Bus-Type Motion Controller

DVP15MC11T

DVP50MC11T

CANopen

EtherCAT

--- Motion Control

2017-10-27

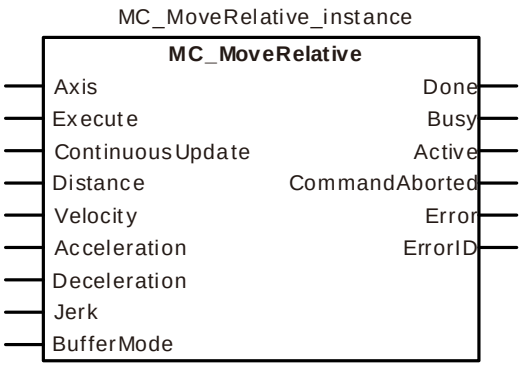
Table of Contents

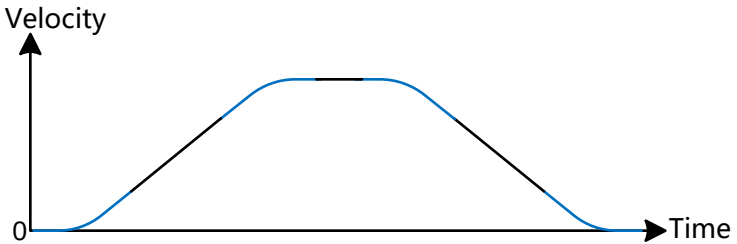
DVP15MC11T
DVP50MC11T
Motion Control Function

Motion Control Function

Overview

- DVP15MC11T is a **high-speed CANopen Bus motion controller** through the high integration of the conventional PLC and motion controller with the following functions.
  - Supports max. 24 real axes and max. 32 (real & virtual) axes in total.
  - Supports the external connection of the incremental encoder and absolute encoder, which can be used as virtual axes.
  - Supports 16-point high-speed input, which can be applied to the interrupt program and position capture.
  - Supports control function related to velocity, position, torque and homing.
  - Supports e-cam function; Max. 64 cam curves can be planned with max. 2048 key points for each curve.
  - Supports e-gear function including the one-master-axis gear and the two-master-axis-combined gear.
  - Supports application function such as rotary cut.
  - Supports G codes.
  - The motion instructions follow the PLCopen2.0 standard.





