **Characteristic of the *Done* output**

  The *Done* output in a motion control function block will be set to True after the motion control function block is executed successfully.
- **Characteristic of the *Busy* output**
  - If a function block has an *Execute* input, the function block uses the *Busy* output to indicate that the execution of the function block is not complete, and new output states (values) are expected to be generated. The *Busy* output is set to True when *Execute* input changes from False to True. When either the *Done* output, the *CommandAborted* output, or the *Error* output is set to True, the *Busy* output is reset.
  - If a function block has an *Enable* input, the function block uses the *Busy* output to indicate that the execution of the function block is not complete, and new output states (values) are expected to be generated. The *Busy* output is set to True when *Enable* input changes from False to True, and remains True when the function block is executing. When the *Busy* output is True, output states (values) are updated continuously.
- **Characteristic of the *CommandAborted/Aborted* output**

  The *CommandAborted* output in a motion control function block is set to True when the execution of the motion control function block is interrupted.
- **Relation between the *Enable* input and the *Valid* output**

  If a function block has an *Enable* input, the function block uses the *Valid* output to indicate whether output data/states are valid. The *Valid* output is set to True only when the *Enable* input is set to True and output data/state becomes valid and available. If an error occurs in a function block, output data/states will not be valid, and the *Valid* output will be set to False. The *Valid* output in a motion control function block will not be True until the error is eliminated and the output data/states become valid.

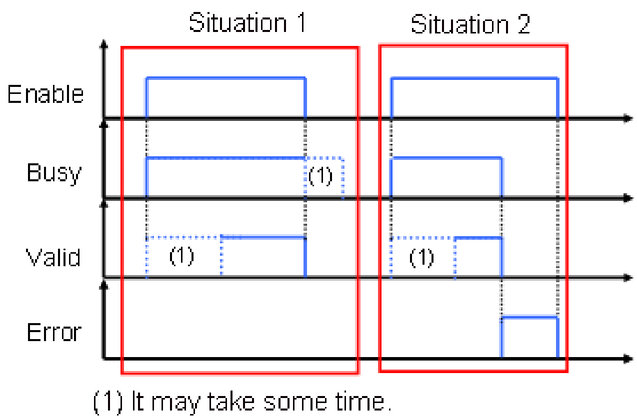
Timing Diagram for Input/Outputs



Situation 1: The execution of the motion control function block is interrupted.

Situation 2: An error occurs in the motion control function block.

Situation 3: The execution of the motion control function block is complete normally.



Situation 1: The motion control function block is executed normally.

Situation 2: An error occurs in the motion control function block.

3.1.2 PDO Mapping

Before starting to use the motion control instructions, you must complete the PDO (Process Data Objects) mapping settings for processing data communications between the ISPSoft software and the AH Motion CPU.

Setting values for PDO mapping:

RxPDO (1600 hex)	Control Word(6040 hex), TargetPosition(607A hex), TargetVelocity(60FF hex), TargetTorque (6071 hex), ModeOfOperation(6060 hex)
TxPDO (1a00 hex)	Status Word(6041 hex), ActualPosition(6064 hex), Velocity actual value(606C hex), ActualTorque (6077 hex), ModeOfOperationDisplay(6061 hex)

For the detailed explanations on PDO setting, refer to the ***corresponding Slave's Manual.***

For the step-by-step PDO setting procedures, refer to ***ISPSoft User Manual.***

3.1.3 List of Motion Control Related Instructions (Sort by function)

	Categories	Name	Description
Single-axis motion control Instructions	Position Control	MC_Home	MC_Home drives the axis to perform the homing operation.
		MC_Stop	MC_Stop decelerates an axis to a stop.
		MC_Halt	MC_Halt halts an axis.
		MC_MoveAbsolute	MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.
		MC_MoveRelative	MC_MoveRelative controls the axis to move a specified relative distance with a specified behavior.
		MC_MoveAdditive	MC_MoveAdditive controls the axis to move an additional distance at a given speed and acceleration.
		MC_MoveSuperImposed	MC_Superimposed controls the axis move a relative superimposed distance at a specified behavior while the axis is moving.
		MC_HalfSuperimposed	MC_HaltSuperimposed halts all superimposed motions of the axis without aborting the previous superimposed motion.
	Velocity Control	MC_MoveVelocity	MC_MoveVelocity performs velocity control on an axis in the position mode with a specified behavior and an average velocity.
		MC_VelocityControl	MC_VelocityControl performs velocity control on an axis in the velocity mode with a specified behavior and an average velocity.

	Categories	Name	Description
	Torque Control	MC_TorqueControl	MC_TorqueControl controls the torque by using the Torque Control Mode of the applied servo drive.
	Synchronized control	MC_CamIn	MC_CamIn performs cam operation by engaging the cam.
		MC_CamOut	Cam operation is ended for the axis specified with the input parameter.
		MC_GearIn	MC_GearIn establishes the gear relation (velocity) between master and slave axis.
		MC_GearOut	MC_GearOut disconnects the gear relation (velocity) between master and slave axis.
		MC_GearInPos	MC_GearInPos establishes the gear relation between master and slave axis with the specified starting synchronization position.
		MC_PhasingAbsolute	MC_PhasingAbsolute shifts the phase of the master axis virtually by a specified absolute phase shift value.
		MC_PhasingRelative	MC_PhasingRelative shifts the phase of the master axis virtually by a specified relative phase shift value.
		MC_CombineAxis	MC_GombineAxes combines the motion of 2 axes by summing or deducting the command positions of the two axes.
	Manual Operation	DFB_MPG	DFB_MPG enables the manual pulse generator (MPG) mode.
	Single axis administrative	MC_Power	MC_Power enables or disables the corresponding servo axis.
		MC_SetTorqueLimit	MC_SetTorqueLimit instruction limits the torque output from the Servo Drive through the torque limit function of the Servo Drive.
		MC_SetPosition	MC_SetPosition changes the current position by shifting the coordinate system of an axis.
		MC_SetOverride	MC_SeOverride changes the velocity override factor so as to change the target velocity of a motion axis.
		MC_ReadActualPosition	This instruction reports the actual axis position continuously when *Enable* is set.
		MC_ReadActualVelocity	This instruction reports the actual axis velocity continuously when *Enable* is set
		MC_ReadActualTorque	This instruction reports the axis torque continuously when *Enable* is set.

	Categories	Name	Description
		MC_ReadStatus	MC_ReadStatus reads the state of the axis and indicates it at the outputs.
		MC_ReadMotionState	This instruction reports details of the axis status relating the on-going motion behavior
		MC_ReadAxisError	MC_ReadStatus reads the state of the axis and indicates it at the outputs.
		MC_Reset	MC_Reset clears axis-related errors
		MC_TouchProbe	MC_TouchProbe captures and records an axis position when a trigger event occurs.
		MC_AbortTrigger	MC_AbortTrigger aborts MC_TouchProbe instructions which are intended to capture trigger events.
		DFB_AxisSetting1	DFB_AxisSetting1 sets motion parameters for the specified axis.
		DFB_AxisSetting2	DFB_AxisSetting2 sets motion parameters for the specified axis.
		DFB_InputPolarity	DFB_InputPolarity sets the polarity of inputs and reads the states of these input terminals.
		DFB_CamMultiRead	DFB_CamRead reads cam points from the specified motion axis.
		DFB_CamMultiWrite	DFB_CamWrite writes cam points to the specified cam curve
		DFB_CamCurve2	DFB_CamCurve2 creates cam curves which are mainly used in rotary cut and flying saw applications.
		DFB_CamCurveUpdate2	DFB_CamCurveUpdate2 updates the cam operation with the modified cam profile in the next cycle.
	Group Motion	DFB_GroupAbsLinear	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified absolute target position.
		DFB_GroupRelLinear	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified relative distance.
		DFB_GroupAbsCircular	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move to the specified absolute target position.
		DFB_GroupRelCircular	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move a specified relative distance.

	Categories	Name	Description
	Stop	DFB_GroupStop	DFB_GroupStop decelerates the group axes to stop or pause to the current position.
	Multi-axes administrative	DFB_GroupEnable	DFB_GroupEnable enables a group of axes for group motion.
		DFB_GroupDisable	DFB_GroupDisable disables the axis group with the specidied group number.
		DFB_GroupReset	DFB_GroupReset resets the axis group which is in the state of “Errorstop”.
		DFB_ReadGroupStatus	DFB_ReadGroupStatus reads the axis numbers in an axis group, and indicates the status of the axis group at *GroupStatus*.
Auxiliary	High speed counter	DFB_HCnt	DFB_HCnt enables the specified high speed counter according to the specified parameters and monitors the count value.
	High speed timer	DFB_HTmr	DFB_HTmr enables the specified high speed timer channel according to the specified parameters and monitors and timed value.
	Comparison	DFB_Compare	DFB_Compare compares the designated source with a specified value and outputs the specified results on a desired device when the comparison result is True.
		DFB_CmpOutRst	DFB_CmpOutRst monitors the output results and clears the output states triggered by the comparators.
	Capture	DFB_Capture2	DFB_Capture2 captures the commanded pulses of the specified axis according to the designated external trigger deivce.
Network	ECAT Communication	DFB_ECATReset	DFB_ECATReset resets an abnormal EtherCAT network.
		DFB_ECATServoRead	DFB_ECATServoRead reads the values of parameters from the Delta servo drive specified on an EtherCAT network.
		DFB_ECATServoWrite	DFB_ECATServoWrite writes the values of parameters into the Delta servo drive specified on an EtherCAT network.
		DFB_SDO_Read	DFB_SDO_Read reads the values of parameters from the specified OD of the EtherCAT Slave via SDO.
		DFB_SDO_Write	DFB_SDO_Write writes the values of parameters into the specified OD of the EtherCAT Slave via SDO.

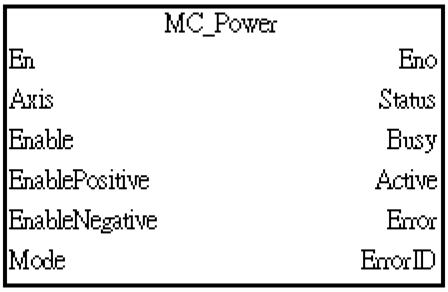
3.2 PLCopen-based Motion Control Instructions

Categories		Name	Description
Single-axis motion control Instructions	Position Control	MC_Home	MC_Home drives the axis to perform the homing operation.
		MC_Stop	MC_Stop decelerates an axis to a stop.
		MC_Halt	MC_Halt halts an axis.
		MC_MoveAbsolute	MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.
		MC_MoveRelative	MC_MoveRelative controls the axis to move a specified relative distance with a specified behavior.
		MC_MoveAdditive	MC_MoveAdditive controls the axis to move an additional distance at a given speed and acceleration.
		MC_MoveSuperImposed	MC_MoveSuperimposed controls the axis move a relative superimposed distance at a specified behavior while the axis is moving.
		MC_HalfSuperimposed	MC_HaltSuperimposed halts all superimposed motions of the axis without aborting the previous superimposed motion.
	Velocity Control	MC_MoveVelocity	MC_MoveVelocity performs velocity control on an axis in the position mode with a specified behavior and an average velocity.
		MC_VelocityControl	MC_VelocityControl performs velocity control on an axis in the velocity mode with a specified behavior and an average velocity.
	Torque Control	MC_TorqueControl	MC_TorqueControl controls the torque by using the Torque Control Mode of the applied servo drive.
	Synchronized cont**rol**	MC_CamIn	MC_CamIn performs cam operation by engaging the cam.
		MC_CamOut	Cam operation is ended for the axis specified with the input parameter.
		MC_GearIn	MC_GearIn establishes the gear relation (velocity) between master and slave axis.
		MC_GearOut	MC_GearOut disconnects the gear relation (velocity) between master and slave axis.
		MC_PhasingAbsolute	MC_PhasingAbsolute shifts the phase of the master axis virtually by a specified absolute phase shift value.
		MC_PhasingRelative	MC_PhasingRelative shifts the phase of the master axis virtually by a specified relative phase shift value.
	Administrative	MC_Power	MC_Power enables or disables the corresponding servo axis.

Categories		Name	Description
		MC_SetTorqueLimit	MC_SetTorqueLimit instruction limits the torque output from the Servo Drive through the torque limit function of the Servo Drive.
		MC_SetPosition	MC_SetPosition changes the current position by shifting the coordinate system of an axis.
		MC_SetOverride	MC_SeOverride changes the velocity override factor so as to change the target velocity of a motion axis.
		MC_ReadActualPosition	This instruction reports the actual axis position continuously when *Enable* is set.
		MC_ReadActualVelocity	This instruction reports the actual axis velocity continuously when *Enable* is set
		MC_ReadActualTorque	This instruction reports the axis torque continuously when *Enable* is set.
		MC_ReadStatus	MC_ReadStatus reads the state of the axis and indicates it at the outputs.
		MC_ReadMotionState	This instruction reports details of the axis status relating the on-going motion behavior
		MC_ReadAxisError	MC_ReadStatus reads the state of the axis and indicates it at the outputs.
		MC_Reset	MC_Reset clears axis-related errors
		MC_TouchProbe	MC_TouchProbe captures and records an axis position when a trigger event occurs.
		MC_AbortTrigger	MC_AbortTrigger aborts MC_TouchProbe instructions which are intended to capture trigger events.

MC_Power

FB/FC	Description
FB	MC_Power enables or disables the corresponding servo axis.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	The axis is ready to be operated when *Enable* is True, and not ready when *Enable* is False.	BOOL	True/False (False)	-
EnablePositive	Enables motion in positive direction when *EnablePositive* is True. Valid only when *Enable* is True.	BOOL	True/False (False)	Continuously updates value when *Enable* is True.
EnableNegative	Enables motion in negative direction when *EnableNegative* is True. Valid only when *Enable* is True.	BOOL	True/False (False)	Continuously updates value when *Enable* is True.
Mode	Specifies the buffering behavior of the instruction when *Enable* changes to False.	eMC_SERVOOFF_MODE*2	0: mcAborting 1: mcBuffered (0)	When *Enable* shifts to False

***Note**:

1. Motion control instructions can control the axis to perform the corresponding motion only after *Enable of* MC_Power changes to True.
2. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

● Outputs

Name	Function	Data type	Output range (Default value)
Status	True when the axis is ready to be operated.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)

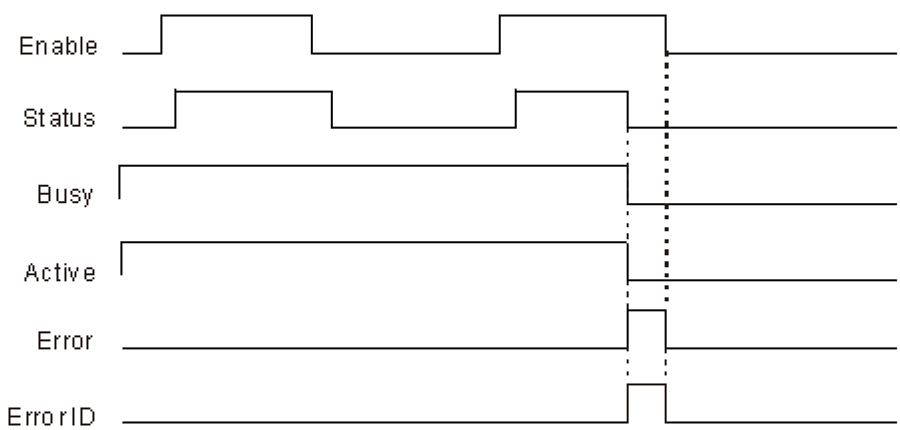
Name	Function	Data type	Output range (Default value)
Active	True when the axis is being controlled.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Status	● When *Enable* shifts to True and the axis is ready to be operated.	● When *Enable* shifts to False. ● When *Error* shifts to True.
Busy*	● When the function block instance is enabled (*En*= True) and no error exists.	● When *Error* shifts to True.
Active*	● When the function block instance is enabled (*En*= True) and no error exists.	● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When the error is cleared.

***Note**: *Busy* and *Active* will not shift to False when *Enable* is reset. The two outputs will be reset only when an error occurs.

■ **Timing Diagram**



● **In-Outs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- The axis specified by *Axis* is ready to be operated and controlled when *Enable* changes to True.
- The effects of using *Enable* with *EnablePositve and EnableNegative* are as below

Enable	*EnablePositive*	*EnableNegative*	Effects
True	True	False	You can move the specified axis in the positive direction.
True	False	True	You can move the specified axis in the negative direction.

- If an associated motion instruction for moving the axis in **positive** direction is to be executed:

Enable	*EnablePositive*	*EnableNegative*	Effects
True	**False**	True/False	1. Cannot move the axis in positive direction. 2. If the axis is moving in positive direction and *EnablePositvie* is False, the associated motion instruction will have an error, and the axis will stop and enter the “Errorstop” state.

- If an associated motion instruction for moving the axis in **negative** direction is to be executed:

Enable	*EnablePositive*	*EnableNegative*	Effects
True	True/False	**False**	1. Cannot move the axis in negative direction. 2. If the axis is moving in positive direction and *EnableNegasitvie* is False, the associated motion instruction will have an error, and the axis will stop and enter the “Errorstop” state.

- If *Enable* changes to False for the axis in the “Standstill” state, the ready status of the specified axis will be cleared. In this case, you cannot control the axis. Also, an error occurs if a motion instruction is executed on an axis whose ready status is cleared. However, you can execute the MC_Power (to power Servo) and MC_Reset (to reset axis errors) instructions on axes that are not in ready status.
- When *Enable* changes to False for the axis which is moving by motion instructions, whether the axis will enter the “Disabled” state immediately or not depends on the setting of ***Mode***.

Mode

Mode determines the behavior to combine the axis motions for this instruction and the previous instruction.
When the instruction is executed;

- The selected mode is valid if the previous instruction is executing.
- The selected mode is invalid if the axis is in “Standstill” state.

The following table lists the available buffer mode settings of MC_Power.

Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

Note: *mode* is evaluated when *Enable* of MC_Power is reset.

Input	Mode	Description
Mode	0: mcAborting	When *Enable* changes to False, the axis will enter the “Disabled” state immediately, and the motion controller will stop the slave axis from moving (within one sync cycle). Be cautious during operation in case of any danger!
	1: mcBuffered	When *Enable* changes to False, the axis will enter the “Disabled” state only after it changes to the “Standstill” state. The axis will complete its motion before entering “Standstill” state.

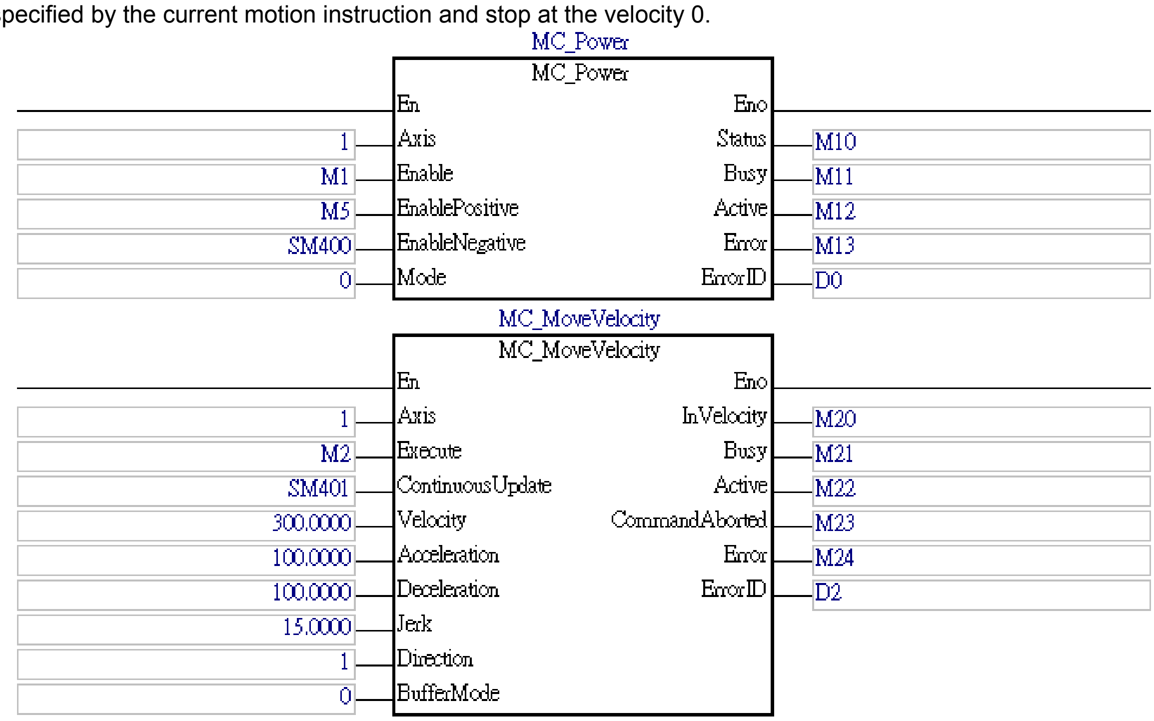
● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

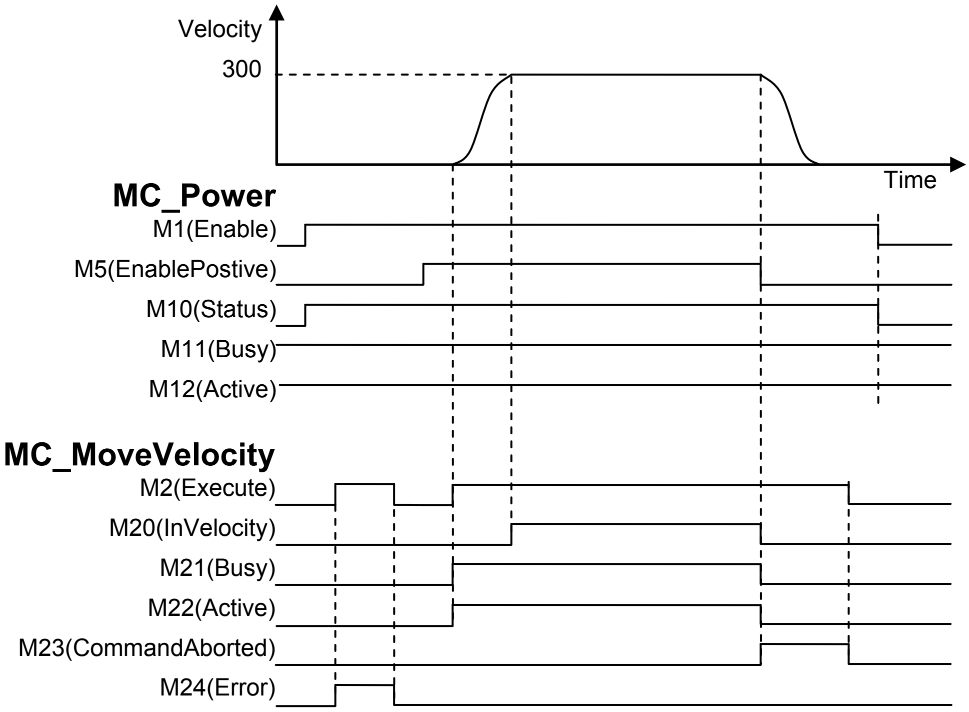
● Programming Example 1

The example below describes the behavior of the *EnablePositive* of MC_Power in combination with MC_MoveVelocity.

When *Enable* is True and *EnablePositive* is False, motion instructions are not allowed to move the specified in positive direction. In this case, there will be an error if the instruction for moving the axis forward is executed. When the axis moves in positive direction and *EnablePositive* changes from True to False, the axis will decelerate its speed at the decelerate rate specified by the current motion instruction and stop at the velocity 0.



Motion diagram:

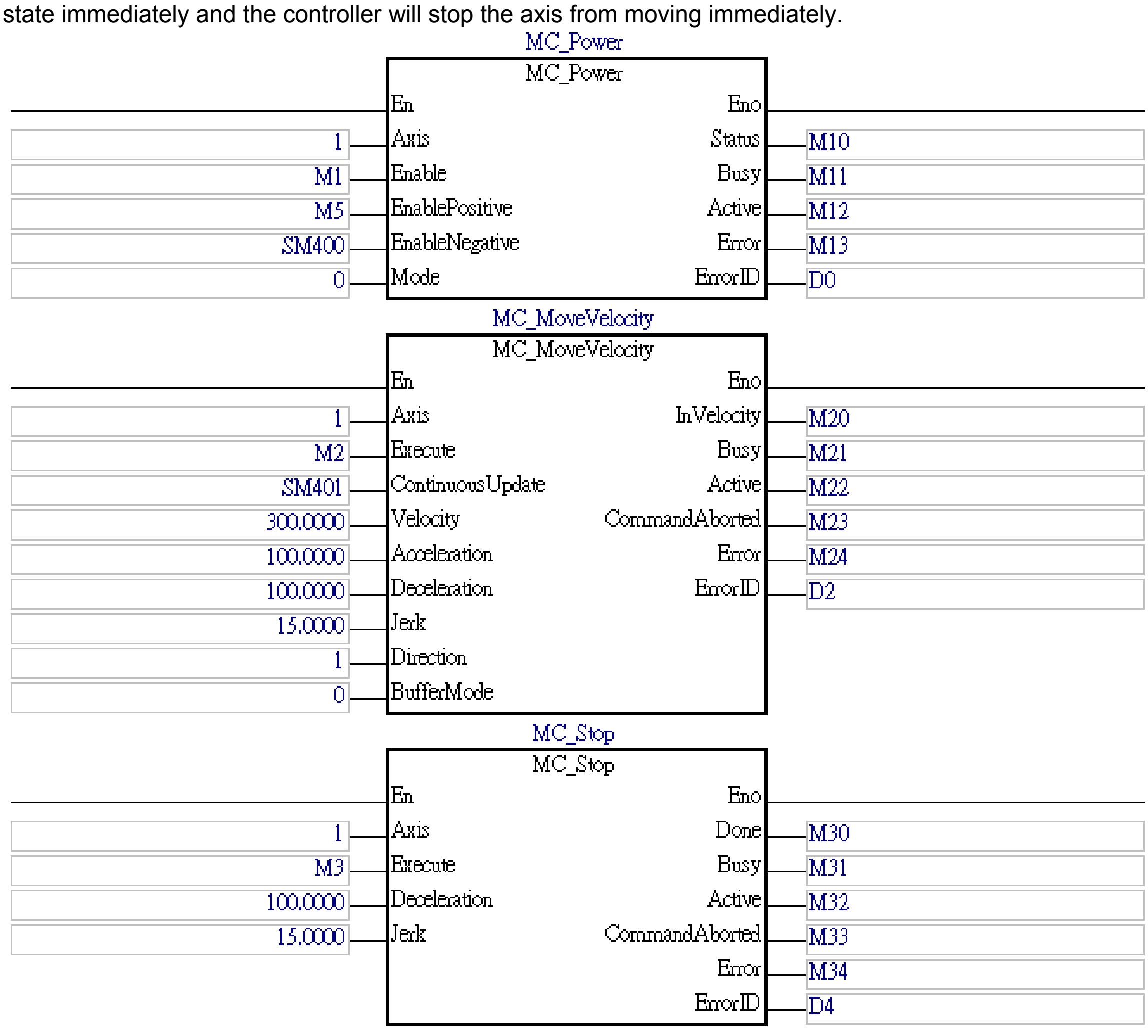


- When M2(*Execute*) changes to True for the first time, the servo motor cannot move and M24(*Error*) changes to True. The servo motor is forbidden to move forward at the moment because M5(*EnablePostive*) is False. When M2(*Execute*) changes to True for the second time with M5(*EnablePostive*)=True , the motion controller starts to move the servo motor forward. When the servo motor reaches the target velocity, M20(*InVelocity*) will change to True.
- When M5(*EnablePostive*) changes to False, the MC_MoveVelocity instruction is aborted and M23(*CommandAborted*) changes to True. Meanwhile the servo motor begins to decelerate its speed at the deceleration rate specified by the MC_MoveVelocity instruction.
- When M2(*Execute*) changes to False, M23(*CommandAborted*) changes to False.
- When M1(*Enable*) changes to False, M10 (*Status*) changes to False.

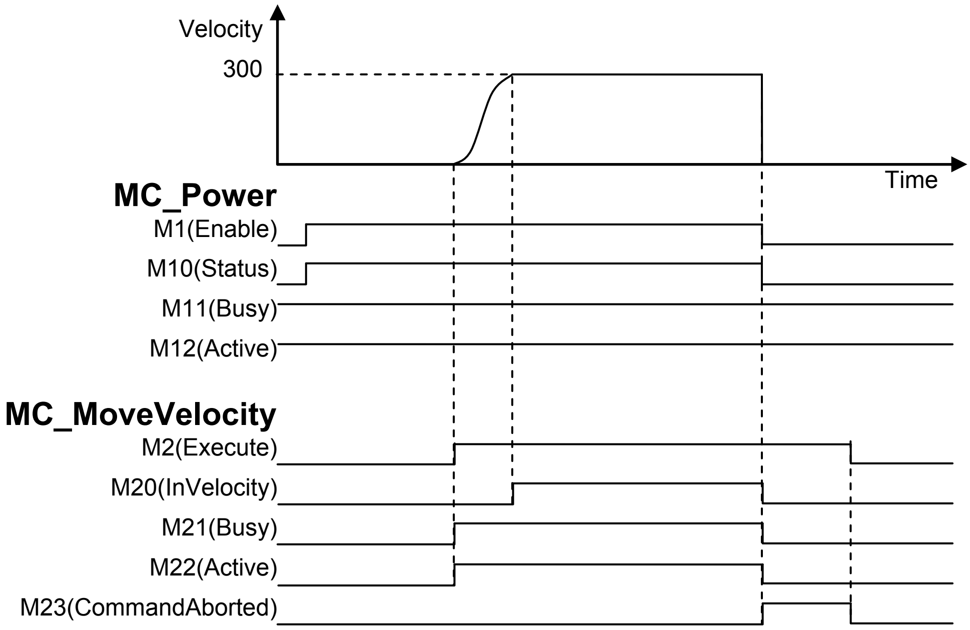
● Programming Example 2

The example of *Mode* input is shown as follows (Mode=0):

When the *Mode* value is 0 (Aborting) and *Enable* changes from True to False for the axis in motion, the axis will enter the Disabled state immediately and the controller will stop the axis from moving immediately.



Motion Diagram:

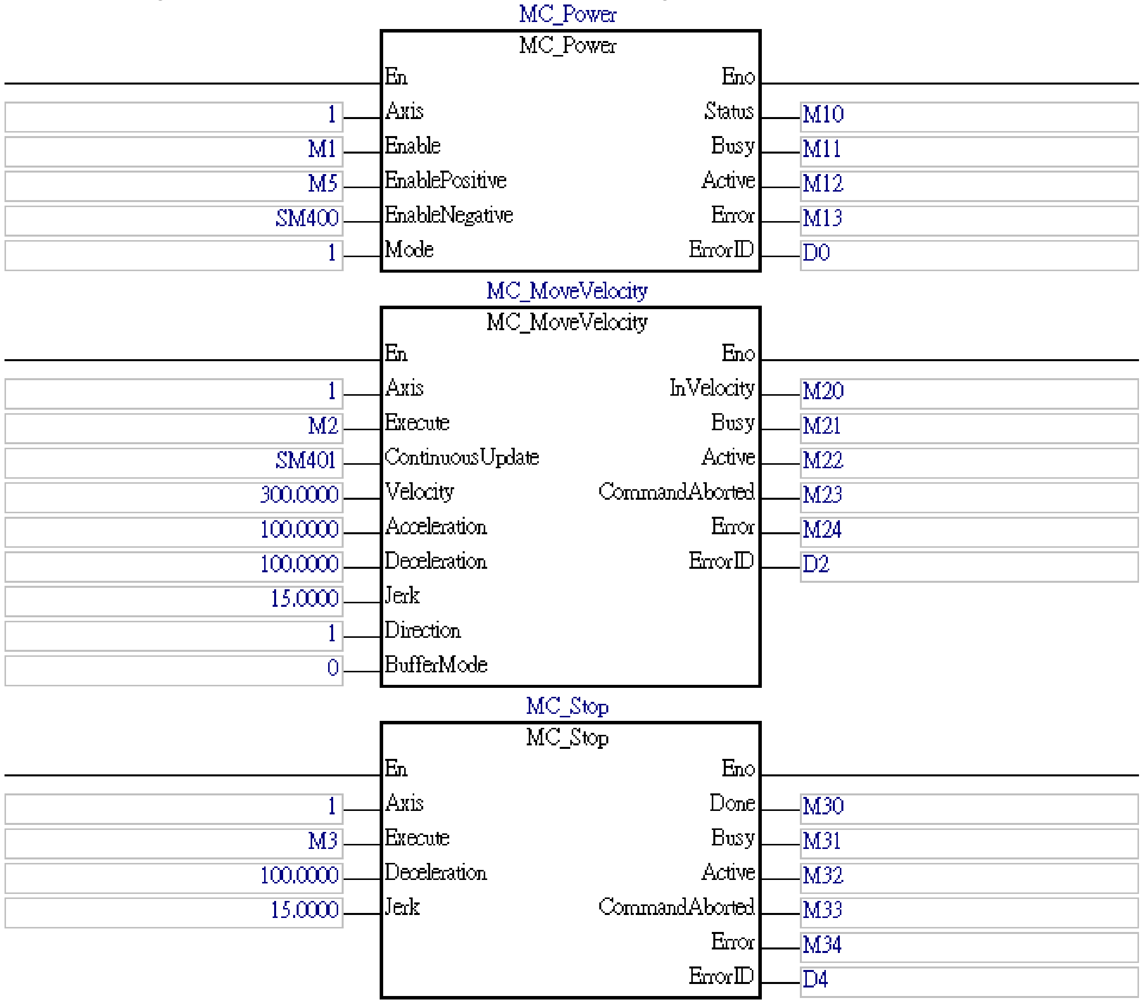


- When M2(*Execute*) changes to True, the motion controller starts to move the servo motor. When the servo motor reaches the target velocity, M20(*InVelocity*) changes to True.
- When M1(*Enable*) changes to False, the servo motor speed reaches 0 and enters the state of "Disabled" and M10(*Status*) changes to False immediately. Meanwhile M23(*CommandAborted*) changes to True and M20(*InVelocity*), M21(*Busy*), and M22(*Active*) change to False.
- When M2(*Execute*) changes to False, M23(*CommandAborted*) changes to False.

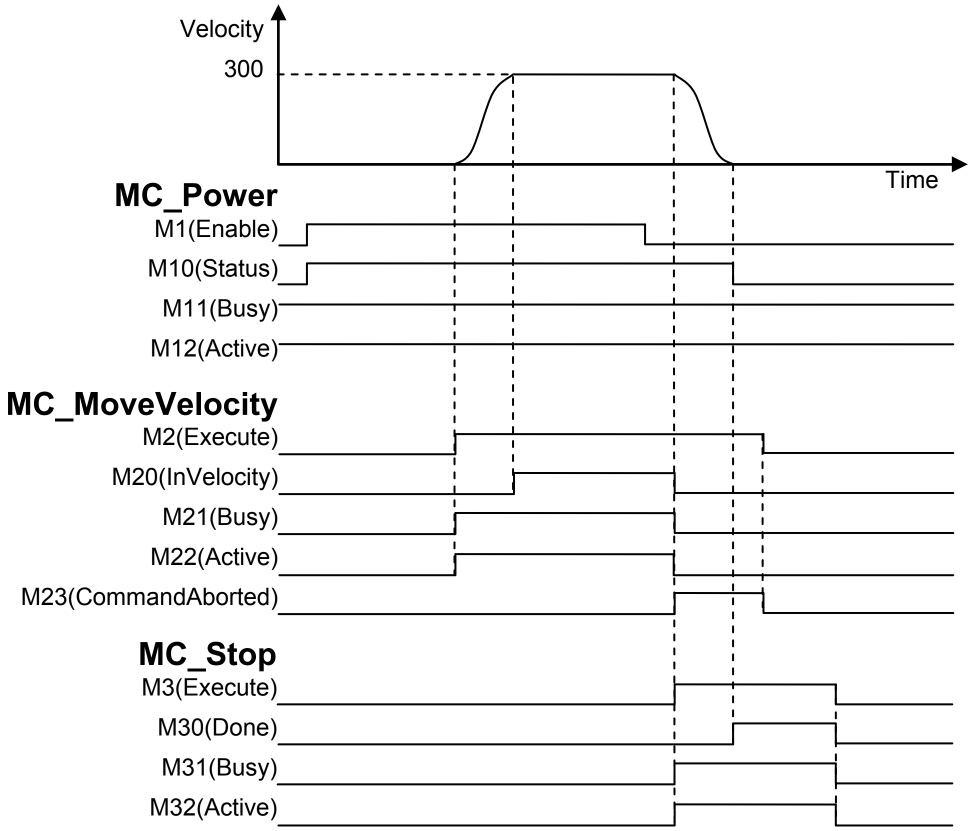
Programming Example 3

The example of *Mode* input is shown as follows (Mode=1):

When the *Mode* value is 1 (Buffered) and *Enable* changes from True to False for the axis in motion, there will be no change for *Status* of MC_Power and the axis state. In this case, the axis will not enter the Disabled state and *Status* of MC_Power will not change from True to False until the axis stops moving.



Motion diagram:

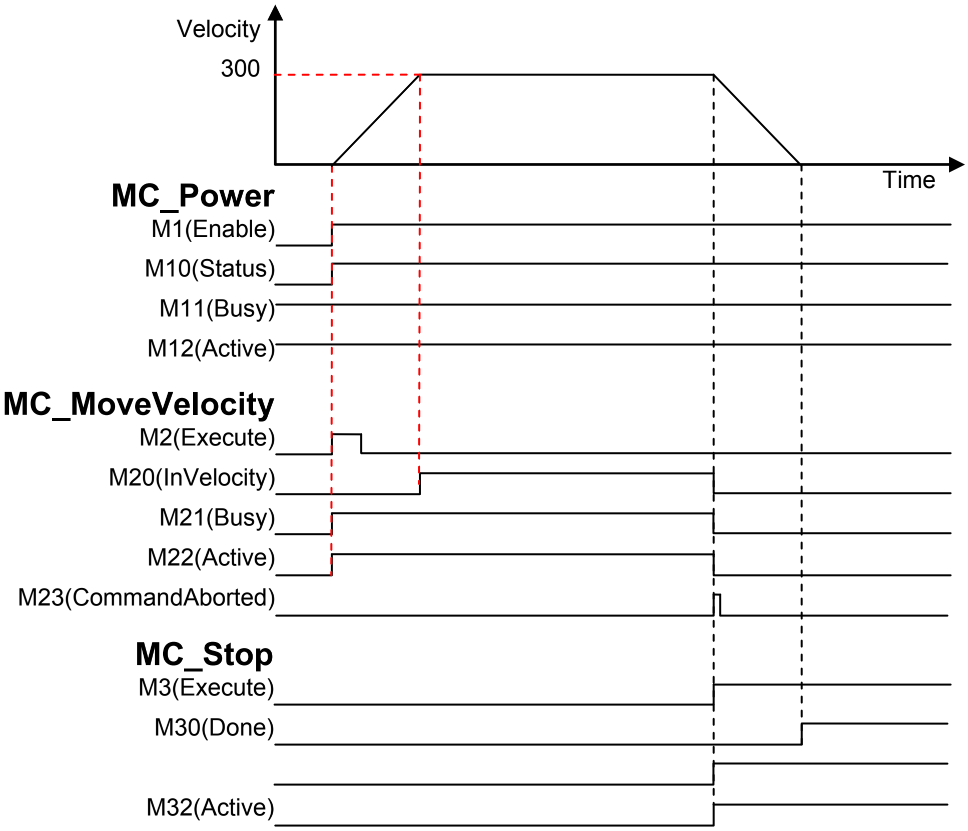


- When M2(*Execute*) changes to True, the motion controller starts to control the servo motor for moving. When the servo motor reaches the target velocity, M20(*InVelocity*) changes to True.
- In this case, when M1(*Enable*) changes to False, the servo motor will not enter the state of Standstill immediately unless the execution of the MC_Stop instruction is completed. When M3(*Execute*) changes to True, the servo motor begins to decelerate. When the servo motor speed reaches 0, M30(*Done*) changes to True. Meanwhile, the servo motor enters the state of "Disabled" and M10(*Status*) change to False.
- When M2(*Execute*) changes to False, M23(*CommandAborted*) changes to False.
- When M3(*Execute*) changes to False, M30(*Done*), M31(*Busy*), and M32(*Active*) change to False.

3

- **Programming Example 4**

The example below shows how to write a motion control program in Structured Text.




```
MC_Power(
        Axis      := 1 ,
        Enable := M1 ,
        EnablePositive := SM400 ,
        EnableNegative := SM400 ,
        Mode := 0 , (* Aborting *)
        Status => M10 ,
        Busy => M11 ,
        Active => M12 ,
        Error => M13 ,
        ErrorID => D0 );

MC_MoveVelocity(
        Axis      := 1 ,
        Execute := M2 ,
        ContinuousUpdate := SM400,
        Velocity := 300 ,
        Acceleration := 100.0 ,
        Deceleration := 100.0 ,
```

```
                    Jerk := 0.0 ,
                    Direction := 0 ,
                    BufferMode := 1 ,    (* emC_BUFFER_MODE.mcAborting *)
                    InVelocity => M20 ,
                    Busy => M21 ,
                    Active => M22 ,
                    CommandAborted => M23 ,
                    Error => M24 ,
                    ErrorID => D2 );

        MC_STOP(
                    Axis := 1 ,
                    Execute := M3 ,
                    Deceleration := 100.0 ,
                    Jerk := 0.0 ,
                    Done => M30 ,
                    Busy => M31 ,
                    Active => M32 ,
                    CommandAborted => M33 ,
                    Error => M34 ,
                ErrorID => D4 );

    (* When MC_Power.Status = TRUE , Execute MC_MoveVelocity *)
    IF M10 THEN
        TMR (T0, 10) ;
    ELSE
        T0 := FALSE ;
    END_IF;

    M2 := M10 & NOT T0 ;

    (* When MC_MoveVelocity.InVelocity = TRUE , Start Timer T1 *)
    IF M20 THEN
        TMR (T1, 100) ;
    ELSE
        T1 := FALSE;
    END_IF;

    (* When Timer T1=10sec , execute MC_Stop *)
    IF T1 THEN
        M3 := TRUE ;
    ELSE
        M3 := FALSE ;
    END_IF;
```

- When M1(*Enable*) is set to True manually, the system will execute as below process.

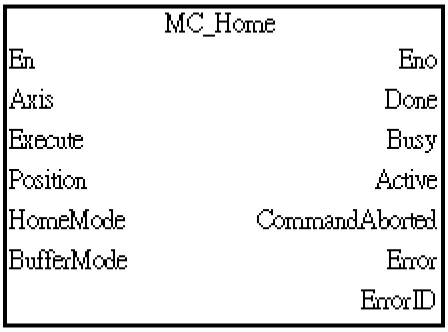
- When M10(*Status*) changes to True, M2 changes to True immediately and the servo motor starts to accelerate. After 1 second, M2(*Execute*) is reset to False because of the activation of Timer 0(T0).
- When the speed reaches 300, M20(*InVelocity*) changes to True and Timer 1(T1) is executed to time for 10 seconds.
- After 10 seconds, M3(*Execute*) is set to True and the servo motor is stopped.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion controller modules: AHxxEMC-5A

MC_Home

FB/FC	Description
FB	MC_Home drives the axis to perform the homing operation.



- *Position* is used to specify the absolute home position after homing is completed.
- The instruction will complete in “Standstill” state if it was executed in “Standstill” state.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Position	Sets the home absolute home position after homing is completed. (Unit: user unit)*1	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False
HomeMode	Specifies the axis behavior of the homing process.	WORD	0-35 (0)	When *Execute* shifts to True and *Busy* is False
BufferMode	Reserved	eMC_BUFFER_MODE		When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

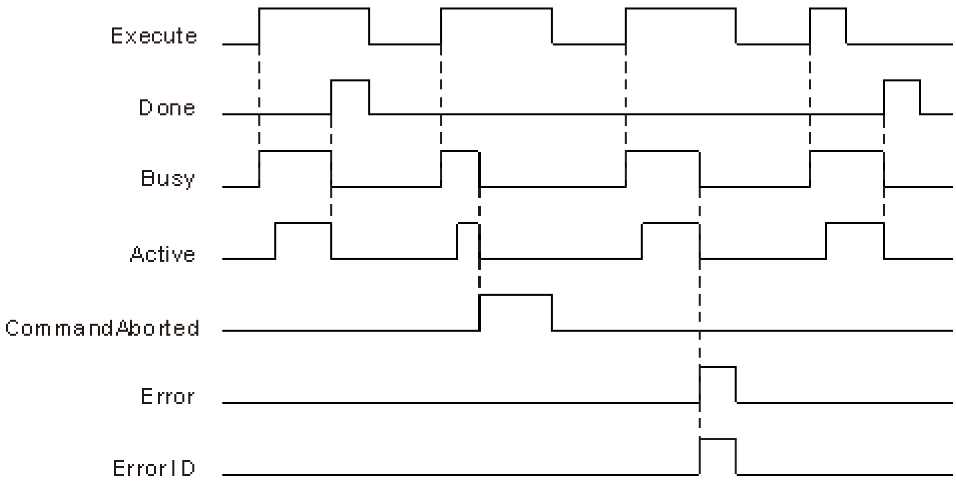
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when homing is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the homing is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
Active	• When the motion on the axis is started.	• When *Done* shifts to False. • When an error occurs • When *CommandAborted* shifts to True. • If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	• When this instruction is aborted by another instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



3

In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- When *Execute* changes to True, the homing operation starts to be performed on the axis specified in *Axis.*
- The parameters used in MC_Home can be set in the axis' parameters.
- There are 36 homing methods supported by Delta motion controllers and Delta servo drives. You can specify the desired homing mode in the instruction.
- When working with the instruction MC_STOP and set the deceleration to 0, users need to check the definition of OD 6085 hex on the corresponding slave.

■ **Software Limit**

◆ When the axis runs beyond the software limit set, an error will be reported.

◆ When the axis is in the homing mode, software limit will not be taken into account. (Even if the axis runs beyond the software limt set, no error will be reported.)

■ **Homing Modes**

Delta servo drives, e.g. ASDA A2-E, supports up to 35 homing modes which are defined in CiA 402*. The supported homing modes are manufacturer dependent. For the supported homing modes of products other than Delta drives, refer to the information provided by the servo drive manufacturer.

Note: CiA stands for CAN in Automation, which is the nonprofit organization that supports standardization of CAN protocols and develops and publishes CiA specifications**.**

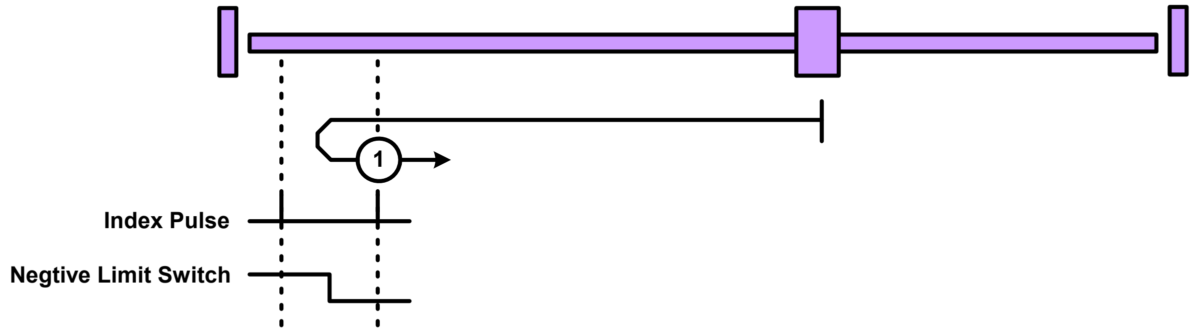
Homing modes	Description	Delta servo drive (ASDA-A2-E)
0	Reserved	None
1	Homing on the negative limit switch and index pulse	OK
2	Homing on the positive limit switch and index pulse	OK
3	Homing on the positive home switch and index pulse	OK
4	Homing on the positive home switch and index pulse	OK
5	Homing on the negative home switch and index pulse	OK
6	Homing on the negative home switch and index pulse	OK
7	Homing on the home switch and index pulse	OK
8	Homing on the home switch and index pulse	OK
9	Homing on the home switch and index pulse	OK
10	Homing on the home switch and index pulse	OK
11	Homing on the home switch and index pulse	OK
12	Homing on the home switch and index pulse	OK
13	Homing on the home switch and index pulse	OK
14	Homing on the home switch and index pulse	OK
15	Reserved	None
16	Reserved	None
17	Like 1 but Homing without an index pulse	OK
18	Like 2 but Homing without an index pulse	OK
19	Like 3 but Homing without an index pulse	OK
20	Like 4 but Homing without an index pulse	OK
21	Like 5 but Homing without an index pulse	OK
22	Like 6 but Homing without an index pulse	OK
23	Like 7 but Homing without an index pulse	OK
24	Like 8 but Homing without an index pulse	OK
25	Like 9 but Homing without an index pulse	OK
26	Like 10 but Homing without an index pulse	OK
27	Like 11 but Homing without an index pulse	OK
28	Like 12 but Homing without an index pulse	OK

Homing modes	Description	Delta servo drive (ASDA-A2-E)
29	Like 13 but Homing without an index pulse	OK
30	Like 14 but Homing without an index pulse	OK
31	Reserved	None
32	Reserved	None
33	Homing on the index pulse	OK
34	Homing on the index pulse	OK
35	Homing on the current position	OK

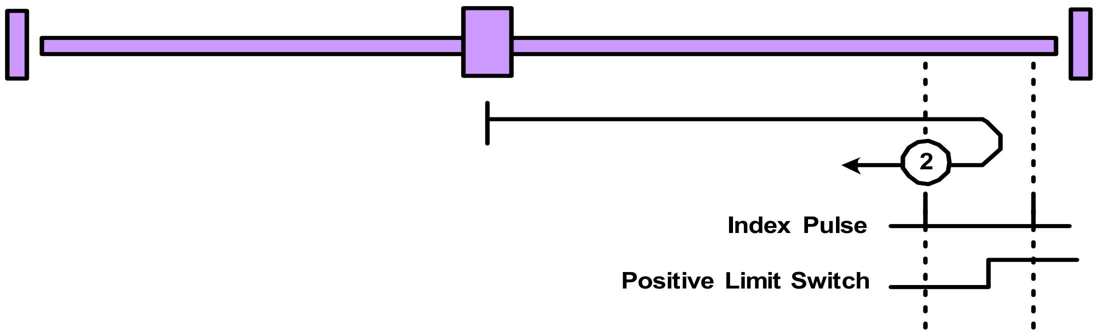
■

■ **Behavior Descriptions of Homing Modes**

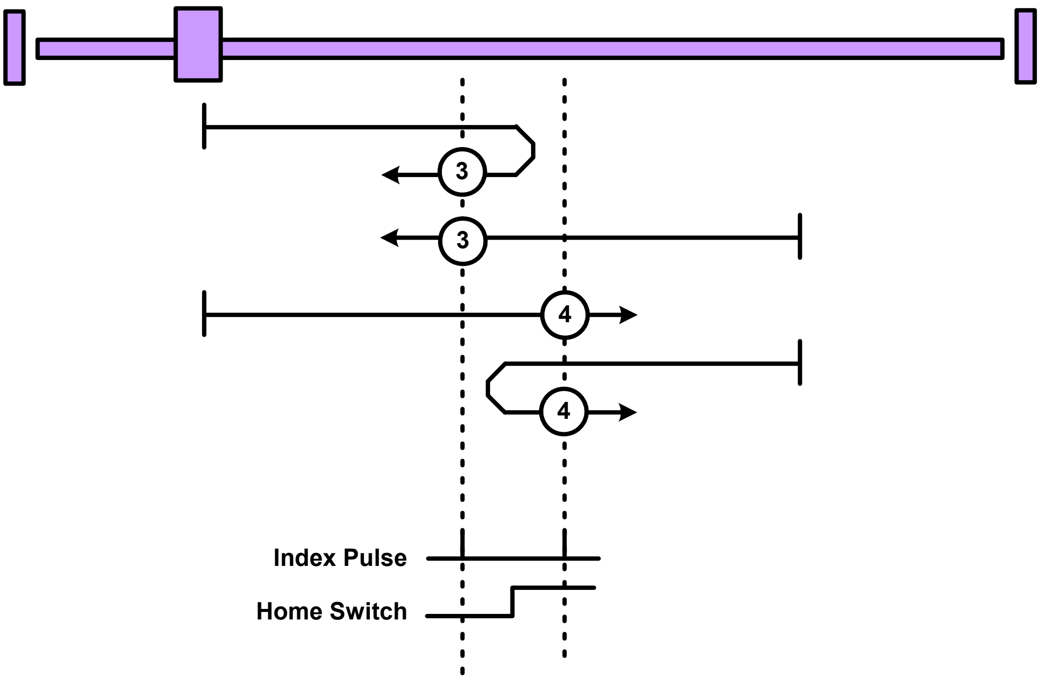
Mode 1: Homing on negative limit switch and index pulse



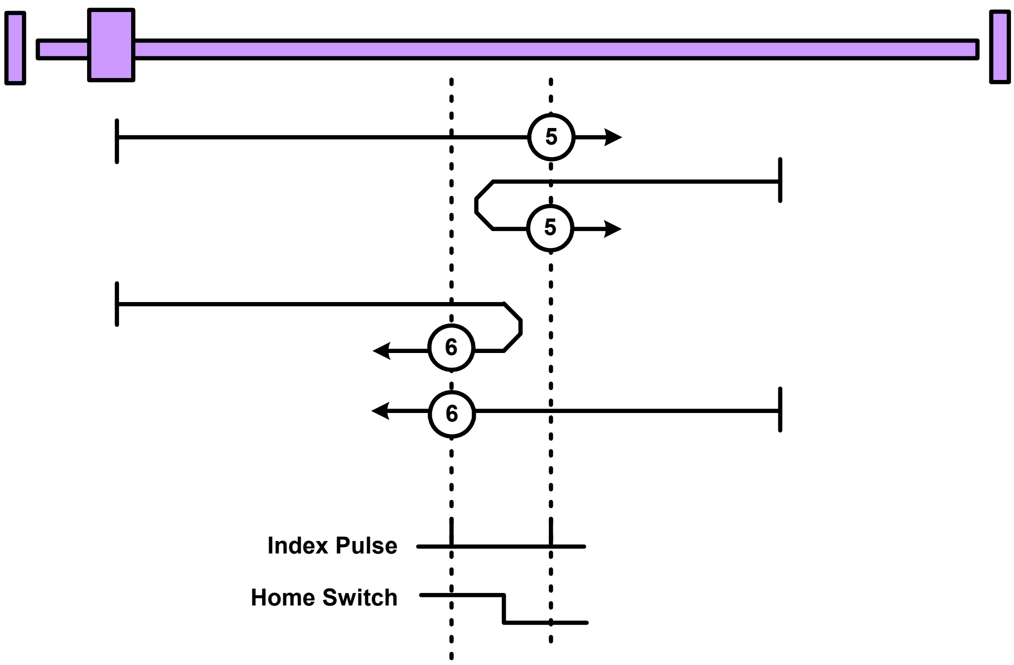
Mode 2: Homing on positive limit switch and index pulse



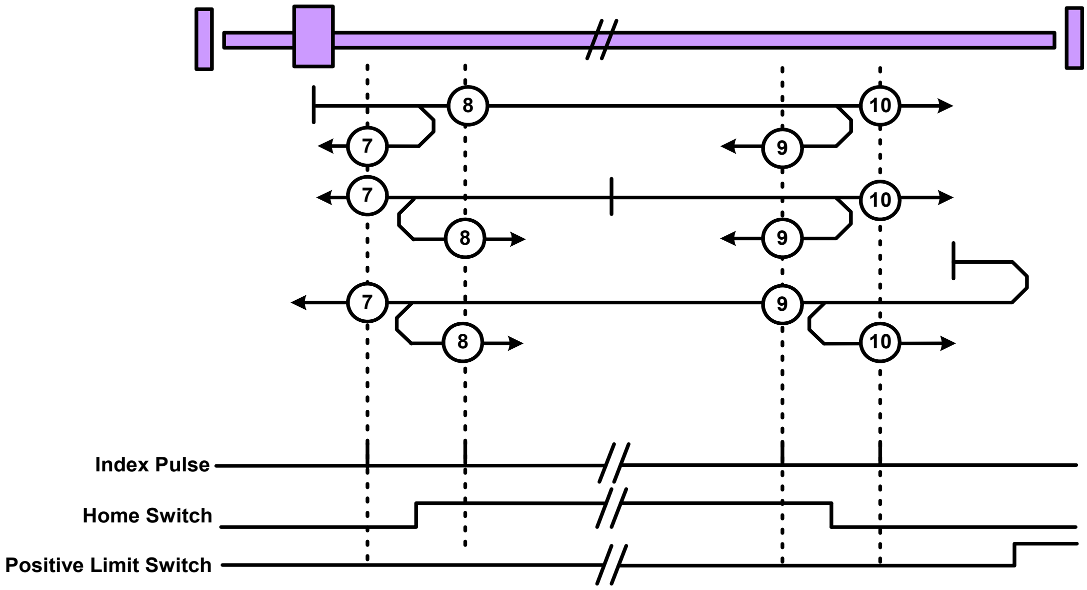
Mode 3 and 4: Homing on positive home switch and index pulse

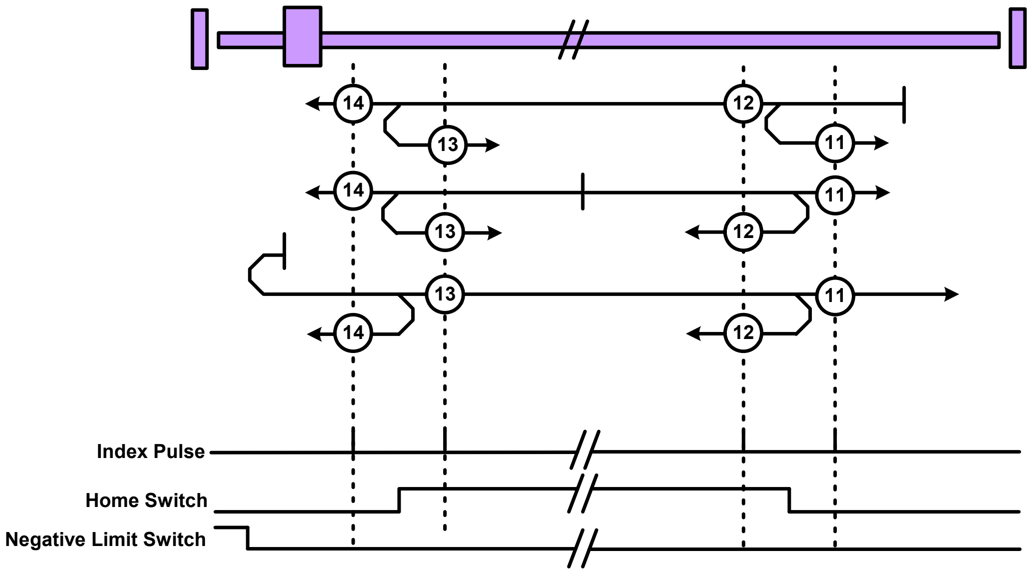


Mode 5 and 6: Homing on negative home switch and index pulse



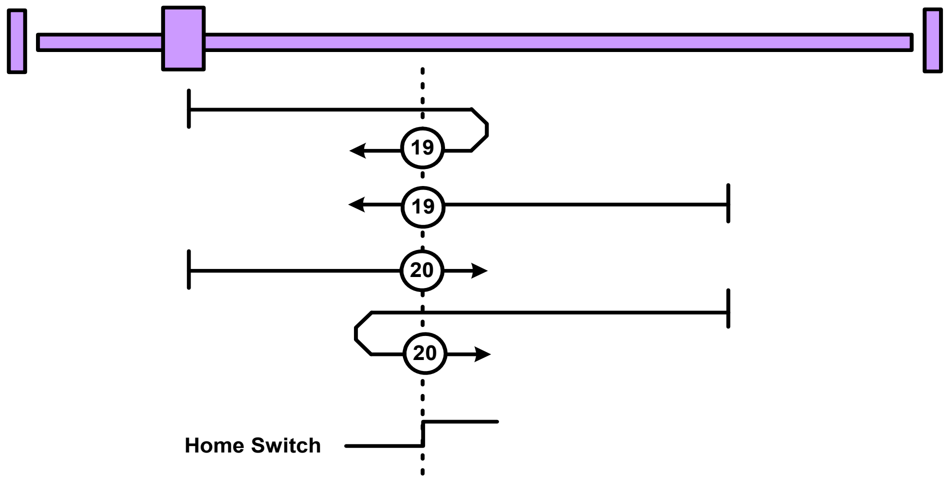
Mode 7 to 14: Homing on home switch and index pulse



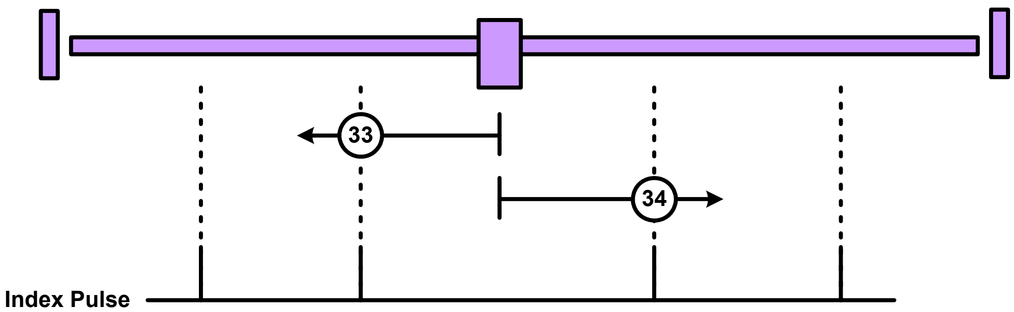


Mode15 and 16: Reserved

Mode 17 to 30: Homing without an index pulse



Mode 31 and 32: Reserved (no picture)



Mode 33 to 34: Homing on index pulse

Mode 35: Homing on current position (no diagram)

Troubleshooting

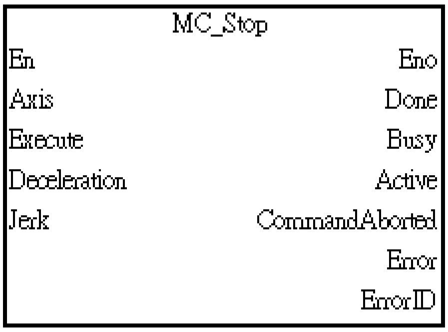
- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_Stop

FB/FC	Description
FB	MC_Stop decelerates an axis to a stop.



- MC_Stop stops a moving axis according to the specified mode and transfers the axis to the state of “Standstill”.
- It aborts any executing instruction. No other instructions can be executed while the axis is in the state of “Stopping”.
- When 0 velocity is reached, *Done* will be True immediately and the state of “Stopping” remains.
- The axis enters the state of “Standstill” when *Done* changes to True and *Execute* changes to False.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Deceleration	Deceleration rate (Unit: user unit/s^2)*	LREAL	Positive number or 0 (0)	When *Execute* is rising edge triggerred and *Busy* is False.
Jerk	Jerk value (Unit: user unit/s^3)*	LREAL	Positive number or 0 (0)	When *Execute* is rising edge and *Busy* is False.

***Note:** Set the deceleration to 0, while executing MC_Stop; the system will go to Immediate Stop or Deceleration Stop mode according to the parameters set in ISPSoft. (When working with the instruction MC_Home and set the deceleration to 0, users need to check the definition of OD 6085 hex on the corresponding slave.)

Refer to **Section 2.2.1 Parameters for Motion Axes: Structure** for explanation on setting axis parameters.

● Outputs

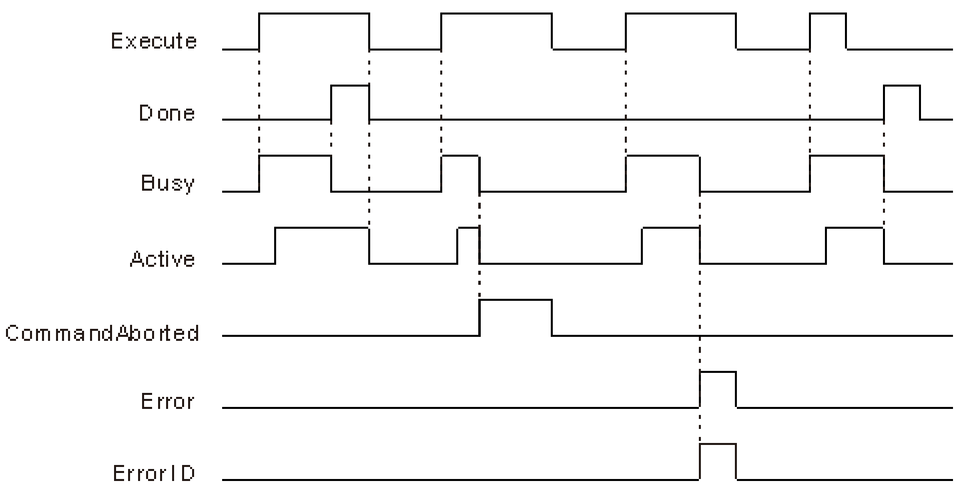
Name	Function	Data type	Output range (Default value)
Done	True when zero velocity is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled. After a complete stop *Active* remains True until the axis is released with *Execute*=False.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)

Name	Function	Data type	Output range (Default value)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	● True when the axis decelerates to a stop and reaches zero velocity.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● True when *Execute* shifts to True and the instruction is executed.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● True when this instruction is started.	● When *Execute* shifts to False and *Done* is True. ● When *Done* shifts to True and *Execute* is False. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True. ● If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	● When this instruction is aborted because another motion control instruction is executed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram



3

● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

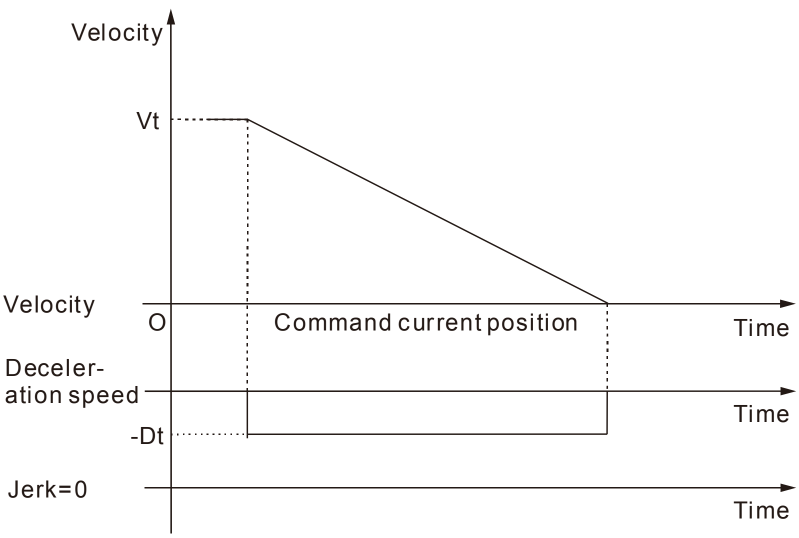
● Function

You can set the inputs *Deceleration* and *Jerk* to specify the deceleration rate and jerk pattern when decelerating to a stop.

-The relationship between the deceleration rate, the velocity, and the jerk values is shown below

Jerk=0

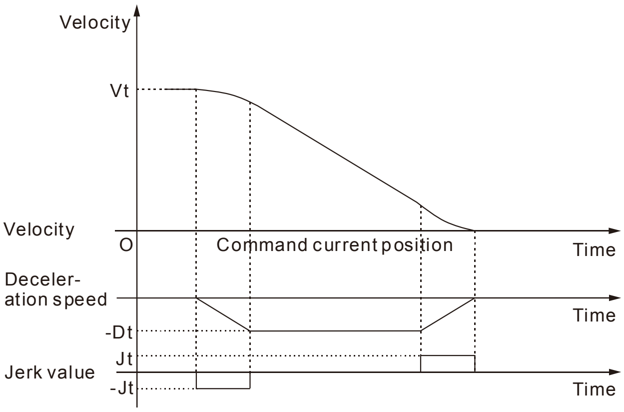
The velocity value is created according to the specified deceleration rate Dt



Vt: Velocity before the deceleration starts, Dt: The specified deceleration speed

Jerk≠0

The velocity value is created according to the specified deceleration rate Dt, which functions as the upper limit to the deceleration speed.



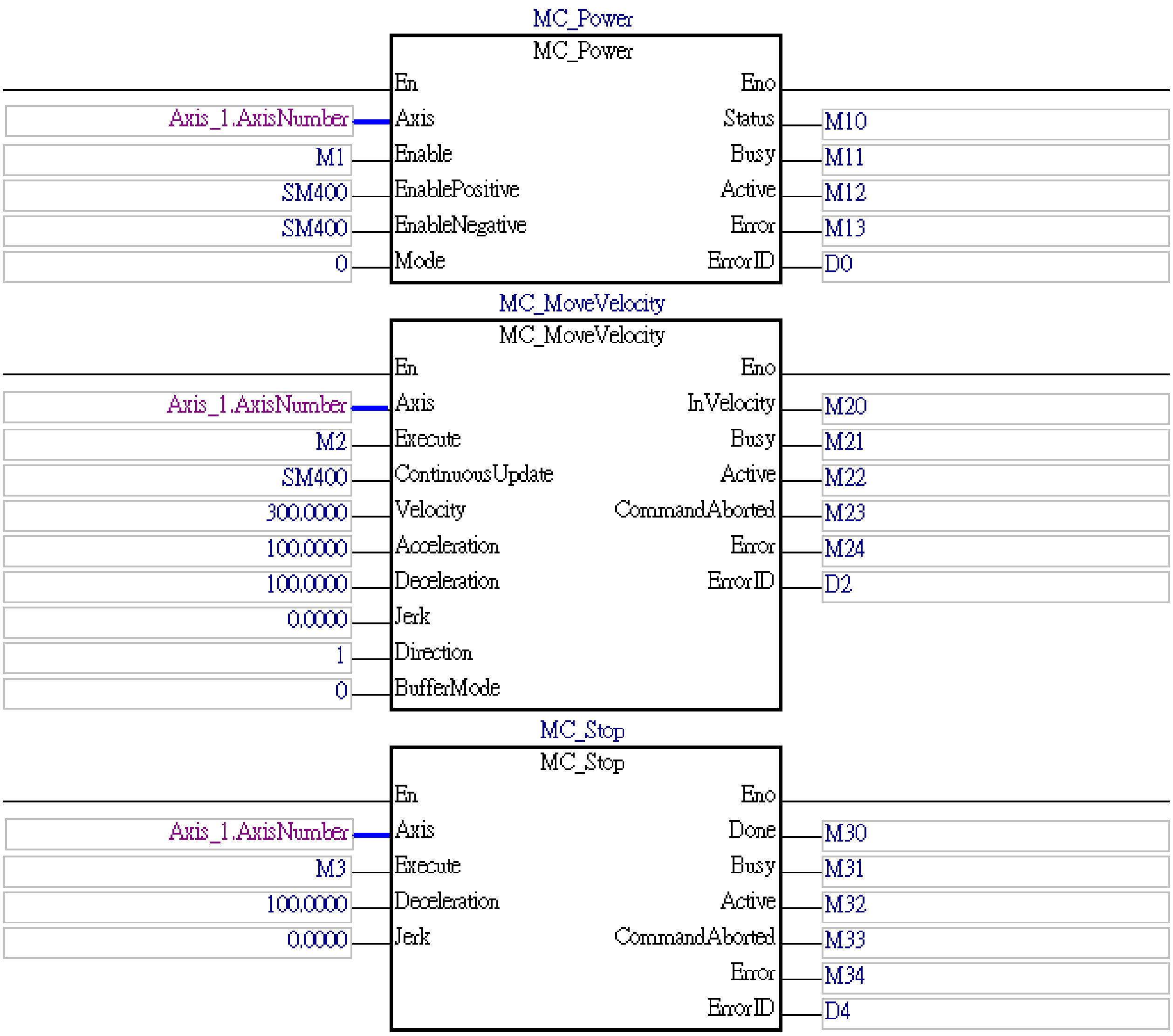
Vt: Velocity before the deceleration slope starts, Dt: The specified deceleration rate, Jt: The specified jerk value

Troubleshooting

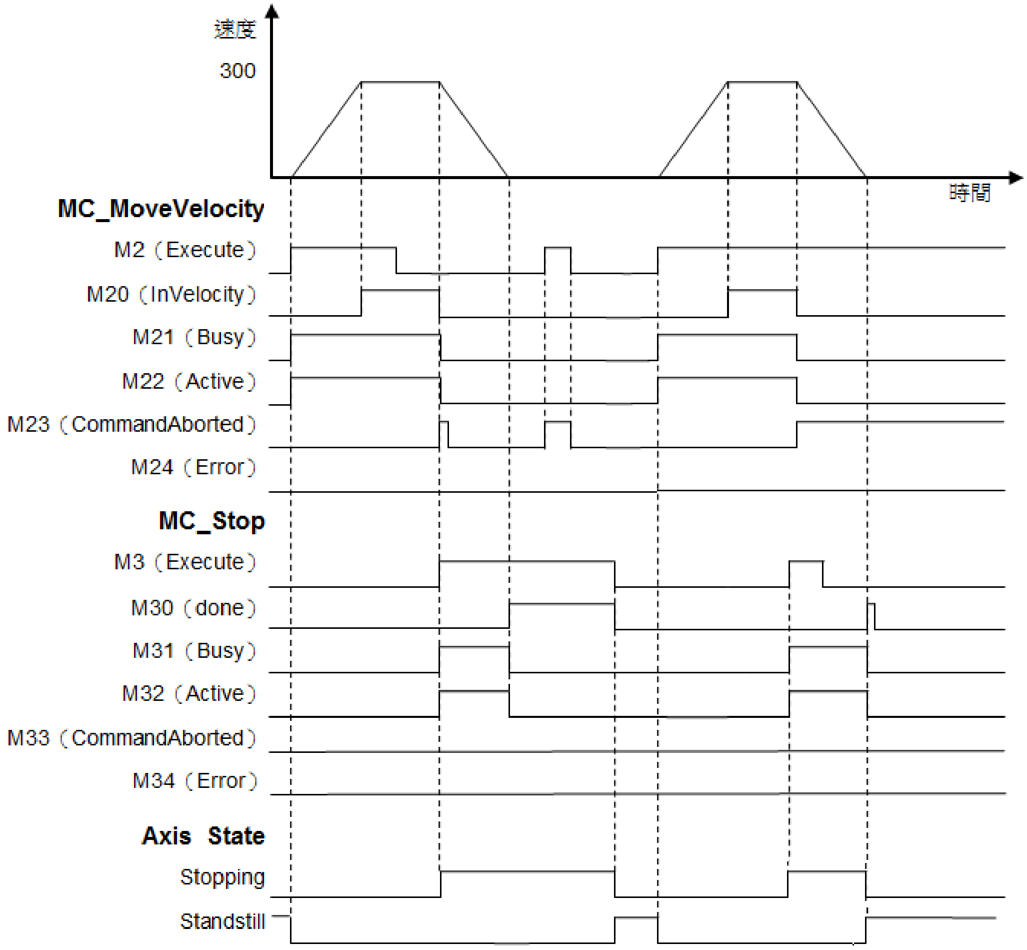
- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

Programming Example

The example below shows the behavior when MC_Stop is used with MC_MoveVelocity.
A moving axis is delcelerating with MC_Stop instruction.
The axis rejects motion instructions when *Execute* of MC_Stop is True. If any motion instruction (ex. MC_MoveVelocity) is executed, when MC_Stop is busy, it will report CommandAborted.


MC_Power
MC_Power
En
Eno
Axis_1.AxisNumber
Axis
Status
M10
M1
Enable
Busy
M11
SM400
EnablePositive
Active
M12
SM400
EnableNegative
Error
M13
0
Mode
ErrorID
D0
MC_MoveVelocity
MC_MoveVelocity
En
Eno
Axis_1.AxisNumber
Axis
InVelocity
M20
M2
Execute
Busy
M21
SM400
ContinuousUpdate
Active
M22
300.0000
Velocity
CommandAborted
M23
100.0000
Acceleration
Error
M24
100.0000
Deceleration
ErrorID
D2
0.0000
Jerk
1
Direction
0
BufferMode
MC_Stop
MC_Stop
En
Eno
Axis_1.AxisNumber
Axis
Done
M30
M3
Execute
Busy
M31
100.0000
Deceleration
Active
M32
0.0000
Jerk
CommandAborted
M33
Error
M34
ErrorID
D4

Motion diagram:



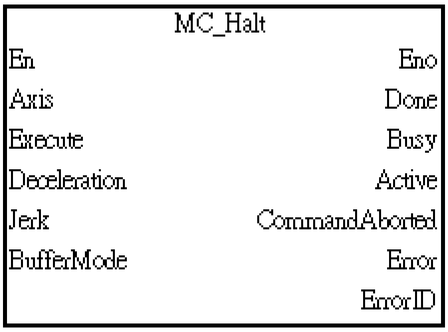
- When M3(*Execute*) of MC_Stop changes to True, it triggers M23(*CommandAboted*) of MC_MoveVelocity at the same time and the motion controller starts to decelerate the axis to a stop. The Axis state is moved to the “Stopping”.
- When the axis reaches zero velocity, M30(*Done*) will change to True. M3(*Execute*) is still True so the axis state remains in the state “Stopping”.
- If MC_MoveVelocity is executed again while the axis state is “Stopping”, the execution of MC_MoveVelocity will be aborted and CommandAborted will be reported (M23=True)
- As soon as M30(*Done*) is set ON and M3(*Execute*) is False, the axis state goes to “Standstill”.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_Halt

FB/FC	Description
FB	MC_Halt halts an axis.



- MC_Halt stops a moving axis which is under normal operation and tansfers the axis to the state “DiscreteMotion” until zero velocity is reached. When the axis stopped, *Done* changes to True and the axis enters the state “Standstill”.
- It is possible to execute another motion instruction during deceleration of the axis if it is not in a buffer mode. The executed motion instruction will abort MC_Halt and operate immediately.
- During the execution of MC_Halt, it is invalid to execute this instruction repeatedly.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Deceleration	Deceleration rate. (Unit: user unit/s^2) *1	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3) *1	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER _MODE*2	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

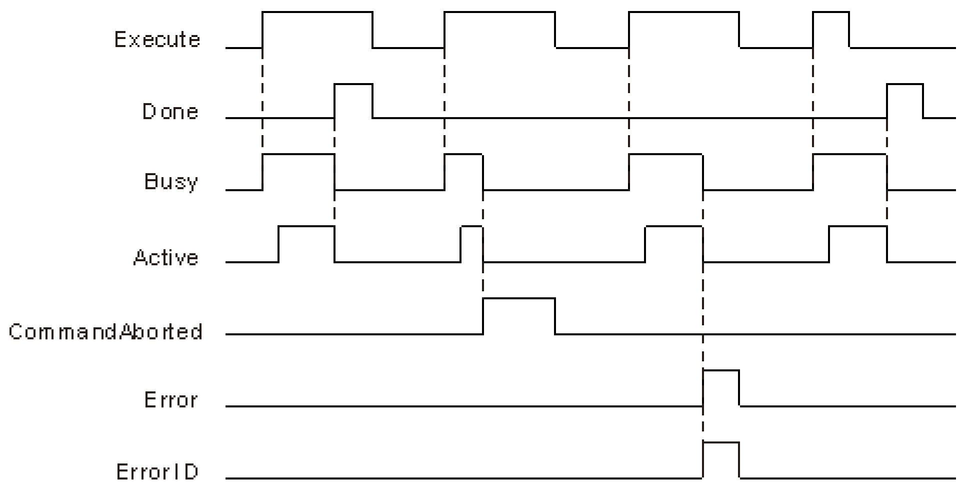
● Outputs

Name	Function	Data type	Output range (Default value)
Done	True when zero velocity is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the axis decelerates to a stop and reaches zero velocity.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* shifts to True and the instruction is executed.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● When this instruction is started.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True. ● If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	● When this instruction is aborted because another motion control instruction is executed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



3

In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

■ **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in "Standstill" state.

The following table lists the available buffer mode settings of MC_Halt

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

The following table lists the buffer effects of MC_Halt.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_Halt	YES	YES	*Done*

For more information of buffer mode, refer to section ***AH Motion Controller – Operation Manual***.

Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.

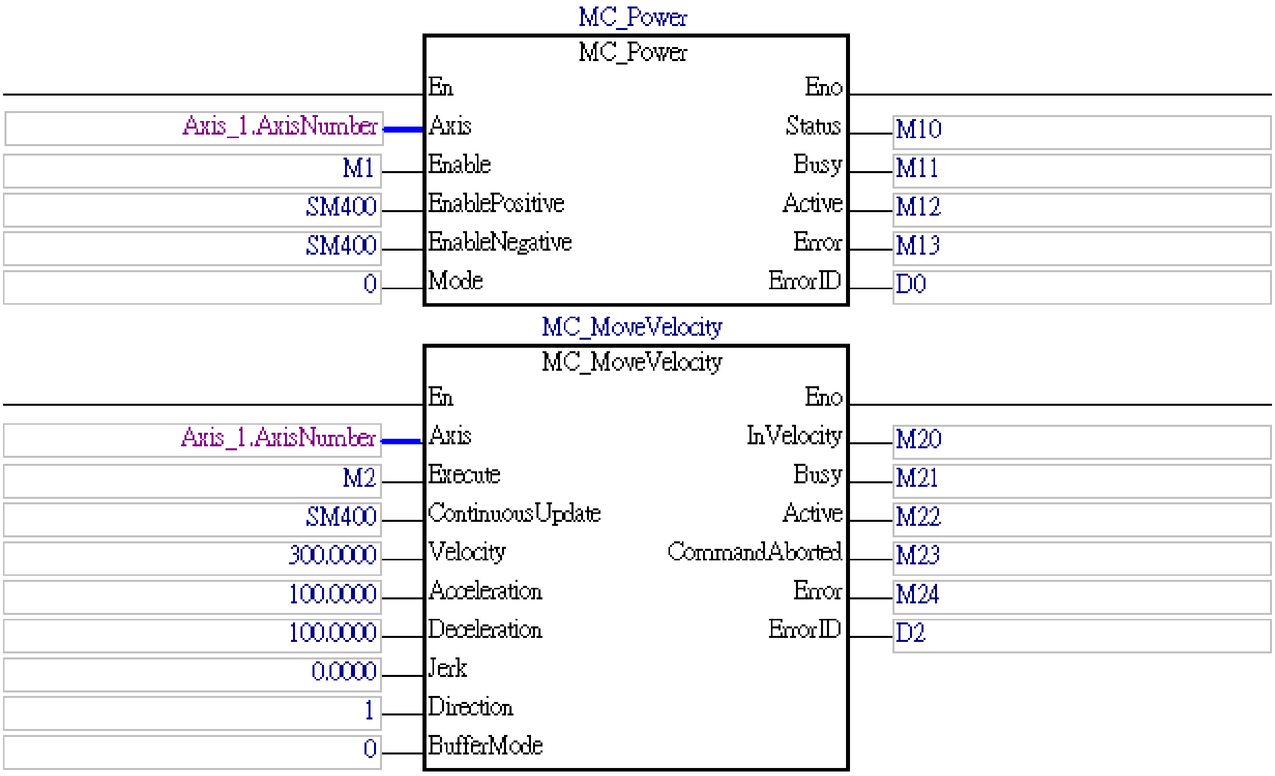
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

● Programming Example

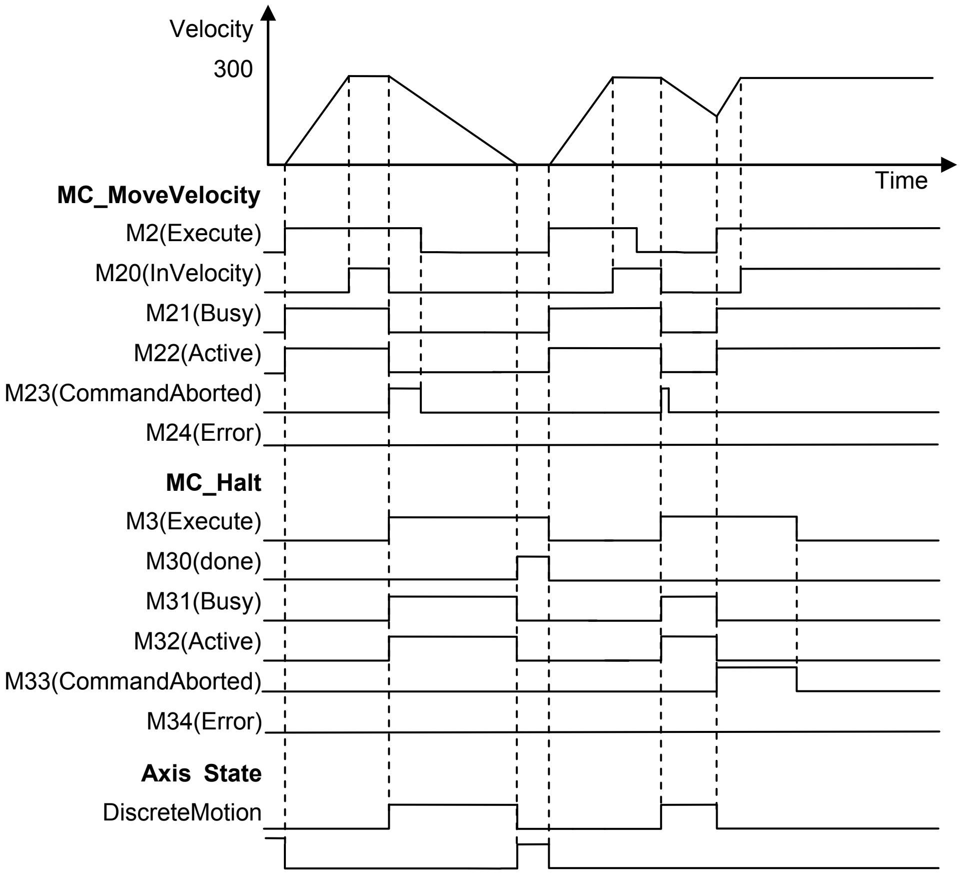
The example below describes the behavior of MC_Halt in combination with MC_MoveVelocity.

The MC_Halt stops MC_MoveVelocity if there is no another instruction executed before the axis enters “Standstill” state.

If MC_MoveVelocity executes again during the deceleration, it will abort MC_Halt immediately and accelerate again without entering “Standstill” state. This re-execution behavior is allowed for MC_Halt but not allowed in MC_Stop.



Motion diagram:



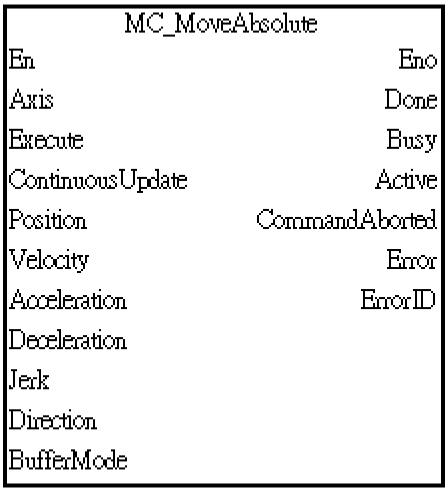
- When M3(*Execute*) of MC_Halt changes to True, it triggers M23(*CommandAboted*) of MC_MoveVelocity at the same time and the motion controller starts to decelerate the axis to a stop. The Axis state is moved to the “DiscreteMotion”.
- When the axis reaches zero velocity, M30(*Done*) will change to True. The axis state transferred to “Standstill”.
- If MC_MoveVelocity is executed again while MC_Halt is decelerating the axis and M3(*Execute*) is True, the execution of MC_Halt will be aborted M33(*CommandAboted*) = True and M2(*Execute*) changes to True. MC_MoveVelocity will accelerate again.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_MoveAbsolute

FB/FC	Description
FB	MC_MoveAbsolute controls the axis to move to the specified absolute target position at a specified behavior.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates *Velocity* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Position	Absolute target position (Unit: user unit)*1	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	Target velocity (Unit: user unit/s)*1	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate (Unit: user unit/s^2)*1	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate (Unit: user unit/s^2)*1	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value (Unit: user unit/s^3)*1	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Direction	Specifies the direction for servo motor rotation. The input is effective only for	eMC_DIRECTION*	1: mcPositiveDirection 2: mcShortestWay 3: mcNegativeDirection	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
	modulo/rotary axis.		4: mcCurrentDirection (1)	
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. When one MC_MoveAbsolute instruction is executed but not finished yet, it is invalid to re-execute the instruction.

3

● Outputs

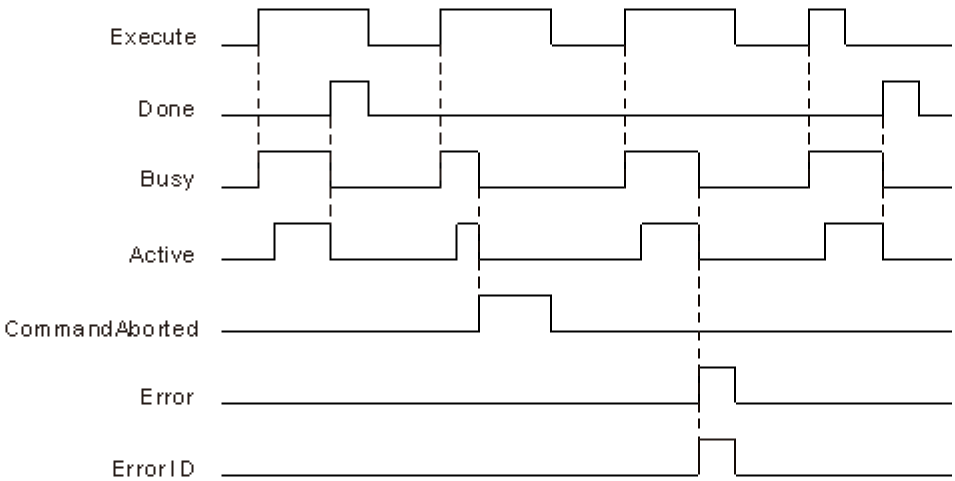
Name	Function	Data type	Output range (Default value)
Done	True when absolute target position is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the absolute positioning is completed.	● When *Execute* shifts to False. ● If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● When the motion on the axis is started	● When *Done* shifts to True. ● When *Error* shifts to True ● When *CommandAborted* shifts to True.

Name	Timing for shifting to True	Timing for shifting to False
		● If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted because of the execution of MC_Stop instruction.	● When Execute shifts to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

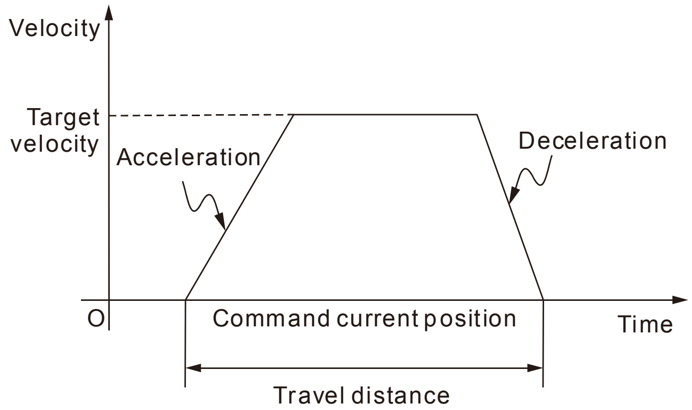
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

The instruction performs absolute positioning with specified target velocity (*Velocity*), acceleration rate (*Acceleration*), deceleration rate (*Deceleration*) and Jerk value (*Jerk*) when *Execute* changes to True.

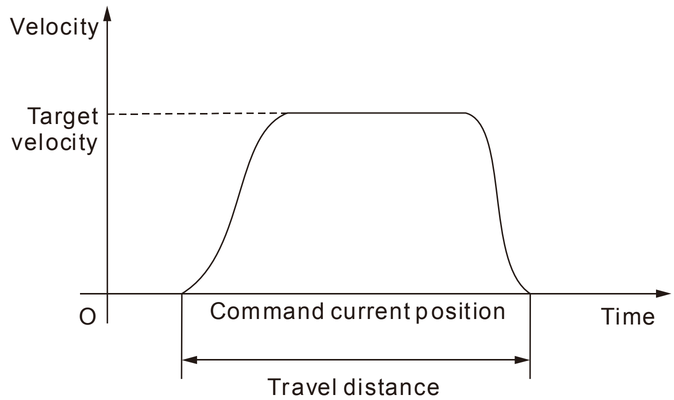
- The motion path of absolute positioning is described as below.

Jerk=0



Jerk≠0

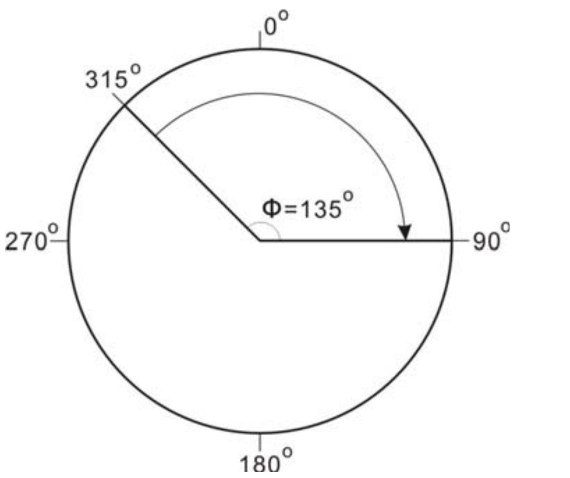
Setting up Jerk value allows you to control the motion path to ramp up (accelerate) or ramp down (decelerate) smoothly. The effects on a motion path with Jerk specified are as below.

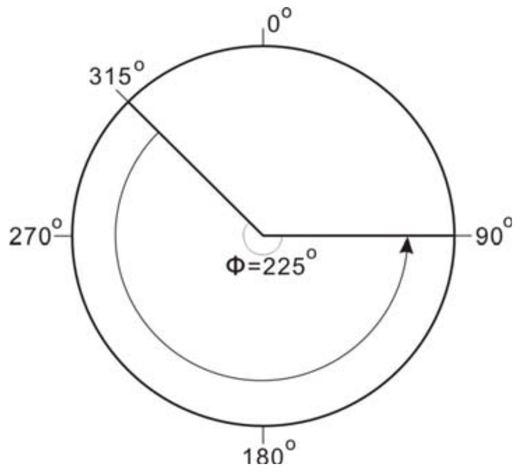


Direction Settings

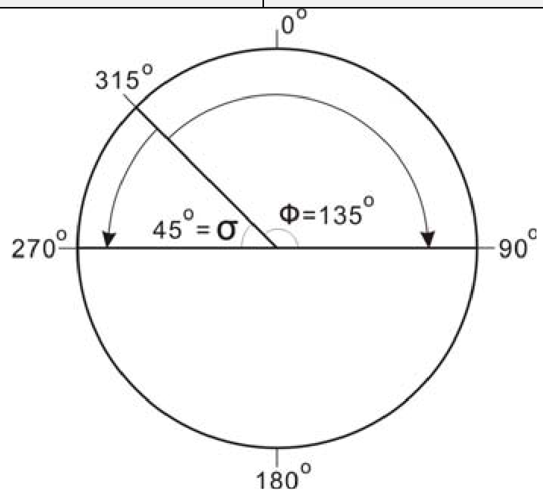
- *Direction* is used to define the rotation of servo axis and is effective only for modulo/rotary axis.
- When the direction value is different, the motion direction and the travel distance of the rotary axis will be different as follows. Suppose the output unit of the physical device is “degree”, the motion direction of the rotary axis is illustrated as follows:

Direction: 1 (Positive direction) Current position: 315° Target position: 90° Movement angle: 135°	*Direction*: 3 (Negative direction) Current position: 315° Target position: 90° Movement angle: 225°

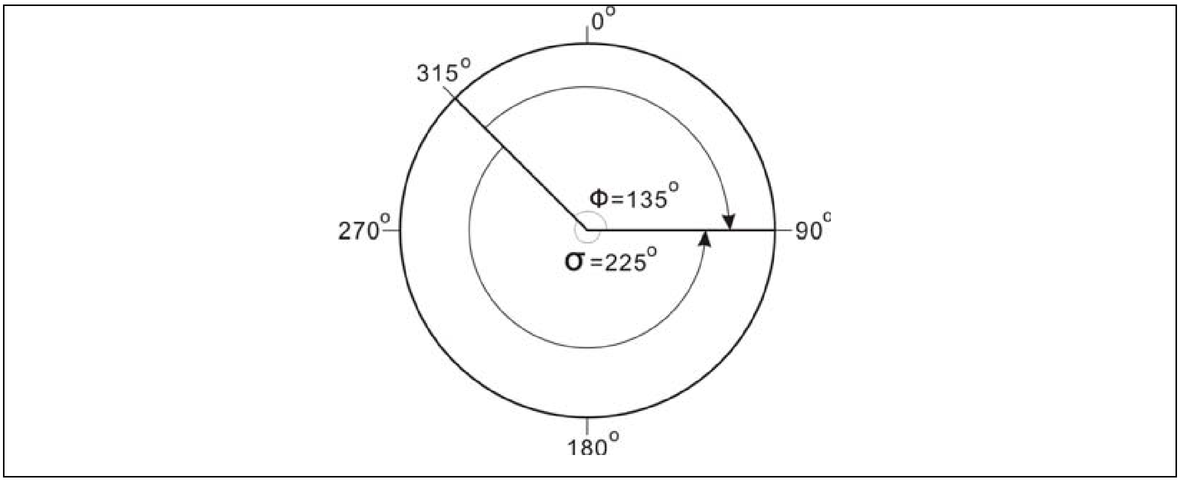




Direction: 2 (Shortest way) Current position: 315° Target position: 90° Movement angle: 135°	*Direction*: 2 (Shortest way) Current position: 315° Target position: 270° Movement angle: 45°



Direction: 4 (Current direction) Rotary axis status: Moving in the negative direction before the function block is executed. Current position: 315° Target position: 90° Movement angle: 225°	*Direction*: 4 (Current direction) Rotary axis status: motionless or moving in the positive direction before the function block is executed. Current position: 315° Target position: 90° Movement angle: 135°



■ **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in “Standstill” state.

The following table lists the available buffer mode settings of MC_MoveAbsolute.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.
2: mcBlendingLow	Takes the lower target velocity as the transit velocity between the current instruction and the buffered instruction. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: mcBlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.
4: mcBlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.
5: mcBlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

The following table lists the buffer effects of MC_ MoveAbsolute.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_MoveAbsolute	YES	YES	*Done*

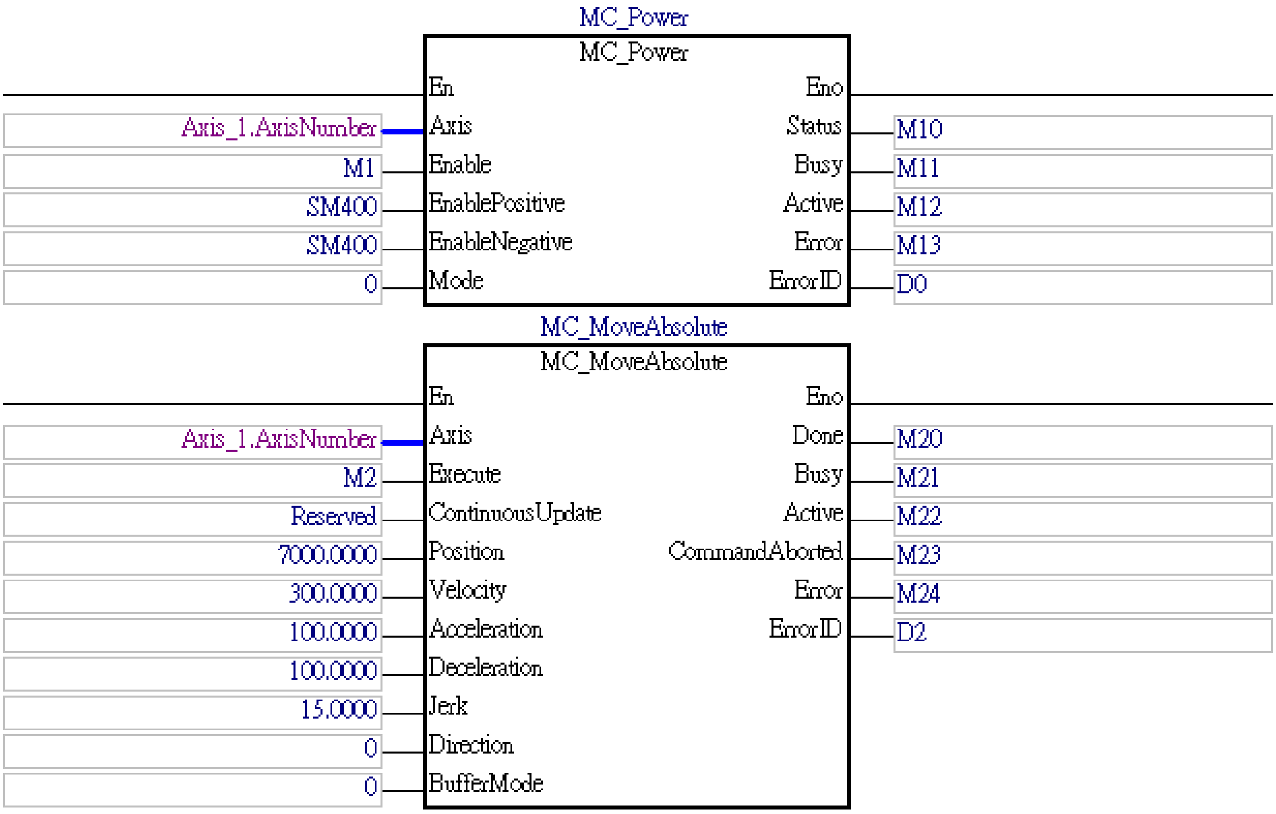
For more information of buffer mode, refer to **AH Motion Controller Motion Control Instructions Manual**.

Troubleshooting

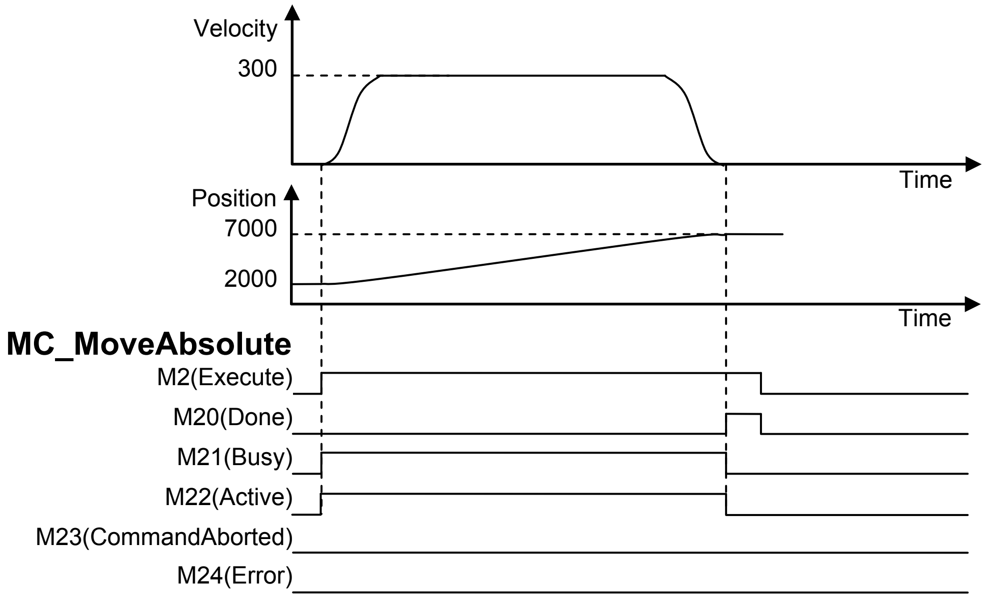
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

Programming Example 1

The example below describes the behavior of the MC_MoveAbsolute instruction.



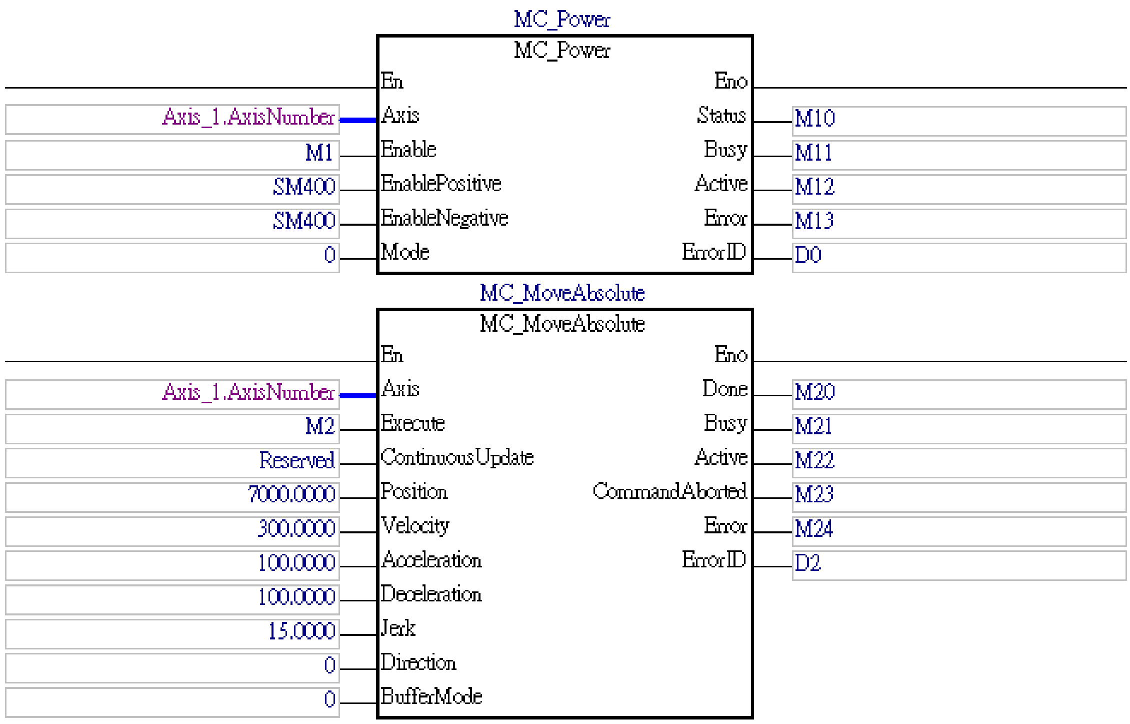
Motion diagram:

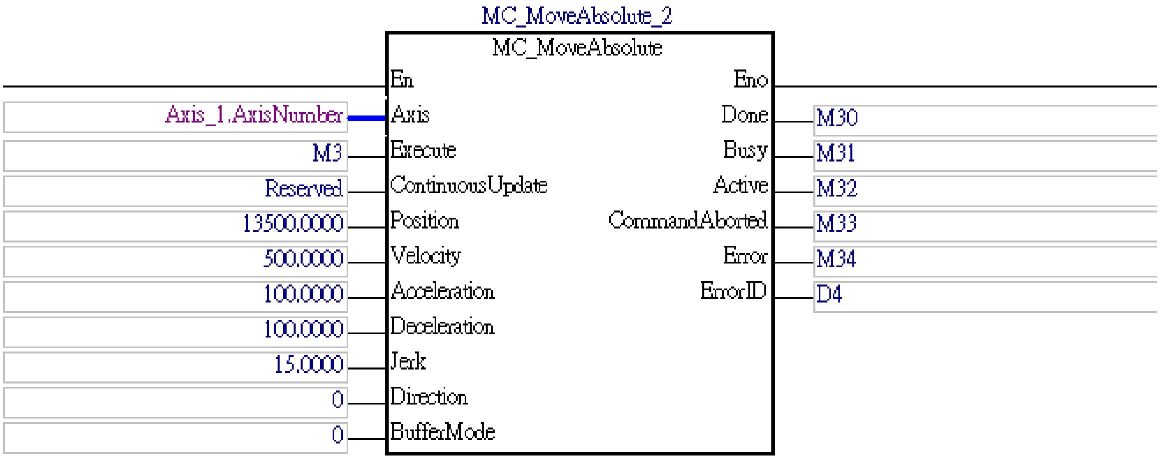


- When M2(*Execute*) changes to True, MC_MoveAbsolute drives the axis to the target position. When the axis reaches the specified target position, M20(*Done*) changes to True, and M21 and M22 change to False.
- When M2(Execute) changes to False, M20(Done) changes to False.
- When the axis reaches the target position, re-execution of the instruction will not move the axis.