Motion Control Principle

Synchronization Mechanism and Servo Control

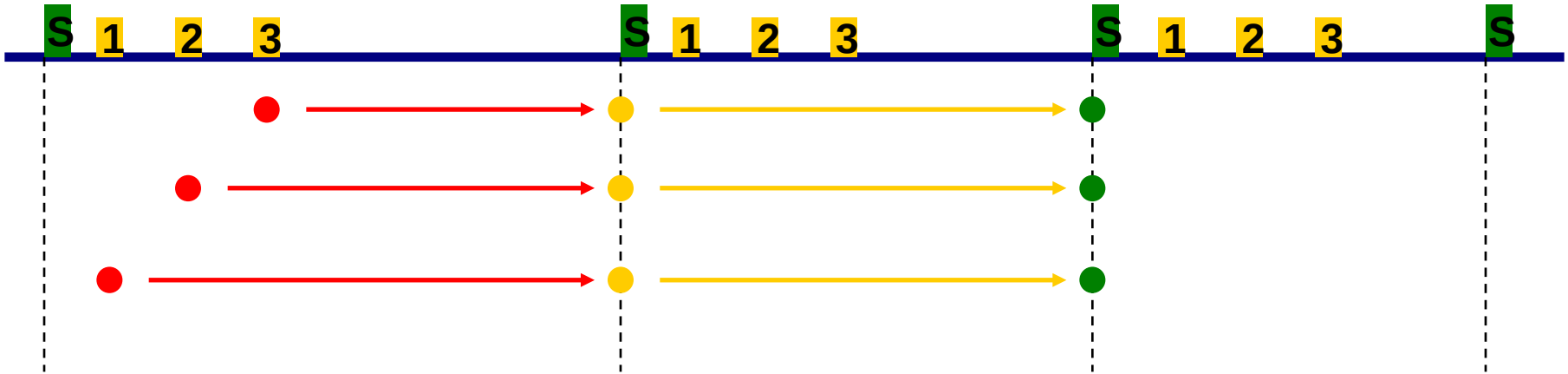
S SYNCH signal

1 The timing for servo drive 1 to receive the command

New control command

New control command is activated

Servo drive reaches command position



Motion Control Principle

Position Interpolation Principle

S SYNCH signal

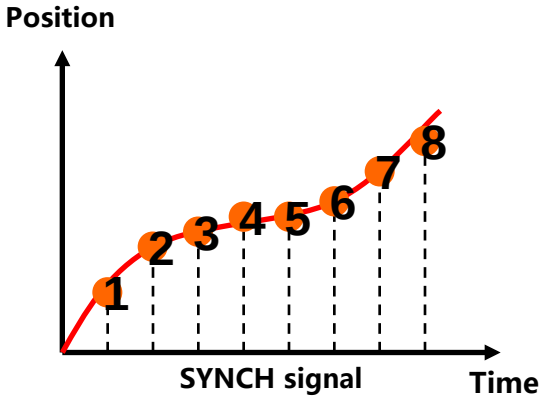
1 The timing for drive 1 to receive the command

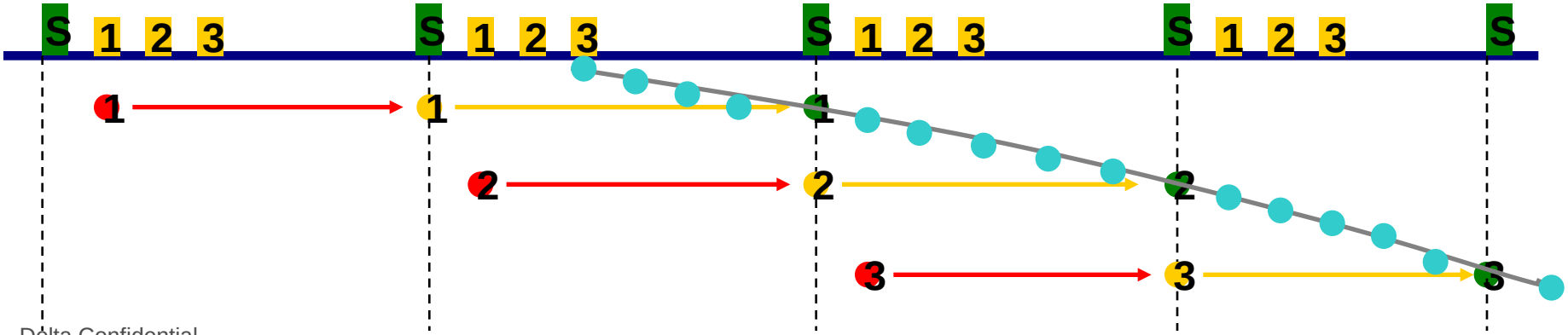
New control command

New control command is activated.

Drive reaches command position.

Command point precisely interpolated by drive (125us)