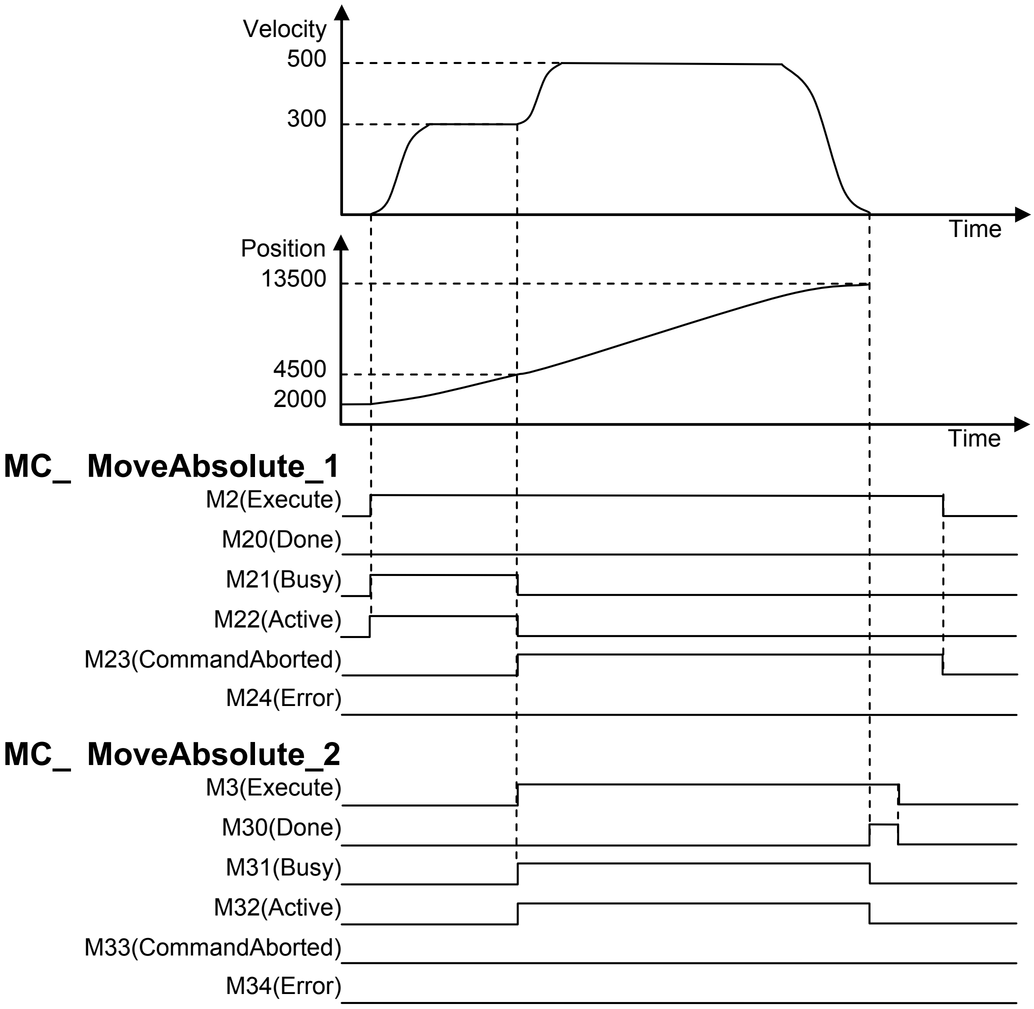
● Programming Example 2

The example below describes the behavior of 2 MC_MoveAbsolute instructions which are connected with each other.





Motion diagram:



- When M2(*Execute*) changes to True, MC_MoveAbsolute drives the axis to the target position. When M3(*Execute*) changes to True, the first MC_MoveAbsolute instruction is aborted and M23(*CommandAborted*) changes to True. Meanwhile, the second MC_MoveAbsolute instruction is executed and the axis will move according to the set parameters of the second MC_MoveAbsolute instruction.

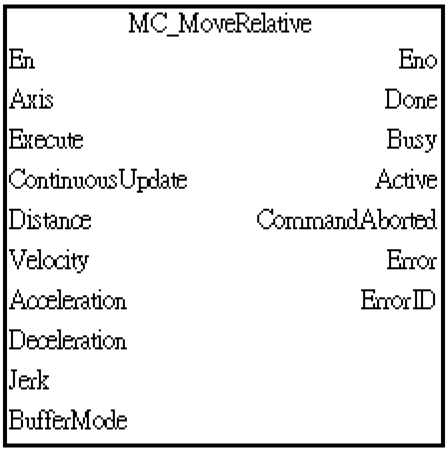
- When the axis reaches the specified target position of the second MC_MoveAbsolute instruction, M30(*Done*) changes to True, and M31(*Busy*) and M32(*Active*) change to False.
- When M3(*Execute*) changes to False, M30(*Done*) changes to False.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_MoveRelative

FB/FC	Description
FB	MC_MoveRelative controls the axis to move a specified relative distance with a specified behavior.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates *Velocity* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Distance	Relative distance to be moved. (Unit: user unit)	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

Outputs

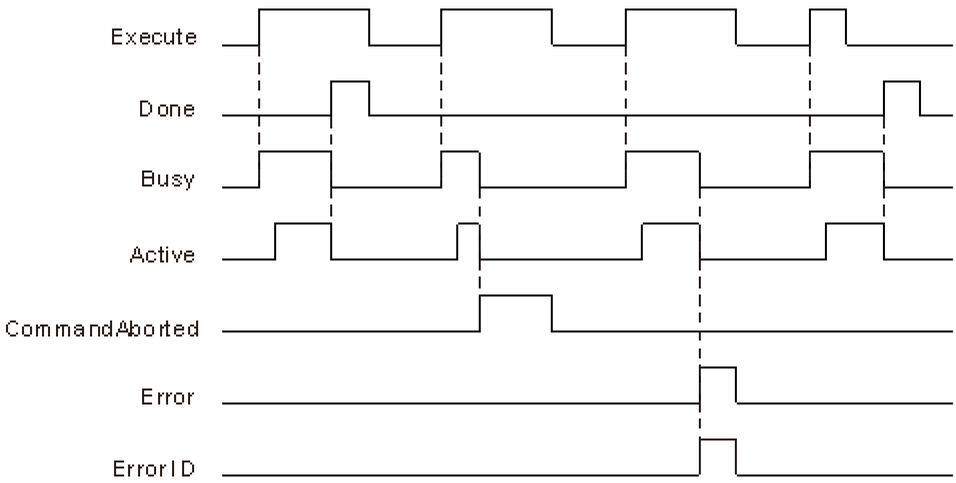
Name	Function	Data type	Output range (Default value)
Done	True when relative distance is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the relative positioning is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* changes to True. • When *Error* changes to True. • When *CommandAborted* shifts to True.
Active	• When the motion on the axis is started.	• When *Done* changes to True. • When *Error* changes to True. • When *CommandAborted* shifts to True. • If *Execute* is False and *Active* shifts to

Name	Timing for shifting to True	Timing for shifting to False
		True, it will be True for only one period and immediately shift to False.
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted because of the execution of MC_Stop instruction.	● When *Execute* changes to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When Execute shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

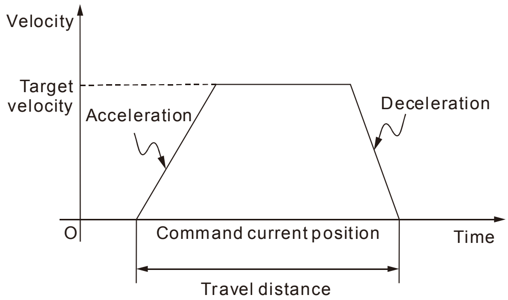
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

The instruction performs relative positioning with specified target velocity (*Velocity*), acceleration rate (*Acceleration*), deceleration rate (*Deceleration*) and Jerk value (*Jerk*) when execute changes to True.

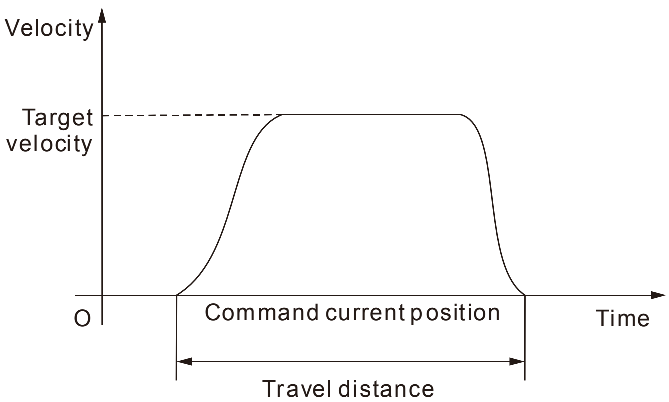
- The motion path of relative positioning is described as below.

Jerk=0



Jerk≠0

Setting up Jerk value allows you to control the motion path to ramp up (accelerate) or ramp down (decelerate) smoothly. The effects on a motion path with Jerk specified are as below.



BufferMode

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in “Standstill” state.

The following table lists the available buffer mode settings of MC_MoveRelative.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.
2: mcBlendingLow	Takes the lower target velocity as the transit velocity between the current instruction and the buffered instruction. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: mcBlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.

4: mcBlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.
5: mcBlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

The following table lists the buffer effects of MC_ MoveRelative.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_MoveRelative	YES	YES	*Done*

For more information of buffer mode, refer **Motion Controller Motion Control Instructions Manual.**

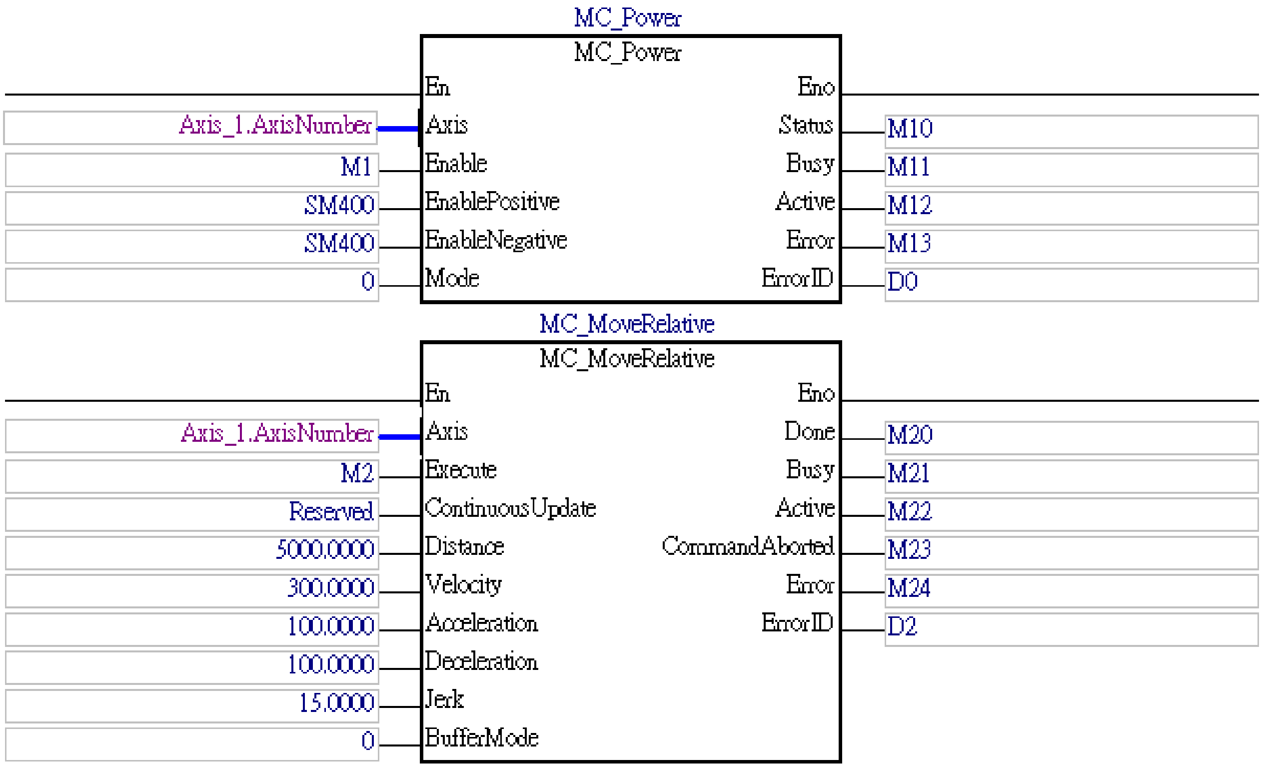
● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, Error will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding Error codes and Indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

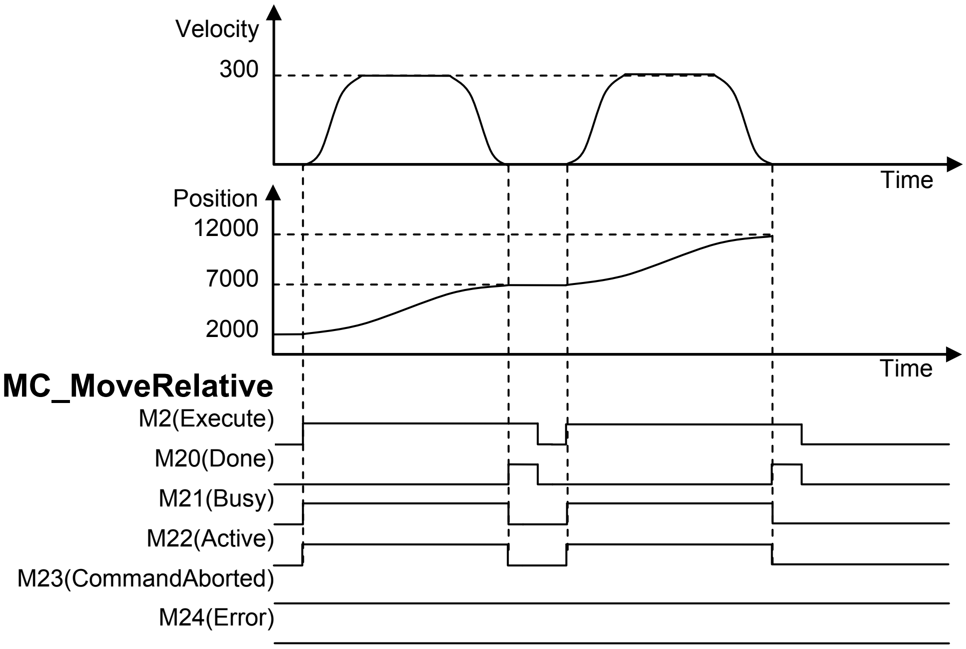
3

● Programming Example 1

The example below describes the behavior of the MC_MoveRelative instruction.



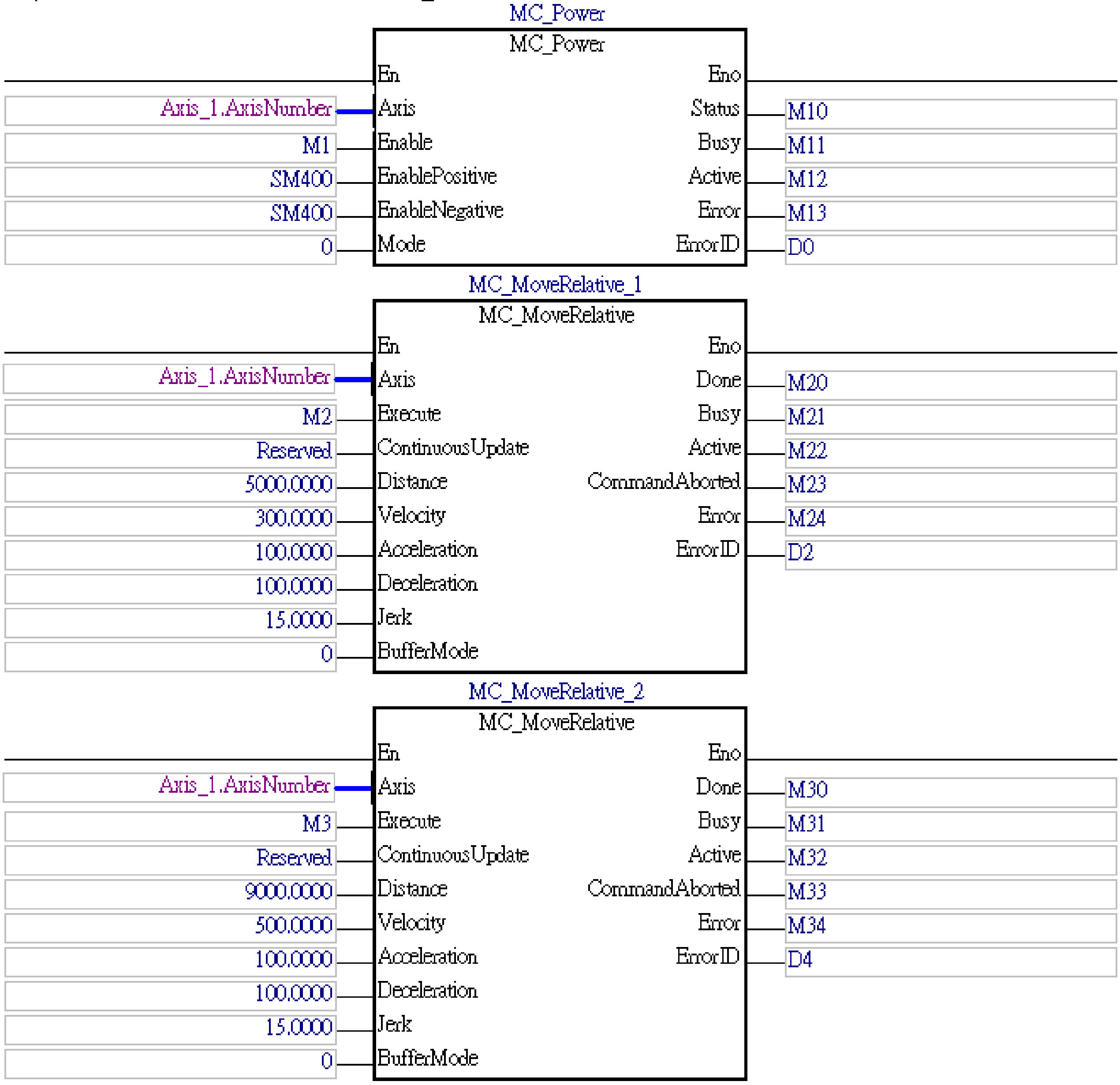
Motion diagram:



- When M2(*Execute*) changes to True, MC_MoveRelative drives the axis to the target position. When the axis moved the specified relative distance (5,000), M20(*Done*) changes to True, and M21 and M22 change to False.
- When M2(*Execute*) changes to False, M20(*Done*) changes to False.
- When the axis completed the specified relative distance(5,000) and then M2 changes to True again, the instruction will be executed again to move another distance(5,000) and reach the position of 12,000. When the specified distance is completed, M20(*Done*) changes to True again.

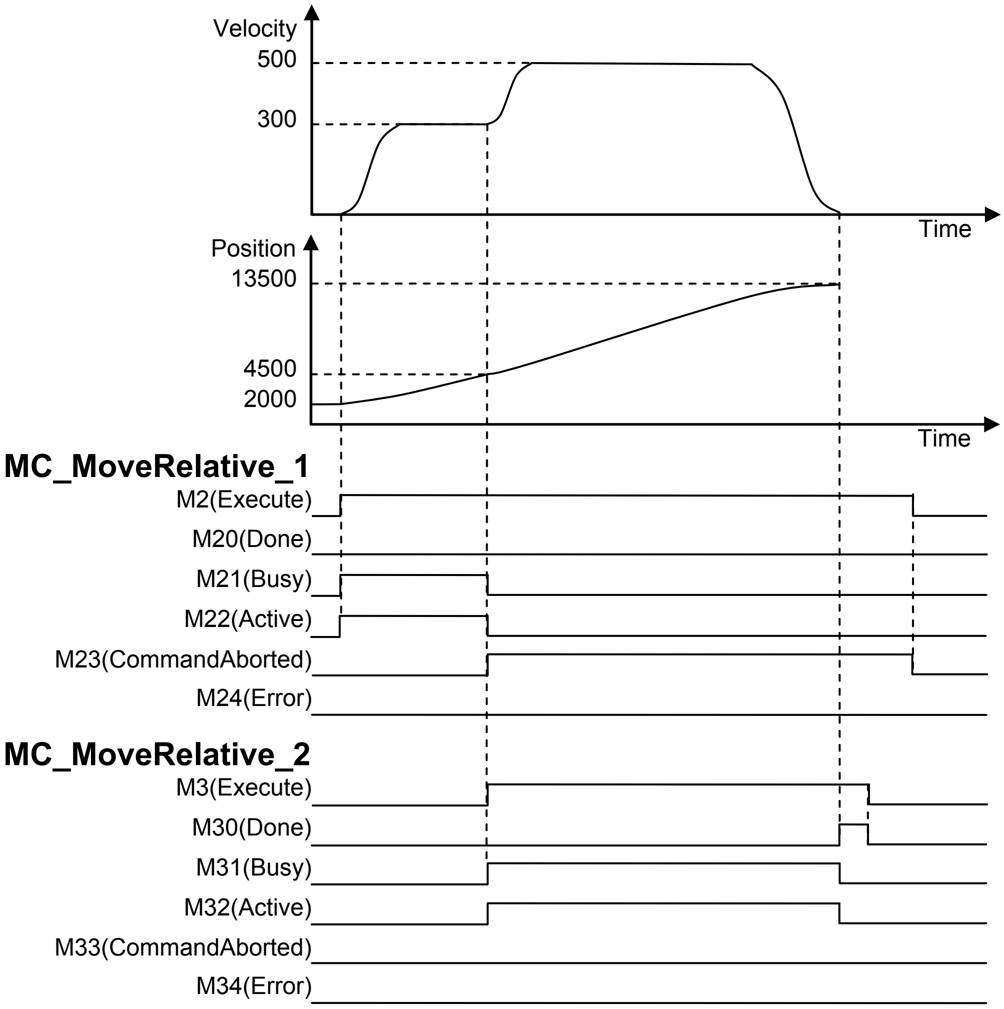
- **Programming Example 2**

The example below describes the behavior of 2 MC_MoveRelative instructions which are connected with each other.



3

Motion diagram:



3

- When M2(*Execute*) changes to True, MC_MoveRelative drives the axis to the target position. When M3(*Execute*) changes to True at the position 4,500, the first MC_MoveRelative instruction is aborted and M23(*CommandAborted*) changes to True. Meanwhile, the second MC_Moverelative instruction is executed and the axis will move for 9,000 according to the set parameters of the second MC_MoveRelative instruction.
- When the axis completed 9,000 which is specified by the second MC_MoveAbsolute instruction, the axis reaches 13,500 and M30(*Done*) changes to True.
- When M3(*Execute*) changes to False, M30(*Done*) changes to False.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_MoveAdditive

FB/FC	Description
FB	MC_MoveAdditive controls the axis to move an additional distance at a given speed and acceleration.

MC_MoveAdditive	
En	Eno
Axis	Done
Execute	Busy
ContinuousUpdate	Active
Distance	CommandAborted
Velocity	Error
Acceleration	ErrorID
Deceleration	
Jerk	
BufferMode	

- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates *Velocity* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Distance	Relative distance to be moved. (Unit: user unit)	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

Outputs

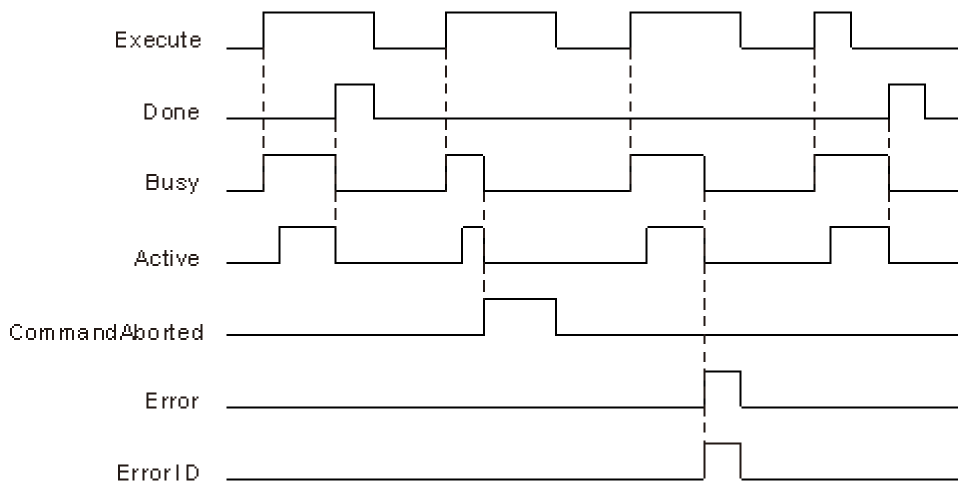
Name	Function	Data type	Output range (Default value)
Done	True when additive distance is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for changing to True	Timing for changing to False
Done	• True when the additive positioning is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• True when *Execute* changes to True.	• When *Done* changes to True. • When *Error* changes to True. • When *CommandAborted* changes to True.
Active	• True when the motion on the axis is started.	• When *Done* changes to True. • When *Error* changes to True • When *Commandaborted* shifts to True. • If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.

Name	Timing for changing to True	Timing for changing to False
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. ● When this instruction is aborted because of the execution of MC_Stop instruction.	● When *Execute* changes to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● In-Outs

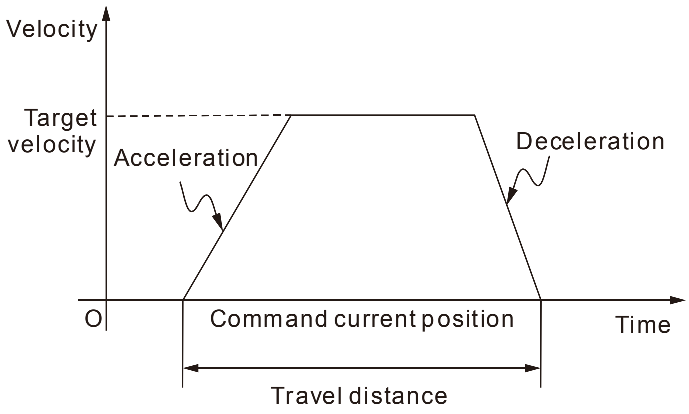
Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Function

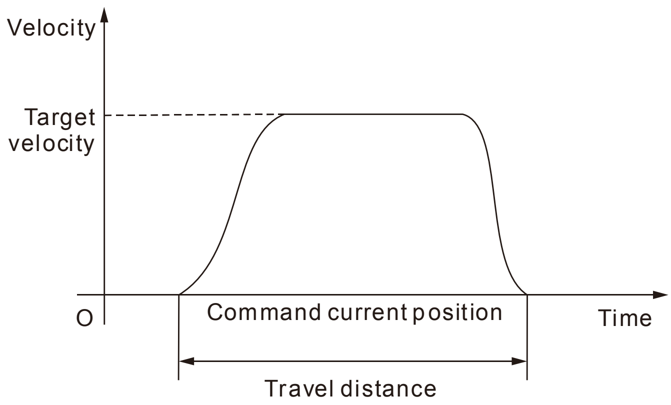
- *MC_MoveAdditive* executes relative positioning based on the target distance of previous positioning instruction, regardless of the completion of the previous positioning instruction.
- When the previous positioning instruction is on-going, executing MC_MoveAdditive will move the axis for the distance which is obtained by summing the previous target distance and the distance specified by MC_MoveReletive.
- If the previous instruction is a velocity instruction, MC_MoveAdditive will abort the execution of the velocity instruction and move the axis according to the given distance specified by MC_MoveAdditive at a given speed, acceleration and deceleration and then stop.
- The motion path of additive positioning is described as below.

Jerk=0



Jerk≠0

Setting up Jerk value allows you to control the motion path to ramp up (accelerate) or ramp down (decelerate) smoothly. The effects on a motion path with Jerk specified are as below.



- **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in “Standstill” state.

The following table lists the available buffer mode settings of MC_MoveAdditive.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.
2: mcBlendingLow	Takes the lower target velocity as the transit velocity between the current instruction and the buffered instruction. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: mcBlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.
4: mcBlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.

5: mcBlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

The following table lists the buffer effects of MC_ MoveAdditive.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_MoveAdditive	YES	YES	*Done*

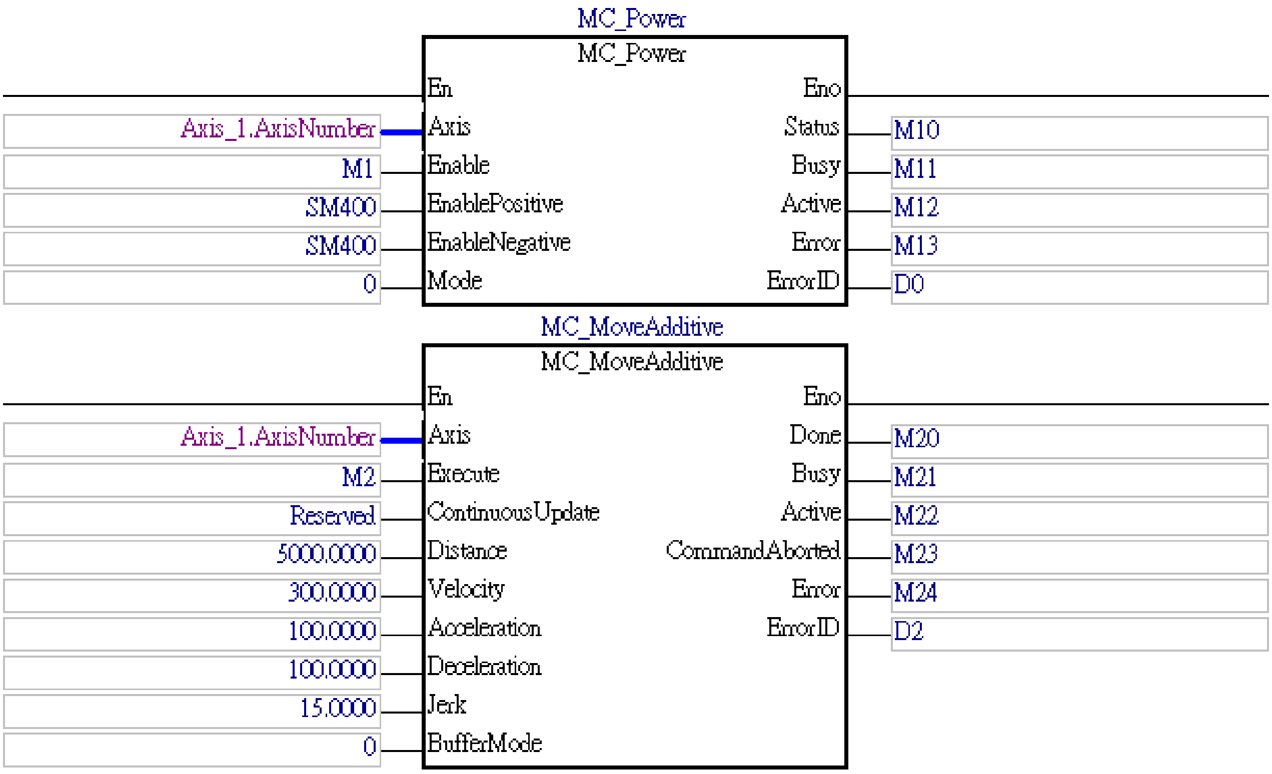
For more information of buffer mode, refer to section **Motion Controller Motion Control Instructions** Manual.

● Troubleshooting

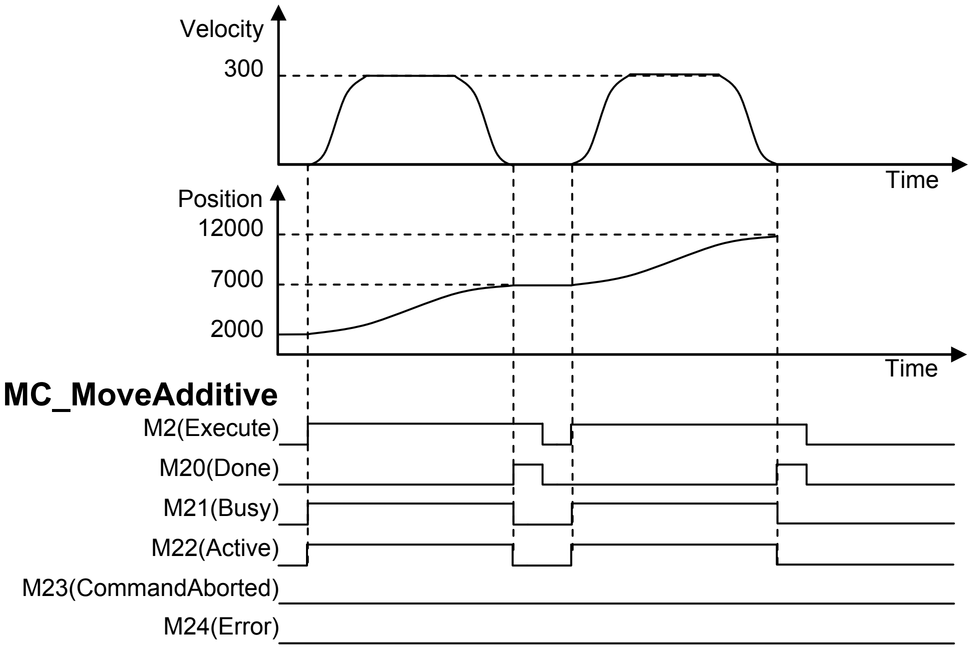
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", Error will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

● Programming Example 1

The example below describes the behavior of the MC_MoveAdditive instruction.



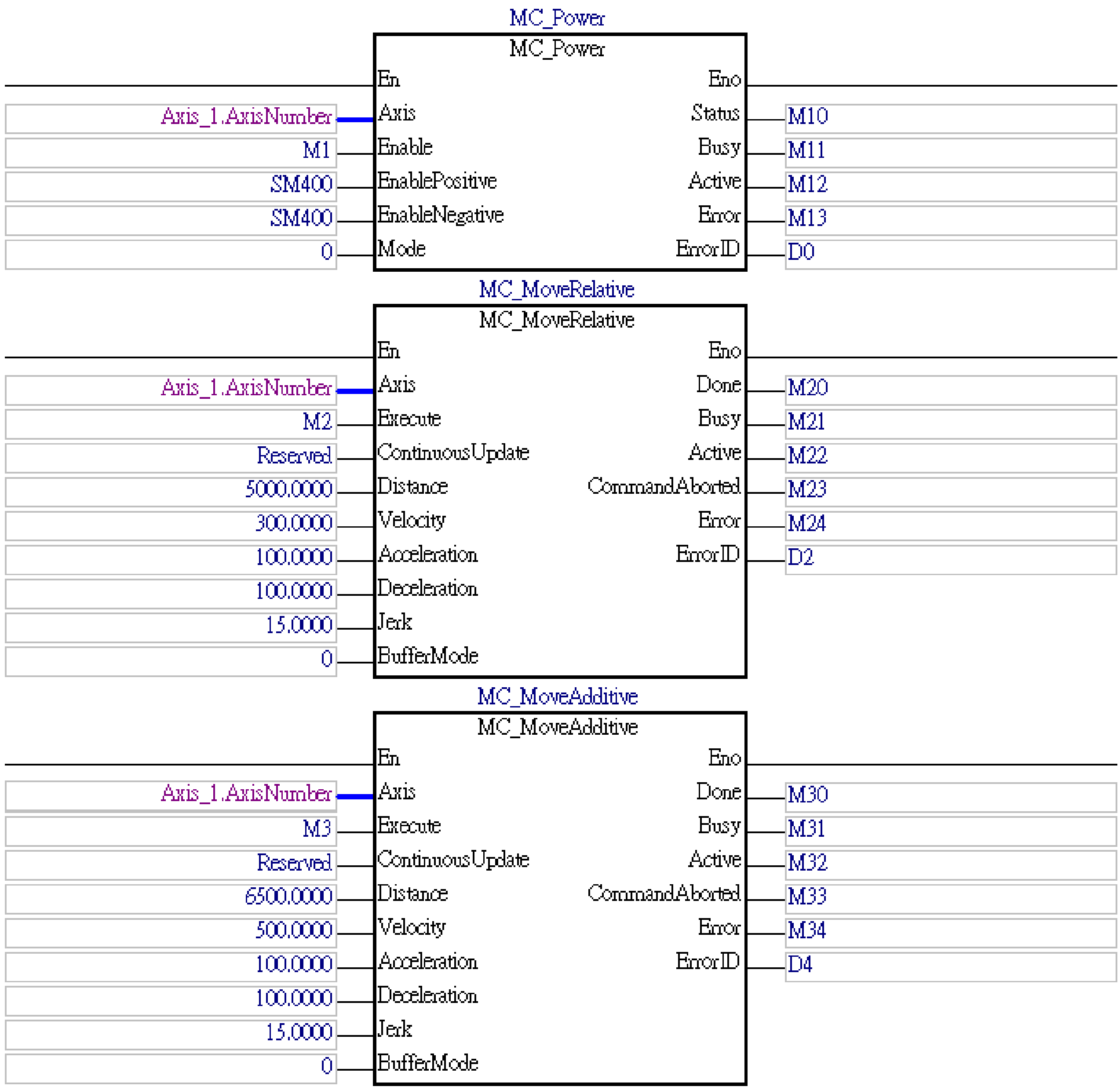
Motion diagram:



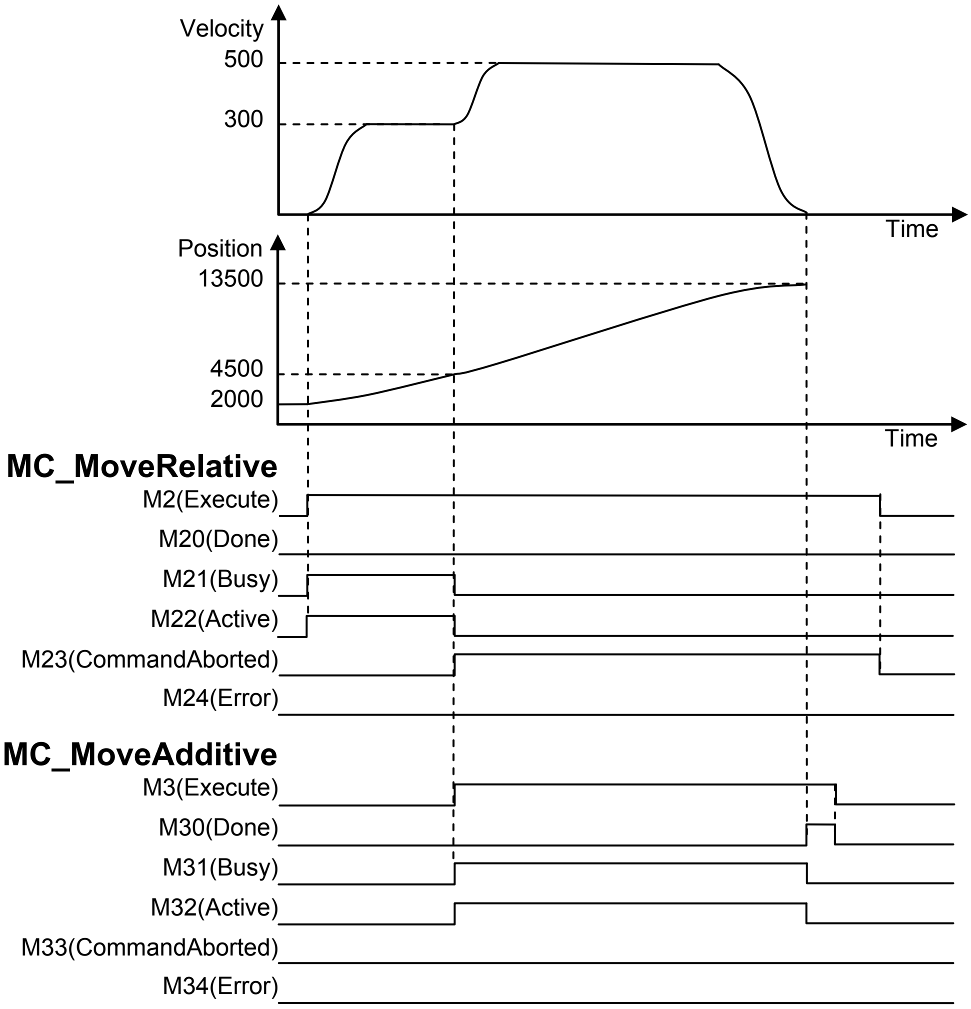
- When M2(*Execute*) changes to True, MC_MoveAdditive controls the axis to rotate. When the axis moved the specified relative distance (5,000), M20(*Done*) changes to True, and M21 and M22 change to False.
- When M2(*Execute*) changes to False, M20(*Done*) changes to False.
- When the axis completed the specified relative distance(5,000) and then M2 changes to True again, the instruction will be executed again to move another distance(5,000) and reach the position 12,000. When the specified distance is completed, M20(*Done*) changes to True again.

- **Programming Example 2**

The example below describes the behavior of MC_MoveRelative and MoveAdditive instructions which are executed in a series.



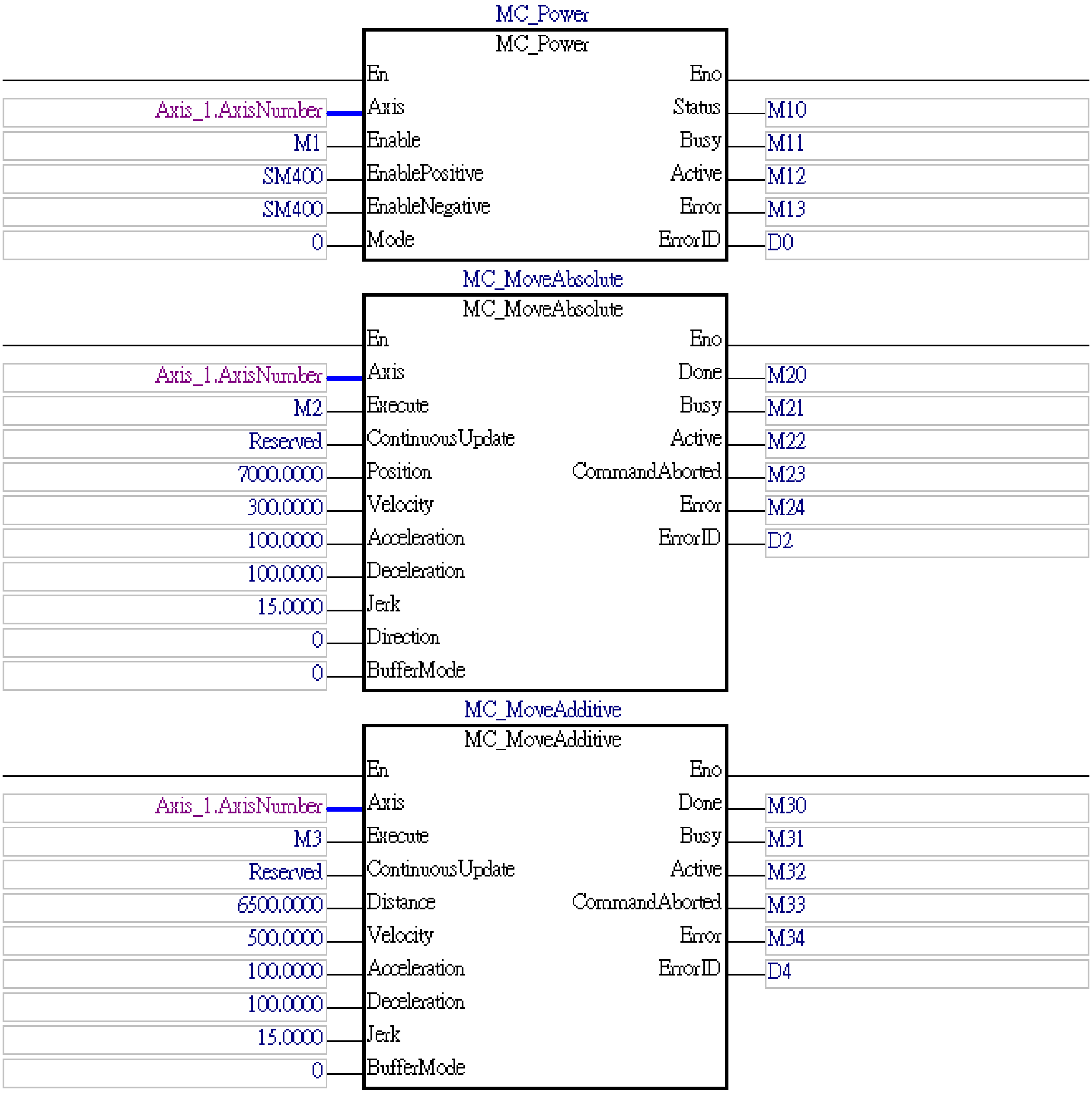
Motion diagram:



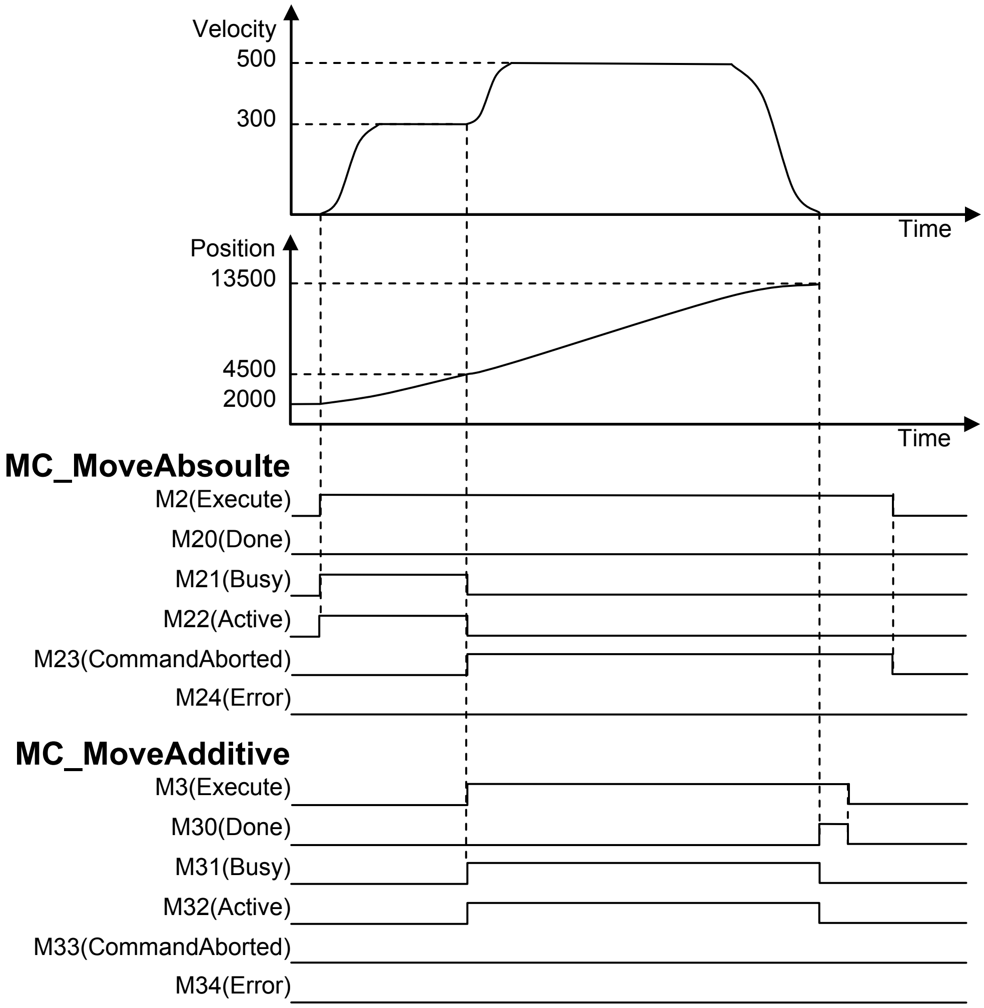
- When M2(*Execute*) changes to True, MC_MoveRelative drives the axis to the target position. A relative distance of 5000 is added to current position 2000, and the commanded position becomes 7000(2000+5000). When M3(*Execute*) changes to True at the position 4,500, the MC_MoveRelative instruction is aborted and M23(*CommandAborted*) changes to True.
- Meanwhile, the MC_MoveAdditive instruction is executed and adds a relative distance of 6500 to the previous commanded position 7000, and results the new commanded position 13500.
- When the axis reaches 13,500, M30(*Done*) changes to True.
- When M3(*Execute*) changes to False, M30(*Done*) changes to False.

- **Programming Example 3**

The example below describes the behavior of MC_MoveAbsolute and MC_MoveAdditive instructions which are executed in a series.



Motion diagram:



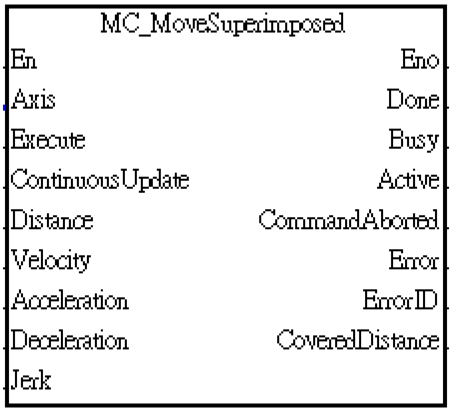
- When M2(*Execute*) changes to True, MC_MoveAbsoulte drives the axis to the commanded absolute position 7000. When M3(*Execute*) changes to True at the position 4,500, the MC_MoveAbsoulte instruction is aborted and M23(*CommandAborted*) changes to True.
- Meanwhile, MC_MoveAdditive instruction is executed and adds relative distance 6500 on the previous commanded position 7000, and results the new commanded position 13500.
- When the axis reaches 13500, M30(*Done*) changes to True.
- When M3(*Execute*) changes to False, M30(*Done*) changes to False.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_MoveSuperimposed

FB/FC	Description
FB	MC_MoveSuperimposed controls the axis to move a relative superimposed distance at a specified behavior while the axis is moving.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates *Velocity* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Distance	Additional relative distance to be moved. (Unit: user unit)	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	Additional target velocity (Unit: user unit/s)	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Additional acceleration rate (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Additional deceleration rate (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Additional jerk value (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Outputs

Name	Function	Data type	Output range (Default value)
Done	True when relative distance is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
Covered Distance	Continuously displays the covered distance moved by the instruction since it was executed.	LREAL	

3

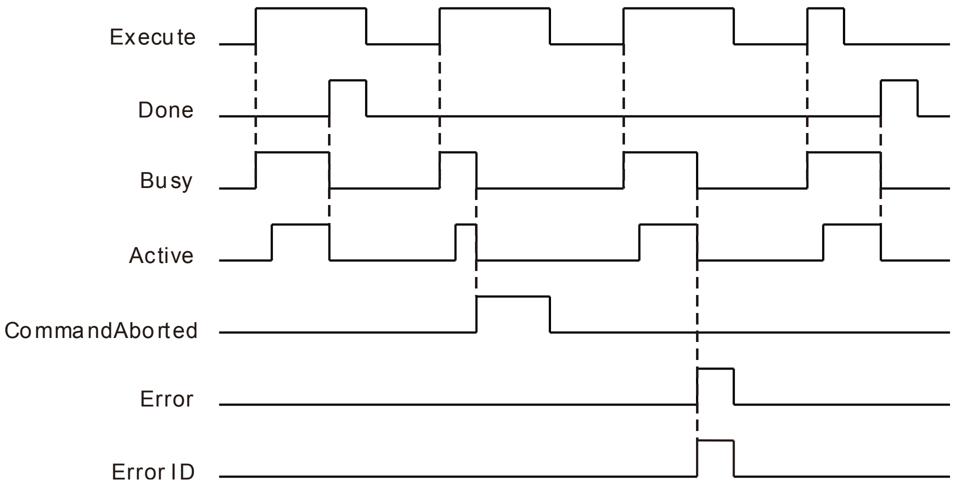
Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the superimposed distance is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* changes to True. • When *Error* changes to True. • When *CommandAborted* shifts to True.
Active	• When the motion on the axis is started.	• When *Done* changes to True. • When *Error* changes to True • When *Commandaborted* shifts to True • If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	• When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
CoveredDistance	• Continuously updates value when *Active* is True.	• Continuously updates value when *Active* is True.
Error/ErrorID	• When an error occurs in the execution	• When *Execute* shifts from True to False.

	conditions or input values for the instruction.(Error code is recorded)	(Error code is cleared)

■

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

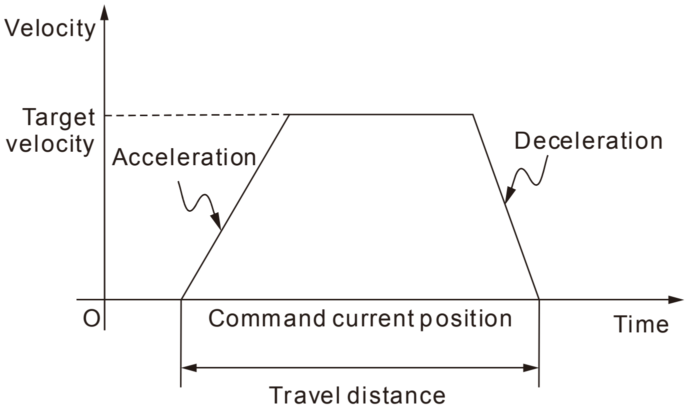
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Function

MC_MoveSuperimposed adds specified distance, acceleration, deceleration and jerk values to the axis in motion. When this instruction is executed, the previous instruction will not be aborted. The axis will react according to the superimposed values which are the sum of the previous instruction and MC_MoveSuperimposed instruction. When the superimposed distance is reached, the axis will resume the operation of the previous instruction until the superimposed total distance is reached.

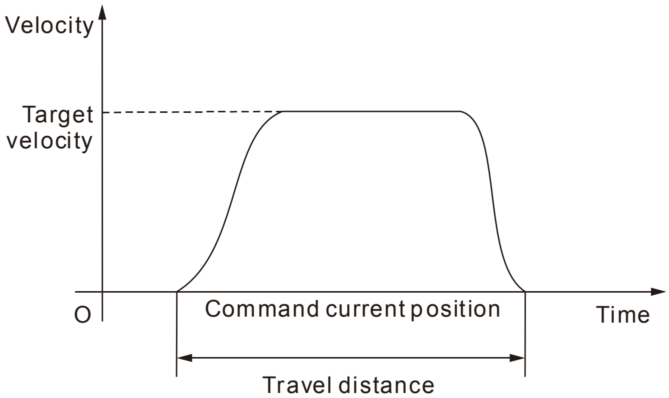
- If MC_MoveSuperimposed is active, any other instruction in aborting mode (except MC_MoveSuperimposed) will terminate both the previous instruction and MC_MoveSuperimposed instruction. In modes other than aborting mode, the previous motion instruction is not aborted.
- If MC_MoveSuperimposed is active and another MC_MoveSuperimposed is executed, only the active MC_MoveSuperimposed instruction is aborted, and the new MC_MoveSuperimposed will replace the active one.
- In all relevant states, MC_MoveSuperimposed changes the velocity and the moved distance of an ongoing motion.
- MC_MoveSuperimposed should be executed when the axis is in the “Synchronized” state.
- The input values of *Acceleration*, *Deceleration*, and *Jerk* are values added to the on-going motion. Therefore, the previous motion instruction always completes within the same period of time regardless of whether a MC_MoveSuperimposed instruction is executed at the same time.
- The motion path of the instruction and the effects of *Jerk* are described as below.

Jerk=0



Jerk≠0

Setting up Jerk value allows you to control the motion path to ramp up (accelerate) or ramp down (decelerate) smoothly. The effects on a motion path with Jerk specified are as below.

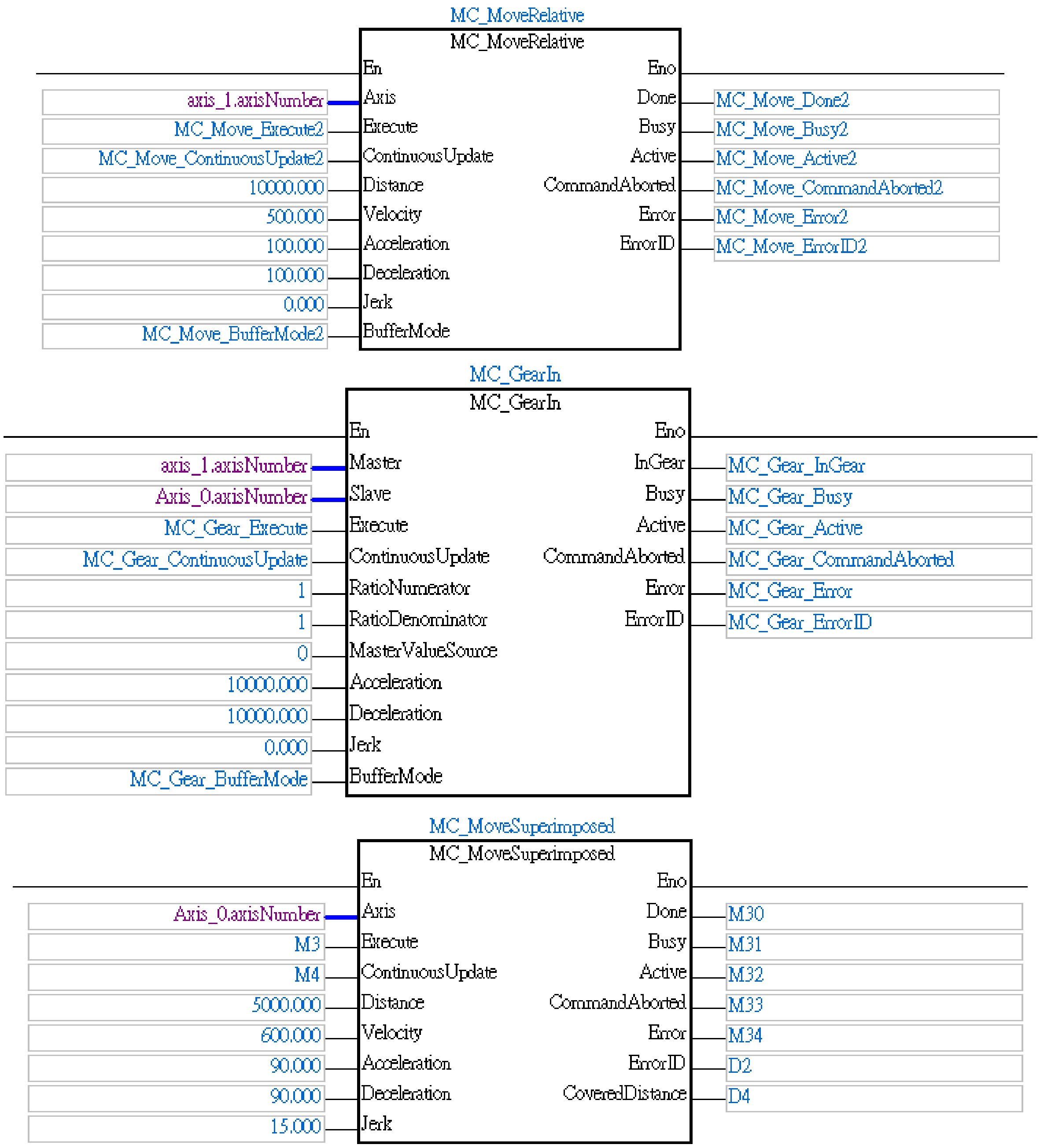


● Troubleshooting

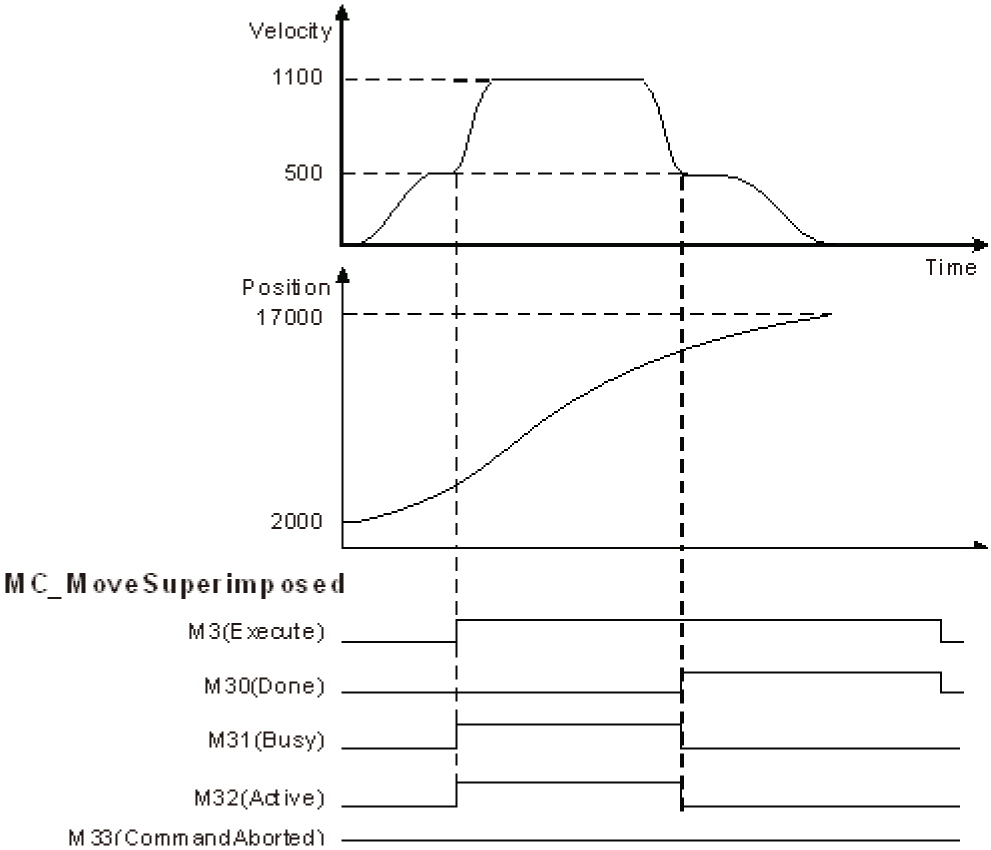
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Programming Example

1. When MC_Gear_InGear (InGear) is True, specifiy the parameters to be superimposed to the designated slave axis, and set M3(*Execute*) to True to execute MC_Superimposed.
2. The instruction will take current position as the reference point and superimpose an extra amount to the slave axis, and the motor will drive the axis to reach the desired position.
3. When the target distance is reached, M30(*Done*) will shift to True.



Motion diagram:



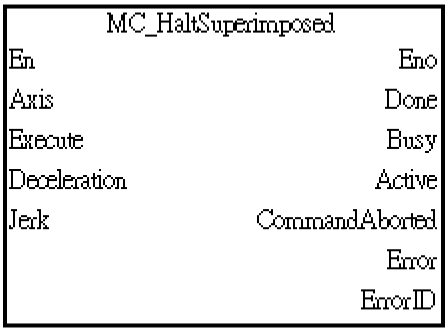
- When M3(*Execute*) changes to True, the MC_MoveSuperImposed instruction starts and applies the additional values(velocity, distance, acceleration, deceleration and jerk) to the axis and the axis performs a superimposed motion path.
- When the specified distance 5,000 of the MC_MoveSuperImposed instruction is completed, M30 changes to True and M31 and M32 changes to False. T
- When M3 changes to False, M30/M31 changes to False as well.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_HaltSuperimposed

FB/FC	Description
FB	MC_HaltSuperimposed halts all superimposed motions of the axis without aborting the previous superimposed motion.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

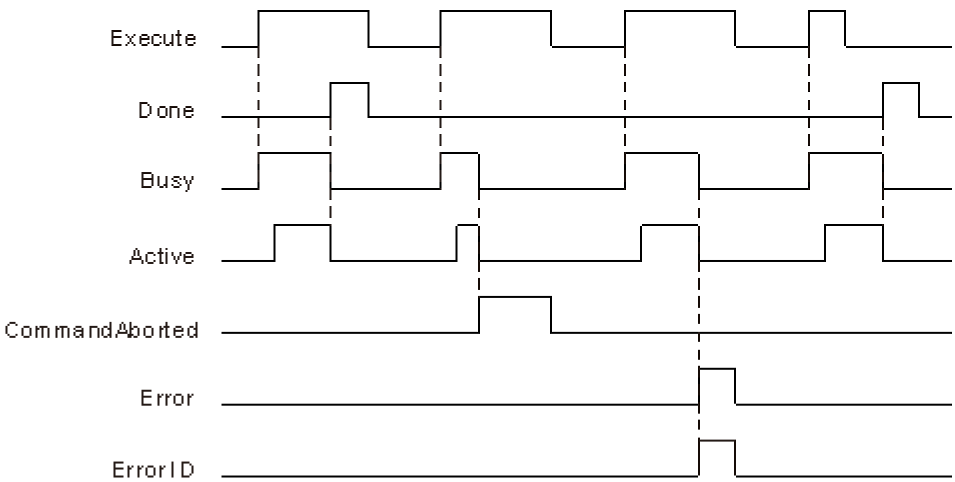
● Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the superimposing effects is cleared.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the superimposing effects is cleared	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* shifts to True and the instruction is executed.	• When *Done* shifts to True. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
Active	• When this instruction is started.	• When *Done* shifts to True. • When *Error* shifts to True. • When *CommandAborted* shifts to True. • If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	• When this instruction is aborted because another motion control instruction is executed.	• When *Execute* shifts from True to False. • If *Execute* is False and *CommandAborted shif*ts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

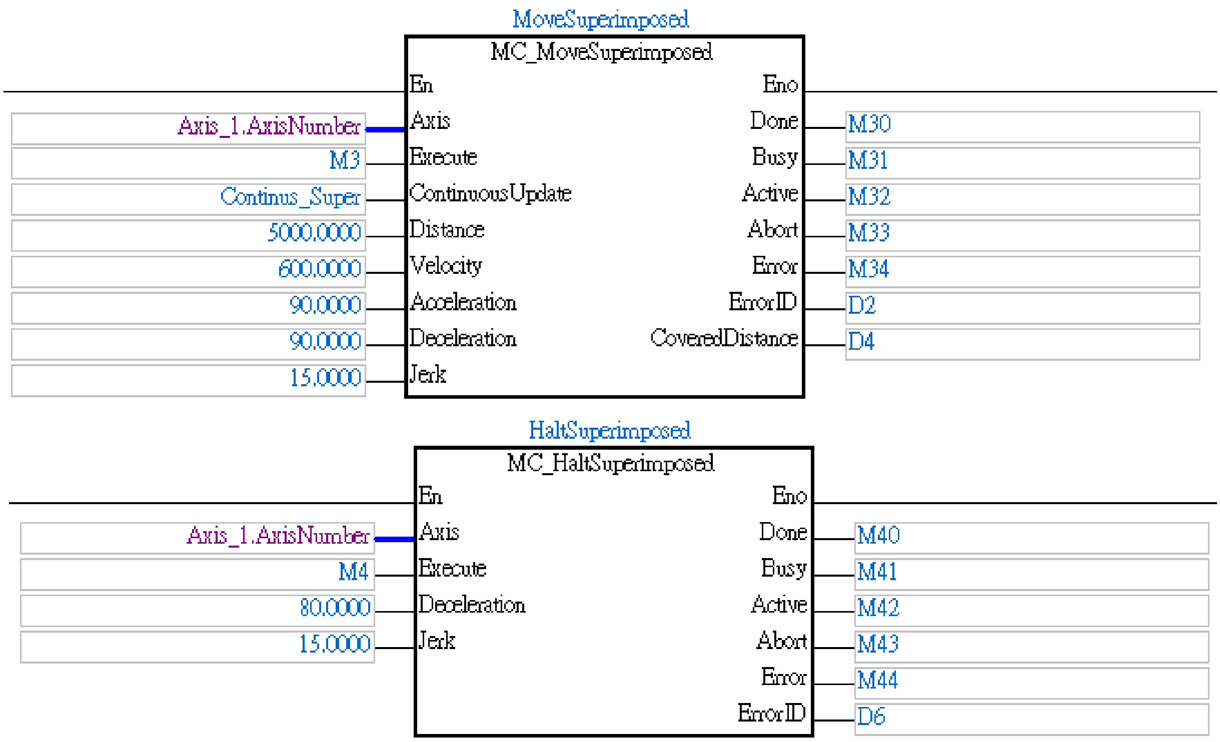
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

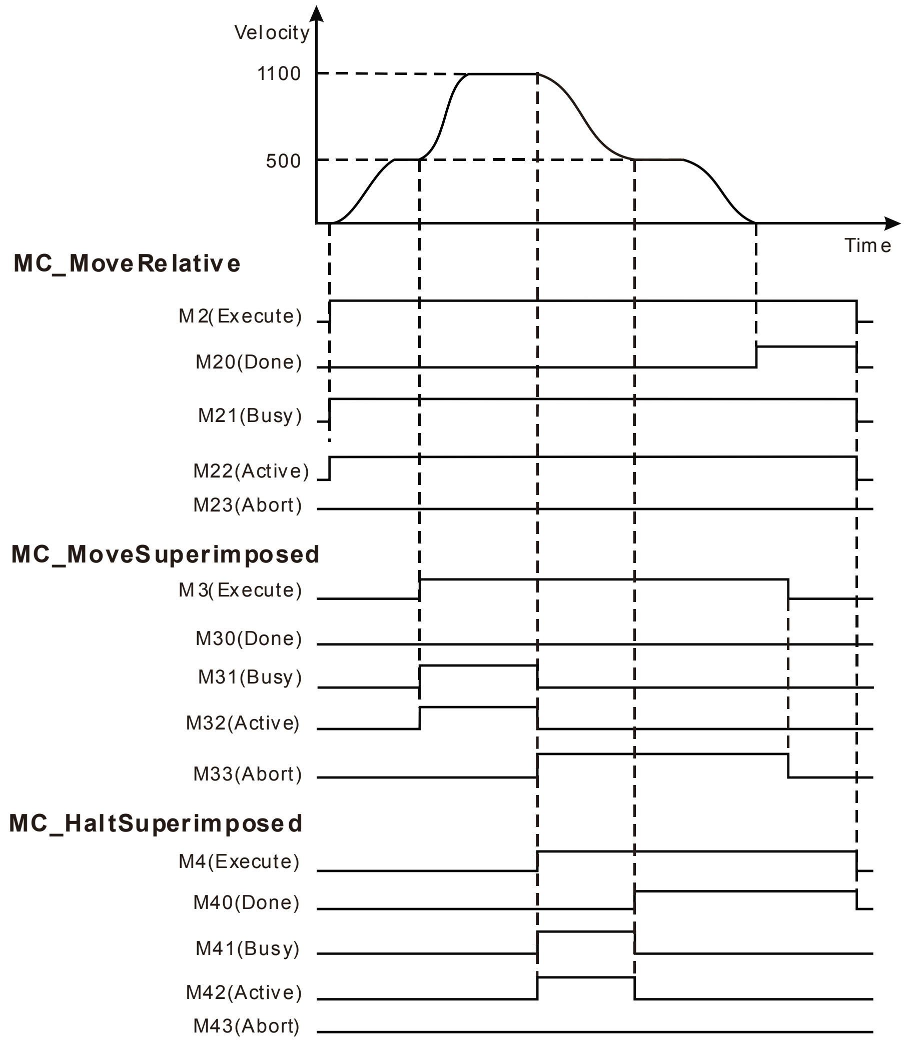
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Programming Example

The example below describes the behavior of MC_HaltSuperimposed in combination with MC_MoveSuperimposed.



Motion diagram:



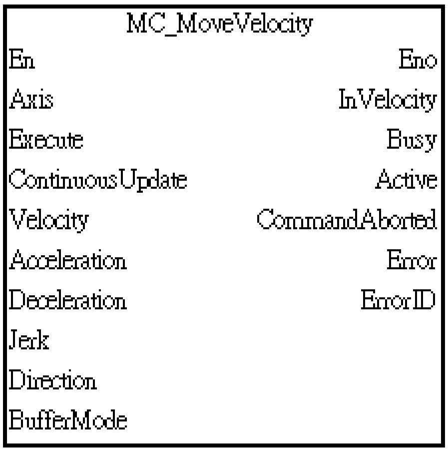
- When M4 (*Execute*) of MC_HaltSuperimposed changes to True, it triggers M33 (*CommandAboted*) of MC_MoveSuperimposed at the same time and the motion controller starts to decelerate the axis according to the values specified in MC_HaltSuperimposed.
- When the effects of MC_MoveSuperimposed are cleared and M40 (*Done*) will change to True.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_MoveVelocity

FB/FC	Description
FB	MC_MoveVelocity performs velocity control on an axis in the position mode with a specified behavior and an average velocity.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates *Velocity* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Direction	Specifies the direction for servo motor rotation. PositiveDirection, NegativeDirection, CurrentDirection	eMC_DIRECTION	1: mcPositiveDirection 3: mcNegativeDirection 4: mcCurrentDirection (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

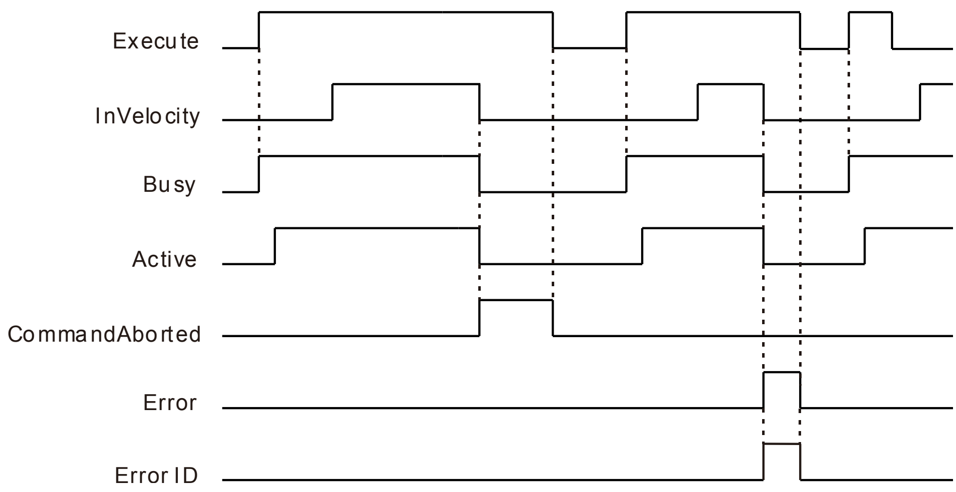
● Outputs

Name	Function	Data type	Output range (Default value)
InVelocity	True when the specified target velocity is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
InVelocity	● When the specified target velocity is reached.	● When *CommandAborted* shifts to True ● When *CommandAborted* shifts to True and the target velocity is changed.
Busy	● When *Execute* shifts to True.	● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● When the motion on the axis is started.	● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted because of the execution of MC_Stop instruction.	● When *Execute* changes to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

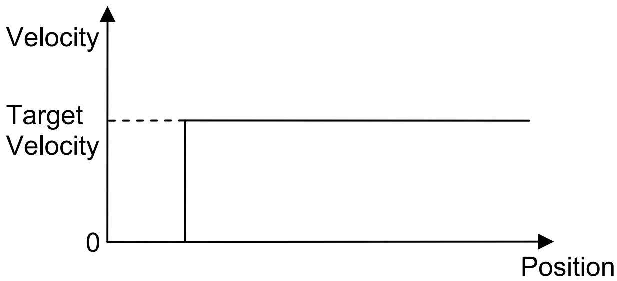
● Function

The instruction performs speed control with specified target velocity (*Velocity*), acceleration rate (*Acceleration*), deceleration rate (*Deceleration*) and Jerk value (*Jerk*) when execute changes to True.

- You can execute another motion instruction to abort the ongoing motion of MC_MoveVelocity.
- The output *InVelocity* is reset when the instruction is aborted by another instruction.
- When the instruction is used with MC_MoveSuperimposed, the output *InVelocity* will be True as long as the specified target velocity equals to the performed actual velocity.
- The motion path of different velocity parameters is described as below.

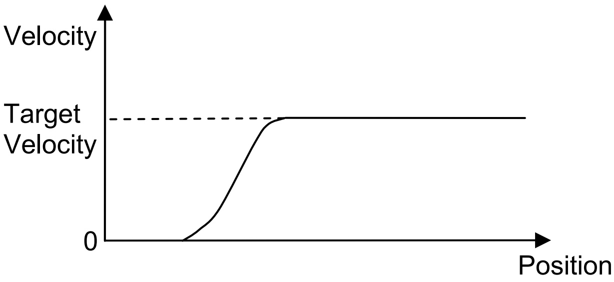
***Acceleration*/*Deceleration*=0**

When the instruction is executed with *Acceleration* or *Deceleration* set to 0, no accelerating or decelerating will be performed on the axis before it reachs the specified target velocity.

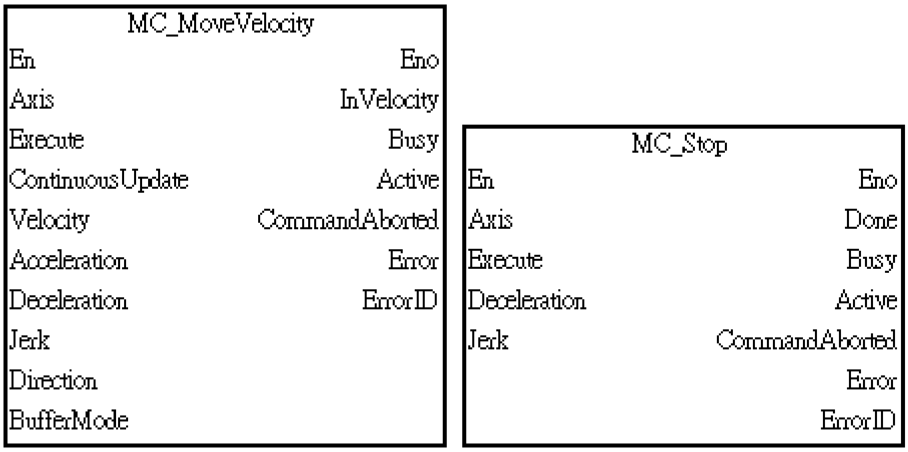


***Acceleration*/*Deceleration*/*Jerk*≠0**

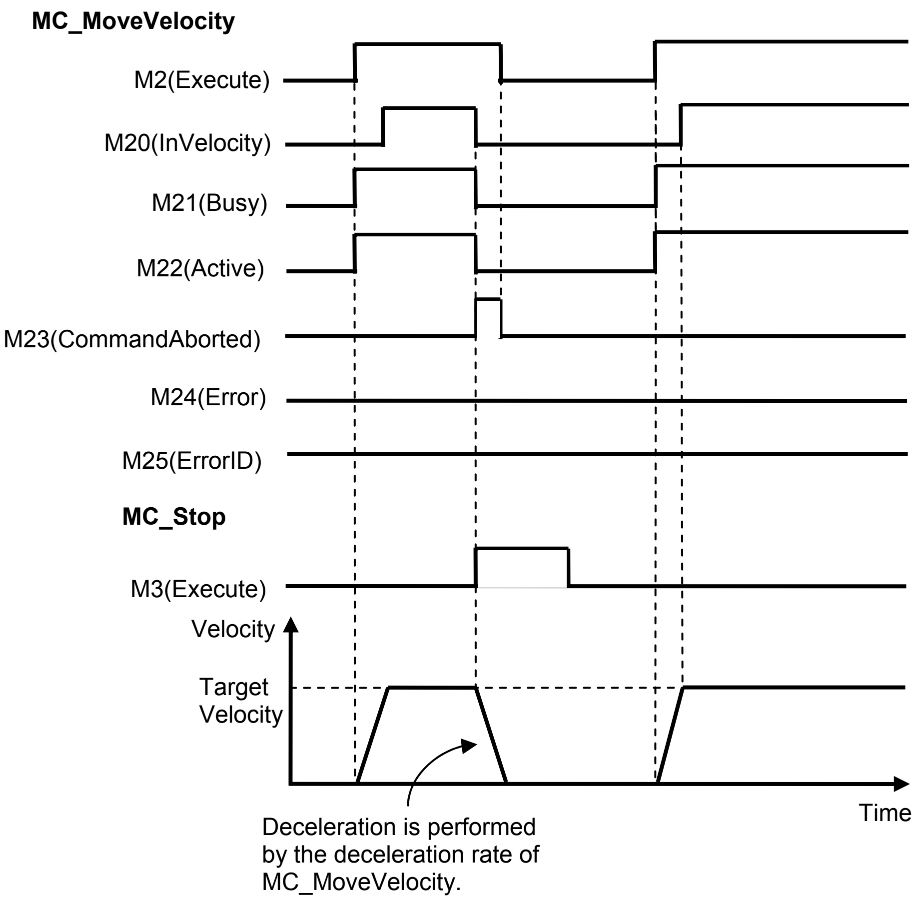
Setting up *Acceleration/Deceleration/Jerk* value allows you to control the motion path to ramp up (accelerate) or ramp down (decelerate) smoothly. The effects on a motion path with Jerk specified are as below.



- The behavior of the MC_MoveVelocity instruction followed by MC_Stop instruction is described as below.



Motion diagram:



- When M2(*Execute*) changes to True, MC_MoveVelocity drives the axis to the target velocity. When the axis reaches the specified target velocity, M20(*InVelocity*) changes to True.
- When M3(*Execute*) changes to True, MC_Stop aborts MC_MoveVelocity and the axis decelerates according to the deceleration rate specified by MC_MoveVelocity.
- If the MC_MoveVelocity is not aborted, it will stay at the specified target velocity, and M20(*InVelocity)* remains True.

- **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in "Standstill" state.

The following table lists the available buffer mode settings of MC_MoveVelocity.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

The following table lists the buffer effects of MC_ MoveVelocity.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_MoveVelocity	YES	YES	*InVelocity*

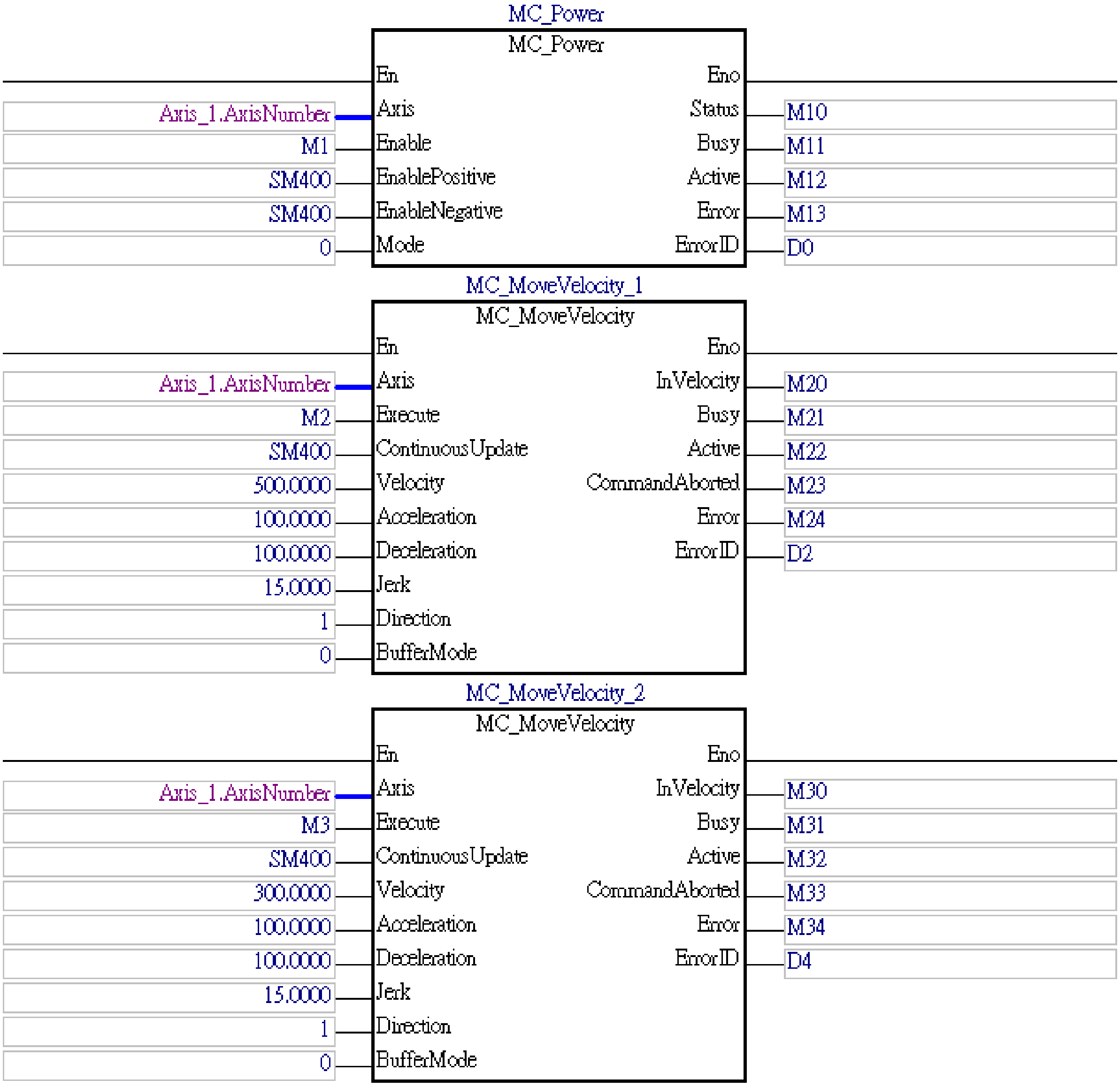
For more information of buffer mode, refer to section **Motion Controller Motion Control Instructions Manual.**

Troubleshooting

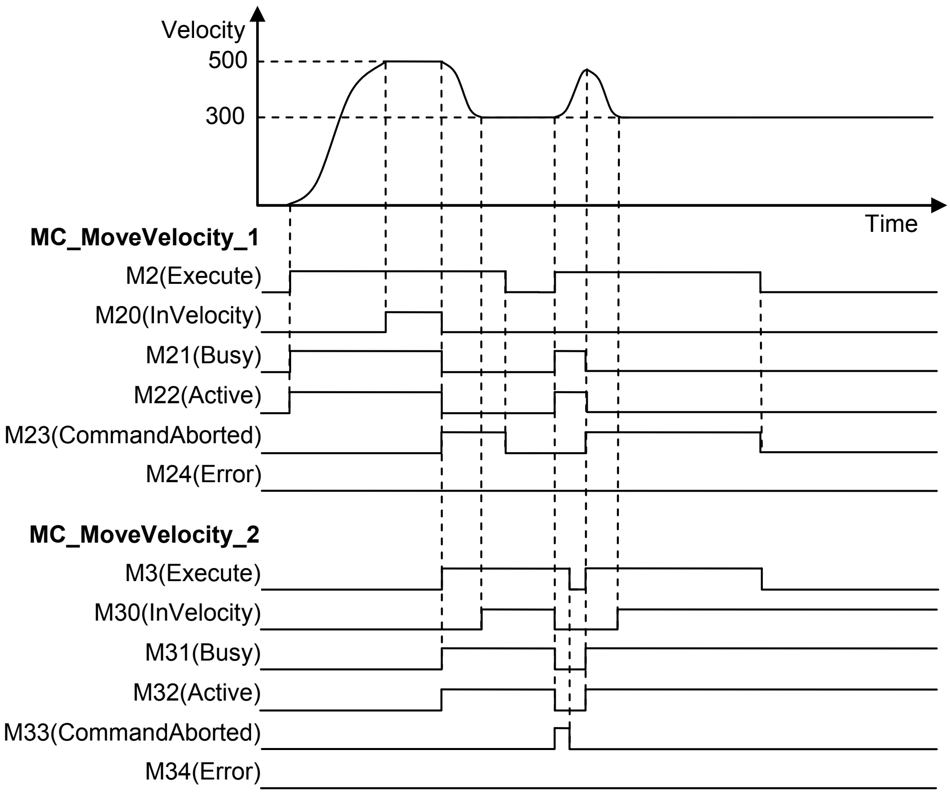
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

- ## Programming Example

The examples of the combination of two MC_MoveVelocity instructions are described as below.



Motion diagram:



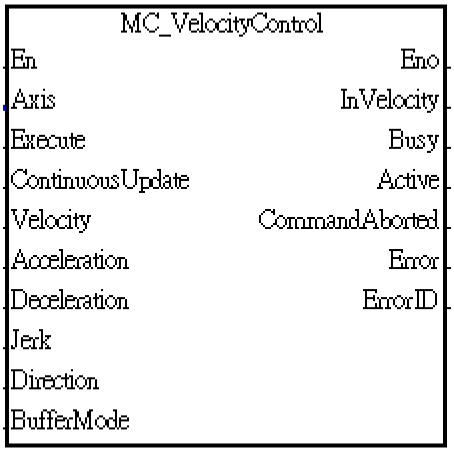
- When M2(*Execute*) changes to True, the first MC_MoveVelocity controls the axis to reach the specified target velocity 500. When it reaches 500, M20(*InVelocity*) changes to True.
- If the M3(*Execute*) changes to True, M20(*InVelocity*) will change to False and M21(*CommanAborted*) will change to True while the second MC_MoveVelocity is executed.
- The second MC_MoveVelocity will decelerate the axis to the velocity 300. When 300 is reached, M30(*InVelocity*) will change to True and remain in this status as long as the velocity is not changed.
- When M2(*Execute*) changes to False, M21(*CommanAborted*) will change to False.
- If the first MC_MoveVelocity is started again by M2(*Execute*) changes to True, the axis will abort the second MC_MoveVelocity and accelerate toward the velocity 500.
- If the sencod MC_MoveVelocity M3(*Execute*) changes to False and and set on again, the sencod instrution aborts the first MC_MoveVelocity. In this case, the axis decelerates again without reaching the target velocity 500 of the first MC_MoveVelocity.
- If MC_MoveVelocity M3(*Execute*) changes to False after reaching the target velocity. M30(*InVelocity*), M31(*Busy*) and M32(*Active*) remain True.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_VelocityControl

FB/FC	Description
FB	MC_MoveVelocity performs velocity control on an axis in the velocity mode with a specified behavior and an average velocity.



Before using MC_VelocityControl instruction, it is required to confirm if the below object data is supported, and use ECAT Builder in ISPSoft to set the below data in the settings of PDO communications.

- Target velocity (60FF hex)
- Modes of operation (6060 hex)
- Modes of operation display (6061 hex)

If one of the above required object data is not set, an error will occur to indicate the problem of missing process data object setting.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates *Velocity* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Velocity	Target velocity. (Unit: user unit/s)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
Direction	Specifies the direction for servo motor rotation. PositiveDirection, NegativeDirection, CurrentDirection	eMC_DIRECTION	1: mcPositiveDirection 3: mcNegativeDirection 4: mcCurrentDirection (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

● Outputs

3

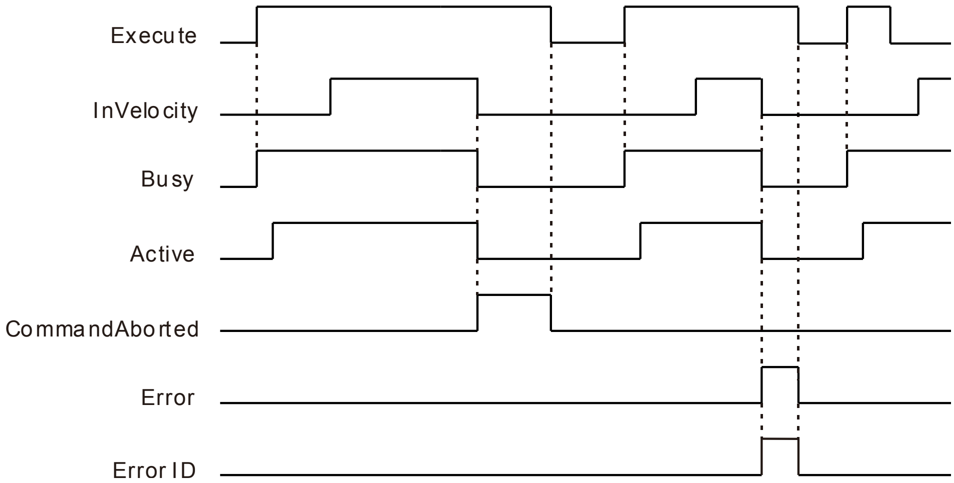
Name	Function	Data type	Output range (Default value)
InVelocity	True when the specified target velocity is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
InVelocity	● When the specified target velocity is reached.	● When *CommandAborted* shifts to True ● When *CommandAborted* shifts to True and the target velocity is changed.
Busy	● When *Execute* shifts to True.	● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● When the motion on the axis is started.	● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted	● When *Execute* changes to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and

Name	Timing for shifting to True	Timing for shifting to False
	because of the execution of MC_Stop instruction.	immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

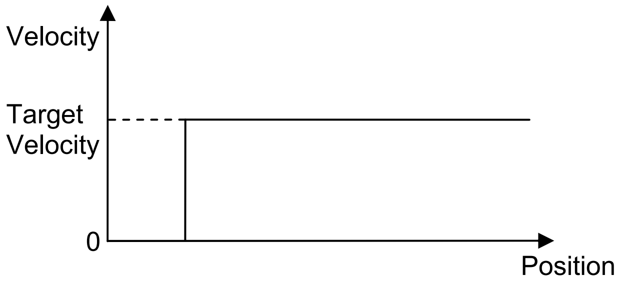
● Function

The instruction performs speed control with specified target velocity (*Velocity*), acceleration rate (*Acceleration*), deceleration rate (*Deceleration*) and Jerk value (*Jerk*) when execute changes to True.

- You can execute another motion instruction to abort the ongoing motion of MC_VelocityControl.
- The output InVelocity is reset when the instruction is aborted by another instruction.
- During the execution of the instruction MC_VelocityControl in CSV control mode, if users execute MC_Stop, MC_Halt or the axis is in the state of Errorstop, when the target velocity (60FF hex) decelerates to 0, the mode will be switched to CSP control mode. Refer to its servo manual for more details on the actual servo behavior.
- The motion path of different velocity parameters is described as below.

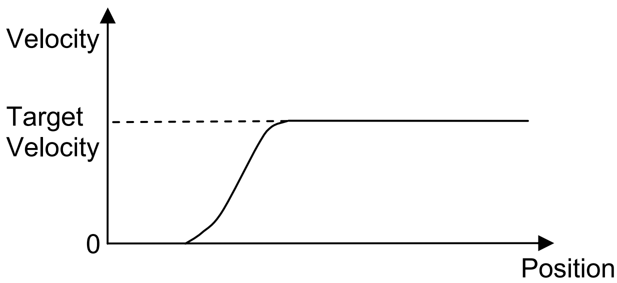
  ***Acceleration*/*Deceleration*=0**

  When the instruction is executed with *Acceleration* or *Deceleration* set to 0, no accelerating or decelerating will be performed on the axis before it reachs the specified target velocity.

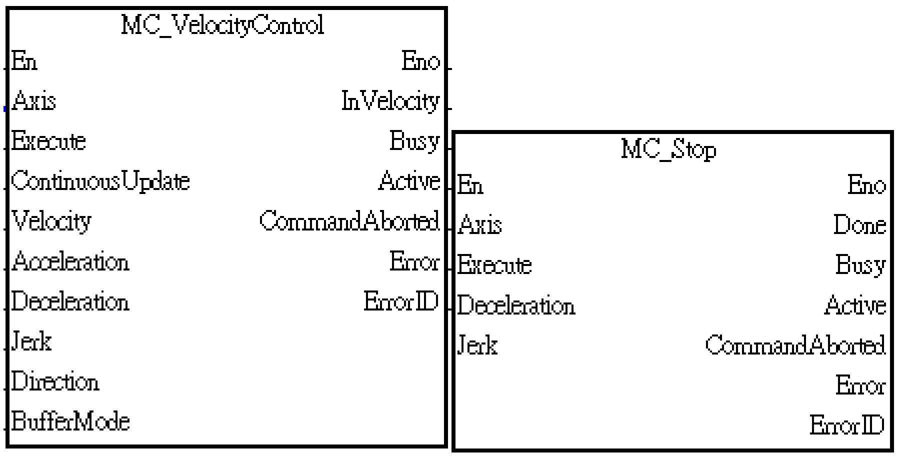


  ***Acceleration*/*Deceleration*/*Jerk*≠0**

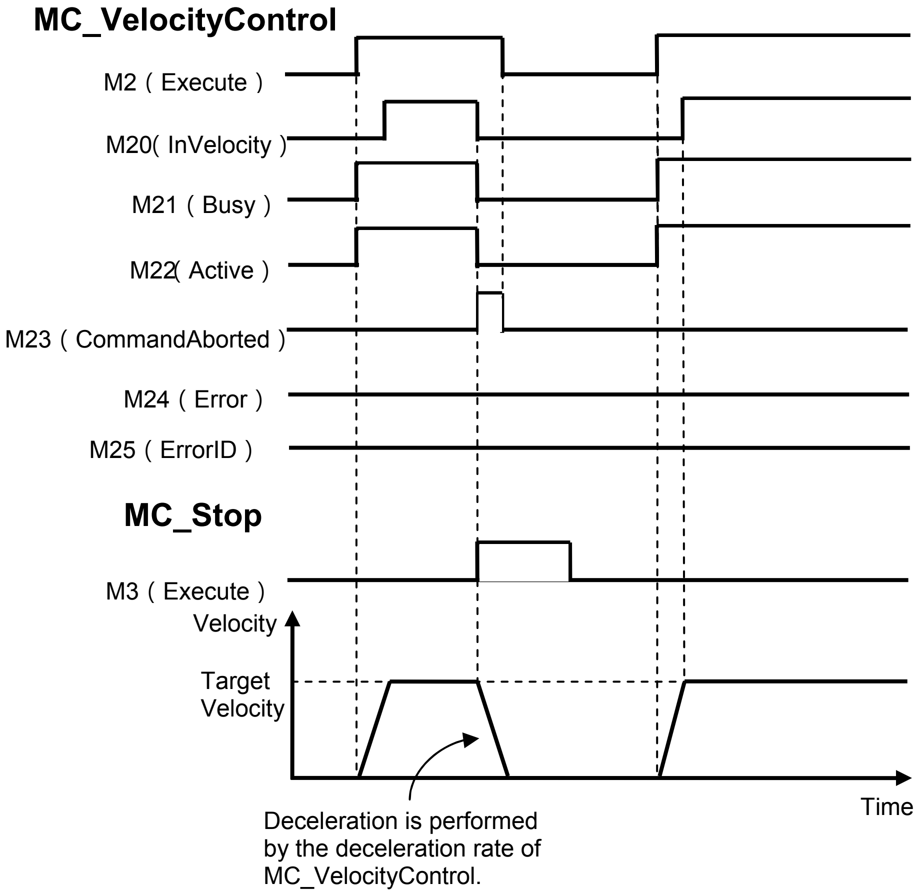
  Setting up *Acceleration/Deceleration/Jerk* value allows you to control the motion path to ramp up (accelerate) or ramp down (decelerate) smoothly. The effects on a motion path with Jerk specified are as below.



- The behavior of the MC_VelocityControl instruction followed by MC_Stop instruction is described as below.



Motion diagram:



- When M2(*Execute*) changes to True, MC_VelocityControl drives the axis to the target velocity. When the axis reaches the specified target velocity, M20(*InVelocity*) changes to True.
- When M3(*Execute*) changes to True, MC_Stop aborts MC_MoveVelocity and the axis decelerates according to the deceleration rate specified by MC_VelocityControl.
- If the MC_VelocityControl is not aborted, it will stay at the specified target velocity, and M20(*InVelocity)* remains True.

- **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in “Standstill” state.

The following table lists the available buffer mode settings of MC_VelocityControl.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

The following table lists the buffer effects of MC_VelocityControl.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_VelocityControl	YES	YES	*InVelocity*

For more information of buffer mode, refer to ***AH Motion Controller – Operation Manual.***

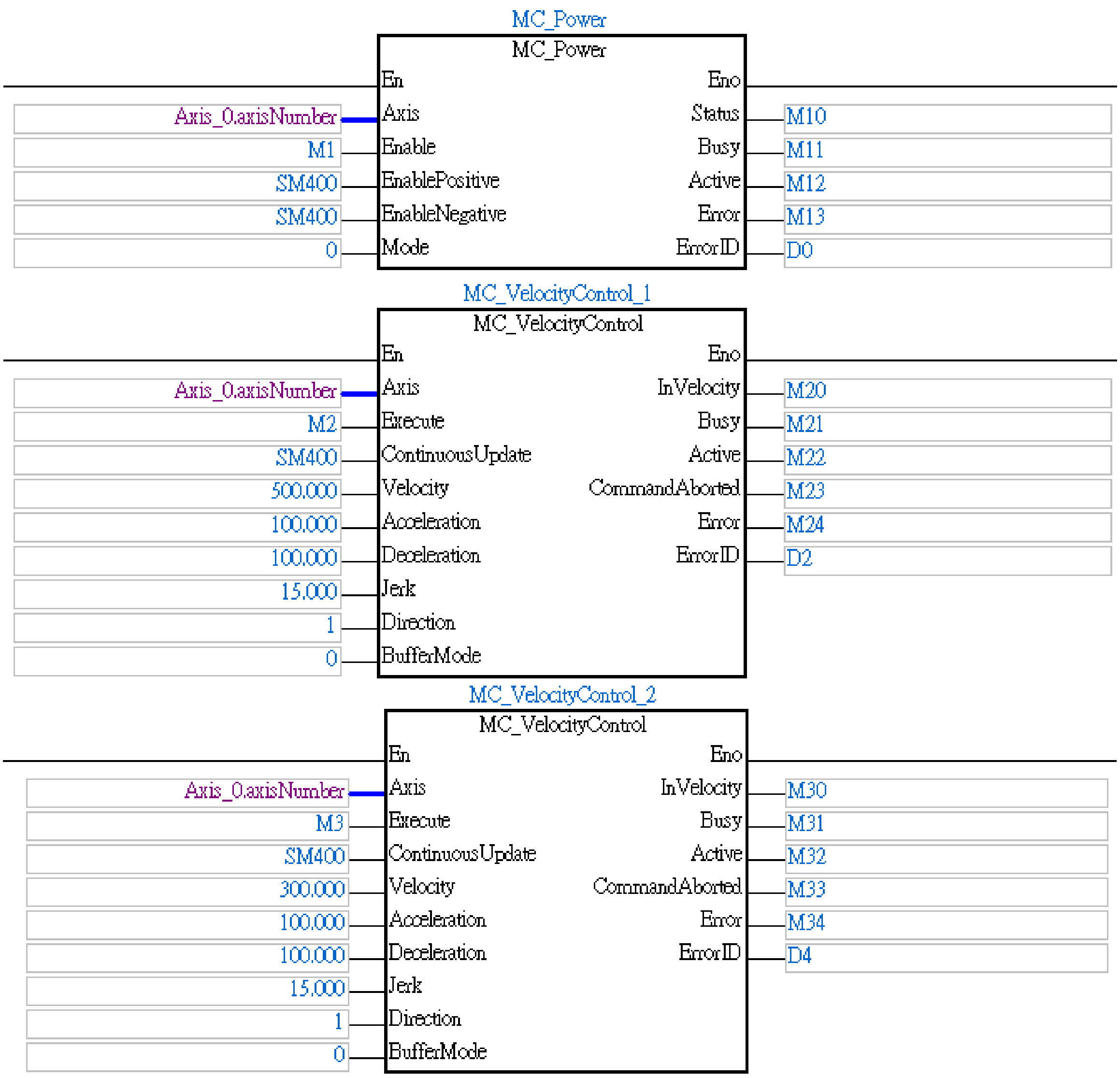
Note: when switching among velocity and other modes (torque/positon), users should refer to the corresponding axis for the real velocity behavor continued.

● Troubleshooting

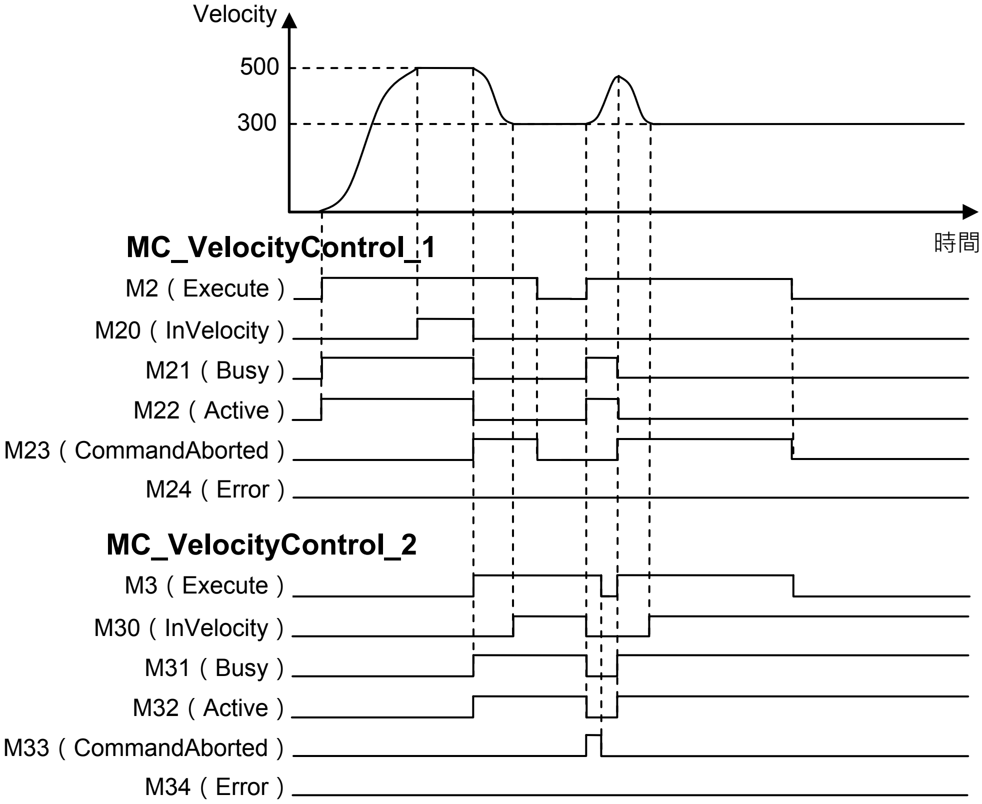
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Programming Example

The examples of the combination of two MC_VelocityControl instructions are described as below.



Motion diagram:



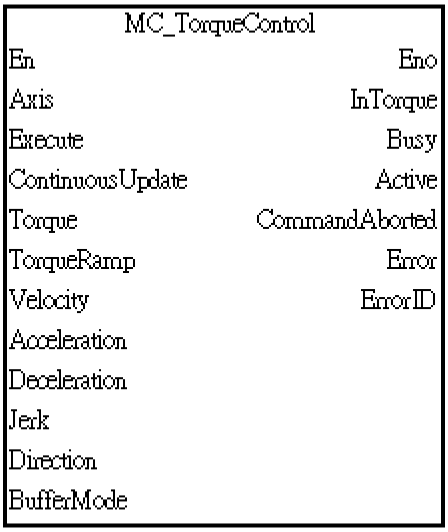
- When M2(*Execute*) changes to True, the first MC_VelocityControl controls the axis to reach the specified target velocity 500. When it reaches 500, M20(*InVelocity*) changes to True.
- If the M3(*Execute*) changes to True, M20(*InVelocity*) will change to False and M21(*CommanAborted*) will change to True while the second MC_VelocityControl is executed.
- The second MC_VelocityControl will decelerate the axis to the velocity 300. When 300 is reached, M30(*InVelocity*) will change to True and remain in this status as long as the velocity is not changed.
- When M2(*Execute*) changes to False, M21(*CommanAborted*) will change to False.
- If the first MC_VelocityControl is started again by M2(*Execute*) changes to True, the axis will abort the second MC_VelocityControl and accelerate toward the velocity 500.
- If the sencod MC_VelocityControl M3(*Execute*) changes to False and and set on again, the sencod instrution aborts the first MC_VelocityControl. In this case, the axis decelerates again without reaching the target velocity 500 of the first MC_VelocityControl.
- If MC_VelocityControl M3(*Execute*) changes to False after reaching the target velocity. M30(*InVelocity*), M31(*Busy*) and M32(*Active*) remain True.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_TorqueControl

FB/FC	Description
FB	MC_TorqueControl controls the torque by using the Torque Control Mode of the applied servo drive.



The actual movement will be limited by other input conditions, depending on the mechanical application conditions.

Before using MC_TorqueControl instruction, it is required to confirm if the below object data is supported, and use ECAT Builder in ISPSoft to set the below data in the settings of PDO communications.

- Target torque (6071 hex)
- Modes of operation (6060 hex)
- Torque actual value (6077 hex)
- Modes of operation display (6061 hex)
- Torque ramp (6087 hex)
- Maximum velocity (6080 hex)

If one of the above required object data is not set, an error will occur to indicate the problem of missing process data object setting.

1. The instruction is also applicable for force control when there is no external load.
2. When one MC_TorqueControl instruction is executed but not finished yet, it is invalid to re-execute the instruction.