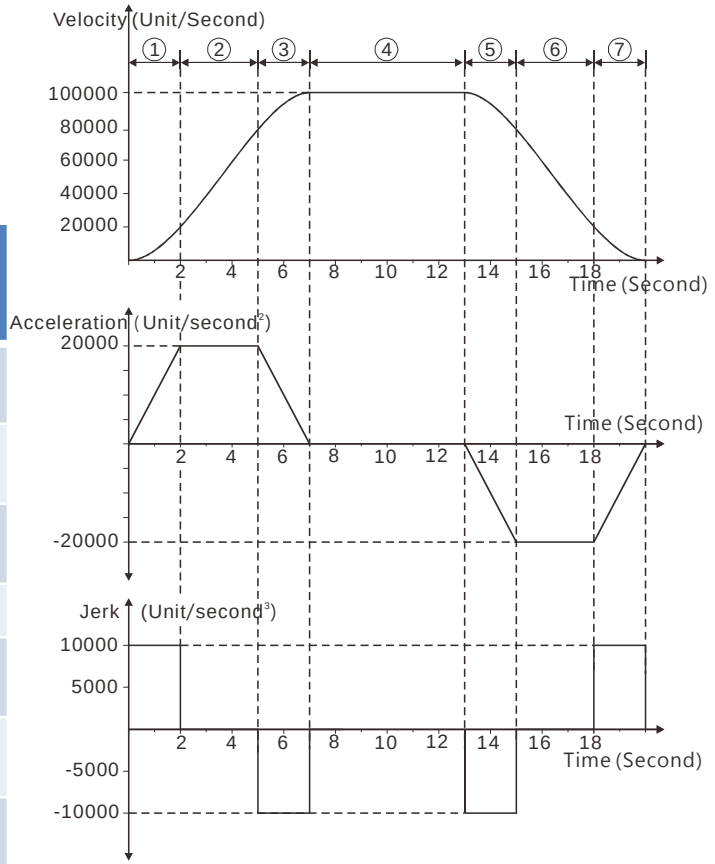
Kinematic Principle

Example

- When an axis runs under the control of DVP15MC11T, its kinematic equation which is the relationship among velocity, acceleration/deceleration and jerk is shown as below.

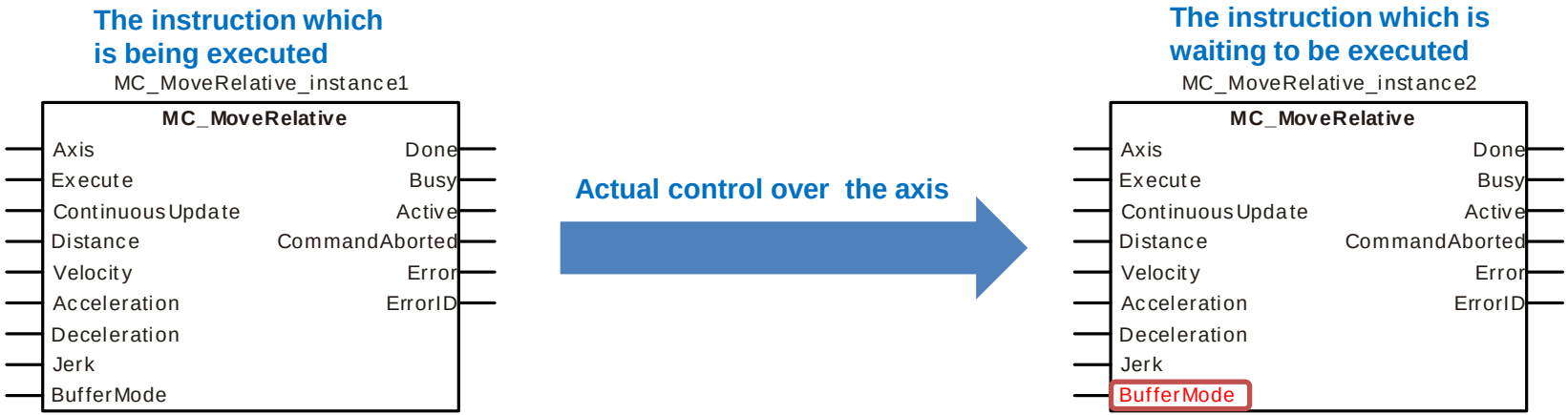
$$Acc(Dec) = v\,dt$$
$$Jerk = Acc\,dt$$

Stage No.	Time (s)	Jerk (Unit/s^3)	Acceleration/ Deceleration (Unit/s^2)	Velocity (Unit/s)	Motion type
1	0~2	10000	Acceleration is increased to 20000	Increasing	The acceleration motion with an increasing acceleration
2	2~5	0	Acceleration stays at 20000	Increasing	The acceleration motion with a constant acceleration
3	5~7	-10000	Acceleration is decreased to 0	Increases to 100000	The acceleration motion with an decreasing acceleration
4	7~13	0	0	100000	The motion at a constant speed
5	13~15	-10000	Deceleration is increased to 20000	Decreasing	The deceleration motion with an increasing deceleration
6	15~18	0	-20000	Decreasing	The deceleration motion with a constant deceleration
7	18~20	10000	Deceleration is decreased to 0	Decreases to 0	The deceleration motion with a decreasing deceleration