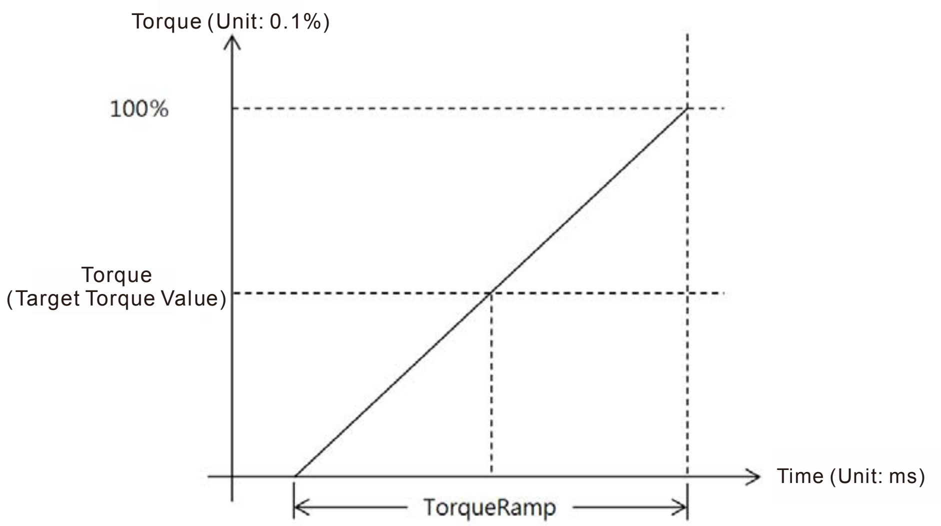
● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-

Name	Function	Data type	Setting value (Default value)	Timing for updating
Continuous Update	Continuously updates *Torque* when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
Torque	The specified target torque (Unit: refer to 6071 hex in the object dictionary)	LREAL	Refer to 6071 hex in the object dictionary (0)	When *Execute* shifts to True and *Busy* is False.
TorqueRamp	The specified maximum time derivative of the desired torque value. (Unit: refer to 6087 hex in the object dictionary)	LREAL	Refer to 6087 hex in the object dictionary (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	The maximum velocity. (Unit: refer to 6080 hex in the object dictionary)	LREAL	Refer to 6080 hex in the object dictionary (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Reserved	-	-	-
Deceleration	Reserved	-	-	-
Jerk	Reserved	-	-	-
Direction*3	Reserved	-	-	-
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*3	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and Busy is False.

***Note:**

1. When the slave is Delta A2-E servo drive, when ContinuousUpdate is ON and the value of Torque is updated, this will cause the velocity to shift.
2. Take A2-E as an example, the torque ramp (6087 hex) will be defined as below:



3. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

Outputs

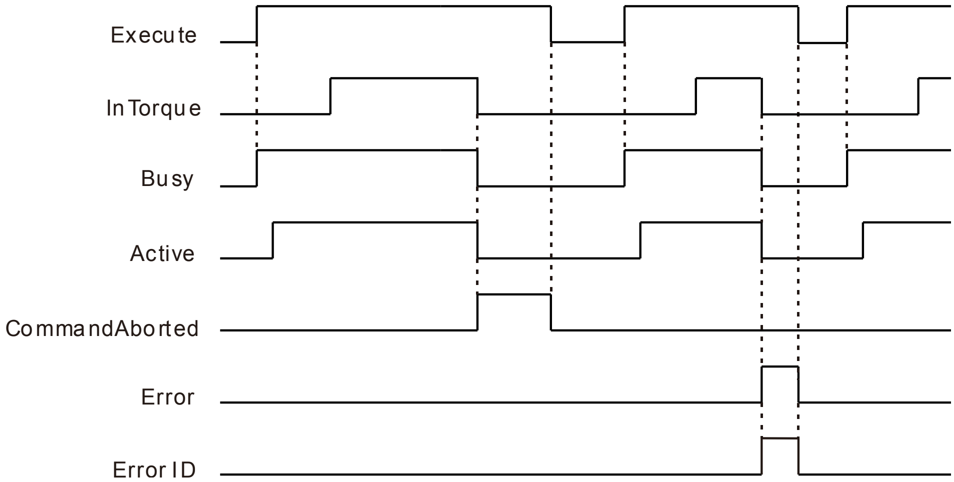
Name	Function	Data type	Output range (Default value)
InTorque	True when the specified torque or force is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

3

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
InTorque	• When the specified target torque is reached.	• When *Error* shifts to True. • When *CommandAborted* shifts to True. • When the instruction is re-executed and the target torque is changed.
Busy	• When *Execute* shifts to True.	• When *Execute* changes to False. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
Active	• When the motion on the axis is started.	• When *Execute* changes to False. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
CommandAborted	• When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



In-Outs

3

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

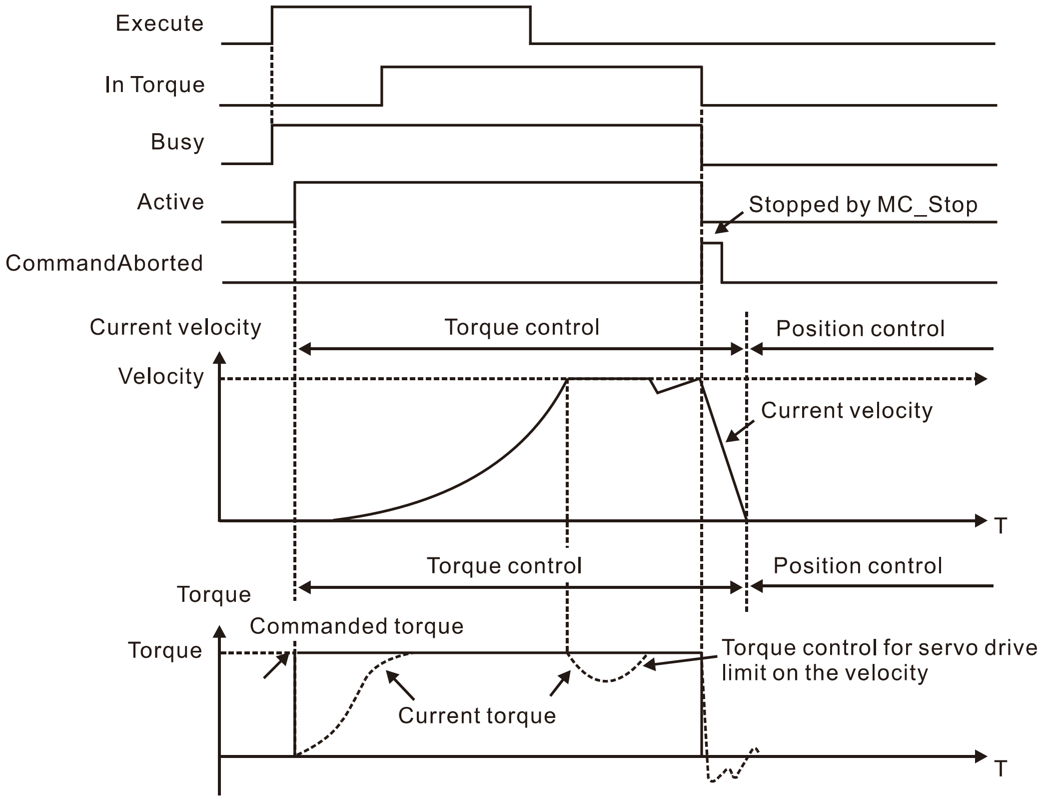
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

The instruction performs torque control with specified limit values such as target torque (*Torque*) and vmaximum velocity (*Velocity*) when *Execute* changes to True. The limit values are torque command values directly specified to the servo drive.

- If you need to do additional tests for monitoring purpose, you can design the specific tests outside of this instruction. For example, monitoring the moved distance could be performed by capturing the actual position during the motion.
- The axis will not remain in torque control status if another non-administrative motion instruction is executed.
- When the command torque equals or is higher than the maximum torque of the servo drive, the system will take the maximum torque of the servo drive as the command torque. However, the actual torque could be affected by the maximum velocity and the actual load, and it could be much lower than the command torque. Once the velocity and load change to the condition that requires the servo drive to exert a bigger torque, the servo drive will perform the operation targeting the maximum torque. In this case, the time derivative of the actual torque may exceed the input value of *TorqueRamp*.
- When the axis exceeds a soft limit, the instruction will be aborted and the axis will enter position mode. In this case, deceleration will be performed on the axis according to the command parameters of the servo drive.
- During the execution of the instruction MC_TorqueControl in PT control mode, if users execute MC_Stop, MC_Halt or the axis is in the state of Errorstop, when QuickStop is complete, the mode will be switched to CSP control mode. Refer to its servo manual for more details on the actual servo behavior.
- The execution of MC_TorqueControl can be stopped by using MC_Stop instruction.

■ Executing and Stopping MC_TorqueControl



■ Switching the Control Modes

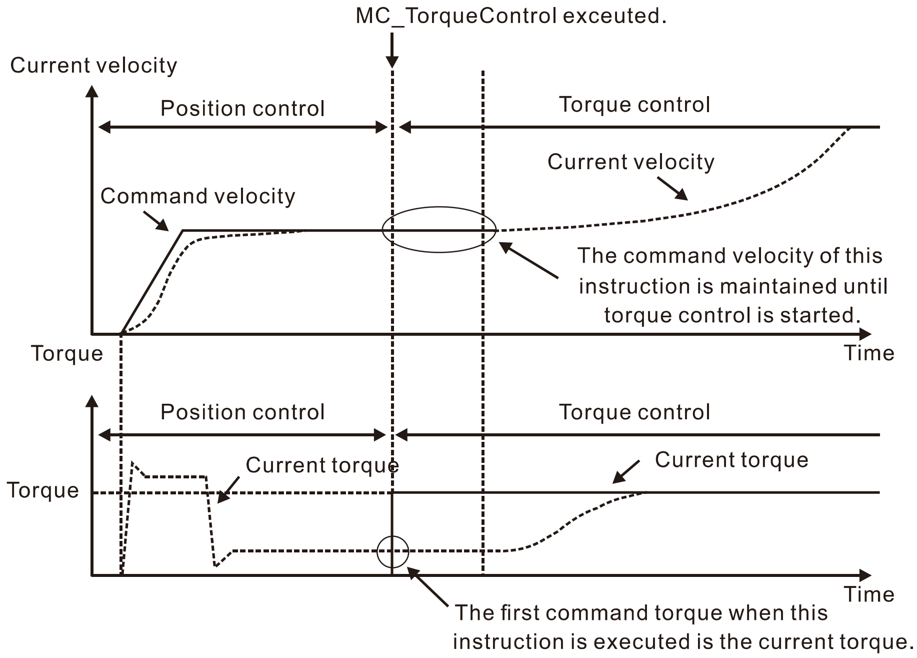
The servo drive remains in the previous Control Mode (Torque/Position) unless it is switched to another mode.

Example 1:

Command: Switching from Position control to Torque control.

Actual: The servo drive operates in Position Control mode until it is switched to Torque Control mode.

Switching from Position Control to Torque Control

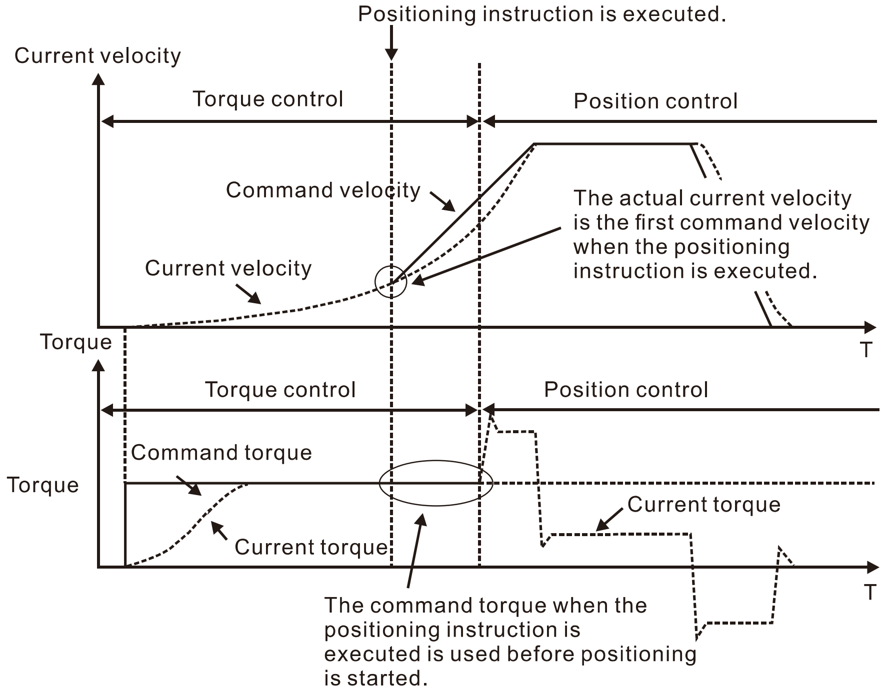


Example 2:

Command: Switching from Torque control to Position control.

Actual: The servo drive operates in Torque control mode until it is switched to Position control mode.

Switching from Torque Control to Position Control

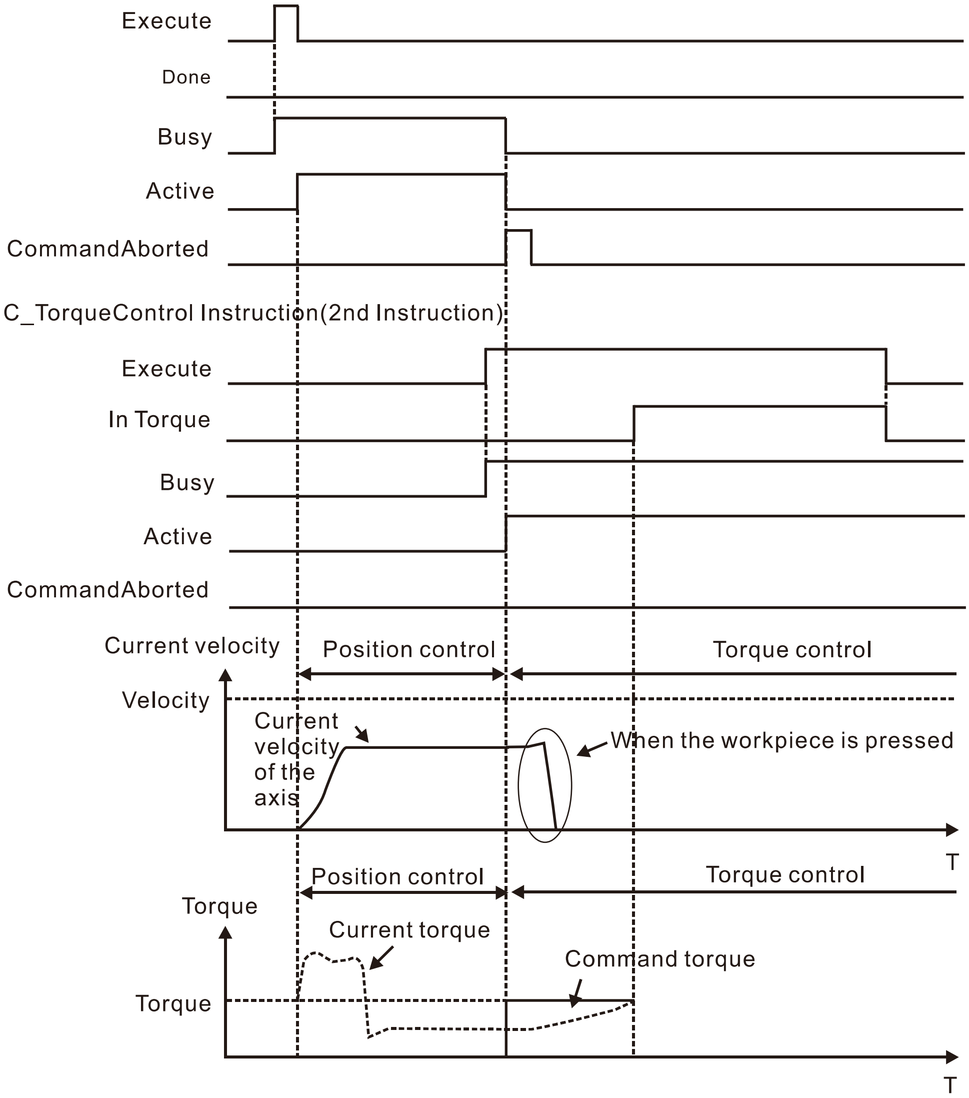


3

■ Using MC_TorqueControl to Abort the Previous Instruction

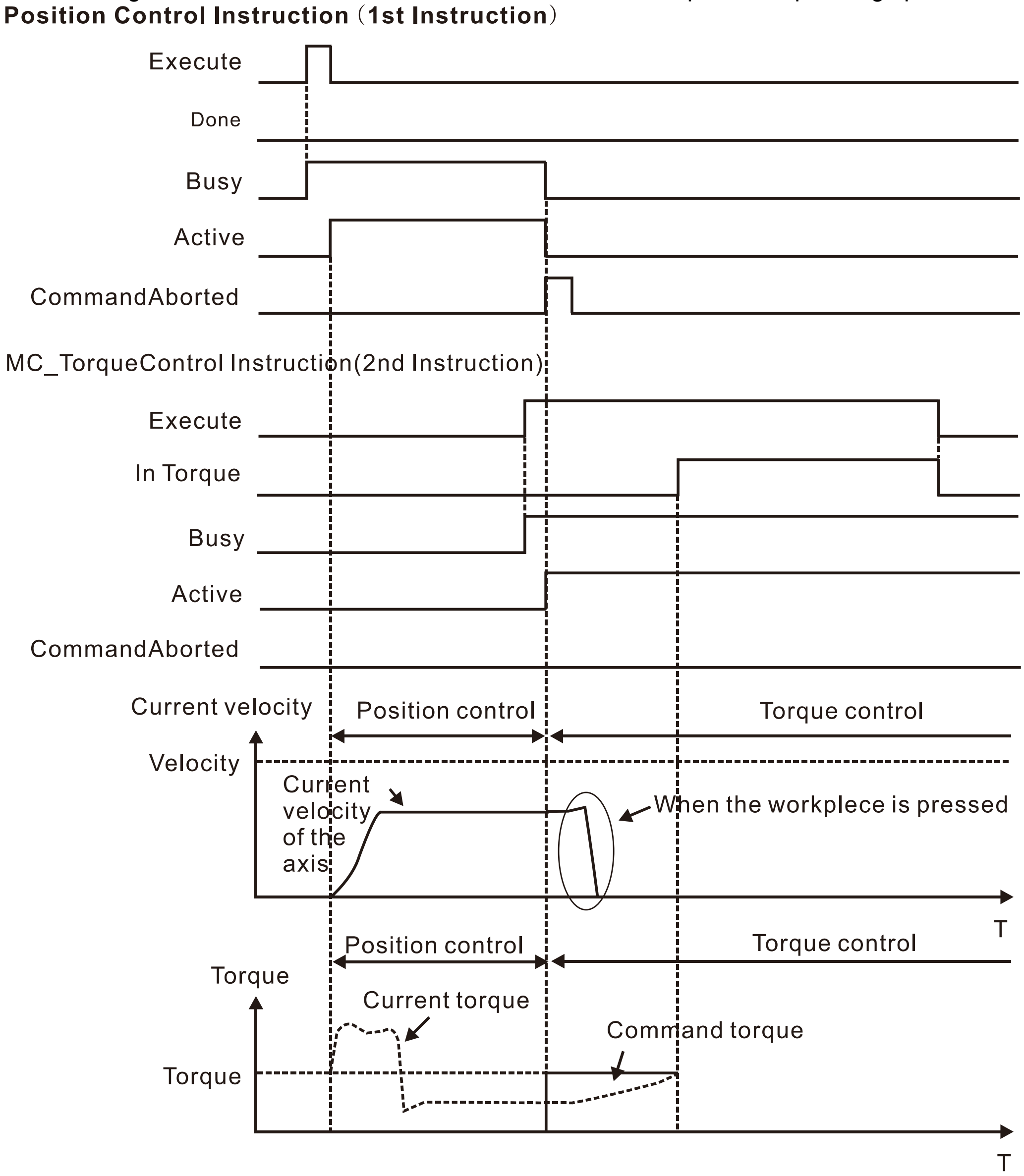
The motion diagram below shows an immediate stop for the axis to perform a pressing operation on the workpiece.

Position Control Instruction（1st Instruction）



■ Using MC_TorqueControl to Buffer the Previous Instruction

The motion diagram below shows a buffered behavior for the axis to perform a pressing operation on the workpiece.



■ BufferMode

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in "Standstill" state.

The following table lists the available buffer mode settings of MC_TorqueControl.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

The following table lists the buffer effects of MC_ TorqueControl.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_ TorqueControl	YES	YES	*InTorque*

For more information of buffer mode, refer to ***AH Motion Controller – Operation Manual.***

● Troubleshooting

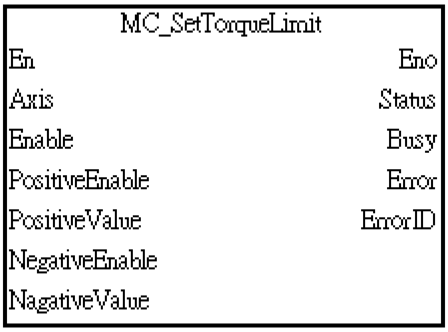
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_SetTorqueLimit

FB/FC	Description
FB	MC_SetTorqueLimit instruction limits the torque output from the servo drive via the torque limit function of the servo drive.



Before using MC_SetTorqueLimit instruction, it is required to confirm if the below object data is supported.

- Positive torque limit (60E0 hex)
- Negative torque limit (60E1 hex)

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	The axis is ready to be operated when *Enable* is True, and not ready when *Enable* is False.	BOOL	True/False (False)	-
PositiveEnable	Enables the limit of positive torque when *PositiveEnable* is True. Valid only when *Enable* is True.	BOOL	True/False (False)	Continuously updates value during busy state.
PositiveValue	Sets the limit of positive torque (Unit: refer to 60E0 hex in the object dictionary)	LREAL	Refer to 60E0 hex in the object dictionary (0)	Continuously updates value during busy state.
NegativeEnable	Enables the limit of negative torque when *NegativeEnable* is True. Valid only when *Enable* is True.	BOOL	True/False (False)	Continuously updates value during busy state.
NagativeValue	Sets the limit of negative torque (Unit: refer to 60E1 hex in the object dictionary)	LREAL	Refer to 60E1 hex in the object dictionary (0)	Continuously updates value during busy state.

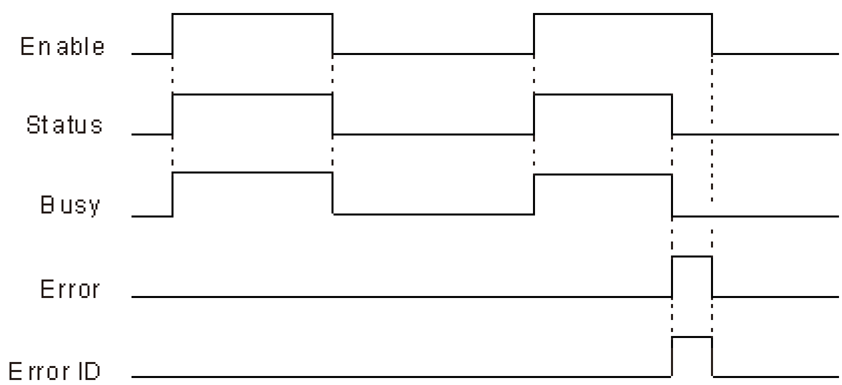
Outputs

Name	Function	Data type	Output range (Default value)
Status	True when the axis is ready to be operated.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Status	• When *Enable* shifts to True. • When MC_Power is enabled.	• When *Enable* shifts to False. • When *Error* shifts to True. • When *Enable* of MC_Power instruciton shifts to False.
Busy	• When *Enable* shifts to True and the instruction is executed	• When *Enable* shifts to False. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)

Timing Diagram



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- The instruction specifies the limitation of torque that is applied to the servo drive.
- This can only be used when it is not in the torque mode.
- Enabling *PositiveEnable* with *Enable*=True limits the torque control of the axis with the specified *PositiveValue.* Enabling *NegativeEnable* with *Enable*=True limits the torque control of the axis with the specified *NegativeValue.*
- When *PositiveEnable*/*NegativeEnable* shifts to False, the value of the positive/negative torque limit will be set and keep in the servo drive.
- When *Enable* of this instruction shifts to False, the values of the positive and the negative torque limits will also be set and keep in the servo drive. Meanwhile *Busy* and *Status* shift to False.

Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.

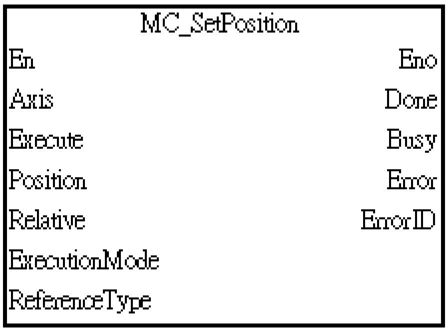
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_SetPosition

FB/FC	Description
FB	MC_SetPosition changes the current position by shifting the coordinate system of an axis.



- The changing of the coordinate system is made by modifying both the current position of the instruction (command position) and the actual position from the feedback signals with the same value, which is set by *Position.*
- The following error between command position and actual position remains the same value.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Position	Specifies the set position. (Unit: user unit)	LREAL	Negative number, positive number, or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Relative	Specifies a relative distance or an absolute position.	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
ExecutionMode	Reserved	-	-	-
ReferenceType	Specifies the source of reference position.	eMC_SOURCE*2	0: mcCommandedValue 2: mcActualValue (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. "*Relative*=True" means the value of *Position* will be added to the current position. "*Relative*=False" means that command current position will be set to the value specified in the parameter *Position*.
2. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

3

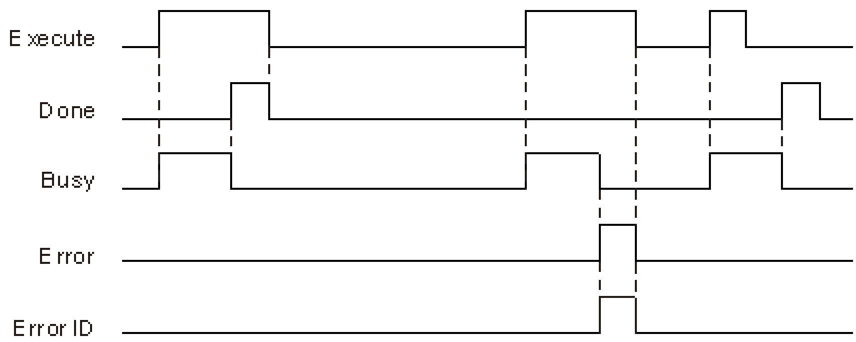
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the position change is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the position change is completed	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* shifts to True and the instruction is executed.	• When *Done* shifts to True. • When *Error* shifts to True
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

Timing Diagram



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- MC_SetPosition changes the current position of the servo axis to the specified target position.
- MC_SetPosition changes the current position by shifting the coordinate system of an axis. The changing of the coordinate system is made by modifying both the current position of the instruction (command position) and the actual position from the feedback signals with the same value, which is set by *Position.*
- The following error between command position and actual position remains the same value, and can be explained by the following equation.
  Actual position after change = Command position − Following error before change
- The instruction can also be used during motion without changing the target distance which is defined and fixed at the beginning of the motion instruction. For example, if the motion started at 0 and MC_SetPosition executes during the motion, the execution effects for both MC_MoveRelative with *Distance*=10000 and MC_MoveAbsolute with *Position*=10000 will be the same.
- When the Count Mode is set to rotary mode (modulo axis), you can specify the set position (*Position*) to a value outside the range of the modulo.

3

Note:

1. Avoid using the MC_SetPosition instruction for a master axis that is in synchronization with instructions such as MC_GearIn, and MC_CAM; this act will cause the slave axis to jump from position to position in inconsistent velocity.
2. If another MC_SetPosition instruction is executed while the current MC_SetPosition is busy, the last instruction will take the priority. In this case, *Done* of the current MC_SetPosition instruction shifts to True, however the position change of the current instruction is not completed.

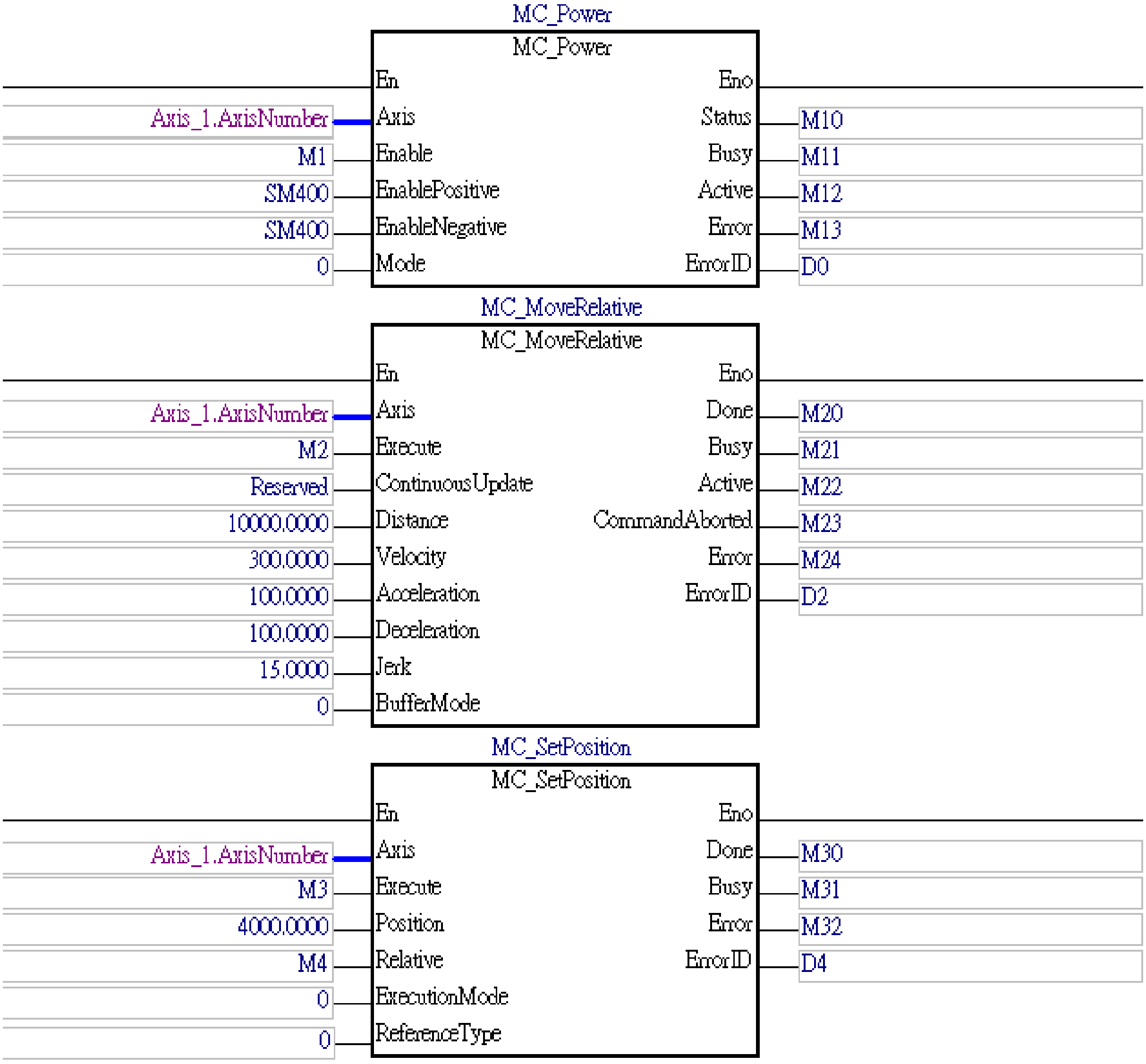
Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

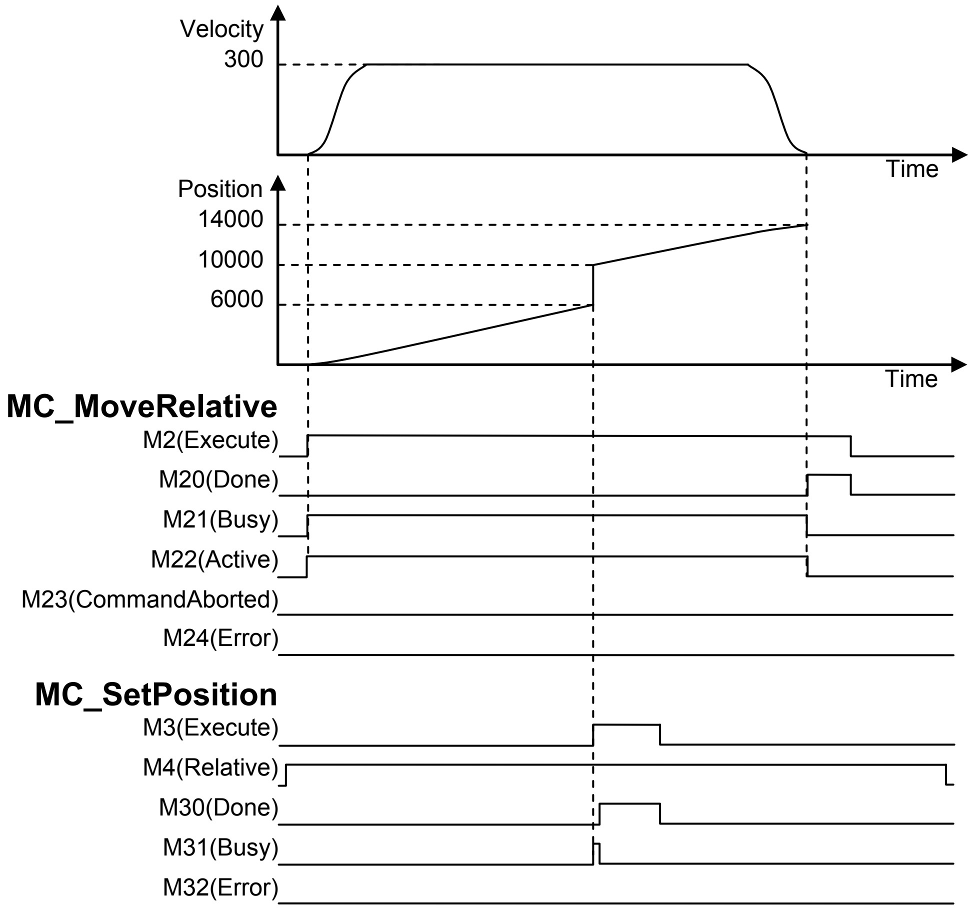
Programming Example 1

This example describes the effects of a combination of MC_MoveRelative and MC_SetPosition with its input *Relative* set to True.

- "*Relative*=True" means the value of *Position* will be added to the current position.
- Execution of the instruction will have no influence on the actual execution results of the MC_MoveRelative instruction which is being executed.



Motion diagram:

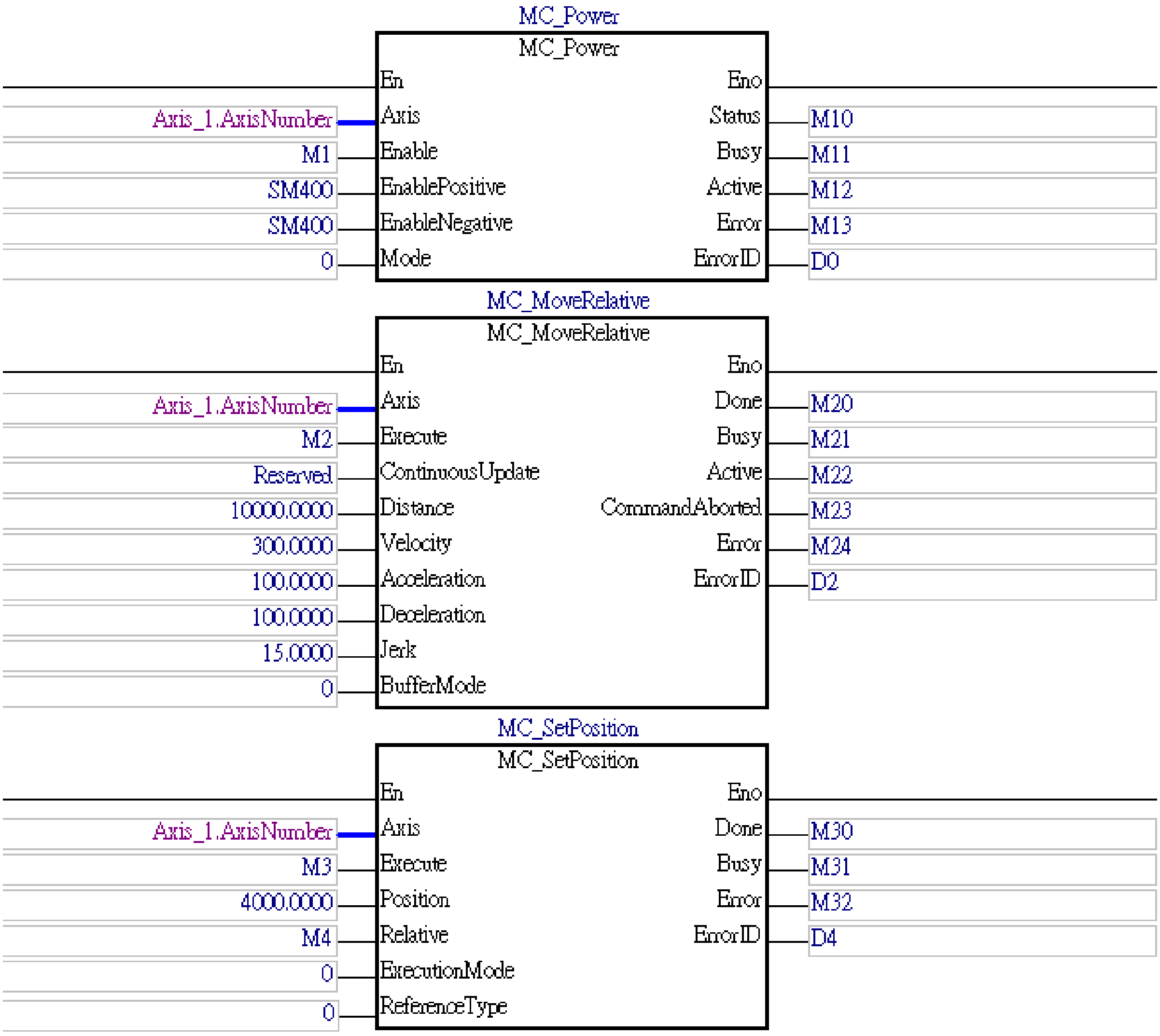


- MC_MoveRelative instruction executes when M2(*Execute*) changes to True. MC_SetPosition instruction executes after a few seconds.
- The set position specified in *Position* is 4,000 with M4(*Relative*)=True, meaning 4,000 is added to current command position 6,000 when MC_SetPosition executes. The command position becomes 10,000 after MC_SetPosition instruction executes. The defined moving distance at the beginning of MC_MoveRelative is 10,000, so MC_MoveRelative keeps going for the remaining 4,000 then reaches 14,000 (14000=6000+4000+(10000-6000)).
- The motion of the physical device corresponding to MC_MoveRelative is not affected after MC_SetPosition changed the command position because the moving distance for the commanded motion remains 10,000, which is the same as the value set in *Distance* of MC_MoveRelative.

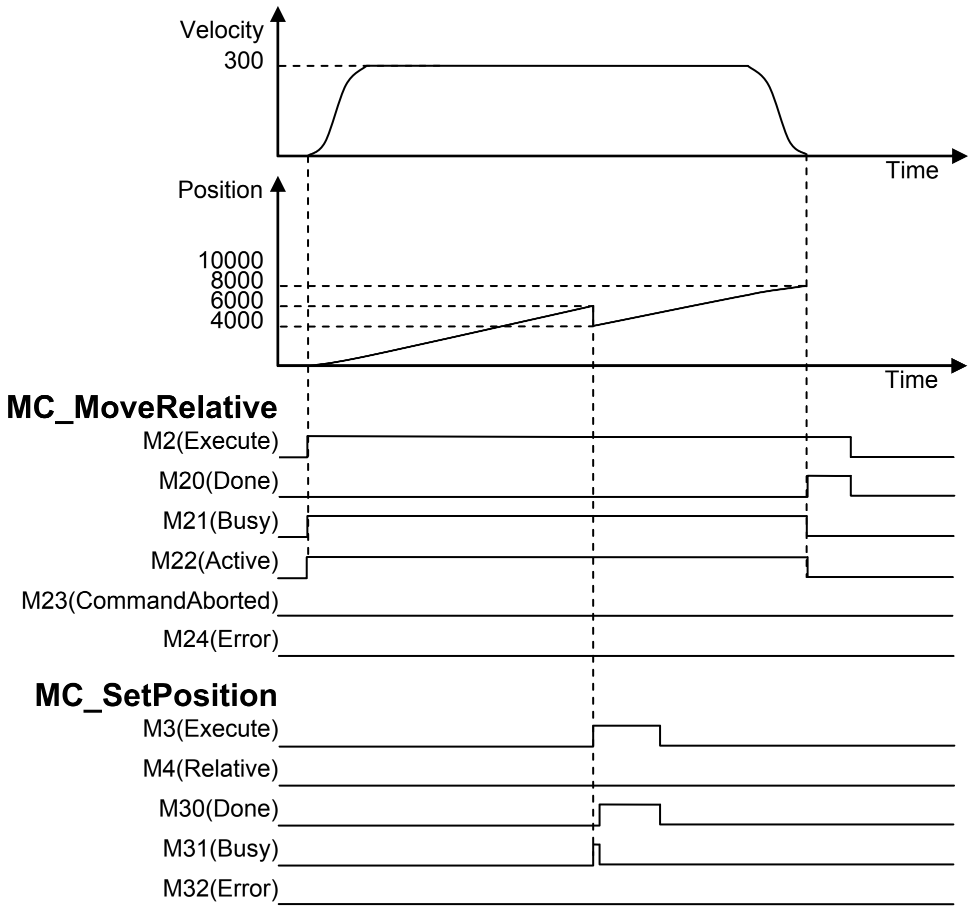
Programming Example 2

This example describes the effects of a combination of MC_MoveRelative and MC_SetPosition with its input *Relative* set to False.

- "*Relative*=False" means that command current position will be set to the value specified in the parameter *Position.*
- Execution of the instruction will have no influence on the actual execution results of the MC_MoveRelative instruction which is being executed.



Motion diagram:

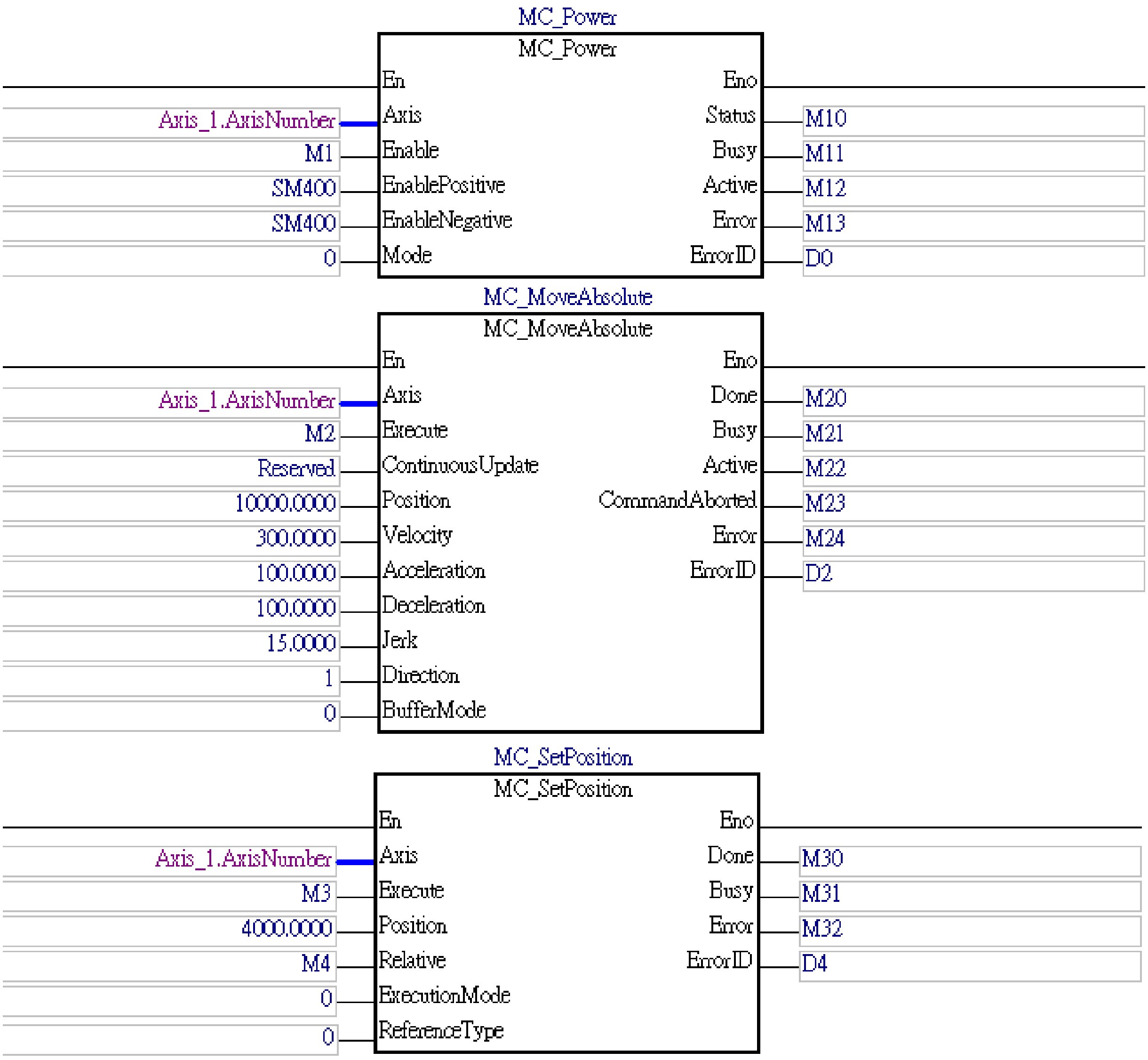


- MC_MoveRelative instruction executes when M2(*Execute*) changes to True. MC_SetPosition instruction executes after a few seconds.
- The set position specified in *Position* is 4,000 with M4(*Relative*)=False, meaning 4,000 is taken to replace the current command position 6,000 when MC_SetPosition executes. The command position becomes 8,000 after MC_SetPosition instruction executes. The defined moving distance at the beginning of MC_MoveRelative is 10,000, so MC_MoveRelative keeps going for the remaining 4,000 then reaches 8,000 (8000=4000+(10000-6000)).
- The motion of the physical device corresponding to MC_MoveRelative is not affected after MC_SetPosition changed the command position because the moving distance for the commanded motion remains 10,000, which is the same as the value set in *Distance* of MC_MoveRelative.

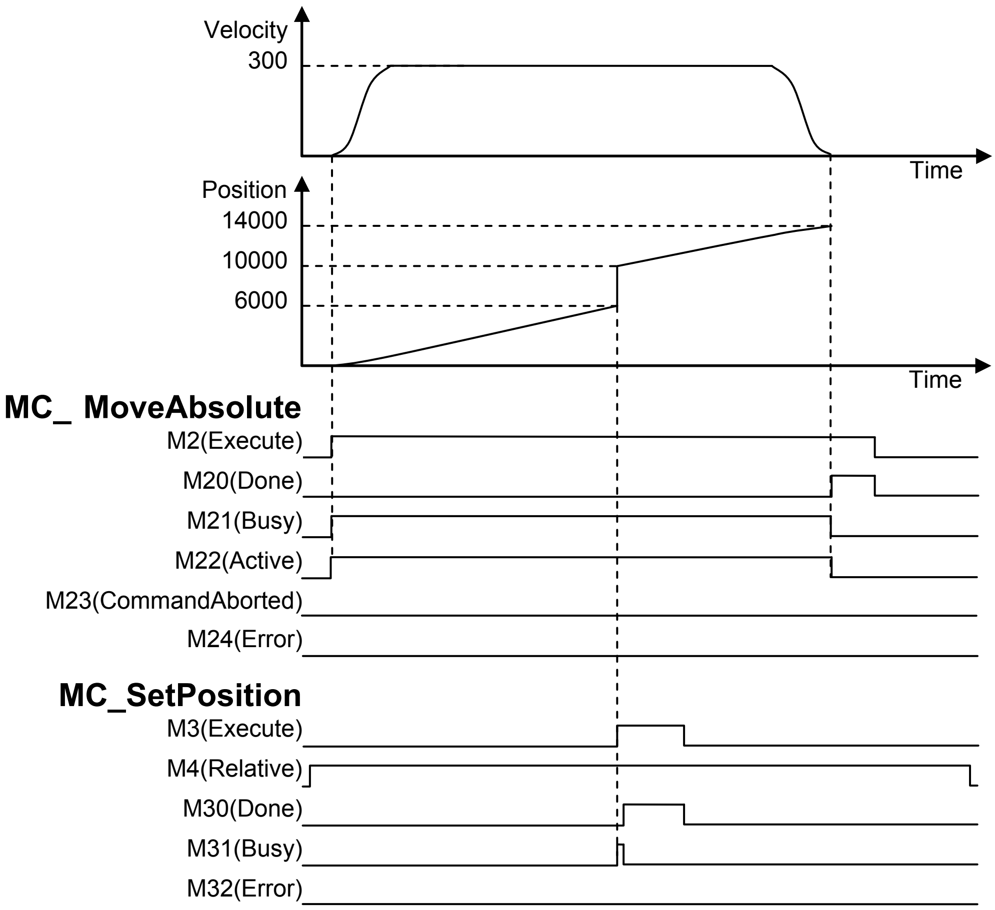
Programming Example 3

This example describes the effects of a combination of MC_MoveAbsolute and MC_SetPosition with its input *Relative* set to True.

- "*Relative*=True" means the value of *Position* will be added to the current position.
- Execution of the instruction will have no influence on the actual execution results of the MC_MoveAbsolute instruction which is being executed.



Motion diagram:



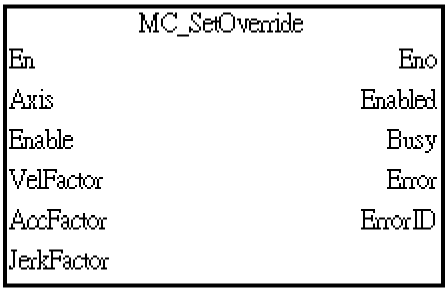
- MC_MoveAbsolute instruction executes when M2(*Execute*) changes to True. MC_SetPosition instruction executes after a few seconds.
- The set position specified in *Position* is 4,000 with M4(*Relative*)=True, meaning 4,000 is added to current command position 6,000 when MC_SetPosition executes. The command position becomes 10,000 after MC_SetPosition instruction executes. The defined moving distance at the beginning of MC_MoveAbsolute is 10,000, so MC_MoveAbsolute keeps going for the remaining 4,000 then reaches 14,000 (14000=6000+4000+(10000-6000)).
- The motion of the physical device corresponding to MC_MoveAbsolute is not affected after MC_SetPosition changed the command position because the moving distance for the commanded motion remains 10,000, which is defined and fixed at the beginning of MC_MoveAbsolute by Position=10000.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_SetOverride

FB/FC	Description
FB	MC_SeOverride changes the velocity override factors so as to change the target velocity of a motion axis.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	The axis is ready to be operated when *Enable* is True. When Enable is True, override factor will be continuously updated.	BOOL	True/False (False)	-
VelFactor	Defines the velocity override factor. (Unit: %)	LREAL	0 to 500 (100)	Continuously updates value during busy state.
AccFactor (Reserved)	Reserved	-	-	-
JerkFactor (Reserved)	Reserved	-	-	-

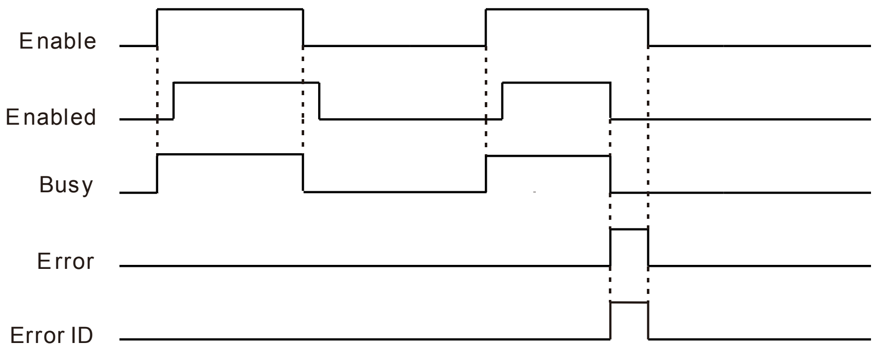
● Outputs

Name	Function	Data type	Output range (Default value)
Enabled	True when the axis is ready to be operated.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Enabled	• When *Enable* shifts to True and the axis is ready to be operated.	• After one period when *Enable* shifts to False. • When *Error* shifts to True.
Busy	• When *Enable* shifts to True and the instruction is executed	• When *Enable* shifts to False. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- The MC_SetOverride instruction changes the target velocity (and acceleration/deceleration/jerk, if applicable) of the motion axis with its specified override factors.
- The new target velocity can be obtained by the equation as below.

  The new target velocity = Current target velocity × The specified override factor
- The override value will command the axis to accelerate or decelerate to the target velocity.
- By setting the velocity override value as 0, you can pause the axis motion while keeping the operation status. Specifying the velocity override factor as 0 will change the target velocity to 0 and decelerate the axis to a velocity of 0 while keeping the axis operation status.
- Whenever *Enable* is True, the override factors will be updated continuously.
- The override factors stay the same when *Enable* shifts to False or an error occurs.
- If another MC_SetOverride is executed during the current execution of MC_SetOverride, the execution of the

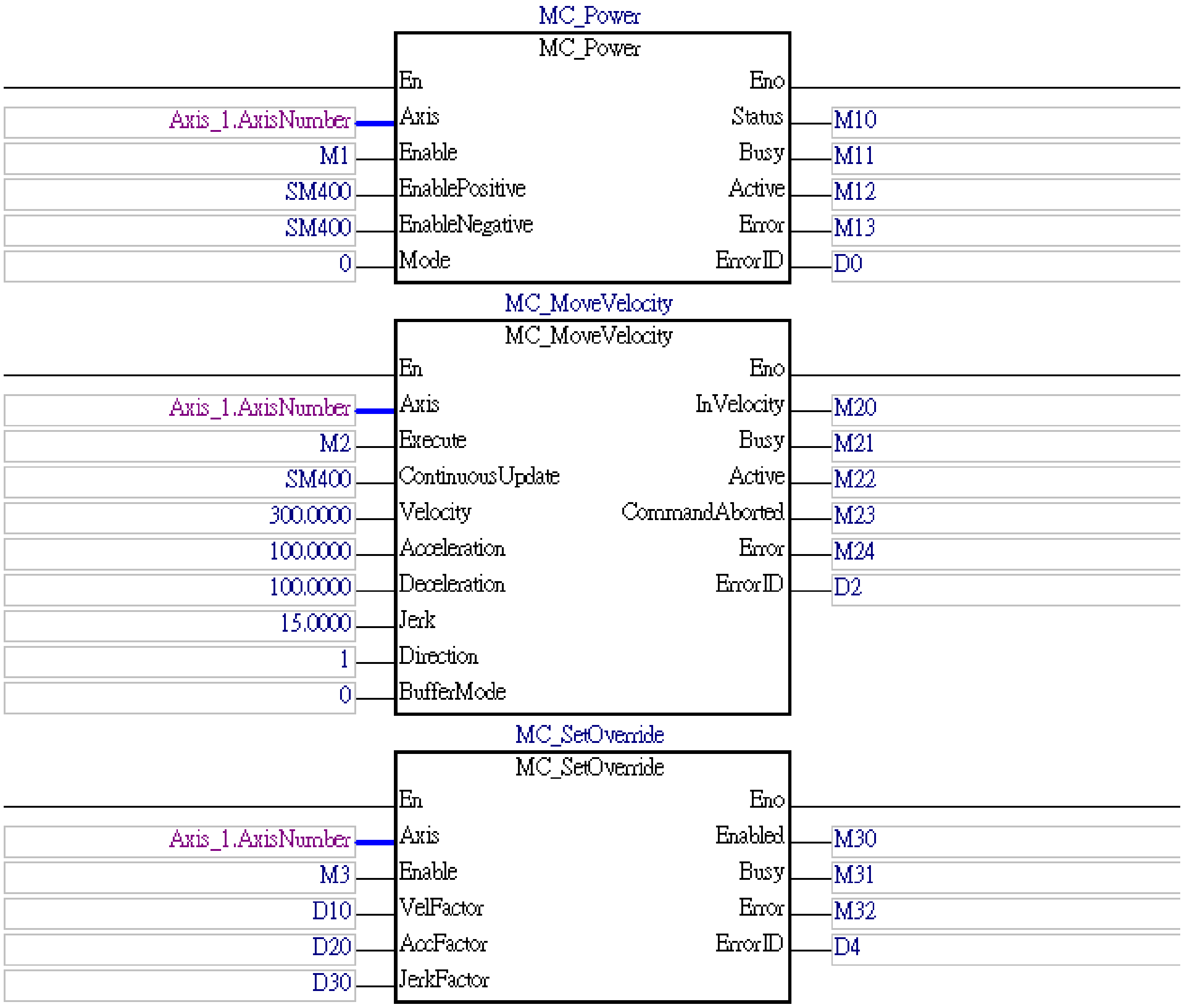
last MC_SetOverride will take the priority. *Enabled* will be True for both instructions.

Troubleshooting

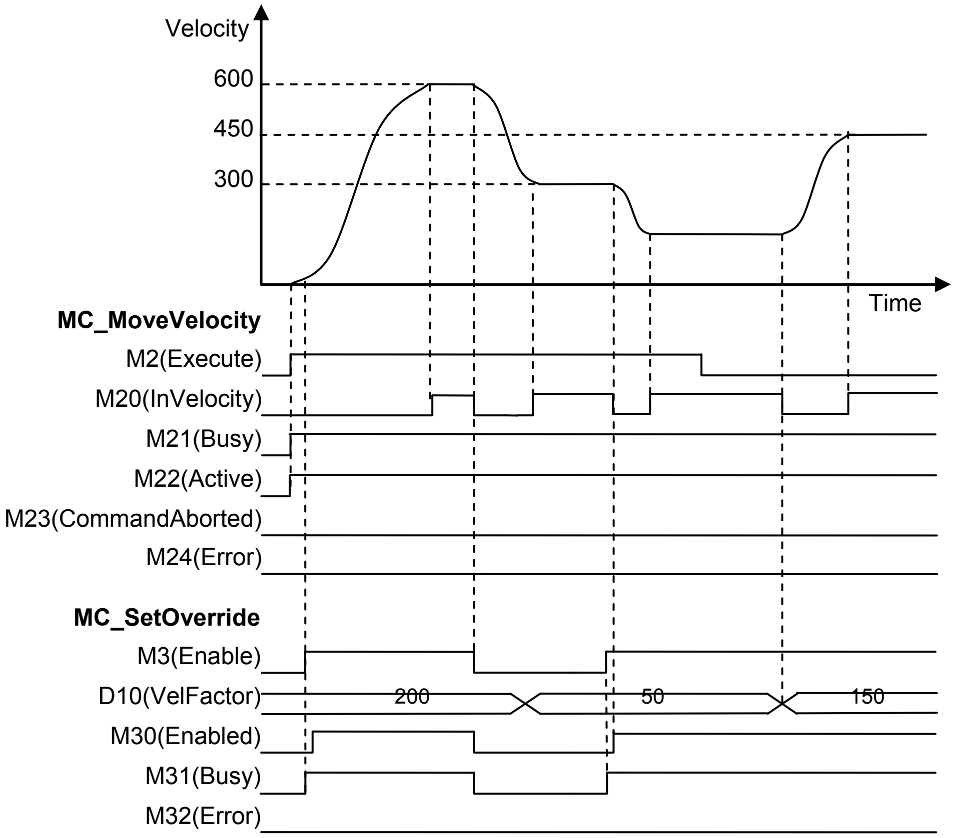
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

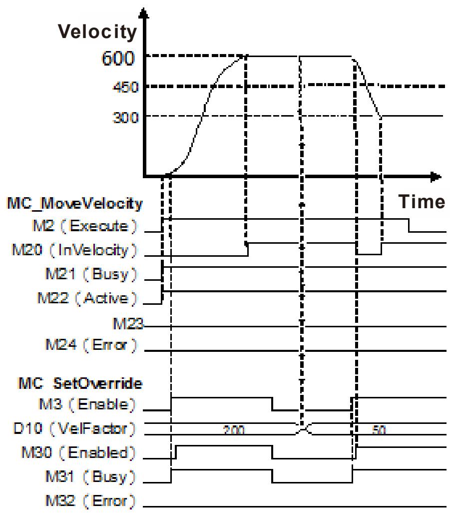
Programming Example

This example describes the influence on the execution results of the MC_MoveVelocity instruction by using the MC_SetOverride instruction.



Motion diagram:





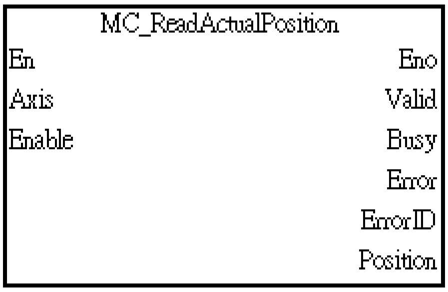
- When the MC_MoveVelocity is being executed without reaching its target velocity(300) yet, executing the MC_SetOverride instruction will change the target velocity from 300 to 600 when MC_Setoverride is enabled with D10(*VelFactor*) set as 200, indicaitng a 200% shift.. When the target velocity of the MC_MoveVelocity reaches the new target velocity(600), M20(*InVelocity)* will change to True.
- When M3(*Enable*) of MC_SetOverride changes to False, the override factor for the target velocity of the axis stays at 600.
- The modification made on *VelFactor* will continuously be updated and the axis will react accordingly as long as *Enable* of MC_SetOverride True. You can observe this behavior by the variation from velocity 600 to 300 (*VelFactor*=50).

- **Supported Products**

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_ReadActualPosition

FB/FC	Description
FB	This instruction reports the actual axis position continuously when *Enable* is set.



1. The output *Valid* shifts to True when the output *Position* is valid. If *Enable* is reset, the data loses its validity, and all outputs are reset. In this case, to update new position data requires enabling this instruction again.
2. The reported value of the actual position at the output is obtained from the servo drive and the unit of the position value is the same as that of the servo drive.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Obtains the value of the outputs continuously while enabled.	BOOL	True/False (False)	-

● Outputs

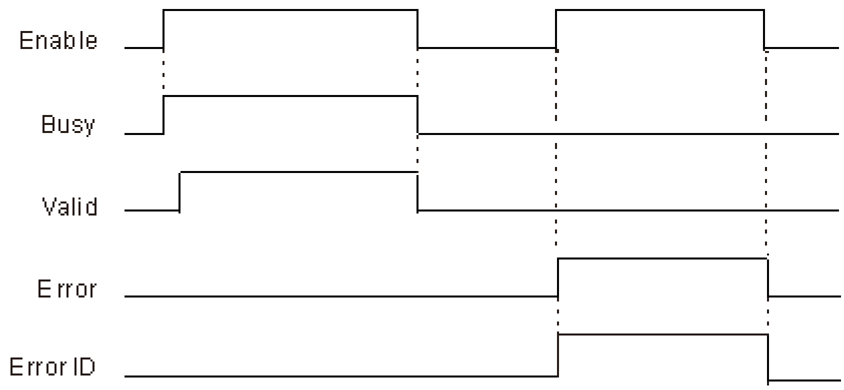
Name	Function	Data type	Output range (Default value)
Valid	True when the axis position at the output is available.	BOOL	True/False (False)
Busy	Indicates there are incoming new output values and the instruction is not yet finished.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
Position	Current absolute position (user unit)	LREAL	Negative number, positive number or 0 (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for changing to False
Valid	● When *Enable* shifts to True and the axis position at the output is available.	● When *Enable* shifts to False. ● When *Error* shifts to True.

Busy	• When *Enable* shifts to True	• When *Enable* shifts to False. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)
Position	Continuously updates value when *Valid* is True.	Continuously updates value when *Valid* is True.

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

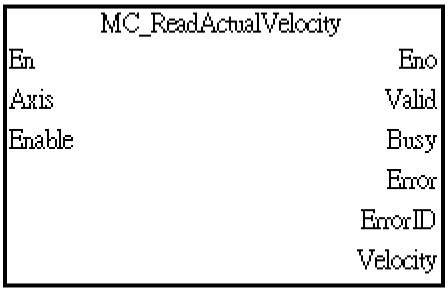
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual*.**

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_ReadActualVelocity

FB/FC	Description
FB	This instruction reports the actual axis velocity continuously when *Enable* is set.



1. The output *Valid* is True when the output *Velocity* is valid. If *Enable* is reset, the data loses its validity, and all outputs are reset. In this case, to update new velocity data requires enabling this instruction again.
2. The reported value of the actual velocity at the output is obtained from the servo drive and the unit of the velocity value is the same as that of the servo drive.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Obtains the value of the outputs continuously while enabled.	BOOL	True/False (False)	-

● Outputs

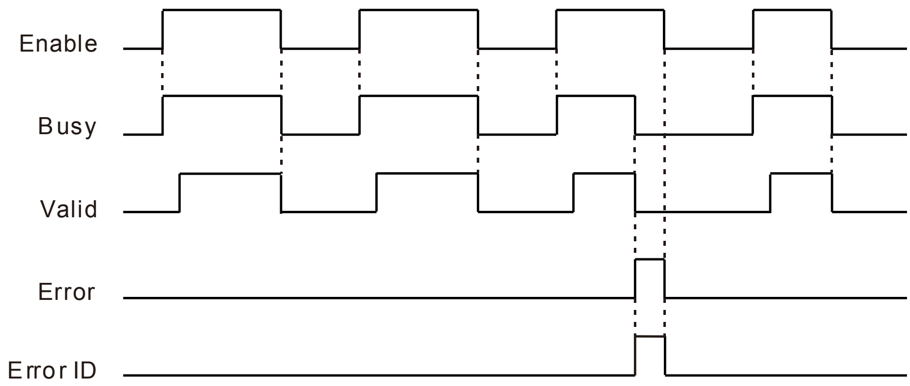
Name	Function	Data type	Output range (Default value)
Valid	True when the axis velocity at the output is available.	BOOL	True/False (False)
Busy	Indicates there are incoming new output values and the instruction is not yet finished.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to table XX.XX for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)
Velocity	The value of the actual velocity (user unit/sec)	LREAL	Negative number, positive number or 0 (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Valid	● When *Enable* shifts to True and the actual	● When *Enable* shifts to False.

	velocity at the output is available.	• When *Error* shifts to True
Busy	• When *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)
Velocity	Continuously updates value when *Valid* is True.	Continuously updates value when *Valid* is True.

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

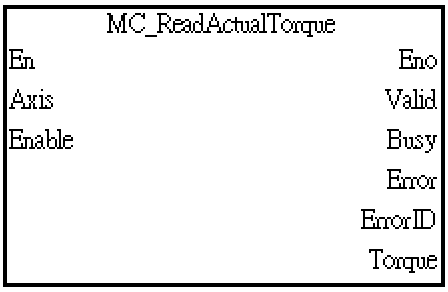
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual*.**

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_ReadActualTorque

FB/FC	Description
FB	This instruction reports the axis torque continuously when *Enable* is set.



1. The output *Valid* is True when the output *Torque* is valid. If *Enable* is reset, the data loses its validity, and all outputs are reset. In this case, to update new velocity data requires enabling this instruction again.
2. The reported value of the actual torque at the output is obtained from the servo drive and the unit of the torque value is the same as that of the servo drive.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Obtains the value of the outputs continuously while enabled.	BOOL	True/False (False)	-

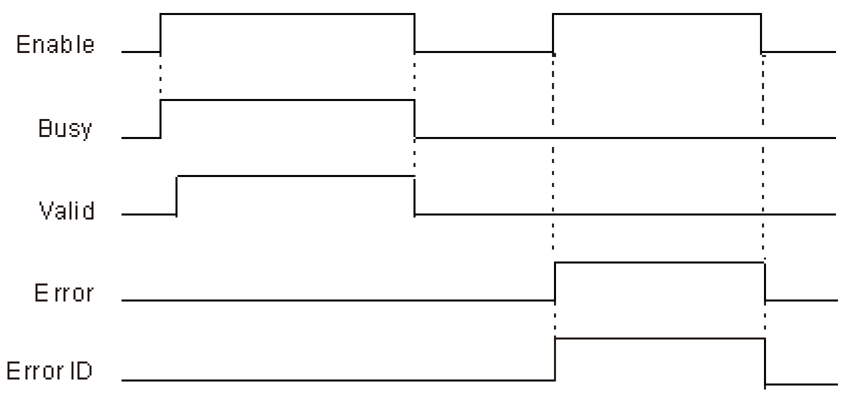
● Outputs

Name	Function	Data type	Output range (Default value)
Valid	True when the axis torque at the output is available.	BOOL	True/False (False)
Busy	Indicates there are incoming new output values and the instruction is not yet finished.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
Torque	The value of the actual torque or force (unit: refer to 6077 hex from the object dictionary of the slave)	LREAL	refer to 6077 hex from the object dictionary of the slave for the available range or use 0 (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Valid	• When *Enable* shifts to True and the actual torque at the output is available.	• When *Enable* shifts to False. • When *Error* shifts to True
Busy	• When *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)
Torque	Continuously updates value when *Valid* is True.	Continuously updates value when *Valid* is True.

■ Timing Diagram



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

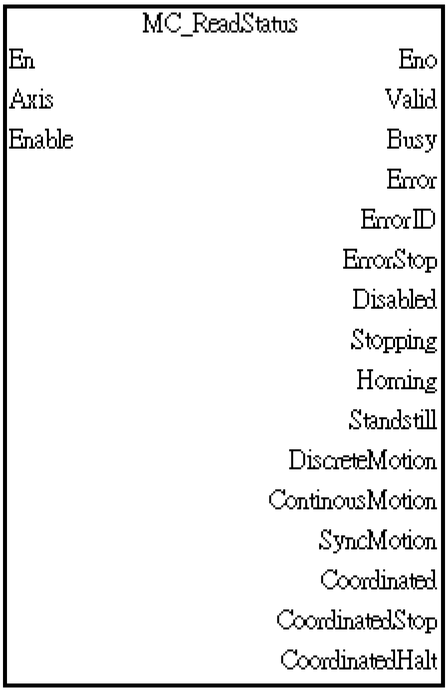
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_ReadStatus

FB/FC	Description
FB	MC_ReadStatus reads the state of the axis and indicates it at the outputs.



- The instruction signals the axis state at the outputs after the execution of this instruction is completed.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Executes the instruction when *Enable* changes to True.	BOOL	True/False (True)	-

Note: This instruction will read the axis state constantly when *Enable* changes to True.

● Outputs

Name	Function	Data type	Output range (Default value)
Valid	True when the axis state at the output is available.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)
ErrorStop	Refer to Chapter 7 Motion Control	BOOL	True/False (False)

Name	Function	Data type	Output range (Default value)
Disabled	Programming of ***AH Motion Controller – Operation Manual*** for the state diagram.	BOOL	True/False (False)
Stopping		BOOL	True/False (False)
Homing		BOOL	True/False (False)
Standstill		BOOL	True/False (False)
DiscreteMotion		BOOL	True/False (False)
ContinousMotion		BOOL	True/False (False)
SyncMotion		BOOL	True/False (False)
Coordinated		BOOL	True/False (False)
CoordinatedStop		BOOL	True/False (False)
CoordinatedHalt		BOOL	True/False (False)

3

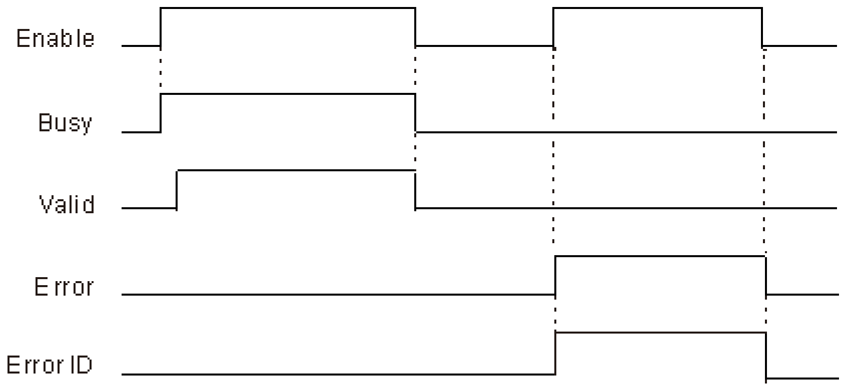
■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Valid	● When *Enable* shifts to True and the axis state at the output is available.	● When *Enable* shifts to False. ● When *Error* shifts to True.
Busy	● When *Enable* shifts to True.	● When *Enable* shifts to False. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Enable* shifts from True to False. (Error code is cleared)
ErrorStop	● When the axis status indicates “ErrorStop”.	● When the axis status did not indicate “ErrorStop”.
Disabled	● When the axis status indicates “Disabled”.	● When the axis status did not indicate “Disabled”.
Stopping	● When the axis status indicates “Stopping”.	● When the axis status did not indicate “Stopping”.
Homing	● When the axis status indicates” Homing”.	● When the axis status did not indicate “Homing”.
Standstill	● When the axis status indicates “Standstill”.	● When the axis status did not indicate “Standstill”.
DiscreteMotion	● When the axis status indicates “DiscreteMotion”.	● When the axis status did not indicate “DiscreteMotion”.
ContinousMotion	● When the axis status indicates “ContinousMotion”.	● When the axis status did not indicate “ContinousMotion”.
SyncMotion	● When the axis status indicates	● When the axis status did not indicate

Name	Timing for shifting to True	Timing for shifting to False
	"SyncMotion".	"SyncMotion".
Coordinated	● When the axis status indicates "Coordinated".	● When the axis status did not indicate "Coordinated".
CoordinatedStop	● When the axis status indicates "CoordinatedStop".	● When the axis status did not indicate "CoordinatedStop".
CoordinatedHalt	● When the axis status indicates "CoordinatedHalt".	● When the axis status did not indicate "CoordinatedHalt".

Note: When *Enable* shifts to False, the states of *ErrorStop, Disabled, Stopping, Homing, Standstil, DiscreteMotion, ContinousMotion, SyncMotion, Coordinated, CoordinatedStop* and *CoordinatedHalt* outputs remain unchanged.

■ Timing Diagram



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* is rising edge triggered.

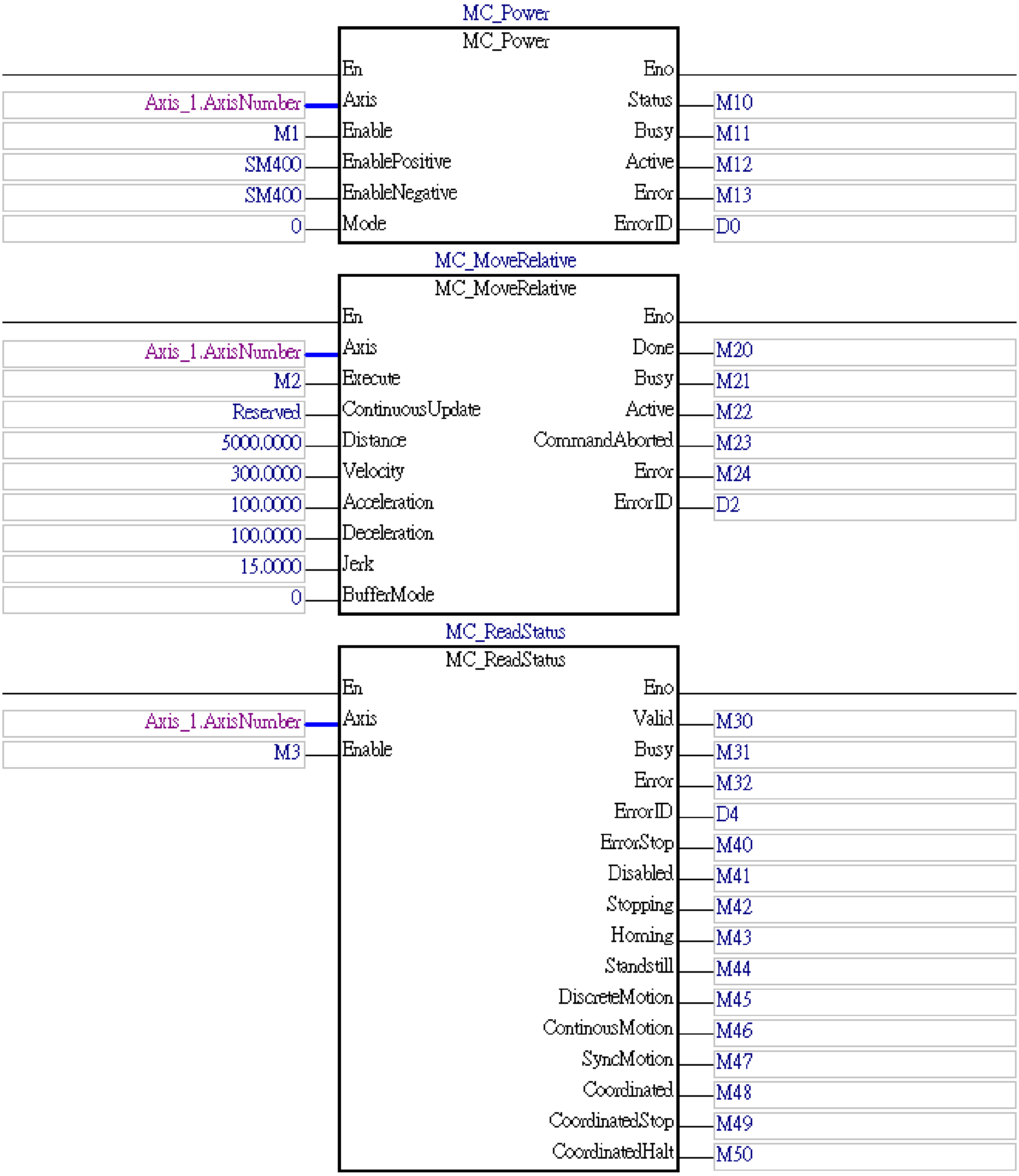
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

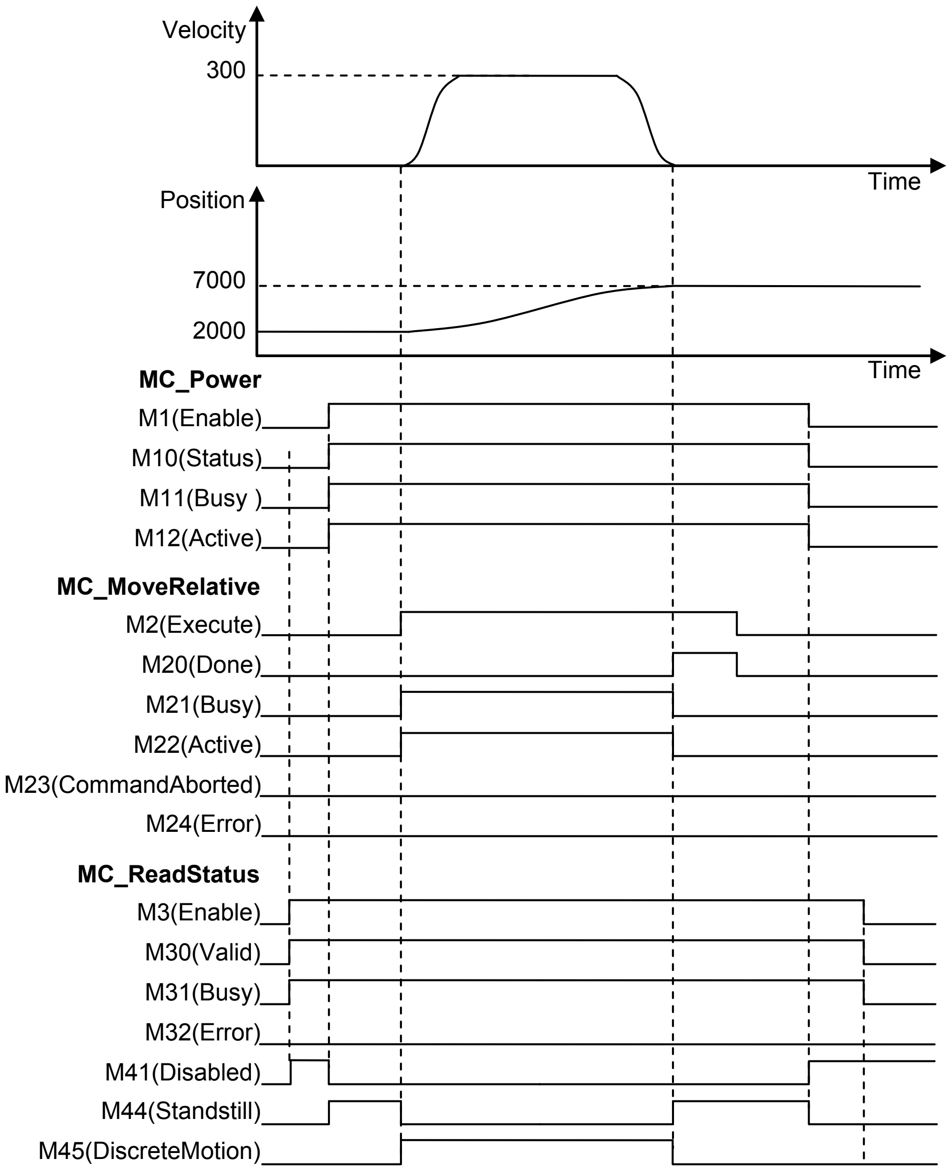
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

The MC_ReadStatus instruction is used as follows.



Motion diagram:



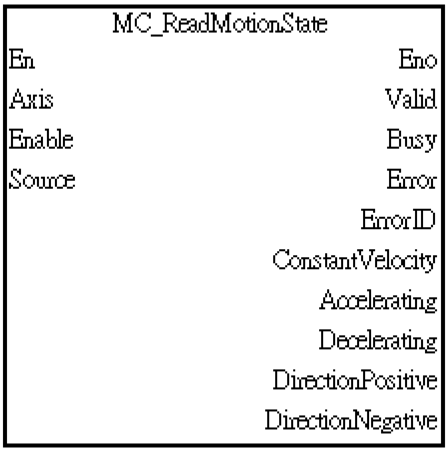
- When MC_ReadStatus is enabled (M3=True), M41 (*Disabled*) change to True, indicating the axis is not yet active.
- When MC_Power is enabled (M1 = True), M44 (*Standstill*) changes to True, and M41 (*Disabled*) changes to False.
- When MC_MoveRelative is executed (M2=True), the axis starts to move from current position to the designated target position. M45 (*DiscreteMotion*) change to True and M44 (*Standstill*) changes to False.
- When the axis reaches the specified target position, M20 (*Done*) and M44 (*Standstill*) change to True, and M45 (DiscreteMotion) changes to False.
- When MC_Power is disabled (M1 = False), M41 (*Disabled*) changes to True, and M44 (*Standstill*) changes to False.
- When MC_ReadStatus is disabled (M3 = False), M41 (*Disabled*) remains unchanged.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_ReadMotionState

FB/FC	Description
FB	MC_ReadMotionState reports details of the axis status relating the on-going motion behavior.



The output *Valid* is True when the outputs relating the axis motion status are valid. If *Enable* is reset, the data loses its validity, and all outputs are reset. In this case, to update new motion state requires enabling this instruction again.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Executes the instruction when *Enable* changes to True.	BOOL	True/False (False)	-
Source	Selects the source of the relating data: Commanded Value: The value commanded by the controller. Actual Value: The actual value on the motion axis.	eMC_SOURCE*	1: mcCommandedValue 2: mcActualValue (0)	When *Enable* is rising-edge triggered and *Busy* is False.

***Note:** Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

● Outputs

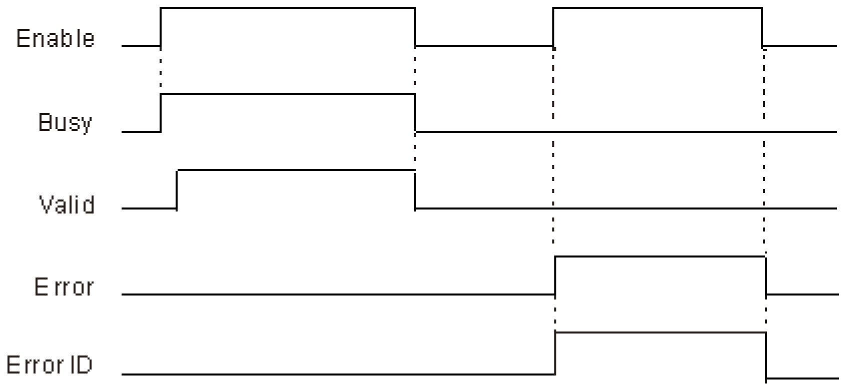
Name	Function	Data type	Output range (Default value)
Valid	True when the axis states at the output is available.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to	DWORD	16#0~16#FFFFFFFF (0)

Name	Function	Data type	Output range (Default value)
	Appendices for error code descriptions.		
ConstantVelocity	Indicates that the current velocity is constant. Velocity might be 0.	BOOL	True/False (False)
Accelerating	Indicates the absolute value of velocity is increasing.	BOOL	True/False (False)
Decelerating	Indicates the absolute value of velocity is decreasing.	BOOL	True/False (False)
DirectionPositive	Indicates the position is increasing.	BOOL	True/False (False)
DirectionNegative	Indicates the position is decreasing.	BOOL	True/False (False)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Valid	• When *Enable* shifts to True and the actual motion states at the output are available.	• When *Enable* shifts to False. • When *Error* shifts to True.
Busy	• When *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)
ConstantVelocity	• When the velocity is constant.	• When the velocity isn’t constant and *Enable* is still True.
Accelerating	• When the velocity is accelerating.	• When the velocity isn’t accelerating and *Enable* is still True.
Decelerating	• When the velocity is decelerating.	• When the velocity isn’t decelerating and *Enable* is still True.
DirectionPositive	• When the moving direction is positive.	• When the moving direction isn’t positive and *Enable* is still True. • When the axis is not moving and *Enable* is still True.
DirectionNegative	• When the moving direction is negative.	• When the moving direction isn’t negative and *Enable* is still True. • When the axis is not moving and *Enable* is still True.

■ **Timing Diagram**



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True.

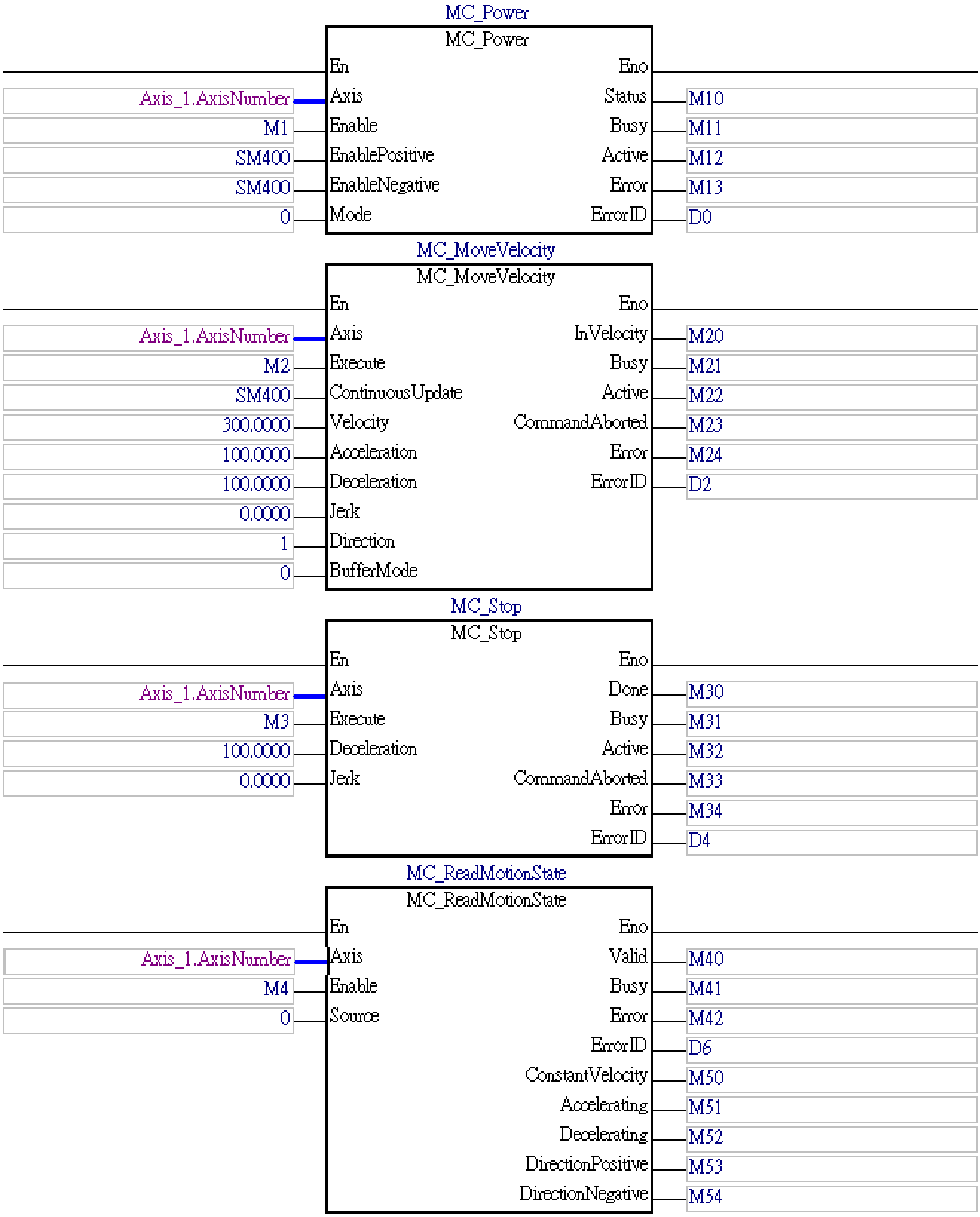
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Troubleshooting

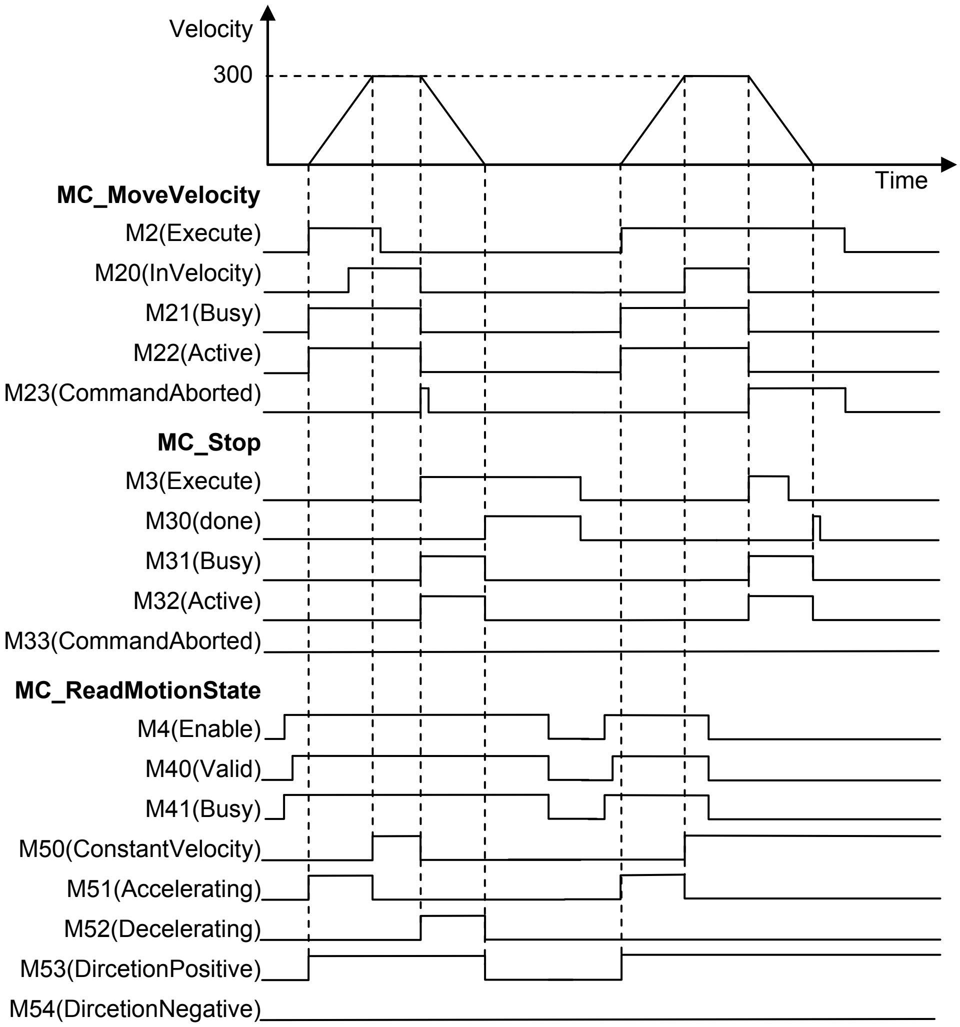
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

The MC_ReadMotionState instruction is used as follows. The example shows how MC_ReadMotionState indicates the motion behavior performed by MC_MoveVelocity and MC_Stop.



Motion diagram:



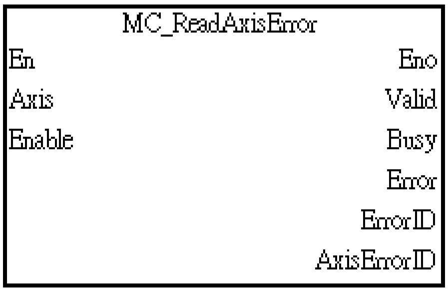
- When MC_ReadMotionState is enabled (M4=True), after that M40(*Valid*) change to True, indicating the motion state is available to be reported.
- When MC_MoveVelocity is executed (M2=True), the axis starts to accelerate to the designated target velocity. M51 (*Accelerating*) and M53 (*DirectionPositive*) change to True, indicating the axis is accelerating in positive direction.
- When the axis reaches the specified target velocity, it operates at a constant speed. M51 (*Accelerating*) changes to False and M50 (*ConstantVelocity*) changes to True.
- When MC_Stop is executed (M3=True), MC_MoveVelocity is aborted and the axis starts to decelerate to a stop. M50 (*ConstantVelocity*) changes to False and M52 (*Decelerating*) changes to True.
- When the axis reaches zero velocity, M52 (*Decelerating*) and M53 (*DirectionPositive*) change to False.
- In the second cycle of the motion behavior, MC_ReadMotionState is disabled (M4=False) while the axis is in constant velocity. In this case, outputs M50 (*ConstantVelocity*) and M53 (*DirectionPositive*) remain True and will not be updated no matter the commanded motion is completed or not.

- **Supported Products**

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_ReadAxisError

FB/FC	Description
FB	MC_ReadAxisError reads the error information of the axis



Note: Axis errors are the errors not relating to the instruction, such as drive errors and communication errors which could be displayed on the panel of the servo drive.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Executes the instruction when *Enable* changes to True.	BOOL	True/False (False)	-

Note: This instruction will read the axis state constantly when *Enable* changes to True.

● Outputs

Name	Function	Data type	Output range (Default value)
Valid	True when the axis error status at the output is available.	BOOL	True/False (False)
Busy	Indicates the instruction is enabled and there is incoming new error status.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
AxisErrorID*	Indicates the error code of servo drive when *Valid* shifts to True.	DWORD	16#0~16#FFFFFFFF (0)

***Note**:

Assume that 1xxx（hex）is indicated on the servo drive, xxx represents the error code of the servo drive. For example, if the servo drive shows AL3E3, *AxisErrorID* is13E3 (hex).

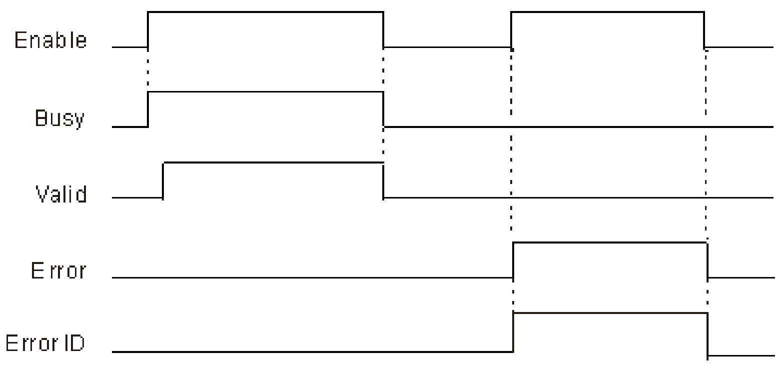
■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Valid	● When *Enable* shifts to True and the axis error status at the output is available	● When *Enable* shifts to False. ● When *Error* shifts to True.
Busy	● When *Enable* shifts to True.	● When *Enable* shifts to False. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction. (Error code is recorded)	● When *Enable* shifts from True to False. (Error code is cleared)
AxisErrorID*	Continuously updates value when *Valid* is True.	Continuously updates value when *Valid* is True.

***Note**: When *Enable* shifts to False, the *AxisErrorID* output remains unchanged.

3

■ Timing Diagram



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Enable* shifts to True.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

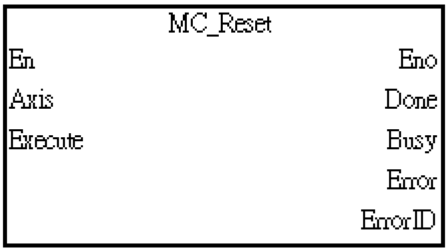
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_Reset

FB/FC	Description
FB	MC_Reset clears axis-related errors



MC_Reset is used to make the state transition from “ErrorStop” to “Standstill” or “Disabled” by clearing all axis-related errors while keeping the output of the function block instruction.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-

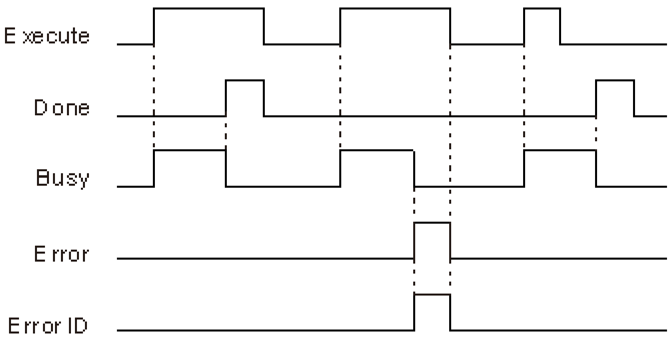
● Outputs

Name	Function	Data type	Output range (Default value)
Done	Indicates the completion of the axis error reset process, i.e. entering “Standstill” or “Disabled” state.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to t **Appendices** for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shiftng to True	Timing for shifting to False
Done	● When axis error reset process is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* shifts to True	● When *Done* shifts to True. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When Execute shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- MC_Reset instruction starts the error reset process for the specified axis in *Axis* when *Execute* shifts to True. It resets axis-related errors and drive errors.
- The instruction can be executed for any axis type.
- Only the axes with errors are applicable for the error reset process.
- If drive errors occur on an axis, the drive errors will be reset prior to the error reset process on the axis-related errors. The reset process for drive errors will continue until either the drive error is reset or the Drive Error Reset Monitoring Time in the axis parameters is reached.
- Errors that occur while error reset process is executing are not cleared. Only errors that existed while *Execute* shifts to True are cleared.

Troubleshooting

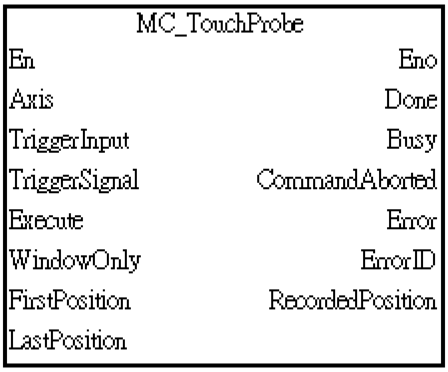
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_TouchProbe

FB/FC	Description
FB	MC_TouchProbe captures and records an axis position when a trigger event occurs.



Before using MC_TouchProbe instruction, it is required to confirm if the below object data is supported, and use ECAT Builder in ISPSoft to set the below data in the settings of PDO communications.

- Touch probe funciton (60B8 hex)
- Touch probe status (60B9 hex)
- Touch probe pos1 pos value (60BA hex)
- Touch probe pos1 neg value (60BB hex)
- Touch probe pos2 pos value (60BC hex)
- Touch probe pos2 neg valu (60BD hex)

If one of the above required object data is not set, an error will occur to indicate the problem of missing process data object setting.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
TriggerSignal	Specifies the trigger signal in controller mode*[1]	BOOL	True/False (False)	Continuously updates value when *Busy* isTrue
Execute	Executes the instruction and starts axis position recording when *Execute* changes to True.	BOOL	True/False (False)	-
WindowOnly	Specifies if the window mask will be applied or not. The window mask is defined by *FirstPosition* and *LastPosition*	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
FirstPosition	Defines the start position (positive direction) of the window mask to capture the trigger event (user unit*[2])	LREAL	Negative number, positive number, or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
LastPosition	Defines the stop position of the window mask to capture the trigger event (user unit*2)	LREAL	Negative number, positive number, or 0 (0)	When *Execute* shifts to True and *Busy* is False.

- **Outputs**

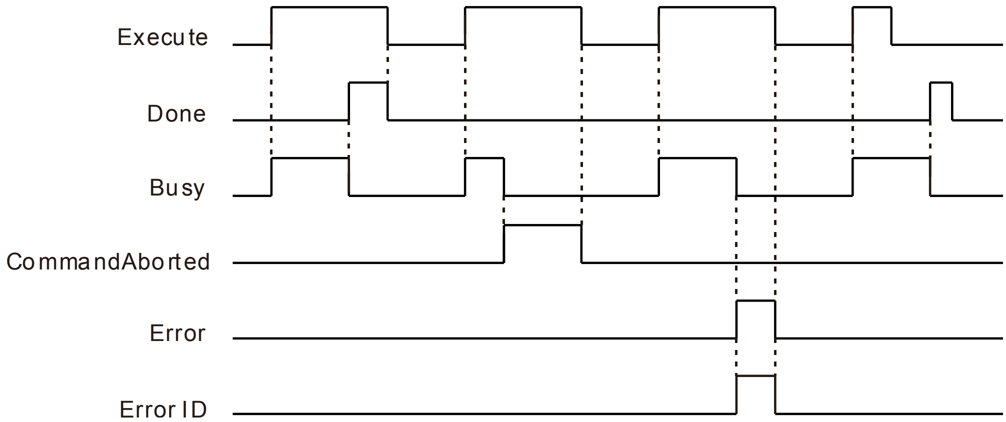
Name	Function	Data type	Output range (Default value)
Done	True when a trigger event is recorded	BOOL	True/False (False)
Busy	True when the instruction is executed. Awaits the completion of trigger event recording.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted (MC_AbortTrigger)	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
RecordedPosition	Indicates the position where trigger event is recorded (in user unit)	LREAL	Negative number, positive number, or 0 (0)

3

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the trigger event is recorded and the instruction is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* shifts to True and the instruction is executed.	• When *Done* shifts to True. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
CommandAborted	• When this instruction is aborted by another instruction (MC_AbortTrigger).	• When *Execute* shifts from True to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* shafts to True and *Busy* is False.
TriggerInput	Specifies the reference to the source of the trigger signal.	MC_TRIGGER_REF	Refer to the below table	When *Execute* is rising edge triggered and *Busy* is False.

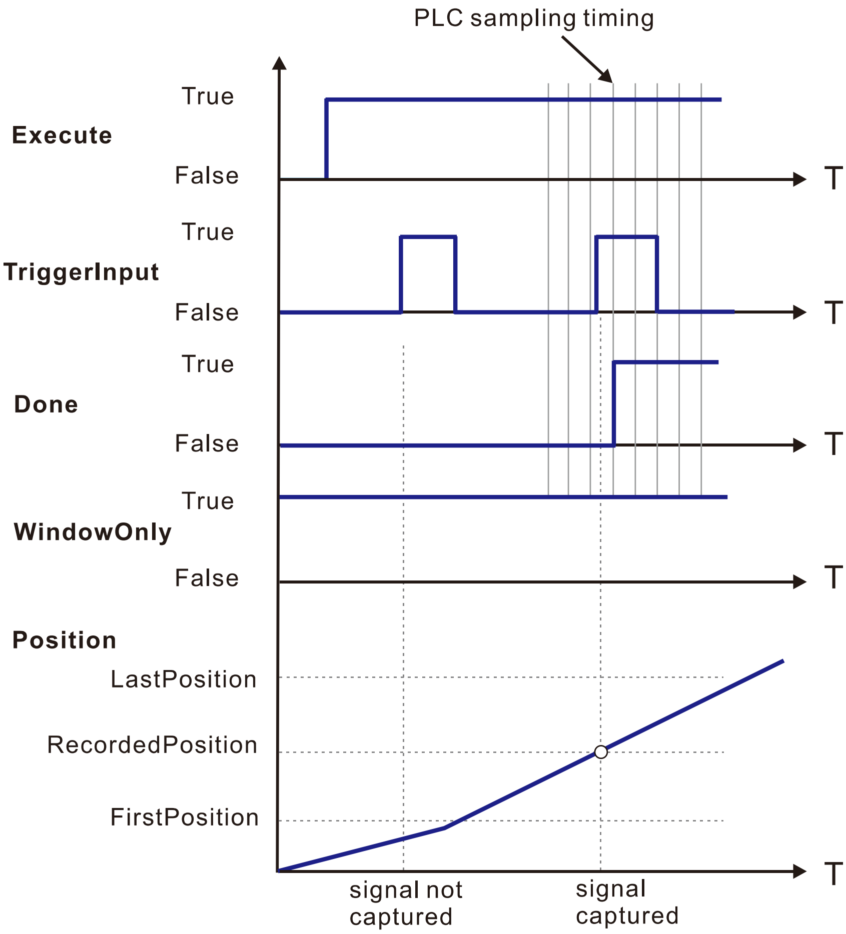
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

■ **MC_TRIGGER_REF**

Name	Data type	Setting value (Default value)	Function
Mode	WORD	K0~K1 (0)	Specifies the mode for triggering 0: Drive mode 1: Controller mode
TouchProbeID	WORD	K0~K1 (0)	Specifies which Capture is to be applied in Drive mode. 0: TouchProbe 1 1: TouchProbe 2
InputDrive	WORD	K0~K1 (0)	Specifies the trigger signal of the servo drive when Drive mode is selected. 0: Drive 1 1: Drive 2
Edge	WORD	K0~K1 (0)	Specifies the edge of the trigger signal in Drive mode. 0: Rising edge 1: Falling edge

Function

- The touch probe operation activates for only one time for recording the very first trigger signal after *Execute* is set as True. When a valid position is captured and recorded, the following trigger signals will be ignored.
- One function block instance should relate to only one MC_TouchProbe instruction.
- If there were multiple function block instances on the same capture and axis, the members of MC_TRIGGER_REF should be added with TouchProbeID, which identifies different TouchProbe actions. The definition of TouchProbeID can be associated to MC_AbortTrigger.
- The operation of MC_TouhcProbe with window mask function is demonstrated as below:



- At the first activation of the trigger input signal, the signal is not accepted because the axis position hasn't reach the specified window mask section.
- When the axis position enters the window mask section, the second activation of the trigger input signal is accepted, and after a period *Done* chnages to True.
- Since Delta A2-E only supports one TouchProbe at the same time, TouchProbe ID can only be 0 when using Delta A2-E along with other funciton blocks. For the limitations of other brand's servo drives, refer to its manual for more details.
- When using Delta A2-E to execute this instruciton, InputDrive=0 indicates D13 is being used as a contact for triggering and InputDrive=1 indicates the servo drive's encorder z is being used as a contact for triggering.

Note:

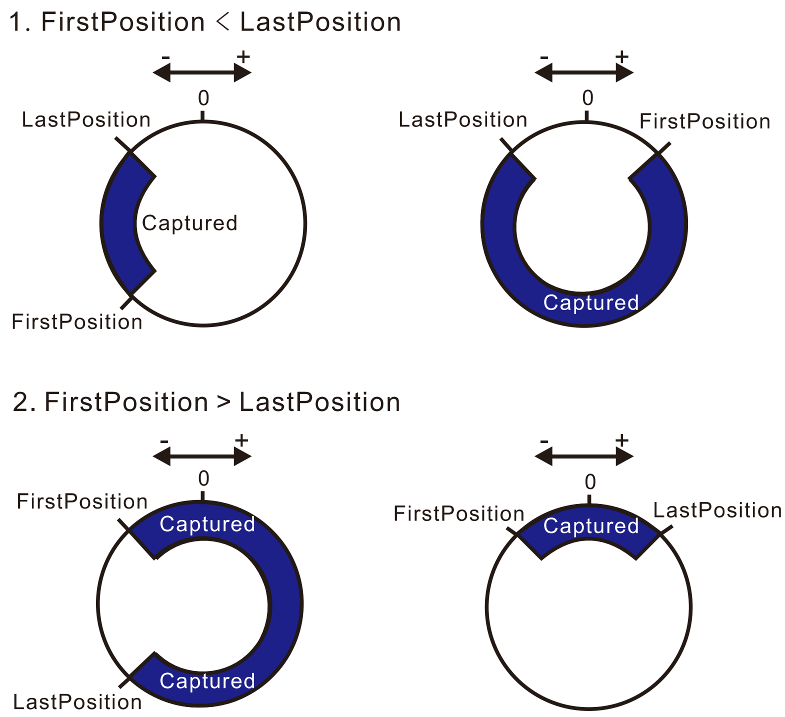
1. Time is needed until the touch probe operation is actually activated. The touch probe operation is not possiblly

to be activated immediately after *WindowOnly* shifts to True and until the touch probe operation is actually activated.