Function of Buffer_Mode

- While an axis is moving under the control of a motion control instruction, another motion instruction can be triggered to execute. There are six buffer modes for selection between the two instructions.
- The buffer mode can be specified by the value of *Buffer_Mode* of the next instruction which is waiting to be executed.



BufferMode

Overview of BufferMode

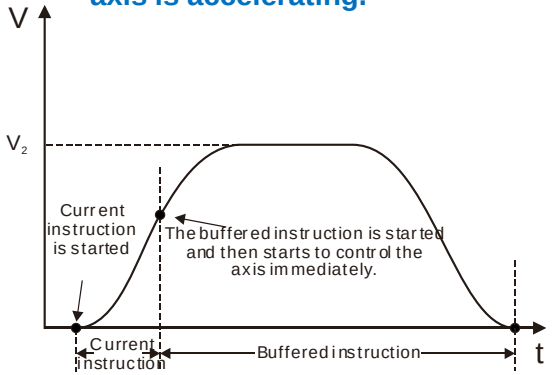
Buffer Mode	Description
0：mcAborting (Aborting)	The instruction being executed is aborted immediately.
1：mcBuffered (Buffered)	The buffered instruction just starts to control the axis after the current instruction execution is completed.
2：mcBlendingLow (Blend with low)	The buffered instruction just starts to control the axis after the target position of the current instruction is reached. The transit velocity is the lower of the target velocities of the current instruction and buffered instruction.
3：mcBlendingPrevious (Blend with previous)	The buffered instruction just starts to control the axis after the target position of the current instruction is reached. The transit velocity is the target velocity of the current instruction.
4：mcBlendingNext (Blend with next)	The buffered instruction just starts to control the axis after the target position of the current instruction is reached. The transit velocity is the target velocity of the buffered instruction.
5：mcBlendingHigh (Blend with high)	The buffered instruction just starts to control the axis after the target position of the current instruction is reached. The transit velocity is the higher of the target velocities of the current instruction and buffered instruction.

Explanation of BufferMode

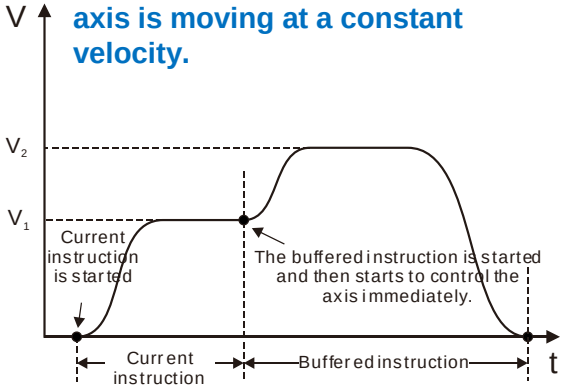
Buffer_Mode=mcAborting

- When *BufferMode* of another instruction is mcAborting, **the currently executed instruction will be aborted immediately** when the instruction is executed.

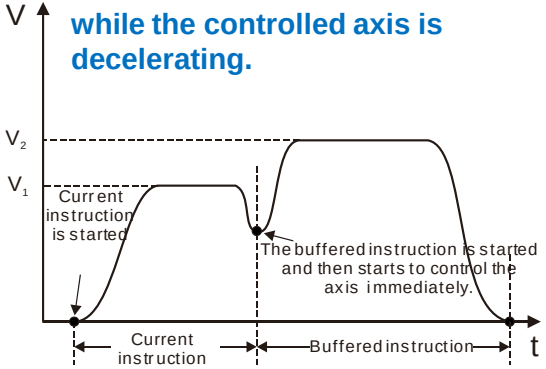
Case 1: Current instruction is aborted while the controlled axis is accelerating.