2. If the window mask is too small, the touch probe operation is not possble. The effective range for the window mask depends on EtherCAT communications and the performance of encoder input or the servo drive.

■ Window Mask Settings

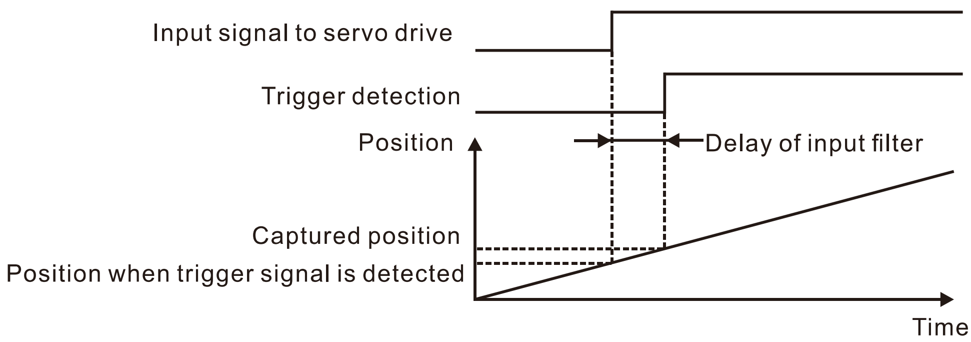
- You can observe the results of different window mask settings when the instruction is used for rotary/modulo axes as below. The difference is resulted from the set values between *FisrPosition* and *LastPosition*.



■ *TriggerInput* Settings

- **Drive Mode**

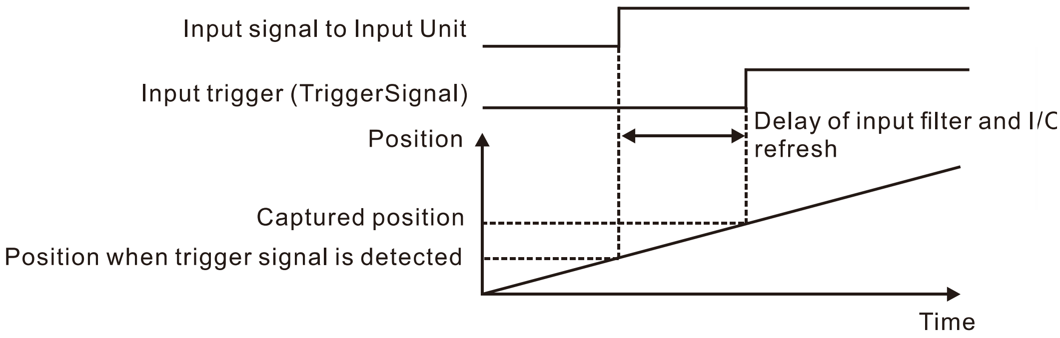
The captured position in Drive Mode is more accurate than it is in Controller Mode since the Drive Mode references the actual position of the servo drive for the capture operation.



- **Controller Mode**

The captured position in Controller Mode is less accurate than it is in Drive Mode due to the scan time (I/O refresh dalay) of the controller.

1. In Controller Mode, you can declare a BOOL variable to be the trigger input signal.
2. The BOOL variable funcitons as trigger input signal can be specified for *TriggerSignal*.



Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

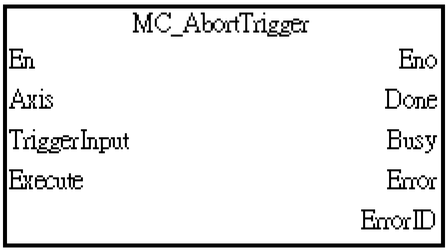
3

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_AbortTrigger

FB/FC	Description
FB	MC_AbortTrigger aborts the instruction MC_TouchProbe which are intended to capture trigger events



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction and aborts trigger events.	BOOL	True/False (False)	-

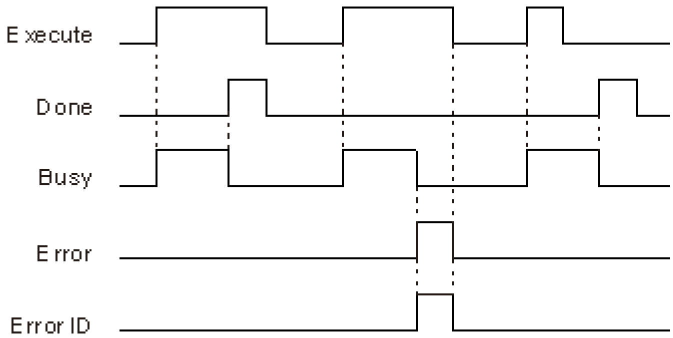
● Outputs

Name	Function	Data type	Output range (Default value)
Done	True when trigger event is aborted	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the capture operation is stopped. ● When the instruction is executed on a capture operation which is not in execution.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* shifts to True and the instruction is executed.	● When *Done* shifts to True. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* shafts to True and *Busy* is False.
TriggerInput	Specifies the reference to the source of the trigger signal.	MC_TRIGGER_REF	Refer to the below table	When *Execute* is rising edge triggered and *Busy* is False.

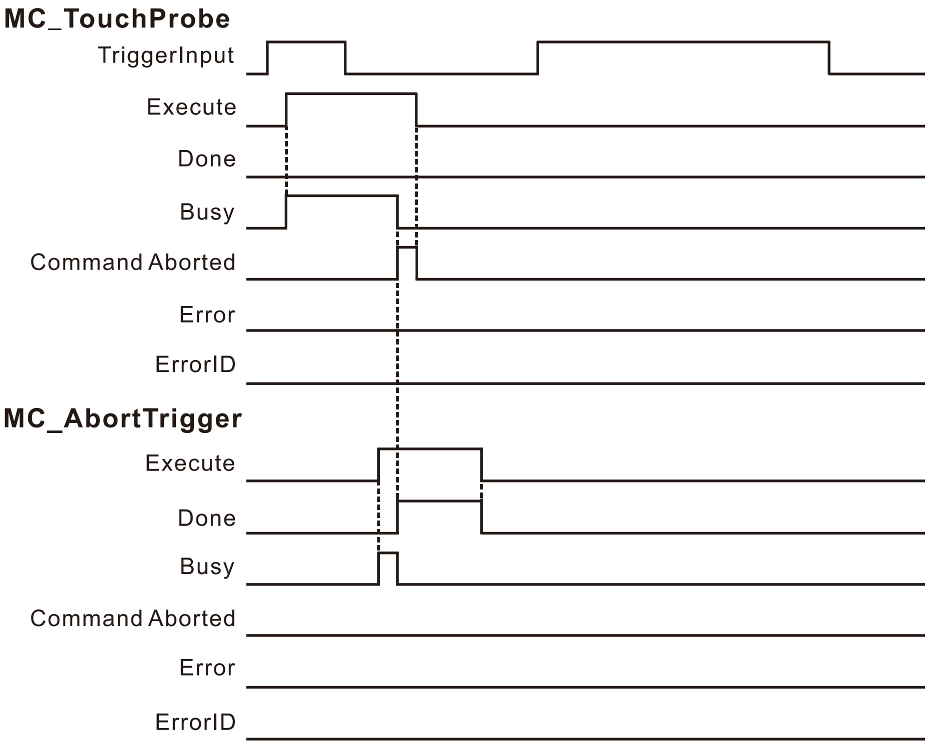
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

■ MC_TRIGGER_REF

Name	Data type	Setting value (Default value)	Function
Mode	WORD	K0~K1 (0)	Specifies the mode for triggering 0: Drive mode 1: Controller mode
TouchProbeID	WORD	K0~K1 (0)	Specifies which Capture is to be applied in Drive mode. 0: TouchProbe 1 1: TouchProbe 2
InputDrive	WORD	K0~K1 (0)	Specifies the trigger signal of the servo drive when Drive mode is selected. 0: Drive 1 1: Drive 2
Edge	WORD	K0~K1 (0)	Specifies the edge of the trigger signal in Drive mode. 0: Rising edge 1: Falling edge

Function

- You can cancel the touch probe operation by using MC_AbortTrigger.
- By setting *Axis* and *TriggerInput* for this instruction you can define the touch probe operation to abort.
- When MC_AbortTrigger is executed on an axis without a touch probe request, MC_AbortTrigger will not do anything and will complete normally. The same behavior also applies when MC_AbortTrigger is executed for a MC_TouchProbe instruction with the condition of “Done=True”.
- The operation of the combination of MC_AbortTrigger and MC_TouchProbe is demonstrated as below. *Done* of MC_AbortTrigger shifts to True for one period after *Execute* shifts to True.



Troubleshooting

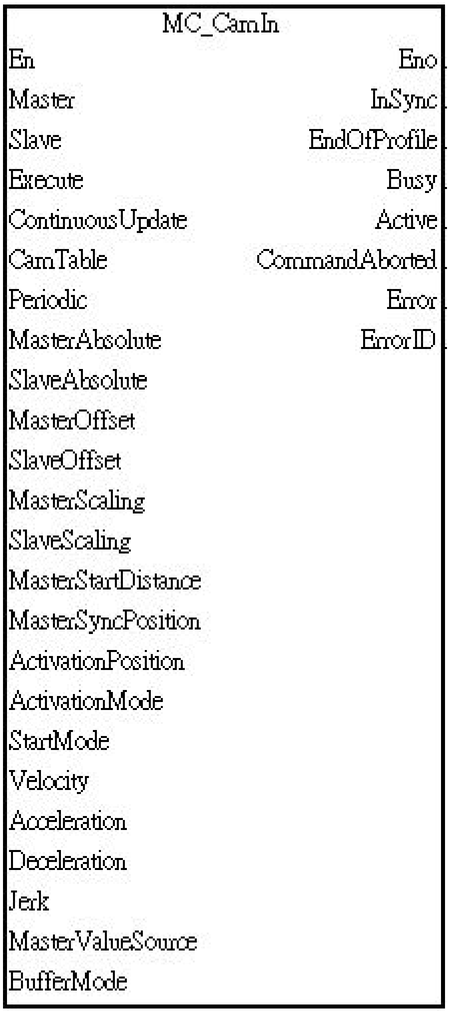
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_CamIn

FB/FC	Description
FB	MC_CamIn performs cam operation by engaging the cam.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates CamTable, MasterScaling, SlaveScaling when Continuousupdate is True	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.

Name	Function	Data type	Setting value (Default value)	Timing for updating
CamTable	Specifies a Cam table	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.
Periodic	Specifies a periodical execution or a one-time operation. True: Periodic; False: Non-periodic.	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
MasterAbsolute	Specifies the positioning mode of the master axis. True: Absolute mode False: Relative mode	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
SlaveAbsolute	Specifies the positioning mode of the slave axis. True: Absolute mode False: Relative mode	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
MasterOffset	Shifts the position of the master axis by the specified offset value.	LREAL	positive number, negative number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
SlaveOffset*2	Shifts the displacement of the slave axis by the specified offset value.	LREAL	positive number, negative number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
MasterScaling	Scales the master axis up and down with the specified factor.	LREAL	positive number (0)	When *Execute* shifts to True and *Busy* is False.
SlaveScaling	Scales the slave axis up and down with the specified factor.	LREAL	positive number (0)	When *Execute* shifts to True and *Busy* is False.
MasterStartDistance	Reserved	-	-	-
MasterSyncPosition	Specifies the relative position of the master axis when it starts to engage	LREAL	positive number, negative number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
ActivationPosition	Specifies the master position when it starts to engage and from	LREAL	positive number, negative number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
	there the slave will start to engage.			
ActivationMode	Specifies the mode of engement 0: Relative 1: Absolute	MC_ACTIVATION_MODE*1	0: Relative 1: Absolute (0)	When *Execute* shifts to True and *Busy* is False.
StartMode	Specifies the engagement behavior of the slave axis	eMC_START_MODE*1	0: mcJump 1: mcRampIn_Shortest 2: mcRampIn_Positive 3: RampIn_Negative (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	The maximum velocity for the engaging behavior specified by *StartMode*. (Unit: user unit/s) *	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	The maximum acceleration rate for the engaging behavior specified by *StartMode.* (Unit: user unit/s) *	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	The maximum deceleration rate for the engaging behavior specified by *StartMode.* (Unit: user unit/s^2) *2	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	The maximum jerk value for the engaging behavior specified by *StartMode.*. (Unit: user unit/s^3) *2	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
MasterValueSource	Specifies the reference position of the master axis.	eMC_SOURCE*1	0: mcCommandedValue 2: mcActualValue (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*1	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
2. The input Slaveoffset only works when the master axis is in absolute mode.

Outputs

Name	Function	Data type	Output range (Default value)
InSync	True when the specified master/slave cam operation is synchronized.	BOOL	True/False (False)
EndOfProfile	Indicates the end point of the cam profile is completed. (Resets automatically)	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

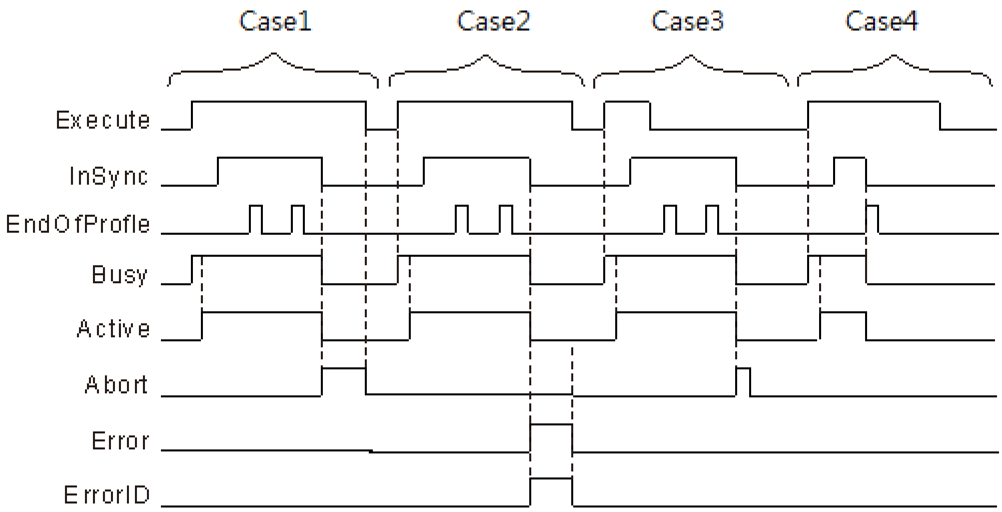
Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
InSync	• When the specified master/slave cam operation is synchronized.	• When *Periodic* is False and *EndOfProfile* shifts to True. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
EndOfProfile*	• When the end point of the cam profile is completed.	• After *EndOfProfile* shifts to True for One period.
Busy	• When *Execute* changes to True.	• When *Execute* shifts to False. • When *Error* shifts to True. • When *CommandAborted* shifts to True.
Active	• When the motion on the axis is started.	• When *Error* shifts to True • When *CommandAborted* shifts to True.
CommandAborted	• When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* shifts to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.

Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

***Note:** *EndOfProfile* shifts to True for one period when the desired cam engagement operation for the phase and displacement is completed and reached the end point of the cam table. *EndOfProfile* functions as an indicating signal of the end of the cam table.

■ Timing Diagram:



Case1

When *Execute* shifts to True and *Busy* is True. After one cycle, *Active* shifts to True. When the slave axis and the master axis are synchronized, *InSync* shifts to True. When the CamIn is at the final stop of the cycle, *EndOfProfile* shifts to True and after one cycle is complete, *EndOfProfile* shifts to False. When the relationship between the master axis and slave axis changes, for example executing MC_CamOut instruction, *CommandAborted* shifts to True while *InSync, Busy, and Active* shift to False. And then *Execute* shifts to False and *CommandAborted* shifts to False.

Case2

When there is an error occurred during the execution of an instruction, *Error* shifts to True and the *ErrorIDs such as InSync, Busy, and Active* shift to False. And then *Execute* shifts to False and *Error* shifts to False. The value in *ErrorID* is 0.

Case3

During the execution of an instruction, if *Execute* shifts to False, the instruction will still be executed and *InSync, EndOfProfile, Busy, and Active* will not be affected either. But after the relationship between the master axis and slave axis changes, *InSync, Busy, and Active* will shift to False. *CommandAborted* shifts to True and after one cycle, *CommandAborted* shifts to False.

Case4

If the CamIn is not executed cyclically (*Periodic*=FASLE), when the CamIn is at the final stop of the cycle, *EndOfProfile* shifts to True and *InSync, Busy, and Active* shift to False. After one cycle, *EndOfProfile* shifts to False.

- **In-Outs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Master	Master Axis 1: 1st axis position 2: 2nd axis position 3: 3rd axis position 4: 4th axis position 5: 5th axis position 6: 6th axis position 7: 7th axis position 8: 8th axis position 9: 9th axis position 10: 10th axis position 11: 11th axis position 12: 12th axis position 13: 13th axis position 14: 14th axis position 15: 15th axis position 16: 16th axis position 17: 17th axis position 18: 18th axis position 19: 19th axis position 20: 20th axis position 21: 21st axis position 22: 22nd axis position 23: 23rd axis position 24: 24th axis position 25: 25th axis position 26: 26th axis position 27: 27th axis position 28: 28th axis position 29: 29th axis position 30: 30th axis position 31: 31st axis position 32: 32nd axis position 200: value of the 1st counter 204: value of the 2nd counter 208: value of the 3rd counter 212: value of the 4th counter 216: value of the 5th counter 220: value of the 6th counter	eMC_Master_SOURCE*1	1: Axis1_Cmd 2: Axis2_Cmd 3: Axis3_Cmd 4: Axis4_Cmd 5: Axis5_Cmd 6: Axis6_Cmd 7: Axis7_Cmd 8: Axis8_Cmd 9: Axis9_Cmd 10: Axis10_Cmd 11: Axis11_Cmd 12: Axis12_Cmd 13: Axis13_Cmd 14: Axis14_Cmd 15: Axis15_Cmd 16: Axis16_Cmd 17: Axis17_Cmd 18: Axis18_Cmd 19: Axis19_Cmd 20: Axis20_Cmd 21: Axis21_Cmd 22: Axis22_Cmd 23: Axis23_Cmd 24: Axis24_Cmd 25: Axis25_Cmd 26: Axis26_Cmd 27: Axis27_Cmd 28: Axis28_Cmd 29: Axis29_Cmd 30: Axis30_Cmd 31: Axis31_Cmd 32: Axis32_Cmd	When *Execute* is rising edge triggered and *Busy* is False.
Slave	Slave Axis*2	WORD	K1~Kn (0)	When *Execute* is rising edge triggered and *Busy* is False.

Note:

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
2. If you specify the same axis number for both master and slave axis, a Master and Slave Defined as Same Axis error will occur.

E-CAM

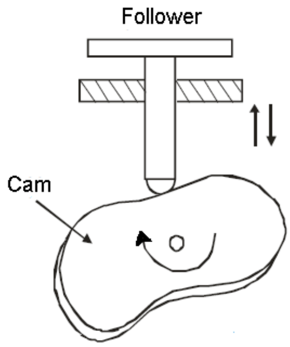
A traditional mechanical cam is composed of a cam, a follower, and a support.

1. A mechanical cam is an input object with irregular shape. It makes a follower move regularly by coming into contact with the follower.

2. When a cam rotates, the follower will go move according to the shape of the cam. Cam followers are available in several designs and variants to meet the requirements of different applications.

3. A support is for supporting a cam and a follower.

As the image shown, for an operation of motion control, the relative movements between axes are achieved by the rotation of a cam.



Use a cam chart to define the relation between a follower and a cam to simulate the movements of a cam controlled by a PLC, making a slave axis to move by the master axis according to their defined relation.

The benefits of using a E-CAM are:

1. Friendly user interface

2. Users can modify the electronic cam data in an electronic cam in software. Users do not need to modify the mechanical design and no depreciation costs.

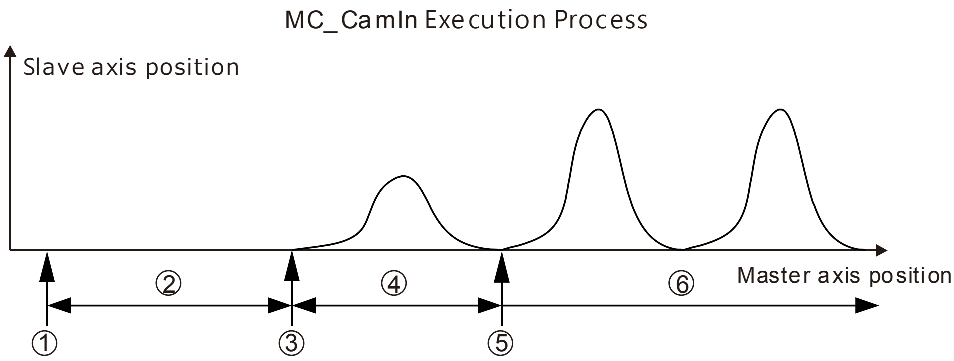
3. Higher acceleration

4. Smoother operation

Function

MC_CamIn instruction is used for using a cam chart to define the relation between a follower and a cam to simulate the movements of a cam controlled by a PLC, making a slave axis to move by the master axis according to their defined relation. And MC_CamOut is used for stopping the relation between a follower and a cam.

- MC_CamIn execution process:



Stage 1: Trigger and execute the MC_CamIn instruction.

Stage 2: Wait for the start of the engagement.

Stage 3: The slave axis starts to perform the engagement action as the master axis reaches the position where the engagement starts.

Stage 4: The engagement is ongoing.

Stage 5: The master axis and slave axis achieve the synchronization as the engagement is completed.

Stage 6: The master axis and slave axis are in the synchronous motion.

Stage 1: Trigger and execute the MC_CamIn instruction.

The *MC_CamIn* instruction is executed at this time and then the slave will enter the state of waiting for the start of the engagement immediately.

Note: The slave axis will stop at once and vibration may occur if the slave axis is in the motion at the moment. All set input parameters of the *MC_CamIn* instruction will be read and retained for use in the execution.

Stage 2: Wait for the start of the engagement.

The slave axis waits for the timing for performing the engagement action in the standstill state. The time to start the engagement is when the master axis passes the position specified by the parameter *ActivationPosition*. In different circumstances, the period of time the slave axis waits for is different. If the master axis is at the position specified by *ActivationPosition* as the *MC_CamIn* instruction is executed, the slave axis starts the engagement action immediately. If the master axis never reaches the position specified by *ActivationPosition,* the slave axis will never start to perform the engagement action and the cam synchronization will never come true. The parameters *ActivationPosition* and *ActivationMode* are used at this stage.

Stage 3: The slave axis starts to perform the engagement action when the master axis passes the position specified by *ActivationPosition.* The parameters, *MasterAbsolute, SlaveAbsolute*, *MasterOffset*, *SlaveOffset*, *MasterScaling* and *SlaveScaling* will work at the moment for making sure of the corresponding relationship between the master axis position and slave axis position and the cam phase.

Stage 4: The engagement is ongoing.

The slave axis performs the engagement in the way specified by the *StartMode* parameter. Besides *StartMode,* the parameters *Velocity, Acceleration* and *Deceleration* also works at this stage. The motion features about velocity, acceleration/ deceleration of the slave axis are determined by these parameters in the engagement.

Stage 5: The engagement is completed and the master axis and slave axis achieve the synchronization.

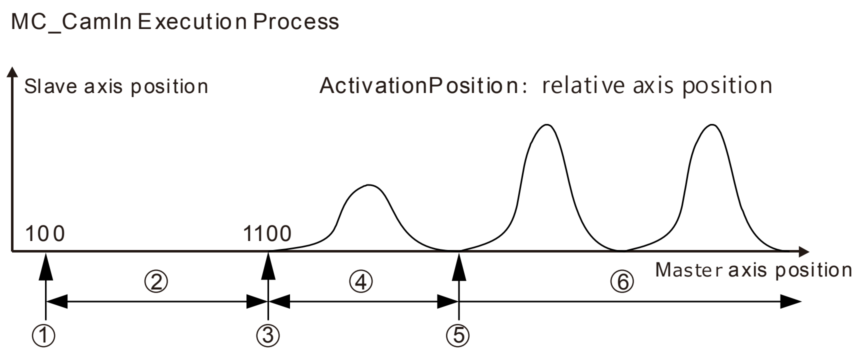
The engagement is completed and the slave axis and master axis achieve the cam synchronization if the cam phase that the master axis and slave axis correspond to meets the planned cam relationship after the slave axis starts to perform the engagement action.

Note: In the figure above, the set master axis position at the time when the engagement begins is greater than the master position at the time when the *MC_CamIn* instruction execution starts. The similar way is also applied to the circumstance that the set master axis position at the time when the engagement begins is less than or equal to the master position at the time when the *MC_CamIn* instruction execution starts.

- **ActivationPosition/ActivationMode**

ActivationMode=0；ActivationPosition, **Relative axis position**

When ActivationMode=0 (Relative), ActivationPosition is the axis positon. The setting of ActivationPosition is an axis position which is relative to the master axis position at the time when the MC_CamIn instruction is executed. The master axis position as the actual engagement starts is the value of ActivationPosition plus the master position of when the MC_CamIn instruction execution begins. For example: The master axis position is 100 and ActivationPosition 1000 at the time when the MC_CamIn instruction execution starts. The master axis position is 1100 (1100=100+1000) as the actual engagement begins.



3

Stage 1: Trigger and execute the MC_CamIn instruction. The master axis absolute position is 100 at the moment.

Stage 2: Wait for the start of the engagement.

Stage 3: The master axis reaches the position for starting the engagement (1100) and the slave axis starts to perform the engagement action.

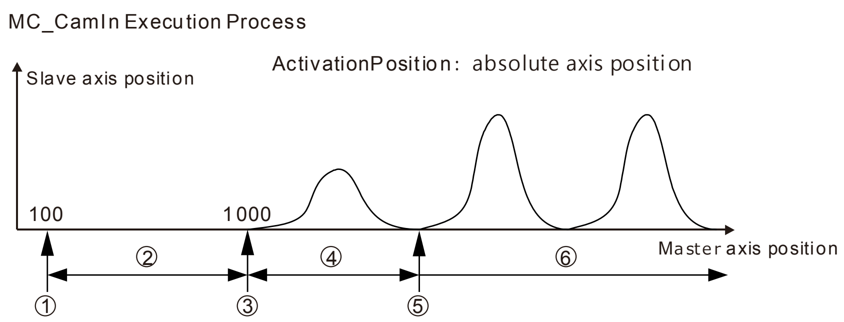
Stage 4: The engagement is ongoing.

Stage 5: The engagement is completed and the master axis and slave axis achieve the synchronization.

Stage 6: The master axis and slave axis are in the synchronous motion.

- ***ActivationMode=1；ActivationPosition*, Absolute axis position**

When ActivationMode =1, ActivationPosition is an axis position which is absolute to the master axis position at the time when the MC_CamIn instruction is executed. The master axis position as the actual engagement starts is ActivationPosition.
For example: The master axis position is 100 and ActivationPosition 1000 at the time when the MC_CamIn instruction execution starts. The master axis position is 1000 (1000= ActivationPosition) as the actual engagement begins.



Stage 1: Trigger and execute the MC_CamIn instruction. The master axis absolute position is 100 at the moment.

Stage 2: Wait for the start of the engagement.

Stage 3: The master axis reaches the position for starting the engagement (1000) and the slave axis starts to perform the engagement action.

Stage 4: The engagement is being conducted.

Stage 5: The engagement is completed and the master axis and slave axis achieve the synchronization.

Stage 6: The master axis and slave axis are in the synchronous motion.

■ Relationship between master axis position and slave axis position

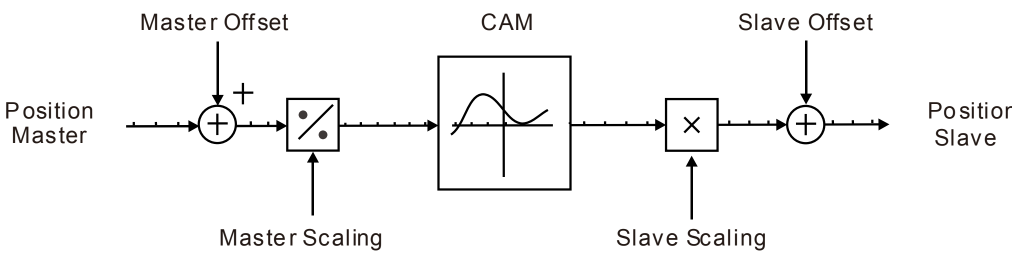
The cam relationship which is planned in the software is the position relationship between the master axis and slave axis. The “position” mentioned here is the cam phase of the master axis / slave axis instead of the actual axis position. If the cam relationship which is planned is seen as the function CAM as below, the input of the function CAM is the master axis cam phase and the output is the slave axis cam phase. The formula is shown as below.

y = CAM (x)

x：The master axis cam phase

y：The slave axis cam phase

The cam phase comes from the axis positions and there is a conversion between them. The conversion between the axis position and cam phase is related with the MasterAbsolute, SlaveAbsolute, MasterOffset, SlaveOffset, MasterScaling and SlaveScaling parameters. For details, refer to relevant sections. The slave axis follows the master axis to make the synchronous cam motion by using the MC_CamIn instruction. In the synchronous cam motion, the corresponding relationship between the master axis position and slave axis position is based on the pre-planned cam relationship (the cam curve or cam table). The process in which the slave axis position is calculated through the master axis position is illustrated as follows.



- Slave position = f_{cam}[(master position + master offset) / master scaling]* slave scaling + slave offset

Method of calculating the master position in the above formula:

When master axis is in absolute mode, master position is the remainder of the current master position divided by modulo; When master axis is in relative mode, master position is the start point position (usually 0) of master axis in the corresponding cam curve.

" f_{cam} " in above formula represents the cam curve relationship between master axis and slave axis.

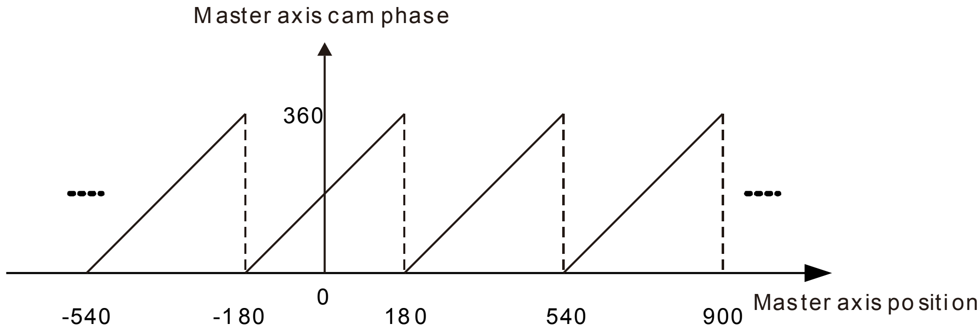
■ ***MasterAbsolute* and *SlaveAbsolute***

MasterAbsolute and *SlaveAbsolute* work at the moment when the engagement starts. That is to say that the corresponding relationship between the axis position and cam phase is built at the beginning of the engagement. (**Note:** The corresponding relationship is not built at the time when the *MC_CamIn* instruction execution begins but when the engagement begins.) After that, the cam phase is calculated according to the corresponding relationship.

- ***MasterAbsolute=False (Relative mode)***

  The master axis position and its cam phase are in the relative relationship as the *MasterAbsolute* parameter is FALSE. That is to say, the master axis position corresponds to its cam phase 0 at the time when the engagement starts. After that, the master cam phase will be calculated according to the corresponding relationship. For example, the master axis is in relative mode, the maximum value of the master axis cam phase in the cam relationship is 360 and the master axis position is 180 at the time when the engagement starts. So the master axis position 180 corresponds its cam phase 0; the master axis position 200 corresponds to its cam phase 20 (20= (200-180) %360) and so on.

  In this circumstance, the master axis position corresponds to its cam phase as shown in the following figure.

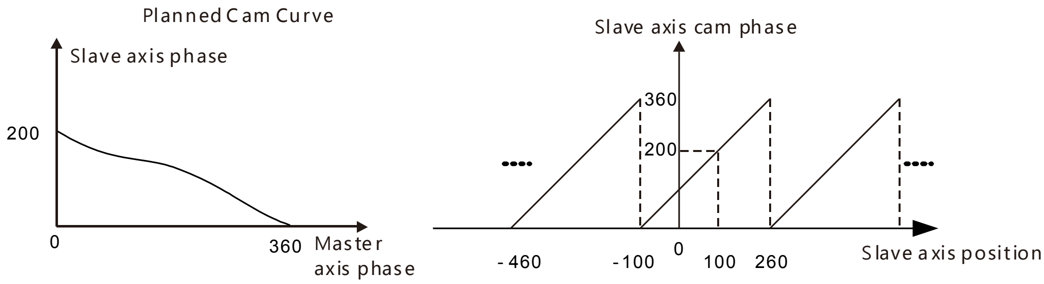


As the *SlaveAbsolute* parameter is FALSE, the slave axis position and its cam phase are in the relative relationship. That is to say, the slave axis cam phase and the master axis cam phase meet the planned cam relationship at the time when the engagement starts. If the slave axis is in relative mode, the method of being sure of the slave axis cam phase is different from the master axis. When the slave axis cam phase is sure, it should meet the condition that the slave axis cam phase and the master axis cam phase meet the planned cam relationship at the time when the engagement starts.

For example, the slave axis is in relative mode, the maximum value of the slave axis cam phase in the cam relationship is 360 and the slave axis position is 100 at the time when the engagement starts. If the master axis cam phase is 0 at the moment (and the slave axis cam phase is 0 as required in the cam relationship), the slave axis position 100 will correspond to its cam phase 0 as shown in the following circumstance 1. If the slave axis cam phase is 200 as required in the cam relationship, the slave axis position 100 will correspond to its cam phase 200 as shown in the following circumstance 2.

Case 1

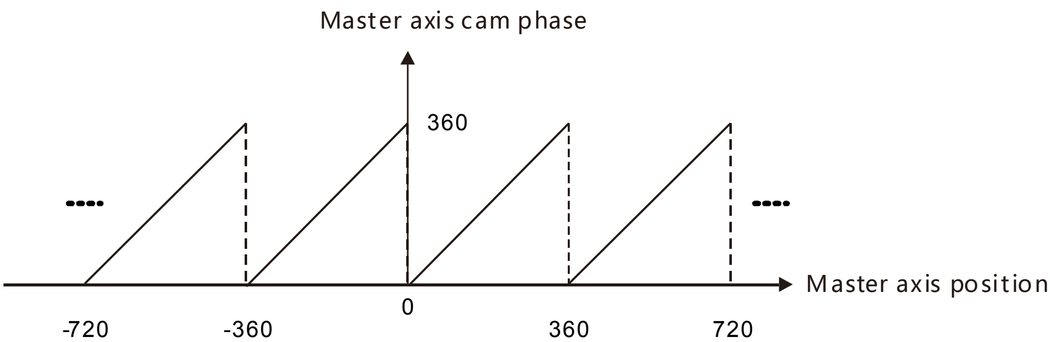
Case 2



MasterAbsolute=True (Absolute mode)

When the MasterAbsolute parameter is TRUE, the master axis position and its cam phase are in the absolute relationship. At any time, the master axis cam phase is equal to the remainder got by dividing the master axis position at that time by the maximum value of the master axis cam phase in the cam relationship.

For example, the master axis is in absolute mode and the maximum value of the master axis in the cam relationship is 360. So its cam phase is 100 as the master axis position is 100 (100=100%360); its cam phase is 140 (140=500%360) as the master axis position is 500 and so on. The master axis position corresponds to its cam phase as shown in the figure below.



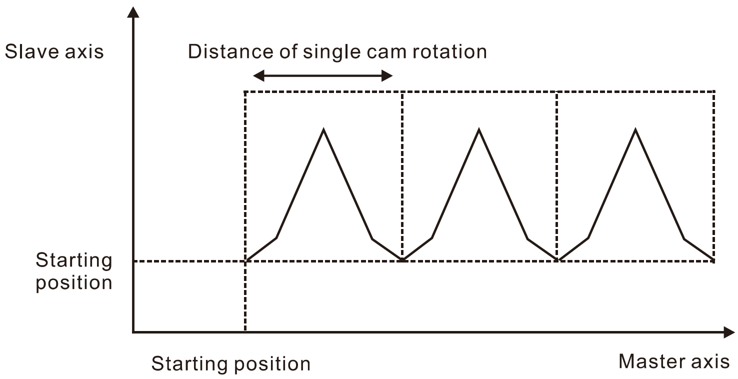
When the SlaveAbsolute parameter is TRUE, the slave axis position and its cam phase are in the absolute relationship. At any time, the slave axis cam phase is equal to the remainder got by dividing the slave axis position at that time by the maximum value of the slave axis cam phase in the cam relationship. When the slave axis is in absolute mode, the corresponding relationship between the slave axis position and its cam phase is consistent with that between the master axis position and its cam phase when the master axis is in absolute mode.

■ Periodic Operation

- If you specify True (periodic) for *Periodic*, the cam motion will be repeated from the start to the end point of the cam table. If you specify False (non-periodic), the cam operation ends when the last point in the cam table is executed.

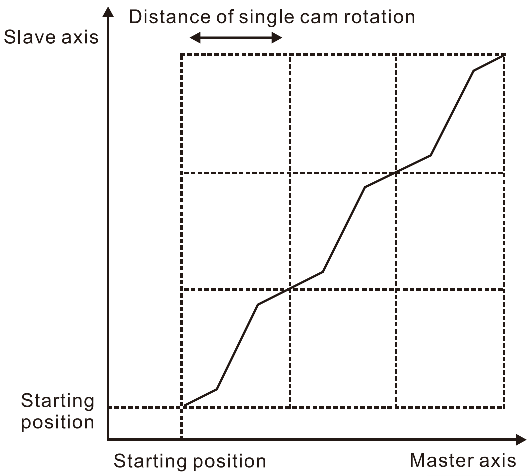
- If the stroke position of the slave axis is the same at the start and end points of the cam table when True (periodic) is set, the cam operates as a reciprocal cam. If the stroke position of the slave axis differs at the start point and end point, the cam operates as a feeding cam. In the following chart, the horizontal axis indicates the master axis and the vertical axis indicates the slave axis.

Rotary/reciprocal Cam Operation



3

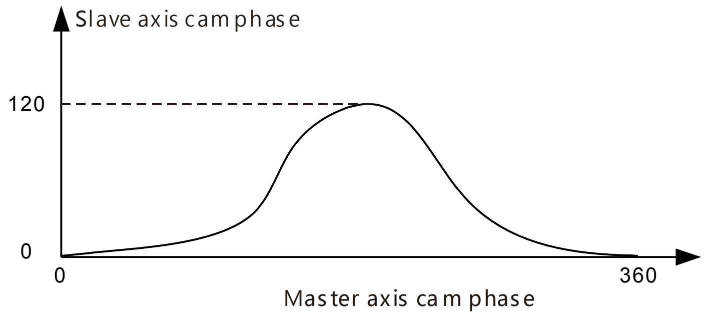
Feeding Cam Operation



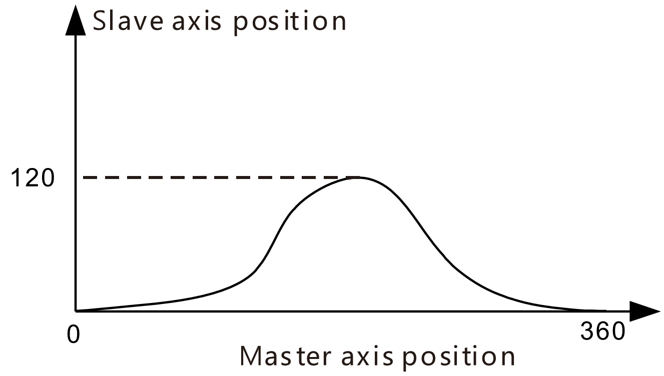
■ ***MasterOffset/MasterScaling/SlaveOffset/Slavescaling***

The cam relationship between the master axis and slave axis is preplanned. But as the cam motion is executed, the position offset or scaling based on the preplanned cam relationship can be performed through setting the *Offset* and *Scaling* parameters. For example, there are various sizes for the same product which is processed. Just one cam relationship need be planned and then changing the values of *Offset* and *Scaling* fits the processing of products of different sizes.

The position offset and scaling of the master axis and slave axis determine the actually executed cam relationship. The effect is described in the following example. The planned cam relationship is shown as the figure below.

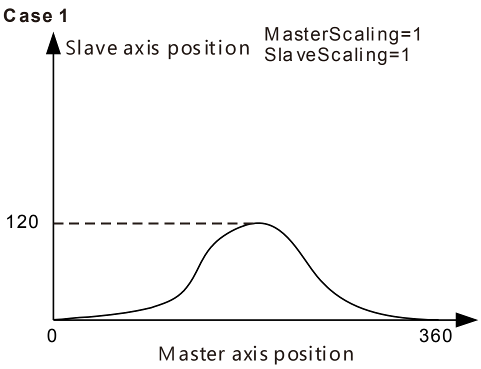


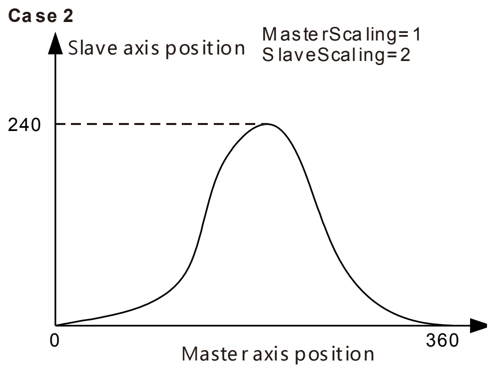
When the master axis and slave axis are both in absolute mode and the engagement begins, the master axis position and slave axis position are both 0. When there is no position offset and scaling (the offset and scaling are default values), the actual master axis position correspond to the actual slave axis position in the execution of the cam motion as shown in the following figure.

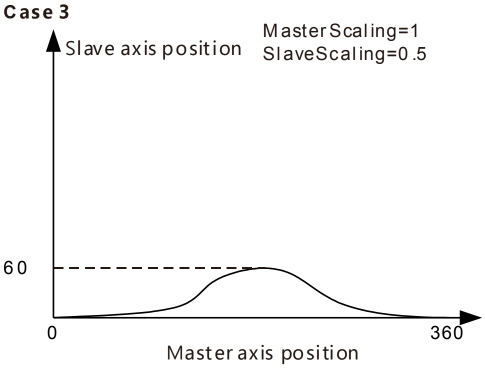


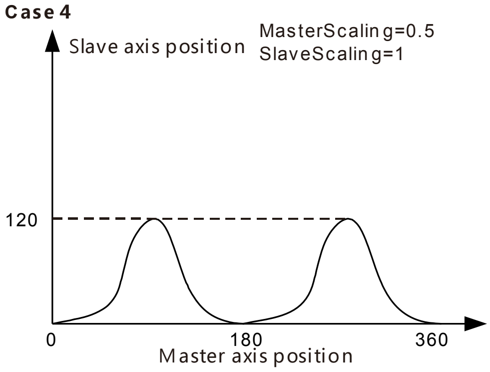
When the offset and scaling are not default values, the corresponding relationship between the actual master axis position and actual slave axis position are affected in the execution of the cam motion as below.

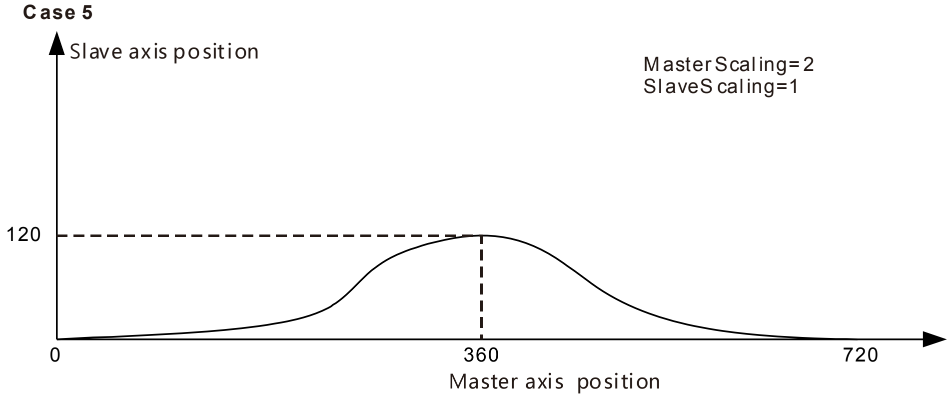
- **MasterOffset:0 and SlaveOffset:0 and the impact of MasterScaling and SlaveScaling on the cam relationship**











Case 1: The actual cam relationship is consistent with the preplanned one as the values of MasterScaling and SlaveScaling are 1 and their offsets are 0.

Case 2: The slave position corresponding to the master axis position is two times what is planned in the cam relationship as the value of *MasterScaling* is 1, *SlaveScaling* is 2 and their offsets are 0.

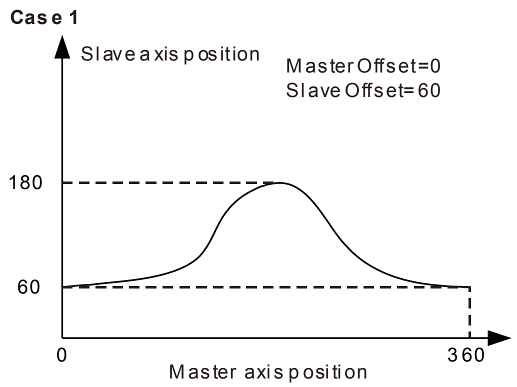
Case 3: The slave position corresponding to the master axis position is 1/2 that in the planned cam relationship as the value of MasterScaling is 1, SlaveScaling is 0.5 and their offsets are 0.

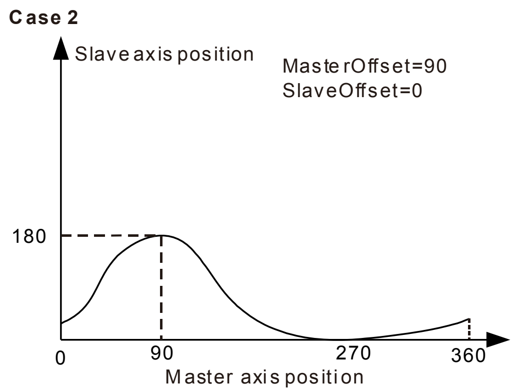
Case 4: The master axis position corresponding to the slave axis position is 1/2 what is planned as the value of *MasterScaling* is 0.5, *SlaveScaling* is 1 and their offsets are 0. If it is observed from the perspective of the cam phase, the master axis cam phase is 1/2 what is preplanned. That is, the master cam cycle changes from 360 to 180 (180=360*0.5) and the slave axis cam phase is unchanged.

Case 5: The master axis position corresponding to the slave axis position is 2 times what is planned as the value of *MasterScaling* is 2, *SlaveScaling* is 1 and their offsets are 0. If it is observed from the perspective of the cam phase, the master axis cam phase is two times the original. That is, the master axis cam cycle changes from 360 to 720 (720=360*2) and the slave axis cam phase is unchanged.

◆ **MasterScaling:1 and SlaveScaling:1 and the impact of MasterOffset and SlaveOffset on the actually executed cam relationship**

MasterOffset means to make the actual axis position curve shifted horizontally in execution of the cam motion. *SlaveOffset* indicates to make the axis position curve shifted vertically in execution of the cam motion.





Case 1： The slave axis position corresponding to the master axis position will add by 60 based on the planned position as *MasterScaling* and *SlaveScaling* are both 1, *MasterOffset* is 0 and *SlaveOffset* is 60.

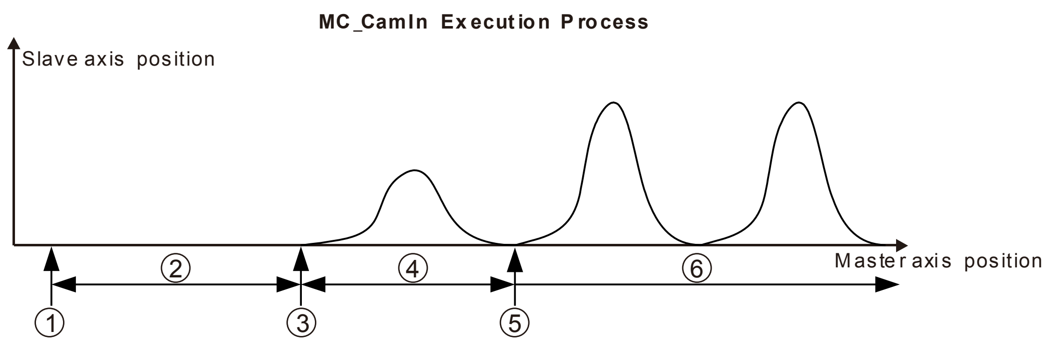
For example, in the planned cam relationship, the master axis position 180 corresponds to the slave axis position 180 and in the actual execution, the corresponding slave axis position is 240 (240=180+60).

Case 2： The master axis position corresponding to the slave axis position will shift (add) by 90 based on the planned position as *MasterScaling* and *SlaveScaling* are 1, *MasterOffset* is 90 and *SlaveOffset* is 0.

For example, in the planned cam relationship, the master axis position 180 corresponds to the slave axis position 180 and in the actual execution, the master axis position 90 corresponds to the slave axis position 180 which is the slave axis position corresponded to by the master axis position 180 (180=90+90) in the planned cam relationship.

- ***StartMode***

In the engagement, the way how the slave axis moves is specified by the *StartMode* parameter. That is, *StartMode* works at stage 4 in the execution of the *MC_CamIn* instruction as shown in the following figure.



Stage 1: Trigger and execute the MC_CamIn instruction.

Stage 2: Wait for the start of the engagement.

Stage 3: The master axis reaches the position where the engagement begins and the slave axis starts to perform the engagement action.

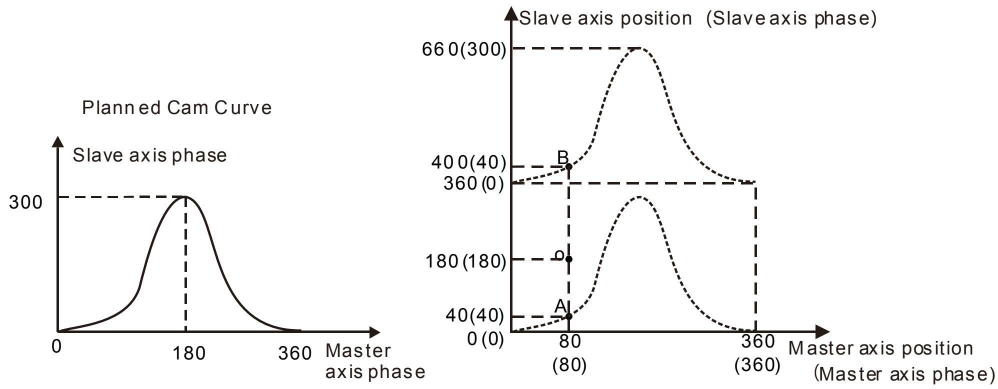
Stage 4: The engagement is ongoing.

Stage 5: The engagement is completed and the master axis and slave axis achieve the synchronization.

Stage 6: The master axis and slave axis are in the synchronous motion.

The cam synchronization requires that the master axis cam phase and the slave axis cam phase meet the defined cam relationship. The engagement process is the process in which the slave axis moves toward the synchronous phase. The synchronous phase and the master axis cam phase meet the defined cam relationship. Since the axis cam phase is cyclic, every cam phase is corresponded to by multiple axis positions. When the engagement occurs, there are many selections for the expected synchronization position. And thus there are several engagement ways for option.

For example, when the engagement starts, the master axis cam phase and slave axis cam phase are 80 and 180 respectively as point O in the following figure. But the defined cam relationship requires that the slave axis cam phase is 40 and thus the synchronous position that the slave axis expects is 40 or 400 (Point A or point B in the following figure) at the moment. The engagement process from Point O to A or Point O to B can be selected via the *StartMode* parameter.

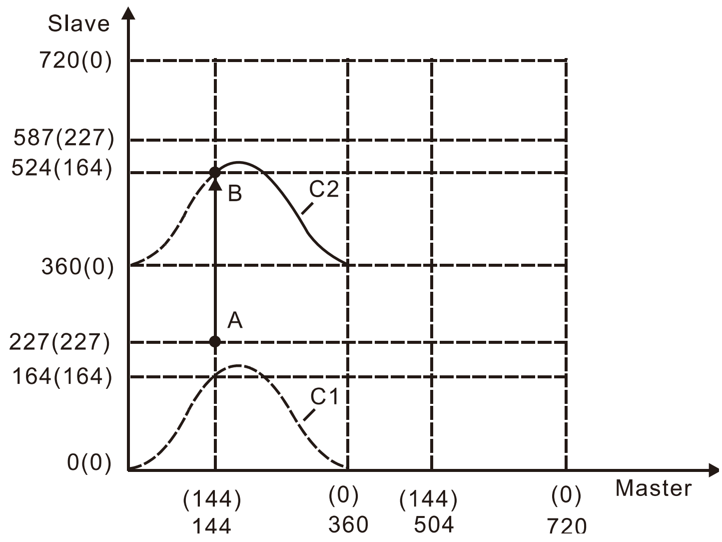


■ *StartMode*

The starting method of cam curves depends on the input *StartMode* in the process of master-slave coupling.

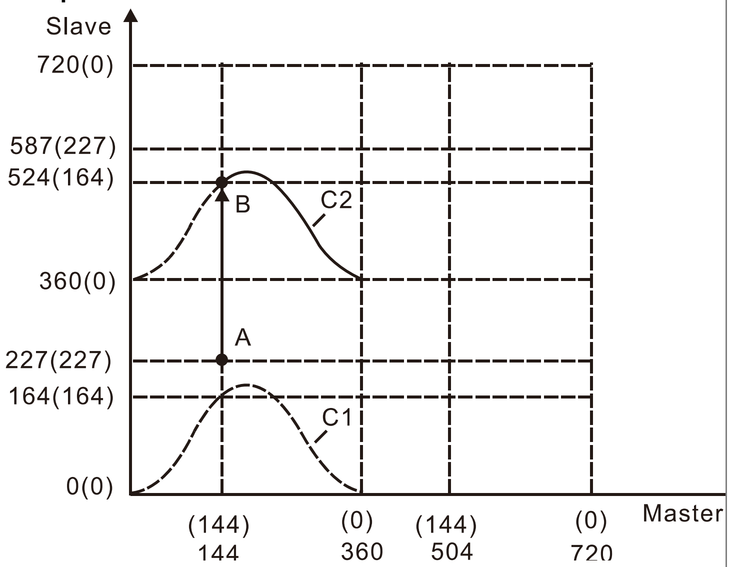
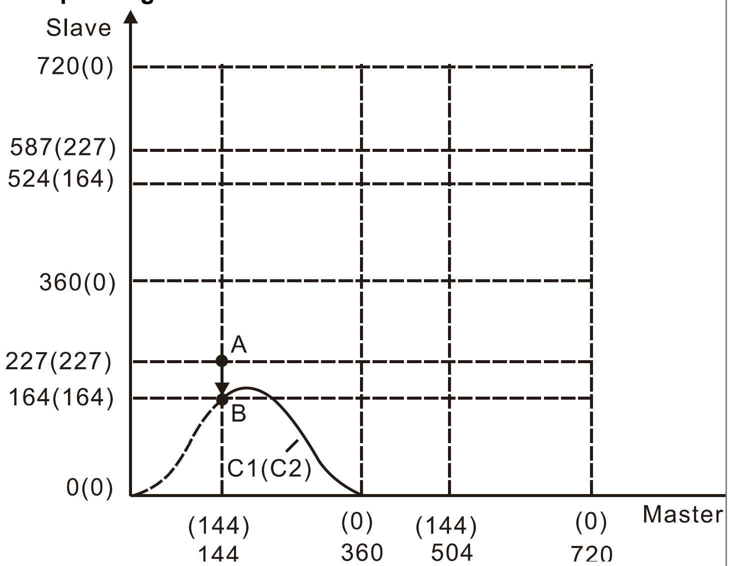
***StartMode*=0 (Jump)**

- The slave axis will jump to the synchronization point in a cycle in the positive direction. The mode may cause the motor to joggle.
- The following figure shows that C1 is a planned cam curve and C2 is an actually performed cam curve. Point A is the (master, slave) position triggering the MC_CamIn instruction and point B is the synchronization point for master and slave axes.

 	The slave axis moves from point A to point B by a jump mode. That is, the slave axis completes the motion from position 227 to 524 in a cycle.

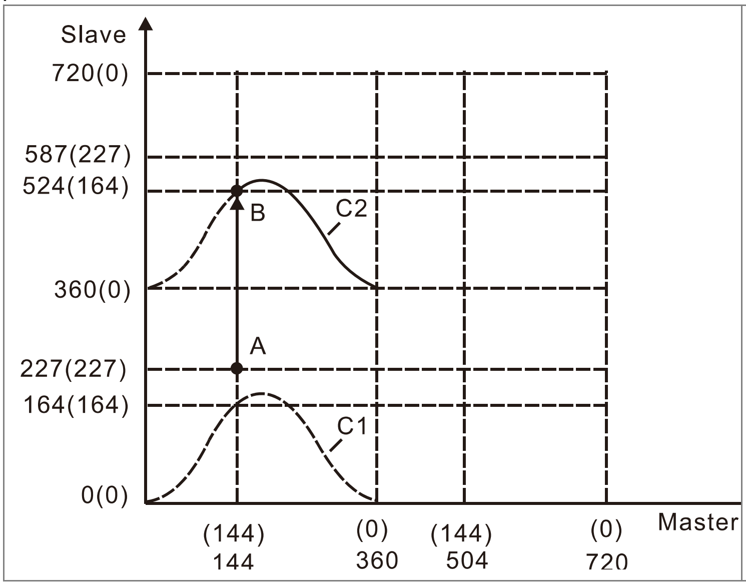
***StartMode*=1 (RampIn_Shortest)**

- The slave axis will move to the synchronization point by taking the shortest way at the set parameters.
- The execution of the engagement action, the slave axis moves toward the position for synchronization by taking the shortest way. (using the shortest distance In Ramp In Positive or Ramp In Negative; if the distances of Ramp In Positive and Ramp In Negative are the same, use the vaule in Ramp In Positive)
- The following figure shows that C1 is a planned cam curve and C2 is an actually performed cam curve. Point A is the (master, slave) position triggering the MC_CamIn instruction and point B is the synchronization point for master and slave axes.
- At the moment, the motion of the slave axis is affected by the *Velocity, Acceleration Deceleration* and *Jerk* parameters.

Ramp In Postive  	The slave axis moves 297 units from the start point A to the synchronization point B (297 = (360-227) + (164-0)) in the positive direction.
Ramp In Negative  	The slave axis negatively moves 63 units from the start point A to the synchronization point B (63=227-164). The shortest distance is Ramp In Negative (63).

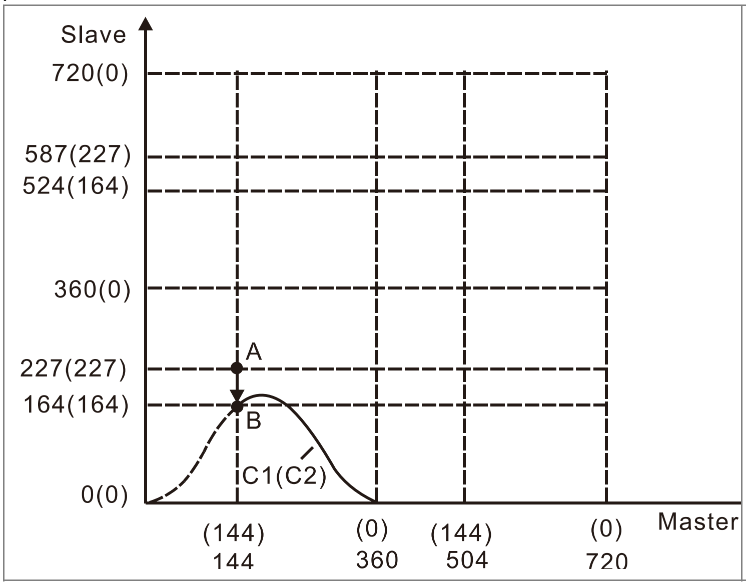
***StartMode=2* (RampIn_Positive)**

- The slave axis will move to the synchronization point in the positive direction at the set parameter values.
- At the moment, the motion of the slave axis is affected by the *Velocity, Acceleration Deceleration* and *Jerk* parameters.

	The slave axis moves from point A to point B in the positive direction.

***StartMode=3* (RampIn_Negative)**

- The slave axis will move to the synchronization point in the negative direction at the set parameter values.
- At the moment, the motion of the slave axis is affected by the *Velocity, Acceleration Deceleration* and *Jerk* parameters.

	The slave axis moves from point A to point B in the negative direction

Note:

1. If the specified input values such as *Velocity, Acceleration, Deceleration* and *Jerk* are too small, the slave axis may not be able to couple with the master axis at the specified sync point, or achieve the synchronization in the specified start mode.

■ **The impact of other instructions on cam operation**

◆ ***MC_CamOut***

The *MC_CamOut* instruction can be used to end the cam operation which is being carried out.

◆ ***MC_SetPosition***

The *MC_SetPosition* instruction has no impact on the being executed motion instructions. Thus, during cam operation, the execution of *MC_SetPosition* instruction for the master axis and slave axis will not affect the cam operation. If the cam operation is triggered after the *MC_SetPosition* instruction is executed, the cam will be affected by the axis position change which is incurred by using the *MC_SetPosition* instruction.

◆ ***MC_Stop* and *MC_Halt***

As the *MC_Stop* and *MC_Halt* instructions are executed on the slave axis, the *MC_CamIn* instruction is aborted, the cam relationship is disconnected and the slave axis decelerates till it stops.

◆ ***MC_Home***

The *MC_Home* instruction cannot be executed on the slave axis but the master axis. As the *MC_Home* instruction is executed on the master axis, the master axis position may have a great change in a very short time, which may cause the vibration of the slave axis. Therefore, the *MC_Home* instruction is recommended to execute after the synchronous relationship between the two axes is disconnected.

■ **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in Standstill state.

The following table lists the available buffer mode settings of MC_CamIn.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

The following table lists the buffer effects of MC_ Camin.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_CamIn	YES	YES	*EndOfProfile*

For more information of buffer mode, refer to ***AH Motion Controller – Operation Manual***.

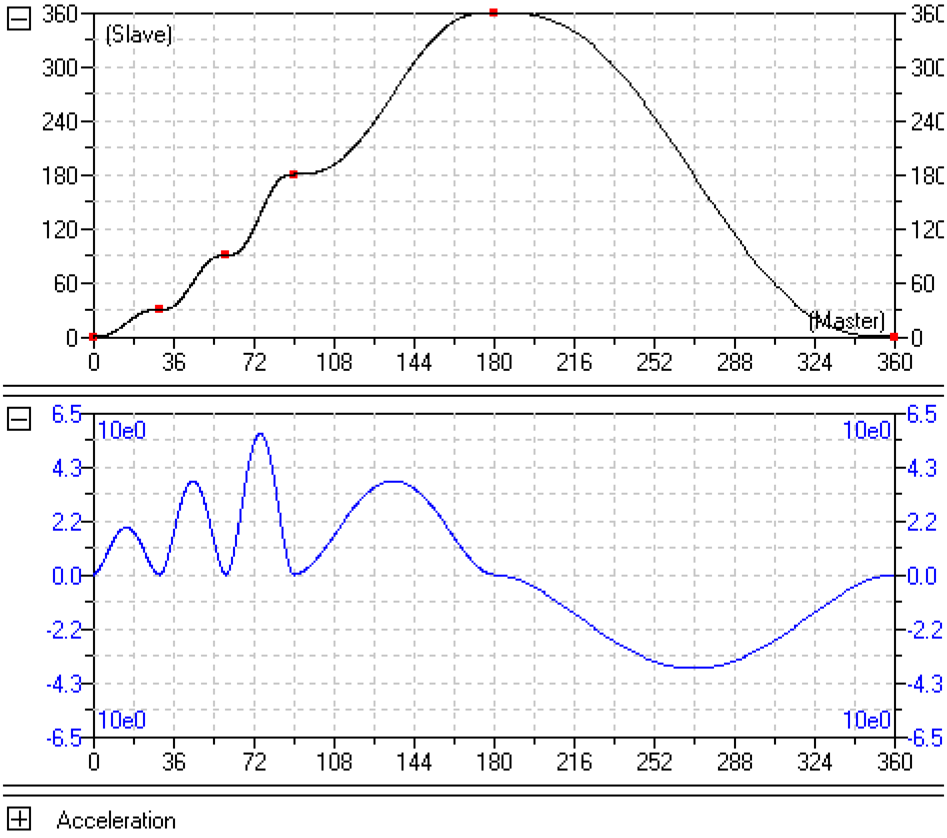
Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

Programming Example

This example explains MC_CamIn instruction execution effects after the parameters for the electronic cam motion are configured.

An electronic cam curve is planed below:

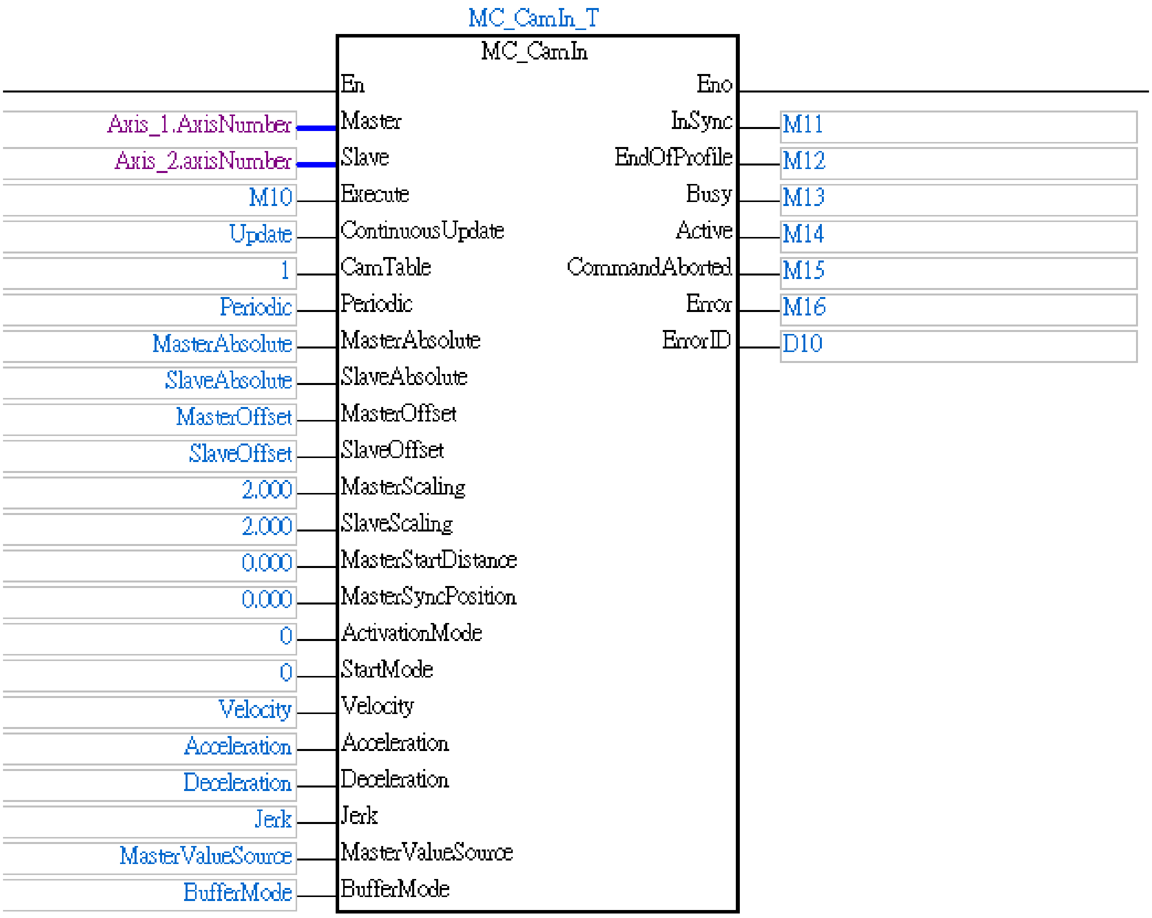


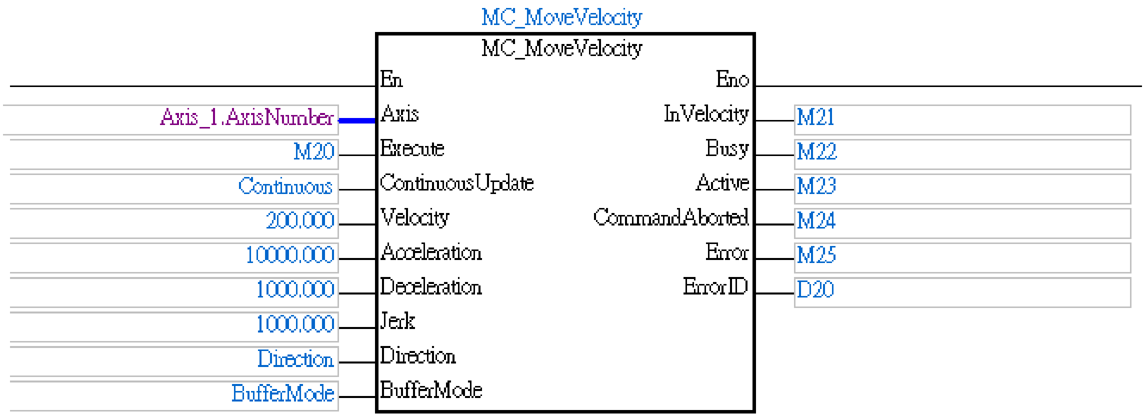
	Master Position	Slave Position	Slave Velocity	Slave Accelerati..
1	0.0000	0.0000	0.0000	0.0000
2	30.0000	30.0000	0.0000	0.0000
3	60.0000	90.0000	0.0000	0.0000
4	90.0000	180.0000	0.0000	0.0000
5	180.0000	360.0000	0.0000	0.0000
6	360.0000	0.0000	0.0000	0.0000

Conditions:

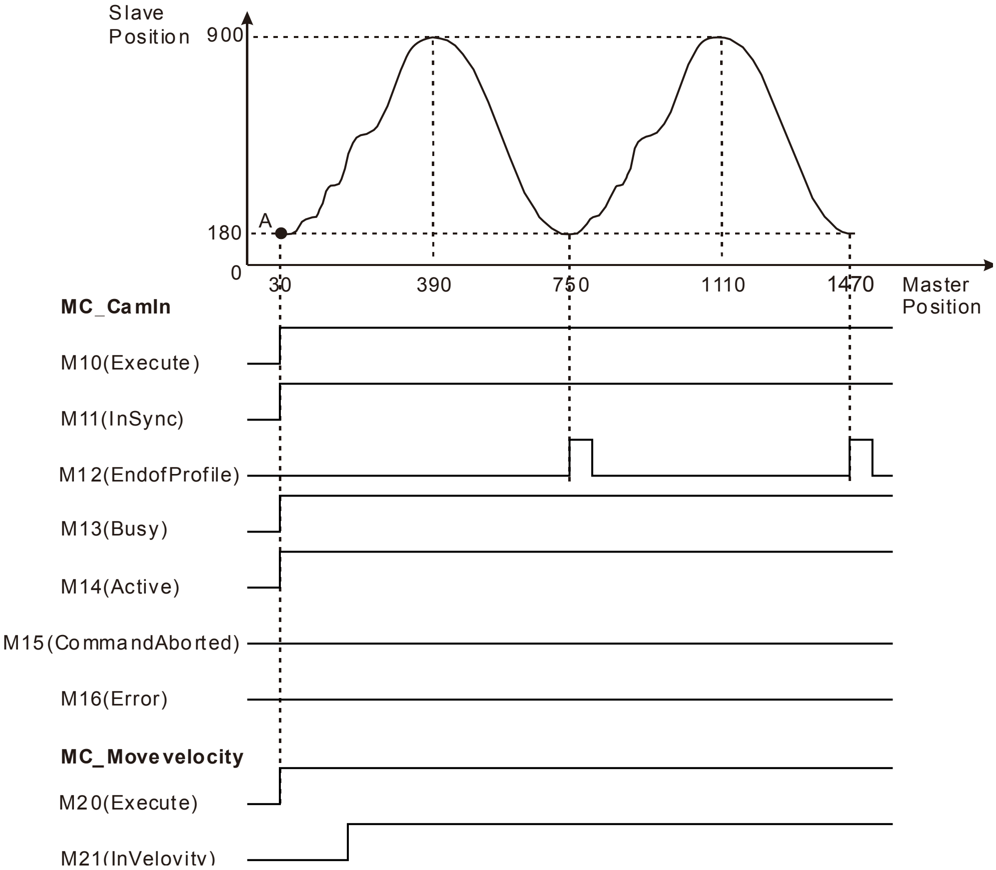
Parameter name	Value and explanation
Cam period of the master axis and slave axis	360
Master Scaling and SlaveScaling	2
MasterOffset	0
SlaveOffset	0
Master axis: absolute/ relative	Relative (False)
Slave axis: absolute/relative	Relative (False)
Periodic	Periodic (True)
StartMode	Jump to the positive target position (0：mcJump)

Function blocks:





Motion diagram:



How the starting point for mapping actual axis position and the cam curve coordinates is calculated:

Slave position = f [(master position + master offset)/ master scaling] * slave scaling + slave offset

Current master position is 30 and slave position is 180;

Master position on the cam profile: master position + master offset)/ master scaling = (0 + 0) / (2) = 0

The cam table shows that slave position is 0 when master position is 0, i.e. f (0)=0.

Slave position on the cam profile: Slave position = [f (0)=0] * slave scaling + slave offset =0 * 2 + 0 = 0.

Therefore, after MC_CamIn is executed, the coordinate of the starting point is (0, 0) in the cam curve which corresponds to the current position (30,180).

Actual master and slave position corresponding to the end point of cam curve:

Actual master position:

To perform displacement on the cam curve, master axis needs to move 360 from point (0, 0) to complete one cam cycle. Since master scaling is 2, master axis needs to move 720 from current position in actual displacement to complete one cycle, i.e. the actual master axis position of the end point: 30+720**=750**.

Actual slave position:

In the cam curve, slave axis moves 360 from point (0, 0) to reach the maximum value. Since slave scaling is 2, slave axis needs to move 720 positively from current position so as to reach the maximum value, i.e. 180+720=900.

In the cam curve, slave axis moves 360 from the maximum value to the end point position: 0. Since slave scaling is 2, slave axis needs to move 720 negatively from the maximum value to reach the end point, i.e. 900-720**=180**.

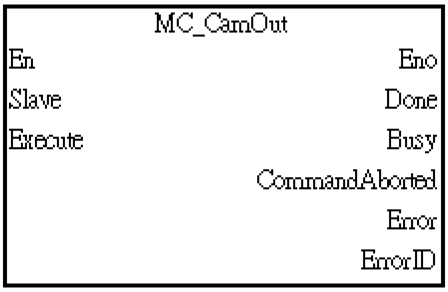
To sum up, the actual end point position is **(750, 180).**

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_CamOut

FB/FC	Description
FB	MC_CamOut disengages the master axis and slave axis from the synchronization operation.



- ## Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-

- ## Outputs

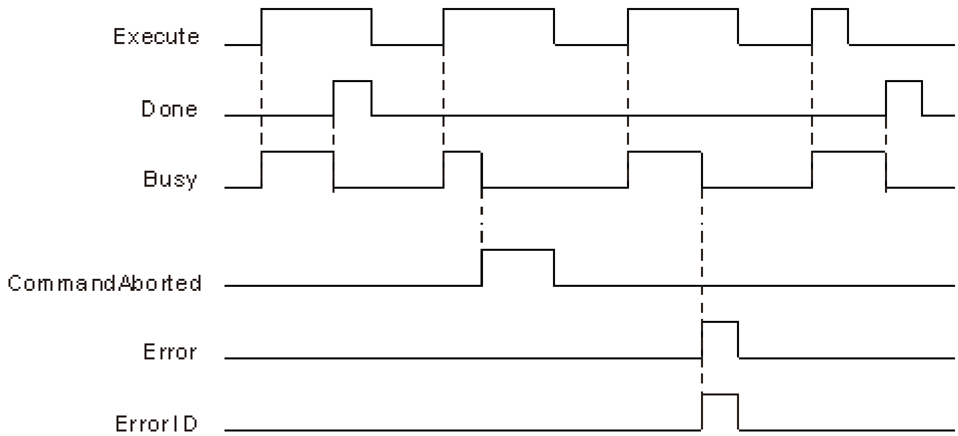
Name	Function	Data type	Output range (Default value)
Done	True when the disengaging is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the disengaging is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True. • When *CommandAborted* shifts to True.

CommandAborted	• When this instruction is aborted by another instruction.	• When *Execute* changes to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Slave	axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

The MC_CamOut instruction ends cam operation of the slave axis. After the cam relationship is disconnected, the slave axis will keep moving at the speed where the cam relationship is disconnected. The axis will be in ContinuousMotion (it has nothing to do with the axis velocity).

- If you execute this instruction on an axis which does not perform the cam operation, an error will occur.

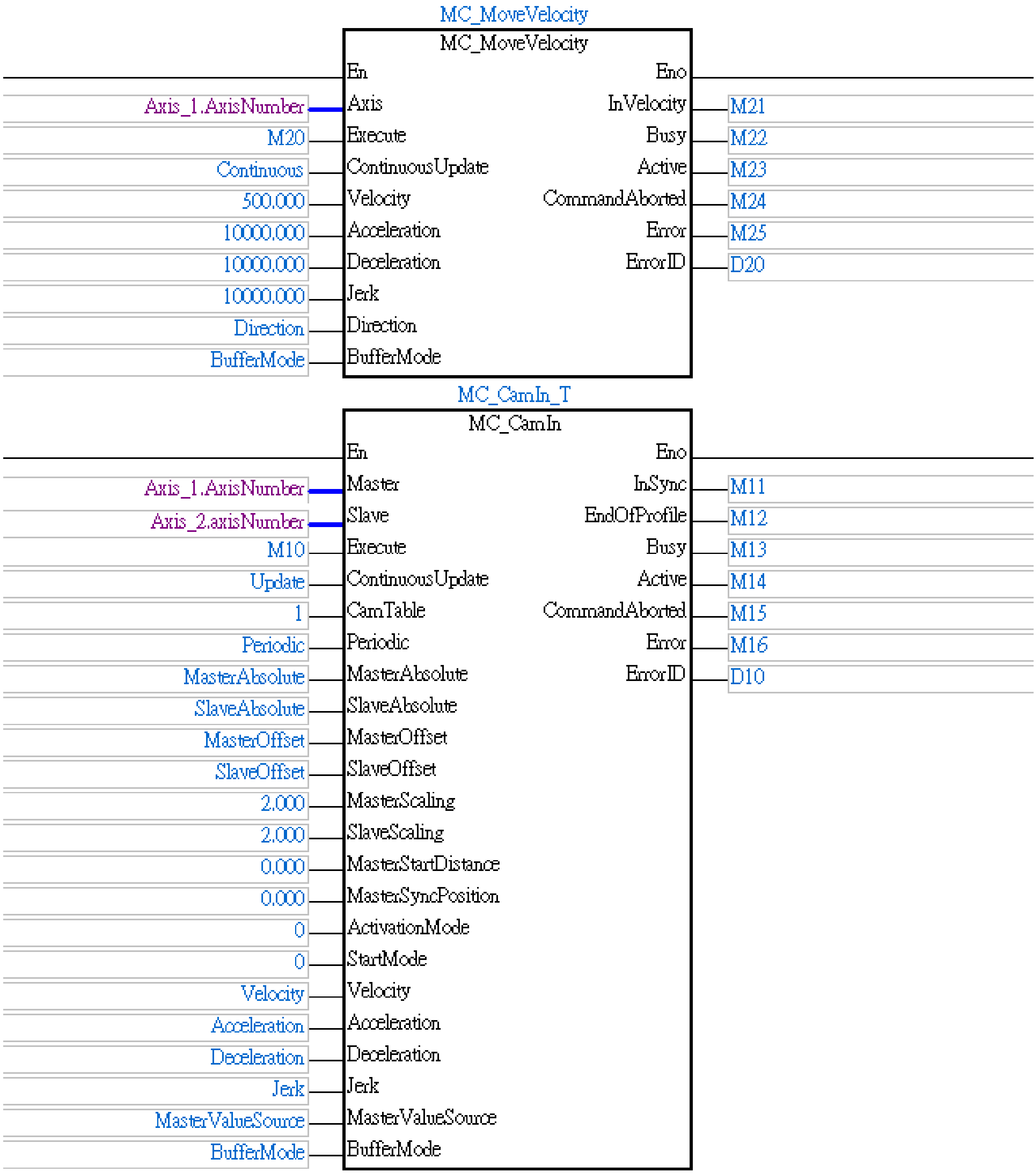
Troubleshooting

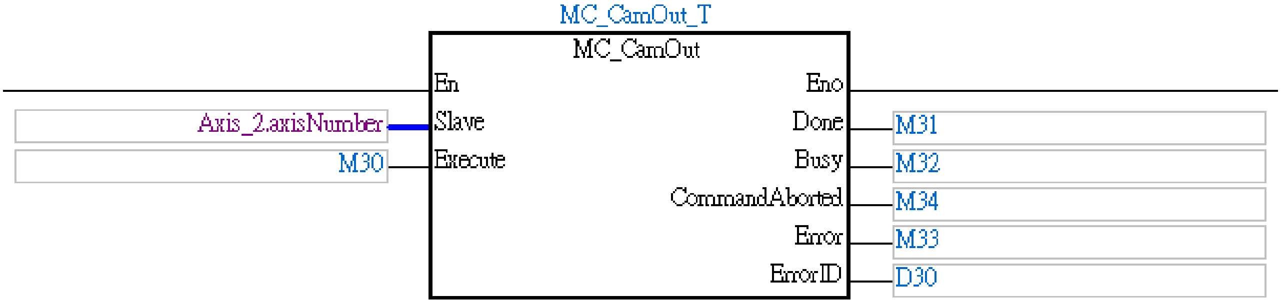
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

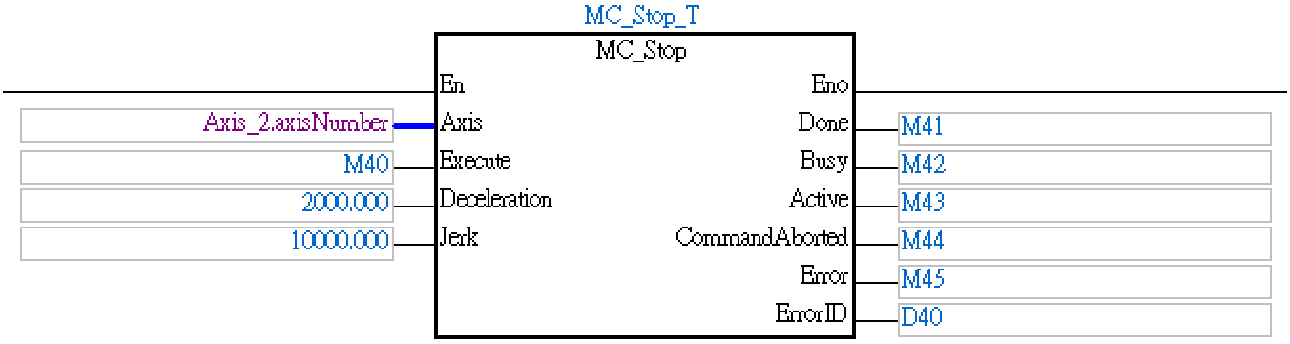
Programming Example

The following example describes the corresponding motion state throughout the cam operation via CAM-related instructions.

Note: the value of *Periodic*, *MasterAbsolute*, *SlaveAbsolute of* MC_CamIn_T is True.

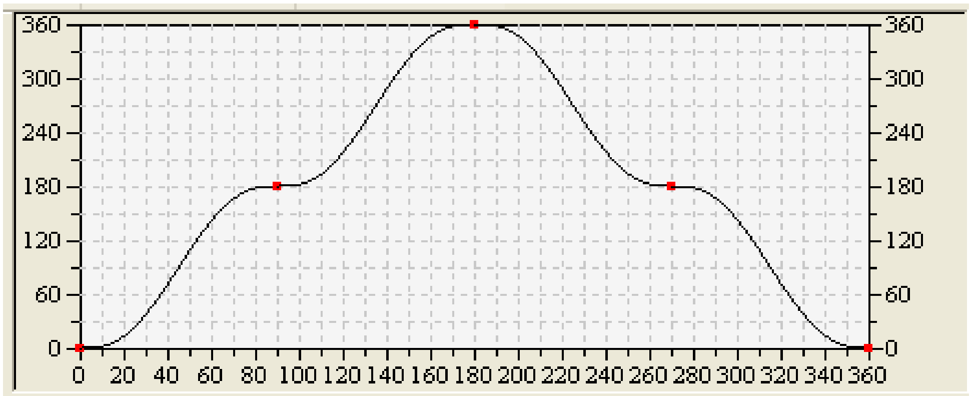






3

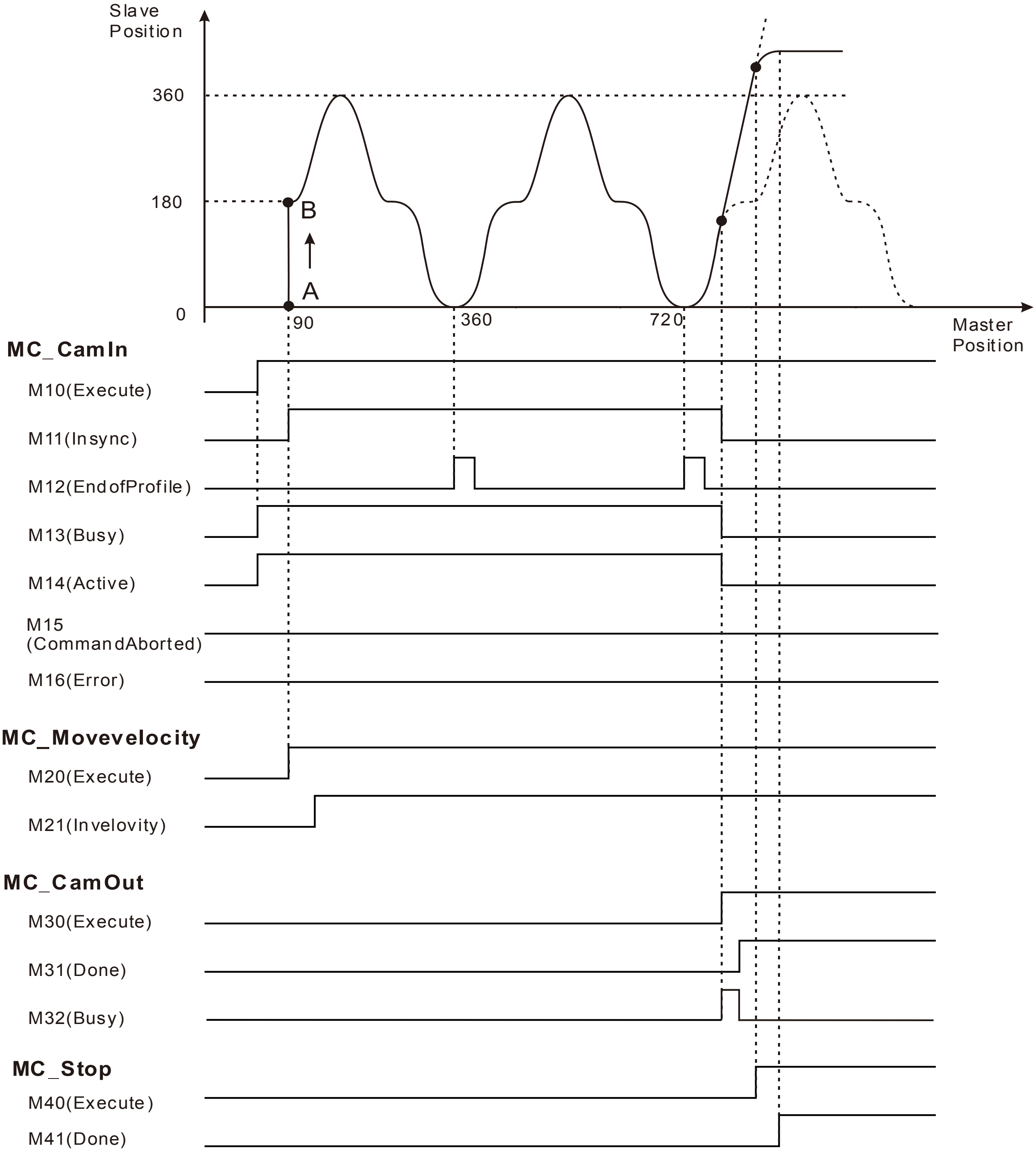
When CamTable ID is 2, the corresponding curve is planned as below:



Element	Master Position	Slave Position	Velocity	Acceleration
Start Point	0.000000	0.000000	0.000000	0.000000
Point	90.000000	180.000000	0.000000	0.000000
Point	180.000000	360.000000	0.000000	0.000000
Point	270.000000	180.000000	0.000000	0.000000
End Point	360.000000	0.000000	0.000000	0.000000

Motion diagram:

The positions of the master axis and slave axis are 90 and 0 respectively as point A shows in the following figure when the MC_CamIn instruction is executed. The motion curve is shown below after the cam operation is performed.



- When M10 changes to True, MC_CamIn is executed. According to the specified cam engagement behavior, the slave axis jumps from point A to point B immediately.
- when M20 changes to True, the master axis executes a velocity instruction and slave axis will start the motion following master axis according to cam curve.
- When M30 changes to True, MC_CamOut is executed and the cam operation is disabled; Slave axis will move at the speed where the cam operation ends.

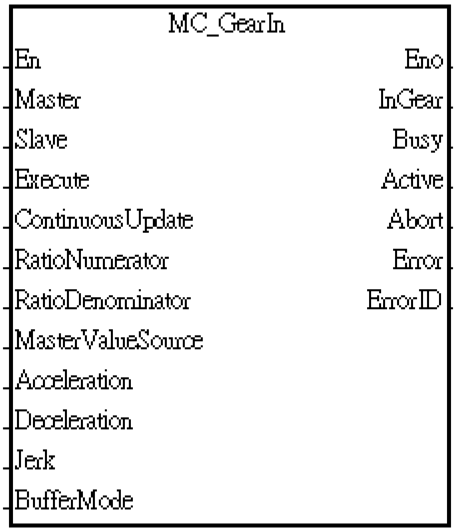
- When M40 changes to True, slave axis starts decelerating to stop.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_GearIn

FB/FC	Description
FB	MC_GearIn establishes the gear relation (velocity) between master and slave axis.



- The gear ratio is defined by the values specified in *RatioNumerator* (numerator) and *RatioDenominator (*denominator*)*.
- A negative gear ratio indicates that the directions for the master and slave axis are opposite.
- You can define the reference source for master axis as commanded value or actual value.
- *Acceleration*, *Deceleration* and *Jerk* allows you to specify a desired behavior for the slave axis to engage with the master axis when the instruction is executed.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Continuous Update	Continuously updates the gear ratio between the master and slave axes when Continuousupdate is True.	BOOL	True/False (False)	When *Active* shifts to True and it will update continuously.
RatioNumerator	The numerator of the electronic gear ratio between the master and slave axes. A negative gear ratio indicates the directions	DINT	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
	for the master and slave axis are opposite.			
RatioDenominator	The denominator of the electronic gear ratio between the master and slave axes. A negative gear ratio indicates the directions for the master and slave axis are opposite.	DWORD	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
MasterValueSource	Specifies the reference position of the master axis.	eMC_SOURCE*1	0: mcCommandedValue 2: mcActualValue (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2) *	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2) *	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3) *	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*1	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

- ## Outputs

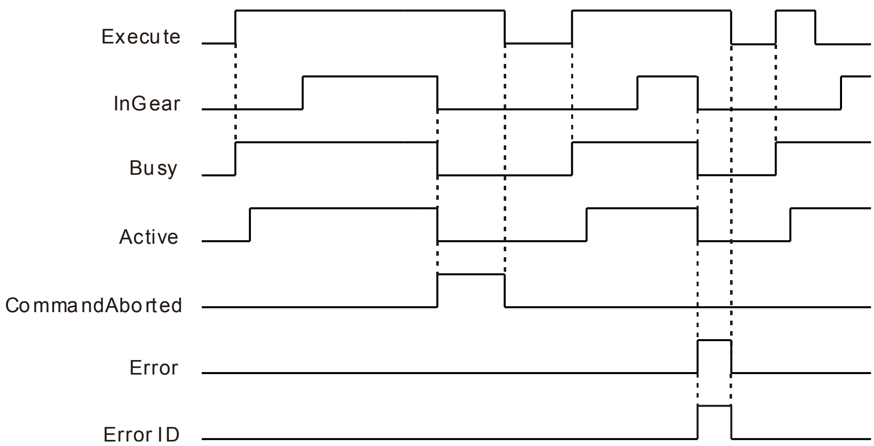
Name	Function	Data type	Output range (Default value)
InGear	True when the gear operation is completed and the in-gear status is acknowledged.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)

Name	Function	Data type	Output range (Default value)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
InGear	• When the slave axis achieves the target velocity and gear operation is completed.	• When *Error* shifts to True. • When *CommandAborted* shifts to True. • When the master axis velocity or the gear ratio is changed.
Busy	• When *Execute* changes to True.	• When *Error* shifts to True. • When *CommandAborted* shifts to True.
Active	• When the motion on the axis is started.	• When *Error* shifts to True. • When *CommandAborted* shifts to True.
CommandAborted	• When this instruction is aborted because of the execution of MC_GearOut instruction. • When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. • When this instruction is aborted because of the execution of MC_Stop instruction.	• When *Execute* shifts to False. • If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram



- **In-Outs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Master	Master axis number 1: 1st axis position 2: 2nd axis position 3: 3rd axis position 4: 4th axis position 5: 5th axis position 6: 6th axis position 7: 7th axis position 8: 8th axis position 9: 9th axis position 10: 10th axis position 11: 11th axis position 12: 12th axis position 13: 13th axis position 14: 14th axis position 15: 15th axis position 16: 16th axis position 17: 17th axis position 18: 18th axis position 19: 19th axis position 20: 20th axis position 21: 21st axis position 22: 22nd axis position 23: 23rd axis position 24: 24th axis position 25: 25th axis position 26: 26th axis position 27: 27th axis position 28: 28th axis position 29: 29th axis position 30: 30th axis position 31: 31st axis position 32: 32nd axis position 200: 1st counter value 204: 2nd counter value 208: 3rd counter value 212: 4th counter value 216: 5th counter value 220: 6th counter value	eMC_Master_SOURCE*1	1: Axis1_Cmd 2: Axis2_Cmd 3: Axis3_Cmd 4: Axis4_Cmd 5: Axis5_Cmd 6: Axis6_Cmd 7: Axis7_Cmd 8: Axis8_Cmd 9: Axis9_Cmd 10: Axis10_Cmd 11: Axis11_Cmd 12: Axis12_Cmd 13: Axis13_Cmd 14: Axis14_Cmd 15: Axis15_Cmd 16: Axis16_Cmd 17: Axis17_Cmd 18: Axis18_Cmd 19: Axis19_Cmd 20: Axis20_Cmd 21: Axis21_Cmd 22: Axis22_Cmd 23: Axis23_Cmd 24: Axis24_Cmd 25: Axis25_Cmd 26: Axis26_Cmd 27: Axis27_Cmd 28: Axis28_Cmd 29: Axis29_Cmd 30: Axis30_Cmd 31: Axis31_Cmd 32: Axis32_Cmd 200: DFB_AC0 204: DFB_AC4 208: DFB_AC8 212: DFB_AC12 216: DFB_AC16 220: DFB_AC20	When *Execute* is rising edge triggered and *Busy* is False.
Slave	Slave axis number	WORD	K1~Kn*2 (0)	When *Execute* is rising edge triggered and *Busy* is False.

Note:

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
2. Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

3

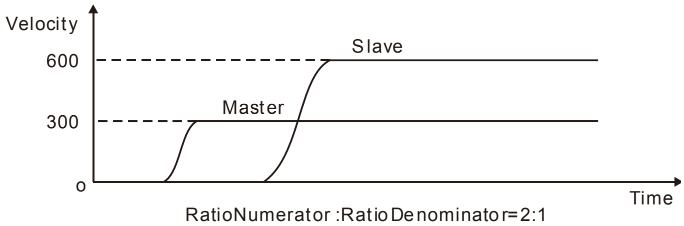
Function

After the gear relation is established, slave axis will follow master axis to move at the given proportional relationship to accomplish the synchronized control of master and slave axis. Master and slave axis could be real or virtual axis or the external encoder master axis.

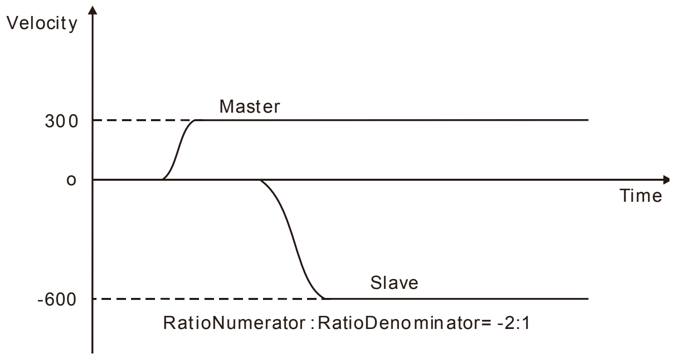
***Note:** If the specified slave axis is executing other motion instruction while MC_GearIn is executed, the executing instruction will be aborted, and the specified axis will start to follow the master axis according to the gear relationship specified by MC_GearIn.

RatioNumerator and ***RatioDenominator***

When gear ratio is positive, the master and slave axes move in the same direction.



When gear ratio is negative, the master and slave axes move in opposite direction.



Acceleration and ***Deceleration***

- If the current speed of the slave axis is lower than the target speed when the MC_GearIn is executed, the slave axis will accelerate to the target speed by the specified acceleration rate (*Acceleration*).
- If the current speed of the slave axis is higher than the target speed when the MC_GearIn is executed, the slave axis will decelerate to the target speed by the specified deceleration rate (*Deceleration*).

BufferMode

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction.

When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in “Standstill” state.

The following table lists the available buffer mode settings of MC_GearIn.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.

The following table lists the buffer effects of MC_GearIn.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_GearIn	YES	YES	*InGear*

For more information of buffer mode, refer to section ***AH Motion Controller – Operation Manual***.

● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Programming Example

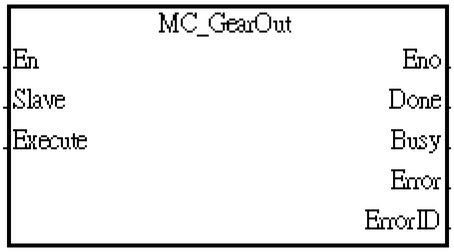
Refer to the programming example of MC_GearOut instruction.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_GearOut

FB/FC	Description
FB	MC_GearOut disconnects the gear relation (velocity) between master and slave axis.



After the execution of MC_GearOut is finished, the disengaged slave axis will keep the velocity where the gear relation is disconnected and become available for executing other motion instructions.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-

Outputs

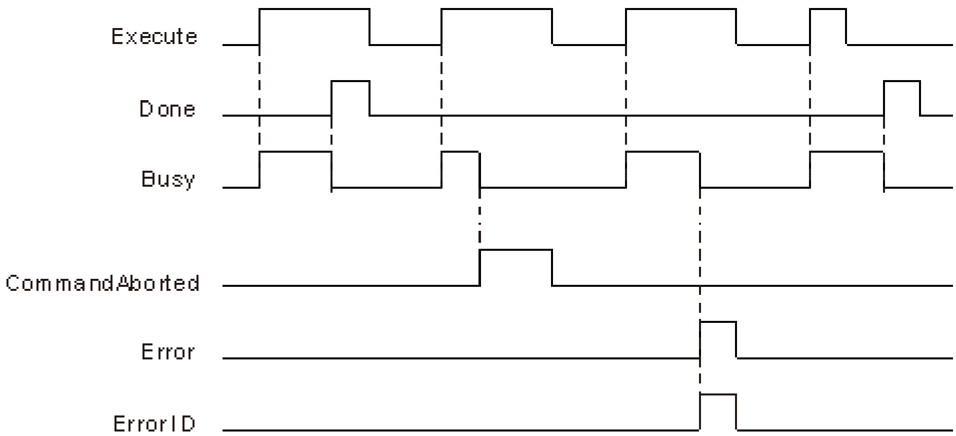
Name	Function	Data type	Output range (Default value)
Done	True when the gear disconnection is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the gear disconnection is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True.

Name	Timing for shifting to True	Timing for shifting to False
		● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
CommandAborted	● When this instruction is aborted by other buffer modes set by mcAborting. ● When this instruction is aborted by MC_Stop.	● When *Execute* changes to False. If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. ● (Error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Slave	Slave axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

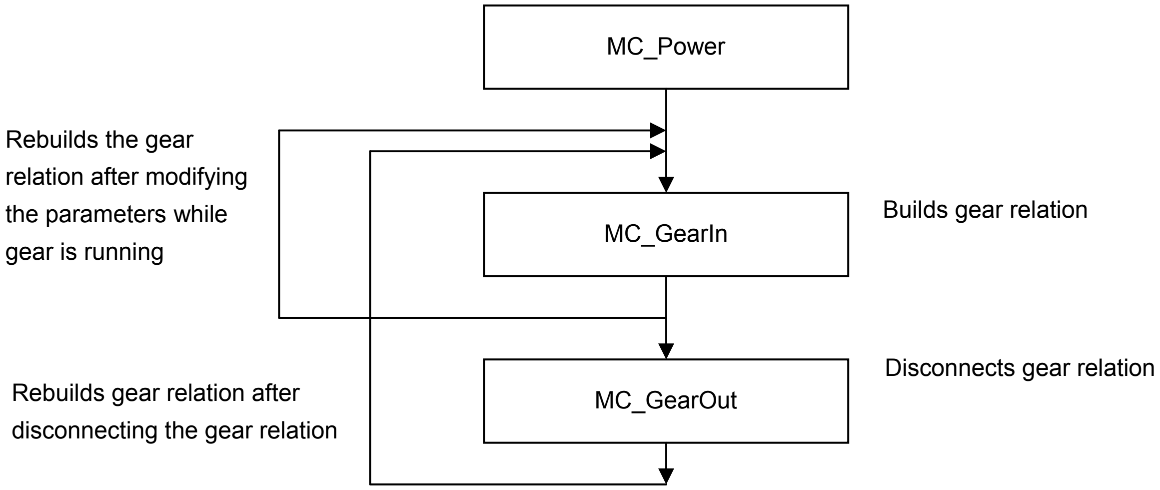
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

3

Function

The MC_GearOut instruction ends gear synchronization of the slave axis. After the gear relationship is disconnected, the slave axis will keep moving at the speed where the gear relationship is disconnected. The axis will be in ContinuousMotion (it has nothing to do with the axis velocity)

- The sequence for execution of the instructions related to electronic gear



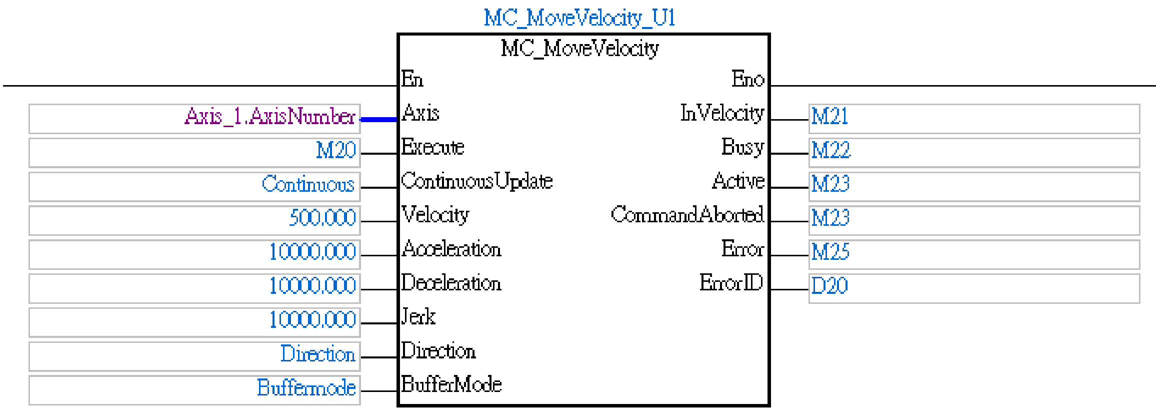
3

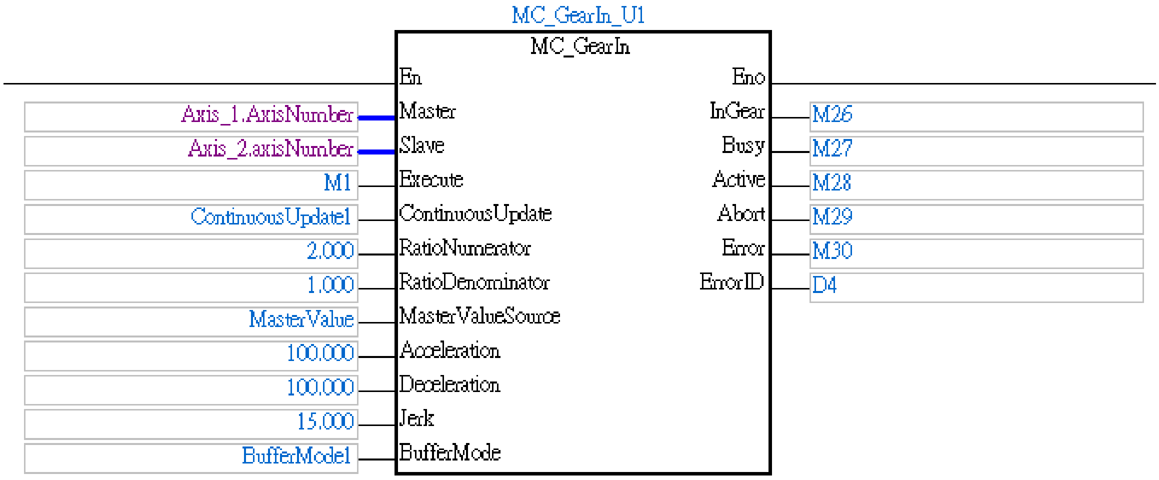
Troubleshooting

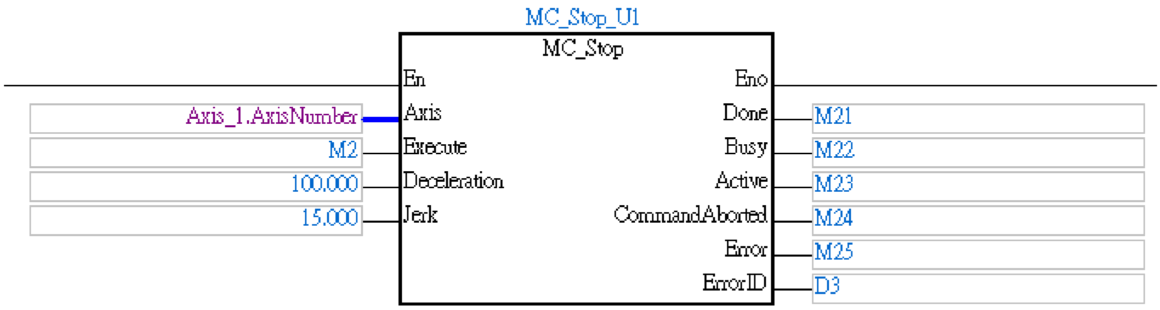
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

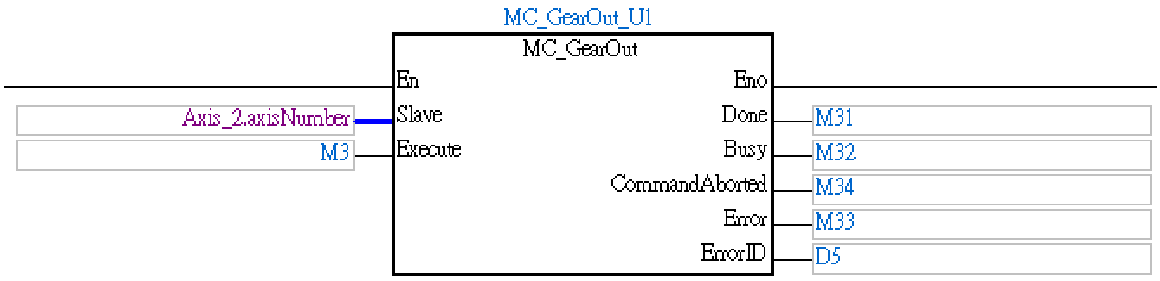
Programming Example

The following example describes the corresponding motion state throughout the gear operation via gear-related instructions.

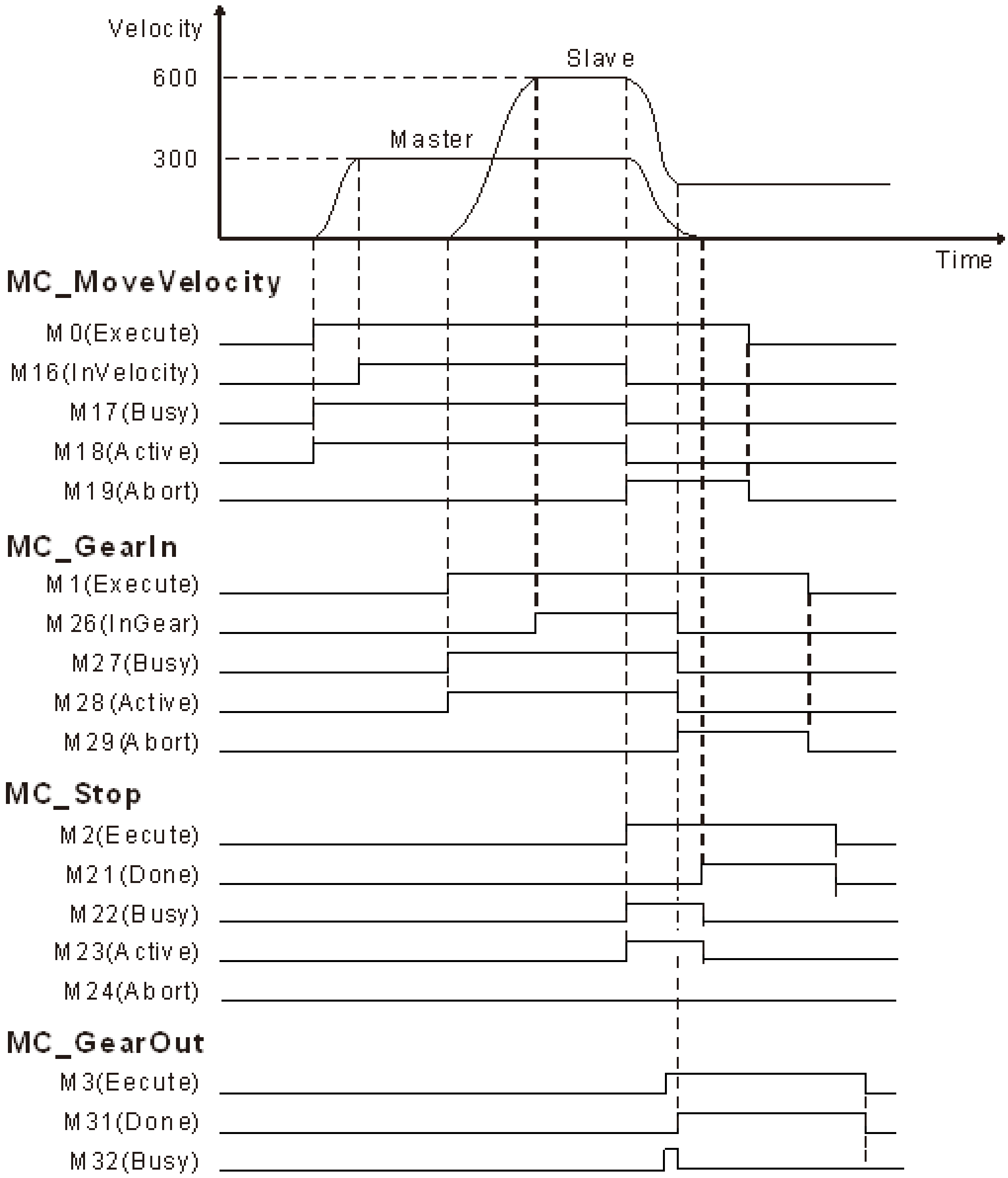



MC_GearIn_U1
MC_GearIn
En
Eno
Axis_1.AxisNumber
Master
InGear
M26
Axis_2.axisNumber
Slave
Busy
M27
M1
Execute
Active
M28
ContinuousUpdate1
ContinuousUpdate
Abort
M29
2.000
RatioNumerator
Error
M30
1.000
RatioDenominator
ErrorID
D4
MasterValue
MasterValueSource
100.000
Acceleration
100.000
Deceleration
15.000
Jerk
BufferMode1
BufferMode


MC_Stop_U1
MC_Stop
En
Eno
Axis_1.AxisNumber
Axis
Done
M21
M2
Execute
Busy
M22
100.000
Deceleration
Active
M23
15.000
Jerk
CommandAborted
M24
Error
M25
ErrorID
D3


MC_GearOut_U1
MC_GearOut
En
Eno
Axis_2.axisNumber
Slave
Done
M31
M3
Execute
Busy
M32
CommandAborted
M34
Error
M33
ErrorID
D5

Motion diagram:



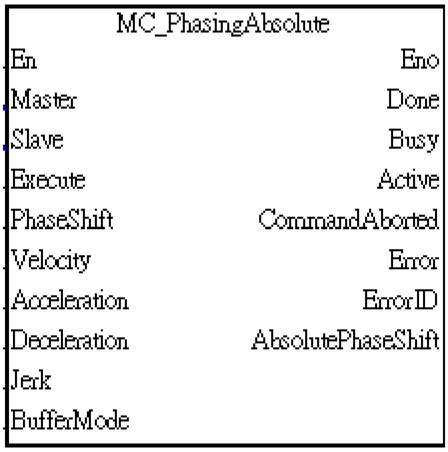
- When M0 (*Execute*) changes to True, M17 and M18 change to True and master axis starts to move.
- When M1 (*Execute*) changes to True, the slave axis starts to catch the master axis. When the velocity of slave axis reaches 2 times the velocity of master axis (numerator=2, denominator=1), M26 (*InGear*) changes to True.
- When M2 (*Execute*) changes to True, master axis executes the MC_Stop instruction.
- In the process of the MC_Stop execution, when M3 (*Execute*) changes to True, MC_GearOut is executed; when the disconnection is completed, M31 (*Done*) changes to True and slave axis will keep moving at the speed when the gear relation is decoupled.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_PhasingAbsolute

FB/FC	Description
FB	MC_PhasingAbsolute shifts the phase of the master axis virtually by a specified absolute phase shift value.



Note:

1. The gear relation between the specified master and slave axes, e.g. electronic gear relationship, electronic cam relationship, should be established before the execution of MC_PhasingAboslute/MC_PhasingRelative.
2. The virtually shifted master axis will impact the motion of the slave axis according to the specified parameters.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
PhaseShift	Specifies the absolute phase shift amount for master axis. (Unit: user unit)*	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	The target velocity of the master axis phase shift amount. (Unit: user unit/s)*	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)*	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)*	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)*	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER _MODE*2	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

● Outputs

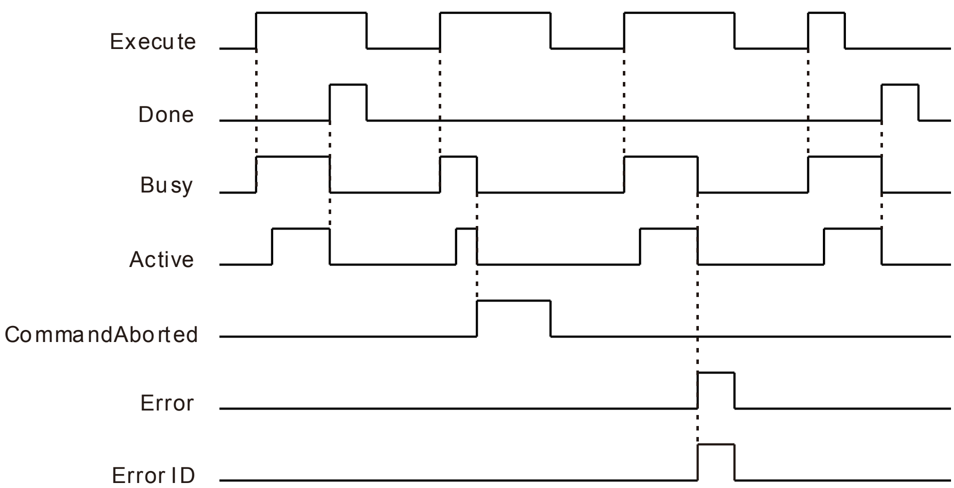
Name	Function	Data type	Output range (Default value)
Done	True when phasing operation is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
AbsolutePhaseShift	Records the absolute phase shift on the master axis continuously.	LREAL	-

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the phasing operation is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● When the motion on the axis is started.	● When *Execute* shifts to False. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. ● When this instruction is aborted because of the execution of MC_Stop instruction.	● When *Execute* changes to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.

Name	Timing for shifting to True	Timing for shifting to False
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. ● (Error code is cleared)
AbsolutePhaseShift	Continuously updates value when *Busy* is True.	Continuously updates value when *Busy* is True.

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Master	Master Axis	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.
Slave	Slave Axis	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Function

- MC_PhasingAbsolute shifts the phase of the master axis virtually by a specified absolute phase shift value.
- The instruction only affects the velocity and position of the slave axis without any influence on the velocity and position of the master axis.
- When MC_PhasingAbsolute instruction is executed but not finished yet, it will not take effects to execute the instruction again.
- The reference zero point for calculating the *PhaseShift* value is the position when the synchronization between master and slave axis starts.
- For detailed explanation of the absolute phasing operation, refer to Programming Example below.

■ **BufferMode**

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction. When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in Standstill state.

The following table lists the available buffer mode settings of MC_PhasingAbsolute.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.
2: mcBlendingLow	Takes the lower target velocity as the transit velocity between the current instruction and the buffered instruction. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: mcBlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.
4: mcBlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.
5: mcBlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

The following table lists the buffer effects of MC_ PhasingAbsolute.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_PhasingAbsolute	YES	YES	*Done*

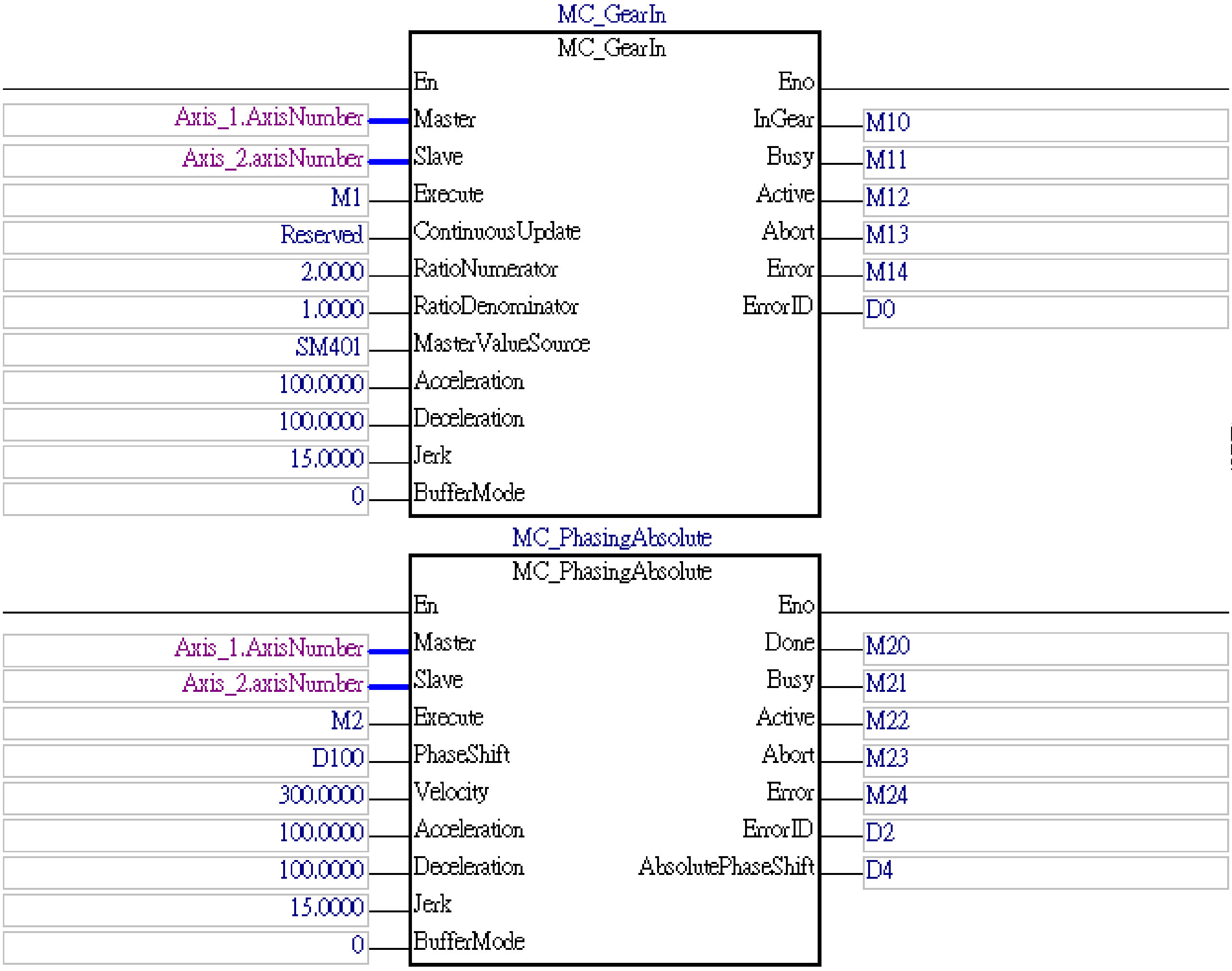
For more information of buffer mode, refer to section ***AH Motion Controller – Operation Manual***.

Troubleshooting

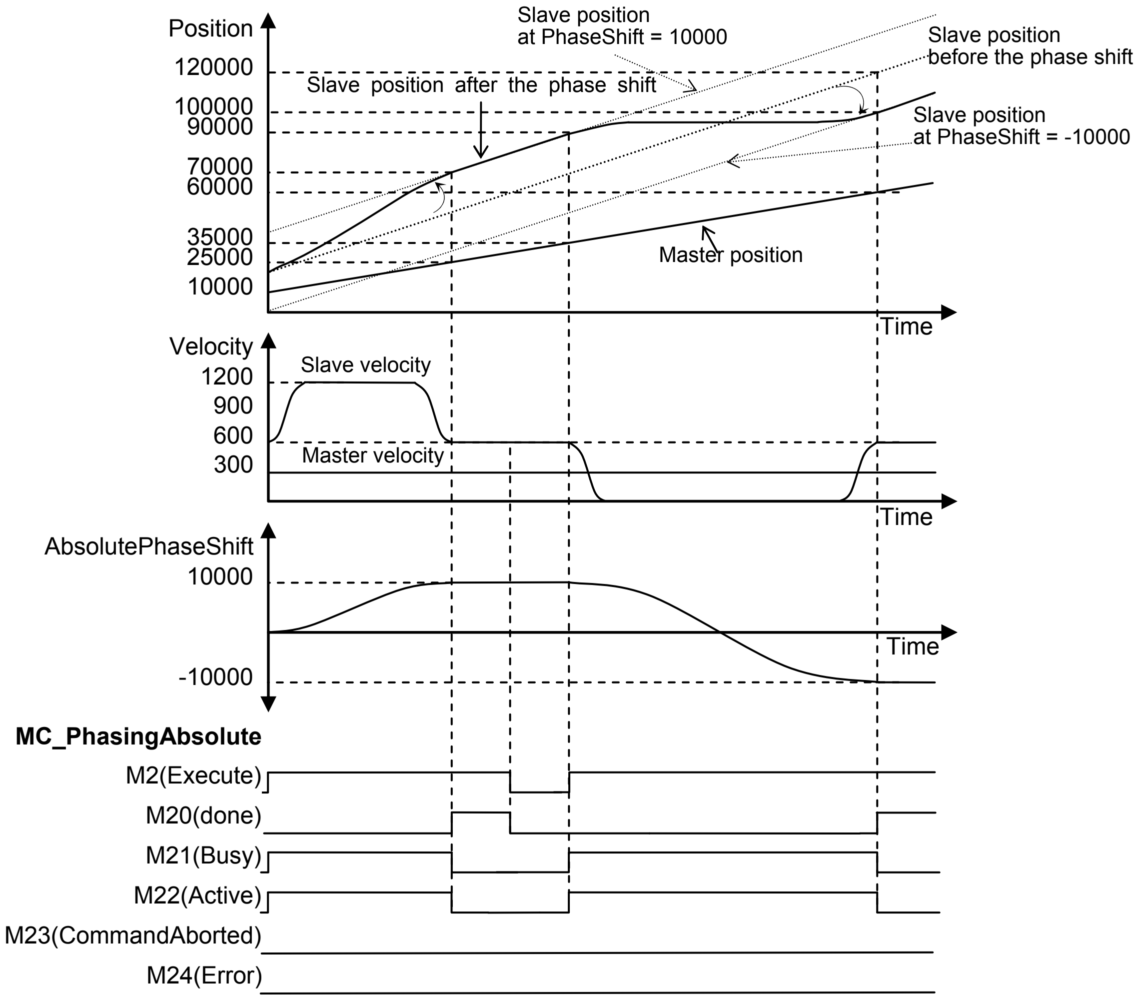
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

- **Programming Example**

When the gear relationship is built between the specified master and slave axes, the MC_PhasingAbsolute instruction will affect the speed and position of the slave axis.



Motion diagram:



- The gear relationship between master axis and slave axis is established after M10 (*InGear*) of MC_GearIn is True. The velocity ratio and position ratio between master axis and slave axis are both 1:2.
- Suppose that master axis moves at a constant speed of 300 and D100 (*PhaseShift*) value is 10,000. When M2(*Execute*) of MC_PhasingAbsolute changes to True, the velocity, acceleration and deceleration and phase shift set in the instruction will be superimposed to the master axis.
- The execution of MC_PhasingAbsolute does not affect the operation of master axis but the operation of slave axis according to gear relationship. As the diagram shows above, when M20 (*Done*) shifts to True at master position = 25,000, the slave position can be obtained as below:

  Actual slave position after the phase shift = (Current master position - previous master position + *PhaseShift* value – **previous shifted amount**) * (RatioNumerator / RatioDenominator) + previous slave position = (25,000-10,000+10,000-**0**)*2+20,000 = 70,000.
  Note: previous shifted amount is deducted because *PhaseShift* is evaluated as an absolute value.
- When D100 (*PhaseShift*) is changed to -10,000 and M2 (*Execute*) changes to True again, the instruction will be executed again. As the diagram shows above, when M20 (*Done*) changes to True at master position 60,000, the slave position can be obtained as below:

  Actual slave position after the phase shift = (Current master position - previous master position + *PhaseShift* value – **previous shifted amount**) * (RatioNumerator / RatioDenominator) + original slave position =

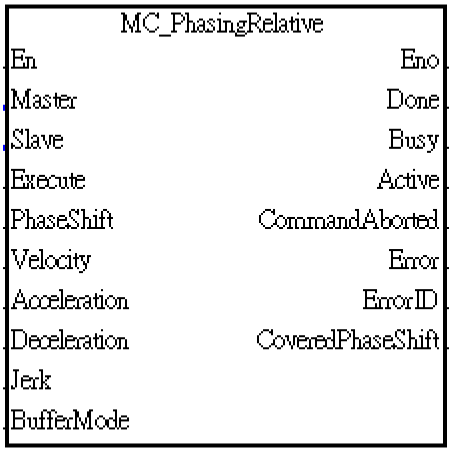
(60,000-35,000-10,000-**10,000**)*2+90,000 = 100,000.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

MC_PhasingRelative

FB/FC	Description
FB	MC_PhasingRelative shifts the phase of the master axis virtually by a specified relative phase shift value.



1. The gear relation between the specified master and slave axes, e.g. electronic gear relationship, electronic cam relationship, should be established before the execution of MC_PhasingAboslute/MC_PhasingRelative.
2. The virtually shifted master axis will impact the motion of the slave axis according to the specified parameters.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
PhaseShift	Specifies the relative phase shift amount for master axis. (Unit: user unit)*	LREAL	Negative number, positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Velocity	The target velocity of the master axis phase shift amount. (Unit: user unit/s)*	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
Acceleration	Acceleration rate. (Unit: user unit/s^2)*	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Deceleration	Deceleration rate. (Unit: user unit/s^2)*	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
Jerk	Jerk value. (Unit: user unit/s^3)*	LREAL	Positive number or 0 (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_ MODE*2	0: mcAborting 1: mcBuffered (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

● Outputs

Name	Function	Data type	Output range (Default value)
Done	True when phasing operation is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
CommandAborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
CoveredPhaseShift	Records the absolute phase shift on the master axis continuously	LREAL	-

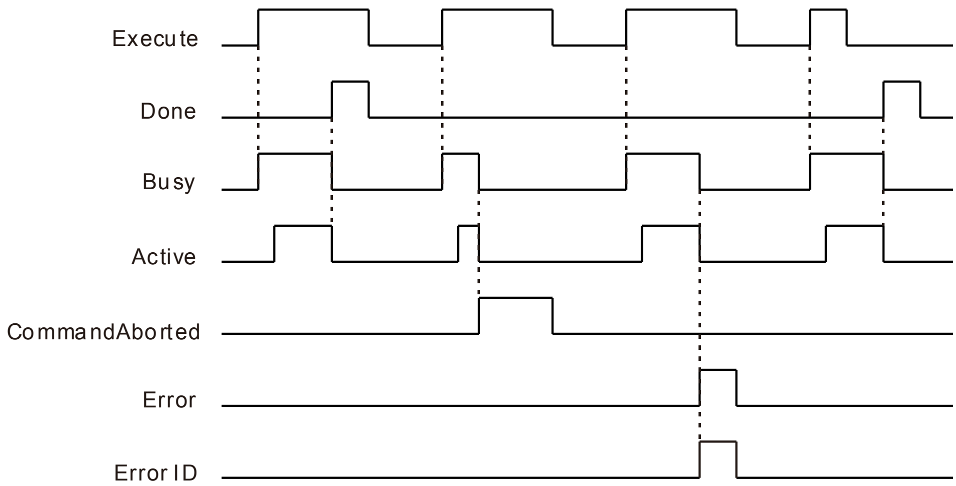
■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the phasing operation is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True.
Active	● When the motion on the axis is started.	● When *Execute* shifts to False. ● When *Error* shifts to True. ● When *CommandAborted* shifts to True. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.
CommandAborted	● When this instruction is aborted by another instruction with the Buffer Mode set to Aborting. ● When this instruction is aborted because of the execution of MC_Stop instruction.	● When *Execute* changes to False. ● If *Execute* is False and *CommandAborted* shifts to True, it will be True for only one period and immediately shift to False.

3

Name	Timing for shifting to True	Timing for shifting to False
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)
CoveredPhaseShift	Continuously updates value when *Busy* is True.	Continuously updates value when *Busy* is True.

■ **Timing Diagram:**



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Master	Master axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.
Slave	Slave axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Function

- MC_PhasingRelative shifts the phase of the master axis virtually by a specified relative phase shift value.
- The instruction only affects the velocity and position of the slave axis without any influence on the velocity and position of the master axis.
- When MC_PhasingRelative instruction is executed but not finished yet, it will not take effects to execute the instruction again.
- The reference zero point for calculating the *PhaseShift* value is the position when the synchronization between master and slave axis starts.
- For detailed explanation of the relative phasing operation, refer to Programming Example below.

BufferMode

BufferMode determines the behavior to combine the axis motions for this instruction and the previous instruction.

When the instruction is executed;

- The selected buffer mode is valid if the previous instruction is executing.
- The selected buffer mode is invalid if the axis is in Standstill state.

The following table lists the available buffer mode settings of MC_PhasingRelative.

Buffer Mode	Function
0: mcAborting	Aborts the ongoing motion. The next instruction takes effect immediately
1: mcBuffered	Automatically executes the next instruction after the ongoing motion is completed.
2: mcBlendingLow	Takes the lower target velocity as the transit velocity between the current instruction and the buffered instruction. (The transit velocity is the velocity that the current instruction uses as the transit point.)
3: mcBlendingPrevious	Takes the target velocity of the current instruction as the transit velocity.
4: mcBlendingNext	Takes the target velocity of the buffered instruction as the transit velocity.
5: mcBlendingHigh	Takes the higher target velocity as the transit velocity between the current instruction and the buffered instruction.

The following table lists the buffer effects of MC_ PhasingRelative.

Instruction	Can be specified as a buffered instruction	Can be followed by a buffered instruction	Relevant signal to activate the next buffered instruction
MC_PhasingRelative	YES	YES	Done

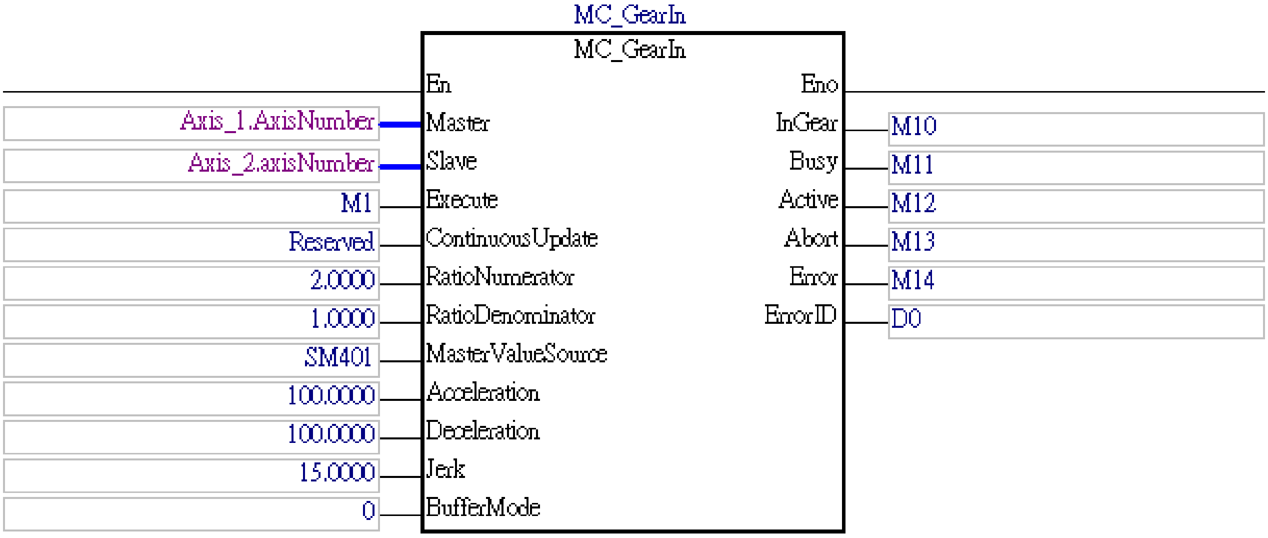
For more information of buffer mode, refer to ***AH Motion Controller – Operation Manual***.

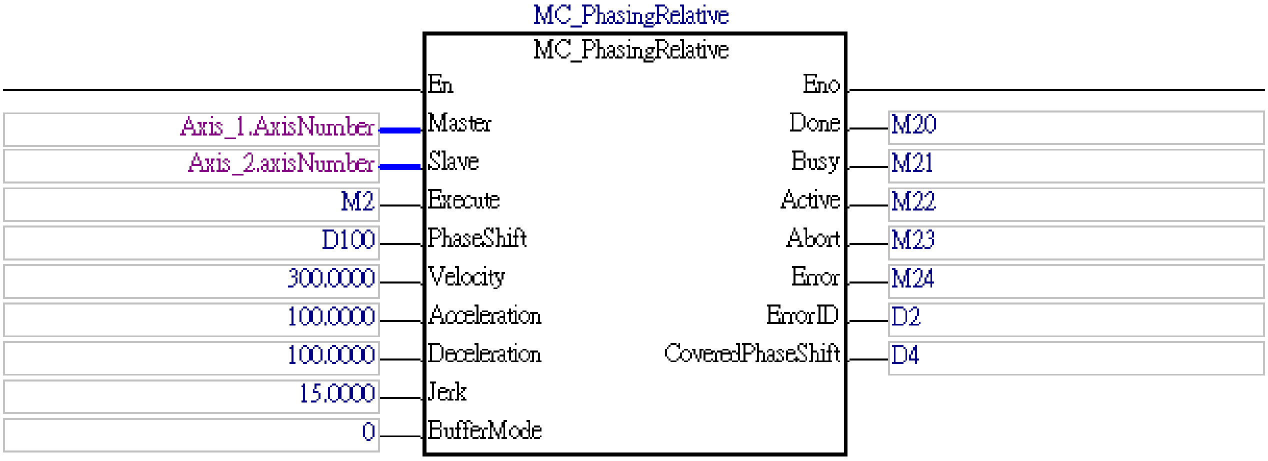
Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

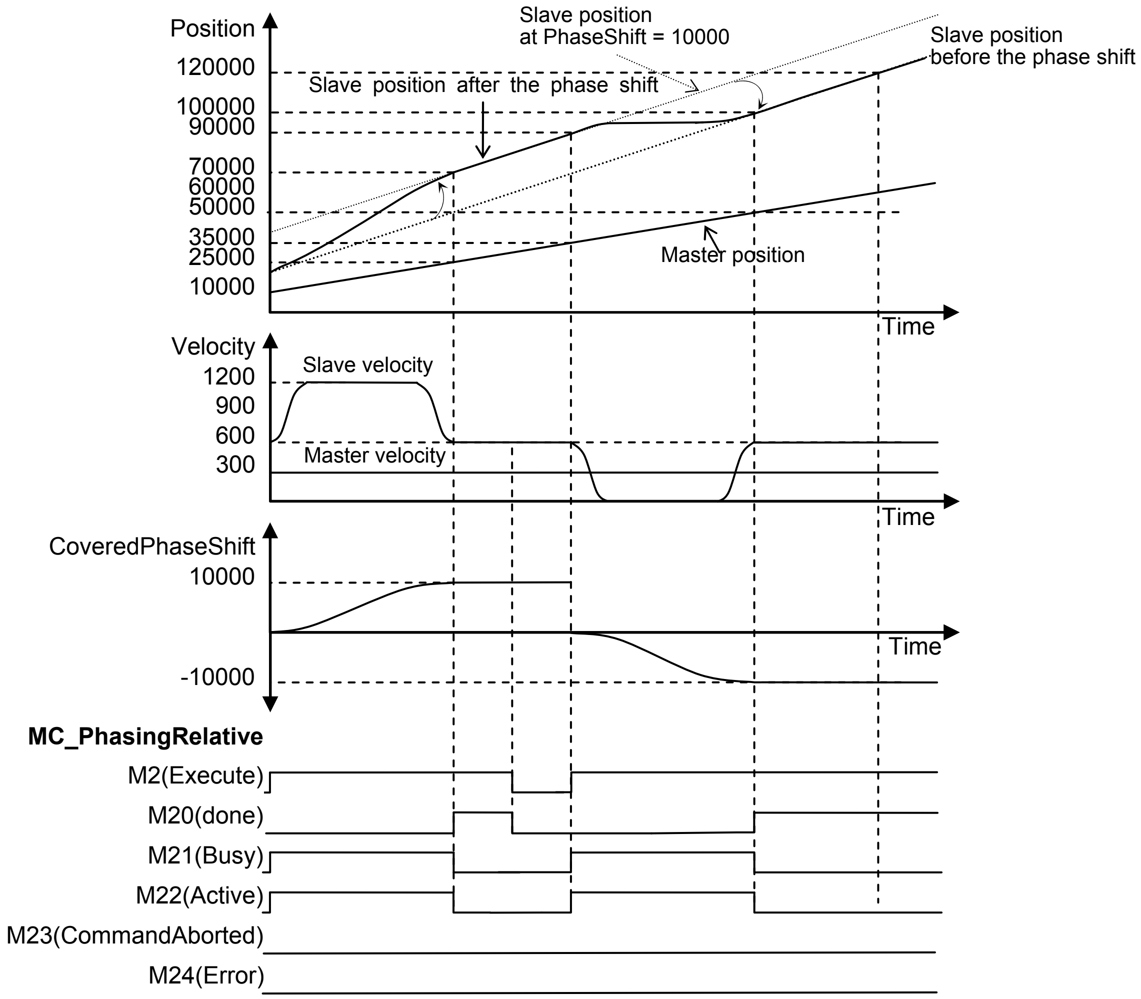
● Programming Example

When the gear relationship is built between the specified master and slave axes, the MC_PhasingRelative instruction will affect the speed and position of the slave axis.





Motion diagram:



- The gear relationship between master axis and slave axis is established after M10 (*InGear*) of MC_GearIn is True. The velocity ratio and position ratio between master axis and slave axis are both 1:2.
- Suppose that master axis moves at a constant speed of 300 and D100 (*PhaseShift*) value is 10,000. When M1(*Execute*) of MC_PhasingAbsolute changes to True, the velocity, acceleration and deceleration and phase shift set in the instruction will be superimposed to the master axis.
- The execution of MC_PhasingAbsolute does not affect the operation of master axis but the operation of slave axis according to gear relationship. As the diagram shows above, when M20 (*Done*) shifts to True at master position 25,000, the slave position can be obtained as below:

  <u>Actual slave position after the phase shift</u> = (<u>Current master position</u> - <u>previous master position</u> + <u>*PhaseShift* value</u>)*(RatioNumerator / RatioDenominator) + <u>previous slave position</u> = (25,000-10,000+10,000)*2+20,000 = 70,000.
- When D100 (*PhaseShift*) is changed to -10,000 and M1 (*Execute*) changes to True again, the instruction will be executed again. As the diagram shows above, when M20 *(Done*) changes to True at master position 50,000, the slave position can be obtained as below:

  <u>Actual slave position after the phase shift</u> = (<u>Current master position</u> - <u>previous master position</u> + <u>*PhaseShift* value</u>)*(RatioNumerator / RatioDenominator) + original slave position = (50,000-35,000-10,000)*2+90,000 = 100,000.

- **Supported Products**

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

3.3 Delta-defined Motion Control Instructions

	Categories	Name	Description
	Single axis Administrative	DFB_AxisSetting1	DFB_AxisSetting1 sets motion parameters for the specified axis.
		DFB_AxisSetting2	DFB_AxisSetting2 sets motion parameters for the specified axis.
		DFB_InputPolarity	DFB_InputPolarity sets the polarity of inputs and reads the states of these input terminals.
		DFB_CamMultiRead	DFB_CamRead reads cam points from the specified motion axis.
		DFB_CamMultiWrite	DFB_CamWrite writes cam points to the specified cam curve
		DFB_CamCurve2	DFB_CamCurve2 creates cam curves which are mainly used in rotary cut and flying saw applications.
		DFB_CamCurveUpdate2	DFB_CamCurveUpdate2 updates the cam operation with the modified cam profile in the next cycle.
	Group Motion	DFB_GroupAbsLinear	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified absolute target position.
		DFB_GroupRelLinear	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified relative distance.
		DFB_GroupAbsCircular	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move to the specified absolute target position.
		DFB_GroupRelCircular	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move a specified relative distance.
	Stop	DFB_GroupStop	DFB_GroupStop decelerates the group axes to stop or pause to the current position.
	Multi-axes Administrative	DFB_GroupEnable	DFB_GroupEnable enables a group of axes for group motion.
		DFB_GroupDisable	DFB_GroupDisable disables the axis group with the specidied group number.
		DFB_GroupReset	DFB_GroupReset resets the axis group which is in the state of “Errorstop”.

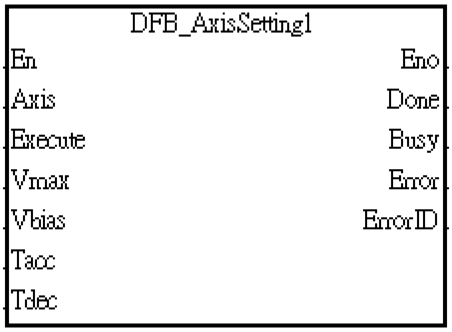
	Categories	Name	Description
		DFB_ReadGroupStatus	DFB_ReadGroupStatus reads the axis numbers in an axis group, and indicates the status of the axis group at *GroupStatus*.
Auxiliary	High speed counter	DFB_HCnt	DFB_HCnt enables the specified high speed counter according to the specified parameters and monitors the count value.
	High speed timer	DFB_HTmr	DFB_HTmr enables the specified high speed timer channel according to the specified parameters and monitors and timed value.
	Comparison	DFB_Compare	DFB_Compare compares the designated source with a specified value and outputs the specified results on a desired device when the comparison result is True.
		DFB_CmpOutRst	DFB_CmpOutRst monitors the output results and clears the output states triggered by the comparators.
	Capture	DFB_Capture2	DFB_Capture2 captures the commanded pulses of the specified axis according to the designated external trigger deivce.
Network	ECAT Communication	DFB_ECATReset	DFB_ECATReset resets an abnormal EtherCAT network.
		DFB_ECATServoRead	DFB_ECATServoRead reads the values of parameters from the Delta servo drive specified on an EtherCAT network.
		DFB_ECATServoWrite	DFB_ECATServoWrite writes the values of parameters into the Delta servo drive specified on an EtherCAT network.
		DFB_SDO_Write	DFB_SDO_Write writes the values of parameters into the specified OD of the EtherCAT Slave via SDO.
		DFB_SDO_Read	DFB_SDO_Read reads the values of parameters from the specified OD of the EtherCAT Slave via SDO.

3.3.1 Single-axis Motion Control Function Blocks

Categories		Name	Description
	Single axis Administrative	DFB_AxisSetting1	DFB_AxisSetting1 sets motion parameters for the specified axis.
		DFB_AxisSetting2	DFB_AxisSetting2 sets motion parameters for the specified axis.
		DFB_InputPolarity	DFB_InputPolarity sets the polarity of inputs and reads the states of these input terminals.
		DFB_CamMultiRead	DFB_CamRead reads cam points from the specified motion axis.
		DFB_CamMultiWrite	DFB_CamWrite writes cam points to the specified cam curve
		DFB_CamCurve2	DFB_CamCurve2 creates cam curves which are mainly used in rotary cut and flying saw applications.

DFB_AxisSetting1

FB/FC	Description
FB	DFB_AxisSetting1 sets motion parameters for the specified axis.



The function blocks Vbias, Tacc and Tdec are for the axis velocity settings in the DFB series function blocks.

This instruction can only be executed when the state is in Disable, Standstill or Coordinated and the axis should be in Standby mode. Otherwise, an error message will be sent.

Note: refer to DFB_AxisSetting2 for setting other motion axis parameters.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Writes in the parameters when *Execute* changes to True.	BOOL	True/False (False)	-
Vmax	Maximum speed (user unit: s)	LREAL	0~2147483647.0 (0)	When *Execute* shifts to True and *Busy* is False.
Vbias	Start-up speed (user unit: s)	LREAL	0~100000.0 (0)	When *Execute* shifts to True and *Busy* is False.
Tacc	Acceleration time (unit: ms)	WORD	0~32767 (0)	When *Execute* shifts to True and *Busy* is False.
Tdec	Deceleration time (unit: ms)	WORD	0~32767 (0)	When *Execute* shifts to True and *Busy* is False.

Outputs

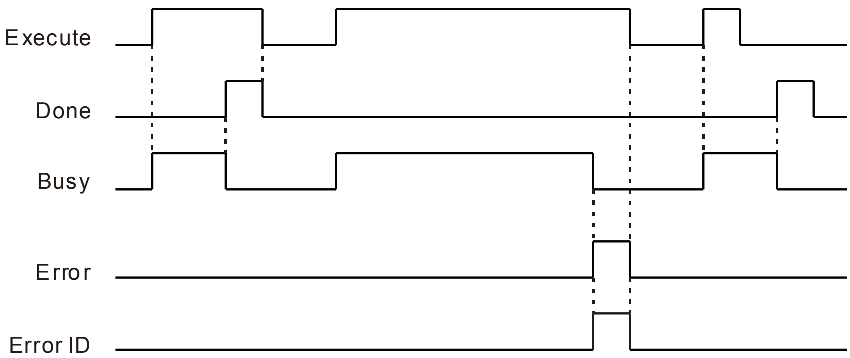
Name	Function	Data type	Output range (Default value)
Done	True when the parameter setting is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs.	DWORD	16#0~16#FFFFFFFF (0)

	Refer to **Appendices** for error code descriptions.		

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the specified target distance is completed	● When Execute shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● In-Outs

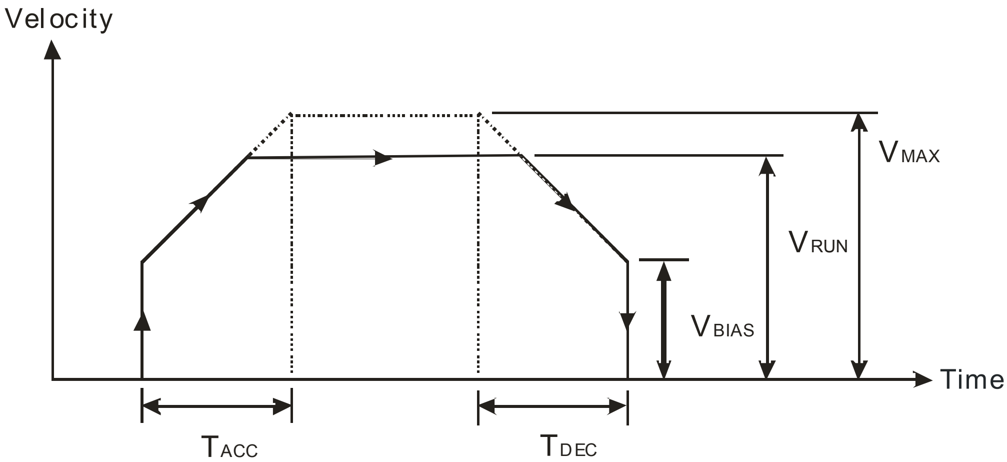
Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising-edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Function

DFB_AxisSetting1 sets motion parameters for the specified axis. You can set the motion parameters including maximum speed of the axis (*Vmax*), the start-up speed (*Vbias*), the time it takes for the start-up speed to increase to its running speed (*Tacc*), and the time it takes for the running speed to decrease to its start-up speed (*Tdec*).

The motion diagram explaining the above parameters is as below.



Note: V_{RUN} indicates the actual operation speed which is calculated based on the specified values of *Tacc*, *Vbias*, *Tdec* and *Vmax*.

Troubleshooting

3

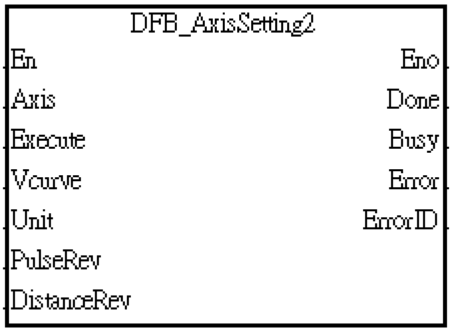
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, Error will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual*****.**

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_AxisSetting2

FB/FC	Description
FB	DFB_AxisSetting2 sets motion parameters for the specified axis.



- DFB _AxisSetting2 sets motion parameters for the specified axis. You can set the motion parameters including velocity curve (*Vcurve*), the pulse output type (*OutputType*), and the user unit system (*Unit*). For example, the setting of pulse number for a motor to rotate once (*PulseRev*) and the moving distance when the motor rotates once (*DistanceRev*) should be specified if mechanical unit system is required.
- This instruction can only be executed when the state is in Disable, Standstill or Coordinated and the axis should be in Standby mode. Otherwise, an error message will be sent.

Note: refer to DFB_AxisSetting1 for setting other motion axis parameters.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Writes in the parameters when *Execute* changes to True.	BOOL	True/False (False)	-
Vcurve	Velocity curve	BOOL	Trapezoid: False S Curve: True (False)	When *Execute* shifts to True and *Busy* is False.
PulseRev	Number of pulses per motor revolution	DWORD	1～99999999 (0)	When *Execute* shifts to True and *Busy* is False.
DistanceRev	Travel distance per motor revolution	LREAL	1~1000000.0 (0)	When *Execute* shifts to True and *Busy* is False.

Outputs

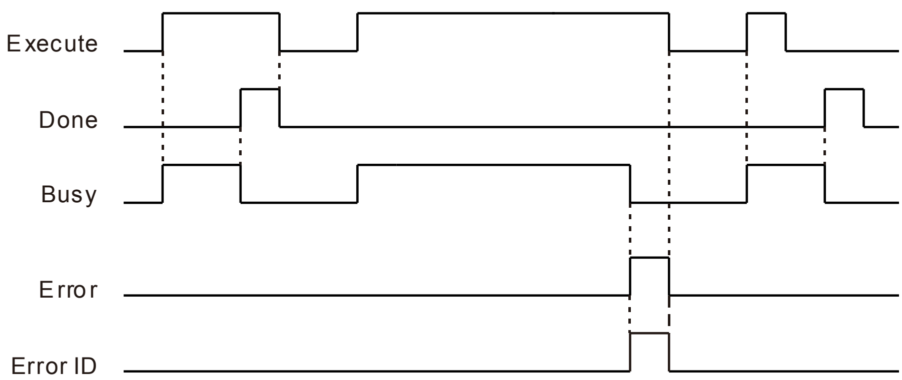
Name	Function	Data type	Output range (Default value)
Done	True when the parameter setting is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)

Name	Function	Data type	Output range (Default value)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

- **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the specified target distance is completed	• When Execute shifts from True to False. • If Execute is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When Execute changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

- **Timing Diagram**



In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising-edge triggered and *Busy* is False.

***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

Troubleshooting

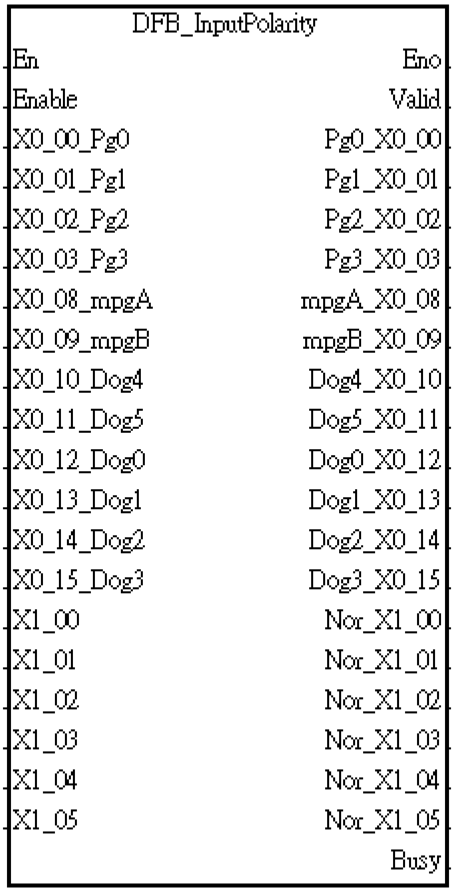
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", Error will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual*.**

- **Supported Products**

  - AH Motion Controller CPU: AHxxEMC-5A
  - AH series motion control modules: AHxxEMC-5A

DFB_InputPolarity

FB/FC	Description
FB	DFB_InputPolarity sets the polarity of inputs and reads the states of these input terminals.



- You can specify the polarity of the inputs as NO (Normally Open) or NC (Normally Closed), and read the states of these input terminals by the outputs of this instruction.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Enables the instruction when Enable changes to True	BOOL	True/False (False)	-
X0_00_Pg0	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_00_Pg1	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.

Name	Function	Data type	Setting value (Default value)	Timing for updating
X0_00_Pg2	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_00_Pg3	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_08_mpgA	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_09_mpgB	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_10_Dog4	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_11_Dog5	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_12_Dog0	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_13_Dog1	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_14_Dog2	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X0_15_Dog3	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X1_00	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X1_01	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X1_02	Polarity (NO/NC) setting	BOOL	mcNO: False	Continuously updates the state when

Name	Function	Data type	Setting value (Default value)	Timing for updating
			mcNC: True (False)	*Valid* is True.
X1_03	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X1_04	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.
X1_05	Polarity (NO/NC) setting	BOOL	mcNO: False mcNC: True (False)	Continuously updates the state when *Valid* is True.

3

- **Outputs**

Name	Function	Data type	Output range (Default value)
Valid	True when the polarity setting is completed.	BOOL	True/False (False)
Busy	True when the instruction is enabled.	BOOL	True/False (False)
Pgo_X0_00	State of the input terminal	BOOL	True/False (False)
Pg1_X0_01	State of the input terminal	BOOL	True/False (False)
Pg2_X0_02	State of the input terminal	BOOL	True/False (False)
Pg3_X0_03	State of the input terminal	BOOL	True/False (False)
mpgA_X0_08	State of the input terminal	BOOL	True/False (False)
mpgB_X0_09	State of the input terminal	BOOL	True/False (False)
Dog4_X0_10	State of the input terminal	BOOL	True/False (False)
Dog5_X0_11	State of the input terminal	BOOL	True/False (False)
Dog0_X0_12	State of the input terminal	BOOL	True/False (False)
Dog1_X0_13	State of the input terminal	BOOL	True/False (False)
Dog2_X0_14	State of the input terminal	BOOL	True/False (False)
Dog3_X0_15	State of the input terminal	BOOL	True/False (False)
Nor_X1_00	State of the input terminal	BOOL	True/False (False)
Nor_X1_01	State of the input terminal	BOOL	True/False (False)
Nor_X1_02	State of the input terminal	BOOL	True/False (False)
Nor_X1_03	State of the input terminal	BOOL	True/False (False)

Name	Function	Data type	Output range (Default value)
Nor_X1_04	State of the input terminal	BOOL	True/False (False)
Nor_X1_05	State of the input terminal	BOOL	True/False (False)

■ Outputs Update Timing

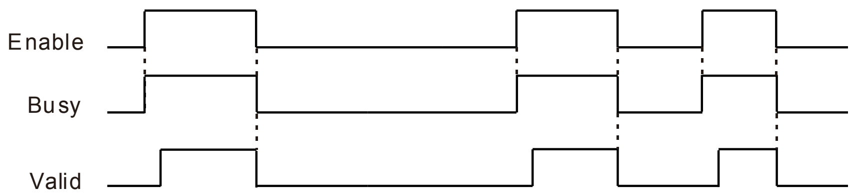
Name	Timing for shifting to True	Timing for shifting to False
Vaild	● When the polarity setting is completed; one scan cycle after *Enable* shifts to True.	● When *Enable* shifts to False.
Busy	● When Enable shifts to True.	● When *Enable* shifts to False.
Pgo_X0_00	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
Pg1_X0_01	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
Pg2_X0_02	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
Pg3_X0_03	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
mpgA_X0_08	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
mpgB_X0_09	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
Dog4_X0_10	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation.

3

	signal is ON during operation.	• When the output is OFF and the external signal is OFF during operation.
Dog5_X0_11	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Dog0_X0_12	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Dog1_X0_13	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Dog2_X0_14	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Dog3_X0_15	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Nor_X1_00	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Nor_X1_01	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Nor_X1_02	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Nor_X1_03	• When the output is ON and the external signal is OFF during operation. • When the output is OFF and the external signal is ON during operation.	• When *Enable* shifts to False. • When the output is ON and the external signal is ON during operation. • When the output is OFF and the external signal is OFF during operation.
Nor_X1_04	• When the output is ON and the external	• When *Enable* shifts to False.

	signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.
Nor_X1_05	● When the output is ON and the external signal is OFF during operation. ● When the output is OFF and the external signal is ON during operation.	● When *Enable* shifts to False. ● When the output is ON and the external signal is ON during operation. ● When the output is OFF and the external signal is OFF during operation.

■ **Timing Diagram**



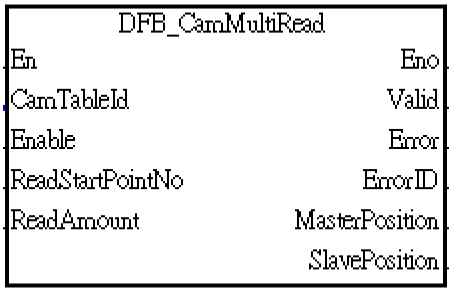
● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

3

DFB_CamMultiRead

FB/FC	Description
FB	DFB_CamRead reads cam points from the specified motion axis.



- The *CamTableId* decides the motion axis of the cam curve where the cam point will be read. *ReadStartPointNo* defines the starting number of the cam point. *ReadAmount* defines the total amount of CAM data to be read from the value specified in ReadStartPointNo (the starting number of the cam point). *MasterPosition* and *SlavePosition* define the master and slave positions (coordinates of the cam point) read from the motion axis.

***Note:**

1. CAM table ID is the number corresponding to the ID Number of the created E-CAM table in ISPSoft.
2. Refer to the resolution of the E-CAM table in the ISPSoft for the range of the data points.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
CamTableId	CAM table ID*1	WORD	K1~K32 (0)	When Enable shifts to True.
Enable	Executes the instruction when *Enable* changes to True. The specified cam point will then be read from the cam curve.	BOOL	True/False (False)	-
ReadStartPoint No	The starting number of the cam point to be read	DWORD	0~2047 (0)	When *Enable* shifts to True. it udates value continuously.
ReadAmount	The total amount of CAM data to be read	WORD	1~64 (0)	When *Enable* shifts to True, it udates value continuously.

● Outputs

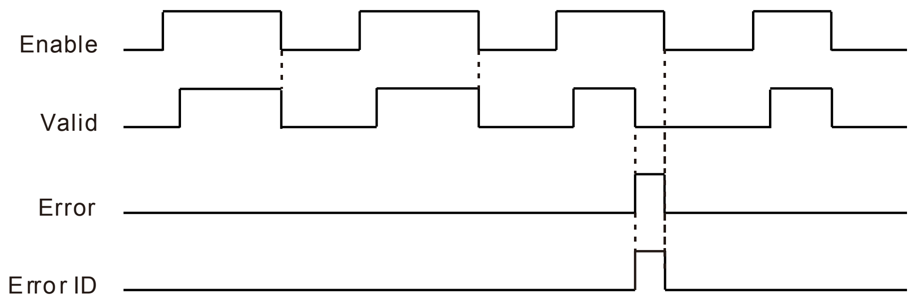
Name	Function	Data type	Output range (Default value)
Valid	True when the specified cam point is read from the cam curve.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

MasterPosition	Master position which is read.	LREAL	Positive value (0)
SlavePosition	Slave position which is read.	LREAL	Positive or negative value (0)

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Valid	● When the specified cam point is read from the cam curve.	● When *Enable* shifts to False.
Error(ErrorID)	● When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	● When *Execute* shifts from True to False. (error code is cleared)
MasterPosition	● Updates value continuously when the instruction is enabled.	● Updates value continuously when the instruction is enabled.
SlavePosition	● Updates value continuously when the instruction is enabled.	● Updates value continuously when the instruction is enabled.

■ **Timing Diagram**



● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

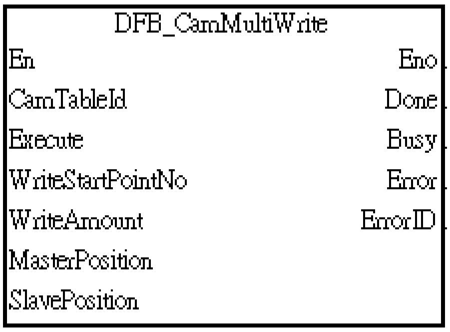
Please refer to the programming example of DFB_CamWrite.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_CamMultiWrite

FB/FC	Description
FB	DFB_CamWrite writes cam points to the specified cam curve



- The *CamTableId* decides the motion axis of the cam curve which is to be written with the cam point. *CamPointNo* defines the number of the cam point; *MasterPosition* and *SlavePosition* define the master and slave positions (coordinates of the cam point) which are to be written into the cam profile.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
CamTableId	CAM table ID*1	WORD	K1~K32 (0)	When Execute shifts to True and Busy is False
Execute	Executes the instruction when *Execute* changes to True. The specified cam point will then be written into the cam curve.	BOOL	True/False (False)	-
WriteStartPointNo	The starting number of the cam point to be writeen	DWORD	0~2047 (0)	When Execute shifts to True and Busy is False
WriteAmount	The total amount of CAM data to be written	WORD	1~64 (0)	When Execute shifts to True and Busy is False
MasterPosition	Master position which is to be written.	Array [64] of LREAL	Positive value (0)	When *Execute* shifts to True and Busy is False
SlavePosition	Slave position which is to be written.	Array [64] of LREAL	Positive or negative value (0)	When *Execute* shifts to True and Busy is False

***Note:**

1. CAM table ID is the number corresponding to the ID Number of the created E-CAM table in ISPSoft.
2. Refer to the resolution of the E-CAM table in the ISPSoft for the range of the data points.

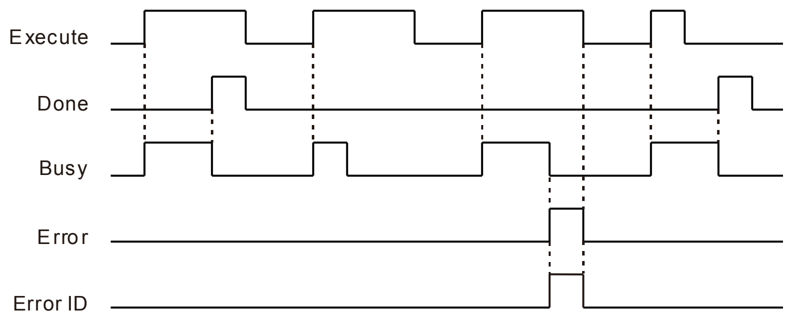
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the specified cam point is written into the cam curve.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the specified cam point is written into the cam curve.	• When *Execute* shifts from True to False. • If Execute is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When Execute changes to True.	• When *Done* shifts to True. • When *Error* shifts to False.
Error(ErrorID)	• When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	• When *Execute* shifts from True to False. (error code is cleared)

Timing Diagram



Troubleshooting

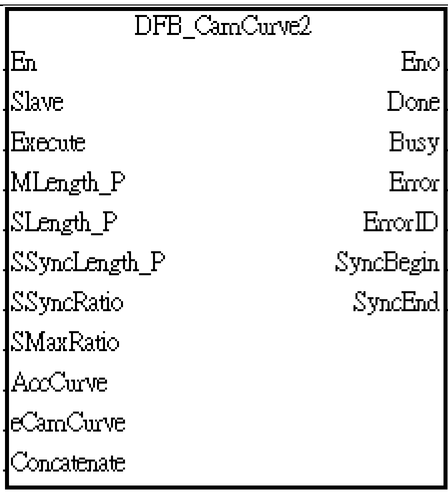
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_CamCurve2

FB/FC	Description
FB	DFB_CamCurve2 creates cam curves which are mainly used in rotary cut and flying saw applications.



- *MLength_P*, *SLength_P*, *SSyncLength_P*, *SSyncRatio* and *SMaxRatio* specify the required physical quantity parameters for creating cam curves.
- *AccCurve* and *aCamCurve* define the acceleration curve in the cam profile and the cam curve type.
- This instruction can be used with *MC_CAMIN* and *DFB_CAMIN2.*
- After the execution of this instruction is done, users need to use *DFB_CamCurveUpdate2* to update.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
CamTableId	CAM table ID*1	WORD	K1~Kn*1 (0)	When *Execute* shifts to True and *Busy* is Fasle.
Execute	Executes the instruction to create the cam curve	BOOL	True/False (False)	-
MLength_P	Specifies the moving distance for master axis.	LREAL	K1~K2147483647 (0)	When *Execute* shifts to True and *Busy* is Fasle.
SLength_P	Specifies the moving distance for slave axis.	LREAL	K1~K2147483647 (0)	When *Execute* shifts to True and *Busy* is Fasle.
SSyncLength_P	Specifes the synchronized distance for the slave axis.	LREAL	K1~K2147483647 (0)	When *Execute* shifts to True and *Busy* is Fasle.
SSyncRatio	Specifies the synchronization ratio between master and slave axes.	REAL	1.1755x10-38~ 3.4028x10+38 (0)	When *Execute* shifts to True and *Busy* is Fasle.
SMaxRatio	Specifies the max. cam synchronization ratio between master and slave axes.	REAL	1.1755x10-38 ~ 3.4028x10+38 (0)	When *Execute* shifts to True and *Busy* is Fasle.
AccCurve*2	Selects the acceleration curve in the cam profile.	eDFB_ACC_CURVE	0：ConstAcc 1：ConstJerk 2：SingleHypot	When *Execute* shifts to True and *Busy* is Fasle.

Name	Function	Data type	Setting value (Default value)	Timing for updating
			3 : Cycloid (0)	
eCamCurve*3	Selects the cam curve type.	eDFB_GEN_CURVE	0 : leftCAM 1 : midCAMall 5 : rightCAM 7 : midCAMbegin 8 : midCAMend 9 : midCAMzero (0)	When *Execute* shifts to True and *Busy* is Fasle.
Concatenate	Defines if reciprocal operation is required. True: periodical False: non-periodical	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is Fasle.

3

***Note:** Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

1. Set value of *AccCurve.* Refer to **Function** of this instruction for the cam profiles of each acceleration curve.

Setting Value	Definition
0	Constant speed
1	Const Acceleration
2	SingleHypot
3	Cycloid

2. Set value of *eCamCurve.* Refer to **Function** of this instruction for the cam profiles of each type.

Setting Value	Definition
0	leftCAM
1	midCAMall
2	midCAMbegin
3	midCAMend
4	rightCAM
5	rightCAM
7	midCAMbegin
8	midCAMend
9	Empty

● Outputs

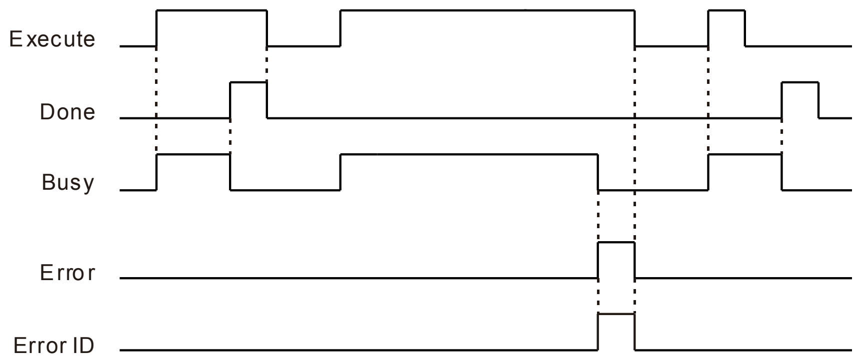
Name	Function	Data type	Output range (Default value)
Done	True when the specified cam point is written into the cam curve.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

SyncBegin	The starting point to synchronizate	LREAL	K0~K2147483647（0）
SyncEnd	The stopping point to synchronizate	LREAL	K0~K2147483647（0）

■ Outputs Update Timing

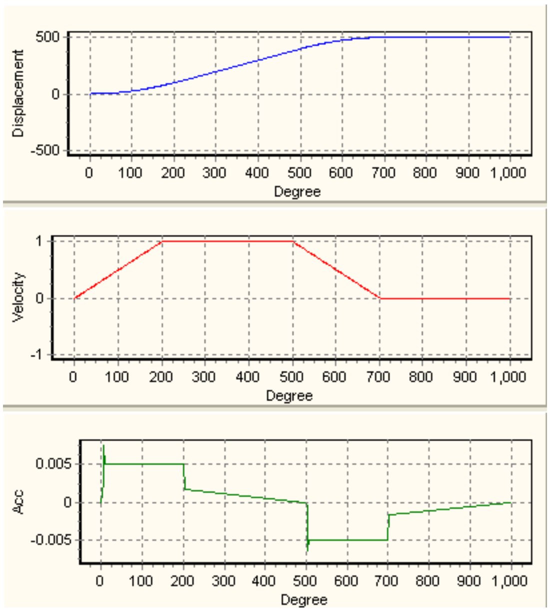
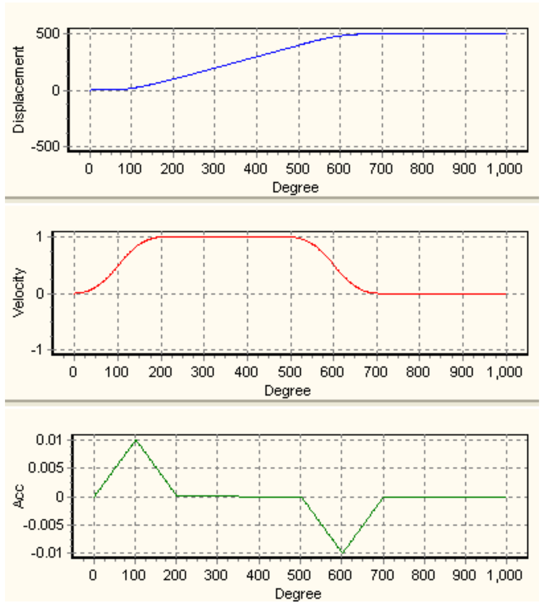
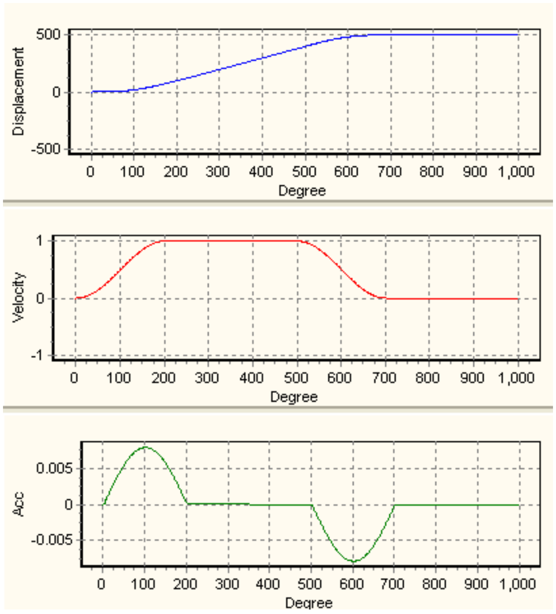
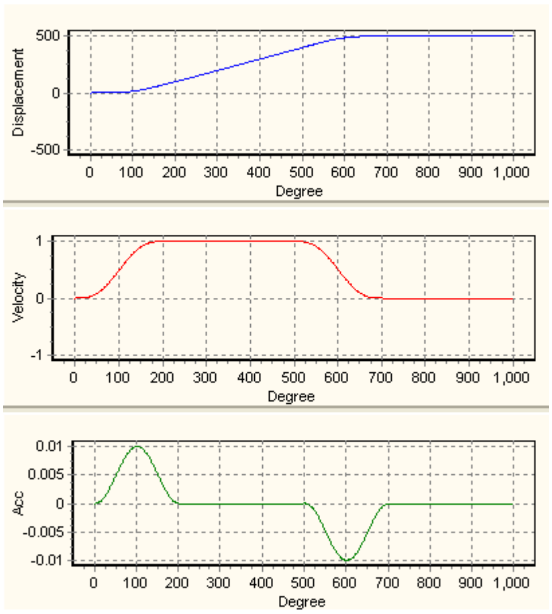
Name	Timing for shifting to True	Timing for shifting to False
Done	• When the specified cam point is written into the cam curve.	• When *Execute* shifts from True to False. • When *Error* shifts to True.
Busy	• When Execute changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	• When *Execute* shifts from True to False. (error code is cleared)
SyncBegin	• When *Done* shifts to True.	• When *Done* shifts to True.
SyncEnd	• When *Done* shifts to True.	• When *Done* shifts to True.

■ Timing Diagram



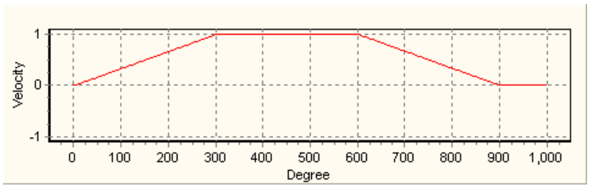
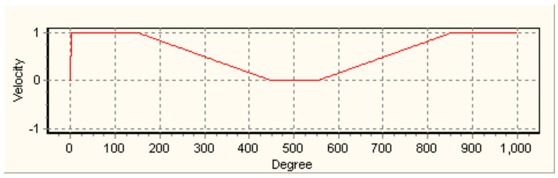
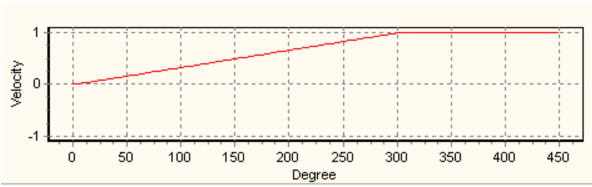
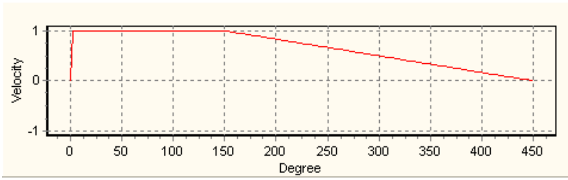
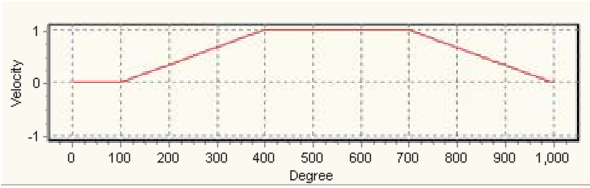
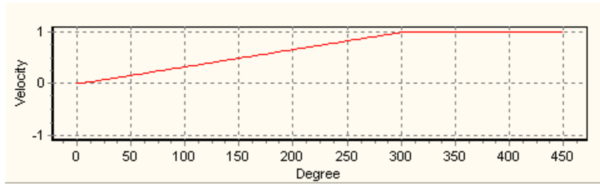
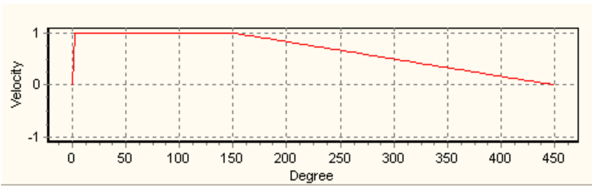
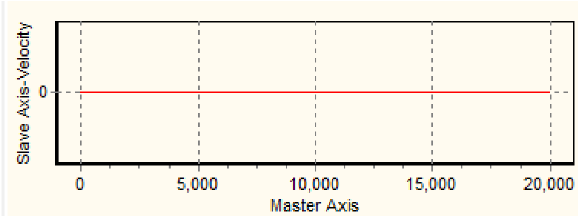
***Note:** Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

- **Explanation**
  - ***AccCurve***

0 : Const speed	1 : Constant Acceleration
	
2 : SingleHypot	3 : Cycloid
	

3

■ ***eCamCurve***

0 : leftCAM	1 : midCAMall
	
2 : midCAMbegin	3 : midCAMend
	
5 : right CAM	7 : midCAMbegin
	 **Note**: In bag packaging applications, this cam profile will set the length of the FIRST package to be the same as the length of one pack of the material.
8 : midCAMend	9 : Empty
 **Note**: In bag packaging applications, this cam profile will set the length of the LAST package to be the same as the length of one pack of the material.	 **Note**: In bag packaging applications, this cam profile will set the length of master axis to be the multiple of the length of the material, and the slave axis velocity will be fixed at 0.

● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Programming Example

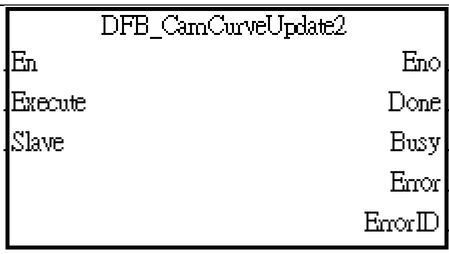
Refer to the programming example of DFB_CamCurveUpdate2 for the programming example and the methods of how to use DFB_CamCurveUpdate2 to work with DFB_CamCurve2 instructions.

- **Supported Products**

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_CamCurveUpdate2

FB/FC	Description
FB	DFB_CamCurveUpdate2 updates the cam operation with the modified cam profile in the next cycle.



- When the cam is in operation, the cam profile modified by DFB_CamCurve2 will not be executed immediately, and the update requires to be triggered by DFB_CamCurveUpdate2.
- When DFB_CamCurveUpdate2 is triggered, the update will be valid in the next cycle.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction to update the cam profile.	BOOL	True/False (False)	-
UpdateImmediately	Update cam profile immediately in this cycle.	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is Fasle.

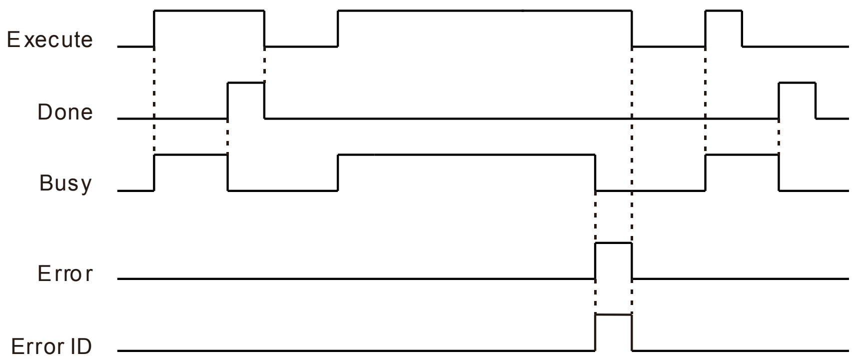
● Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the cam curve is updated.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	0x0000, 0x3100, 0x3101, 0x3102 (0)

■ Outputs Update Timing

Name	Timing for changing to True	Timing for changing to False
Done	● When the cam curve is updated.	● When the motion stops. ● When *Error* shifts to True.
Busy	● When the instruction is executed	● When *Done* shifts to True. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. ● (Error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Slave	Slave axis number	WORD	K1~K32 (0)	When *Enable* is rising edge triggered and *Busy* is False.

***Note:** Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

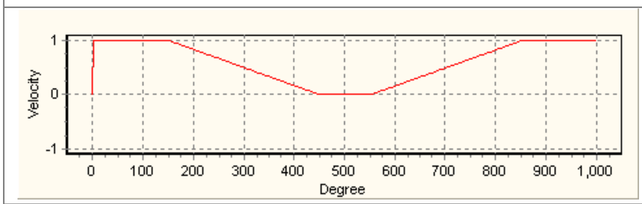
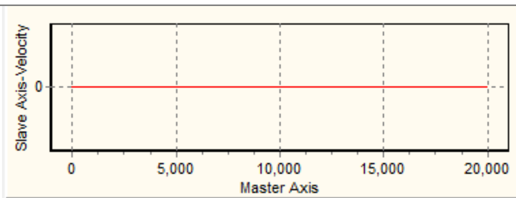
● Troubleshooting

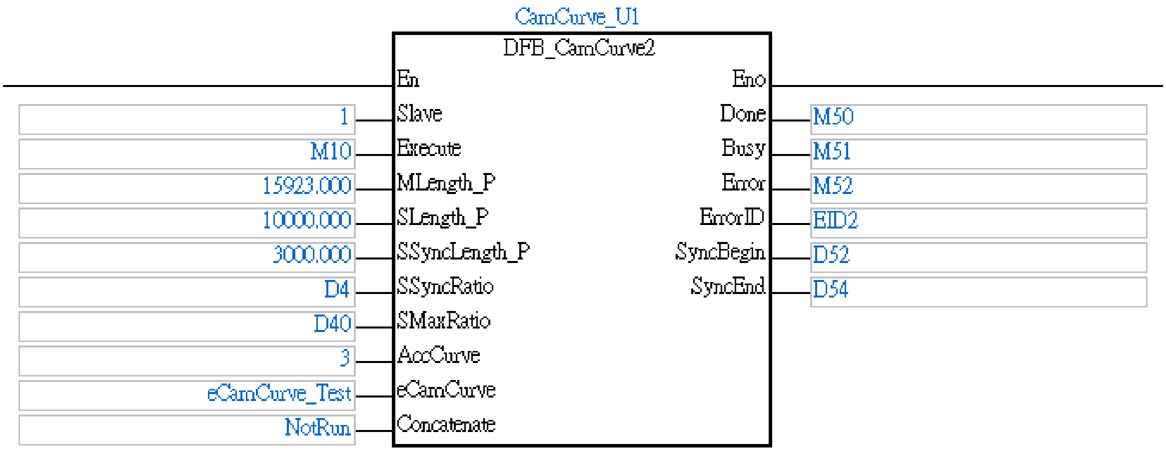
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

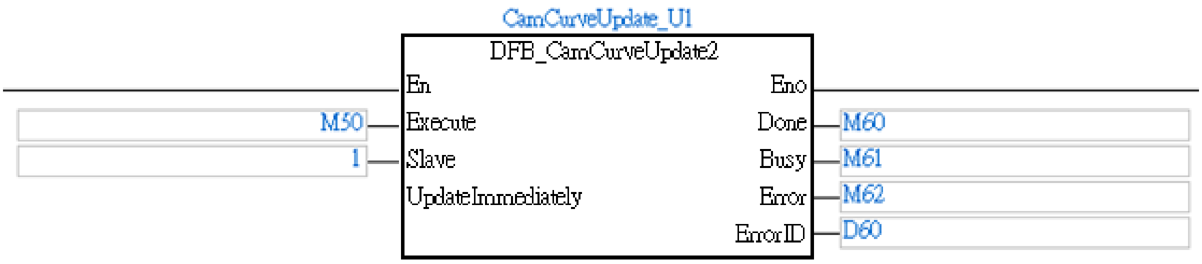
● Programming Example

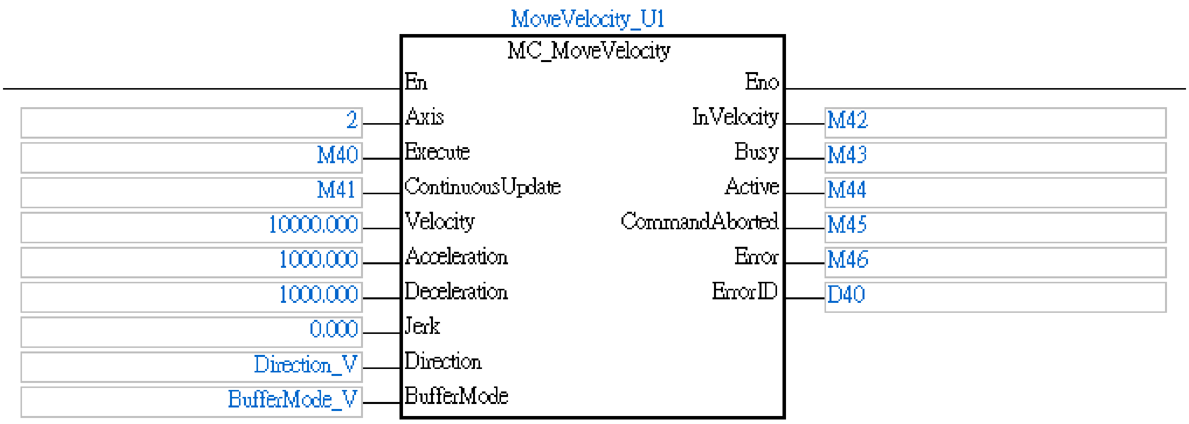
The example demonstrates the switching from cam curve 1 (*eCamCurve*=1) to cam curve 9 (*eCamCurve*=9) when the cam is in operation. To change the cam profile, you will have to modifiy the value in *eCamCurve* of DFB_CamCurve2.

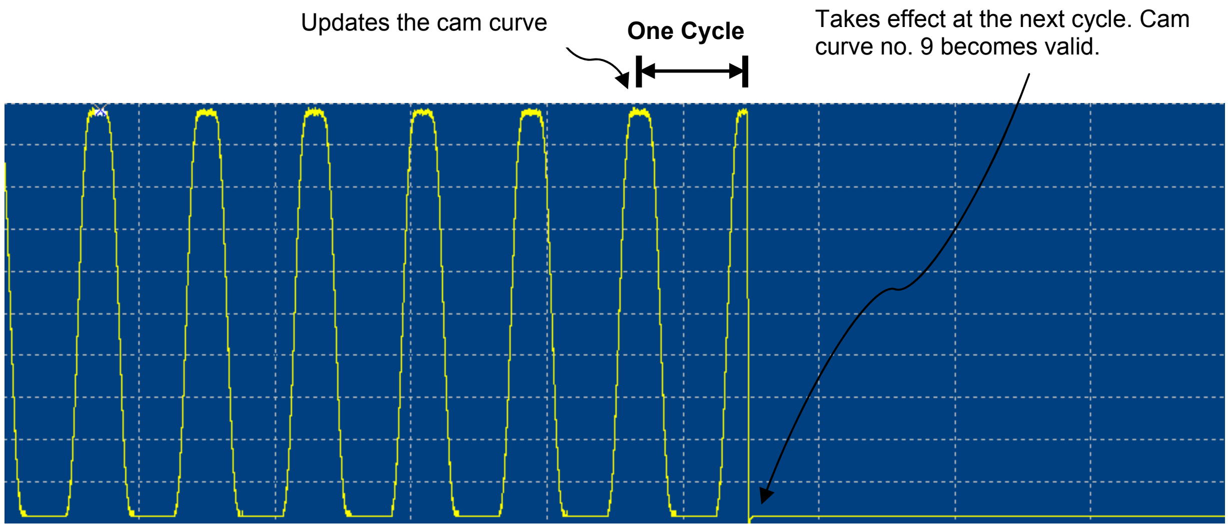
Note: In actual application, the common switching process would be from cam curve 1 to cam curve 8, then cam curve 9. The direct switching from cam curve 1 to cam curve 9 is simply for demonstrating the effects and results.

1 : midCAMall	9 : Empty
	


CamCurve_U1
DFB_CamCurve2
En
Eno
1
Slave
Done
M50
M10
Execute
Busy
M51
15923.000
MLength_P
Error
M52
10000.000
SLength_P
ErrorID
EID2
3000.000
SSyncLength_P
SyncBegin
D52
D4
SSyncRatio
SyncEnd
D54
D40
SMaxRatio
3
AccCurve
eCamCurve_Test
eCamCurve
NotRun
Concatenate


CamCurveUpdate_U1
DFB_CamCurveUpdate2
En
Eno
M50
Execute
Done
M60
1
Slave
Busy
M61
UpdateImmediately
Error
M62
ErrorID
D60


MoveVelocity_U1
MC_MoveVelocity
En
Eno
2
Axis
InVelocity
M42
M40
Execute
Busy
M43
M41
ContinuousUpdate
Active
M44
10000.000
Velocity
CommandAborted
M45
1000.000
Acceleration
Error
M46
1000.000
Deceleration
ErrorID
D40
0.000
Jerk
Direction_V
Direction
BufferMode_V
BufferMode



1. Set M10(*Execute*) to True to establish the specified cam curve no.1 (DFB_CamCurve2). Set M50(*Execute*) to True to update the cam curve(DFB_CamCurveupdate2). In this case, if SM400(*Enable*) is True, the current cam curve obtained at *CAMCurveNow* will indicate 1 (DFB_CamCurveUpdateState).
2. Set M70 to True to enable the cam operation.
3. Set M40 to True to drive the master axis to move at 10,000pps.
4. Reset M10 and modify the value at eCamCurve to cam curve no. 9.
5. Set M10 to True again to establish the new cam curve, and set M50 to True to update the cam curve. The cam in operation will update its cam curve to cam curve no. 9 at the next cycle. When the cam curve is updated, *CAMCurveNow* of DFB_CamCurveUpdateState2 will indicate "9".

● Supported Products

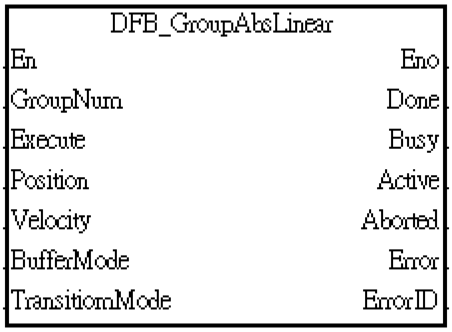
- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

3.3.2 Multi-axis Motion Control Function Blocks

Categories		Name	Description
	Group Motion	DFB_GroupAbsLinear	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified absolute target position.
		DFB_GroupRelLinear	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified relative distance.
		DFB_GroupAbsCircular	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move to the specified absolute target position.
		DFB_GroupRelCircular	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move a specified relative distance.
	Stop	DFB_GroupStop	DFB_GroupStop decelerates the group axes to stop.
	Administrative	DFB_GroupEnable	DFB_GroupEnable enables a group of axes for group motion.
		DFB_GroupDisable	DFB_GroupDisable disables the axis group with the specidied group number.
		DFB_GroupReset	DFB_GroupReset resets the axis group which is in the state of “Errorstop”.
		DFB_ReadGroupStatus	DFB_ReadGroupStatus reads the axis numbers in an axis group, and indicates the status of the axis group at *GroupStatus*.

DFB_GroupAbsLinear

FB/FC	Description
FB	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified absolute target position.



Note: linear interpolation requires at least 2 axes to be enabled for the axis group.

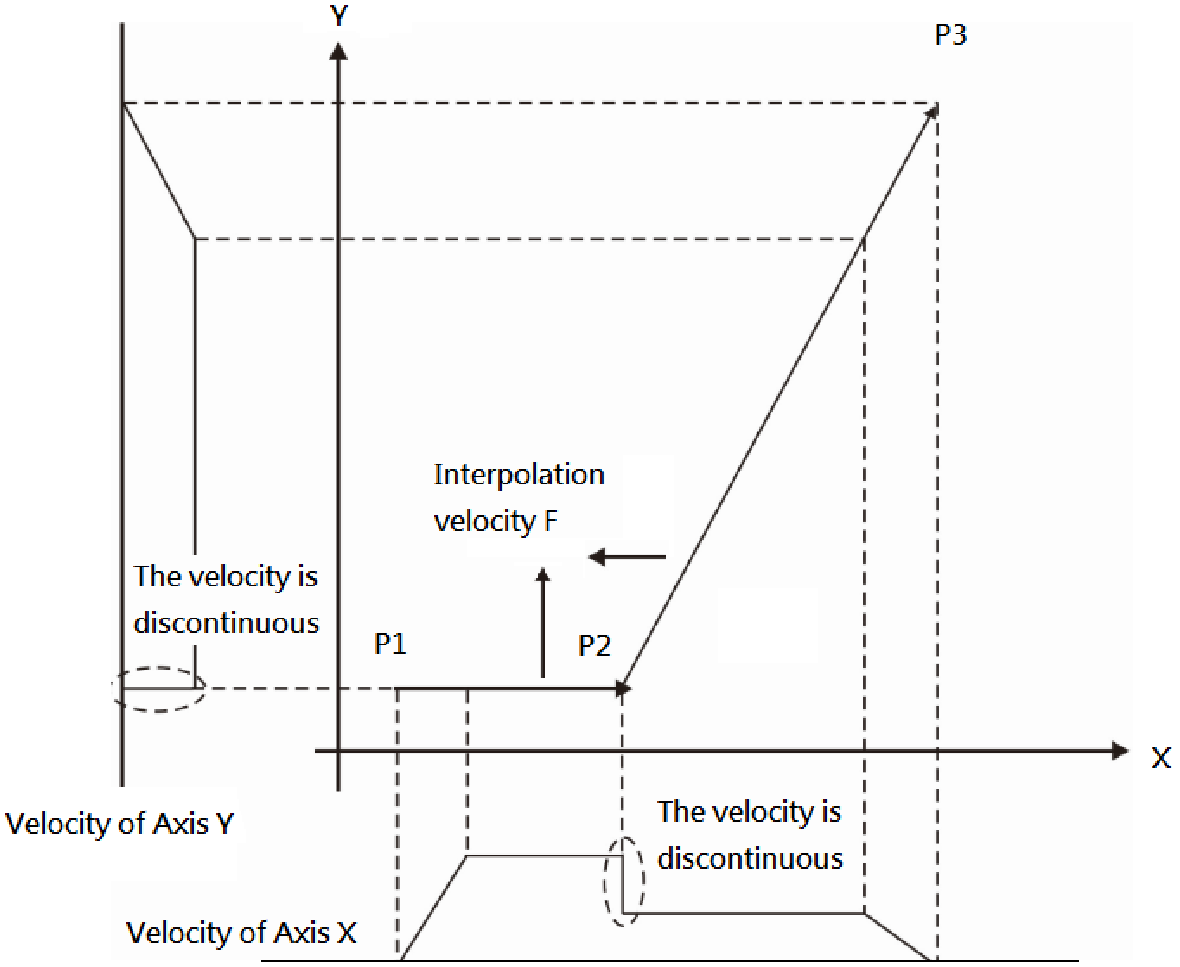
● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Position	Absolute target position for each axis in the group. (Unit: user unit)	LREAL[6]	[_,_,_,_,_,_] Negative number or positive number ([0,0,0,0,0,0])	When *Execute* shifts to True and *Busy* is False.
Velocity	Interpolation speed (Unit: user unit/s)	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*1	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.
TransitionMode	Specifies if a round corner should be applied during the transition*1	WORD	0: no effect 1: round corner 2: round corner but ignoring the deceleration time (0)	When *Execute* shifts to True and *Busy* is False.

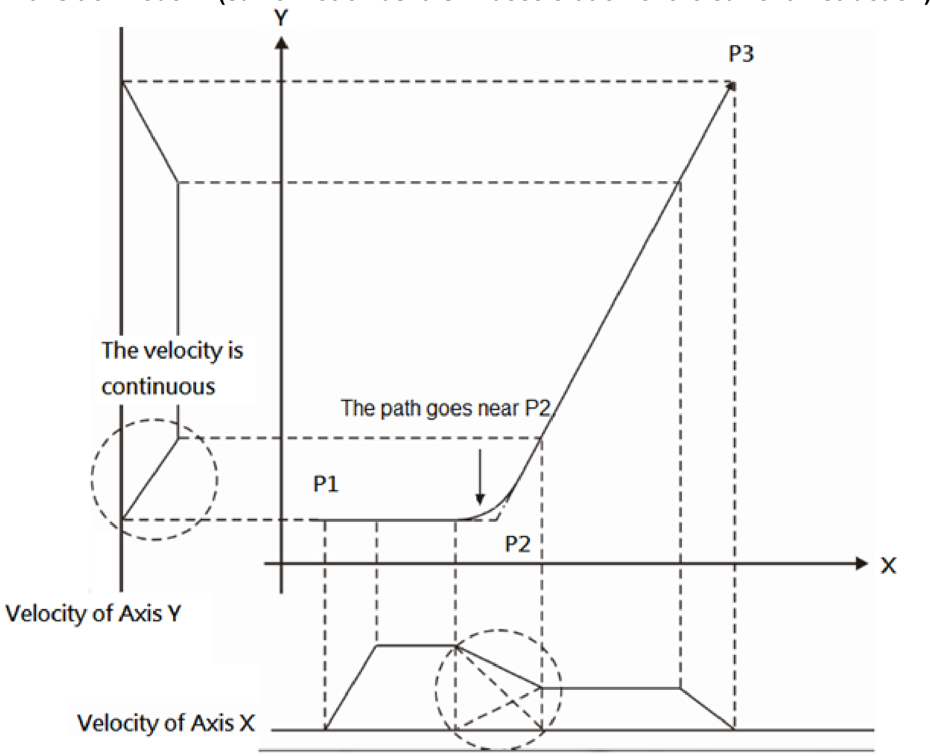
***Note:**

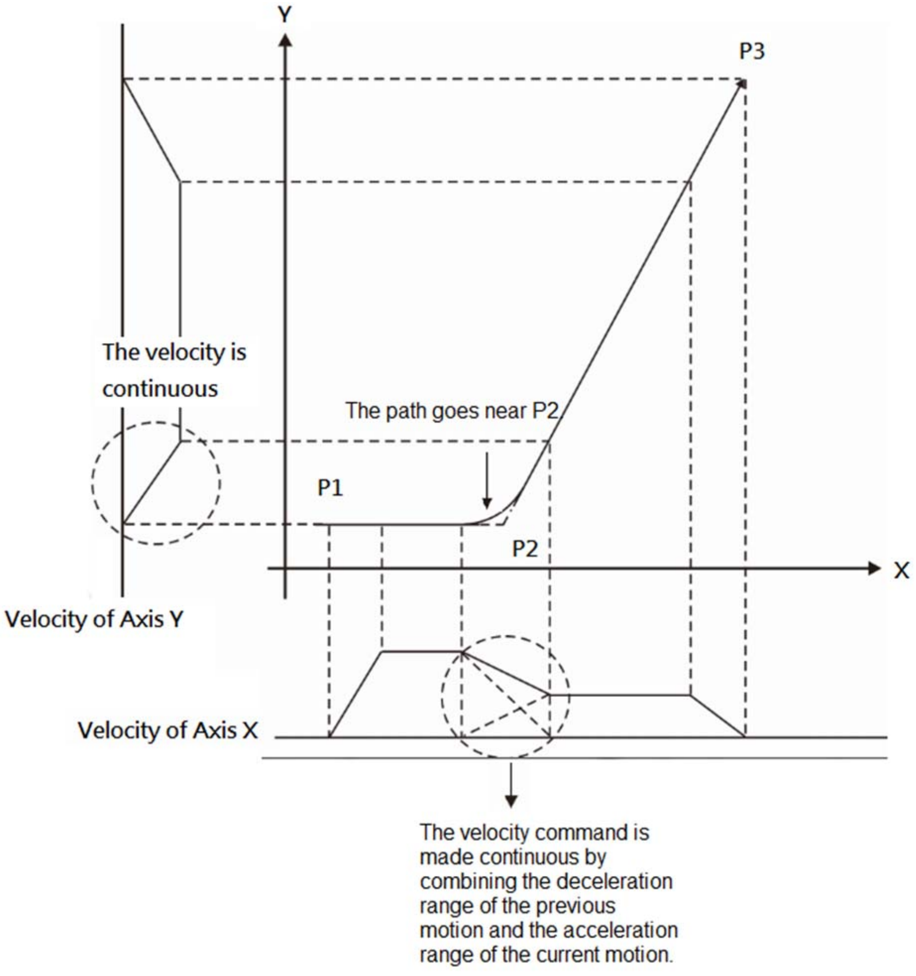
1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
2. TransitionMode can be used to reduce the noise and vibration that may occur during the changes of the interpolation motion.

TransitionMode: 0 (no effect)



TransitionMode: 1 (same motion as it is in deceleration of the current instruction)





TransitionMode: 2 (same motion as it is in deceleration of the axis parameters)

● Outputs

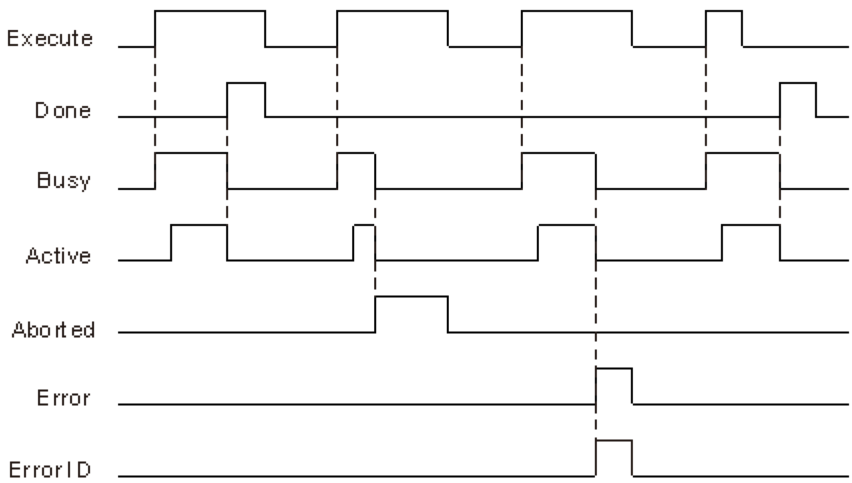
Name	Function	Data type	Output range (Default value)
Done	True when absolute target position is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
Aborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

3

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the absolute positioning is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *Aborted* shifts to True.
Active	● When the motion on the axis is started	● When *Done* shifts to True. ● When *Error* shifts to True ● When *Aborted* shifts to True. ● If *Execute* is False and *Active* shifts to True, it will be True for only one period and immediately shift to False.
Aborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted because of the execution of MC_Stop instruction. ● When the instruction is aborted by DFB_GroupStop instruction ● When the instruction is aborted by DFB_GroupImmediateStop instruction.	● When *Execute* shifts to False. ● If *Execute* is False and *Aborted* shifts to True, it will be True for only one period and immediately shift to False.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



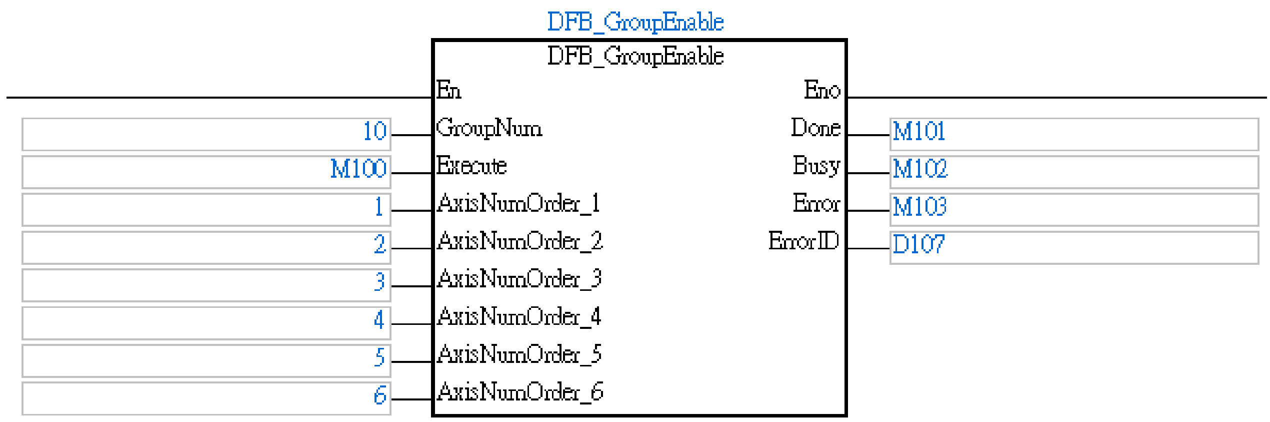
● Troubleshooting

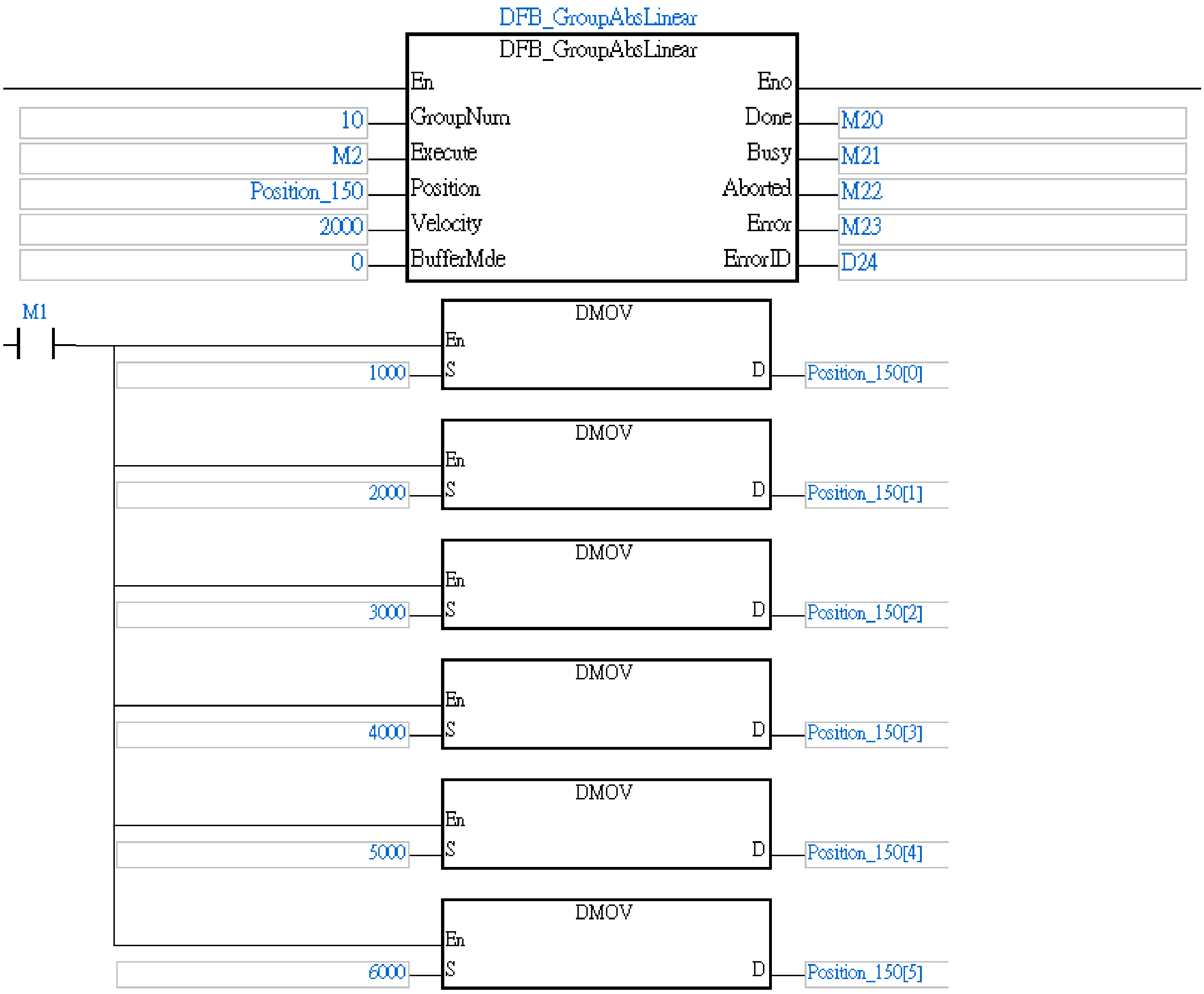
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

● Programming Example

The example performs an linear interpolation from current position to the target position as below.

Axis group	Target position
Axis1	1000
Axis2	2000
Axis3	3000
Axis4	4000
Axis5	5000
Axis6	6000





- When M2(*Execute*) changes to True, DFB_GroupAbsLinear drives the axes to the absolute target position with linear interpolation.
- When the axes reach the specified target position (1000, 2000, 3000, 4000, 5000, 6000), M20(*Done*) changes to True, and M21 and M22 change to False.
- When M2(*Execute*) changes to False, M20 (*Done*) changes to False.
- When the axes reach the target position, re-execution of the instruction will not move any axes.

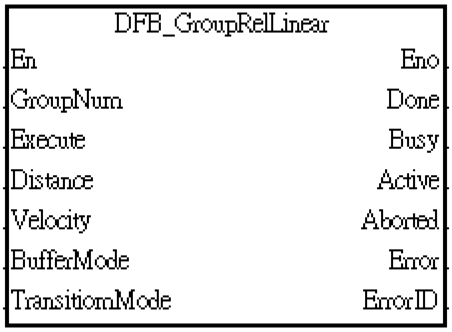
● Supported Products

– AH Motion Controller CPU: AHxxEMC-5A

– AH series motion control modules: AHxxEMC-5A

DFB_GroupRelLinear

FB/FC	Description
FB	DFB_GroupAbsLinear controls the axis group to perform linear interpolation to move to the specified relative distance.



Note: linear interpolation requires at least 2 axes to be enabled for the axis group.

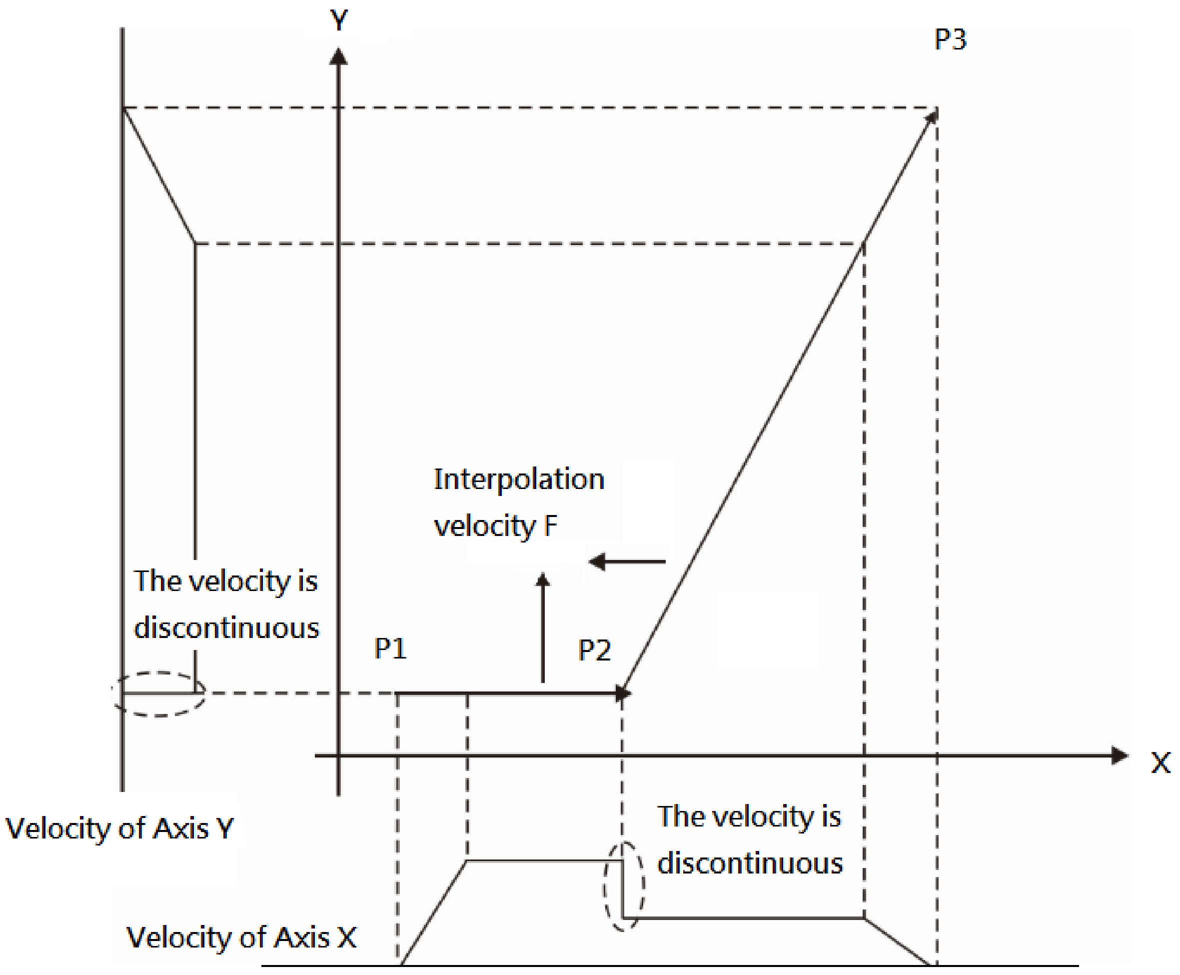
- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
Distance	Relative distance to be moved for each axis in the group. (Unit: user unit)*1	LREAL[6]	[_,_,_,_,_,_] Negative number or positive number ([0,0,0,0,0,0])	When *Execute* shifts to True and *Busy* is False.
Velocity	Interpolation speed (Unit: user unit/s)*1	LREAL	Positive number (0)	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*1	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.
TransitionMode	Specifies if a round corner should be applied during the transition*2	WORD	0: no effect 1: round corner 2: round corner but ignoring the deceleration time (0)	When *Execute* shifts to True and *Busy* is False.