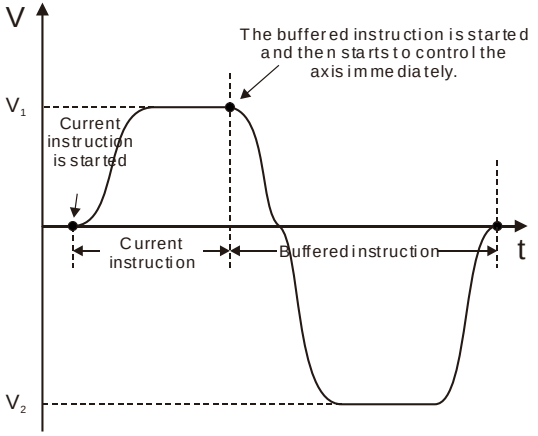
Case 2: Current instruction is aborted while the controlled axis is moving at a constant velocity.



Case 3: Current instruction is aborted while the controlled axis is decelerating.



Case 4: The velocity direction of the buffered instruction is opposite to the current instruction.

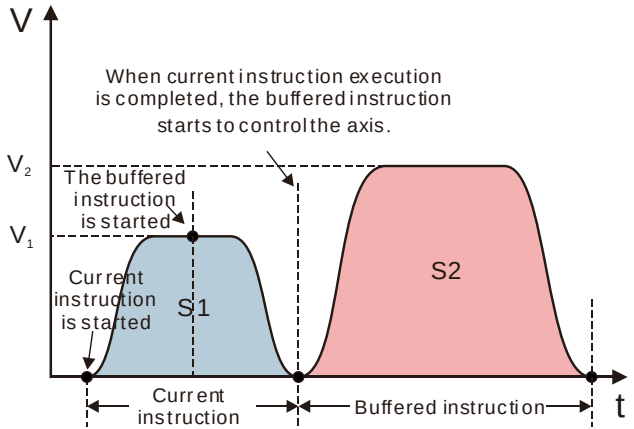


Explanation of BufferMode

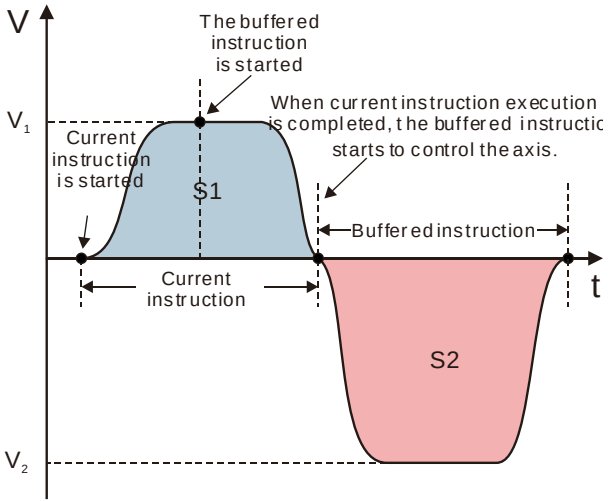
Buffer_Mode=mcBuffered

- When *BufferMode* of the buffered instruction is *mcBuffered*, the instruction will just start to control the axis after the current instruction reaches the target position.

Case 1: When the direction of the buffered instruction is the same as that of the current instruction.



Case 2: When the direction of the buffered instruction is opposite to that of the current instruction.