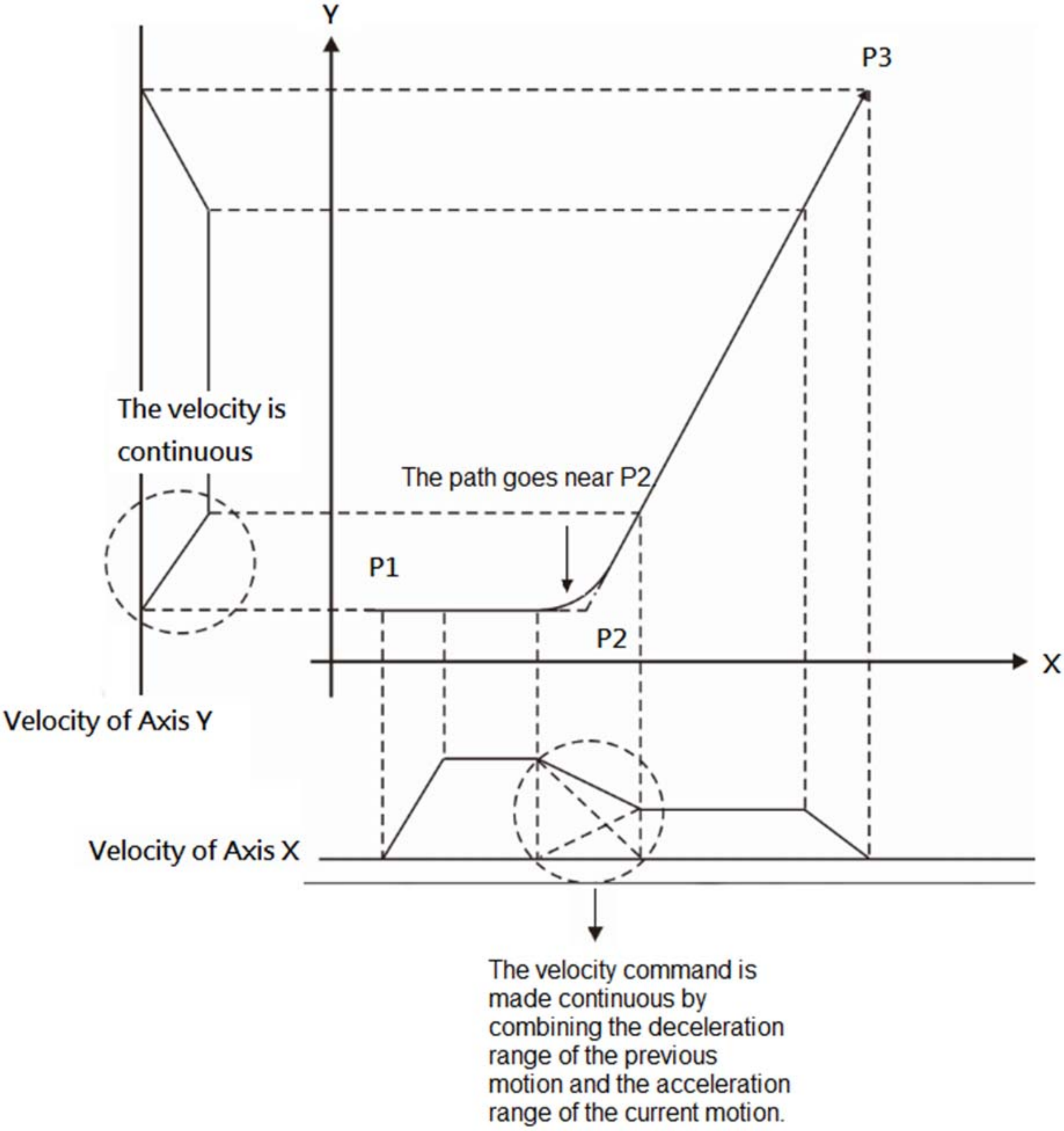
***Note:**

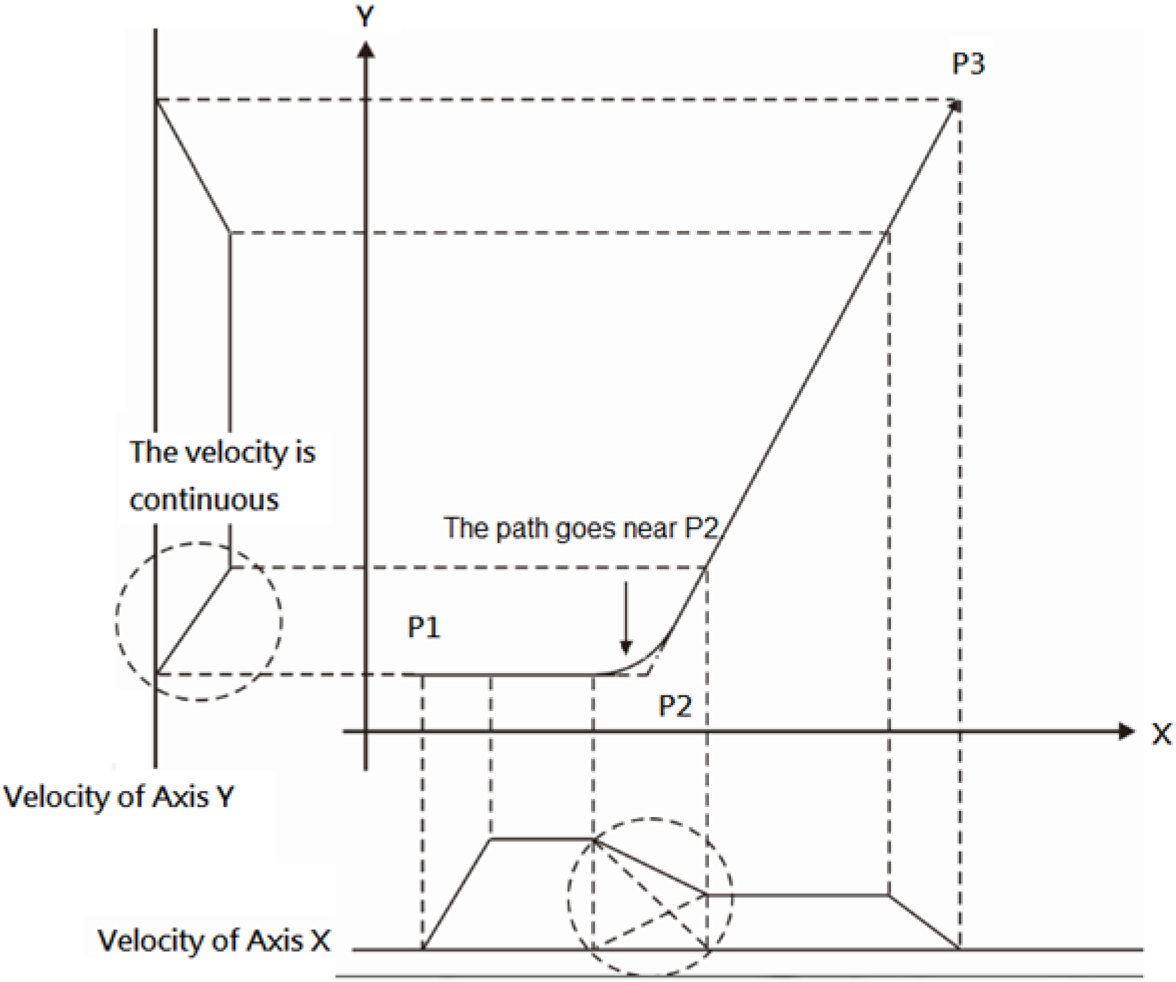
1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
2. TransitionMode can be used to reduce the noise and vibration that may occur during the changes of the interpolation motion.

TransitionMode: 0 (no effect)



TransitionMode: 1 (same motion as it is in deceleration of the current instruction)





TransitionMode: 2 (same motion as it is in deceleration of the axis parameters)

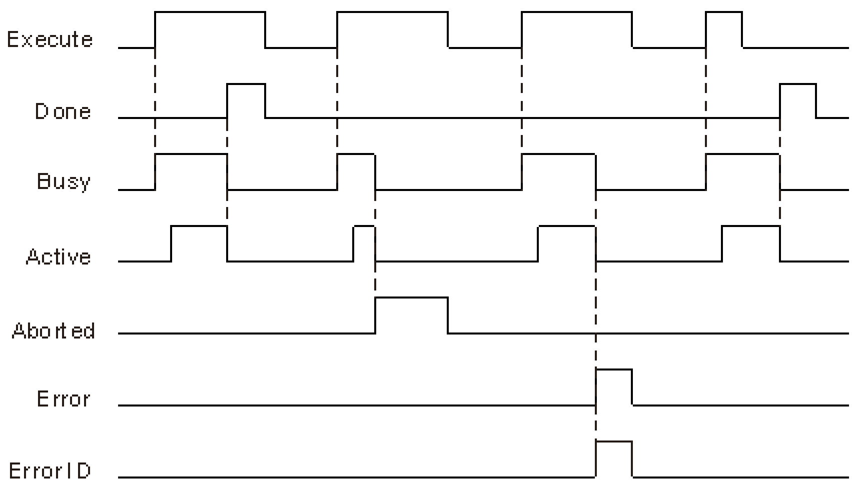
- ## Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the target position is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
Aborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the absolute positioning is completed.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True. • When *Aborted* shifts to True.
Active	• When the motion on the axis is started	• When *Done* shifts to True. • When *Error* shifts to True • When *Aborted* shifts to True.
Aborted	• When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. • When this instruction is aborted because of the execution of MC_Stop instruction. • When the instruction is aborted by DFB_GroupStop instruction • When the instruction is aborted by DFB_GroupImmediateStop instruction.	• When *Execute* shifts to False. • If *Execute* is False and *Aborted* shifts to True, it will be True for only one period and immediately shift to False.
Error(ErrorID)	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram



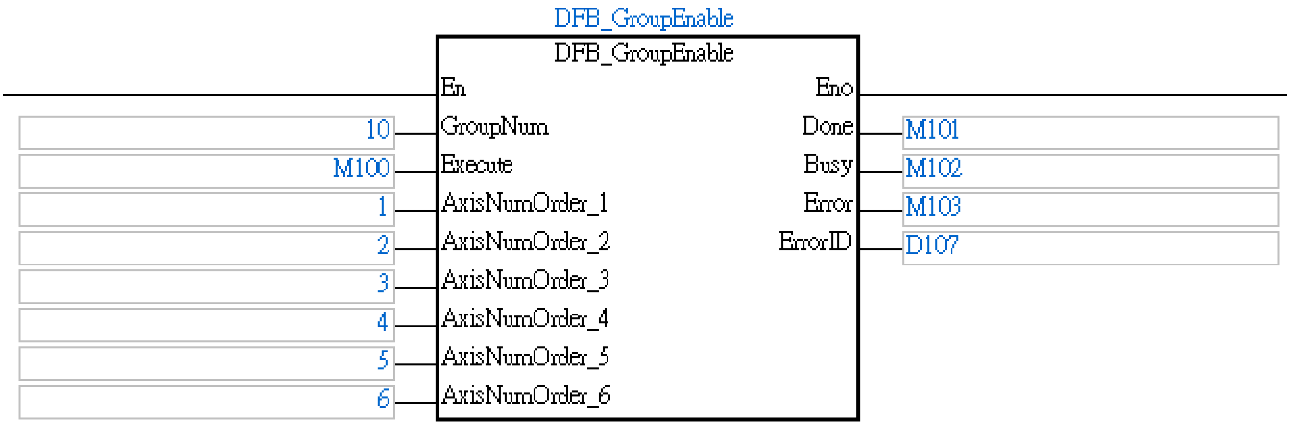
Troubleshooting

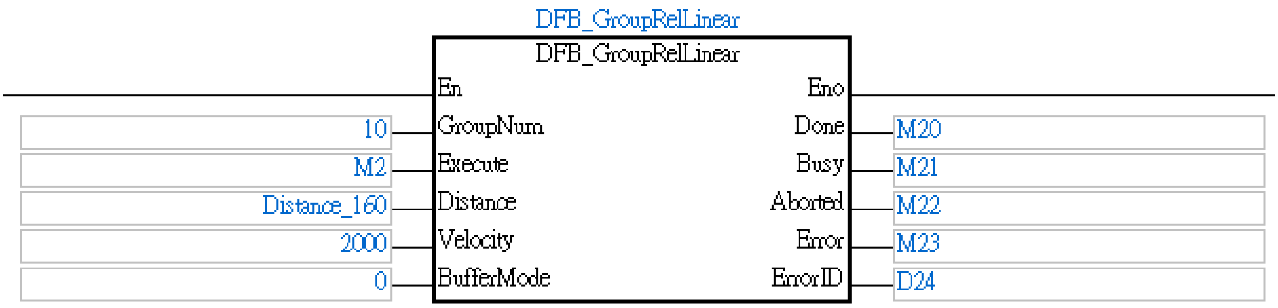
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

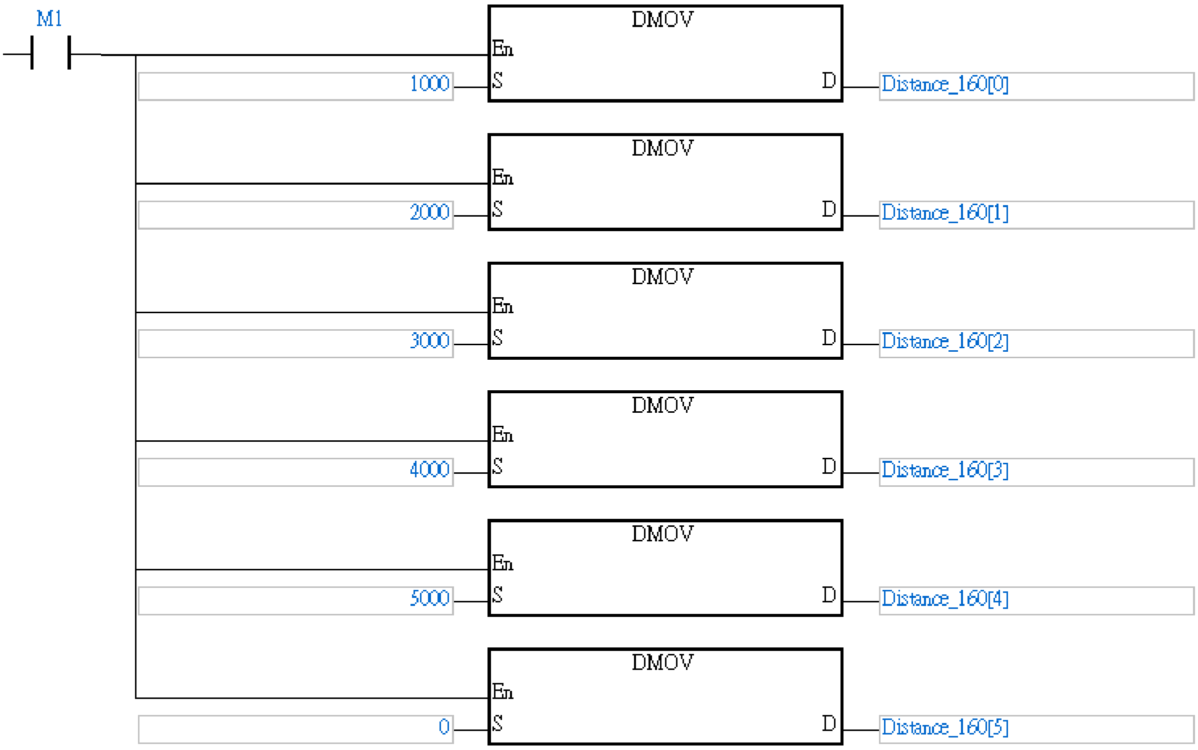
Programming Example

The example performs an linear interpolation from current position to the target position as below.

Axis group	Current position	Relative distance	Target position
Axis1	1000	1000	2000
Axis2	1000	2000	3000
Axis3	1000	3000	4000
Axis4	1000	4000	5000
Axis5	1000	5000	6000
Axis6	1000	0	1000







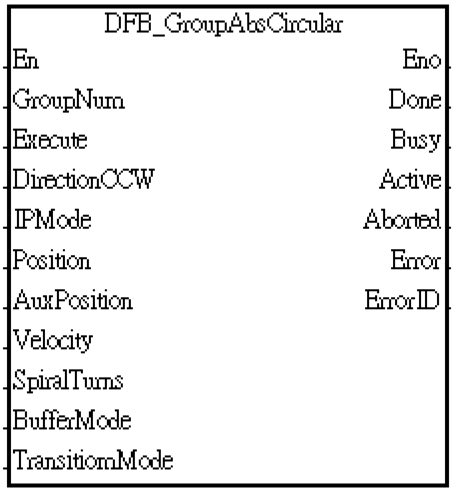
- When M2(*Execute*) changes to True, MC_GroupRelLinear drives the axes to move a relative distance to the target position with linear interpolation.
- When the specified relative distances (1000, 2000, 3000, 4000, 5000) are completed, M20(Done) changes to True, and M21 and M22 change to False.
- When M2(Execute) changes to False, M20(Done) changes to False.
- When the axes completed the specified relative distance(1000, 2000, 3000, 4000, 5000) and then M2 changes to True again, the instruction will be executed again to move another distance (1000, 2000, 3000, 4000, 5000) and reach the position of 3000, 5000,7000, 9000 and 11000.
- When the specified distance is completed, M20(Done) changes to True again.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_GroupAbsCircular

FB/FC	Description
FB	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move to the specified absolute target position.



Note: Arc/Helix interpolation requires at least 2 successive axis numbers to be specified starting from *AxisNumOrder_1* to *AxisNumOrder_2* of DFB_GroupEnable, otherwise an error will occur.

- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
DirectionCCW	Specifies the direction for the arc/circular motion.*1	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
IPMode	Specifies the way to define the center of the circle.*2	eDFB_IPMODE*3	0: radius_length 1: center_point	When *Execute* shifts to True and *Busy* is False.
Position	Specifies the target position for the 1st and 2nd axes, and the elevation of the helix for the 3rd axis. (Unit: user unit)*3	LREAL[3]	[_,_,_] Negative, positive value, 0 ([0,0,0])	When *Execute* shifts to True and *Busy* is False.
AuxPosition	Specifies the value of radius(R) or the coordinates(I, J) of the	LREAL [2]	[_,_] Negative, positive value, 0 ([0,0])	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
	center of the circle.*2			
Velocity	Interpolation speed (Unit: user unit/s) *3	LREAL	Positive value (0)	When *Execute* shifts to True and *Busy* is False.
SpiralTurns	Specifies the number of spirals in a helix.	DWORD	0~65535	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*3	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.
TransitionMode	Specifies if a round corner should be applied during the transition*2	WORD	0: no effect 1: round corner 2: round corner but ignoring the deceleration time (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Parameters of *DirectionCCW*:

State	Definition
False	Clockwise
True	Counterclockwise

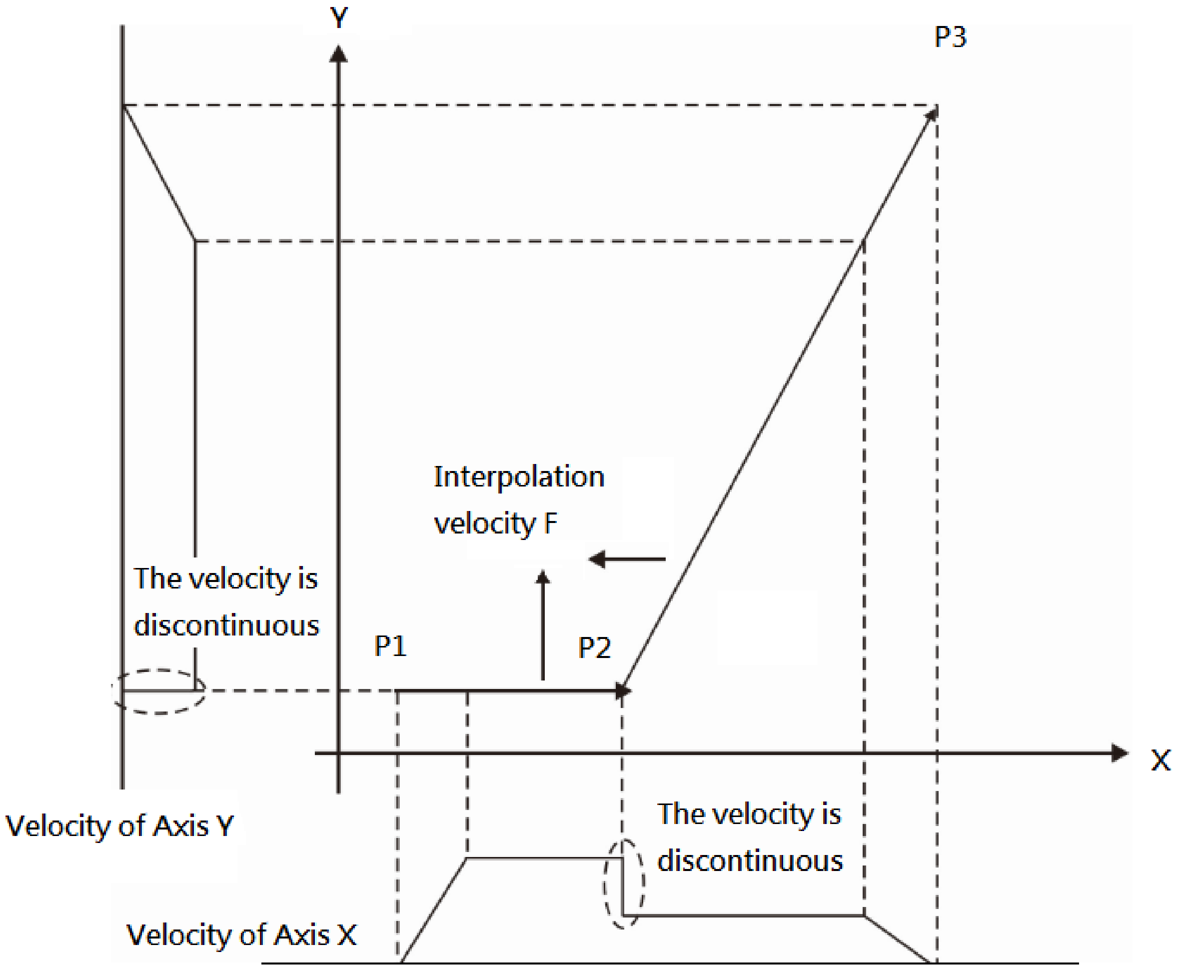
2. Parameters of *IPMode* and *AuxPosition*:

Input value of IPMode	Definition	Value 1 of AuxPosition	Value 2 of AuxPosition
0	Radius(R)	Length of radius(R)	N/A
1	Center of circle(I,J)	Value of I	Value of J

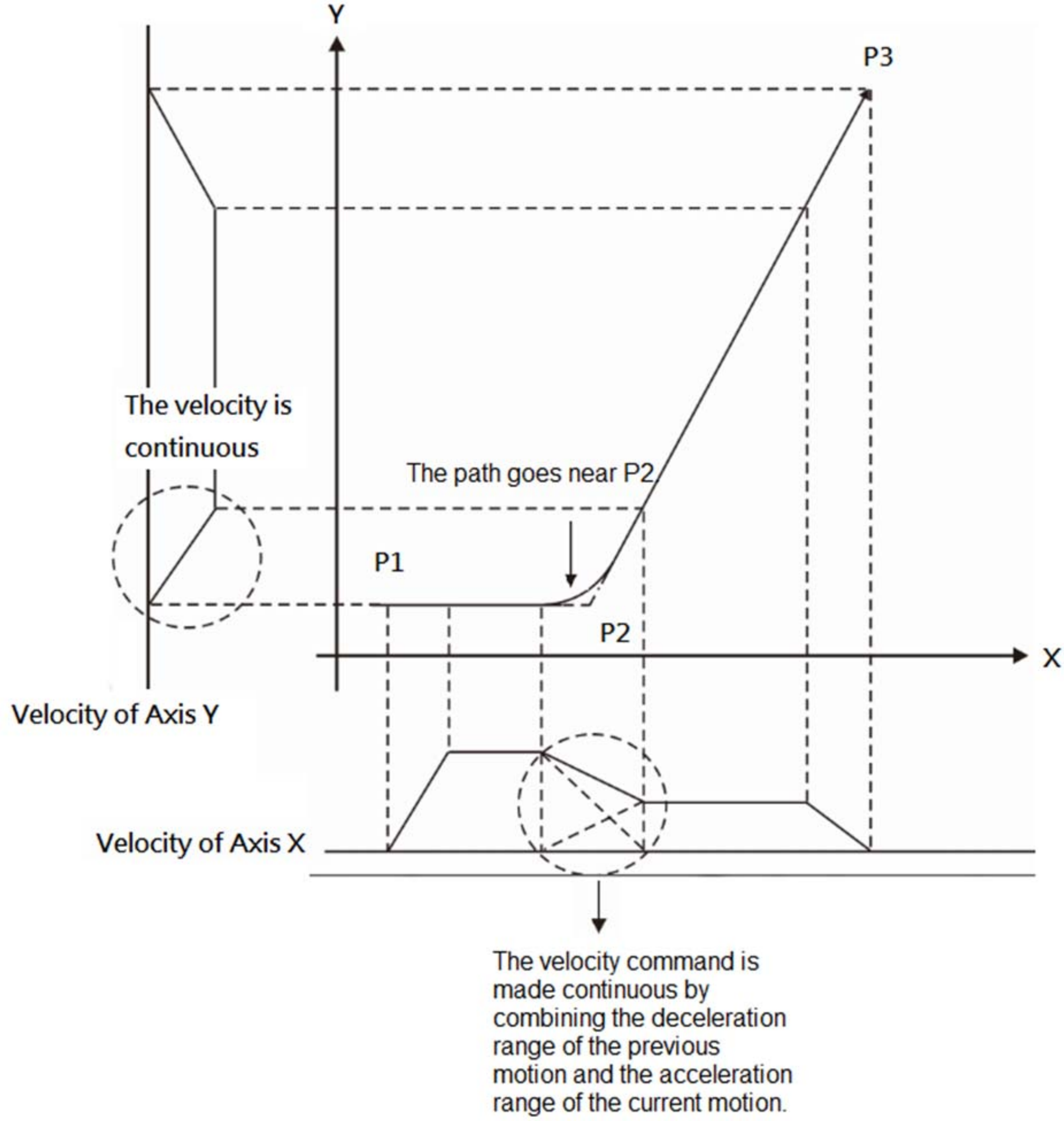
3. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
4. TransitionMode can be used to reduce the noise and vibration that may occur during the changes of the interpolation motion.

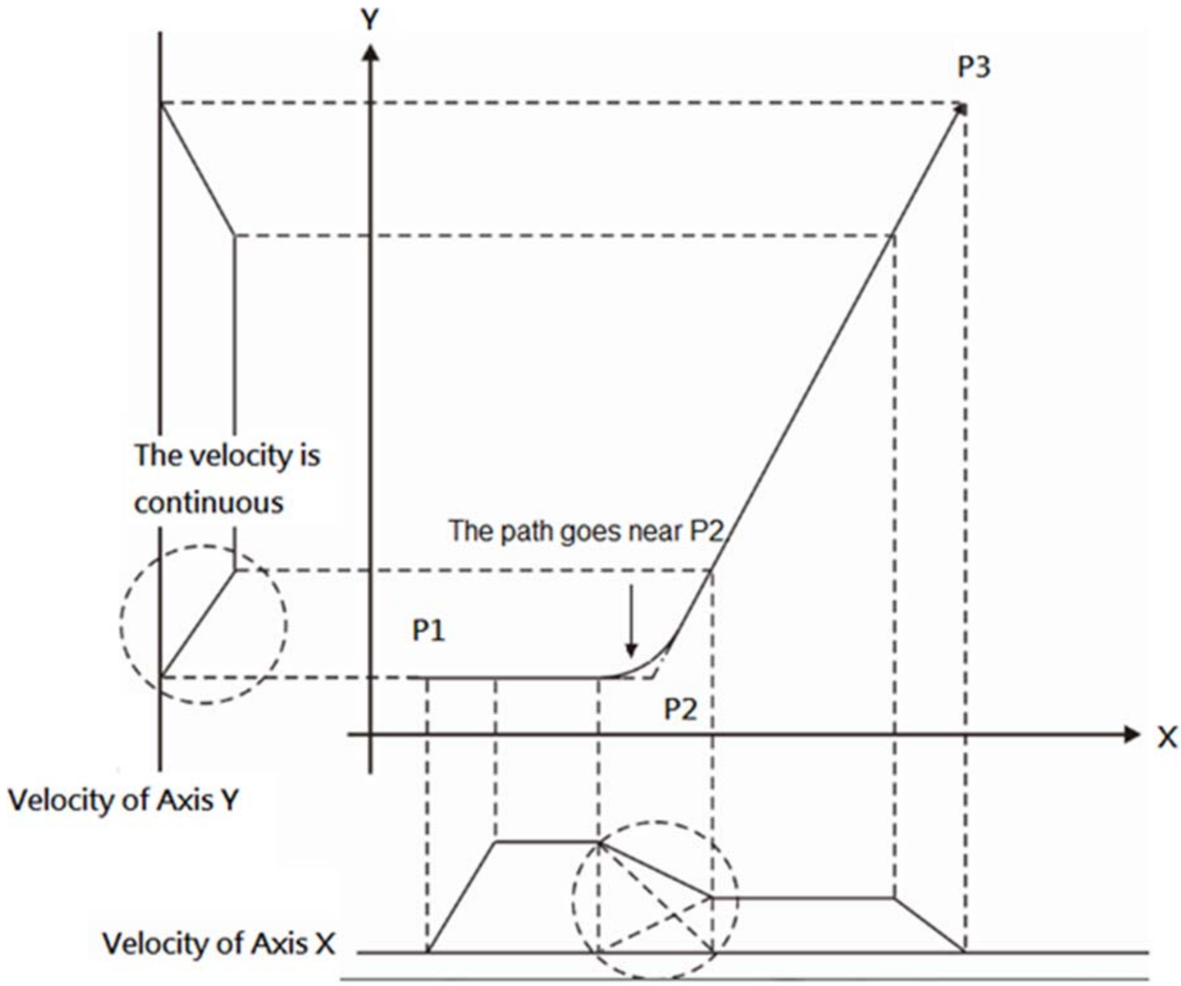
3

TransitionMode: 0 (no effect)



TransitionMode: 1 (same motion as it is in deceleration of the current instruction)





TransitionMode: 2 (same motion as it is in deceleration of the axis parameters)

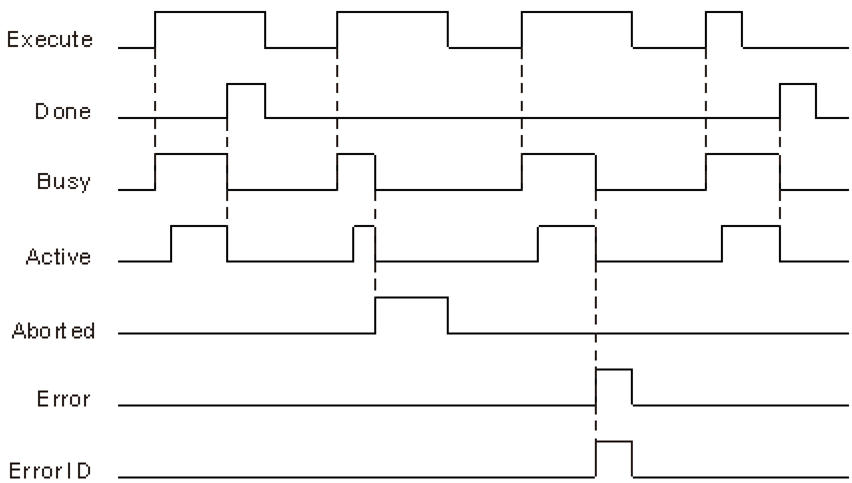
- ## Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the target position is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
Aborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the absolute positioning is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *Aborted* shifts to True.
Active	● When the motion on the axis is started	● When *Done* shifts to True. ● When *Error* shifts to True ● When *Aborted* shifts to True.
Aborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted because of the execution of MC_Stop instruction. ● When the instruction is aborted by DFB_GroupStop instruction ● When the instruction is aborted by DFB_GroupImmediateStop instruction.	● When *Execute* shifts to False. ● If *Execute* is False and *Aborted* shifts to True, it will be True for only one period and immediately shift to False.
Error(ErrorID)	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram

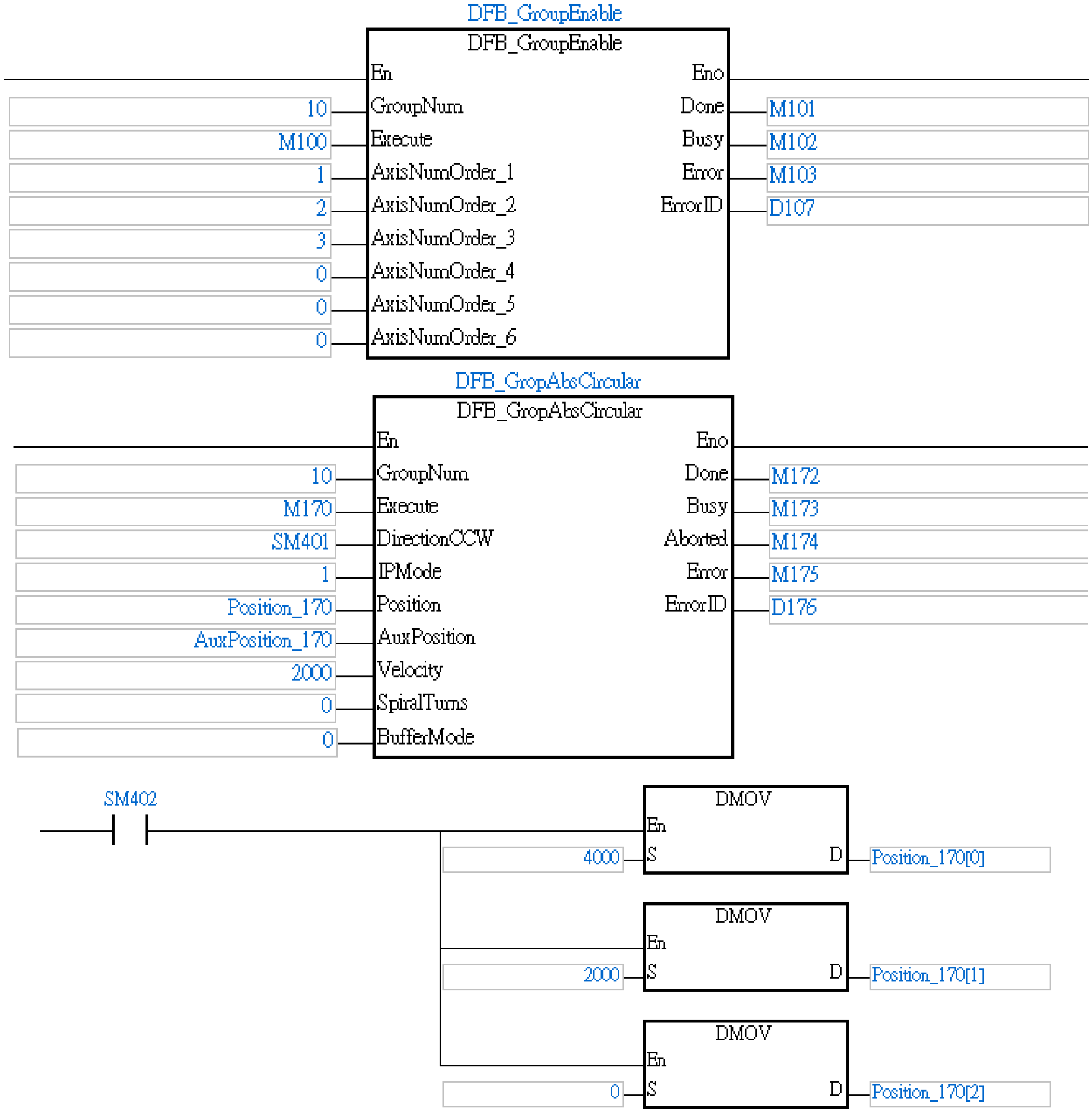


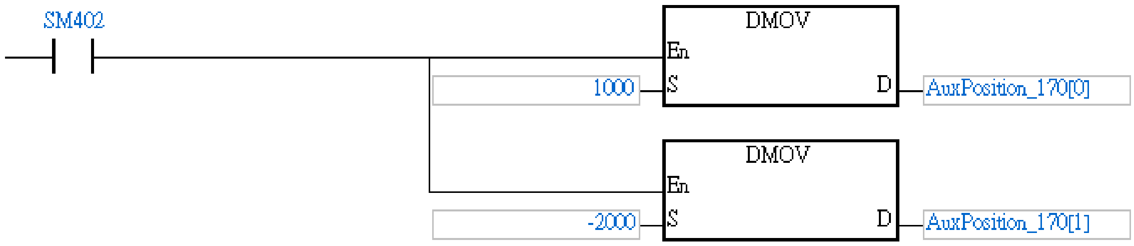
Troubleshooting

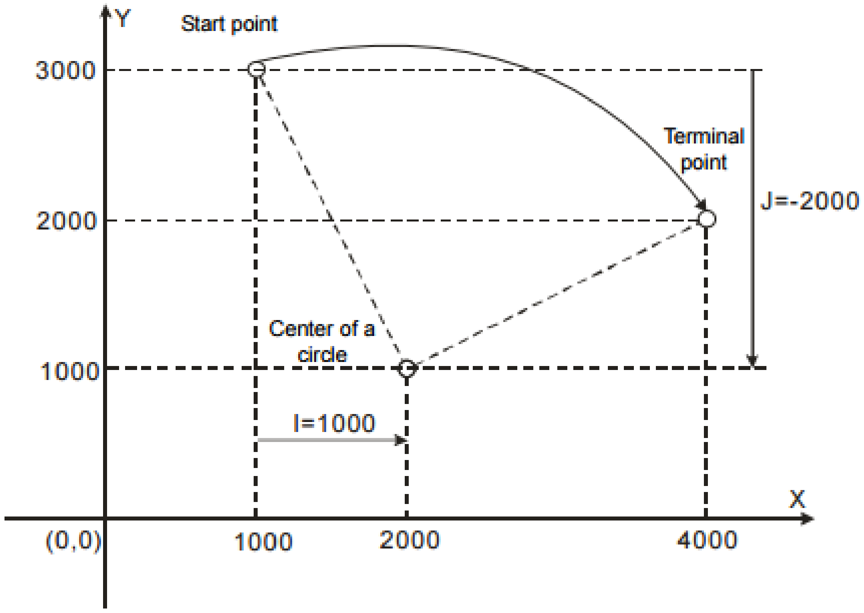
- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

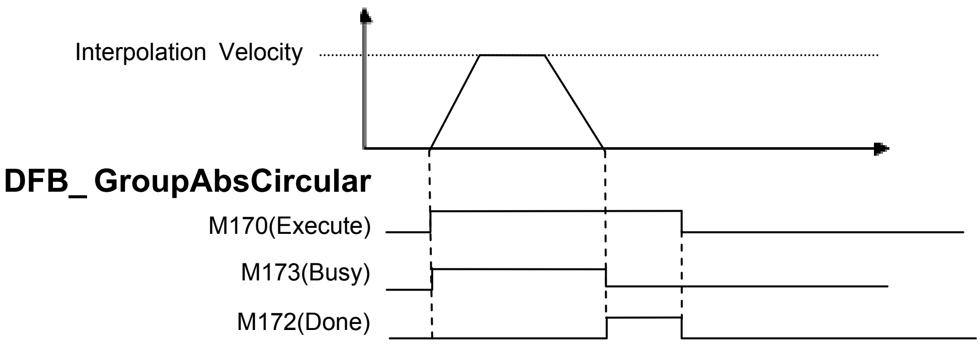
Programming Example

The example performs an arc interpolation from current position (1000, 3000) to the target position(4000, 2000) with clockwise interpolation.









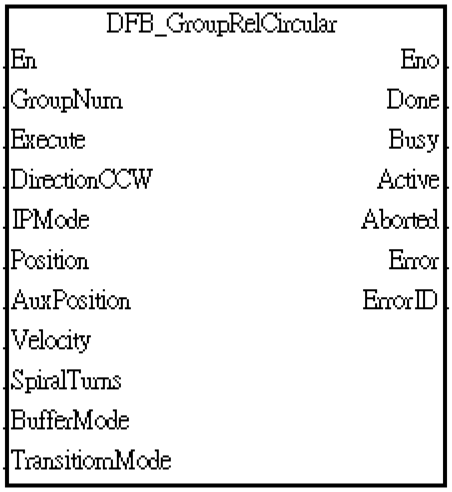
- When M170(*Execute*) changes to True, DFB_GroupAbsCircular drives the axes from the start point to the terminal point with clockwise arc interpolation.
- When the axes reaches the specified target position(4000,2000), M172(*Done*) changes to True and M173(*Busy*) changes to False.
- When M170(*Execute*) changes to False, M172(*Done*) changes to False.
- When the axes reach the target position, re-execution of the instruction will not move any axes because the absolute position is reached.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_GroupRelCircular

FB/FC	Description
FB	DFB_GroupAbsCircular controls the axis group to perform arc/circular or helix interpolation to move a specified relative distance.



Note: Arc/Helix interpolation requires at least 3 successive axis numbers to be specified starting from *AxisNumOrder_1* to *AxisNumOrder_3* of DFB_GroupEnable, otherwise an error will occur.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
DirectionCCW	Specifies the direction for the arc/circular motion.*1	BOOL	True/False (False)	When *Execute* shifts to True and *Busy* is False.
IPMode	Specifies the way to define the center of the circle.*2	eDFB_IPMODE*3	0: radius_length 1: center_point	When *Execute* shifts to True and *Busy* is False.
Distance	Specifies the distance that each axis moves. (Unit: user unit)	LREAL[6]	[_,_,_] Negative, positive value, 0 ([0,0,0])	When *Execute* shifts to True and *Busy* is False.
AuxPosition	Specifies the value of radius(R) or the coordinates(I, J) of the center of the circle.*2	LREAL [2]	[_,_] Negative, positive value, 0 ([0,0])	When *Execute* shifts to True and *Busy* is False.

Name	Function	Data type	Setting value (Default value)	Timing for updating
Velocity	Interpolation speed (Unit: user unit/s) *3	LREAL	Positive value (0)	When *Execute* shifts to True and *Busy* is False.
SpiralTurns	Specifies the number of spirals in a helix.	DWORD	0~65535	When *Execute* shifts to True and *Busy* is False.
BufferMode	Specifies the buffering behavior of the instruction.	eMC_BUFFER_MODE*3	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh (0)	When *Execute* shifts to True and *Busy* is False.
TransitionMode	Specifies if a round corner should be applied during the transition*2	WORD	0: no effect 1: round corner 2: round corner but ignoring the deceleration time (0)	When *Execute* shifts to True and *Busy* is False.

***Note:**

1. Parameters of *DirectionCCW*:

State	Definition
False	Clockwise
True	Counterclockwise

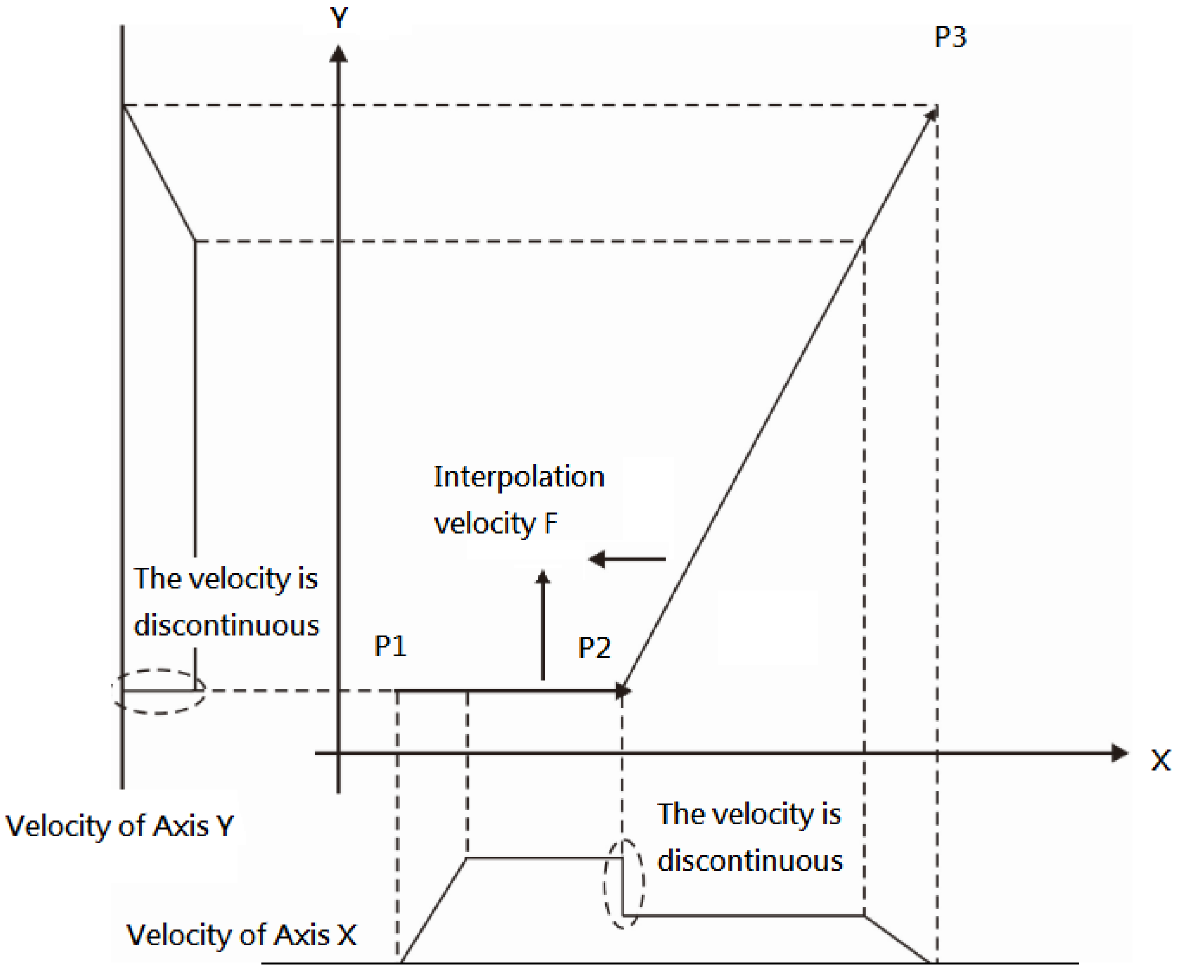
2. Parameters of *IPMode* and *AuxPosition*:

Input value of IPMode	Definition	Value 1 of AuxPosition	Value 2 of AuxPosition
0	Radius(R)	Length of radius(R)	N/A
1	Center of circle(I,J)	Value of I	Value of J

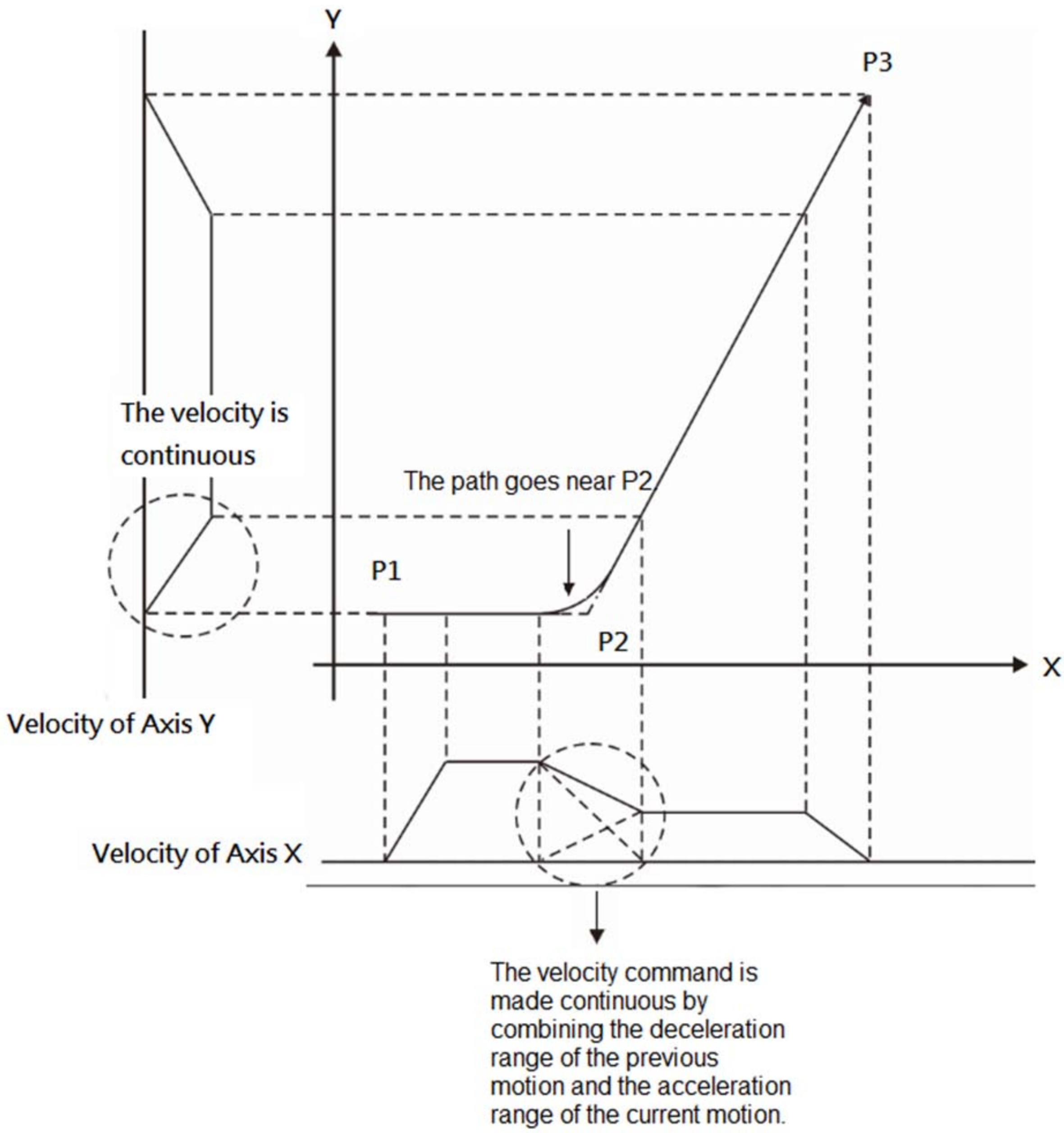
3. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
4. TransitionMode can be used to reduce the noise and vibration that may occur during the changes of the interpolation motion.

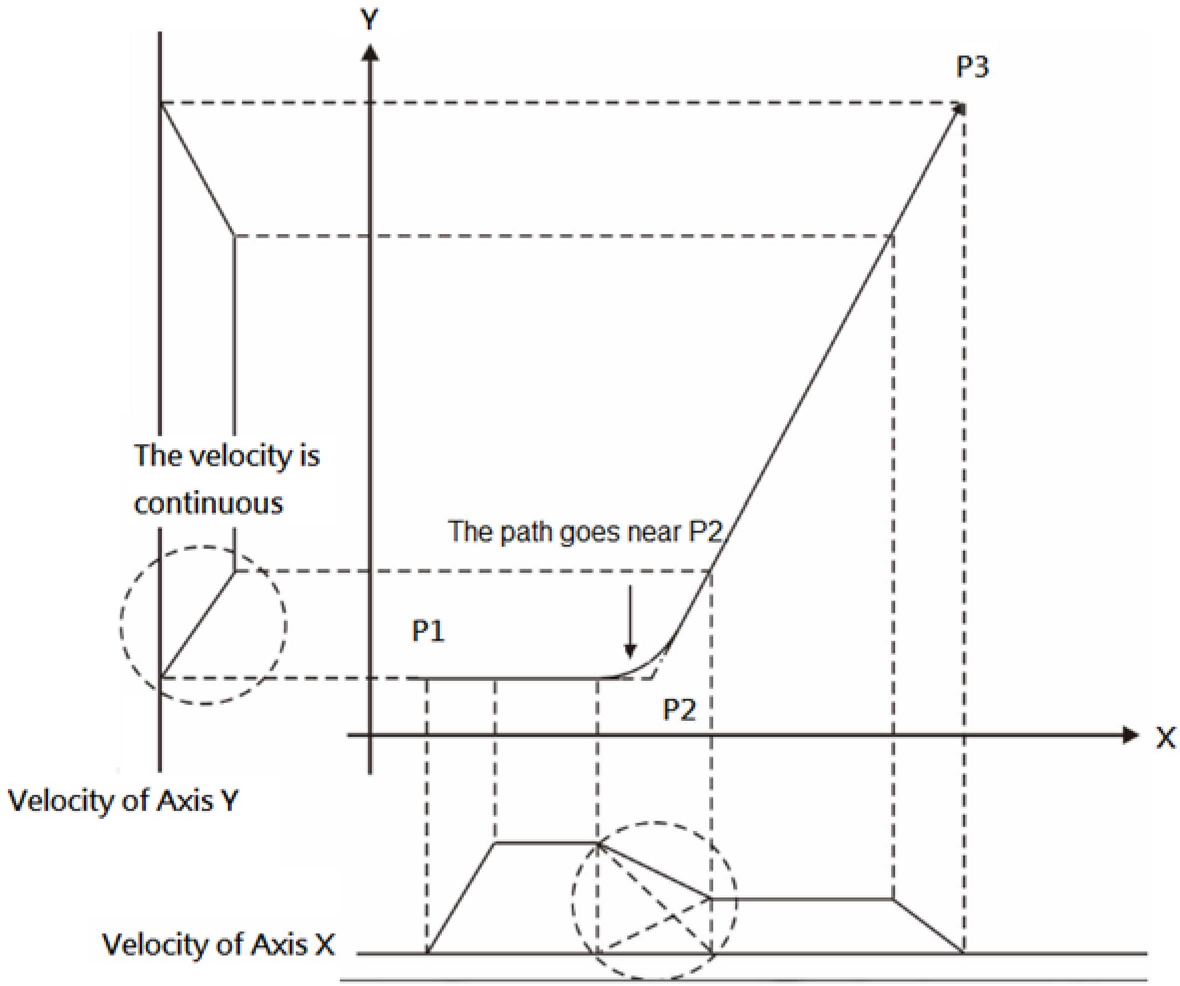
3

TransitionMode: 0 (no effect)



TransitionMode: 1 (same motion as it is in deceleration of the current instruction)





TransitionMode: 2 (same motion as it is in deceleration of the axis parameters)

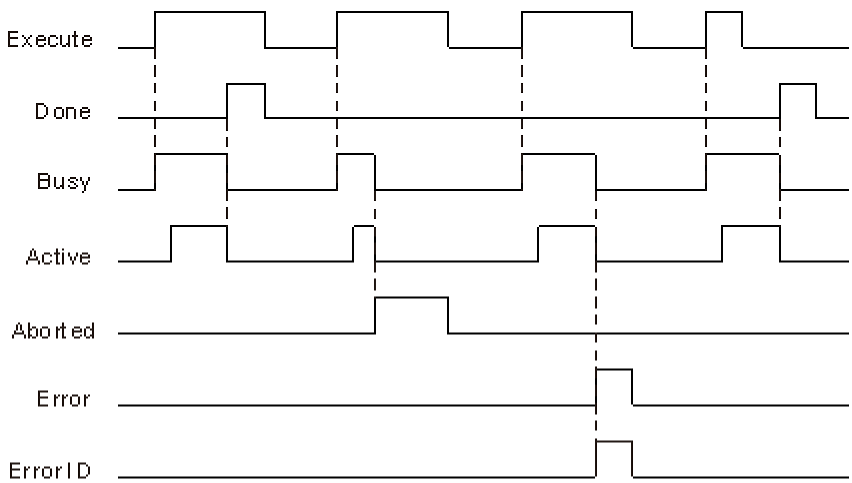
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the target position is reached.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Active	True when the axis is being controlled.	BOOL	True/False (False)
Aborted	True when the instruction is aborted.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code when the error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the absolute positioning is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True. ● When *Aborted* shifts to True.
Active	● When the motion on the axis is started	● When *Done* shifts to True. ● When *Error* shifts to True ● When *Aborted* shifts to True.
Aborted	● When this instruction is aborted by another instruction with the Buffer Mode set to mcAborting. ● When this instruction is aborted because of the execution of MC_Stop instruction. ● When the instruction is aborted by DFB_GroupStop instruction ● When the instruction is aborted by DFB_GroupImmediateStop instruction.	● When *Execute* shifts to False. ● If *Execute* is False and *Aborted* shifts to True, it will be True for only one period and immediately shift to False.
Error(ErrorID)	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ Timing Diagram

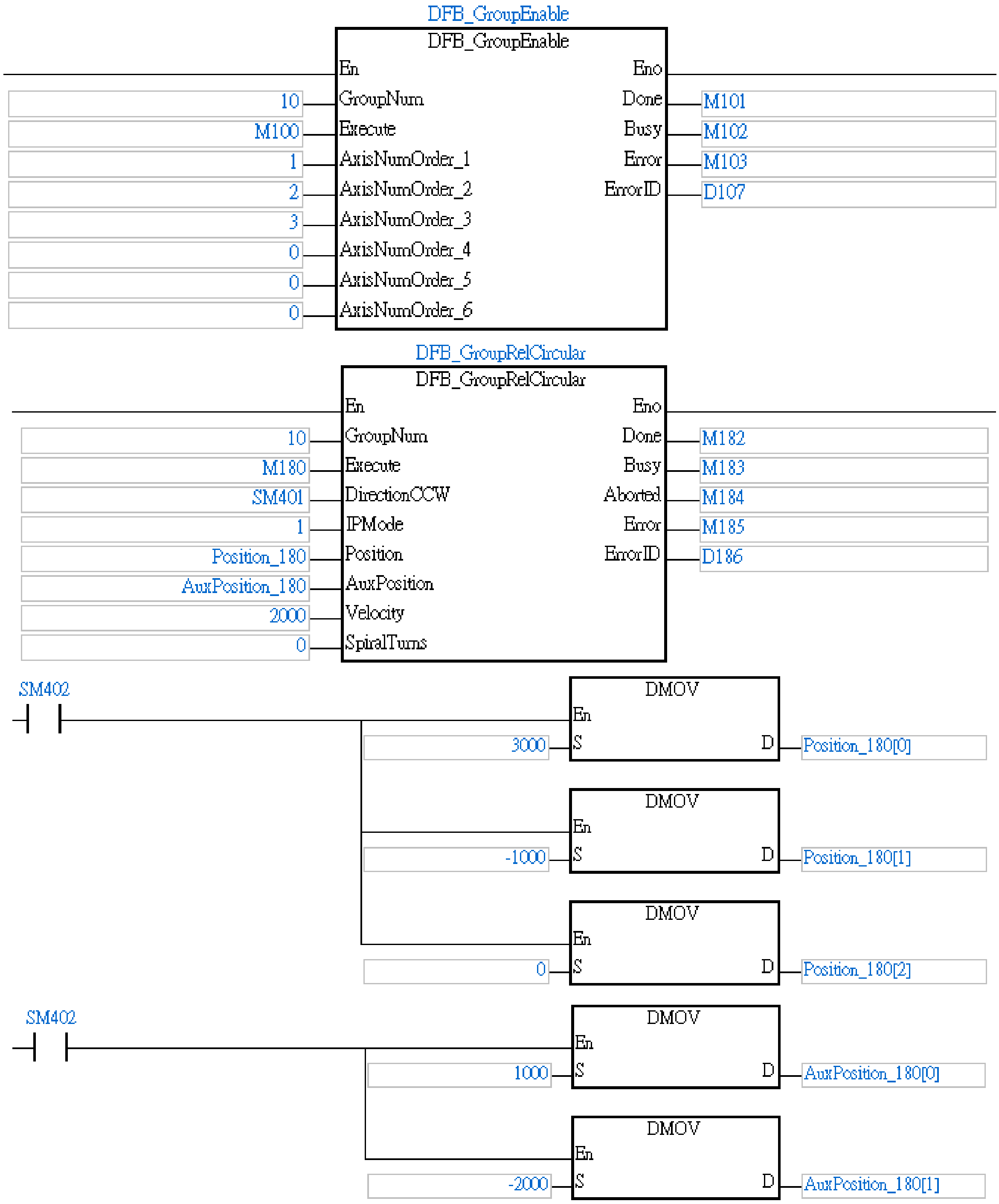


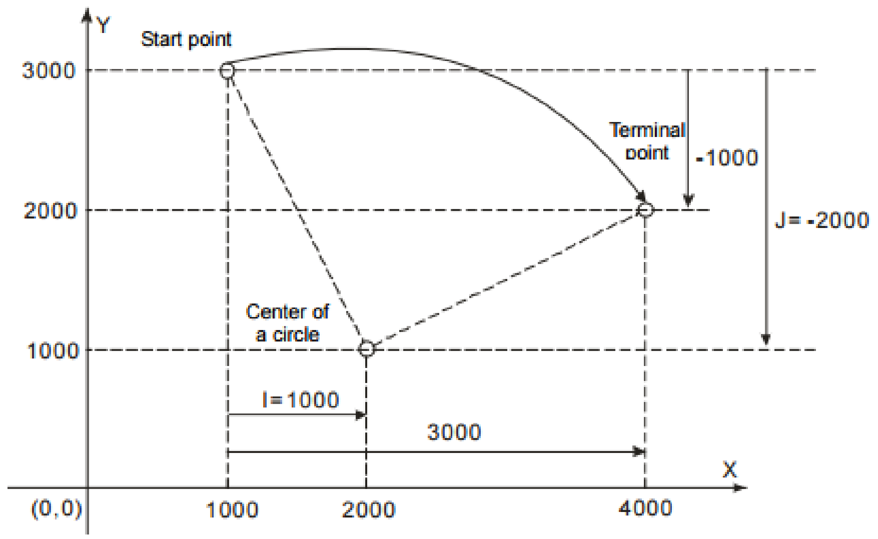
Troubleshooting

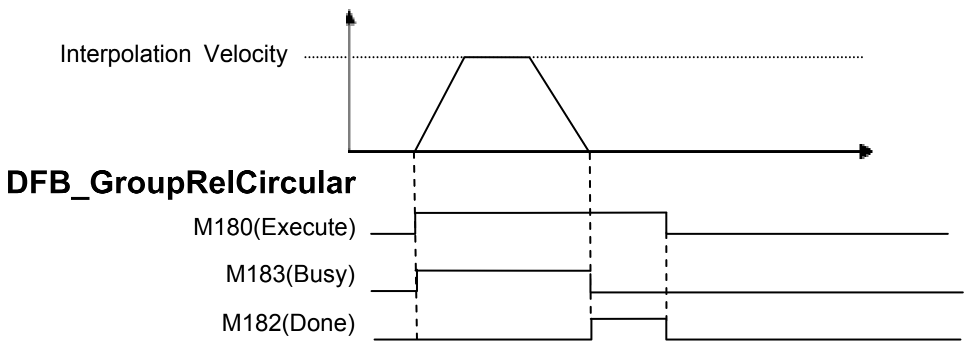
- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

Programming Example

The example performs an arc interpolation from current position (1000, 3000) to the target position(4000, 2000) with clockwise interpolation.







- When M180(*Execute*) changes to True, DFB_GroupRelCircular drives the axes from the start point (1000, 300) to the terminal point (4000, 2000) with clockwise interpolation.
- When the axes reaches the specified target position(4000,2000), M182(*Done*) changes to True, and M183(*Busy*) changes to False.
- When M180(*Execute*) changes to False, M182(*Done*) changes to False.
- When the axes reach the target position, re-execution of the instruction will move another relative distance from current position according to the input parameters.
- When M180(*Execute*) changes to True, the axes will perform circular interpolation from the current point (4000, 2000).

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_GroupStop

FB/FC	Description
FB	DFB_GroupStop decelerates the group axes to stop or pause.

- The axis group specified by *GroupNum* will be decelerated to stop or pause.
- The group motion can be G-code motion, linear interpolation or circular interpolation.
- The axis states will enter Stopping after the execution of DFB_GroupStop.
- The axis state Stopping will go till the velocity becomes zero or when *Execute* shifts to False. When the velocity becomes zeron, *Done* shifts to Ture. And when *Done* is True and *Execute* shifts to False, the axis state will enter Standby.

Note:

1. For more information about the axis states after DFB_GroupStop is executed, refer to **7.4 State Transitions** of ***AH Motion Controller – Operation Manual***.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-
StopMode	The axis stops.	WORD	0: Stop 1: Pause (0)	When *Execute* shifts to True and *Busy* is False. When *Execute* shifts to Fault and *Done* is True.

● Outputs

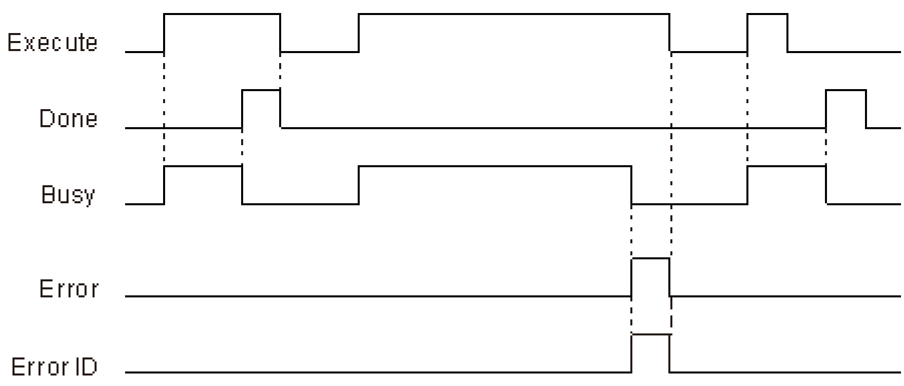
Name	Function	Data type	Output range (Default value)
Done	True when the group motion is stopped.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code	DWORD	16#0~16#FFFFFFFF (0)

	descriptions.		

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the group motion is stopped.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True.
Error(ErrorID)	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● Troubleshooting

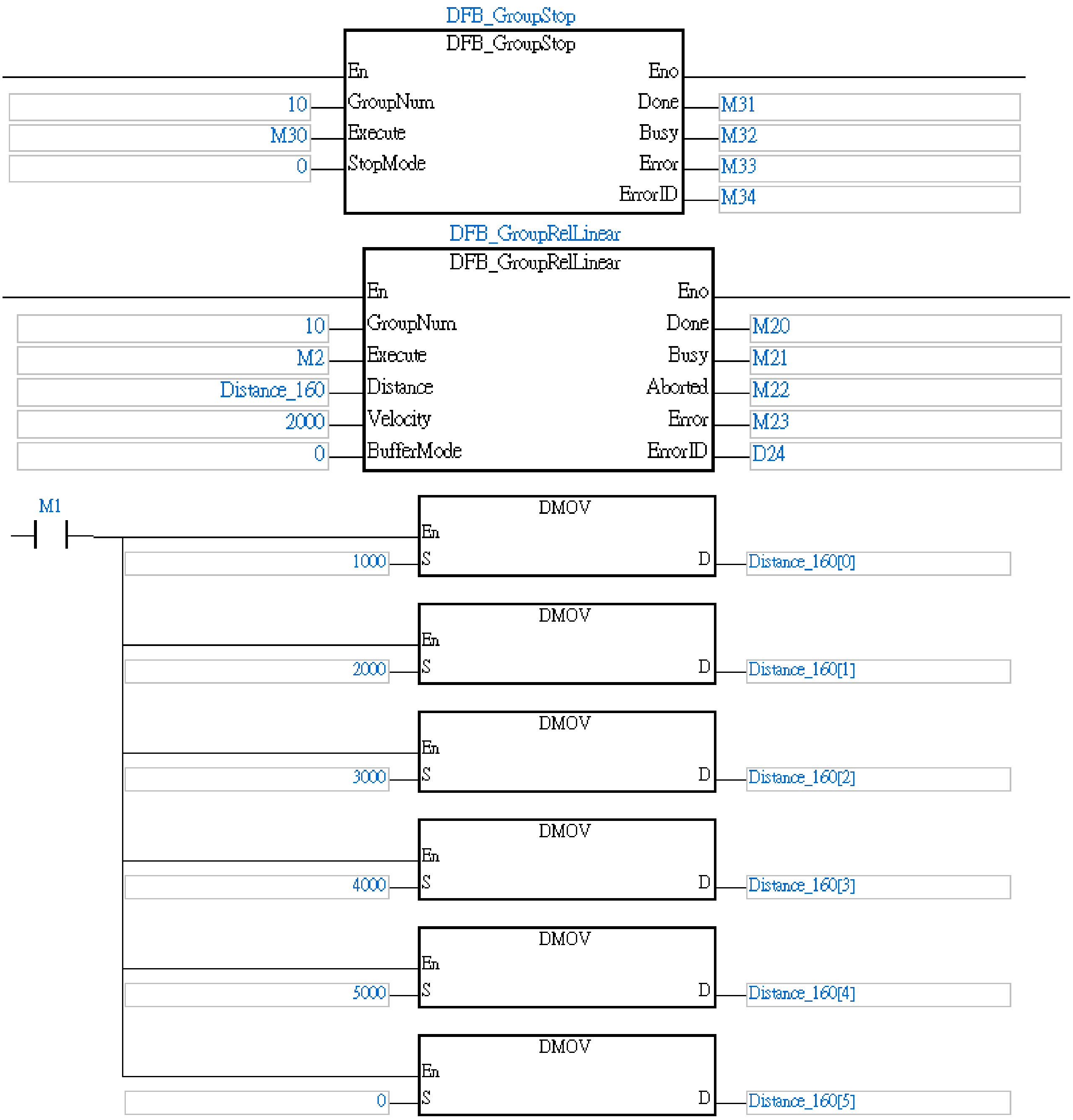
- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

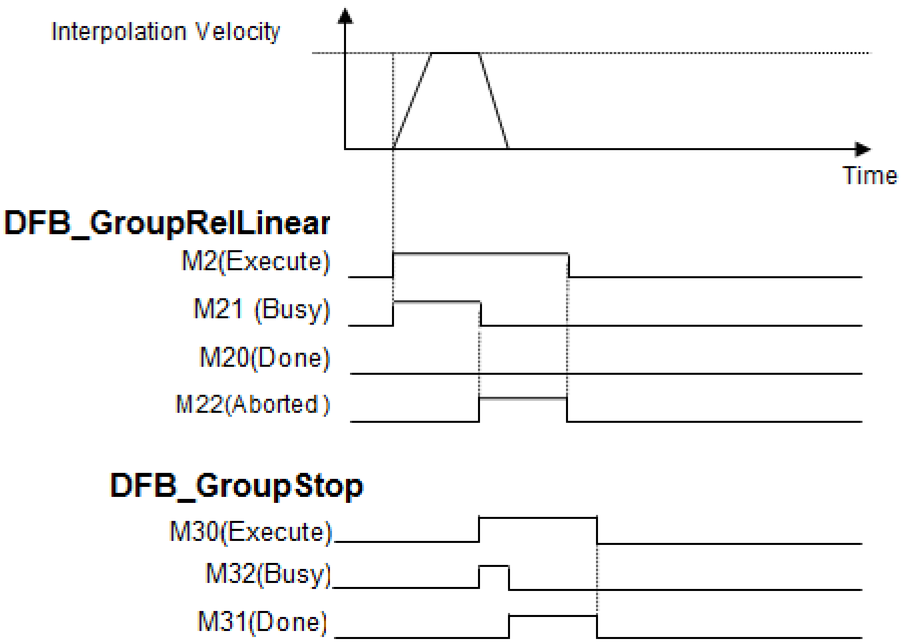
The example uses DFB_GroupStop to decelerate the ongoing group motion to stop.

Example 1: DFB_GroupStop (0)
This examples shows 6 axes in linear interpolation to move from the current position to the relative target position (relative distance) and then use DFB_GroupStop to have the axes in motion to decelerate to stop.

Axis group	Current position	Relative distance	Target position
Axis1	1000	1000	2000
Axis2	1000	2000	3000
Axis3	1000	3000	4000
Axis4	1000	4000	5000
Axis5	1000	5000	6000



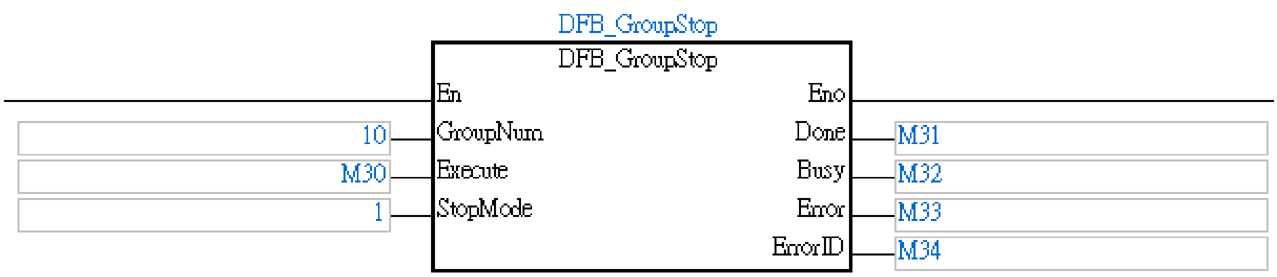
1. Set M2 (Execute) to True, DFB_GroupRelLinear start to do linear interpolation to move from the current position to the relative target position (relative distance).
2. Set M30 to True (mode: 0) before the on-going DFB_GroupRelLinear is completed.
3. M22 (*Aborted*) changes to True, indicating the DFB_GroupRelLinear is aborted.
4. M31 (*Done*) changes to True, when DFB_GroupRelLinear decelerates to stop and the axis state is in Stopping.
5. When M30 (*Execute*) changes to False, the axis state is in Standby.

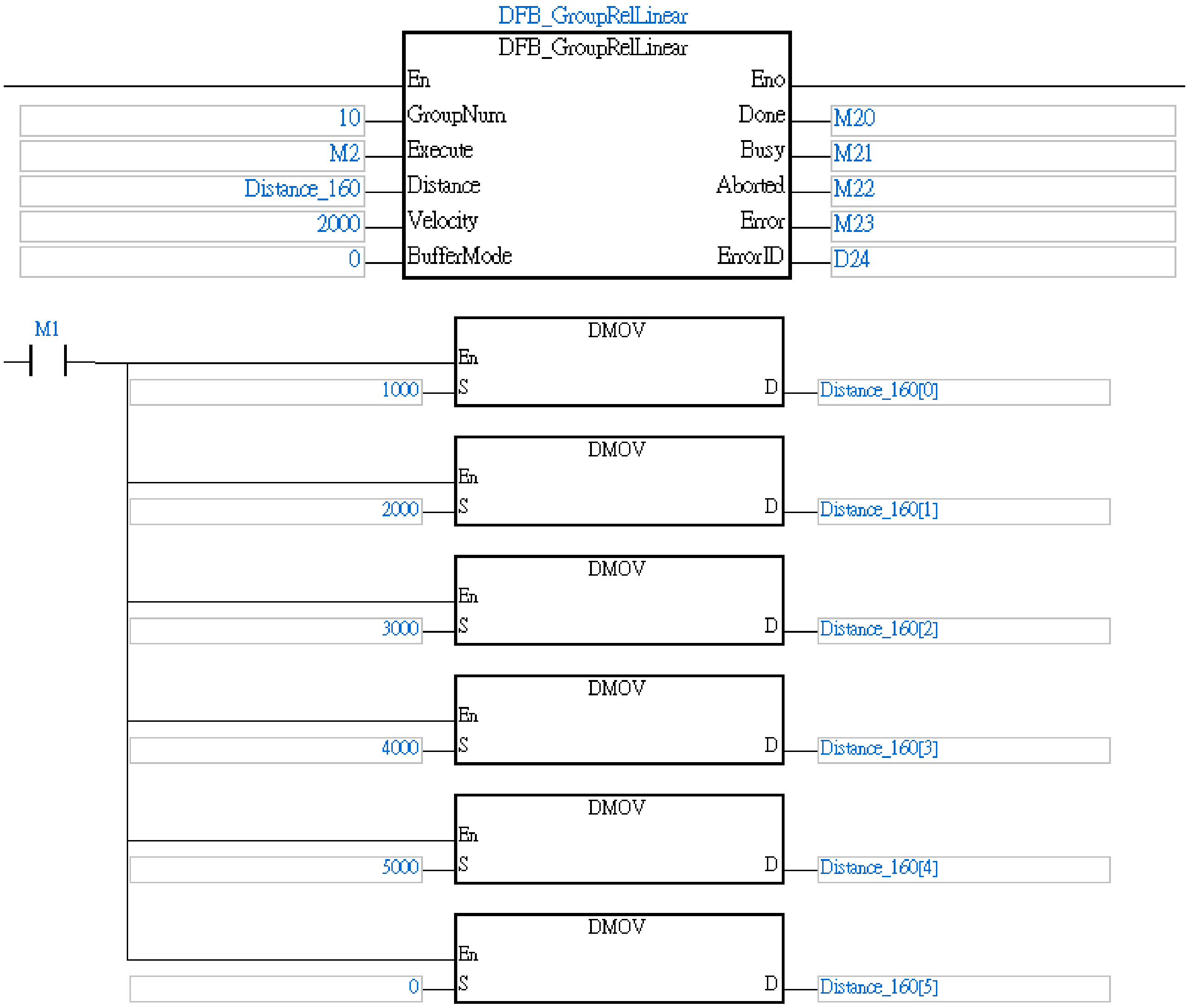


The example uses DFB_GroupStop to decelerate the ongoing group motion to pause.

Example 2: DFB_GroupStop (0)
This examples shows 6 axes in linear interpolation to move from the current position to the relative target position (relative distance) and then use DFB_GroupStop to have the axes in motion to decelerate to pause.

Axis group	Current position	Relative distance	Target position
Axis1	1000	1000	2000
Axis2	1000	2000	3000
Axis3	1000	3000	4000
Axis4	1000	4000	5000
Axis5	1000	5000	6000





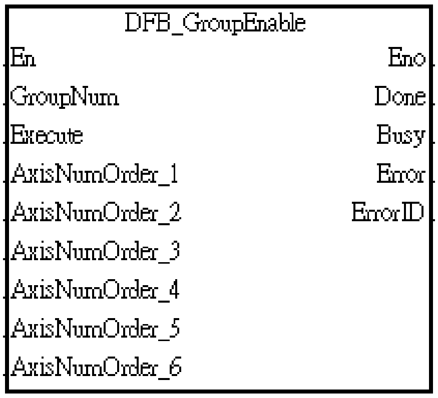
1. Set M2 (Execute) to True, DFB_GroupRelLinear start to do linear interpolation to move from the current position to the relative target position (relative distance).
2. Set M30 to True (mode: 0) before the on-going DFB_GroupRelLinear is completed.
3. M22 (Aborted) changes to True, indicating the DFB_GroupRelLinear is aborted.
4. M31 (Done) changes to True, when DFB_GroupRelLinear decelerates to stop and the axis state is in Stopping.
5. When M30 (Execute) changes to False, the axis state is in Standby.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_GroupEnable

FB/FC	Description
FB	DFB_GroupEnable enables a group of axes for group motion.



- When the specified axis group is enabled, it will change its state from “Disabled” to “Standby”, and the designated axes in the group will change their state from “Standstill” to “Coordinated”.
- When the instruction is executed, it will check whether the designated axes are in “Standstill”, i.e. ready to be used for group motion, otherwise the specified axis group will enter “Errorstop”. In this case, you will have to use DFB_GroupReset to reset the error situation.

Note: refer to **7.4 State Transitions** of ***AH Motion Controller* – *Operation Manual*** for more information about state transitions.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Writes in the parameters when *Execute* changes to True.	BOOL	True/False (False)	-
AxisNumOrder_1	Designates the axis number for the 1st axis (X-axis)	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.
AxisNumOrder_2	Designates the axis number for the 2nd axis (Y-axis)	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.
AxisNumOrder_3	Designates the axis number for the 3rd axis (Z-axis)	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.
AxisNumOrder_4	Designates the axis number for the 4th axis (A-axis)	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.
AxisNumOrder_5	Designates the axis number for the 5th axis (B-axis)	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.

AxisNumOrder_6	Designates the axis number for the 6th axis (C-axis)	WORD	1~32 (0)	When *Execute* shifts to True and *Busy* is False.

Outputs

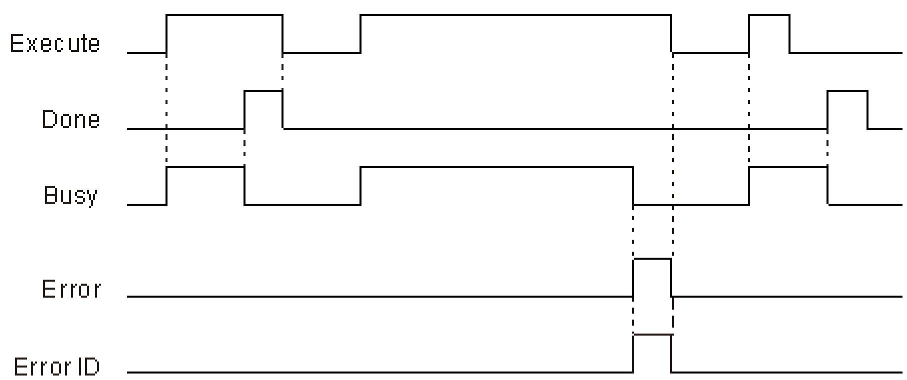
Name	Function	Data type	Output range (Default value)
Done	True when the parameter setting is completed.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

3

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the axis group is enabled	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error(ErrorID)	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

Timing Diagram



Function

- *AxisNumOrder_1~ AxisNumOrder_6* indicate the 6 dimensions which are also the 6 axes of the coordinates: X, Y, Z, A, B, and C. You need to specify “0” to the axis number input which is not to be used.
- The first axis, *AxisNumOrder_1*, should always be given an axis number.

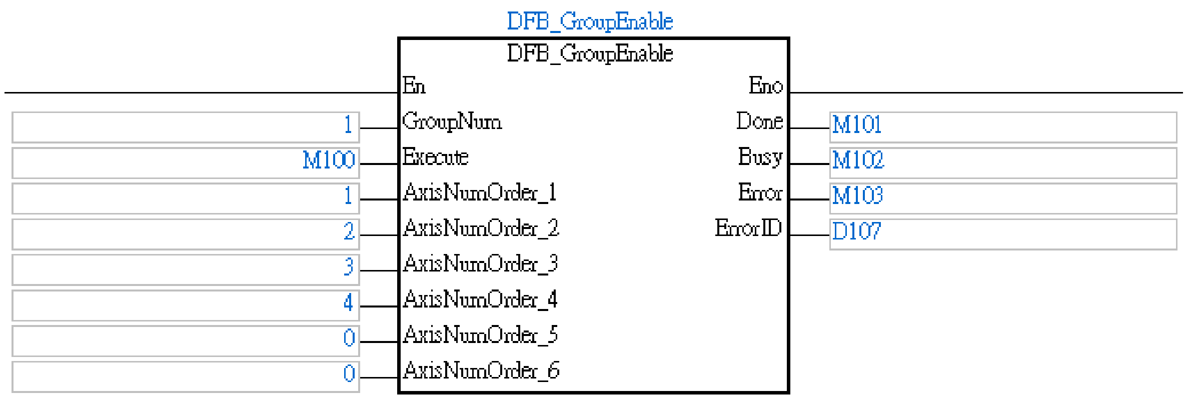
- Linear interpolation requires at least 2 axes, which can be continuous or discontinuous, to be specified between *AxisNumOrder_1* and *AxisNumOrder_6*.
- Arc interpolation and G-code require at least 3 successive axis numbers to be specified starting from *AxisNumOrder_1* to *AxisNumOrder_3*. For G-code applications, you can refer to DFB_GroupGcodeRun instruction.
- The setting range for axis number is between 1 and 32. The specified axis number should not be repeated among the inputs of *AxisNumOrder_1* to *AxisNumOrder_6*.

Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Programming Example

You can use DFB_GroupEnable to set up the number of axes in a group based on the limit and requirements of the coordinated axes for the actual applications.



1. If you want to use axis 1~ 4 to execute group motion for absolute positioning, firstly you need to set up the axis group by DFB_GroupEnable.
2. Set M100 to True to specify axis 1~4 to the axis group number 1.
3. When M101 changes to True, the enabling of group axes is completed, and the axis group will change its state from “Disabled” to “Standby”. Also, the designated axes in the group will change their state from “Standstill” to “Coordinated”.
4. After the axis group is enabled, you can use group number 1 to perform the absolute positioning with interpolation.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_GroupDisable

FB/FC	Description
FB	DFB_GroupDisable disables the axis group with the specidied group number.

- The axes in the disabled group will be available for single axis motion.
- When the specified axis group is disabled, it will change its state from “Standby” to “Disabled”, and the designated axes in the group will change their state from “Coordinated” to “Standstill”.

Note: refer to **7.4 State Transitions** of ***AH Motion Controller – Operation Manual*** for more information about state transitions.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Writes in the parameters when *Execute* changes to True.	BOOL	True/False (False)	-

● Outputs

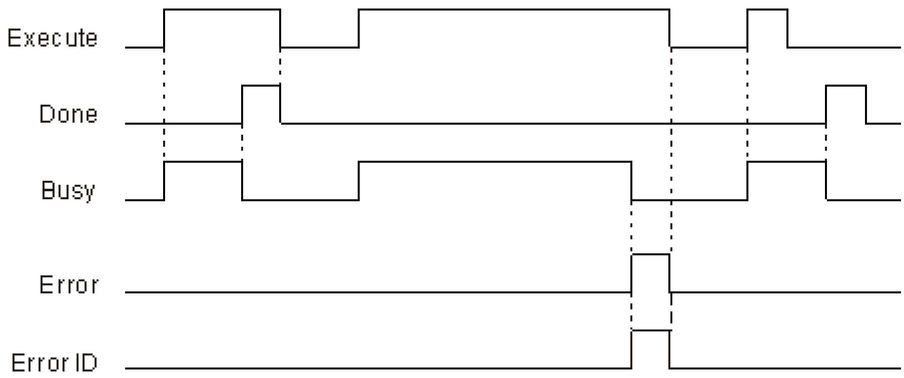
Name	Function	Data type	Output range (Default value)
Done	True when the axis group is disabled.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	● When the parameter setting is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.

Name	Timing for shifting to True	Timing for shifting to False
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error(ErrorID)	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**

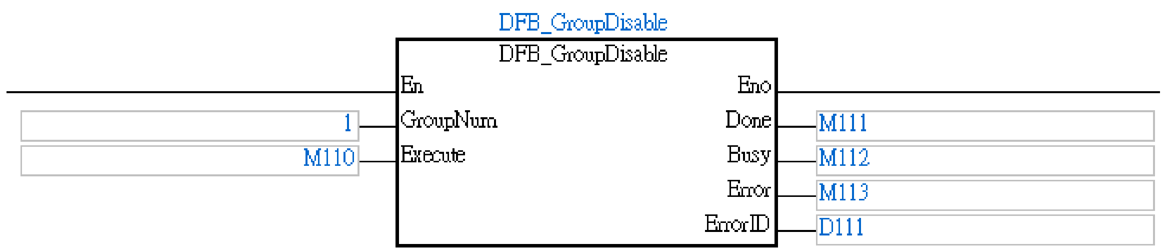


Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Programming Example

The example disables the axis group which is enabled by DFB_GroupEnable.



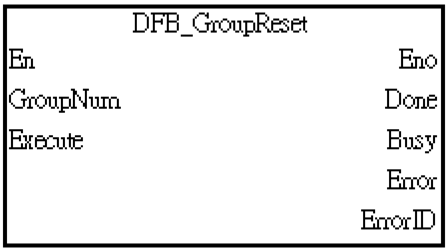
1. Suppose that group number 1 is enabled by DFB_GroupEnable. DFB_GroupDisable can be used to disable group number 1, so as to perform single axis motion on the axes used in this axis group.
2. Set M110 to True the disable the axis group (1) which is designated by *GroupNum*.
3. When M111 changes to True, the disabling of group number 1 is completed.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_GroupReset

FB/FC	Description
FB	DFB_GroupReset resets the axis group which is in the state of “Errorstop”.



- The state of the axis group will enter “Standby” after DFB_GroupReset is executed.
- The axis group will be available for group motion in “Standby” state.

Note: For more information about the state of axis as well as group and the timing to use DFB_GroupReset, refer to **7.4 State Transitions** of ***AH Motion Controller – Operation Manual***.

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Execute* shifts to True and *Busy* is False.
Execute	Executes the instruction when *Execute* changes to True.	BOOL	True/False (False)	-

Outputs

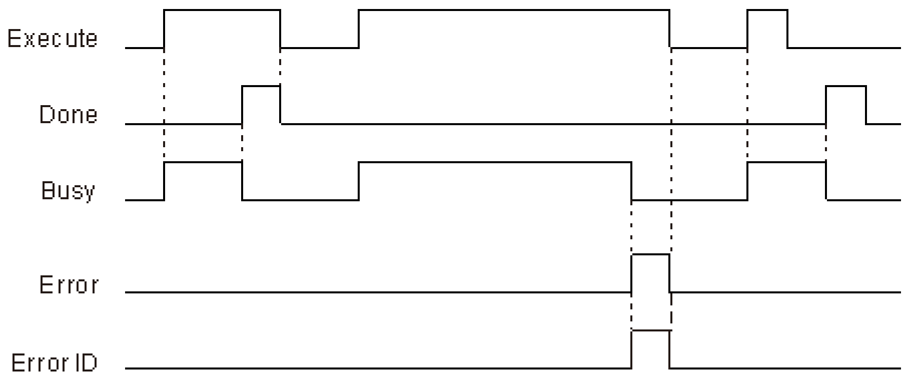
Name	Function	Data type	Output range (Default value)
Done	True when the axis group is reset.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Done	• When the axis group is reset	• When Execute shifts from True to False. • If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True.

Name	Timing for shifting to True	Timing for shifting to False
		● When *Error* shifts to True.
Error(ErrorID)	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



● Troubleshooting

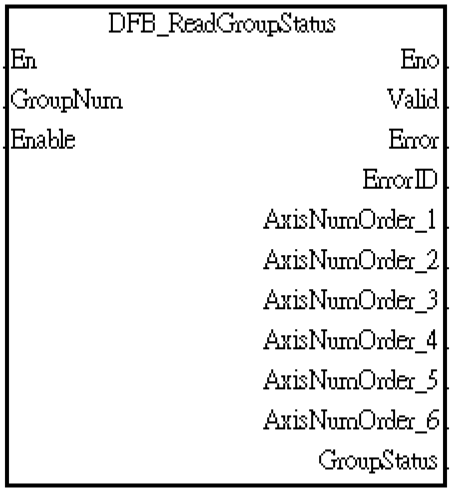
- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_ReadGroupStatus

FB/FC	Description
FB	DFB_ReadGroupStatus reads the axis numbers in an axis group, and indicates the status of the axis group at *GroupStatus*.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
GroupNum	Specifies the number of the axis group.	WORD	1~16 (0)	When *Enable* shifts to True.
Enable	Reads the axis numbers and the status of the axis group.	BOOL	True/False (False)	-

- **Outputs**

Name	Function	Data type	Output range (Default value)
Valid	True when the axis group state at the output is available.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions	DWORD	16#0~16#FFFFFFFF (0)
AxisNumOrder_1	Indicates the axis number for the 1^{st} axis (X-axis)	WORD	0~32(0)
AxisNumOrder_2	Indicates the axis number for the 2^{nd} axis (Y-axis)	WORD	0~32(0)

Name	Function	Data type	Output range (Default value)
AxisNumOrder_3	Indicates the axis number for the 3rd axis (Z-axis)	WORD	0~32(0)
AxisNumOrder_4	Indicates the axis number for the 4th axis (A-axis)	WORD	0~32(0)
AxisNumOrder_5	Indicates the axis number for the 5th axis (B-axis)	WORD	0~32(0)
AxisNumOrder_6	Indicates the axis number for the 6th axis (C-axis)	WORD	0~32(0)
GroupStatus	Indicates the axis group status*1	eMC_GROUP_STATE_MACHINE*2	0: GroupDisable 256: GroupStandby 512: GroupStopping 576: GroupMoving 768: GroupErrorStop (0)

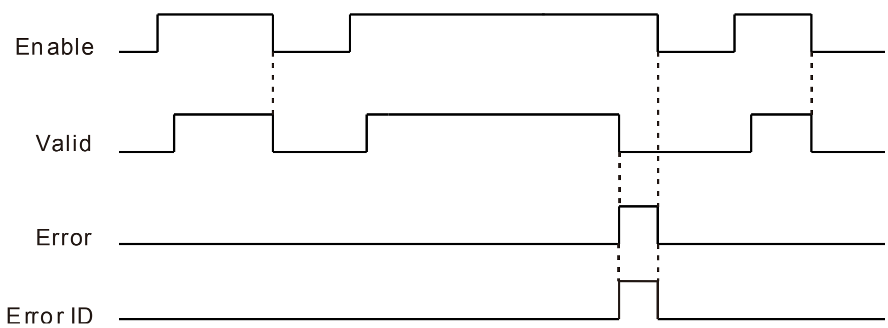
Note:

1. Refer to **7.4 State Transitions** of ***AH Motion Controller – Operation Manual*** for more information about state transitions.
2. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Valid	• When the axis group state at the output is available.	• When *Enable* shifts to False. • When *Error* shifts to True.
Error(ErrorID)	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)
AxisNumOrder_1 … AxisNumOrder_6	Updates value continuously when *Enable* is True	Updates value continuously when *Enable* is True
GroupStatus	Updates value continuously when *Enable* is True	Updates value continuously when *Enable* is True

■ Timing Diagram

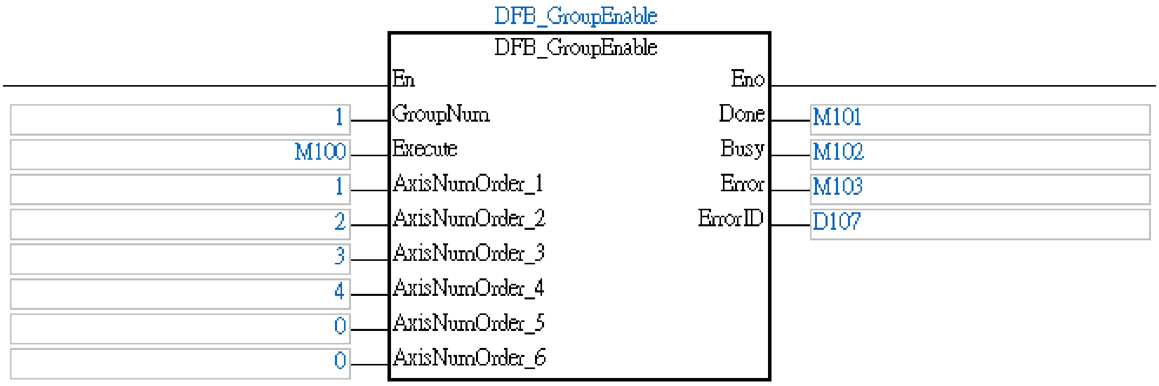


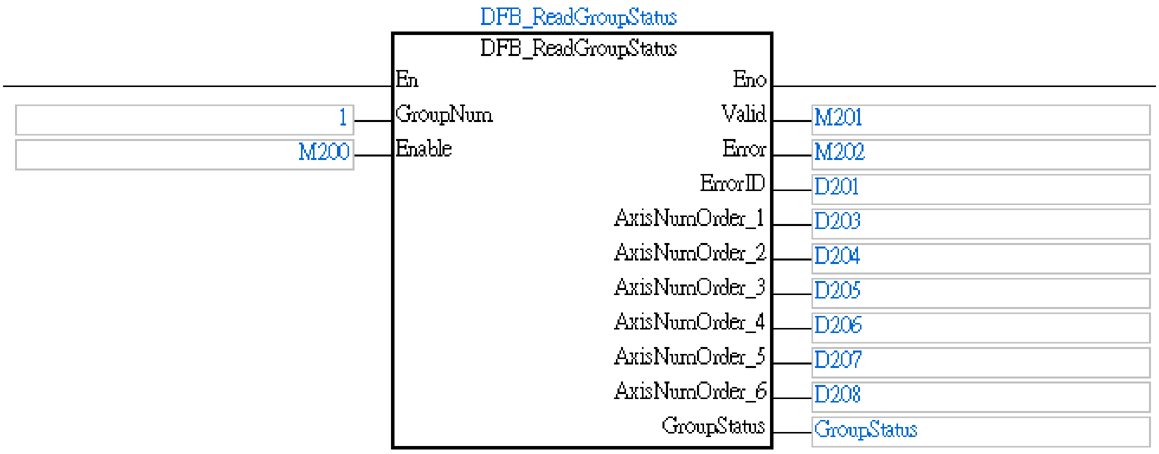
Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Programming Example

The example uses DFB_ReadGroupStatus to read the axis numbers and the group status when the specified group is enabled.





1. DFB_GroupEnable is required to enable the axis group for executing coordinated motion in 4 axes (axis 1 to axis 4).
2. Set M100 to True to enable the axis group number 1 which uses axis 1 to axis 4.
3. M101(*Done*)=True indicates the completion of the enabling of axis group number 1. Group state will change from “Disabled” to “Standby”, and the axis state in the group will change from “Standstill” to “Coordinated”.
4. When the axis group is enabled, it can be used for executing coordinated motion or multi-axis interpolation.
5. When the group motion is in operation, you can read the status of axis group number 1 by DFB_ReadGroupStatus .
6. Set M200 to True to read the axis numbers and the group state at the outputs of DFB_ReadGroupStatus.

Supported Products

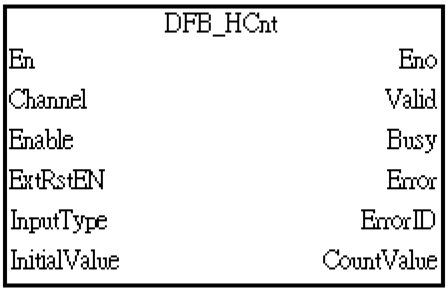
- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

3.3.3 Auxiliary Motion Control Function Blocks

Categories		Name	Description
Auxiliary	High speed counter	DFB_HCnt	DFB_HCnt enables the specified high speed counter according to the specified parameters and monitors the count value.
Auxiliary	High speed timer	DFB_HTmr	DFB_HTmr enables the specified high speed timer channel according to the specified parameters and monitors and timed value.
Auxiliary	Comparison	DFB_Compare	DFB_Compare compares the designated source with a specified value and outputs the specified results on a desired device when the comparison result is True.
Auxiliary	Comparison	DFB_CmpOutRst	DFB_CmpOutRst monitors the output results and clears the output states triggered by the comparators.
Auxiliary	Capture	DFB_Capture2	DFB_Capture2 captures the commanded pulses of the specified axis according to the designated external trigger deivce.

DFB_HCnt

FB/FC	Description
FB	DFB_HCnt enables the specified high speed counter according to the specified parameters and monitors the count value.



- You can specify the high speed counter to be enabled by *Channel*, the input pulse type by *InputType*, the initial value of the counter by *InitiaValue*, and trigger the corresponding X signal to clear the count value by setting *ExtRstEN* as True.
- To read the count value during the high speed counter operation, you can monitor the output *CountValue*.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Channel	Specifies the counting channel of the high speed counter.	eDFB_HCNT*[1]	200: DFB_AC0 204: DFB_AC4 208: DFB_AC8 212: DFB_AC12 216: DFB_AC16 220: DFB_AC20 (0)*[2]	When *Enable* shifts to True
Enable	Enables the specified high speed counter when *Enable* changes to True.	BOOL	True/False(False)	-
ExtRstEN	Resets the counter	BOOL	True/False(False)	When *Enable* shifts to True
InputType	Input pulse type U/D: counting up/down P/D: pulse/direction A/B: A/B-phase 4A/B: quadruple A/B-phase	eDFB_HCNT_INTYPE	0: UD 1: PD 2: AB 3: AB4 (0)	When *Enable* shifts to True
InitiaValue	Initial value of the specified counter	DWORD	Negative integer, positive integer or 0 (0)	When *Enable* shifts to True

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.
2. The setting values, devices, corresponding terminals for *Channel* inputs:

Channel	Setting value	Device	Corresponding terminals for counting
0	200	AC00	X0.8, X0.9
1	204	AC04	X0.10, X0.11
2	208	AC08	X0.12, X0.13
3	212	AC12	X0.14, X0.15
4	216	AC16	X1.0, X1.1
5	220	AC20	X1.2, X1.3

3. For resetting the counter value, refer to the selected DFB_Hcnt Channel and its corresponding terminals.

Channel	Corresponding terminals for resetting
0	X0.0
1	X0.1
2	X0.2
3	X0.3
4	X1.4
5	X1.5

● Outputs

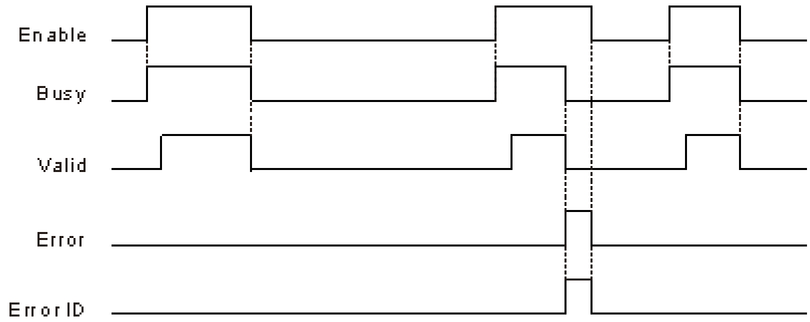
Name	Function	Data type	Output range (Default value)
Valid	True when the output value is valid.	BOOL	True/False (False)
Busy	True when the instruction is enabled.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
CountValue	Count value of the counter	DWORD	Continuously updates value when *Valid* is True.

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Valid	● When the output value is valid; one scan cycle after *Enable* shifts to True.	● When *Enable* shifts to False. ● When *Error* shifts to True.
Busy	● When *Enable* shifts to True.	● When *Enable* shifts to False. ● When *Error* shifts to True.

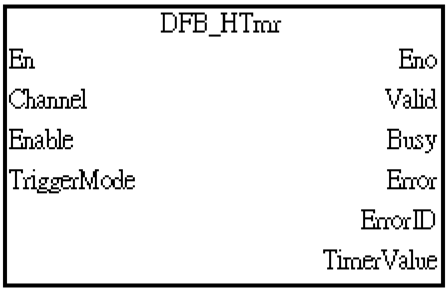
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



DFB_HTmr

FB/FC	Description
FB	DFB_HTmr enables the specified high speed timer channel according to the specified parameters and monitors and timed value.



- You can specify the high speed timer to be enabled by *Channel* and the timing mode by *TriggerMode.*
- To read the timed value during the high speed timer operation, you can monitor the output *TimerValue*.

● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Channel	Specifies the timing channel of the high speed timer.	WORD	200: DFB_AC0 204: DFB_AC4 208: DFB_AC8 212: DFB_AC12 (0) *1	When *Enable* shifts to True
Enable	Enables the specified high speed timer when *Enable* changes to True.	BOOL	True/False(False)	-
TriggerMode	Timing mode settings*2 **Up_Down**: measuring the interval between the rising edge and the falling edge of a pulse. **Up_Up**: measuring the interval between the rising edge of a pulse and the rising edge of the following pulse.	BOOL	mcUp_Down: False mcUp_Up: True (False)	When *Enable* shifts to True

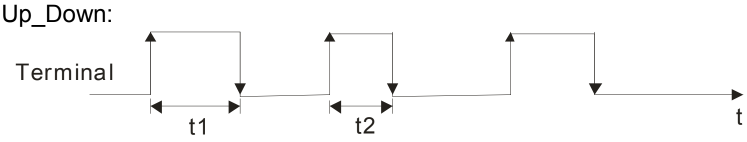
***Note**:

1. The setting value of *Channel* input

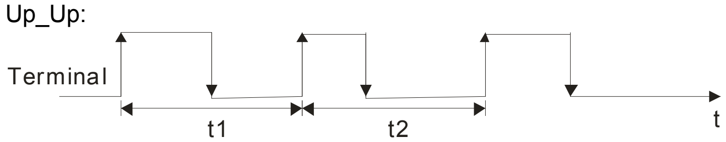
Channel	Setting value	Terminals
0	200	X0.0
1	204	X0.1
2	208	X0.2
3	212	X0.3

2. Timing mode

Up_Down:



Up_Up:



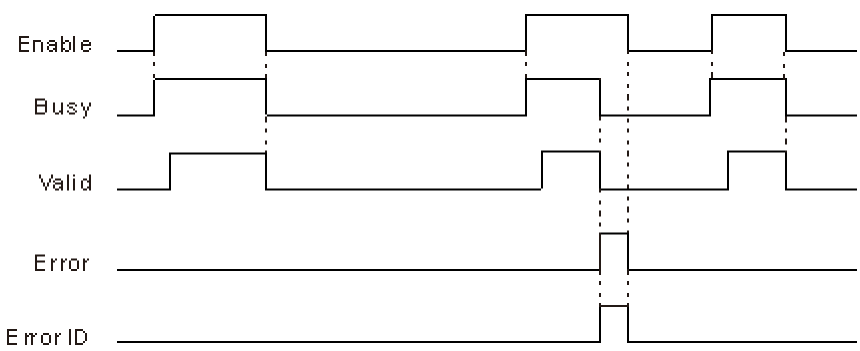
- **Outputs**

Name	Function	Data type	Output range (Default value)
Valid	True when the output value is valid	BOOL	True/False (False)
Busy	True when the instruction is enabled.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
TimerValue	Timed value (Unit: 0.01us)	DWORD	Continuously updates value when *Valid* is True.

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Valid	• When the output value is valid; one scan cycle after *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True.
Busy	• When *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Execute* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**

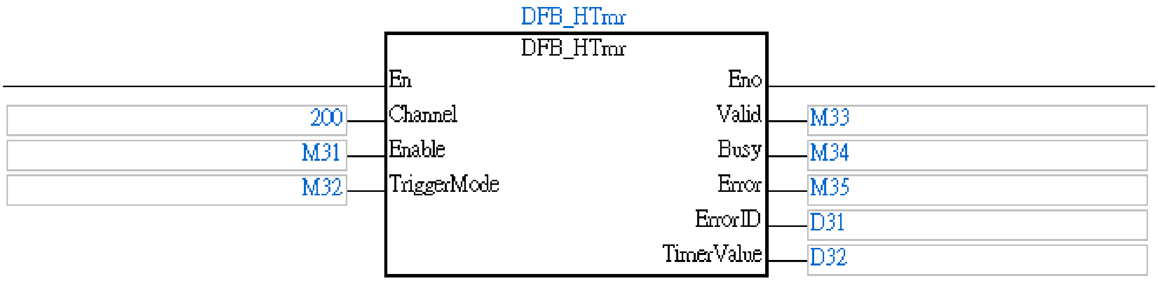


Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Programming Example

Using channel 0 (input X0.0) with the *TriggerMode* from True shifting to False to obtain a timer value of 10 seconds (*TimerValue*):



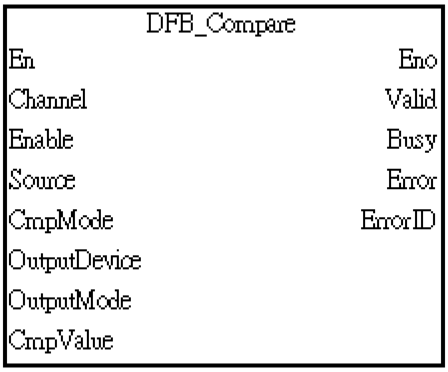
1. Set M32=ON to specify the timing mode as UP_UP, measuring the interval between rising edge and falling edge of a pulse.
2. Set M31=ON to enable channel 0 for high speed timing.
3. Trigger the rising edge of X0.0 to start timing, and trigger the falling edge of X0.0 after 10 seconds.
4. Check the value in D32 to confirm if a timed value approximately equals to 10 seconds is recorded.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_Compare

FB/FC	Description
FB	DFB_Compare compares the designated source value and the setting value and then to Set or Reset the desired device when the comparison result is True or False.



- You can specify the high speed comparator to be enabled by *Channel*, designate the comparison source by *Source*, and set the parameters for comparison by *CmpMode and CmpValue.*
- When the comparison result is True, DFB_Compare will outputs the results according to the settings of *OutputDevice and OutputMode.*

Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Channel	Specifies the number of the high speed comparator.	eDFB_COMP	0: Ch0 1: Ch1 2: Ch2 3: Ch3 4: Ch4 5: Ch5 6: Ch6 7: Ch7 8: Ch8 9: Ch9 10: Ch10 11: Ch11 12: Ch12 13: Ch13 14: Ch14 15: Ch15 (0)	When *Enable* shifts to True
Enable	Enables the specified high speed comparator when *Enable* changes to True.	BOOL	True/False(False)	-

Name	Function	Data type	Setting value (Default value)	Timing for updating
Source	Designates the source of the comparison. 0: Axis 1 1: Axis 2 2: Axis 3 3: Axis 4 4: DFB Hcnt CH0 setting value 5: DFB Hcnt CH1 setting value 6: DFB Hcnt CH2 setting value 7: DFB Hcnt CH3 setting value	eDFB_COMP_SOURCE	0: Axis 1 1: Axis 2 2: Axis 3 3: Axis 4 4: DFB_AC0 5: DFB_AC4 6: DFB_AC8 7: DFB_AC12 (0)	When *Enable* shifts to True
CmpMode	Comparison condition 0: Equal (=) 1: Bigger_Equal (≧) 2: Smaller_Equal (≦)	eDFB_COMP_MOD*1	0: Equal 1: Bigger_Equal 2: Smaller_Equal (0)	When *Enable* shifts to True
OutputDevice	Designates the output device when the comparison result is True 0: set Y0.8 1: set Y0.9 2: set Y0.10 3: set Y0.11 4: reset DFB Hcnt CH0 setting value 5: reset DFB Hcnt CH1 setting value 6: reset DFB Hcnt CH2 setting value 7: reset DFB Hcnt CH3 setting value	eDFB_COMP_OUTDEV	0: SetY08 1: SetY09 2: SetY10 3: SetY11 4: RstAC0 5: RstAC4 6: RstAC8 7: RstAC12 (0)	When *Enable* shifts to True
OutputMode	Specifies the output method CmpSet: set ON the device CmpRst: reset the device	BOOL	mcCmpSet: True mcCmpRst: False (False)	When *Enable* shifts to True
CmpValue	Specifies the comparison value	DWORD	Positive integer, negative integer or 0 (0)	When *Enable* shifts to True

***Note:**

1. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

● Outputs

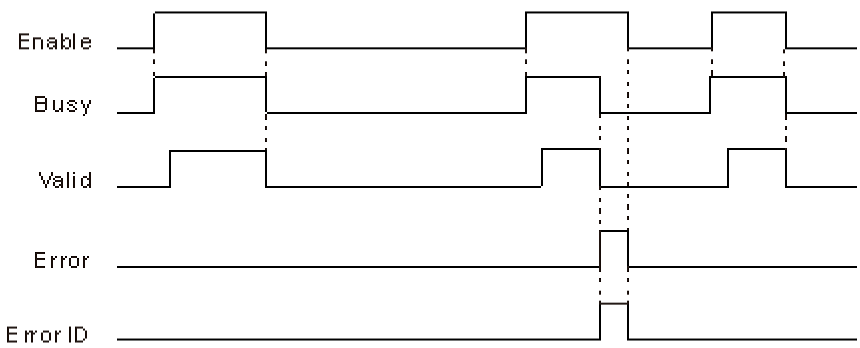
Name	Function	Data type	Output range (Default value)
Valid	True when the output value is valid.	BOOL	True/False (False)
Busy	True when the instruction is enabled.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs.	DWORD	16#0~16#FFFFFFFF (0)

	Refer to **Appendices** for error code descriptions.		

■ **Outputs Update Timing**

Name	Timing for shifting to True	Timing for shifting to False
Valid	• When the output value is valid; one scan cycle after *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True.
Busy	• When *Enable* shifts to True.	• When *Enable* shifts to False. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	• When *Enable* shifts from True to False. (Error code is cleared)

■ **Timing Diagram**

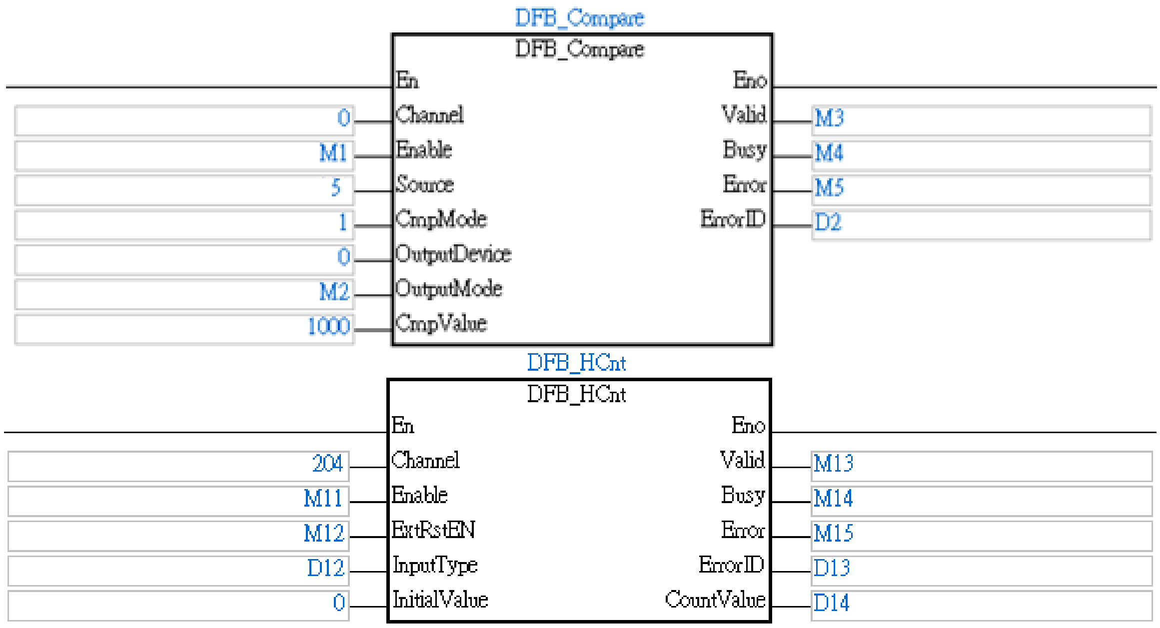


● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

This example compares the counting value of channel 1 (DFB_Hcnt CH1) with the specified value 1000(*CmpValue*) with the condition of "≧" (*CmpMode*=1), and set Y0.8 (*OutputDevice*=0) =ON (*OutputMode*=True) when the comparison result is True.



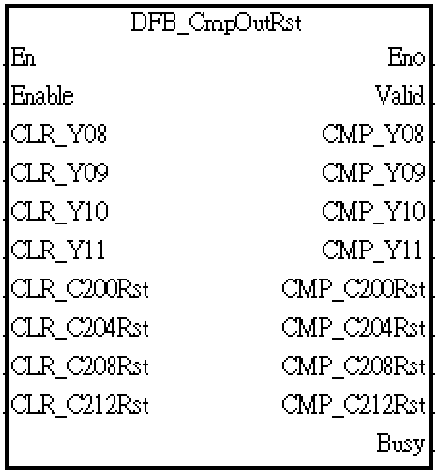
1. Set M2(*OutputMode*) to True and *OutputDeivce*=0 to specify the output operation of the instruction when the comparison result is True: Set Y0.8=ON.
2. Set M1=ON to enable the comparator of number 0.
3. Set M11=ON to enable the counting on channel 1.
4. Confirm if Y0.8 is set to True when the count value in D14 is greater or equals to 1000.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_CmpOutRst

FB/FC	Description
FB	DFB_CmpOutRst monitors the output results and clears the output states triggered by the comparators.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Enable	Enables the instruction when *Enable* changes to True.	BOOL	True/False(False)	-
CLR_Y0 08	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.
CLR_Y0 09	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.
CLR_Y0 10	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.
CLR_Y0 11	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.
CLR_AC0Rst	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.
CLR_AC4Rst	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.

Name	Function	Data type	Setting value (Default value)	Timing for updating
CLR_AC8Rst	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.
CLR_AC12Rst	Resets the output state.	BOOL	True/False(False)	Continuously updates value when *Enable* is True.

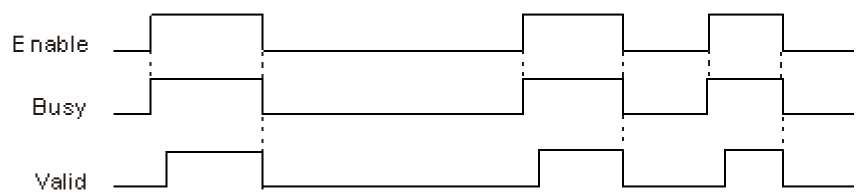
- ## Outputs

Name	Function	Data type	Output range (Default value)
Valid	True when the output value is valid.	BOOL	True/False (False)
CMP_Y0 08	Indicates the comparison output states	BOOL	True/False (False)
CMP_Y0 09	Indicates the comparison output states	BOOL	True/False (False)
CMP_Y0 10	Indicates the comparison output states	BOOL	True/False (False)
CMP_Y0 11	Indicates the comparison output states	BOOL	True/False (False)
CMP_AC0Rst	Indicates the comparison output states	BOOL	True/False (False)
CMP_AC4Rst	Indicates the comparison output states	BOOL	True/False (False)
CMP_AC8Rst	Indicates the comparison output states	BOOL	True/False (False)
CMP_AC12Rst	Indicates the comparison output states	BOOL	True/False (False)
Busy	True when the instruction is enabled.	BOOL	True/False (False)

■ Outputs Update Timing

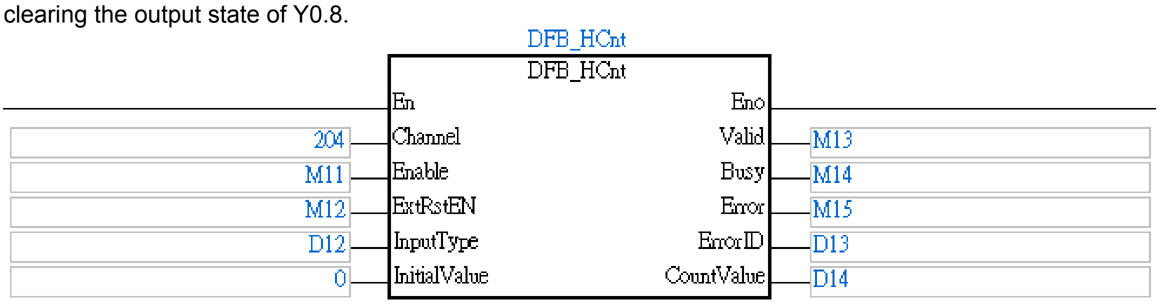
Name	Timing for shifting to True	Timing for shifting to False
Valid	● When the output value is valid; one scan cycle after *Enable* shifts to True.	● When *Enable* shifts to False.
CMP_Y0 08~ CMP_AC12Rst	Continuously updates value when *Valid* is True	Continuously updates value when *Valid* is True
Busy	● When *Enable* shifts to True.	● When *Enable* shifts to False.

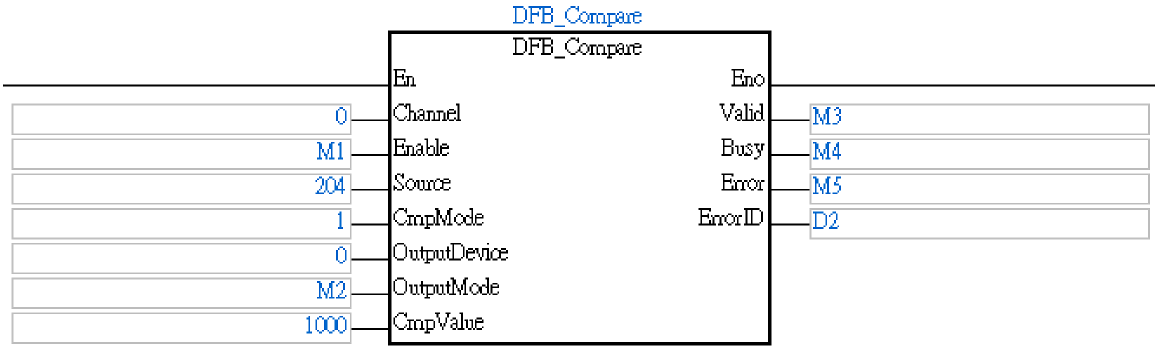
■ Timing Diagram

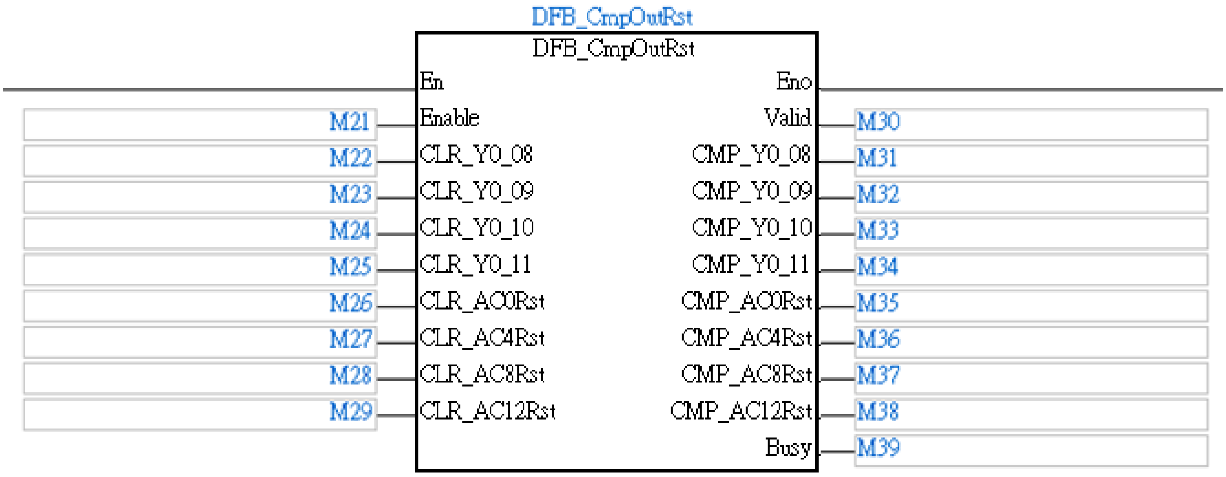


● Programming Example

This example uses DFB_CmpOutRst to monitor and reset the comparison results activated by DFB_Compare which compares the counting value of channel 1 (AC04) with the specified value 1000(*CmpValue*) with the condition of "≧" (*CmpMode*=1), and set Y0.8=ON when the comparison result is True. You can use DFB_CmpOutRst to monitor if M31 (Y0.8) is ON. If you want to reset the output state of Y0.8, set the corresponding input M22 (*CLR_Y0 08*) to ON for clearing the output state of Y0.8.







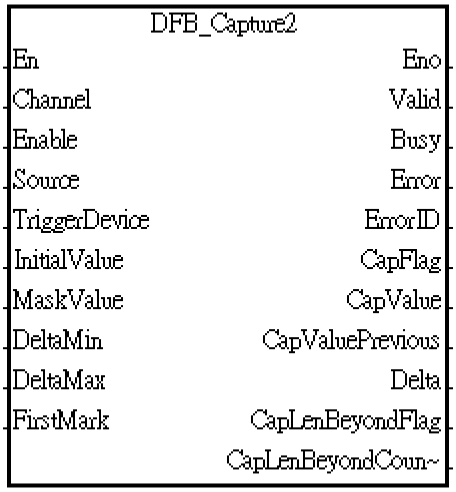
1. Set M11(*Enable*) to True to enable the counting on channel 1.
2. Set M2(*OutputMode)* to True and *OutputDeivce*=0 to specify the output operation of DFB_Compare when the comparison result is True: Set Y0.8=ON.
3. Set M1=ON to enable the comparator of number 0.
4. Use DFB_CmpOutRst to monitor if M31(*CMP_Y0 08*) =ON and Y0.8 is set to True when the count value in D14 is greater or equals to 1000.
5. Set M22(*CLR_Y0 08*) to True and M21(*Enable*) to True to clear the output state of Y0.8
6. Confirm if Y0.8 is reset and M31(*CMP_Y0 08*) is False.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_Capture2

FB/FC	Description
FB	DFB_Capture2 captures the commanded pulses of the specified axis according to the designated external trigger deivce.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Channel	Specifies the number of the Capture.	eDFB_CAP*1	0: Ch0 1: Ch1 2: Ch2 3: Ch3 4: Ch4 5: Ch5 6: Ch6 7: Ch7 (0)	When *Enable* shifts to True.
Enable	Enables the specified Capture when *Enable* changes to True.	BOOL	True/False (False)	-
Source	Designates the source of the Capture. 0: Axis 1 current position 1: Axis 2 current position 2: Axis 3 current position 3: Axis 4 current position 4: DFB Hcnt CH0 setting value 5: DFB Hcnt CH1 setting value 6: DFB Hcnt CH2 setting value 7: DFB Hcnt CH3 setting value	eDFB_CAP_SOURCE*1	0: Axis 1 1: Axis 2 2: Axis 3 3: Axis 4 4: DFB_AC0 5: DFB_AC4 6: DFB_AC8 7: DFB_AC12 (0)	When *Enable* shifts to True.

Name	Function	Data type	Setting value (Default value)	Timing for updating
TriggerDevice	Designates the external trigger device for the Capture 0: trigger by X0.0 signal 1: trigger byX0.1 signal 2: trigger by X0.2 signal 3: trigger by X0.3 signal 8: trigger by X0.8 signal 9: trigger by X0.9 signal 10: trigger by X0.10 signal 11: trigger by X0.11 signal 12: trigger by X0.12 signal 13: trigger by X0.13 signal 14: trigger by X0.14 signal 15: trigger by X0.15 signal	eDFB_CAP_TRIG_DEV*[1]	0: X0p0 1: X0p1 2: X0p2 3: X0p3 8: X0p8 9: X0p9 10: X0p10 11: X0p11 12: X0p12 13: X0p13 14: X0p14 15: X0p15 (0)	When *Enable* shifts to True.
InitialValue	Specifies the initial value of the reference for the mask range. (Unit: pulse)	LREAL	-2,147,483,648 ~ 2,147,483,647 (0)	When *Enable* shifts to True.
MaskValue	Specifies the value used to define the mask range (Unit: pulse)	LREAL	0~2,147,483,647 (0)	When *Enable* shifts to True.
DeltaMin	Defines the minimum difference between each Capture. (Unit: pulse) **Note**: If *DeltaMin* and *DeltaMax* are both set to 0, the system will not check if the difference between between each Capture is within proper range or not.	LREAL	-2,147,483,648 ~ 2,147,483,647 (0)	Updates value continuously hwen *Enable* is True.
DeltaMax	Defines the maximum difference between each Capture. (Unit: pulse) **Note**: If *DeltaMin* and *DeltaMax* are both set to 0, the system will not check if the difference between between each Capture is within proper range or not.	LREAL	-2,147,483,648 ~ 2,147,483,647 (0)	Updates value continuously hwen *Enable* is True.
FirstMark	Selects the reference of the mask range. True: Use the first captured value as the reference for the mask range.	BOOL	True/False (False)	When *Enable* shifts to True.

Name	Function	Data type	Setting value (Default value)	Timing for updating
	False: Use the initial value as the reference for the mask range.			

***Note:**

2. Refer to **Section 2.4 Data Type Unit (DUT): ENUM** for explanation on using enumerations.

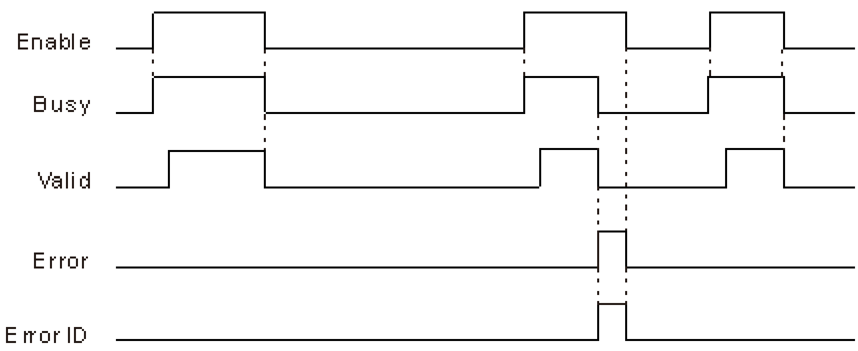
Outputs

Name	Function	Data type	Output range (Default value)
Valid	True when the output value is valid.	BOOL	True/False (False)
Busy	True when the instruction is enabled.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	0x0000, 0x3100, 0x3101, 0x3102 (0)
CapFlag	Indicates that the current Capture is valid. (The flag shifts to Ture for one scan cycle and will be reset immediatly)	BOOL	True/False (False)
CapValue	The captured value (Unit: pulse)	LREAL	-2,147,483,648 ~ 2,147,483,647 (0)
CapValuePrevious	The previous captured value (Unit: pulse)	LREAL	-2,147,483,648 ~ 2,147,483,647 (0)
Delta	The difference between the previous and the current captured values. (Unit: pulse)	LREAL	-2,147,483,648 ~ 2,147,483,647 (0)
CapLenBeyondFlag	Indicates that a capture is failed. (The flag shifts to Ture for one scan cycle and will be reset immediatly)	BOOL	True/False (False)
CapLenBeyondCount	Counts the number of the failed Capture.	DWORD	0~2,147,483,647 (0)

■ Outputs Update Timing

Name	Timing for shifting to True	Timing for shifting to False
Valid	● When the values at the outputs are available.	● When the motion stops. ● When *Enable* shifts to False. ● When *Error* shifts to True.
Busy	● When the instruction is enabled.	● When *Enable* shifts to False.
Error/ErrorID	● When the specified axis is already in motion. ● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When *Enable* shifts from True to False. (Error code is cleared)
CapFlag	Updates value continuously when *Valid* is True.	Updates value continuously when *Valid* is True.
CapValue	Updates value continuously when *Valid* is True.	Updates value continuously when *Valid* is True.
CapValuePrevious	Updates value continuously when *Valid* is True.	Updates value continuously when *Valid* is True.
Delta	Updates value continuously when *Valid* is True.	Updates value continuously when *Valid* is True.
CapLenBeyondFlag	Updates value continuously when *Valid* is True.	Updates value continuously when *Valid* is True.
CapLenBeyondCount	Updates value continuously when *Valid* is True.	Updates value continuously when *Valid* is True.

■ Timing Diagram



● Function

■ *MaskValue*

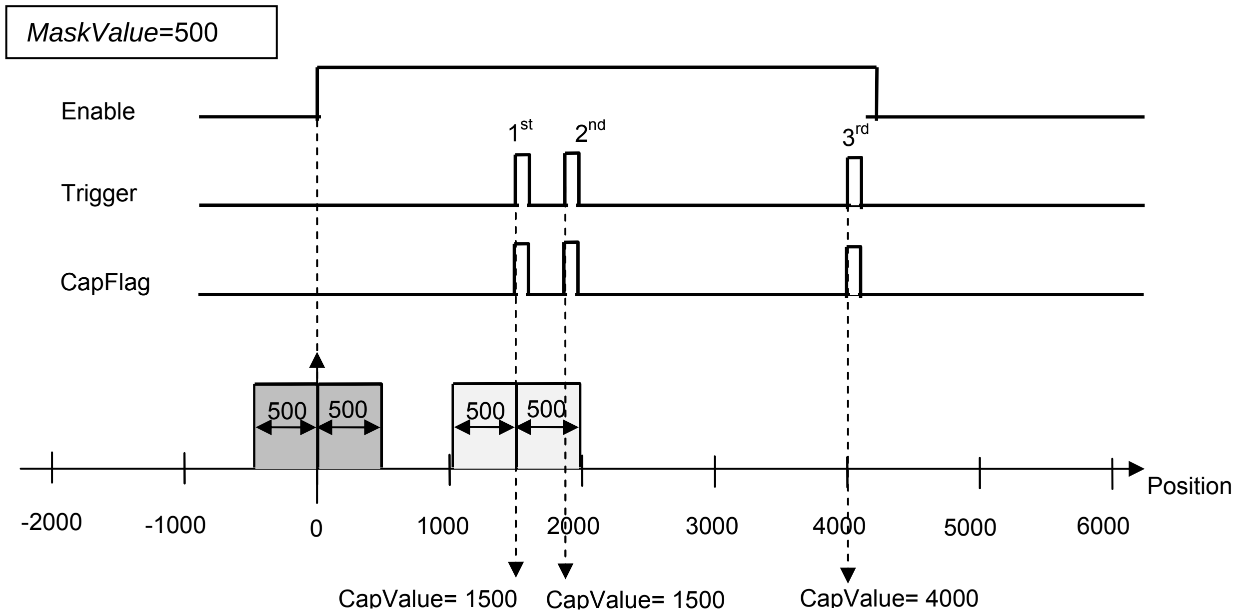
Refer to the below diagram for the explanation of *MaskValue:*

1. Specify 500 to *MaskValue*, and 0 to *InitialValue*. When *Enable* is set to True, the initial value wille be the center reference of the mask range, and the mask range will be -500~500. Within the mask range, the Capture will be invalid.

2. In below diagram, the 1st Capture occurred out of the mask range (-500~500), so the captured value will change from 0 (initial value) to 1500.

3. When the captured value changes to 1500, 1500 will become the reference center of the mask range, therefore the capture occurred within 1000~2000 will be invalid. When the 2nd Capture is triggered, the captured value will remain 1500.

4. When the 3rd Capture is triggered out of the mask range between 1000 and 2000, the captured value will be updated as 4000.



■ ***DeltaMin, DeltaMax, CapLenBeyondFlag, CapLenBeyondCount***

1. *DeltaMin/DeltaMax* defines the minimum/maximum difference between each Capture.

2. The function of *DeltaMin/DeltaMax* is to judge if a trigger mark is missed and the Capture is not executed. For example, if the value of *DeltaMin* is 1000 and *DeltaMax* is 1300, when the detected distance between 2 Capture exceeds 1000~1300, the system will flag this situation as trigger mark missing.

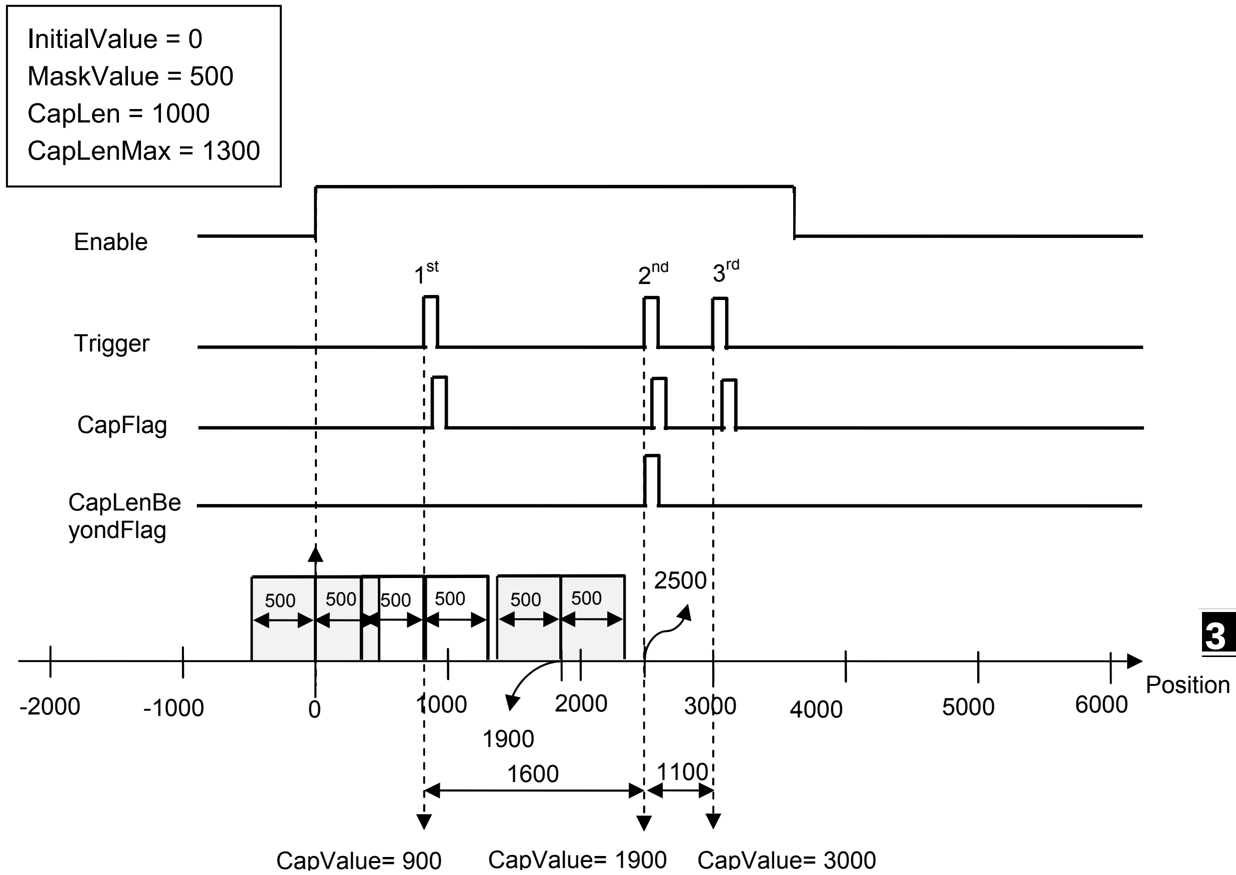
3. When a mark missing condition occurs, *CapLenBeyondFlag* shifts to Ture for one scan cycle and will be reset immediately. At the same time *CapLenBeyondCount* counts 1.

4. Refer to the below diagram for the explanation of these inputs and outputs:

   a. *InitialValue*=0, so the mask range is between -500~500. The 1st Capture occurs at 900 which is bigger than the mask range, so the captured value is 900.

   b. The 2nd Capture occurs at 2500. Because *DeltaMax* is set to 1300 and *DeltaMin* is set to 1100 (1100-1300), a trigger mark missing condition is flagged for a scan cycle.

   c. The 3rd Capture occurs at 3700. Because the difference between 3700 and the previous captured value 2500 is 1200, which is within the range of 1100~1300 *(DeltaMin/DeltaMax)*, also 3700 is out of the mask range 2000~3000, the captured value changes to 3700 in this case, and *CapLenBeyondFlag* will not change to True.



■ ***FirstMark***

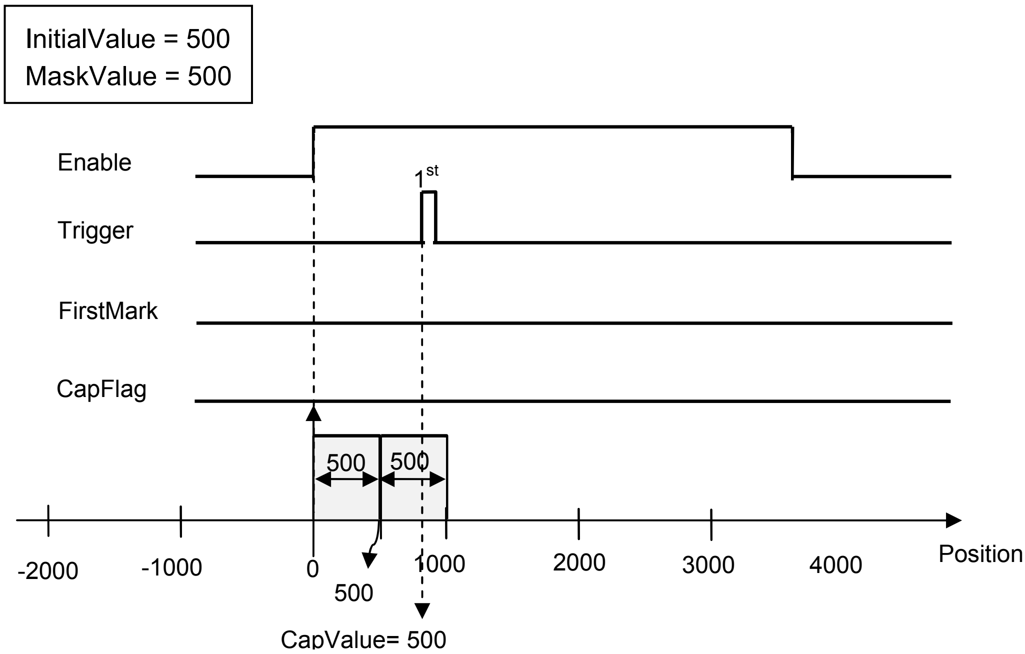
FirstMark selects the reference of the mask range.

1. *FirstMark*=False

   When the instruction is enabled with the condition *FirstMark*=False, the instruction will use the initial value (*InitialValue*) as the reference center for the mask range when the first Capture occurs. Refer to below diagram:

   a. When *InitialValue*=600, the initial captured value will be 600 when the instruction is enabled. In this case, 600 will also be the reference center for the mask range when the first Capture occurs.

   b. Because *FirstMark*=False, the mask range will be 100~1100. In the diagram the 1st Capture occurs within the mask range, so the captured value will remain the initial value 600.



2. *FirstMark*=True

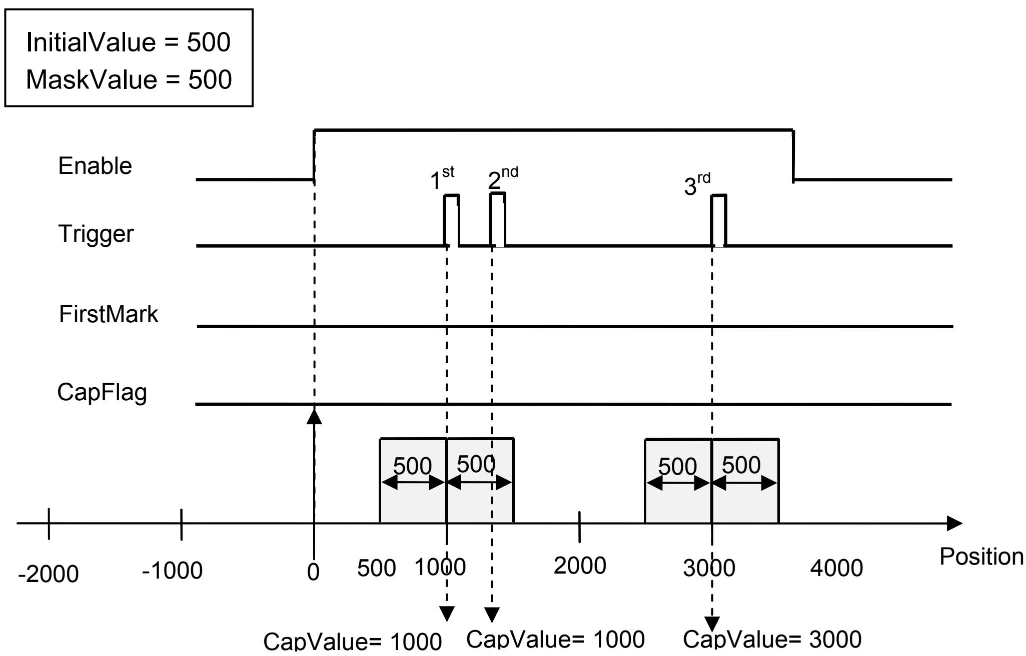
   When the instruction is enabled with the condition *FirstMark*=True, the instruction will use the first captured value as the reference center for the mask range when the first Capture occurs. Refer to below diagram:

   a. When *InitialValue*=600, the initial captured value will be 600 when the instruction is enabled

   b. Because *FirstMark*=True, the instruction will use the first captured value as the reference center for the mask range. The mask function will be invalid until the first capture occurs. In the diagram the 1st Capture occurs at 1000, so the captured value changes from 600 to 1000, which is also the reference center of the mask range. The mask range will be 500~1500.

   c. The 2nd Capture occurs withn the mask range 500~1500, so the captured value remains 1000.

   d. The 3rd Capture occurs at 3000, which is out of the mask range 500~1500, so the captured value changes to 3000.

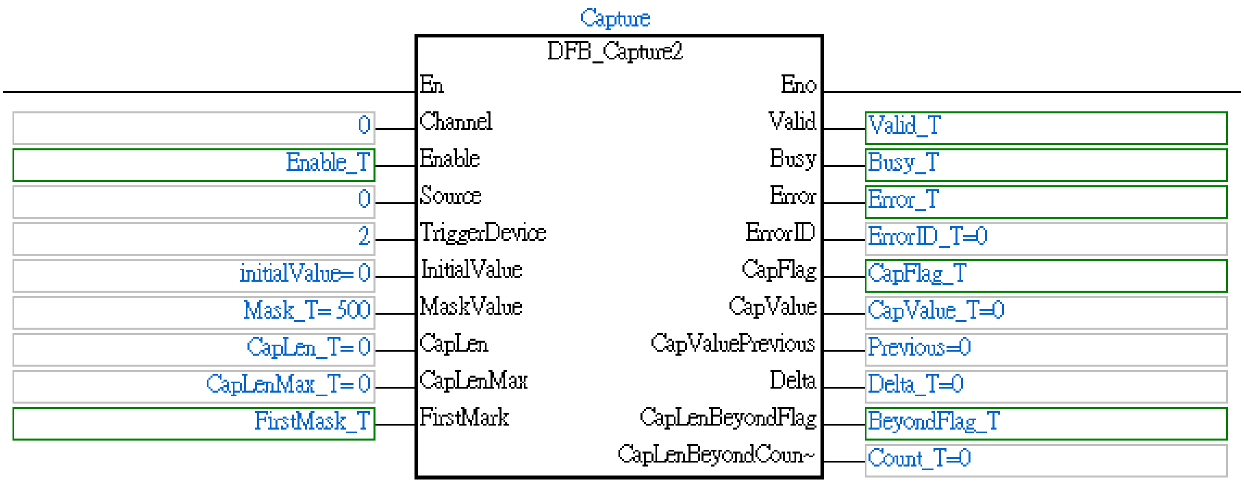


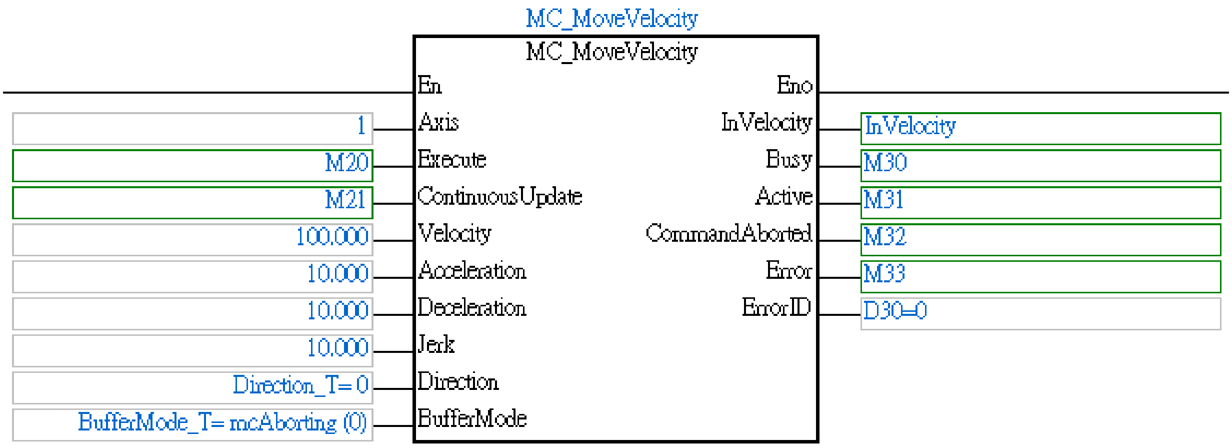
● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of “Errorstop”, *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual.***

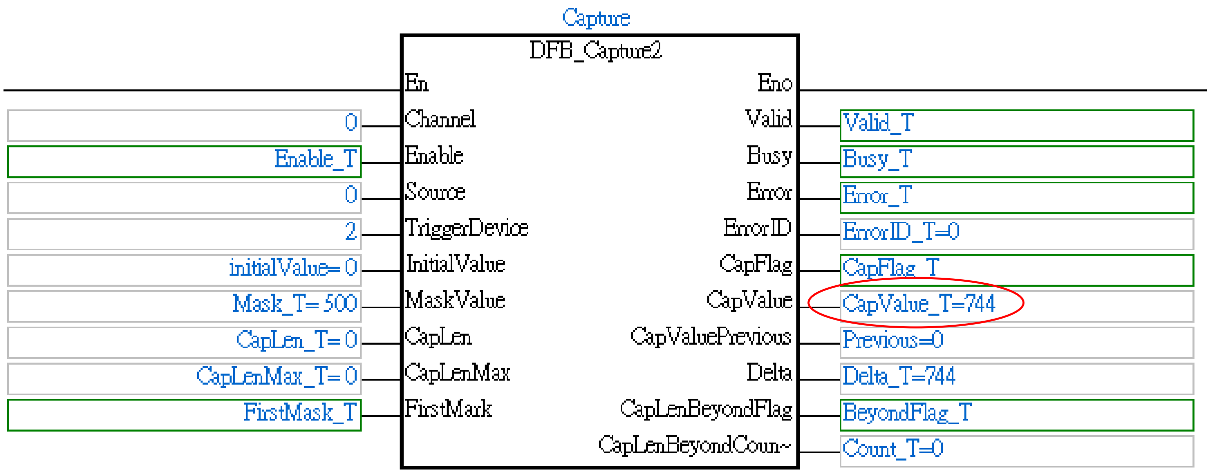
● Programming Example

This example uses DFB_Capture2 and MC_MoveVelocity to perform the Capture function.





1. Set *MaskValue*=500 and *InitialValue*=0. When the instruction is enabled, it will take the initial value 0 as the reference center of the mask range. The mask range will be -500~500, and the Capture occurs within this range will be invalid.
2. The 1st Capture occurs at 744 which is out of the range of -500~500, so the captured value (*CapValue*) shows 744, as the below circled area.



3. When the captured value changes to 744, the instruction will tak 744 as the reference center for the mask range, i.e. Capture triggered within 244~1244 will not be valid. If the 2nd Capture occurs at 1000, the captured value will remain 744.

3

● Supported Products

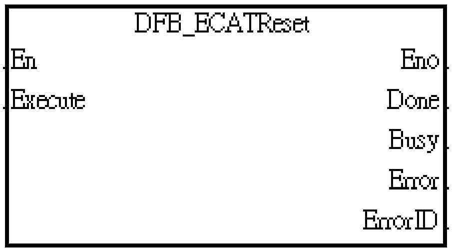
- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

3.3.4 Network Function Blocks

Categories		Name	Description
Network	Communication	DFB_ECATReset	DFB_ECATReset resets an abnormal EtherCAT network.
		DFB_ECATServoRead	DFB_ECATServoRead reads the values of parameters from the Delta servo drive specified on an EtherCAT network.
		DFB_ECATServoWrite	DFB_ECATServoWrite writes the values of parameters into the Delta servo drive specified on an EtherCAT network.
		DFB_SDO_Read	DFB_SDO_Read reads the values of parameters from the specified OD of the EtherCAT Slave via SDO.
		DFB_SDO_Write	DFB_SDO_Write writes the values of parameters into the specified OD of the EtherCAT Slave via SDO.

DFB_ECATReset

FB/FC	Description
FB	DFB_ECATReset resets an abnormal EtherCAT network.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction to reset an EtherCAT network when *Execute* changes to True.	BOOL	True/False (False)	-

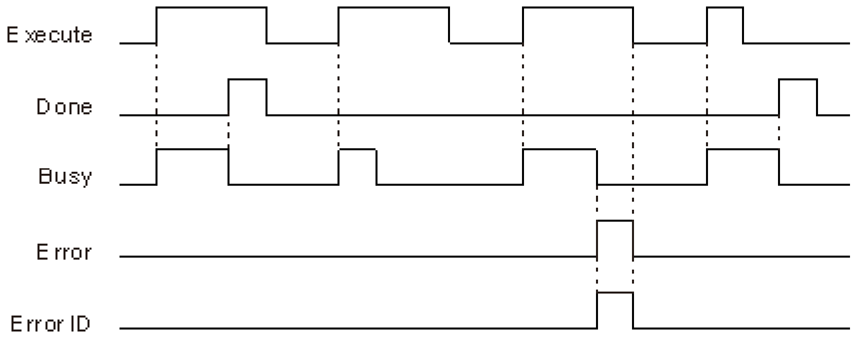
● Outputs

Name	Function	Data type	Output range (Default value)
Done	Indicates the completion of the network reset process.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to t **Appendices** for error code descriptions	DWORD	0, 16#3901, 16#3909

■ Outputs Update Timing

Name	Timing for shiftng to True	Timing for shifting to False
Done	● When network reset process is completed.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, it will be True for only one period and immediately shift to False.
Busy	● When *Execute* shifts to True	● When *Done* shifts to True. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction.(Error code is recorded)	● When Execute shifts from True to False. (Error code is cleared)

■ **Timing Diagram**



Function

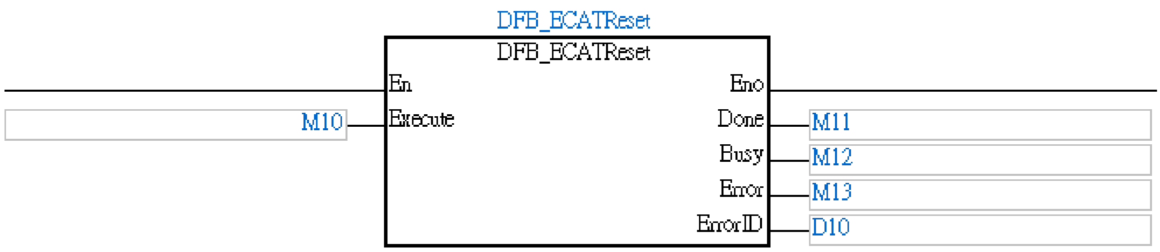
After an EtherCAT network is reset by DFB_ECATReset, users will have to use MC_Power to enable the motion control CPU and the servo drive which are used in the EtherCAT network.

Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of "Errorstop", *Error* will change to True and the axis will stop moving. You can refer to *ErrorID* (Error Code) to address the problem.
- Information regarding error codes and indicators are attached as Appendices for a quick reference. The detailed troubleshooting procedures are explained in ***AH Motion Controller – Operation Manual***.

Programming Example

When the EtherCAT network is abnormal, you can execute DFB_ECATReset to reset the network. Set M10 to True and confirm if M11 changes to True, indicating the completion of the network reset process.

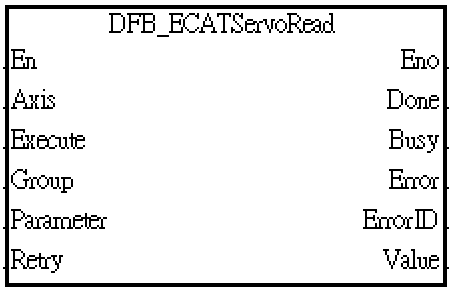


Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_ECATServoRead

FB/FC	Description
FB	DFB_ECATServoRead reads the values of parameters from the Delta servo drive specified on an EtherCAT network.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True. The value of the specified parameter will then be read from the servo drive.	BOOL	True/False (False)	-
Group	Group number (Please refer to the user manual of the applied Delta servo drive for more information.)	WORD	0~9 (0)	When *Execute* shifts to True and *Busy* is False.
Parameter	Parameter number (Please refer to the user manual of the applied Delta servo drive for more information.)	WORD	0~99 (0)	When *Execute* shifts to True and *Busy* is False.
Retry	Number of times for auto-retry when an error occured on reading the parameter values.	WORD	0~65535 (0)	When *Execute* shifts to True and *Busy* is False.

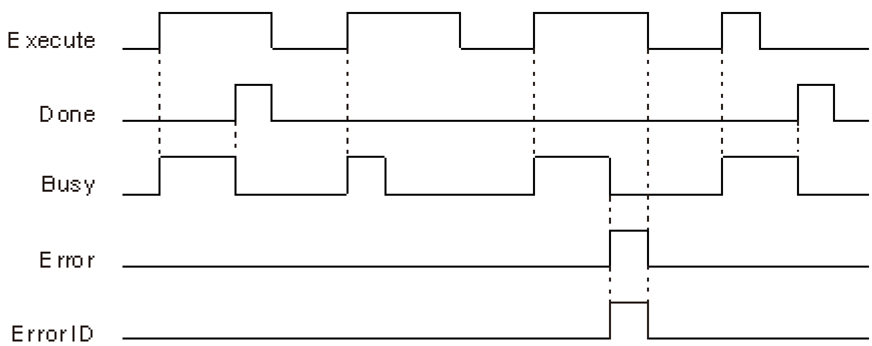
● Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the specified data is read.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

■ **Outputs Update Timing**

Name	Timing for changing to True	Timing for changing to False
Done	● When the value of the specified parameter is read.	● When *Execute* shifts from True to False. ● If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	● When *Execute* changes to True.	● When *Done* shifts to True. ● When *Error* shifts to True.
Error/ErrorID	● When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	● When *Execute* shifts from True to False. (error code is cleared)

■ **Timing Diagram**



● In-Outs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Axis	Motion axis number	WORD	K1~Kn* (0)	When *Execute* is rising edge triggered and *Busy* is False.

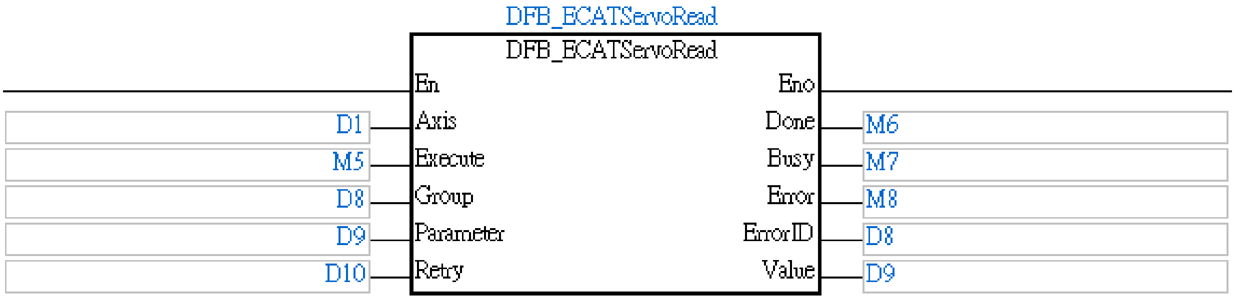
***Note**: Kn refers to the maximum supported axes of different models. For example, AH20EMC-5A supports K1~K32.

● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

The example uses DFB_ECATServoRead to read the values of the specified parameters from the Delta servo drive. For details of the servo parameters, refer to the manuals of the Delta servo drive.



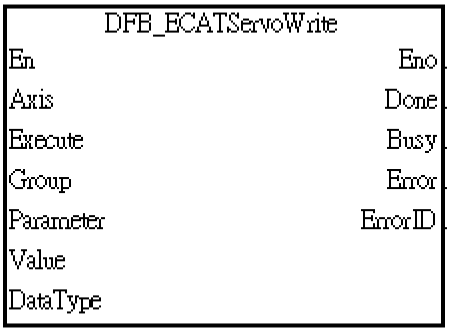
1. If you want to read the value of parameter P1-44, specifiy 1 to D8 (*Group*), 44 to D9 (*Parameter*) and the desired retry times to D10 (*Retry*).
2. Set M5 (*Execute*) to True to read the designated parameter and confirm the read value in D9 (*Value*).

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_ECATServoWrite

FB/FC	Description
FB	DFB_ECATServoWrite writes the values of parameters into the Delta servo drive specified on an EtherCAT network.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True. The value of the specified parameter will then be written into the servo drive.	BOOL	True/False (False)	-
Group	Group number (Please refer to the user manual of the applied Delta servo drive for more information.)	WORD	0~9 (0)	When *Execute* shifts to True and *Busy* is False.
Parameter	Parameter number (Please refer to the user manual of the applied Delta servo drive for more information.)	WORD	0~99 (0)	When Execute shifts to True and *Busy* is False.
Value	Value of a parameter (Please refer to the user manual of the applied Delta servo drive for more information.)	DWORD	16#0~16#FFFFFFFF (0)	When Execute shifts to True and *Busy* is False.
DataType	Data length	WORD	0: mc16bits: 0 1: mc32bits: 1	When Execute shifts to True and Busy is False.
Retry	Number of times for auto-retry when an error occured on reading the parameter values.	WORD	0~65535 (0)	When *Execute* shifts to True and *Busy* is False.

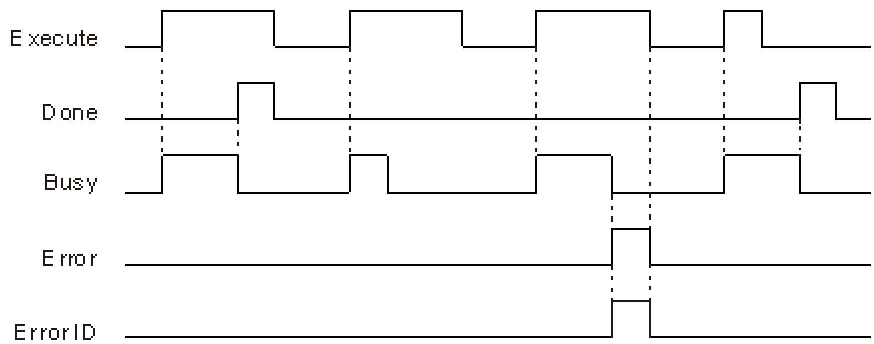
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the specified data is read.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	WORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for changing to True	Timing for changing to False
Done	• When the value of the specified parameter is written.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	• When *Execute* shifts from True to False. (error code is cleared)

Timing Diagram

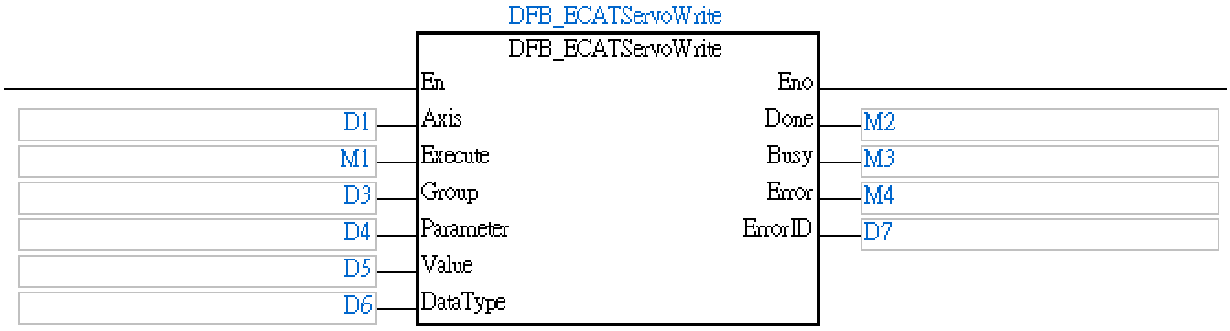


Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Programming Example

The example uses DFB_ECATServoWrite to write the values of the specified parameters to the Delta servo drive. For details of the servo drive parameters, refer to the manuals of the Delta servo drive.



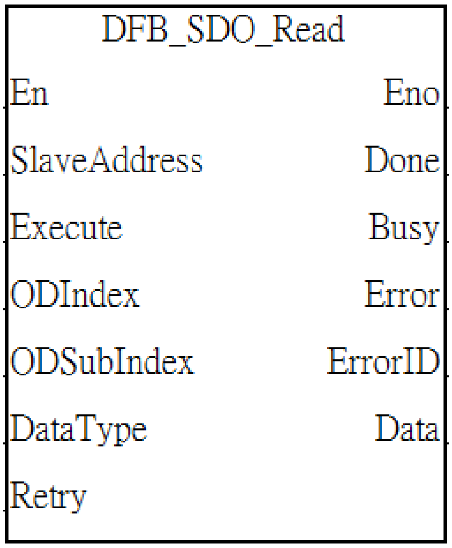
1. If you want to write 128 to the parameter P1-44, specifiy 1 to D3 (*Group*), 44 to D4 (*Parameter*), 128 to D5 (*Value*) and 0 to D6 (*Data Type*).
2. Set M1 (*Execute*) to True to write the value and confirm if M2 (*Done*) changes to True, indicating the completion of the data writing process.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_SDO_Read

FB/FC	Description
FB	D DFB_SDO_Read reads the values of parameters from the specified OD of the EtherCAT Slave via SDO.



● Inputs

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True. The value of the specified parameter will then be read.	BOOL	True/False (False)	-
SlaveAddress	ECAT Slave ID (Refer to manuals of the device that is used as slave)	WORD	1~9999 (0)	When *Execute* shifts to True and *Busy* is False.
ODIndex	ECAT Slave ODIndex (Refer to manuals of the device that is used as slave)	WORD	Refer to manuals of the device that is used as slave (0)	When *Execute* shifts to True and *Busy* is False.
ODSubIndex	ECAT Slave ODSubIndex (Refer to manuals of the device that is used as slave)	WORD	Refer to manuals of the device that is used as slave (0)	When *Execute* shifts to True and *Busy* is False.
DataType	Bit lenghth of the data to be written in OD	WORD	3: UINT08bits 0: UINT16bits 1: UINT32bits 2: UINT64bits (reserved) 7: SINT08bits 4: SINT16bits 5: SINT32bits 6: SINT64bits (reserved)	When *Execute* shifts to True and *Busy* is False.

			8: FLOAT32bits (reserved) 9: FLOAT64bits (reserved) (0)	
Retry	Number of times for auto-retry when an error occured on reading the parameter values.	INT	0~100 (0)	When *Execute* shifts to True and *Busy* is False.

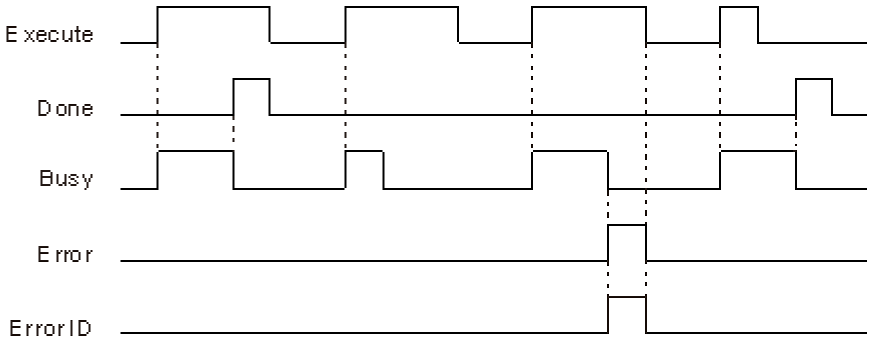
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the specified data is read.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)
Data	Data to be read	LREAL	K-2147483648~ K2147483647 (0)

Outputs Update Timing

Name	Timing for changing to True	Timing for changing to False
Done	• When the value of the specified parameter is read.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	• When *Execute* shifts from True to False. (error code is cleared)

■ **Timing Diagram**

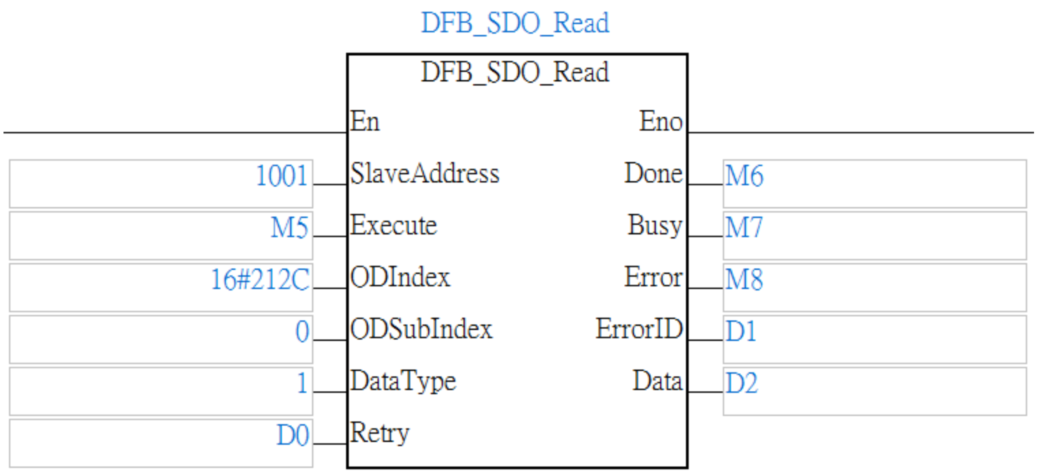


● Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

● Programming Example

The example uses DFB_SDO_Read to read the values of the specified parameters from the slave 1001 (Delta ECAT servo drive). For details of the servo drive parameters and the contents of mapping OD, refer to the relative Delta manuals.



1. If you want to read the value of parameter P1-44, specifiy 0x212C to the filed of the Index and 0 to the field of SubIndex and make sure if the type of 0x212C is UDINT in the table of CoE Object-Dictionary in ECAT Builder. The DataType shuld be 1 and then specify a desired retry times to D0 (*Retry*).

CoE Object-Dictionary

Index	Name	Type
0x212A	DRV's Parameter P1-42	UINT
0x212B	DRV's Parameter P1-43	UINT
0x212C	DRV's Parameter P1-44	UDINT
0x212D	DRV's Parameter P1-45	UDINT
0x212E	DRV's Parameter P1-46	UDINT
0x212F	DRV's Parameter P1-47	UINT
0x2130	DRV's Parameter P1-48	UINT
0x2131	DRV's Parameter P1-49	UINT
0x2132	DRV's Parameter P1-50	UINT
0x2133	DRV's Parameter P1-51	UINT
0x2134	DRV's Parameter P1-52	UINT
0x2135	DRV's Parameter P1-53	UINT
0x2136	DRV's Parameter P1-54	UDINT
0x2137	DRV's Parameter P1-55	UINT
0x2138	DRV's Parameter P1-56	UINT
0x2139	DRV's Parameter P1-57	UINT
0x213A	DRV's Parameter P1-58	UINT
0x213B	DRV's Parameter P1-59	UINT

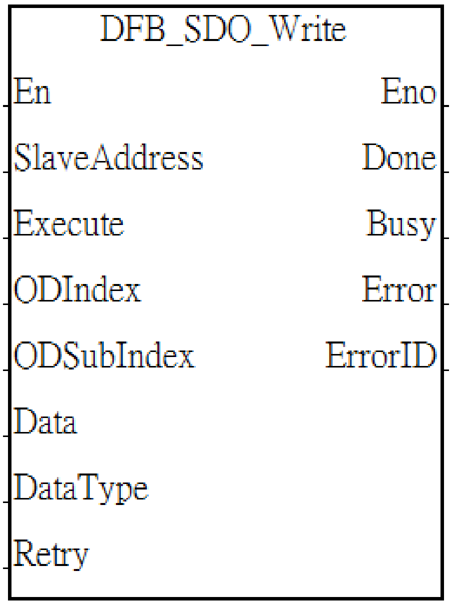
2. Set M5 (*Execute*) to True to read the OD data from the designated slave and when M6 (*Done*) changes to True, it indicates reading is complete; confirm the read value in D2 (*Value*).

● Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

DFB_SDO_Write

FB/FC	Description
FB	DFB_SDO_Write writes the values of parameters into the specified OD of the EtherCAT Slave via SDO.



- **Inputs**

Name	Function	Data type	Setting value (Default value)	Timing for updating
Execute	Executes the instruction when *Execute* changes to True. The value of the specified parameter will then be written.	BOOL	True/False (False)	-
SlaveAddress	ECAT Slave ID (Refer to manuals of the device that is used as slave)	WORD	1~9999 (0)	When *Execute* shifts to True and *Busy* is False.
ODIndex	ECAT Slave ODIndex (Refer to manuals of the device that is used as slave)	WORD	Refer to manuals of the device that is used as slave (0)	When *Execute* shifts to True and *Busy* is False.
ODSubIndex	ECAT Slave ODSubIndex (Refer to manuals of the device that is used as slave)	WORD	Refer to manuals of the device that is used as slave (0)	When *Execute* shifts to True and *Busy* is False.
Data	Data to be written in OD	LREAL	K-2147483648~ K2147483647 (0)	When *Execute* shifts to True and *Busy* is False.
DataType	Bit lenghth of the data to be written in OD		3: UINT08bits 0: UINT16bits 1: UINT32bits 2: UINT64bits	When *Execute* shifts to True and *Busy* is False.

			(reserved) 7: SINT08bits 4: SINT16bits 5: SINT32bits 6: SINT64bits (reserved) 8: FLOAT32bits (reserved) 9: FLOAT64bits (reserved) (0)	
Retry	Number of times for auto-retry when an error occured on writing the parameter values.	INT	0~100 (0)	When *Execute* shifts to True and *Busy* is False.

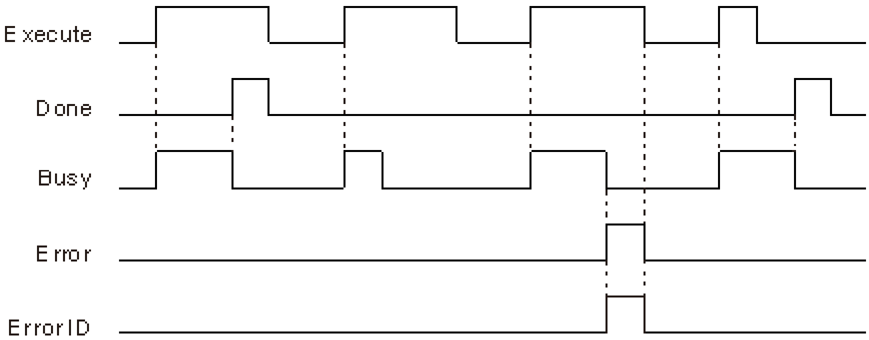
Outputs

Name	Function	Data type	Output range (Default value)
Done	True when the specified data is read.	BOOL	True/False (False)
Busy	True when the instruction is executed.	BOOL	True/False (False)
Error	True if an error occurs.	BOOL	True/False (False)
ErrorID	Indicates the error code if an error occurs. Refer to **Appendices** for error code descriptions.	DWORD	16#0~16#FFFFFFFF (0)

Outputs Update Timing

Name	Timing for changing to True	Timing for changing to False
Done	• When the value of the specified parameter is written.	• When *Execute* shifts from True to False. • If *Execute* is False and *Done* shifts to True, *Done* will be True for only one period and immediately shift to False.
Busy	• When *Execute* changes to True.	• When *Done* shifts to True. • When *Error* shifts to True.
Error/ErrorID	• When an error occurs in the execution conditions or input values for the instruction. (error code is recorded)	• When *Execute* shifts from True to False. (error code is cleared)

■ **Timing Diagram**

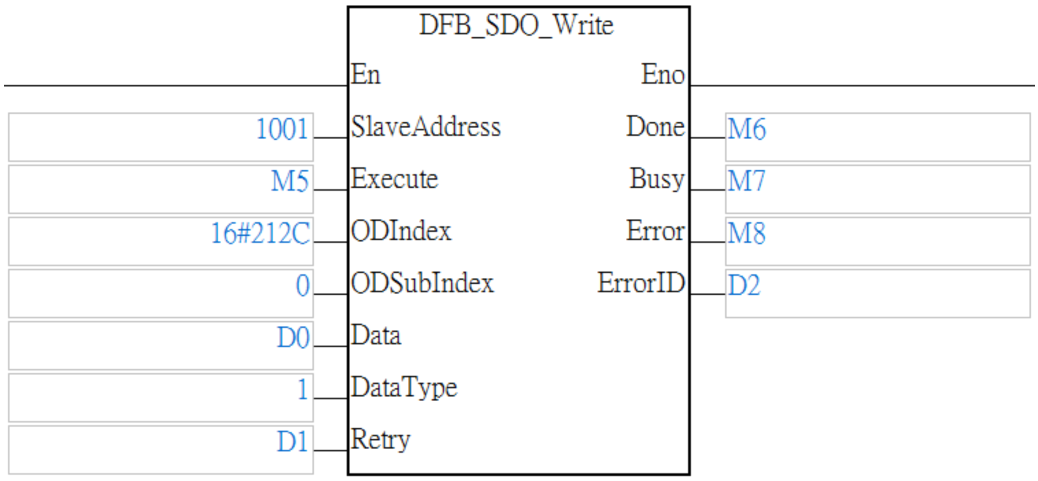


Troubleshooting

- If an error occurs during the execution of the instruction or when the axis is in the state of Errorstop, Error will change to True and the axis will stop moving. You can refer to ErrorID (Error Code) to address the problem.
- Information regarding error codes and indicators and the associated troubleshooting information are attached as Appendices for a quick reference. For the complete troubleshooting of the system, refer to ***AH Motion Controller – Operation Manual***.

Programming Example

The example uses DFB_SDO_Write to write the values of the specified parameters in the slave 1001 (Delta ECAT servo drive). For details of the servo drive parameters and the contents of mapping OD, refer to the relative Delta manuals.



1. If you want to write 128 to the parameter P1-44, specifiy 0x212C to the filed of the Index and 0 to the field of SubIndex and make sure if the type of 0x212C is UDINT in the table of CoE Object-Dictionary in ECAT Builder. The DataType shuld be 1 and then specify a desired retry times to D1 (Retry).
2. Set M5 (Execute) to True to read the OD data from the designated slave and when M6 (Done) changes to True, it indicates writing is complete.

Supported Products

- AH Motion Controller CPU: AHxxEMC-5A
- AH series motion control modules: AHxxEMC-5A

A

Appendices

Table of Contents

A.1. Table of Data Type Unit(DUT): Enum

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_OUTTYPE	0: UD 1: PD 2: AB	Setting pulse output type 0: clockwise / counterclockwise pulse output(counting up/down) 1: Pulse+Direction 2: A/B-phase	DFB_AxisSetting2 Interface: *OutputType*
eDFB_UNIT	0: Motor 1: Machine 2: Compound	Unit setting of the coordinate system 0: motot unit 1: mechanical system 2: compound unit	FB: DFB_AxisSetting2 Interface: *Unit*
eDFB_MODE	4096: AxisIdle 256: AxisStopping 4353: AbsSeg1 4354: RelSeg1 4355: AbsSeg2 4356: RelSeg2 4357: TrSeg1 4358: Jog 4359: Mpg 4362: GearIn 4363: CamIn 4608: GcodeStopping 4609: GcodeRun 4864: InterpolationStopping	0x000: axis indling 0x100: axis stopping 0x101: absolute single-speed motion 0x102: relative single-speed motion 0x103: absolute two-speed motion 0x104: relative two-speed motion 0x105: triggering single-speed motion 0x107: Jog motion 0x108: manual pulse generator 0x10A: electronic gear 0x10B: electronic cam 0x200: G-code stopping 0x201: G-code running 0x300: interpolation stopping	DFB_AxisStatus Interface: *Mode*
eDFB_SDODataType	0: mc16bits 1: mc32bits	0: writing in 16-bit data 1: writing in 32-bit data	DFB_ECATServoWrite Interface: *DataType*

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_SELECT_DEV	0: M_DEV 5: D_DEV 6: W_DEV 7: ALL	0: reading M devices from SD card 5: reading D devices from SD card 6: reading W devices from SD card 7: reading (M/D/W) devices from SD card	DFB_SDDevRead Interface: *Device*
eDFB_ENGAGE_TYPE	0: ByCapture0 1: ByCapture1 2: ByCapture2 3: ByCapture3 4: ByCapture4 5: ByCapture5 6: ByCapture6 7: ByCapture7 -1: Direct	0: engaging when Capture 0 is triggered 1: engaging when Capture 1 is triggered 2: engaging when Capture 2 is triggered 3: engaging when Capture 3 is triggered 4: engaging when Capture 4 is triggered 5: engaging when Capture 5 is triggered 6: engaging when Capture 6 is triggered 7: engaging when Capture 7 is triggered -1: engaging directly	DFB_GearIn2/DFB_CamIn2 Interface: *extTrgCAPno*
eDFB_ACC_CURVE	0: Polynomial_0order 1: Polynomial_1order 2: SingleHypot 3: Cycloid	Acceleration curve type 0: 0-order polynomial (constant) curve 1: 1st order polynomial curve 2: single hypotenuse curve 3: cycloid curve	DFB_CamCurve/ DFB_CamCurve2/ DFB_FlyCut2/ DFB_HorizontalFlowWrapper Interface: *AccCurve*

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_GEN_CURVE	0: leftCAM 5: rightCAM 1: midCAMall 9: midCAMzero 7: midCAMbegin 8: midCAMend	Cam curve type 0: left cam 5: right cam 1: middle cam 9: middle cam zero 7: middle cam begins 8: middle cam ends	DFB_CamCurve / DFB_CamCurve2/ DFB_FlyCut2 Interface: *eCamCurve*
eDFB_HCNT	0: AC0 1: AC4 2: AC8 3: AC12 4: AC16 6: AC20	High speed counters for motion control 0: high speed counter 1 1: high speed counter 2 2: high speed counter 3 3: high speed counter 4 4: high speed counter 5 6: high speed counter 6	DFB_HCnt Interface: *Channel*
eDFB_HCNT_INTYPE	0: UD 1: PD 2: AB 3: AB4	Setting pulse input type 0: clockwise / counterclockwise pulse output(counting up/down) 1: Pulse+Direction 2: A/B-phase 3: 4A/B-phase	DFB_HCnt Interface: *InputType*
eDFB_HTMR	0: AC0 1: AC4 2: AC8 3: AC12	High speed timers for motion control 0: high speed timer 1 1: high speed timer 2 2: high speed timer 3 3: high speed timer 4	DFB_HTmr Interface: *Channel*

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_COMP	0: Ch0 1: Ch1 2: Ch2 3: Ch3 4: Ch4 5: Ch5 6: Ch6 7: Ch7	0: channel 0 1: channel 1 2: channel 2 3: channel 3 4: channel 4 5: channel 5 6: channel 6 7: channel 7	DFB_Compare Interface: *Channel*
eDFB_COMP_SOURCE	0: Axis1 1: Axis2 2: Axis3 3: Axis4 4: AC0 5: AC4 6: AC8 7: AC12 8: AC16	0: Axis 1 1: Axis 2 2: Axis 3 3: Axis 4 4: high speed counter 1 5: high speed counter 2 6: high speed counter 3 7: high speed counter 4 8: high speed counter 5	DFB_Compare Interface: *Source*
eDFB_COMP_MODE	0: Equal 1: Bigger_Equal 2: Smaller_Equal	0: equal 1: bigger or equal 2: smaller or equal	DFB_Compare Interface: *Mode*
eDFB_COMP_OUTDEV	0: SetY08 1: SetY09 2: SetY10 3: SetY11 4: RstAC0 5: RstAC4 6: RstAC8 7: RstAC12	0: set Y0.8 1: set Y0.9 2: set Y0.10 3: set Y0.11 4: reset the value of high speed counter 1 5: reset the value of high speed counter 2 6: reset the value of high speed counter 3 7: reset the value of high speed counter 4	DFB_Compare Interface: *OutPutDevice*

A1

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_CAP	0: Ch0 1: Ch1 2: Ch2 3: Ch3 4: Ch4 5: Ch5 6: Ch6 7: Ch7	0: channel 0 1: channel 1 2: channel 2 3: channel 3 4: channel 4 5: channel 5 6: channel 6 7: channel 7	DFB_Capture/ DFB_Capture2 Interface: *Source*
eDFB_CAP_TRIG_DEV	0: X0p0 1: X0p1 2: X0p2 3: X0p3 8: X0p8 9: X0p9 10: X0p11 11: X0p11 12: X0p12 13: X0p13 14: X0p14	0: trigger by X0.0 signal 1: trigger byX0.1 signal 2: trigger by X0.2 signal 3: trigger by X0.3 signal 8: trigger by X0.8 signal 9: trigger by X0.9 signal 10: trigger by X0.10 signal 11: trigger by X0.11 signal 12: trigger by X0.12 signal 13: trigger by X0.13 signal 14: trigger by X0.14 signal	DFB_Capture/ DFB_Capture2 Interface: *TriggerDevice*
eDFB_CAP_SOURCE	0: Axis1 1: Axis2 2: Axis3 3: Axis4 4: AC0 5: AC4 6: AC8 7: AC12 8: AC16	0: capture axis 1 1: capture axis 2 2: capture axis 3 3: capture axis 4 4: capture high speed counter 1 5: capture high speed counter 2 6: capture high speed counter 3 7: capture high speed counter 4 8: capture high speed counter 5	DFB_Capture/ DFB_Capture2 Interface: *Source*

Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eDFB_HALT_CLK_SOURCE	0: slaveEOP 1: masterEOP 2: extern	0: end point of slave cam 1: end point of master cam 2: external input of the function block	DFB_FlyCut2 Interface: *Halt_ClkSource*
eMC_STATE_MACHINE	0: Unknown 1: ErrorStop 2: Disabled 3: Standstill 4: Homing 5: Stopping 6: ContinuousMotion 7: SynchronizedMotion 8: DiscreteMotion 9: Coordinated 10: CoordinatedHalt 11: CoordinatedStop	0: Unknown 1: ErrorStop 2: Disabled 3: Standstill 4: Homing 5: Stopping 6: ContinuousMotion 7: SynchronizedMotion 8: DiscreteMotion 9: Coordinated 10: CoordinatedHalt 11: CoordinatedStop	-
eMC_GROUP_STATE_MACHINE	0: GroupDisable 256: GroupStandby 512: GroupStopping 576: GroupMotion 768: GroupErrorStop	0: GroupDisable 256: GroupStandby 512: GroupStopping 576: GroupMotion 768: GroupErrorStop	
eMC_BUFFER_MODE	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0: aborting the ongoing motion 1: buffering when ongoing motion is done 2: blending with the lowest velocity 3: blending with the velocity of the previous motion 4: blending with the velocity of the next motion 5: blending with the highest velocity	Interface: *BufferMode*

A1

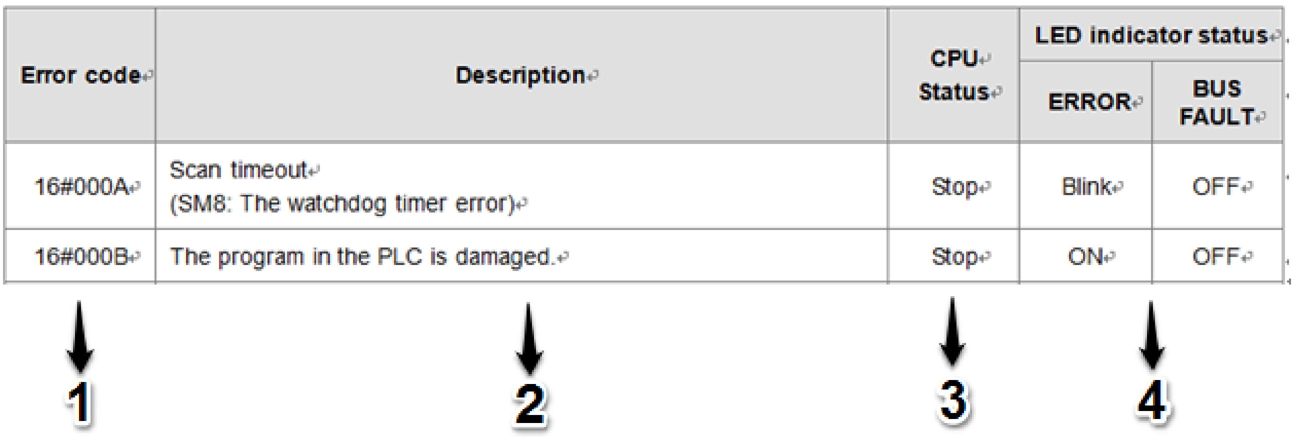
Data Type	Value	Description	Applicable Function Block Instruction and its Interface
eMC_DIRECTION	1: mcPositiveDirection 2: mcShortestWay 3: mcNegativeDirection 4: mcCurrentDirection	1: positive direction 2: shortest way 3: negative direction 4: current direction	Interface: *Direction*
eMC_SOURCE	0: mcCommandedValue 1: mcSetValue 2: mcActualValue	0: command value 1: set value 2: actual value	MC_ReadMotionState Interface: *Source* MC_CamIn/ MC_GearIn/ MC_GearInPos Interface: *MasterValueSource* MC_CombineAxes Interface: *MasterValueSourceM1/ MasterValueSourceM2* MC_DigitalCamSwitch Interface: *ValueSource*
eMC_SYNC_MODE	1: mcRampIn_Shortest 2: mcRampIn_Positive 3: mcRampIn_Negative	1: (reserved) 2: (reserved) 3: (reserved)	MC_GearInPos Interface: *SyncMode*
eMC_START_MODE	0: mcJump 1: mcRampIn_Shortest 2: mcRampIn_Positive 3: mcRampIn_Negative 4: mcAbsolute 5: mcRelative	0: jump in immediately 1: shortest path 2: positinve path 3: negative path 4: (reserved) 5: (reserved)	MC_CamIn Interface: *StartMode*
eMC_COMBINE_MODE	0: mcAddAxes 1: mcSubAxes	0: adding 1: subtracting	MC_CombineAxes Interface: *CombineMode*
eMC_SERVOOFF_MODE	0: mcAborting 1: mcBuffered	0: Aborting 1: mcBuffered	MC_Power Interface: *Mode*

A.2. Error Codes and Troubleshooting

When an error occurs, you can address the problem by the error codes and indicators and find out the corrective actions for troubleshooting the error. For detailed troubleshooting procedures, refer to ***AH Motion Controller– Operation Manual***.

A.2.1. Error Codes and Indicators

- **Columns**



Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#000A	Scan timeout (SM8: The watchdog timer error)	Stop	Blink	OFF
16#000B	The program in the PLC is damaged.	Stop	ON	OFF

Items provided in the table		
1	**Error code**	If the error occurs in the system, the error code is generated
2	**Description**	The description of the error
3	**CPU status**	If the error occurs, the CPU stops running, keeps running, or in the status defined by users. **Stop**: The CPU stops running when the error occurs. **Keep**: The CPU keeps running when the error occurs. **Self-defined**: The status of the CPU can be defined by users. Please refer to section 8.2.1 in Operation Manual for more information.
4	**LED indicator status**	LED indicator status: If the error occurs, the LED indicator is ON, OFF, or Blinking. **RUN:** Operating status of the CPU **ERROR**: Error status of the CPU **BUS FAULT**: Error status of the I/O bus **SYSTEM**: System status of the CPU

- **LED indicators**

The AH Motion CPU can function as a motion CPU or a motion module. The effective LED indicators are different according to the applications of AH Motion CPU, either in CPU mode or in Module mode:

Mode	LED indicator	Description

CPU	RUN	Operating status of the CPU ON: The user program is being executed. OFF: The execution of the user program stops. Blinking: The CPU runs in debug mode.
	ERROR	Error status of the CPU ON: A serious error occurs in the CPU. OFF: The system is normal. Blinking: A slight error occurs in the CPU.
	BUS FAULT	Error status of the I/O bus ON: A serious error occurs in the I/O bus. OFF: The I/O bus is normal. Blinking: A slight error occurs in the I/O bus.
	SYSTEM	System status of the CPU module ON: The external input/output is forced ON/OFF. OFF: The system is in the default status. Blinking: The CPU module is being reset./The retained values in the devices are being cleared .
Module	RUN	Operating status of the motion CPU functioning as a motion module ON: The user program is being executed. OFF: The execution of the user program stops. Blinking: The motion module runs in debug mode.
	ERROR	Error status of the motion CPU functioning as a motion module ON: A serious error occurs in the module. OFF: The system is normal. Blinking: A slight error occurs in the module.

AHxxEMC-5A

After a program is written into an AH Motion series CPU, the ERROR LED indicator will blink and an error flag will be ON if an error occurs in main program or amotion subroutine. The reason for the error occurring in the main program or amotion subroutine may be that the use of operands (devices) is incorrect, syntax is incorrect, or the setting of motion parameters is incorrect. You can know the reasons for the errors occurring in an AH Motion series CPU by means of the error codes (hexadecimal codes) stored in error registers.

■ **Error flags and registers:**

SM*: Special auxiliary relay SR*: Special data register	Program error	Motion error
	POU	mn=10~41 (10: 1st axis; 41: 32nd axis)
Error flag	-	AMmn49
Operation error	SM0	-
The operation error is locked	SM1	-
Syntax (Instruction/Operand) check error	SM5	-
Operation error code	SR0	-
Operation error address (step)	SR1/SR2	-
Syntax check error code	SR4	ARmn41
Syntax check error address (step)	SR5/SR6	-

Note**: you can refer to ***AH Motion Controller– Operation Manual for the detailed explanation of SM and SR.

■ **Error codes and indicators**

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#000A	Scan timeout (SM8: The watchdog timer error)	Stop	Blinking	Keep
16#000B	The program in the PLC is damaged.	Stop	ON	Keep
16#000C	The program downloaded to the PLC is incorrect.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#000D	The CPU parameter is damaged.	Stop	ON	Keep
16#000E	The program or the parameter is being downloaded, and therefore the PLC can not run.	Stop	Blinking	Keep
16#000F	The original program in the PLC is damaged.	Continue	Keep	Keep
16#0010	The access to the memory in the CPU is denied.	Stop	ON	Keep
16#0011	The PLC ID is incorrect. (SM9)	Continue	ON	Keep
16#0012	The PLC password is incorrect.	Continue	ON	Keep
16#0013	The I/O module can not run/stop. (SM10)	Stop	Keep	ON
16#0014	The procedure of restoring the system can not be executed. (SM9)	Stop	ON	ON
16#0015	The module table is incorrect. (SM10)	Stop	ON	Keep
16#0016	The module setting is incorrect. (SM10)	Stop	ON	Keep
16#0017	The device which is associated with the data register is incorrect. (SM10)	Stop	ON	Keep
16#0018	The serial port is abnormal. (SM9)	Continue	Blinking	Keep
16#0019	The USB is abnormal. (SM9)	Continue	Blinking	Keep
16#001A	The contents of the system backup file (.dup file) are incorrect.	Continue	Blinking	Keep
16#001B	Timed interrupt 0 is set incorrectly.	Stop	ON	Keep
16#001C	Timed interrupt 1 is set incorrectly.	Stop	ON	Keep
16#001D	Timed interrupt 2 is set incorrectly.	Stop	ON	Keep
16#001E	Timed interrupt 3 is set incorrectly.	Stop	ON	Keep
16#001F	The watchdog timer is set incorrectly.	Stop	ON	Keep
16#0020	The setting of the fixed scan time is incorrect.	Stop	ON	Keep
16#0021	The setting of the fixed scan time is incorrect.	Stop	ON	Keep
16#0022	The CPU parameter downloaded to the PLC is incorrect.	Stop	ON	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#0023	The Y state (STOP->RUN) section in the PLC Parameter Setting window is set incorrectly.	Stop	ON	Keep
16#0024	There is no IO module on the backplane.	Continue	Keep	Keep
16#0026	The Communication Ratio box in the Communication Loading of Scan Time (%) section in the PLC Parameter Setting window is set incorrectly.	Stop	ON	Keep
16#0027	The latching auxiliary relay range which is set is incorrect.	Stop	ON	Keep
16#0028	The latching data register range which is set is incorrect.	Stop	ON	Keep
16#0029	The latching timer range which is set is incorrect.	Stop	ON	Keep
16#002A	The latching counter range which is set is incorrect.	Stop	ON	Keep
16#002B	The latching 32-bit counter range which is set is incorrect.	Stop	ON	Keep
16#0033	The communication setting of COM1 is incorrect. (SM9)	Continue	Blinking	Keep
16#0034	The setting of the station address of COM1 is incorrect. (SM9)	Continue	Blinking	Keep
16#0035	The setting of the communication type of COM1 is incorrect. (SM9)	Continue	Blinking	Keep
16#0038	The communication setting of COM2 is incorrect. (SM9)	Continue	Blinking	Keep
16#0039	The setting of the station address of COM2 is incorrect. (SM9)	Continue	Blinking	Keep
16#003A	The setting of the communication type of COM2 is incorrect. (SM9)	Continue	Blinking	Keep
16#0050	The memories in the latched special auxiliary relays are abnormal.	Stop	ON	Keep
16#0051	The latched special data registers are abnormal.	Stop	ON	Keep
16#0052	The memories in the latched auxiliary relays are abnormal.	Stop	ON	Keep
16#0053	The latched timers are abnormal.	Stop	ON	Keep
16#0054	The latched counters are abnormal.	Stop	ON	Keep
16#0055	The latched 32-bit counters are abnormal.	Stop	ON	Keep
16#0056	The memories in the latched timers are abnormal.	Stop	ON	Keep
16#0057	The memories in the latched counters are abnormal.	Stop	ON	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#0058	The memories in the latched 32-bit counters are abnormal.	Stop	ON	Keep
16#0059	The latched data registers are abnormal.	Stop	ON	Keep
16#005A	The latched working registers are abnormal.	Stop	ON	Keep
16#005E	The memory card is initialized incorrectly. (SM453)	Continue	Blinking	Keep
16#005F	The data is read from the inexistent file in the memory card, or the data is written into the inexistent file in the memory card. (SM453)	Continue	Blinking	Keep
16#0061	The capacity of the memory card is not large enough. (SM453)	Continue	Blinking	Keep
16#0062	The memory card is write protected. (SM453)	Continue	Blinking	Keep
16#0063	An error occurs when the data is written into the memory card. (SM453)	Continue	Blinking	Keep
16#0064	The file in the memory card can not be read. (SM453)	Continue	Blinking	Keep
16#0065	The file in the memory card is a read-only file. (SM453)	Continue	Blinking	Keep
16#0066	An error occurs when the system is backupped.	Continue	Blinking	Keep
16#0067	The length of the restored system data exceeds the system data length of CPU module	Continue	Blinking	Keep
16#1401	An error occurs when the data in the I/O module is accessed. (SM9)	Stop	Keep	ON
16#1402	The actual arrangement of the I/O modules is not consistent with the module table. (SM9)	Stop	Keep	ON
16#1403	An error occurs when the data is read from the module. (SM9)	Stop	Keep	ON
16#1405	The setting parameter of the module is not found. (SM9)	Stop	Keep	ON
16#140B	The number of network modules exceeds the limit. (SM9)	Stop	Keep	ON
16#140C	The checksum of the high-speed data exchange is incorrect.	Stop	Keep	ON
16#140D	The ID of the actual power supply module is not the same as the ID of the power supply module set in HWCONFIG. (SM9)	Stop	Keep	ON
16#140E	The amount of data exchanged at a high speed exceeds the maximum amount supported.	Stop	Keep	ON

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#140F	High-speed data exchange error	Stop	Keep	ON
16#1801	There is no interrupt service routine in the CPU module.	Continue	Keep	Keep
16#2000	There is no END in the program in the PLC. (SM5)	Stop	Blinking	Keep
16#2001	The program is incorrect. There is a syntax error.	Stop	Blinking	Keep
16#2002	GOEND is used incorrectly. (SM5)	Stop	Blinking	Keep
16#2003	The devices used in the program exceed the range. (SM0/SM5)	Self-defined	Blinking	Keep
16#2004	The part of the program specified by the label used in CJ/JMP is incorrect, or the label is used repeatedly. (SM0/SM5)	Stop	Blinking	Keep
16#2005	The N value used in MC is not the same as the corresponding N value used in MCR, or the number of N values used in MC is not the same as the number of N values used in MCR. (SM5)	Stop	Blinking	Keep
16#2006	The N values used in MC do not start from 0, or the N values used in MC are not continuous. (SM5)	Stop	Blinking	Keep
16#2007	The operands used in ZRST are not used properly. (SM5)	Stop	Blinking	Keep
16#200A	Invalid instruction (SM5)	Stop	Blinking	Keep
16#200B	The operand **n** or the other constant operands exceed the range. (SM0/SM5)	Self-defined	Blinking	Keep
16#200C	The operands overlap. (SM0/SM5)	Self-defined	Blinking	Keep
16#200D	An error occurs when the binary number is converted into the binary-coded decimal number. (SM0/SM5)	Self-defined	Blinking	Keep
16#200E	The string does not end with 0x00. (SM0/SM5)	Self-defined	Blinking	Keep
16#200F	The instruction does not support the modification by an index register. (SM5)	Stop	Blinking	Keep
16#2010	1. The instruction does not support the device. 2. Encoding error 3. The instruction is a 16-bit instruction, but the constant operand is a 32-bit code. (SM5)	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2011	The number of operands is incorrect. (SM5)	Stop	Blinking	Keep
16#2012	Incorrect division operation (SM0/SM5).	Self-defined	Blinking	Keep
16#2013	The value exceeds the range of values which can be represented by the floating-point numbers. (SM0/SM5)	Self-defined	Blinking	Keep
16#2014	The task designated by TKON/TKOFF is incorrect, or exceeds the range. (SM5)	Stop	Blinking	Keep
16#2015	There are more than 32 levels of nested program structures supported by CALL. (SM0)	Self-defined	Blinking	Keep
16#2016	There are more than 32 levels of nested program structures supported by FOR/NEXT. (SM0/SM5)	Self-defined	Blinking	Keep
16#2017	The number of times FOR is used is different from the number of times NEXT is used. (SM5)	Stop	Blinking	Keep
16#2018	There is a label after FEND, but there is no SRET. Or there is SRET, but there is no label. (SM5)	Stop	Blinking	Keep
16#2019	The interrupt task is not after FEND. (SM5)	Stop	Blinking	Keep
16#201A	IRET/SRET is not after FEND. (SM5)	Stop	Blinking	Keep
16#201B	There is an interrupt task, but there is no IRET. There is IRET, but there is not interrupt task. (SM5)	Stop	Blinking	Keep
16#201C	End is not at the end of the program. (SM5)	Stop	Blinking	Keep
16#201D	There is CALL, but there is no MAR. (SM5)	Stop	Blinking	Keep
16#201E	The function code used in MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#201F	The length of the data set in MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#2020	The communication command received by using MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep
16#2021	The checksum of the command received by using MODRW is incorrect. (SM102/SM103)	Self-defined	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2022	The format of the command used in MODRW does not conform to the ASCII format. (SM102/SM103)	Self-defined	Blinking	Keep
16#2023	There is a communication timeout when MODRW is executed. (SM102/SM103)	Self-defined	Blinking	Keep
16#2024	The setting value of the communication timeout is invalid when RS is executed. (SM102/SM103)	Self-defined	Blinking	Keep
16#2025	There is a communication timeout when RS is executed. (SM102/SM103)	Self-defined	Blinking	Keep
16#2026	The interrupt number used in RS is incorrect.	Self-defined	Keep	Keep
16#2027	The execution of FWD is abnormal.	Self-defined	Blinking	Keep
16#2028	The execution of REV is abnormal.	Self-defined	Blinking	Keep
16#2029	The execution of STOP is abnormal.	Self-defined	Blinking	Keep
16#202A	The execution of RSDT is abnormal.	Self-defined	Blinking	Keep
16#202B	The execution of RSTEF is abnormal.	Self-defined	Blinking	Keep
16#202C	I/O interrupt service routine 0 does not exist.	Stop	Blinking	Keep
16#202D	I/O interrupt service routine 1 does not exist.	Stop	Blinking	Keep
16#202E	I/O interrupt service routine 2 does not exist.	Stop	Blinking	Keep
16#202F	I/O interrupt service routine 3 does not exist.	Stop	Blinking	Keep
16#2030	I/O interrupt service routine 4 does not exist.	Stop	Blinking	Keep
16#2031	I/O interrupt service routine 5 does not exist.	Stop	Blinking	Keep
16#2032	I/O interrupt service routine 6 does not exist.	Stop	Blinking	Keep
16#2033	I/O interrupt service routine 7 does not exist.	Stop	Blinking	Keep
16#2034	I/O interrupt service routine 8 does not exist.	Stop	Blinking	Keep
16#2035	I/O interrupt service routine 9 does not exist.	Stop	Blinking	Keep
16#2036	I/O interrupt service routine 10 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2037	I/O interrupt service routine 11 does not exist.	Stop	Blinking	Keep
16#2038	I/O interrupt service routine 12 does not exist.	Stop	Blinking	Keep
16#2039	I/O interrupt service routine 13 does not exist.	Stop	Blinking	Keep
16#203A	I/O interrupt service routine 14 does not exist.	Stop	Blinking	Keep
16#203B	I/O interrupt service routine 15 does not exist.	Stop	Blinking	Keep
16#203C	I/O interrupt service routine 16 does not exist.	Stop	Blinking	Keep
16#203D	I/O interrupt service routine 17 does not exist.	Stop	Blinking	Keep
16#203E	I/O interrupt service routine 18 does not exist.	Stop	Blinking	Keep
16#203F	I/O interrupt service routine 19 does not exist.	Stop	Blinking	Keep
16#2040	I/O interrupt service routine 20 does not exist.	Stop	Blinking	Keep
16#2041	I/O interrupt service routine 21 does not exist.	Stop	Blinking	Keep
16#2042	I/O interrupt service routine 22 does not exist.	Stop	Blinking	Keep
16#2043	I/O interrupt service routine 23 does not exist.	Stop	Blinking	Keep
16#2044	I/O interrupt service routine 24 does not exist.	Stop	Blinking	Keep
16#2045	I/O interrupt service routine 25 does not exist.	Stop	Blinking	Keep
16#2046	I/O interrupt service routine 26 does not exist.	Stop	Blinking	Keep
16#2047	I/O interrupt service routine 27 does not exist.	Stop	Blinking	Keep
16#2048	I/O interrupt service routine 28 does not exist.	Stop	Blinking	Keep
16#2049	I/O interrupt service routine 29 does not exist.	Stop	Blinking	Keep
16#204A	I/O interrupt service routine 30 does not exist.	Stop	Blinking	Keep
16#204B	I/O interrupt service routine 31 does not exist.	Stop	Blinking	Keep
16#2054	External interrupt service routine 40 does not exist.	Stop	Blinking	Keep
16#2055	External interrupt service routine 41 does not exist.	Stop	Blinking	Keep
16#2056	External interrupt service routine 42 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2057	External interrupt service routine 43 does not exist.	Stop	Blinking	Keep
16#2058	External interrupt service routine 44 does not exist.	Stop	Blinking	Keep
16#2059	External interrupt service routine 45 does not exist.	Stop	Blinking	Keep
16#205A	External interrupt service routine 46 does not exist.	Stop	Blinking	Keep
16#205B	External interrupt service routine 47 does not exist.	Stop	Blinking	Keep
16#205C	External interrupt service routine 48 does not exist.	Stop	Blinking	Keep
16#205D	External interrupt service routine 49 does not exist.	Stop	Blinking	Keep
16#205E	External interrupt service routine 50 does not exist.	Stop	Blinking	Keep
16#205F	External interrupt service routine 51 does not exist.	Stop	Blinking	Keep
16#2060	External interrupt service routine 52 does not exist.	Stop	Blinking	Keep
16#2061	External interrupt service routine 53 does not exist.	Stop	Blinking	Keep
16#2062	External interrupt service routine 54 does not exist.	Stop	Blinking	Keep
16#2063	External interrupt service routine 55 does not exist.	Stop	Blinking	Keep
16#2064	External interrupt service routine 56 does not exist.	Stop	Blinking	Keep
16#2065	External interrupt service routine 57 does not exist.	Stop	Blinking	Keep
16#2066	External interrupt service routine 58 does not exist.	Stop	Blinking	Keep
16#2067	External interrupt service routine 59 does not exist.	Stop	Blinking	Keep
16#2068	External interrupt service routine 60 does not exist.	Stop	Blinking	Keep
16#2069	External interrupt service routine 61 does not exist.	Stop	Blinking	Keep
16#206A	External interrupt service routine 62 does not exist.	Stop	Blinking	Keep
16#206B	External interrupt service routine 63 does not exist.	Stop	Blinking	Keep
16#206C	External interrupt service routine 64 does not exist.	Stop	Blinking	Keep
16#206D	External interrupt service routine 65 does not exist.	Stop	Blinking	Keep
16#206E	External interrupt service routine 66 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#206F	External interrupt service routine 67 does not exist.	Stop	Blinking	Keep
16#2070	External interrupt service routine 68 does not exist.	Stop	Blinking	Keep
16#2071	External interrupt service routine 69 does not exist.	Stop	Blinking	Keep
16#2072	External interrupt service routine 70 does not exist.	Stop	Blinking	Keep
16#2073	External interrupt service routine 71 does not exist.	Stop	Blinking	Keep
16#2074	External interrupt service routine 72 does not exist.	Stop	Blinking	Keep
16#2075	External interrupt service routine 73 does not exist.	Stop	Blinking	Keep
16#2076	External interrupt service routine 74 does not exist.	Stop	Blinking	Keep
16#2077	External interrupt service routine 75 does not exist.	Stop	Blinking	Keep
16#2078	External interrupt service routine 76 does not exist.	Stop	Blinking	Keep
16#2079	External interrupt service routine 77 does not exist.	Stop	Blinking	Keep
16#207A	External interrupt service routine 78 does not exist.	Stop	Blinking	Keep
16#207B	External interrupt service routine 79 does not exist.	Stop	Blinking	Keep
16#207C	External interrupt service routine 80 does not exist.	Stop	Blinking	Keep
16#207D	External interrupt service routine 81 does not exist.	Stop	Blinking	Keep
16#207E	External interrupt service routine 82 does not exist.	Stop	Blinking	Keep
16#207F	External interrupt service routine 83 does not exist.	Stop	Blinking	Keep
16#2080	External interrupt service routine 84 does not exist.	Stop	Blinking	Keep
16#2081	External interrupt service routine 85 does not exist.	Stop	Blinking	Keep
16#2082	External interrupt service routine 86 does not exist.	Stop	Blinking	Keep
16#2083	External interrupt service routine 87 does not exist.	Stop	Blinking	Keep
16#2084	External interrupt service routine 88 does not exist.	Stop	Blinking	Keep
16#2085	External interrupt service routine 89 does not exist.	Stop	Blinking	Keep
16#2086	External interrupt service routine 90 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2087	External interrupt service routine 91 does not exist.	Stop	Blinking	Keep
16#2088	External interrupt service routine 92 does not exist.	Stop	Blinking	Keep
16#2089	External interrupt service routine 93 does not exist.	Stop	Blinking	Keep
16#208A	External interrupt service routine 94 does not exist.	Stop	Blinking	Keep
16#208B	External interrupt service routine 95 does not exist.	Stop	Blinking	Keep
16#208C	External interrupt service routine 96 does not exist.	Stop	Blinking	Keep
16#208D	External interrupt service routine 97 does not exist.	Stop	Blinking	Keep
16#208E	External interrupt service routine 98 does not exist.	Stop	Blinking	Keep
16#208F	External interrupt service routine 99 does not exist.	Stop	Blinking	Keep
16#2090	External interrupt service routine 100 does not exist.	Stop	Blinking	Keep
16#2091	External interrupt service routine 101 does not exist.	Stop	Blinking	Keep
16#2092	External interrupt service routine 102 does not exist.	Stop	Blinking	Keep
16#2093	External interrupt service routine 103 does not exist.	Stop	Blinking	Keep
16#2094	External interrupt service routine 104 does not exist.	Stop	Blinking	Keep
16#2095	External interrupt service routine 105 does not exist.	Stop	Blinking	Keep
16#2096	External interrupt service routine 106 does not exist.	Stop	Blinking	Keep
16#2097	External interrupt service routine 107 does not exist.	Stop	Blinking	Keep
16#2098	External interrupt service routine 108 does not exist.	Stop	Blinking	Keep
16#2099	External interrupt service routine 109 does not exist.	Stop	Blinking	Keep
16#209A	External interrupt service routine 110 does not exist.	Stop	Blinking	Keep
16#209B	External interrupt service routine 111 does not exist.	Stop	Blinking	Keep
16#209C	External interrupt service routine 112 does not exist.	Stop	Blinking	Keep
16#209D	External interrupt service routine 113 does not exist.	Stop	Blinking	Keep
16#209E	External interrupt service routine 114 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#209F	External interrupt service routine 115 does not exist.	Stop	Blinking	Keep
16#20A0	External interrupt service routine 116 does not exist.	Stop	Blinking	Keep
16#20A1	External interrupt service routine 117 does not exist.	Stop	Blinking	Keep
16#20A2	External interrupt service routine 118 does not exist.	Stop	Blinking	Keep
16#20A3	External interrupt service routine 119 does not exist.	Stop	Blinking	Keep
16#20A4	External interrupt service routine 120 does not exist.	Stop	Blinking	Keep
16#20A5	External interrupt service routine 121 does not exist.	Stop	Blinking	Keep
16#20A6	External interrupt service routine 122 does not exist.	Stop	Blinking	Keep
16#20A7	External interrupt service routine 123 does not exist.	Stop	Blinking	Keep
16#20A8	External interrupt service routine 124 does not exist.	Stop	Blinking	Keep
16#20A9	External interrupt service routine 125 does not exist.	Stop	Blinking	Keep
16#20AA	External interrupt service routine 126 does not exist.	Stop	Blinking	Keep
16#20AB	External interrupt service routine 127 does not exist.	Stop	Blinking	Keep
16#20AC	External interrupt service routine 128 does not exist.	Stop	Blinking	Keep
16#20AD	External interrupt service routine 129 does not exist.	Stop	Blinking	Keep
16#20AE	External interrupt service routine 130 does not exist.	Stop	Blinking	Keep
16#20AF	External interrupt service routine 131 does not exist.	Stop	Blinking	Keep
16#20B0	External interrupt service routine 132 does not exist.	Stop	Blinking	Keep
16#20B1	External interrupt service routine 133 does not exist.	Stop	Blinking	Keep
16#20B2	External interrupt service routine 134 does not exist.	Stop	Blinking	Keep
16#20B3	External interrupt service routine 135 does not exist.	Stop	Blinking	Keep
16#20B4	External interrupt service routine 136 does not exist.	Stop	Blinking	Keep
16#20B5	External interrupt service routine 137 does not exist.	Stop	Blinking	Keep
16#20B6	External interrupt service routine 138 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20B7	External interrupt service routine 139 does not exist.	Stop	Blinking	Keep
16#20B8	External interrupt service routine 140 does not exist.	Stop	Blinking	Keep
16#20B9	External interrupt service routine 141 does not exist.	Stop	Blinking	Keep
16#20BA	External interrupt service routine 142 does not exist.	Stop	Blinking	Keep
16#20BB	External interrupt service routine 143 does not exist.	Stop	Blinking	Keep
16#20BC	External interrupt service routine 144 does not exist.	Stop	Blinking	Keep
16#20BD	External interrupt service routine 145 does not exist.	Stop	Blinking	Keep
16#20BE	External interrupt service routine 146 does not exist.	Stop	Blinking	Keep
16#20BF	External interrupt service routine 147 does not exist.	Stop	Blinking	Keep
16#20C0	External interrupt service routine 148 does not exist.	Stop	Blinking	Keep
16#20C1	External interrupt service routine 149 does not exist.	Stop	Blinking	Keep
16#20C2	External interrupt service routine 150 does not exist.	Stop	Blinking	Keep
16#20C3	External interrupt service routine 151 does not exist.	Stop	Blinking	Keep
16#20C4	External interrupt service routine 152 does not exist.	Stop	Blinking	Keep
16#20C5	External interrupt service routine 153 does not exist.	Stop	Blinking	Keep
16#20C6	External interrupt service routine 154 does not exist.	Stop	Blinking	Keep
16#20C7	External interrupt service routine 155 does not exist.	Stop	Blinking	Keep
16#20C8	External interrupt service routine 156 does not exist.	Stop	Blinking	Keep
16#20C9	External interrupt service routine 157 does not exist.	Stop	Blinking	Keep
16#20CA	External interrupt service routine 158 does not exist.	Stop	Blinking	Keep
16#20CB	External interrupt service routine 159 does not exist.	Stop	Blinking	Keep
16#20CC	External interrupt service routine 160 does not exist.	Stop	Blinking	Keep
16#20CD	External interrupt service routine 161 does not exist.	Stop	Blinking	Keep
16#20CE	External interrupt service routine 162 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20CF	External interrupt service routine 163 does not exist.	Stop	Blinking	Keep
16#20D0	External interrupt service routine 164 does not exist.	Stop	Blinking	Keep
16#20D1	External interrupt service routine 165 does not exist.	Stop	Blinking	Keep
16#20D2	External interrupt service routine 166 does not exist.	Stop	Blinking	Keep
16#20D3	External interrupt service routine 167 does not exist.	Stop	Blinking	Keep
16#20D4	External interrupt service routine 168 does not exist.	Stop	Blinking	Keep
16#20D5	External interrupt service routine 169 does not exist.	Stop	Blinking	Keep
16#20D6	External interrupt service routine 170 does not exist.	Stop	Blinking	Keep
16#20D7	External interrupt service routine 171 does not exist.	Stop	Blinking	Keep
16#20D8	External interrupt service routine 172 does not exist.	Stop	Blinking	Keep
16#20D9	External interrupt service routine 173 does not exist.	Stop	Blinking	Keep
16#20DA	External interrupt service routine 174 does not exist.	Stop	Blinking	Keep
16#20DB	External interrupt service routine 175 does not exist.	Stop	Blinking	Keep
16#20DC	External interrupt service routine 176 does not exist.	Stop	Blinking	Keep
16#20DD	External interrupt service routine 177 does not exist.	Stop	Blinking	Keep
16#20DE	External interrupt service routine 178 does not exist.	Stop	Blinking	Keep
16#20DF	External interrupt service routine 179 does not exist.	Stop	Blinking	Keep
16#20E0	External interrupt service routine 180 does not exist.	Stop	Blinking	Keep
16#20E1	External interrupt service routine 181 does not exist.	Stop	Blinking	Keep
16#20E2	External interrupt service routine 182 does not exist.	Stop	Blinking	Keep
16#20E3	External interrupt service routine 183 does not exist.	Stop	Blinking	Keep
16#20E4	External interrupt service routine 184 does not exist.	Stop	Blinking	Keep
16#20E5	External interrupt service routine 185 does not exist.	Stop	Blinking	Keep
16#20E6	External interrupt service routine 186 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20E7	External interrupt service routine 187 does not exist.	Stop	Blinking	Keep
16#20E8	External interrupt service routine 188 does not exist.	Stop	Blinking	Keep
16#20E9	External interrupt service routine 189 does not exist.	Stop	Blinking	Keep
16#20EA	External interrupt service routine 190 does not exist.	Stop	Blinking	Keep
16#20EB	External interrupt service routine 191 does not exist.	Stop	Blinking	Keep
16#20EC	External interrupt service routine 192 does not exist.	Stop	Blinking	Keep
16#20ED	External interrupt service routine 193 does not exist.	Stop	Blinking	Keep
16#20EE	External interrupt service routine 194 does not exist.	Stop	Blinking	Keep
16#20EF	External interrupt service routine 195 does not exist.	Stop	Blinking	Keep
16#20F0	External interrupt service routine 196 does not exist.	Stop	Blinking	Keep
16#20F1	External interrupt service routine 197 does not exist.	Stop	Blinking	Keep
16#20F2	External interrupt service routine 198 does not exist.	Stop	Blinking	Keep
16#20F3	External interrupt service routine 199 does not exist.	Stop	Blinking	Keep
16#20F4	External interrupt service routine 200 does not exist.	Stop	Blinking	Keep
16#20F5	External interrupt service routine 201 does not exist.	Stop	Blinking	Keep
16#20F6	External interrupt service routine 202 does not exist.	Stop	Blinking	Keep
16#20F7	External interrupt service routine 203 does not exist.	Stop	Blinking	Keep
16#20F8	External interrupt service routine 204 does not exist.	Stop	Blinking	Keep
16#20F9	External interrupt service routine 205 does not exist.	Stop	Blinking	Keep
16#20FA	External interrupt service routine 206 does not exist.	Stop	Blinking	Keep
16#20FB	External interrupt service routine 207 does not exist.	Stop	Blinking	Keep
16#20FC	External interrupt service routine 208 does not exist.	Stop	Blinking	Keep
16#20FD	External interrupt service routine 209 does not exist.	Stop	Blinking	Keep
16#20FE	External interrupt service routine 210 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#20FF	External interrupt service routine 211 does not exist.	Stop	Blinking	Keep
16#2100	External interrupt service routine 212 does not exist.	Stop	Blinking	Keep
16#2101	External interrupt service routine 213 does not exist.	Stop	Blinking	Keep
16#2102	External interrupt service routine 214 does not exist.	Stop	Blinking	Keep
16#2103	External interrupt service routine 215 does not exist.	Stop	Blinking	Keep
16#2104	External interrupt service routine 216 does not exist.	Stop	Blinking	Keep
16#2105	External interrupt service routine 217 does not exist.	Stop	Blinking	Keep
16#2106	External interrupt service routine 218 does not exist.	Stop	Blinking	Keep
16#2107	External interrupt service routine 219 does not exist.	Stop	Blinking	Keep
16#2108	External interrupt service routine 220 does not exist.	Stop	Blinking	Keep
16#2109	External interrupt service routine 221 does not exist.	Stop	Blinking	Keep
16#210A	External interrupt service routine 222 does not exist.	Stop	Blinking	Keep
16#210B	External interrupt service routine 223 does not exist.	Stop	Blinking	Keep
16#210C	External interrupt service routine 224 does not exist.	Stop	Blinking	Keep
16#210D	External interrupt service routine 225 does not exist.	Stop	Blinking	Keep
16#210E	External interrupt service routine 226 does not exist.	Stop	Blinking	Keep
16#210F	External interrupt service routine 227 does not exist.	Stop	Blinking	Keep
16#2110	External interrupt service routine 228 does not exist.	Stop	Blinking	Keep
16#2111	External interrupt service routine 229 does not exist.	Stop	Blinking	Keep
16#2112	External interrupt service routine 230 does not exist.	Stop	Blinking	Keep
16#2113	External interrupt service routine 231 does not exist.	Stop	Blinking	Keep
16#2114	External interrupt service routine 232 does not exist.	Stop	Blinking	Keep
16#2115	External interrupt service routine 233 does not exist.	Stop	Blinking	Keep
16#2116	External interrupt service routine 234 does not exist.	Stop	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#2117	External interrupt service routine 235 does not exist.	Stop	Blinking	Keep
16#2118	External interrupt service routine 236 does not exist.	Stop	Blinking	Keep
16#2119	External interrupt service routine 237 does not exist.	Stop	Blinking	Keep
16#211A	External interrupt service routine 238 does not exist.	Stop	Blinking	Keep
16#211B	External interrupt service routine 239 does not exist.	Stop	Blinking	Keep
16#211C	External interrupt service routine 240 does not exist.	Stop	Blinking	Keep
16#211D	External interrupt service routine 241 does not exist.	Stop	Blinking	Keep
16#211E	External interrupt service routine 242 does not exist.	Stop	Blinking	Keep
16#211F	External interrupt service routine 243 does not exist.	Stop	Blinking	Keep
16#2120	External interrupt service routine 244 does not exist.	Stop	Blinking	Keep
16#2121	External interrupt service routine 245 does not exist.	Stop	Blinking	Keep
16#2122	External interrupt service routine 246 does not exist.	Stop	Blinking	Keep
16#2123	External interrupt service routine 247 does not exist.	Stop	Blinking	Keep
16#2124	External interrupt service routine 248 does not exist.	Stop	Blinking	Keep
16#2125	External interrupt service routine 249 does not exist.	Stop	Blinking	Keep
16#2126	External interrupt service routine 250 does not exist.	Stop	Blinking	Keep
16#2127	External interrupt service routine 251 does not exist.	Stop	Blinking	Keep
16#2128	An action in a sequential function chart is incorrectly assigned qualifiers related to time.	Stop	Blinking	Keep
16#2129	The modifier R is assigned to an action in a sequential function chart incorrectly.	Stop	Blinking	Keep
16#3040	Data in the E-CAM exceeds the range or does not exist.	Continue	Keep	Keep
16#3100	Input parameters exceed the available setting range.	Continue	Blinking	Keep
16#3102	An error occurs in a sub-function block inside the function block.	Continue	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3103	The distance between the detecting sensors used for identifying exeptional bags is a negative value.	Continue	Blinking	Keep
16#3104	Phasing is executed again before the previous phasing is completed.	Continue	Blinking	Keep
16#3105	Superimposing is executed again before the previous superimposing is completed.	Continue	Blinking	Keep
16#3106	Chain position compensation is triggered before the previous compensation is completed.	Continue	Blinking	Keep
16#3107	Film axis position compensation is triggered before the previous compensation is completed.	Continue	Blinking	Keep
16#3108	Knife position compensation is triggered before the previous compensation is completed.	Continue	Blinking	Keep
16#3400	Motion axis number is incorrect.	Continue	Keep	Keep
16#3401	SDO Data Type setting error (0~199)	Continue	Keep	Keep
16#3404	The number of the counting channel exceeds the available setting range.	Continue	Keep	Keep
16#3405	A negative value is given to *Velocity.*	Continue	Blinking	Keep
16#340A	Homing mode setting error.	Continue	Blinking	Keep
16#340B	Target distance is 0.	Continue	Blinking	Keep
16#3410	User unit setting error; or the output pulse type setting error.	Continue	Blinking	Keep
16#3411	Velocity factor overrides setting error.	Continue	Blinking	Keep
16#3414	Pulse type setting error in DFB_HCnt.	Continue	Keep	Keep
16#3415	Comparison confdition setting error in DFB_Compare.	Continue	Keep	Keep
16#3419	Master axis position is negative value.	Continue	Blinking	Keep
16#341B	Maxmimum speed setting error.	Continue	Blinking	Keep
16#3429	G-code compiling error.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#342A	G-code pogram source error.	Continue	Keep	Keep
16#342B	G-code ID setting error.	Continue	Keep	Keep
16#342C	Gcode is in operation.	Continue	Keep	Keep
16#342D	Gcode grammer is being checked.	Continue	Keep	Keep
16#342E	The setting of the Gcode Filter is out of the range.	Continue	Keep	Keep
16#3430	This group number already exists.	Continue	Keep	Keep
16#3431	Motion axis number is repeated in the same group in DFB_GroupEnable.	Continue	Keep	Keep
16#3432	The specified group number does not exist.	Continue	Keep	Keep
16#3433	The number of axes is insufficient for the specified group axes motion.	Continue	Keep	Keep
16#3434	DFB_GroupDisable is executed when group motion is in progress.	Continue	Keep	Keep
16#3435	Motion axis number is repeated between different groups when DFB_GroupEnable is enabled.	Continue	Keep	Keep
16#3436	The axis number of the first order should be a positive number other than 0.	Continue	Keep	Keep
16#3437	The group number exceeds the setting range.	Continue	Keep	Keep
16#3438	The designated group is in “ErrorStop” state.	Continue	Keep	Keep
16#343A	Group is executing the function block ImmediateStop.	Continue	Keep	Keep
16#343B	Errors occur in other axes of the group.	Continue	Blinking	Keep
16#3461	The required communication parameters for PDO settings are not specified.	Continue	Blinking	Keep
16#3463	The designated ECAT Slave does not exist.	Continue	Keep	Keep
16#3500	The axis is not in “Ready” state.	Continue	Blinking	Keep
16#3501	The selected channel has been used in FB.	Continue	Blinking	Keep
16#3502	It is not allowed to set positions.	Continue	Blinking	Keep
16#3505	An error occurs when writing cam data.	Continue	Keep	Keep
16#3506	The axis is in ”Coordinated” state.	Continue	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3507	The axis is in “ErrorStop” state.	Continue	Blinking	Keep
16#3508	The axis is not in “StandStill” state.	Continue	Blinking	Keep
16#3509	The axis is in “Stopping” state.	Continue	Blinking	Keep
16#350B	The time to acceleration is too short.	Continue	Blinking	Keep
16#350C	The time to deceleration is too short.	Continue	Blinking	Keep
16#350D	The CAM data length for reading is out of the setting range.	Continue	Blinking	Keep
16#350E	The CAM data length for writing is out of the setting range.	Continue	Blinking	Keep
16#350F	The axis is in “Synchronized” state.	Continue	Blinking	Keep
16#3512	Cam data does not exist.	Continue	Keep	Keep
16#3526	The movement error occurs before the axis.	Continue	Blinking	Keep
16#3600	The state of axis is incorrect.	Continue	Blinking	Keep
16#3601	The limit of the number of buffering instructions is reached	Continue	Blinking	Keep
16#3602	A multiple instructions which are not allowed to be executed at the same time are executed.	Continue	Blinking	Keep
16#3603	Buffermode parameter setting error	Continue	Blinking	Keep
16#3604	Errors occur on the motion direction of the function block	Continue	Blinking	Keep
16#3605	P1 exceeds the available range	Continue	Blinking	Keep
16#3606	P2 exceeds the available range	Continue	Blinking	Keep
16#3607	V1 exceeds the available range	Continue	Blinking	Keep
16#3608	V2 exceeds the available range	Continue	Blinking	Keep
16#3612	It has reached the positive limit.	Continue	Blinking	Keep
16#3613	It has reached the negative limit.	Continue	Blinking	Keep
16#3614	The servo limit is exceeded.	Continue	Blinking	Keep
16#3617	The acceleration exceeds the setting range.	Continue	Blinking	Keep
16#3618	The disceleration exceeds the setting range.	Continue	Blinking	Keep
16#3619	The station address does not exist.	Continue	Keep	Keep
16#3620	The schedule buffer section of SDO is full.	Continue	Keep	Keep
16#3622	SDO OD data type is not matched.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3623	SDO is overtime.	Continue	Keep	Keep
16#3624	SDO data written error	Continue	Keep	Keep
16#3625	SDO data reading error	Continue	Keep	Keep
16#3626	SDO retry time exceeds the setting range.	Continue	Keep	Keep
16#3800	Motion network disconnected during the execution of the instruction.	Continue	Blinking	Keep
16#3801	EtherCAT axis error occurs on the motion network	Continue	Blinking	Keep
16#3900	Failed to re-connect to the motion network.	Continue	Blinking	Keep
16#3904	Motion network master can not read Slave parameters via SDO.	Continue	Blinking	Keep
16#3905	Motion network master can not write Slave parameters via SDO.	Continue	Blinking	Keep
16#3906	Torque limit setting error in MC_SetTorqueLimit	Continue	Blinking	Keep
16#3907	The function is not available for imaginary axes.	Continue	Blinking	Keep
16#3909	The motion network is currently executing other network functions.	Continue	Blinking	Keep
16#390C	Error occurred on the axis during operation.	Continue	Blinking	Keep
16#3910	Disengage when the axes are not in engaging state.	Continue	Blinking	Keep
16#3911	Software limit error	Continue	Blinking	Keep
16#3912	The value in the input contact of the function block exceeds the rotary axis range.	Continue	Blinking	Keep
16#3913	Synchronization for engagement fails	Continue	Blinking	Keep
16#3914	GearInPos velocity is set too small	Continue	Blinking	Keep
16#3915	GearInPos jerk is set too small	Continue	Blinking	Keep
16#3916	GearInPos engagement time is set too small	Continue	Blinking	Keep
16#3917	GearInPos the velocity of the main axis is 0 when the engagement started	Continue	Blinking	Keep
16#3918	The enagement velocity is larger than the AxisVelocityMax	Continue	Blinking	Keep
16#3919	GearInPos the main axis moves in opposite direction	Continue	Blinking	Keep
16#3920	GearInPost the acceleration is set too small.	Continue	Blinking	Keep
16#3921	GearInPosMasterStartDist is out of range.	Continue	Blinking	Keep
16#3922	GearInPos engaging displacement is too small.	Continue	Blinking	Keep
16#3923	GearInPos engaging displacement is too large.	Continue	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#3924	GearInPos the velocity of the main axis starts to change when engaging started.	Continue	Blinking	Keep
16#3950	Capture cannot be used when the pulse speed is beyond 1MHz.	Continue	Keep	Keep
16#3951	CamCurve wrong input (for example, Concatenate cannot be true or other parameters are out of range.)	Continue	Keep	Keep
16#3953	Capture uses the same channel number repeatedly.	Continue	Keep	Keep
16#3954	Torque ramp fail to write	Continue	Blinking	Keep
16#3955	Torque velocity fail to write	Continue	Blinking	Keep
16#3A00	CAM table setting error	Continue	Blinking	Keep
16#3A01	CamIn master setting error	Continue	Blinking	Keep
16#3A02	CamIn CAM table changed too soon	Continue	Blinking	Keep
16#3A03	CamIn activation mode setting exceeds the available range	Continue	Blinking	Keep
16#3A04	CamIn start mode setting exceeds the available range	Continue	Blinking	Keep
16#3A05	CamIn master scaling is set to 0.0	Continue	Blinking	Keep
16#3A06	CamIn slave scaling is set to 0.0	Continue	Blinking	Keep
16#3A10	CamIn master start position is set too Small.	Continue	Blinking	Keep
16#3A13	CamIn the velocity is set too small.	Continue	Blinking	Keep
16#3A15	CamIn jerk is set too small	Continue	Blinking	Keep
16#3A16	CamIn maximum acceleration is set too small	Continue	Blinking	Keep
16#3A17	CamIn Start mode distance is set too small	Continue	Blinking	Keep
16#3A18	CamIn Start mode distance is set too large	Continue	Blinking	Keep
16#3A19	Too many CamIn are wait to start	Continue	Blinking	Keep
16#3A20	Master is moving in the negative direction.	Continue	Blinking	Keep
16#3A21	CamIn is cancelled when it is not in “CamIn” state.	Continue	Blinking	Keep
16#3D00	EtherCAT ENI file does not match current hardware configuration.	Continue	Blinking	Keep
16#3D01	Slave lost in motion network.	Continue	Blinking	Keep
16#3D03	EtherCAT DC time is set too small	Continue	Blinking	Keep
16#6001	Illegal IP address (SM1107)	Continue	Blinking	Keep
16#6002	Illegal netmask address (SM1107)	Continue	Blinking	Keep
16#6003	Illegal gateway mask (SM1107)	Continue	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#6004	The IP address filter is set incorrectly. (SM1108)	Continue	Blinking	Keep
16#6006	The static ARP table is set incorrectly. (SM1108)	Continue	Blinking	Keep
16#600D	The RJ45 port is not connected. (SM1100)	Continue	Keep	Keep
16#620D	The length of the data which needs to be sent in **a UDP Socket Configuration** window is illegal.	Continue	Keep	Keep
16#6212	There is no response from the remote device after the timeout period.	Continue	Keep	Keep
16#6213	The data received exceeds the limit.	Continue	Keep	Keep
16#6214	The remote device refuses the connection.	Continue	Keep	Keep
16#6400	The number of TCP connections reaches the upper limit, or the flag which is related to the sending of the data is not set to ON.	Continue	Keep	Keep
16#6401	The remote device aborts the connection.	Continue	Keep	Keep
16#6402	There is no response from the remote device after the timeout period.	Continue	Keep	Keep
16#6403	The remote IP address used in the applied instruction is illegal.	Continue	Keep	Keep
16#6404	The MODBUS function code not supported is received.	Continue	Keep	Keep
16#6405	The number of data which will be received is not consistent with the actual length of the data.	Continue	Keep	Keep
16#6501	The remote device involved in the data exchange does not respond after the timeout period. (SM828~SM955)	Continue	OFF	OFF
16#6502	The remote device involved in the data exchange does not respond correctly. (SM828~SM955)	Continue	OFF	OFF
16#6700	MODBUS TCP data exchange initialization error	Continue	Keep	Keep
16#6701	MODBUS TCP data exchange timeout	Continue	Keep	Keep
16#6702	MODBUS TCP data receiving error	Continue	Keep	Keep
16#7002	This function is not available for CPU modules.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#7203	Invalid access code	Continue	Keep	Keep
16#7401	Function code error	Continue	Keep	Keep
16#7402	The packet exceeds the max. data length.	Continue	Keep	Keep
16#7407	Non-ASCII characters exist in the command.	Continue	Keep	Keep
16#7408	PLC is in RUN mode	Continue	Keep	Keep
16#740A	The CPU memory is being written or failed to be written.	Continue	Keep	Keep
16#740B	The Clear or Reset operation is in progress.	Continue	Keep	Keep
16#740C	The backplane number in a communication command is incorrect.	Continue	Keep	Keep
16#740D	The slot number in a communication command is incorrect.	Continue	Keep	Keep
16#740E	Error occurs when the memory is being cleared.	Continue	Keep	Keep
16#740F	Communication timeout	Continue	Keep	Keep
16#7410	The function code for responding the instruction is inconsistent.	Continue	Keep	Keep
16#7412	Data cannot be downloaded to CPU becaue SW1 is ON.	Continue	Keep	Keep
16#757D	The number of times users can enter the PLC password is 0.	Continue	Keep	Keep
16#757E	Incorrect PLC password	Continue	Keep	Keep
16#8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	Continue	Keep	Keep
16#8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	Continue	Keep	Keep
16#8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	Continue	Keep	Keep
16#8230	The CPU parameter downloaded is incorrect. The IP address is illegal.	Continue	Blinking	Keep
16#8231	The CPU parameter downloaded is incorrect. The netmask address is illegal.	Continue	Blinking	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#8232	The CPU parameter downloaded is incorrect. The gateway address is illegal.	Continue	Blinking	Keep
16#8233	The CPU parameter downloaded is incorrect. The IP address filter is set incorrectly.	Continue	Blinking	Keep
16#8235	The CPU parameter downloaded is incorrect. The static ARP table is set incorrectly.	Continue	Blinking	Keep
16#8236	A CPU parameter downloaded is incorrect. The NTP client service is set incorrectly.	Continue	Keep	Keep
16#8240	A CPU parameter downloaded is incorrect. The data exchange by means of Ethernet is set incorrectly	Continue	Keep	Keep
16#8242	Gcode Data ERROR	Continue	Blinking	Keep
16#8243	ECAM Data ERROR	Continue	Blinking	Keep
16#8244	ENI Data ERROR	Continue	Blinking	Keep
16#8245	EtherCat Data ERROR	Continue	Blinking	Keep
16#8246	Axes Parameters Data ERROR	Continue	Blinking	Keep
16#8247	External Gcode Data ERROR	Continue	Blinking	Keep
16#8522	A module configuration is being scanned.	Continue	Keep	Keep
16#853B	An I/O module is not configured.(wirte error)	Continue	Keep	Keep
16#853C	An I/O module does not exist. (wirte error)	Continue	Keep	Keep
16#854B	An I/O module is not configured. (read error)	Continue	Keep	Keep
16#854C	An I/O module does not exist. (read error)	Continue	Keep	Keep
16#8572	The checksum of the module configuration table is incorrect.	Continue	Keep	Keep
16#8576	The checksum of the module parameter setting is incorrect.	Continue	Keep	Keep
16#857A	The checksum of the module parameter mapping table is incorrect.	Continue	Keep	Keep
16#85E1	An I/O interrupt number is incorrect.	Continue	Keep	Keep
16#85E2	An I/O interrupt service routine does not exist.	Continue	Keep	Keep
16#860F	System restoration error	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#8611	No memory card exists, or the memory card format is incorrect.	Continue	Keep	Keep
16#9A33	An error occurs when COM1 communicates with slave 19 by Modbus.	Continue	Keep	Keep
16#9A34	An error occurs when COM1 communicates with slave 20 by Modbus.	Continue	Keep	Keep
16#9A35	An error occurs when COM1 communicates with slave 21 by Modbus.	Continue	Keep	Keep
16#9A47	COM1 receives no response from slave 7 by Modbus.	Continue	Keep	Keep
16#9B01	An error occurs when the Modbus connection of COM2 is initialized.	Continue	Keep	Keep
16#9B21	An error occurs when COM2 communicates with slave 1 by MODBUS.	Continue	Keep	Keep
16#9B22	An error occurs when COM2 communicates with slave 2 by MODBUS.	Continue	Keep	Keep
16#9B23	An error occurs when COM2 communicates with slave 3 by MODBUS.	Continue	Keep	Keep
16#9B24	An error occurs when COM2 communicates with slave 4 by MODBUS.	Continue	Keep	Keep
16#9B25	An error occurs when COM2 communicates with slave 5 by MODBUS.	Continue	Keep	Keep
16#9B26	An error occurs when COM2 communicates with slave 6 by MODBUS.	Continue	Keep	Keep
16#9B27	An error occurs when COM2 communicates with slave 7 by MODBUS.	Continue	Keep	Keep
16#9B28	An error occurs when COM2 communicates with slave 8 by MODBUS.	Continue	Keep	Keep
16#9B29	An error occurs when COM2 communicates with slave 9 by MODBUS.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B2A	An error occurs when COM2 communicates with slave 10 by MODBUS.	Continue	Keep	Keep
16#9B2B	An error occurs when COM2 communicates with slave 11 by MODBUS.	Continue	Keep	Keep
16#9B2C	An error occurs when COM2 communicates with slave 12 by MODBUS.	Continue	Keep	Keep
16#9B2D	An error occurs when COM2 communicates with slave 13 by MODBUS.	Continue	Keep	Keep
16#9B2E	An error occurs when COM2 communicates with slave 14 by MODBUS.	Continue	Keep	Keep
16#9B2F	An error occurs when COM2 communicates with slave 15 by MODBUS.	Continue	Keep	Keep
16#9B30	An error occurs when COM2 communicates with slave 16 by MODBUS.	Continue	Keep	Keep
16#9B31	An error occurs when COM2 communicates with slave 17 by MODBUS.	Continue	Keep	Keep
16#9B32	An error occurs when COM2 communicates with slave 18 by MODBUS.	Continue	Keep	Keep
16#9B33	An error occurs when COM2 communicates with slave 19 by MODBUS.	Continue	Keep	Keep
16#9B34	An error occurs when COM2 communicates with slave 20 by MODBUS.	Continue	Keep	Keep
16#9B35	An error occurs when COM2 communicates with slave 21 by MODBUS.	Continue	Keep	Keep
16#9B36	An error occurs when COM2 communicates with slave 22 by MODBUS.	Continue	Keep	Keep
16#9B37	An error occurs when COM2 communicates with slave 23 by MODBUS.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B38	An error occurs when COM2 communicates with slave 24 by MODBUS.	Continue	Keep	Keep
16#9B39	An error occurs when COM2 communicates with slave 25 by MODBUS.	Continue	Keep	Keep
16#9B3A	An error occurs when COM2 communicates with slave 26 by MODBUS.	Continue	Keep	Keep
16#9B3B	An error occurs when COM2 communicates with slave 27 by MODBUS.	Continue	Keep	Keep
16#9B3C	An error occurs when COM2 communicates with slave 28 by MODBUS.	Continue	Keep	Keep
16#9B3D	An error occurs when COM2 communicates with slave 29 by MODBUS.	Continue	Keep	Keep
16#9B3E	An error occurs when COM2 communicates with slave 30 by MODBUS.	Continue	Keep	Keep
16#9B3F	An error occurs when COM2 communicates with slave 31 by MODBUS.	Continue	Keep	Keep
16#9B40	An error occurs when COM2 communicates with slave 32 by MODBUS.	Continue	Keep	Keep
16#9B41	COM2 receives no response from slave 1 by MODBUS.	Continue	Keep	Keep
16#9B42	COM2 receives no response from slave 2 by MODBUS.	Continue	Keep	Keep
16#9B43	COM2 receives no response from slave 3 by MODBUS.	Continue	Keep	Keep
16#9B44	COM2 receives no response from slave 4 by MODBUS.	Continue	Keep	Keep
16#9B45	COM2 receives no response from slave 5 by MODBUS.	Continue	Keep	Keep
16#9B46	COM2 receives no response from slave 6 by MODBUS.	Continue	Keep	Keep
16#9B47	COM2 receives no response from slave 7 by MODBUS.	Continue	Keep	Keep
16#9B48	COM2 receives no response from slave 8 by MODBUS.	Continue	Keep	Keep
16#9B49	COM2 receives no response from slave 9 by MODBUS.	Continue	Keep	Keep