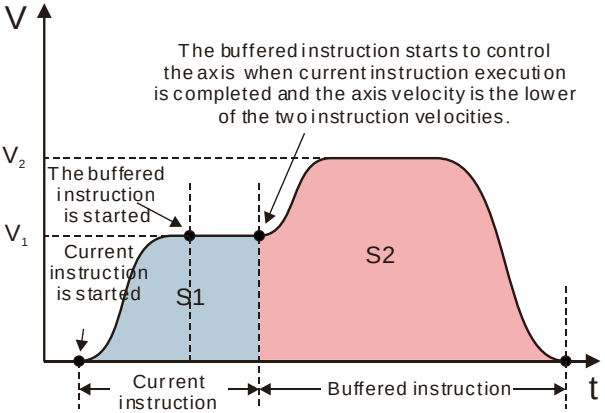


Explanation of BufferMode

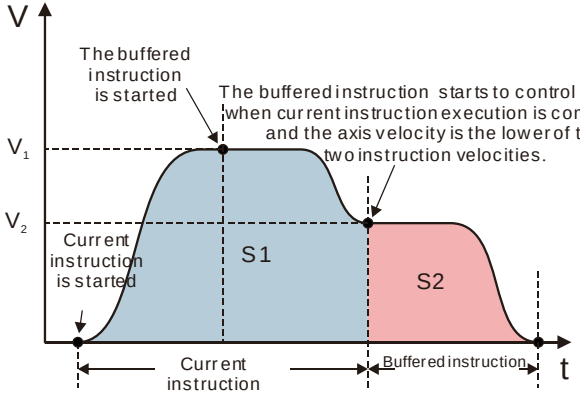
Buffer_Mode=mcBlendingLow

- When *BufferMode* of the buffered instruction is *mcBufferedLow*, the instruction will just start to control the axis after the current instruction reaches the target position and **the lower of the target velocities of the current instruction and buffered instruction is taken as the transit velocity.**

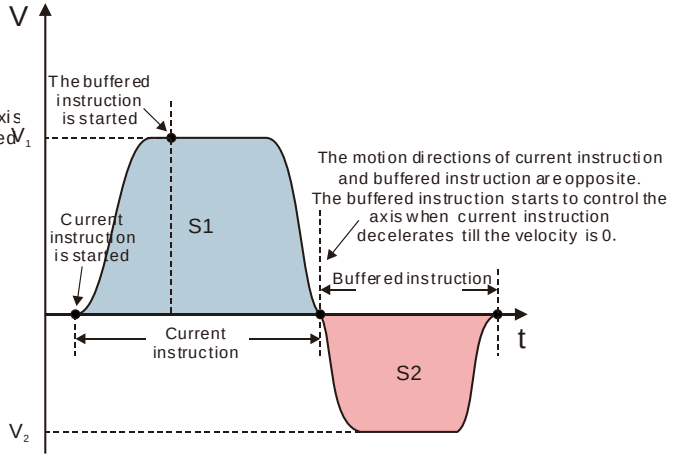
Case 1: When the velocity of the current instruction is less than that of the buffered instruction.



Case 2: When the velocity of the current instruction is greater than that of the buffered instruction.



Case 3: When the velocity direction of the current instruction is opposite to that of the buffered instruction.