Error code	Description	CPU Status	LED indicator status	
			ERROR	BUS FAULT
16#9B4A	COM2 receives no response from slave 10 by MODBUS.	Continue	Keep	Keep
16#9B4B	COM2 receives no response from slave 11 by MODBUS.	Continue	Keep	Keep
16#9B4C	COM2 receives no response from slave 12 by MODBUS.	Continue	Keep	Keep
16#9B4D	COM2 receives no response from slave 13 by MODBUS.	Continue	Keep	Keep
16#9B4E	COM2 receives no response from slave 14 by MODBUS.	Continue	Keep	Keep
16#9B4F	COM2 receives no response from slave 15 by MODBUS.	Continue	Keep	Keep
16#9B50	COM2 receives no response from slave 16 by MODBUS.	Continue	Keep	Keep
16#9B51	COM2 receives no response from slave 17 by MODBUS.	Continue	Keep	Keep
16#9B52	COM2 receives no response from slave 18 by MODBUS.	Continue	Keep	Keep
16#9B53	COM2 receives no response from slave 19 by MODBUS.	Continue	Keep	Keep
16#9B54	COM2 receives no response from slave 20 by MODBUS.	Continue	Keep	Keep
16#9B55	COM2 receives no response from slave 21 by MODBUS.	Continue	Keep	Keep
16#9B56	COM2 receives no response from slave 22 by MODBUS.	Continue	Keep	Keep
16#9B57	COM2 receives no response from slave 23 by MODBUS.	Continue	Keep	Keep
16#9B58	COM2 receives no response from slave 24 by MODBUS.	Continue	Keep	Keep
16#9B59	COM2 receives no response from slave 25 by MODBUS.	Continue	Keep	Keep
16#9B5A	COM2 receives no response from slave 26 by MODBUS.	Continue	Keep	Keep
16#9B5B	COM2 receives no response from slave 27 by MODBUS.	Continue	Keep	Keep
16#9B5C	COM2 receives no response from slave 28 by MODBUS.	Continue	Keep	Keep
16#9B5D	COM2 receives no response from slave 29 by MODBUS.	Continue	Keep	Keep
16#9B5E	COM2 receives no response from slave 30 by MODBUS.	Continue	Keep	Keep
16#9B5F	COM2 receives no response from slave 31 by MODBUS.	Continue	Keep	Keep
16#9B60	COM2 receives no response from slave 32 by MODBUS.	Continue	Keep	Keep

■

■

Analog I/O Modules and Temperature Measurement Modules

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A000	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A001	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A002	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A003	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A004	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A005	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A006	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A007	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware.	Blinking	
16#A400	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware.	ON	
16#A401	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware.	ON	
16#A402	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware.	ON	
16#A403	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware.	ON	
16#A404	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware.	ON	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A405	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.	ON	
16#A406	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware.	ON	
16#A407	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware.	ON	
16#A600	Hardware failure	ON	
16#A601	The external voltage is abnormal.	ON	
16#A602	Internal error The CJC is abnormal.	ON	
16#A603	Internal error The factory correction is abnormal.	ON	
16#A800	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A801	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A802	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A803	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A804	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A805	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A806	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware.	OFF	
16#A807	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware.	OFF	

*With regard to the errors related to the input signals' exceeding the range of inputs which can be received by the hardware and the conversion values' exceeding the limits, whether the error code generated is within the range between 16#A000 and 16#A00F, within the range between 16#A400 and 16#A40F, or within the range between 16#A800~16#A80F depends on the LED indicator status defined by users.

AH02HC-5A/AH04HC-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A001	The linear accumulation in channel 0 exceeds the range.	Blinking	
16#A002	The scale set for channel 0 exceeds the range.	Blinking	
16#A003	The number of cycles set for channel 0exceeds the range.	Blinking	
16#A004	The comparison value set for channel 0 exceeds the range.	Blinking	
16#A005	A limit value set for channel 0 is incorrect.	Blinking	
16#A006	The interrupt number set for channel 0 exceeds the range.	Blinking	
16#A011	The linear accumulation in channel 1 exceeds the range.	Blinking	
16#A012	The scale set for channel 1 exceeds the range.	Blinking	
16#A013	The number of cycles set for channel 1 exceeds the range.	Blinking	
16#A014	The comparison value set for channel 1 exceeds the range.	Blinking	
16#A015	A limit value set for channel 1 is incorrect.	Blinking	
16#A016	The interrupt number set for channel 1 exceeds the range.	Blinking	
16#A021	The linear accumulation in channel 2 exceeds the range.	Blinking	
16#A022	The scale set for channel 2 exceeds the range.	Blinking	
16#A023	The number of cycles set for channel 2 exceeds the range.	Blinking	
16#A024	The comparison value set for channel 2 exceeds the range.	Blinking	
16#A025	A limit value set for channel 2 is incorrect.	Blinking	
16#A026	The interrupt number set for channel 2 exceeds the range.	Blinking	
16#A031	The linear accumulation in channel 3 exceeds the range.	Blinking	
16#A032	The scale set for channel 3 exceeds the range.	Blinking	
16#A033	The number of cycles set for channel 3 exceeds the range.	Blinking	
16#A034	The comparison value set for channel 3 exceeds the range.	Blinking	
16#A035	A limit value set for channel 3 is incorrect.	Blinking	
16#A036	The interrupt number set for channel 3 exceeds the range.	Blinking	

AH05PM-5A/AH10PM-5A/AH15PM-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	Error
16#A002	The subroutine has no data.	Blinking	
16#A003	CJ, CJN, and JMP have no matching pointers.	Blinking	
16#A004	There is a subroutine pointer in the main program.	Blinking	
16#A005	Lack of the subroutine	Blinking	
16#A006	The pointer is used repeatedly in the same program.	Blinking	
16#A007	The subroutine pointer is used repeatedly.	Blinking	
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	Blinking	
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	Blinking	
16#A00A	The pointer used in JMP is the same as a subroutine pointer.	Blinking	
16#A00B	Target position (I) of the single speed is incorrect.	Blinking	
16#A00C	Target position (II) of the single-axis motion is incorrect.	Blinking	
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Blinking	
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	Blinking	
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Blinking	
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Blinking	
16#A011	The setting of the JOG speed is incorrect.	Blinking	
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	Blinking	
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	Blinking	
16#A014	The limit switch is reached.	Blinking	
16#A015	The device which is used exceeds the device range.	Blinking	
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Blinking	
16#A018	The conversion into the floating-point number is incorrect.	Blinking	
16#A019	The conversion into the binary-coded decimal number is incorrect.	Blinking	
16#A01A	Incorrect division operation (The divisor is 0.)	Blinking	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	Error
16#A01B	General program error	Blinking	
16#A01C	LD/LDI has been used more than nine times.	Blinking	
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Blinking	
16#A01E	SRET is used between RPT and RPE.	Blinking	
16#A01F	There is no M102 in the main program, or there is no M2 in the motion program.	Blinking	
16#A020	The wrong instruction is used, or the device used exceeds the range.	Blinking	

AH20MC-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A002	The subroutine has no data.	Blinking	
16#A003	CJ, CJN, and JMP have no matching pointers.	Blinking	
16#A004	There is a subroutine pointer in the main program.	Blinking	
16#A005	Lack of the subroutine	Blinking	
16#A006	The pointer is used repeatedly in the same program.	Blinking	
16#A007	The subroutine pointer is used repeatedly.	Blinking	
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	Blinking	
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	Blinking	
16#A00B	Target position (I) of the single speed is incorrect.	Blinking	
16#A00C	Target position (II) of the single-axis motion is incorrect.	Blinking	
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Blinking	
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	Blinking	
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Blinking	
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Blinking	
16#A011	The setting of the JOG speed is incorrect.	Blinking	

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	Blinking	
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	Blinking	
16#A014	The limit switch is reached.	Blinking	
16#A015	The device which is used exceeds the device range.	Blinking	
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Blinking	
16#A018	The conversion into the floating-point number is incorrect.	Blinking	
16#A019	The conversion into the binary-coded decimal number is incorrect.	Blinking	
16#A01A	Incorrect division operation (The divisor is 0.)	Blinking	
16#A01B	General program error	Blinking	
16#A01C	LD/LDI has been used more than nine times.	Blinking	
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Blinking	
16#A01E	SRET is used between RPT and RPE.	Blinking	
16#A01F	Incorrect division operation (The divisor is 0.)	Blinking	
16#A020	The wrong instruction is used, or the device used exceeds the range.	Blinking	

AH10EN-5A / AH15EN-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	Error
16#A001	The IP address of host 1 conflicts with another system on the network.	Blinking	
16#A002	The IP address of host 2 conflicts with another system on the network.	Blinking	
16#A003	DHCP for host 1 fails.	Blinking	
16#A004	DHCP for host 2 fails.	Blinking	
16#A401	Hardware error	ON	
16#A402	The initialization of the system fails.	ON	

A2

AH10SCM-5A / AH15SCM-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A002	The setting of the UD Link is incorrect, or the communication fails.	Blinking	
16#A401	Hardware error	ON	
16#A804	The communication through the communication port is incorrect.	OFF	
16#A808	MODBUS communication error	OFF	

AH10DNET-5A

Error code	Description	LED indicator status		
		CPU	Module	
		BUS FAULT	MS	NS
16#A0F0	The node ID of AH10DNET-5A is the same as other node ID on the network, or exceeds the range.	The red light blinks.	The green light blinks.	The red light is ON.
16#A0F1	No slave is put on the scan list of AH10DNET-5A.	The red light blinks.	The green light blinks.	The green light is ON.
16#A0F2	The working voltage of AH10DNET-5A is low.	The red light blinks.	The red light blinks.	The red light blinks.
16#A0F3	AH10DNET-5A enters the test mode.	The red light blinks.	The orange light is ON.	The orange light is ON.
16#A0F4	The bus of AH10DNET-5A is switched OFF.	The red light blinks.	The green light is ON.	The red light is ON.
16#A0F5	AH10DNET-5A detects that there is no network power supply to the DeviceNet.	The red light blinks.	The red light blinks.	The red light is ON.
16#A0F6	Something is wrong with the internal memory of AH10DNET-5A.	The red light blinks.	The red light is ON.	The green light blinks.
16#A0F7	Something is wrong with the data exchange unit of AH10DNET-5A.	The red light blinks.	The red light is ON.	The green light blinks.
16#A0F8	The product ID of AH10DNET-5A is incorrect.	The red light blinks.	The red light is ON.	The green light blinks.
16#A0F9	An error occurs when the data is read from AH10DNET-5A, or when the data is written into AH10DNET-5A.	The red light blinks.	The red light is ON.	The red light is ON.
16#A0FA	The node ID of AH10DNET-5A is the same as that of the slave set in the scan list.	The red light blinks.	The green light is ON.	The red light is ON.

Error code	Description	LED indicator status		
		CPU	Module	
		BUS FAULT	MS	NS
16#A0FB	The data exchange between AH10DNET and AH CPU failed.	The red light blinks.	The green light is ON.	The green light is ON.
16#A0FC	Errors occur in the slaves, on the module of an AHRTU-DNET backplane, or on the AHRTU-DNET backplane connection.	The red light blinks.	The red light blinks.	The green light is ON.

AH10PFBM-5A

Error code	Description	LED indicator status			
		CPU		MODULE	
		BUS FAULT	RUN	SYS	DP
16#A001	The master is not set.	The red light blinks.	The green light is ON.	The green light is ON.	The green light blinks.
16#A003	The master station enters the test mode.	The red light blinks.	The green light is ON.	The green light is ON.	The green light is ON.
16#A005	A timeout occurs when chips inside the master station communicate.	The red light blinks.	The green light is ON.	The green light is ON.	The green light is ON.
16#A00B	A timeout occurs when AH10PFBM-5A exchanges data exchange with a PLC.	The red light blinks.	The green light is ON.	The green light is ON.	The green light is ON.
16#A402	The PLC does not assign the I/O mapping area to the master.	The red light is ON.	The green light is ON.	The green light is ON.	The green light is ON.
16#A404	Master initializing error	The red light is ON.	The green light is ON.	The green light is ON.	The green light is ON.
16#A406	Internal storage unit error	The red light is ON.	The green light is ON.	The green light is ON.	The green light is ON.
16#A407	Data exchange unit error	The red light is ON.	The green light is ON.	The green light is ON.	The green light is ON.
16#A408	Master serial number detection error	The red light is ON.	The green light is ON.	The green light is ON.	The green light is ON.
16#A4E2	The master detects that all the slaves are offline.	The red light is ON.	OFF	The green light is ON.	The red light is ON.
	The master detects that some of the slaves are offline.	The red light is ON.	OFF	The green light is ON.	The red light blinks.
16#A4E6	The master detects that an error occurs in the module connected to AHRTU-PFBS-5A.	The red light is ON.	The green light is ON.	The green light is ON.	The green light is ON.

AH10PFBS-5A

Error code	Description	LED indicator status		
		CPU	MODULE	
		BUS FAULT	RUN	NET
16#A4F0	The node address of AH10PFBS-5A exceeds the valid range.	The red light is ON.	The green light is ON.	The green light is ON.
16#A4F1	Internal hardware error	The red light is ON.	The green light is ON.	The green light is ON.
16#A4F2	Parameter error	The red light is ON.	The green light is ON.	The green light is ON.
16#A4F3	Configuration error	The red light is ON.	The green light is ON.	The green light is ON.
16#A4F4	GPIO detection error	The red light is ON.	The green light is ON.	The green light is ON.
16#A4F5	AH10PFBS-5A enters the mode of factory test.	The red light is ON.	The green light is ON.	The green light is ON.
16#A4F6	1. AH10PFBS-5A has not been connected to the PROFIBUS-DP network. 2. PROFIBUS-DP master has not configured AH10PFBS-5A slave or the configured node address of AH10PFBS-5A is inconsistent with that of the actually connected one.	The red light is ON.	The green light is ON.	The red light is ON.

AH10COPM-5A

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A0B0	AH10COPM-5A does not send a heartbeat message after a set period of time.	Blinking	The red light flashes twice.
16#A0B1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Blinking	OFF
16#A0B2	The master station selected does not send a node guarding message after a set period of time.	Blinking	The red light flashes twice.
16#A0E0	AH10COPM-5A receives an emergency message from a slave station.	Blinking	OFF
16#A0E1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Blinking	OFF

Error code	Description	LED indicator status	
		CPU	Module
		BUS FAULT	ERROR
16#A0E2	AH10COPM-5A does not receive a PDO from a slave station.	Blinking	OFF
16#A0E3	An automatic SDO is not downloaded successfully.	Blinking	OFF
16#A0E4	A PDO parameter is not set successfully.	Blinking	OFF
16#A0E5	A key parameter is set incorrectly.	Blinking	OFF
16#A0E6	The actual network configuration is not the same as the network configuration set.	Blinking	OFF
16#A0E7	The control of the errors in a slave station is not sent after a set period of time.	Blinking	The red light flashes twice.
16#A0E8	The master station address is the same as a slave station address.	Blinking	OFF
16#A0F1	No slave station is added to the node list in CANopen builder.	Blinking	OFF
16#A0F3	An error occurs in AH10COPM-5A.	Blinking	OFF
16#A0F4	The bus used is off.	Blinking	The red light is ON.
16#A0F5	The node address of AH10COPM-5A is set incorrectly.	Blinking	OFF
16#A0F6	Internal error: An error occurs in the manufacturing process in the factory.	Blinking	OFF
16#A0F7	Internal error: GPIO error	Blinking	OFF
16#A0F8	Hardware error	Blinking	OFF
16#A0F9	Low voltage	Blinking	OFF
16#A0FA	An error occurs in the firmware of AH10COPM-5A.	Blinking	OFF
16#A0FB	The transmission registers in AH10COPM-5A are full.	Blinking	OFF
16#A0FC	The reception registers in AH10COPM-5A are full.	Blinking	OFF

A.2.2. Error Codes and Troubleshooting

AHxxEMC-5A

You can get the corrective actions from the tables below according to the error codes.

ERROR Indicator ON

Error Code	Description	Corrective action
16#000B	The program in the PLC is damaged.	Download the program again.
16#000D	The CPU parameters are damaged.	Reset the CPU parameter, and download it.
16#0010	The access to the memory in the CPU is denied.	Download the program or parameters again. If the problem still occurs, please contact the manufacturer.
16#0011	The PLC ID is incorrect. (SM9)	Please check the PLC ID.
16#0012	The PLC password is incorrect. (SM9)	Please check the PLC password.
16#0014	The procedure of restoring the system can not be executed. (SM9)	The contents of the system backup file are incorrect, or the file does not exist in the path specified. If the file exists and the procedure of restoring the system can not be executed, please back up the system again. If the error still occurs, please contact the manufacturer. (You can refer to ***AH Motion Controller – Operation Manual*** for more details about using memory cards)
16#0015	The module table is incorrect. (SM10)	The module table stored in the CPU module is incorrect. Compare the module table in HWCONFIG with the actual module configuration, and download the module table again.
16#0016	The module setting is incorrect. (SM10)	The module setting stored in the CPU module is incorrect. Check whether the version of the module inserted in the slot is the same as the version of the module in HWCONFIG. After the version of the module is updated, users can download the module setting again.
16#0017	The data register exceeds the device range. (SM10)	The data register stored in the CPU module exceeds the device range. Check whether the module parameter in HWCONFIG is correct, and download the module parameter again.

Error Code	Description	Corrective action
16#001B	Timed interrupt 0 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001C	Timed interrupt 1 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001D	Timed interrupt 2 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001E	Timed interrupt 3 is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#001F	The watchdog timer is set incorrectly.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#0020	The setting of the fixed scan time is incorrect.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#0021	The setting of the fixed scan time is incorrect.	Set the CPU parameter in HWCONFIG again, and download the CPU parameter again.
16#0022	The CPU parameter downloaded to the PLC is incorrect.	Download the CPU parameter again.
16#0023	CPU parameters setting error. The state of Y devices when the CPU is set from STOP to RUN is incorrect	Adjust the CPU parameters setting in HWCONFIG and download it to PLC again.
16#0026	The Communication Ratio box in the Communication Loading of Scan Time (%) section in the PLC Parameter Setting window is set incorrectly.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0027	The latching auxiliary relay range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0028	The latching data register range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0029	The latching timer range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.

Error Code	Description	Corrective action
16#002A	The latching counter range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#002B	The latching 32-bit counter range which is set is incorrect.	Reset the CPU or set the CPU to the default settings, and download the program and parameters again.
16#0050	The memories in the latched special auxiliary relays are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0051	The latched special data registers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0052	The memories in the latched auxiliary relays are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0053	The latched timers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0054	The latched counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0055	The latched 32-bit counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0056	The memories in the latched timers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0057	The memories in the latched counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#0058	The memories in the latched 32-bit counters are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.

Error Code	Description	Corrective action
16#0059	The latched data registers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.
16#005A	The latched working registers are abnormal.	After users reset the CPU module or restore it to the factory setting, they can download the program and the parameter again.

ERROR Indicator Blinking

Error Code	Description	Corrective action
16#000A	Scan timeout (SM8: The watchdog timer error)	1. Check the setting of the watchdog timer in HWCONFIG. 2. Check whether the program causes the long scan time
16#000C	The program downloaded to the PLC is incorrect.	After users compile the program again, they can download the program again.
16#000E	The program or the parameter is being downloaded, and therefore the PLC can not run.	After the program or the parameter is downloaded to the PLC, users can try to run the PLC.
16#0018	The serial port is abnormal. (SM9)	Retry the connection. If the error still occurs, please contact the factory.
16#0019	The USB is abnormal. (SM9)	Retry the connection. If the error still occurs, please contact the factory.
16#001A	The contents of the system backup file (.dup file) are incorrect.	Create the system backup file again.
16#0033	The communication setting of COM1 is incorrect. (SM9)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0034	The setting of the station address of COM1 is incorrect. (SM9)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.

Error Code	Description	Corrective action
16#0035	The setting of the communication type of COM1 is incorrect. (SM9)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0038	The communication setting of COM2 is incorrect. (SM9)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0039	The setting of the station address of COM2 is incorrect. (SM9)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#003A	The setting of the communication type of COM2 is incorrect. (SM9)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#0066	An error occurs when the system is backed up.	1. Check whether the memory card is normal, and whether the capacity of the memory card is large enough. 2. Retry the backup procedure. If the error still occurs, please contact the factory.
16#0067	The size of the PLC parameters restored exceeds the size of the PLC parameters of the CPU module.	The error code is appeared to indicate alarm only.
16#2000	There is no END in the program in the PLC. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.
16#2001	The program is incorrect. There is a syntax error.	Check the program, compile the program again, and download the program again.
16#2002	GOEND is used incorrectly. (SM5)	Check the program, compile the program again, and download the program again.

Error Code	Description	Corrective action
16#2003	The devices used in the program exceed the range. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2004	The part of the program specified by the label used in CJ/JMP is incorrect, or the label is used repeatedly. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2005	The N value used in MC is not the same as the corresponding N value used in MCR, or the number of N values used in MC is not the same as the number of N values used in MCR. (SM5)	Check the program, compile the program again, and download the program again.
16#2006	The N values used in MC do not start from 0, or the N values used in MC are not continuous. (SM5)	Check the program, compile the program again, and download the program again.
16#2007	The operands used in ZRST are not used properly. (SM5)	Check the program, compile the program again, and download the program again.
16#200A	Invalid instruction (SM5)	Check the program, compile the program again, and download the program again.
16#200B	The operand **n** or the other constant operands exceed the range. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200C	The operands overlap. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200D	An error occurs when the binary number is converted into the binary-coded decimal number. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200E	The string does not end with 0x00. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#200F	The instruction does not support the modification by an index register. (SM5)	Check the program, compile the program again, and download the program again.

Error Code	Description	Corrective action
16#2010	1. The instruction does not support the device. 2. Encoding error 3. The instruction is a 16-bit instruction, but the constant operand is a 32-bit code. (SM5)	Check the program, compile the program again, and download the program again.
16#2011	The number of operands is incorrect. (SM5)	Check the program, compile the program again, and download the program again.
16#2012	Incorrect division operation (SM0/SM5).	Check the program, compile the program again, and download the program again.
16#2013	The value exceeds the range of values which can be represented by the floating-point numbers. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2014	The task designated by TKON/TKOFF is incorrect, or exceeds the range. (SM5)	Check the program, compile the program again, and download the program again.
16#2015	There are more than 32 levels of nested program structures supported by CALL. (SM0)	Check the program, compile the program again, and download the program again.
16#2016	There are more than 32 levels of nested program structures supported by FOR/NEXT. (SM0/SM5)	Check the program, compile the program again, and download the program again.
16#2017	The number of times FOR is used is different from the number of times NEXT is used. (SM5)	Check the program, compile the program again, and download the program again.
16#2018	There is a label after FEND, but there is no SRET. Or there is SRET, but there is no label. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.
16#2019	The interrupt task is not after FEND. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.

Error Code	Description	Corrective action
16#201A	IRET/SRET is not after FEND. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.
16#201B	There is an interrupt task, but there is no IRET. There is IRET, but there is not interrupt task. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.
16#201C	End is not at the end of the program. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.
16#201D	There is CALL, but there is no MAR. (SM5)	1. Compile the program again, and download the program again. 2. Reinstall ISPSoft, compile the program again, and download the program again.
16#201E	The function code used in MODRW is incorrect. (SM102/SM103)	Check the usage of the instruction and the setting of the operands. Please refer to the explanation of the instruction MODRW in AH500 Programming Manual for more information.
16#201F	The length of the data set in MODRW is incorrect. (SM102/SM103)	Check the usage of the instruction and the setting of the operands. Please refer to the explanation of the instruction MODRW in AH500 Programming Manual for more information.
16#2020	The communication command received by using MODRW is incorrect. (SM102/SM103)	Check whether the slave supports the function code and the specified operation.
16#2021	The checksum of the command received by using MODRW is incorrect. (SM102/SM103)	1. Check whether there is noise, and retry the sending of the command. 2. Check whether the slave operates normally.
16#2022	The format of the command used in MODRW does not conform to the ASCII format. (SM102/SM103)	Make sure that the format of the command conforms to the ASCII format.

Error Code	Description	Corrective action
16#2023	There is a communication timeout when MODRW is executed. (SM102/SM103)	Check whether the slave operates normally, and whether the connection is normal.
16#2024	The setting value of the communication timeout is invalid when RS is executed. (SM102/SM103)	1. Check the program and the related special data registers. 2. Set the communication port parameter for the CPU module in HWCONFIG again.
16#2025	There is a communication timeout when RS is executed. (SM102/SM103)	Check whether the slave operates normally, and whether the connection is normal.
16#2026	The interrupt number used in RS is incorrect.	Check whether the the interrupt service routine used in RS is downloaded.
16#2027	The execution of FWD is abnormal.	Please refer to AH500 Programming Manual, and check the instruction FWD.
16#2028	The execution of REV is abnormal.	Please refer to AH500 Programming Manual, and check the instruction REV.
16#2029	The execution of STOP is abnormal.	Please refer to AH500 Programming Manual, and check the instruction STOP.
16#202A	The execution of RSDT is abnormal.	Please refer to AH500 Programming Manual, and check the instruction RSDT.
16#202B	The execution of RSTEF is abnormal.	Please refer to AH500 Programming Manual, and check the instruction RSTEF.
16#202C | 16#204B	I/O interrupt service routine 0 does not exist. | I/O interrupt service routine 31 does not exist.	Download I/O interrupt service routine 0 (I/O interrupt 0) | Download I/O interrupt service routine 31 (I/O interrupt 31)
16#2054 | 16#2127	External interrupt service routine 40 does not exist. | External interrupt service routine 251 does not exist.	Download external interrupt service routine 40 (external interrupt 40) | Download external interrupt service routine 251 (external interrupt 251)

Error Code	Description	Corrective action
16#2128	An action in a sequential function chart is incorrectly assigned qualifiers related to time.(SM0/SM1)	Check whether qualifiers related to time are duplicated when setting SFC action.
16#2129	The modifier R is assigned to an action in a sequential function chart incorrectly. (SM0/SM1)	Check whether there are conflict settings between properties when setting SFC action.
16#3040	Data in E-CAM exceeds the setting range or does not exist.	**Cause**: the E-CAM number has been input in the function block but it exceeds the setting range **Action**: modify the E-CAM number and set it within the setting range
16#3100	Input parameters exceed the available setting range.	Check whether the input parameters exceed the available setting range.
16#3102	An error occurs in a sub-function block inside the function block.	Re-execute the function block instruction.
16#3103	The distance between the detecting sensors used for identifying exeptional bags is a negative value.	Check whether the positions of the detecting sensors are correct.
16#3104	Phasing is executed again before the previous phasing is completed.	**Cause**: the instruction is executed again when *Done* is still False. **Action**: re-execute the instruction again.
16#3105	Superimposing is executed again before the previous superimposing is completed.	**Cause**: the instruction is executed again when *Done* is still False. **Action**: re-execute the instruction again.
16#3106	Chain position compensation is triggered before the previous compensation is completed.	**Cause**: the master axis moves too fast to allow the previous compensation to be finished. In this case, the compensation is triggered again. **Action**: adjust all packaging related parameters according to the application requirements.
16#3107	Film axis position compensation is triggered before the previous compensation is completed.	**Cause**: the master axis moves too fast to allow the previous compensation to be finished. In this case, the compensation is triggered again. **Action**: adjust all packaging related parameters according to the application requirements.

A2

A2

Error Code	Description	Corrective action
16#3108	Knife position compensation is triggered before the previous compensation is completed.	**Cause**: the master axis moves too fast to allow the previous compensation to be finished. In this case, the compensation is triggered again. **Action**: adjust all packaging related parameters according to the application requirements..
16#3405	A negative or 0 is given to *Velocity*.	**Cause**: the value given to *Velocity* is negative or 0. **Action**: set the velocity to a positive value and re-execute the instruction.
16#340A	Homing mode setting error.	**Cause**: homing mode is not set to a value between 1 and 35. **Action**: set homing mode to a value between 1 and 35 and re-execute the instruction.
16#340B	Target distance is 0.	**Cause**: target distance of this instruction is not set to 0. **Action**: set target distance to a positive value and re-execute the instruction.
16#3410	User unit setting error; or the output pulse type setting error.	**Cause**: user unit setting of this instruction is not set to 0~2. **Action**: set the user unit to 0~2 and re-execute the instruction.
16#3411	Velocity factor overrides setting error.	**Cause**: velocity factor of this instruction is not set to 0~500. **Action**: set the velocity factor to 0~500 and re-execute the instruction.
16#3419	Master axis position is negative value.	**Cause**: master axis position is set to a negative value or 0. **Action**: set the master axis position to a positive value and re-execute the instruction.
16#341B	Maxmimum speed setting error.	**Cause**: maxmimum speed is not set to1~1,000,000. **Action**: set the maxmimum speed to 1~1,000,000 and re-execute the instruction.
16#343B	Error occurred in the other axis of the group	**Cause**: error occurred in the other axis of the group **Action:** use DFB_GroupReset to clear the error code

Error Code	Description	Corrective action
16#3461	The required communication parameters for PDO settings are not specified	**Cause**: the required communication parameters for PDO settings are not specified when the function block is in execution. **Action**: re-execute ECAT Builder and specify the required parameters for the function block.
16#3500	The axis is not in the ready state.	**Cause:** the axis is busy **Action:** stop the current operation or wait till the operation stops
16#3502	The position is not allowed for setup.	**Cause:** the target position of the function block exceeds the software limit or an axis of rotation or the position is an illegal one. **Action:** set up a new target position
16#3506	The axis is in “Coordinated”	**Cause**: the axis is in “Coordinated” when MC_stop is executed. **Action**: confirm that *Execute* =Fasle and *Done*=True in MC_Stop. Use DFB_GroupReset to reset the axis to “Standby” and use DFB_GroupDisable to disable the group motion.
16#3507	The axis is in “ErrorStop”	**Cause**: The axis is in “ErrorStop” when the instruction is executed. **Action**: use MC_Reset to reset the axis error status.
16#3508	The axis is not in “Standstill“	**Cause**: the axis is not in “Standstill” when the instruction is executed. **Action**: execute MC_Reset and confirm is the axis is in “Standstill”
16#3509	The axis is in the “Stopping“ state	**Cause:** the axis is in the “Stopping“ state. **Action:** close the MC_Stop function block and have the state go back to StandStill
16#350B	The acceleration time for the axis is too short.	**Cause:** the accleration time for the axis is set too short. **Action:** adjust the accleration or increase the time to acclerate for the axis.
16#350C	The deceleration time for the axis is too short.	**Cause:** the deceleration time for the axis is set too short. **Action:** adjust the deceleration or increase the time to decelerate for the axis.
16#350D	The E-CAM data exceeds the to-read range	**Cause:** the to-read data length exceeds the setting range **Action:** adjust the to-read data length (1~256)

Error Code	Description	Corrective action
16#350E	The E-CAM data exceeds the to-be written range.	**Cause:** the to-read data length exceeds the setting range **Action:** adjust the to-read data length (1~256)
16#350F	The axis is in the “Synchronized“ state.	**Cause:** the axis is in the “Synchronized“ state when the MC_SetOverride is started **Action:** do not have the axis in the “Synchronized“ state
16#3526	Error occurs in the previous movement of the axis	**Cause:** error occurs in the axis **Action:** clear error
16#3600	The state of axis is incorrect.	**Cause**: the axis is not in the ready state to execute the instruction. **Action**: this error will cause the axis state to be in ErrorStop; users need to execute MC_Reset to have the axis state back to StandStill. Check the state description and see if there are any contradictions.
16#3601	The limit of the number of buffering instructions is reached	**Cause**: the number of buffering instructions (with buffer mode enabled) reached 20. **Action**: 1. The error status will lead the axis to “ErrorStop”. In this case, execute MC_Reset to set the axis back to “Standstill”. 2. Make sure the total number of buffering instructions is less than 20 before executing current instruction.
16#3602	A multiple instructions which are not allowed to be executed at the same time are executed.	**Cause**: the instruction is executed when another instruction is in execution at the same time. (Both do not support simultaneously execution) **Action**: use MC_Reset to clear the axis error, and set the axis state to “StandStill.”
16#3603	Buffermode parameter setting error	**Cause**: the set value in *Buffermode* is not valid. **Action**:use MC_Reset to clear the axis error, and specify the input parameters again.
16#3604	Errors occur on the motion direction of the function block	**Cause**: the moving direction of the axis is not correct. **Action**: use MC_Reset to clear the axis error, and specify the input parameters again.

Error Code	Description	Corrective action
16#3605	P1 exceeds the available range	**Cause**: the target position is not specified with an available value. **Action**: use MC_Reset to clear the axis error, and specify the input parameters again.
16#3606	P2 exceeds the available range	**Cause**: the target position is not specified with an available value. **Action**: use MC_Reset to clear the axis error, and specify the input parameters again.
16#3607	V1 exceeds the available range	**Cause**: the target velocity is not specified with an available value. **Action**: use MC_Reset to clear the axis error, and specify the input parameters again.
16#3608	V2 exceeds the available range	**Cause**: the target velocity is not specified with an available value. **Action**: use MC_Reset to clear the axis error, and specify the input parameters again.
16#3612	It has reached the positive limit.	**Cause**: positive limit is reached. **Action**: use MC_Reset to clear the axis error, and move the position potively or negatively to the proprer position.
16#3613	It has reached the negative limit.	**Cause**: negative limit is reached. **Action**: use MC_Reset to clear the axis error, and move the position potively or negatively to the proprer position.
16#3614	The servo limit is exceeded.	**Cause**: the sevo drive limit is reached. **Action**: use MC_Reset to clear the axis error, and move the position potively or negatively to the proprer position.
16#3617	The acceleration exceeds the setting range.	**Cause:** when executing, the acceleration exceeds the maximum acceleration value, or buffering exceeds the maximum acceleration value. **Action:** set up the function block or the acceleration value
16#3618	The deceleration exceeds the setting range.	**Cause: :** when executing, the acceleration exceeds the maximum deceleration value, or buffering exceeds the maximum deceleration value. **Action:** set up the function block or the deceleration value

Error Code	Description	Corrective action
16#3800	Motion network disconnected during the execution of the instruction.	Check whether the network cable is detached or the network is disconnected.
16#3801	EtherCAT axis error occurs on the motion network	**Cause**: the motion axis reports an alarm or an error during the motion. **Action**: read the axis states and errors by using related function blocks, and reset the axis error by using MC_Reset.
16#3900	Failed to re-connect to the motion network.	**Cause**: After the motion network is reset, the CPU cannot re-connect to the motion network. **Action**: 1. Check whether the network cable is detached or the network is disconnected. 2. Check whether the connected servo drive is powered on.
16#3904	Motion network master can not read Slave parameters via SDO.	Check whether the parameter reading settings of Group and Parameter matches the available range of the servo drive.
16#3905	Motion network master can not write Slave parameters via SDO.	1. Check whether the parameter writing settings of Group and Parameter matches the available range of the servo drive. 2. Check whether the specified values to be written are within the available setting range for the parameters.
16#3906	Torque limit setting error in MC_SetTorqueLimit	**Cause**: the specified value for *PositiveValue* or *NegtiveValue* is invalid. **Action**: Check whether the specified value for *PositiveValue* or *NegtiveValue* is within available setting range of the servo drive.
16#3907	The function is not available for imaginary axes.	**Cause:** the function is not for imaginary axis. **Action:** change the imaginary axis to the real axis
16#3909	The motion network is currently executing other network functions.	Check the read/write status of SDO to see if the motion network is executing other network functions.
16#390C	Axis error occurs during the movement.	**Cause:** Axis error occurs during the movement. **Action:** the system will send the function block with the axis state back and users can learn what the error code is and then use MC_Rest to clear this error.

Error Code	Description	Corrective action
16#3910	Cancel the engagement when there is no engagement.	**Cause:** not executing the mc_gearin, mc_gearinpos, mc_combineaxes, but to execute mc_gearout **Action:** when the axis does not execute mc_gearin, mc_gearinpos, mc_combineaxes, do not execute mc_gearout
16#3911	Software limit error	**Cause**: the axis reached the software limit. **Action**: use MC_Reset to clear the error and use MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity or DFB_MPG to move the axis back to the proper range.
16#3912	The input contact of the funciton block exceeds the axis of rotation range	**Cause:** The input contact of the funciton block exceeds the axis of rotation range **Action:** modify the input to have it within the axis of rotation range
16#3913	Synchronization for engagement fails	**Cause:** before completing the engagement, the velocity of the main axis has changed **Action:** before completing the engagement, do not change the velocity of the main axis
16#3914	GearInPos velocity is set too small	**Cause:** maximum velocity of GearInPos has set too small **Action:** set a bigger maximum velocity
16#3915	GearInPos jerk is set too small	**Cause:** maximum jerk of GearInPos has set too small **Action:** set a bigger maximum jerk
16#3916	GearInPos engagement time is set too small	**Cause:** GearInPos engagement time too short **Action:** increase the MasterStartDistance
16#3917	GearInPos the velocity of the main axis is 0 when the engagement started	**Cause:** GearInPos the velocity of the main axis is 0 when the engagement started **Action:** not to set the velocit of the main axis 0
16#3918	The enagement velocity is larger than the AxisVelocityMax	**Cause:** The enagement velocity is larger than the AxisVelocityMax **Action:** modify the maximum of the axis velocity
16#3919	GearInPos the main axis moves in opposite direction	**Cause:** GearInPos the main axis moves in opposite direction **Action:** make the main axis and the auxiliary axis move in the same direction
16#3920	GearInPos acceleration is set too small	**Cause:** the acceleration or deceleration is set too small **Action:** increase the acceleration or deceleration
16#3921	GearInPosMasterStartDist out of range	**Cause:** MasterStartDist setting out of range **Action:** check the starting and ending position of the enagement
16#3922	GearInPos slave synchnoization position is set too small	**Cause:** SlaveSyncPosition is set too small **Action:** increase the setting value of SlaveSyncPosition, or increase the acceleration or deceleration

Error Code	Description	Corrective action
16#3923	GearInPos slave synchnoization position is set too big	**Cause:** SlaveSyncPosition is set too big **Action:** decrease the setting value of SlaveSyncPosition, or increase the acceleration or deceleration
16#3924	GearInPos the velocity of the main axis changes when the engagement started	**Cause:** before InSync, the velocity of the main axis changes **Action:** before InSync, fix the velocity of the main axis
16#3954	Torque ramp fail to write	**Cause:** TorqueRamp is set to 0. **Action:** TorqueRamp cannot be set as 0.
16#3955	Torque velocity fail to write	**Cause:** MC_TorqueControl value is not supported by the servo **Action:** check the servo manual to see the supported setting range
16#3A00	CAM table setting error	**Cause:** the designated CAM Table is not existed. **Action:** add a newCAM Table or set up a new setting in an existed CAM Table
16#3A01	CamIn master setting error	**Cause:** setting error in the source of the master axis **Action:** set up the source of the master axis again
16#3A02	CamIn CAM table changed too soon	**Cause:** CamIn change to another CAM table when the one started has not finished **Action:** enable the CAM function again
16#3A03	CamIn activation mode setting exceeds the available range	**Cause:** Activation Mode is not 0 or 1 **Action:** set up the setting vale and restart the CAM funciton
16#3A04	CamIn start mode setting exceeds the available range	**Cause:** CAM Start Mode setting value is not within the range 0~3 **Action:** set the CAM Start Mode value in the range
16#3A05	CamIn master scaling is set to 0.0	**Cause:** CAM Master Scaling is set to 0 **Action:** set the Master Scaling to a value other than 0 and restart CAM again
16#3A06	CamIn slave scaling is set to 0.0	**Cause:** CAM Slave Scaling is set to 0 **Action:** set the Slave Scaling to a value other than 0 and restart CAM
16#3A10	CamIn master start position is set too Small.	**Cause:** CAM master start positon is set too small **Action:** set the Start Position value bigger and restart CMA
16#3A13	CamIn the velocity is set too small.	**Cause:** CAM Start Mode velocity is set too small **Action:** set the velocity to a bigger value and restart CAM
16#3A15	CamIn jerk is set too small	**Cause:** the jerk for the CAM Start Mode is set too small **Action:** set the jerk value to a bigger value and restart CAM
16#3A16	CamIn maximum acceleration is set too small	**Cause:** CAM Start Mode maximum acceleration is set too small **Action:** set the maximum acceleration to a bigger value and restart CAM

Error Code	Description	Corrective action
16#3A17	CamIn Start mode distance is set too small	**Cause:** CAM Start Mode distance is set too small **Action:** set the Sync Position to a bigger value and restart CAM
16#3A18	CamIn Start mode distance is set too large	**Cause:** CAM Start Mode distance is set too large **Action:** set the Sync Position to a smaller value and restart CAM
16#3A19	Too many CamIn are wait to start	Cause: more than 5 sets of CAMs are waiting to start on the same axis Action: do not start all 5 CAMs at the same time
16#3A20	Master is moving in the negative direction.	**Cause:** Master is moving in the negative direction while the slave is in the middle of the CAM movement. **Action:** use MC_Reset to clear error
16#3A21	CamIn is cancelled when it is not in “CamIn” state.	Cause: designated Slave axis is not in “CamIn” state. Action: use MC_Reset to clear error
16#3D00	EtherCAT ENI file does not match current hardware configuration.	**Cause**: EtherCAT ENI file in the system does not match current EtherCAT configuration. **Action**: download again the ENI file that matches current EtherCAT configuration.
16#3D01	Slave lost in motoin network	**Cause**: slave lost during the motion network communication. **Action**: reconnect to the motion netowrk
16#3D03	EtherCAT DC time is set too small	**Cause:** EtherCAT DC time setting error 8-axis minimum 500us; 16-axis minimum 1000us; 32-axis minimum 2000us **Action**: check the current axis number and set up the DC time again
16#6001	Illegal IP address (SM1107)	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6002	Illegal netmask address	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.

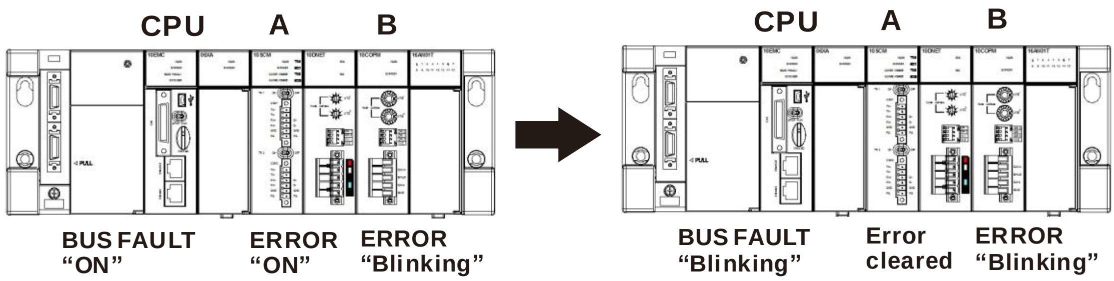
A2

Error Code	Description	Corrective action
16#6003	Illegal gateway mask	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6004	The IP address filter is set incorrectly.	Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6006	The static ARP table is set incorrectly.	Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#8242	Gcode Data ERROR	1. Use ISPSoft to download Gcode again
16#8243	ECAM Data ERROR	1. Use ISPSoft to download internal E-CAM again
16#8244	ENI Data ERROR	1. Use ECAT Builder to download the Ethernet parameter again
16#8245	EtherCat Data ERROR	1. Use ECAT Builder to download the Ethernet parameter again
16#8246	Axes Parameters Data ERROR	1. Use ISPSoft to download axes parameters again
16#8247	External Gcode Data ERROR	1. Make sure the SD card is installed and use ISPSoft to download the external Gcode

A2

BUS FAULT Indicator ON

The BUS FAULT indicator on the CPU would be ON to indicate an error on CPU, or to indicate an error on I/O module together with the ERROR indicator on an I/O module. If an error occurs in an I/O module, the status of the BUS FAULT indicator on the CPU will be the same as that of the ERROR indicator on the I/O module. If multiple errors occur in the I/O modules, the BUS FAULT indicator on the CPU will keep ON (not blinking). For example, if the ERROR indicator on module A is ON and the ERROR indicator on module B blinks, the BUS FAULT indicator will keep ON. When the error in I/O module A is cleared, module B will blink and the BUS FAULT indicator will blink as well. Refer to the section **A.5.1** for more information about the indicator behaviors of each module.



You can get the corrective actions for the CPU errors indicated by the BUS FAULT indicator from the table below. If the error code you obtained is not listed in the table below, you can check if an error occurs on the I/O modules. Refer to the following content of this section for more information about the troubleshooting for I/O modules.

Error Code	Description	Corrective action
16#0013	The I/O module can not run/stop. (SM10)	Check whether the setting of the parameter for the module is correct. If the setting is correct, please check whether the module breaks down. If the error still occurs, please contact the manufacturer.
16#0014	The procedure of restoring the system can not be executed. (SM9)	The contents of the system backup file are incorrect, or the file does not exist in the path specified. If the file exists and the procedure of restoring the system can not be executed, please backing up the system again. If the error still occurs, please contact the manufacturer.
16#1401	An error occurs when the data in the I/O module is accessed. (SM9)	Please contact the factory.
16#1402	The actual arrangement of the I/O modules is not consistent with the module table. (SM9)	Check whether the module table in HWCONFIG is consistent with the actual arrangement of the I/O modules.
16#1403	An error occurs when the data is read from the module. (SM9)	Check whether the module operates normally. If the error still occurs, please contact the factory.
16#1405	The setting parameter of the module is not found. (SM9)	Set the parameter in HWCONFIG again, and download it.
16#140B	The number of network modules exceeds the limit. (SM9)	Please decrease the number of network modules to the number supported by the system.
16#140C	The checksum of the high-speed data exchange is incorrect	Please check the version of the firmware installed on the module, and contact the factory.
16#140D	The ID of the actual power supply module is not the same as the ID of the power supply module set in HWCONFIG.	Check whether the power supply configuration in HWCONFIG is consistenet with the actial arrangement of the power supply module.
16#140E	The amount of data exchanged at a high speed exceeds the maximum amount supported.	Check the firmware version and contact the supplier.
16#140F	High-speed data exchange error	Check the firmware version and contact the supplier.

A2

BUS FAULT Indicator Blinking

If the BUS FAULT indicator blinks, check the operating state of the module. Refer to sections **A.5.1** for more information about the indicators behaviors of each module, and refer to the following content of this section for more information about the troubleshooting for I/O modules.

Others

Error Code	Description	Corrective action
16#000F	The original program in the PLC is damaged.	After users compile the program again, they can download the program again.
16#0024	There is no IO module on the backplane.	Check whether the IO module is on the backplane.
16#005E	The memory card is initialized incorrectly. (SM453)	Check whether the memory card breaks down.
16#005F	The file to be read does not exist in the memory card; or the file directory to write in a file does not exist. (SM453)	Check whether the file name and file directory is correct.
16#0061	The storage capacity of the memory card is not enough. (SM453)	Check whether the storage capacity of the memory card is enough, or whether the memory card breaks down.
16#0062	The memory card is write-protected. (SM453)	Check whether the memory card is write-protected.
16#0063	An error occurs when the data is written into the memory card. (SM453)	Check whether the file path is correct, or whether the memory card breaks down.
16#0064	The file in the memory card can not be read. (SM453)	Check whether the file path is correct, or whether the file is damaged.
16#0065	The file in the memory card is a read-only file. (SM453)	Users need to set the file so that the file is not a read-only file.
16#1801	There is no interrupt service routine in the CPU module.	Check whether a corresponding interrupt service routine is created in the PLC program (24V LV Detection)
16#3400	Axis setting error (1~32)	**Cause:** motion axis number is not between 1 and 32 **Action:** set the axis number between 1 and 32 and re-execute the function block

Error Code	Description	Corrective action
16#3401	SDO DataType setting error (0~199)	**Cause:** data type cannot be matched with the object library **Action:** confirm the object library of the slave station
16#3404	The number of the channel exceeds the available setting range	**Cause:** the input channel exceeds the setting range **Action:** set up the channel number for input again and re-execute the function block
16#3414	Pulse type counter setting error	**Cause:** the pulse type range is set other than 0~3 **Action:** set up the pulse type range and re-execute the function block
16#3415	Comparison condition setting error	**Cause:** the comparison condition is set other than 0~2 **Action:** set up the comparison condition again and re-execute the function block
16#3429	Gcode format error	**Cause:** G code file contains unsupported G code or the format is wrong **Action:** check the G code file contents and replace the unsupported G code with a supported one or fix the G code format. After that re-download the G code file.
16#342A	Gcode program source error	**Cause**: the designated G code file cannot be found in the AH Motion Controller PLC or the external SD card. **Action:** set up the Gcode ID in the function block and make sure the designated file is in the AH Motion Contoller PLC or the external SD card. Re-execute the function block.
16#342B	GcodeID setting error	**Cause:** the value of the Gcode ID is not 1 ~ 136 **Action:** set up the GcodeID between 1 and 136 and re-execute the function block.
16#342C	Gcode is in operation	**Cause:** the corresponding axis is executing DFB_GroupGcodeRun. **Action:** after the DFB_GroupGcodeRun is complete, re-execute the function block.
16#342D	Gcode grammar is being checked.	**Cause:** the corresponding axis is executing DFB_GroupGcodeSyntax. Action: after the DFB_GroupGcodeRun is complete, re-execute the function block.
16#342E	The setting of the Gcode Filter is out of the range.	**Cause:** The setting of the Gcode Filter is out of the range, over 1000 or less than 0. **Action:** set up a reasonable value in Gcode Filter and re-execute the function block.
16#3430	GroupNum already exists.	Cause: GroupNum already exists. Action: use DFB_GroupReset to clear this error.

A2

Error Code	Description	Corrective action
16#3431	Motion axis number is used repeatly in the same group.	**Cause:** DFB_GroupEnable, one of the AxisNumorder1~ AxisNumorder6 is used repeatly in the same group. **Action:** set up the AxisNumorder again and re-execute the function block
16#3432	The specified group number does not exist.	**Cause:** The specified group number does not exist. **Action:** set up an valid group name and enable it.
16#3433	The number of axes is insufficient for the specified group axes motion.	**Cause:** The number of axes is insufficient for the specified group axes motion. Action: set up the group name of the function block, the axis number should be as many as required, for example, a liner interpolation motion requires 2 axes while an arc interpolation motion requires 3 axes to complete the task.
16#3434	DFB_GroupDisable is executed when group motion is in progress.	**Cause:** DFB_GroupDisable is executed when group motion is in progress. **Action:** use DFB_GroupReset to clear this error.
16#3435	The same motion axis number is used repeatly in the different groups.	**Cause:** DFB_GroupEnable, one of the AxisNumorder1~ AxisNumorder6 is used repeatly in the different groups. **Action:** set up the AxisNumorder again and re-execute the function block.
16#3436	The axis number of the first order should be a positive number or a number other than 0.	**Cause:** AxisNumorder1 is a negative number or zero in DFB_GroupEnable. **Action:** set up the value of AxisNumorder1 again and re-execute the function block.
16#3437	The group number exceeds the setting range.	**Cause:** the value in GroupNum is not in the range of 1~16. **Action:** set up the value of GroupNum again and re-execute the function block.
16#3438	The designated group is in “ErrorStop” state.	**Cause:** the designated group is in “ErrorStop” state. **Action:** use DFB_GroupReset to clear the error
16#343A	Group is executing the function block ImmediateStop.	**Cause:** group is executing the function block ImmediateStop. **Action:** use DFB_GroupReset to clear the error
16#3463	The designated ECAT Slave does not exist.	**Cause:** the designated ECAT Slave does not exist. **Action:** make sure the designated ECAT Slave can perform axis operation.
16#3501	The selected channel has been used in FB.	**Cause:** the channel to be set has already been used **Action:** select a channel that is not used or free the used channel for setup
16#3505	An error occurs when writing CAM data.	**Cause:** read the CAM data but to find the data is not as it is written. **Action:** re-execute the CAM function block to write
16#3512	CAM data does not exist.	**Cause:** CAM data does not exist. **Action:** make sure the CAM data is correct and download the CAM data again.

A2

Error Code	Description	Corrective action
16#3619	The station address does not exist.	**Cause:** The station address does not exist. **Action:** check the address is existed and clear the error and re-execute the function block.
16#3620	The schedule buffer section of SDO is full.	**Cause:** The schedule buffer section of SDO is full. **Action:** wait till the schedule buffer secton of SDO is less full and re-execute the function block.
16#3622	SDO OD data type is not matched.	**Cause:** SDO OD data type is not matched. **Action:** check the OD data type is correct and re-execute the function block.
16#3623	SDO is overtime.	**Cause:** SDO is overtime. **Action:** check the connection and re-execute the function block.
16#3624	SDO data written error	**Cause:** error occurs in the slave **Action:** solve the problem and re-execute the function block
16#3625	SDO data reading error	**Cause:** error occurs in the slave **Action:** solve the problem and re-execute the function block
16#3626	SDO retry time exceeds the setting range.	**Cause:** SDO retry time exceeds the setting range. **Action:** check the address and re-execute the function block
16#3950	Capture cannot be used when the pulse speed is beyond 1MHz.	**Cause:** Capture cannot be used when the pulse speed is beyond 1MHz **Action:** slow dow the pulse speed and re-execute the function block
16#3951	CamCurve wrong input (for example, Concatenate cannot be true or other parameters are out of range.)	**Cause:** CamCurve contact input parameters are out of range **Action:** (1) turn the PLC off and then on, and execute the DFB_CamCure2 and set concatenate option to false. After the execution of the function block is done, set the concatenate to true and then execute the DFB_CamCure2. (2) check the other parameters to see if they are reasonable
16#3953	Capture uses the same channel number repeatedly.	**Cause:** DFB-Capture FB uses the same channel number repeatedly at the same time. **Action:** use other unused channel number instead.
16#600D	The RJ45 port is not connected. (SM1100)	**Cause:** RJ45 prot is not connected. **Action:** check the communication cable
16#620D	The length of the data which needs to be sent in a UDP Socket Configuration window is illegal.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6212	There is no response from the remote device after the timeout period.	Make sure that the remote device is connected.

Error Code	Description	Corrective action
16#6213	The data received exceeds the limit.	1. Check the program and the related special data registers. 2. Set the Ethernet parameter for the CPU module in HWCONFIG again.
16#6214	The remote device refuses the connection.	Make sure that the remote device operates normally.
16#6400	The number of TCP connections reaches the upper limit, or the flag which is related to the sending of the data is not set to ON.	1. Check whether the flag which is related to the sending of the data in the program is modified. 2. Retry the setting of the flag and the sending of the packet.
16#6401	The remote device aborts the connection.	Check whether the remote device support the MODBUS port (502).
16#6402	There is no response from the remote device after the timeout period.	Check whether the remote device operate normally.
16#6403	The remote IP address used in the applied instruction is illegal.	Check whether the program is correct.
16#6404	The MODBUS function code not supported is received.	Check the command transmitted from the remote device.
16#6405	The number of data which will be received is not consistent with the actual length of the data.	Check the command transmitted from the remote device.
16#6501	The remote device involved in the data exchange does not respond after the timeout period. (SM828~SM955)	Check the device whose connection number corresponds to the error flag, and check whether it is connected normally.
16#6502	The remote device involved in the data exchange does not respond correctly. (SM828~SM955)	Check the device whose connection number corresponds to the error flag, and check whether it is connected normally.
16#6700	MODBUS TCP data exchange initialization error	Check the setting value and download the data again.
16#6701	MODBUS TCP data exchange timeout	Confirm if the device to be connected supports MODBUS communication protocol.
16#6702	MODBUS TCP data receiving error	Confirm if the device to be connected supports MODBUS communication protocol.

Error Code	Description	Corrective action
16#7002	This function is not available for CPU modules.	Check the CPU firmware version.
16#7203	Invalid access code	Check the content of the packet sent by the device to be connected.
16#7401	Function code error	Check the content of the packet sent by the device to be connected.
16#7402	The packet exceeds the max. data length.	Check the content of the packet sent by the device to be connected.
16#7407	Non-ASCII characters exist in the command.	Check the content of the packet sent by the device to be connected.
16#7408	PLC is in RUN mode	Data download for program or CPU parameters is not allowed when PLC is in RUN mode.
16#740A	The CPU memory is being written or failed to be written.	Flash/SD card is being written. Please try again later.
16#740B	The Clear or Reset operation is in progress.	The RST/CLR operation is in progress. Please try again later.
16#740C	The backplane number in a communication command is incorrect.	Please check the PLC firmware and the software version and contact the supplier.
16#740D	The slot number in a communication command is incorrect.	Please check the PLC firmware and the software version and contact the supplier.
16#740E	Error occurs when clearing the memory	Please try again. If the error occurs again, contact the supplier.
16#740F	Communication timeout	Check if the device to be connected is in normal operation.
16#7410	The received Function Code doesn’t match the current Function Code.	Check the packet content sent by the remote device.
16#7412	Data cannot be downloaded to CPU because SW1 is ON.	Confirm that SW1 is OFF.
16#757D	The number of times users can enter the PLC password is 0.	The password retry limit is reached. Please power on the PLC again.
16#757E	Incorrect PLC password	Check if the password is correct.

Error Code	Description	Corrective action
16#8105	The contents of the program downloaded are incorrect. The program syntax is incorrect.	Download the program again.
16#8106	The contents of the program downloaded are incorrect. The length of the execution code exceeds the limit.	Download the program again.
16#8107	The contents of the program downloaded are incorrect. The length of the source code exceeds the limit.	Download the program again.
16#8230	The CPU parameter downloaded is incorrect. The IP address is illegal.	Check the network related parameter which is downloaded to the CPU.
16#8231	The CPU parameter downloaded is incorrect. The netmask address is illegal.	Check the network related parameter which is downloaded to the CPU.
16#8232	The CPU parameter downloaded is incorrect. The gateway address is illegal.	Check the network related parameter which is downloaded to the CPU.
16#8233	The CPU parameter downloaded is incorrect. The IP address filter is set incorrectly.	Check the network related parameter which is downloaded to the CPU.
16#8235	The CPU parameter downloaded is incorrect. The static ARP table is set incorrectly.	1. Check the Ethernet parameters of the CPU in HWCONFIG. 2. Check if CPU firmware version matches the HWCONFIG version
16#8236	The CPU parameter downloaded is incorrect: wrong NTP settings	1. Check the Ethernet parameters of the CPU in HWCONFIG. 2. Check if CPU firmware version matches the HWCONFIG version
16#8240	The CPU parameter downloaded is incorrect: Ether iLink	Redownload the parameters after modifying the configurations
16#8522	Auto scanning is in prograss	Auto scanning of module configuration is in progress. Please try again later.
16#853B	An I/O module is not configured.(wirte error)	Check if the module configuration in HWCONFIG is correct.

Error Code	Description	Corrective action
16#853C	An I/O module does not exist. (wirte error)	Check if the module configuration in HWCONFIG is correct.
16#854B	An I/O module is not configured. (read error)	Check if the module configuration in HWCONFIG is correct.
16#854C	An I/O module does not exist. (read error)	Check if the module configuration in HWCONFIG is correct.
16#8572	The checksum of the module configuration table is incorrect.	Please check the PLC firmware and the software version and contact the supplier.
16#8576	The checksum of the module parameter setting is incorrect.	Please check the PLC firmware and the software version and contact the supplier.
16#867A	The checksum of the module parameter mapping table is incorrect.	Please check the PLC firmware and the software version and contact the supplier.
16#85E1	An I/O interrupt number is incorrect.	Please check the PLC firmware version and contact the supplier.
16#85E2	An I/O interrupt service routine does not exist.	Please check if the corresponding interrupt program for the CPU is downloaded.
16#860F	System restoration error	The contents of the system backup file are incorrect, or the file does not exist in the path specified. If the file exists and the procedure of restoring the system can not be executed, please backing up the system again. If the error still occurs, please contact the manufacturer.
16#8611	No memory card exists, or the memory card format is incorrect.	The system cannot detect the memory card. Format the memory card and try again.
16#9A33	An error occurs when COM1 communicates with slave 19 by Modbus or PLC Link.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9A34	An error occurs when COM1 communicates with slave 20 by Modbus or PLC Link.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9A35	An error occurs when COM1 communicates with slave 21 by Modbus or PLC Link.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9A47	COM1 receives no response from slave 7 by Modbus or PLC Link. (SM1591)	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B01	An error occurs when the Modbus connection of COM2 is initialized.	Reset the parameters of Modbus conection of COM2 in the HWCONFIG.
16#9B21	An error occurs when COM2 communicates with slave 1 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B22	An error occurs when COM2 communicates with slave 2 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B23	An error occurs when COM2 communicates with slave 3 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B24	An error occurs when COM2 communicates with slave 4 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B25	An error occurs when COM2 communicates with slave 5 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B26	An error occurs when COM2 communicates with slave 6 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B27	An error occurs when COM2 communicates with slave 7 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B28	An error occurs when COM2 communicates with slave 8 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B29	An error occurs when COM2 communicates with slave 9 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B2A	An error occurs when COM2 communicates with slave 10 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2B	An error occurs when COM2 communicates with slave 11 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2C	An error occurs when COM2 communicates with slave 12 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2D	An error occurs when COM2 communicates with slave 13 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2E	An error occurs when COM2 communicates with slave 14 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B2F	An error occurs when COM2 communicates with slave 15 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B30	An error occurs when COM2 communicates with slave 16 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B31	An error occurs when COM2 communicates with slave 17 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B32	An error occurs when COM2 communicates with slave 18 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B33	An error occurs when COM2 communicates with slave 19 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B34	An error occurs when COM2 communicates with slave 20 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B35	An error occurs when COM2 communicates with slave 21 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B36	An error occurs when COM2 communicates with slave 22 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B37	An error occurs when COM2 communicates with slave 23 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B38	An error occurs when COM2 communicates with slave 24 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B39	An error occurs when COM2 communicates with slave 25 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3A	An error occurs when COM2 communicates with slave 26 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3B	An error occurs when COM2 communicates with slave 27 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3C	An error occurs when COM2 communicates with slave 28 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3D	An error occurs when COM2 communicates with slave 29 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B3E	An error occurs when COM2 communicates with slave 30 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B3F	An error occurs when COM2 communicates with slave 31 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B40	An error occurs when COM2 communicates with slave 32 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B41	COM2 receives no response from slave 1 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B42	COM2 receives no response from slave 2 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B43	COM2 receives no response from slave 3 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B44	COM2 receives no response from slave 4 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B45	COM2 receives no response from slave 5 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B46	COM2 receives no response from slave 6 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B47	COM2 receives no response from slave 7 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B48	COM2 receives no response from slave 8 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B49	COM2 receives no response from slave 9 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4A	COM2 receives no response from slave 10 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4B	COM2 receives no response from slave 11 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4C	COM2 receives no response from slave 12 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4D	COM2 receives no response from slave 13 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4E	COM2 receives no response from slave 14 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B4F	COM2 receives no response from slave 15 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B50	COM2 receives no response from slave 16 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B51	COM2 receives no response from slave 17 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B52	COM2 receives no response from slave 18 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B53	COM2 receives no response from slave 19 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B54	COM2 receives no response from slave 20 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B55	COM2 receives no response from slave 21 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B56	COM2 receives no response from slave 22 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B57	COM2 receives no response from slave 23 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B58	COM2 receives no response from slave 24 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B59	COM2 receives no response from slave 25 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5A	COM2 receives no response from slave 26 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5B	COM2 receives no response from slave 27 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Error Code	Description	Corrective action
16#9B5C	COM2 receives no response from slave 28 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5D	COM2 receives no response from slave 29 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5E	COM2 receives no response from slave 30 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B5F	COM2 receives no response from slave 31 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.
16#9B60	COM2 receives no response from slave 32 by MODBUS.	1. Check the communication setting between the connecting devices. 2. Check if the communication cable is damaged.

Analog I/O Modules and Temperature Measurement Modules

Error code	Description	Corrective action
16#A000	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 0 exceeds the range of inputs which can be received by the hardware.
16#A001	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 1 exceeds the range of inputs which can be received by the hardware.
16#A002	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 2 exceeds the range of inputs which can be received by the hardware.
16#A003	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 3 exceeds the range of inputs which can be received by the hardware.

Error code	Description	Corrective action
16#A004	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 4 exceeds the range of inputs which can be received by the hardware.
16#A005	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 5 exceeds the range of inputs which can be received by the hardware.
16#A006	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 6 exceeds the range of inputs which can be received by the hardware.
16#A007	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator blinks.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 7 exceeds the range of inputs which can be received by the hardware.
16#A400	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 0 exceeds the range of inputs which can be received by the hardware.
16#A401	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 1 exceeds the range of inputs which can be received by the hardware.
16#A402	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 2 exceeds the range of inputs which can be received by the hardware.
16#A403	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 3 exceeds the range of inputs which can be received by the hardware.
16#A404	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 4 exceeds the range of inputs which can be received by the hardware.
16#A405	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 5 exceeds the range of inputs which can be received by the hardware.

Error code	Description	Corrective action
16#A406	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 6 exceeds the range of inputs which can be received by the hardware.
16#A407	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is ON.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 7 exceeds the range of inputs which can be received by the hardware.
16#A600	Hardware failure	1. Check whether the backplane is normal. 2. Check whether the module operate normally.
16#A601	The external voltage is abnormal.	Check whether the external 24 V power supply to the module is normal.
16#A602	Internal error The CJC is abnormal.	Please contact the manufacturer.
16#A603	Internal error The factory correction is abnormal.	Please contact the manufacturer.
16#A800	The signal received by channel 0 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 0 exceeds the range of inputs which can be received by the hardware.
16#A801	The signal received by channel 1 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 1 exceeds the range of inputs which can be received by the hardware.
16#A802	The signal received by channel 2 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 2 exceeds the range of inputs which can be received by the hardware.
16#A803	The signal received by channel 3 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 3 exceeds the range of inputs which can be received by the hardware.
16#A804	The signal received by channel 4 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 4 exceeds the range of inputs which can be received by the hardware.
16#A805	The signal received by channel 5 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether The signal received by channel 5 exceeds the range of inputs which can be received by the hardware.

Error code	Description	Corrective action
16#A806	The signal received by channel 6 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 6 exceeds the range of inputs which can be received by the hardware.
16#A807	The signal received by channel 7 exceeds the range of inputs which can be received by the hardware. (The ERROR LED indicator is OFF.)	Check the module parameter in HWCONFIG. Check whether the signal received by channel 7 exceeds the range of inputs which can be received by the hardware.

AH02HC-5A/AH04HC-5A

Error code	Description	Corrective action
16#A001	The linear accumulation in channel 1 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR0 to ON by means of FROM/TO.
16#A002	The scale set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 1 should be in the range of 0 to 32767.
16#A003	The number of cycles set for channel 1exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 1 should be in the range of 2 to 60.
16#A004	The comparison value set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 1 should be in the range of -999999999 to 999999999.
16#A005	A limit value set for channel 1 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 1 should be in the range of -200000 to 200000.
16#A006	The interrupt number set for channel 1 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 1 should be in the range of 0 to 31.
16#A011	The linear accumulation in channel 1 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR28 to ON by means of FROM/TO.
16#A012	The scale set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 2 should be in the range of 0 to 32767.
16#A013	The number of cycles set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 2 should be in the range of 2 to 60.
16#A014	The comparison value set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 2 should be in the range of -999999999 to 999999999.

A2

Error code	Description	Corrective action
16#A015	A limit value set for channel 2 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 2 should be in the range of -200000 to 200000.
16#A016	The interrupt number set for channel 2 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 2 should be in the range of 0 to 31.
16#A021	The linear accumulation in channel 3 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR56 to ON by means of FROM/TO.
16#A022	The scale set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 3 should be in the range of 0 to 32767.
16#A023	The number of cycles set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 3 should be in the range of 2 to 60.
16#A024	The comparison value set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 3 should be in the range of -999999999 to 999999999.
16#A025	A limit value set for channel 3 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 3 should be in the range of -200000 to 200000.
16#A026	The interrupt number set for channel 3 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 3 should be in the range of 0 to 31.
16#A031	The linear accumulation in channel 4 exceeds the range.	To clear the linear accumulation, users need to set bit 1 in CR84 to ON by means of FROM/TO.
16#A032	The scale set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The scale set for channel 4 should be in the range of 0 to 32767.
16#A033	The number of cycles set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The number of cycles set for channel 4 should be in the range of 2 to 60.
16#A034	The comparison value set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The comparison value set for channel 4 should be in the range of -999999999 to 999999999.
16#A035	A limit value set for channel 4 is incorrect.	Check the module parameter in HWCONFIG. A limit value of set for channel 4 should be in the range of -200000 to 200000.
16#A036	The interrupt number set for channel 4 exceeds the range.	Check the module parameter in HWCONFIG. The interrupt number set for channel 4 should be in the range of 0 to 31.

AH05PM-5A/AH10PM-5A/AH15PM-5A

The programs and the setting which are mentioned in the table below are edited in PMSoft version 2.02 or above.

Error code	Description	Corrective action
16#A002	The subroutine has no data.	A program should be written in the subroutine.
16#A003	CJ, CJN, and JMP have no matching pointers.	Write the pointers which match CJ, CJN, and JMP respectively.
16#A004	There is a subroutine pointer in the main program.	The subroutine pointer can not be in the main program.
16#A005	Lack of the subroutine	The nonexistent subroutine can not be called.
16#A006	The pointer is used repeatedly in the same program.	The pointer can not be used repeatedly in the same program.
16#A007	The subroutine pointer is used repeatedly.	The subroutine pointer can not be used repeatedly.
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	The pointer used in JMP can not be used repeatedly in different subroutines.
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	The pointer used in JMP can not be the same as the pointer used in CALL.
16#A00A	The pointer used in JMP is the same as a subroutine pointer.	The pointer used in JMP can not be the same as a subroutine pointer.
16#A00B	Target position (I) of the single speed is incorrect.	The target position (I) of the single speed should be set correctly.
16#A00C	Target position (II) of the single-axis motion is incorrect.	Check whether target position (II) of the single-axis motion and target position (I) of the single-axis motion are in opposite directions.
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Set the speed of the single-axis motion.
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	The setting value can not be zero.
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Set the speed of returning to zero properly. (The setting value can not be zero.)
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Set the speed of returning to zero. The deceleration should be less than the speed of returning to zero. (The setting value can not be zero.)
16#A011	The setting of the JOG speed is incorrect.	The setting value can not be zero.
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.

Error code	Description	Corrective action
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A014	The limit switch is reached.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A015	The device which is used exceeds the device range.	Use the device which does not exceed the device range.
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Use the16-bit index register/32-bit index register which does not exceed the device range.
16#A018	The conversion into the floating-point number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A019	The conversion into the binary-coded decimal number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A01A	Incorrect division operation (The divisor is 0.)	Modify the operation to prevent the divisor from being zero.
16#A01B	General program error	Modify the program to make the syntax correct.
16#A01C	LD/LDI has been used more than nine times.	Modify the program to prevent LD/LDI from being used more than nine times.
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Modify the program to prevent more than one level of nested program structure supported by RPT/RPE from being used.
16#A01E	SRET is used between RPT and RPE.	Modify the program to prevent SRET from being used between RPT and RPE.
16#A01F	There is no M102 in the main program, or there is no M2 in the motion program.	Modify the program so that there is M102 in the main program, or modify the program so that there is M2 in the motion program.
16#A020	The wrong instruction is used, or the device used exceeds the range.	Check and modify the program to prevent the wrong instruction from being used, or check whether the device used exceeds the device range.

AH20MC-5A

The programs and the setting which are mentioned in the table below are edited in PMSoft version 2.02 or above.

Error code	Description	Corrective action
16#A002	The subroutine has no data.	A program should be written in the subroutine.
16#A003	CJ, CJN, and JMP have no matching pointers.	Write the pointers which match CJ, CJN, and JMP respectively.

Error code	Description	Corrective action
16#A004	There is a subroutine pointer in the main program.	The subroutine pointer can not be in the main program.
16#A005	Lack of the subroutine	The nonexistent subroutine can not be called.
16#A006	The pointer is used repeatedly in the same program.	The pointer can not be used repeatedly in the same program.
16#A007	The subroutine pointer is used repeatedly.	The subroutine pointer can not be used repeatedly.
16#A008	The pointer used in JMP is used repeatedly in different subroutines.	The pointer used in JMP can not be used repeatedly in different subroutines.
16#A009	The pointer used in JMP is the same as the pointer used in CALL.	The pointer used in JMP can not be the same as the pointer used in CALL.
16#A00A	The pointer used in JMP is the same as a subroutine pointer.	The pointer used in JMP can not be the same as a subroutine pointer.
16#A00B	Target position (I) of the single speed is incorrect.	The target position (I) of the single speed should be set correctly.
16#A00C	Target position (II) of the single-axis motion is incorrect.	Check whether target position (II) of the single-axis motion and target position (I) of the single-axis motion are in opposite directions.
16#A00D	The setting of speed (I) of the single-axis motion is incorrect.	Set the speed of the single-axis motion.
16#A00E	The setting of speed (II) of the single-axis motion is incorrect.	The setting value can not be zero.
16#A00F	The setting of the speed (V_{RT}) of returning to zero is incorrect.	Set the speed of returning to zero properly. (The setting value can not be zero.)
16#A010	The setting of the deceleration (V_{CR}) of returning to zero is incorrect.	Set the speed of returning to zero. The deceleration should be less than the speed of returning to zero. (The setting value can not be zero.)
16#A011	The setting of the JOG speed is incorrect.	The setting value can not be zero.
16#A012	The positive pulses generated by the single-axis clockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A013	The negative pulses generated by the single-axis counterclockwise motion are inhibited.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.
16#A014	The limit switch is reached.	The error occurs because the limit sensor is triggered. Check the status of the limit sensor, and check whether the motor operates normally.

A2

Error code	Description	Corrective action
16#A015	The device which is used exceeds the device range.	Use the device which does not exceed the device range.
16#A017	An error occurs when the device is modified by a 16-bit index register/32-bit index register.	Use the 16-bit index register/32-bit index register which does not exceed the device range.
16#A018	The conversion into the floating-point number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A019	The conversion into the binary-coded decimal number is incorrect.	Modify the operation to prevent the abnormal number from occurring.
16#A01A	Incorrect division operation (The divisor is 0.)	Modify the operation to prevent the divisor from being zero.
16#A01B	General program error	Modify the program to make the syntax correct.
16#A01C	LD/LDI has been used more than nine times.	Modify the program to prevent LD/LDI from being used more than nine times.
16#A01D	There is more than one level of nested program structure supported by RPT/RPE.	Modify the program to prevent more than one level of nested program structure supported by RPT/RPE from being used.
16#A01E	SRET is used between RPT and RPE.	Modify the program to prevent SRET from being used between RPT and RPE.
16#A01F	There is no M102 in the main program, or there is no M2 in the motion program.	Modify the program so that there is M102 in the main program, or modify the program so that there is M2 in the motion program.
16#A020	The wrong instruction is used, or the device used exceeds the range.	Check and modify the program to prevent the wrong instruction from being used, or check whether the device used exceeds the device range.

AH10EN-5A / AH15EN-5A

Error code	Description	Remedy
16#A001	The IP address of host 1 conflicts with another system on the network.	1. Contact the network administrator, and check whether the IP address is correct. 2. Check the module parameter in HWCONFIG.
16#A002	The IP address of host 2 conflicts with another system on the network.	1. Contact the network administrator, and check whether the IP address is correct. 2. Check the module parameter in HWCONFIG.
16#A003	DHCP for host 1 fails.	Please contact the network administrator
16#A004	DHCP for host 2 fails.	Please contact the network administrator
16#A401	Hardware error	Please restore the hardware to the factory setting. If the error still occurs, please contact the factory.
16#A402	The initialization of the system fails.	Please restore the system to the factory setting. If the error still occurs, please contact the factory.

AH10SCM-5A / AH15SCM-5A

Error code	Description	Corrective action
16#A002	The setting of the UD Link is incorrect, or the communication fails.	Check the setting in SCMSoft, and download the setting again.
16#A401	Hardware error	Please contact the manufacturer.
16#A804	The communication through the communication port is incorrect.	1. Check whether the communication cable is connected well. 2. Check the parameter in HWCONFIG, and the parameter. Download the parameter again.
16#A808	MODBUS communication error	1. Check whether the communication cable is connected well. 2. Check the parameter in HWCONFIG, and the parameter. Download the parameter again.

AH10DNET-5A

Error code	Description	Remedy
16#A0F0	The node ID of AH10DNET-5A is the same as other node ID on the network, or exceeds the range.	Make sure that the node ID of AH10DNET-5A is the only one on the network. If the node ID of AH10DNET-5A is not the only one on the network, please change the node ID, and supply power to AH10DNET-5 again.
16#A0F1	No slave is put on the scan list of AH10DNET-5A.	Put slaves on the scan list, and then download the scan list to AH10DNET-5A.
16#A0F2	The working voltage of AH10DNET-5A is low.	Check whether the working voltage of AH10DNET-5A and that of an AH500 series CPU module are normal.
16#A0F3	AH10DNET-5A enters the test mode.	Switch IN 1 on the module OFF, and supply power to AH10DNET-5A again.
16#A0F4	The bus of AH10DNET-5A becomes OFF.	1. Check whether the communication cable is normal, and whether the shielded cable is grounded. 2. Check whether the serial transmission speeds of other devices on the network are the same. 3. Check whether the both ends of the cable are connected to 121 Ω terminal resistors. 4. Supply power to AH10DNET-5A again.
16#A0F5	AH10DNET-5A detects that there is no power supply to the DeviceNet network.	Check whether the communication cable is normal, and whether the network power supply is normal.
16#A0F6	Something is wrong with the internal memory of AH10DNET-5A.	Supply power to AH10DNET-5A again. If the error still occurs, please contact the factory.

A2

Error code	Description	Remedy
16#A0F7	Something is wrong with the data exchange unit of AH10DNET-5A.	Supply power to AH10DNET-5A again. If the error still occurs, please contact the factory.
16#A0F8	The product ID of AH10DNET-5A is incorrect.	Supply power to AH10DNET-5A again. If the error still occurs, please contact the factory.
16#A0F9	An error occurs when the data is read from AH10DNET-5A, or when the data is written into AH10DNET-5A.	Supply power to AH10DNET-5A again. If the error still occurs, please contact the factory.
16#A0FA	The node ID of AH10DNET-5A is the same as that of the slave set on the scan list.	Method 1: Set the node ID of AH10DNET-5A again. The new node ID can not be the same as the node ID of the slave set on the scan list. Supply power to AH10DNET-5A again. Method 2: Put no slave on the scan list, and download the blank scan list to AH10DNET-5A through the simulated online mode in the software. Supply power to AH10DNET-5A again.
16#A0FB	The data exchange between AH10DNET and AH CPU failed.	Supply power to the AH10DNET and AH CPU and try to exchange data again. If the issue continuses, contact the factory.
16#A0FC	Errors occur in the slaves, on the module of an AHRTU-DNET backplane, or on the AHRTU-DNET backplane connection.	Check whether the node number has changed. Check if the network connection cable is secured and working fine. Check if the network transmission cable does not exceed the maximum communication distance (refer to AH500 module manual section 10.3.3 for more information). Do not exceed the maximum communication distance to ensure a stable network. Check if the module on the backplane is working fine. Check if the AHRTU-DNET backplane connection is working fine.

A2

AH10PFBM-5A

Error code	Description	Remedy
16#A001	The master is not set.	Download appropriate setting.
16#A003	The master station enters the test mode.	Just repower it.
16#A005	A timeout occurs when chips inside the master station communicate.	Download the appropriate configuration again. If the error still occurs, please contact the factory.
16#A00B	A timeout occurs when AH10PFBM-5A exchanges data exchange with a PLC.	Repower AH10PFBM-5A . If the error still occurs, please contact the factory.
16#A402	The PLC does not assign the I/O mapping area to the master.	Assign the appropriate I/O mapping area to the master via ISPSoft.

Error code	Description	Remedy
16#A404	Master initializing error	Contact the factory if the error still exists after repowering AH10PFBM-5A.
16#A406	Internal storage unit error	Contact the factory if the error still exists after repowering AH10PFBM-5A.
16#A407	Data exchange unit error	Contact the factory if the error still exists after repowering AH10PFBM-5A.
16#A408	Master serial number detection error	Contact the factory if the error still exists after repowering AH10PFBM-5A.
16#A4E2	The master detects that the slave is offline.	1. Check whether the PROFIBUS-DP bus connection is normal. 2. Check whether both of the ends of the network have terminal resistors.
16#A4E6	The master detects that an error occurs in the module connected to AHRTU-PFBS-5A.	Check the modules connected to AHRTU-PFBS-5A.

AH10PFBS-5A

Error code	Description	Remedy
16#A4F0	The node address of AH10PFBS-5A exceeds the valid range.	The node address of AH10PFBS-5A must be in the range of 1 to 125.
16#A4F1	Internal hardware error	If the error still exists after repowering AH10PFBS-5A, replace it with a new one.
16#A4F2	Parameter error	Check whether the GSD file AH10PFBS-5A is using is correct.
16#A4F3	Configuration error	Check whether the GSD file AH10PFBS-5A is using is correct.
16#A4F4	GPIO detection error	If the error still exists after repowering AH10PFBS-5A, replace it with a new one.
16#A4F5	AH10PFBS-5A enters the mode of factory test.	Repower AH10PFBS-5A after setting its node address between 1~125.
16#A4F6	1. AH10PFBS-5A has not been connected to the PROFIBUS-DP network. 2. PROFIBUS-DP master has not configured AH10PFBS-5A slave or the configured node address of AH10PFBS-5A is	1. Check whether the communication cable between AH10PFBS-5A and PROFIBUS-DP master is in normal status. 2. Ensure that AH10PFBS-5A slave has been configured to PROFIBUS-DP master and the configured node address of AH10PFBS-5A is consistent with that of the actually connected one. 3. Check whether the PROFIBUS-DP master works normally.

Error code	Description	Remedy
	inconsistent with that of the actually connected one.	

AH10COPM-5A

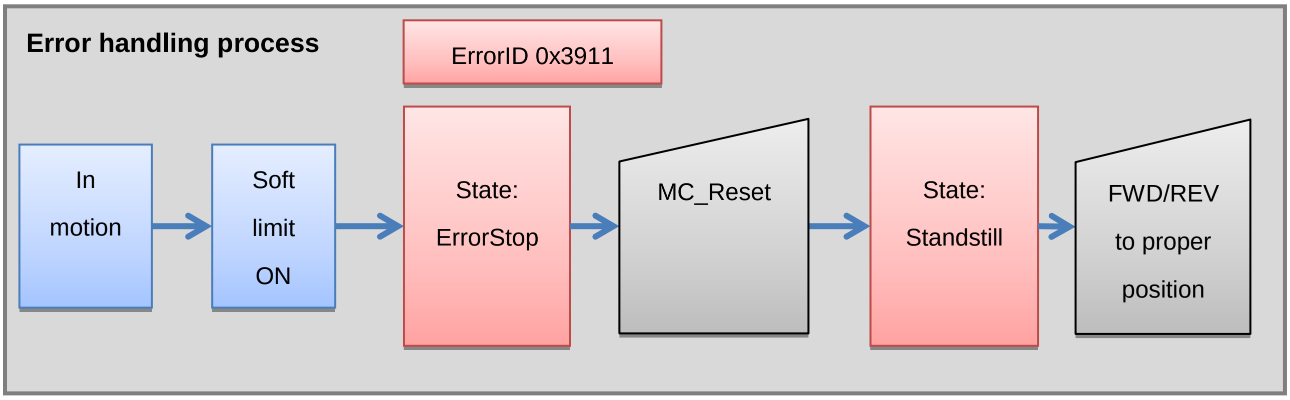
Error code	Description	Remedy
16#A0B0	AH10COPM-5A does not send a heartbeat message after a set period of time.	Check whether the bus cable on the CANopen network created is connected correctly.
16#A0B1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Set the length of the PDO in the slave station again, and then download the setting to AH10COPM-5A.
16#A0B2	The master station selected does not send a node guarding message after a set period of time.	Check whether the bus cable on the CANopen network created is connected correctly.
16#A0E0	AH10COPM-5A receives an emergency message from a slave station.	Use the function block CANopen_EMCY to read relevant information.
16#A0E1	The length of a PDO that a slave station sends is not the same as the length of the PDO set in the node list.	Set the length of the PDO in the slave station again, and then download the setting to AH10COPM-5A.
16#A0E2	AH10COPM-5A does not receive a PDO from a slave station.	Make sure that the PDOs in the slave station are set correctly.
16#A0E3	An automatic SDO is not downloaded successfully.	Make sure that the automatic SDO is et correctly.
16#A0E4	A PDO parameter is not set successfully.	Make sure that the setting of the PDO parameter is legal.
16#A0E5	A key parameter is set incorrectly.	Make sure that the slave stations connected are the same as the slave stations set.
16#A0E6	The actual network configuration is not the same as the network configuration set.	Make sure that the power supplied to the slave stations connected is normal and the network created is connected correctly.
16#A0E7	The control of the errors in a slave station is not sent after a set period of time.	
16#A0E8	The master station address is the same as a slave station address.	Set the master station address or the slave station address again, and make sure the new station address is not the same as a slave station address.

Error code	Description	Remedy
16#A0F1	No slave station is added to the node list in CANopen builder.	Add slave stations to the node list, and download the configuration to AH10COPM-5A.
16#A0F3	An error occurs in AH10COPM-5A.	Download parameters again. If the error still occurs, please replace AH10COPM-5A.
16#A0F4	The bus used is off.	Please check whether the bus cable on the CANopen network created is connected correctly, make sure that the serial transmission speeds of all the nodes on the network are the same, and power AH10COPM-5A again.
16#A0F5	The node address of AH10COPM-5A is set incorrectly.	The node address of AH10COPM-5A must be in the range of 1 to 127.
16#A0F6	Internal error: An error occurs in the manufacturing process in the factory.	Power AH10COPM-5A again. If the error still occurs, please replace AH10COPM-5A.
16#A0F7	Internal error: GPIO error	
16#A0F8	Hardware error	
16#A0F9	Low voltage	Make sure that the power supplied to AH10COPM-5A is normal.
16#A0FA	An error occurs in the firmware of AH10COPM-5A.	Power AH10COPM-5A again.
16#A0FB	The transmission registers in AH10COPM-5A are full.	Please make sure that the bus cable on the CANopen network created is connected correctly, and power AH10COPM-5A again.
16#A0FC	The reception registers in AH10COPM-5A are full.	Please make sure that the bus cable on the CANopen network created is connected correctly, and power AH10COPM-5A again.

A.2.3. Troubleshooting for Limitation Errors

Troubleshooting for the software limit errors

The controller system checks the software limits before or during the motion by the error code 0x3911. When the operation exceeds the software limits, the error code will be indicated and the axis will enter “ErrorStop”. Servo drive will not report this error since the error handling in this case is controlled by the controller. Note: for details on the softwaere limit setups, refert to ISPSoft manual. Do not set the values too close to the value of the software upper limit.



Troubleshooting for the hardware limit errors

When the servo drive is driving a motion, the servo will stop when CWL(Clockwise limit) or CCWL(Counterclockwise limit) is On, no matter it's running forward or reversely. AL014(CWL) or AL015(CCWL) will indicate such error.