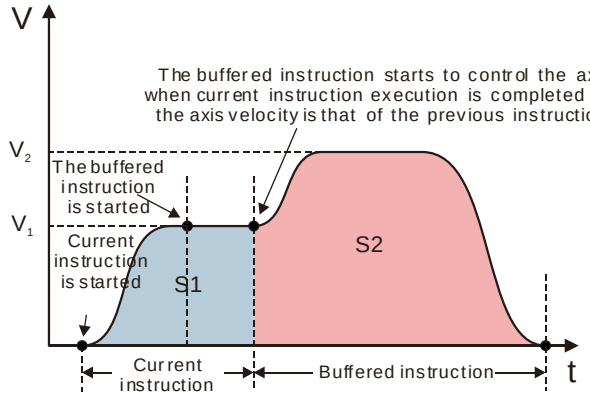


Explanation of BufferMode

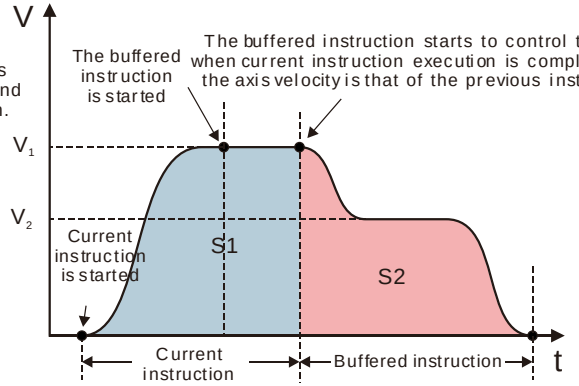
Buffer_Mode=mcBlendingPrevious

- When *BufferMode* of the buffered instruction is *mcBufferedPrevious*, the instruction will just start to control the axis after the current instruction reaches the target position and **the target velocity of the current instruction is taken as the transit velocity.**

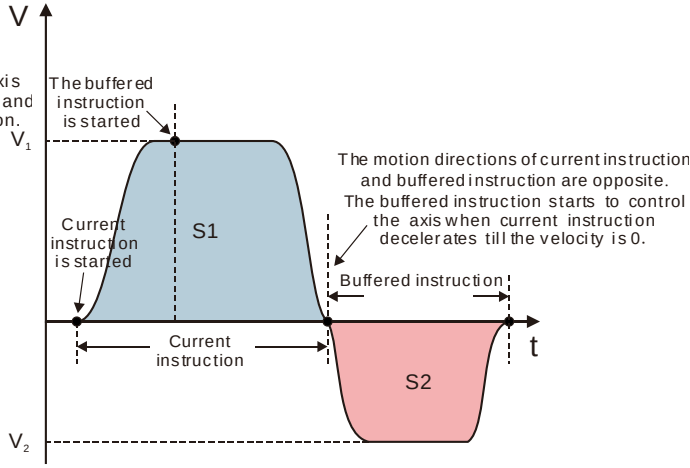
Case 1 : When the target velocity of the current instruction is less than that of the buffered instruction.



Case 2 : When the target velocity of the current instruction is greater than that of the buffered instruction.



Case 2 : When the velocity direction of the current instruction is opposite to that of the buffered instruction.

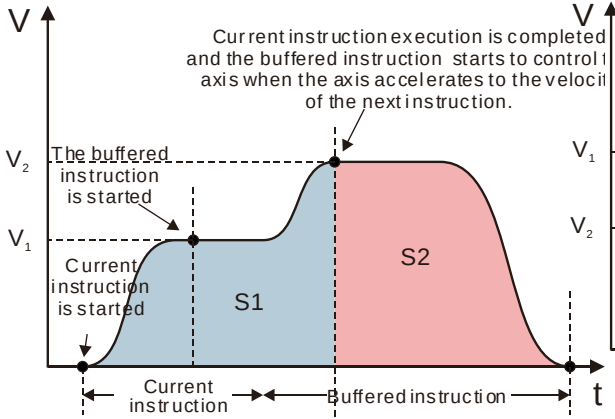


Explanation of BufferMode

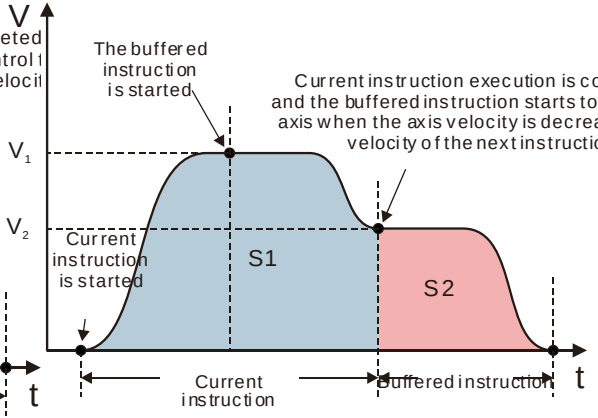
Buffer_Mode=mcBlendingNext

- When *BufferMode* of the buffered instruction is *mcBufferedNext*, the instruction will just start to control the axis after the current instruction reaches the target position and **the target velocity of the buffered instruction is taken as the transit velocity.**

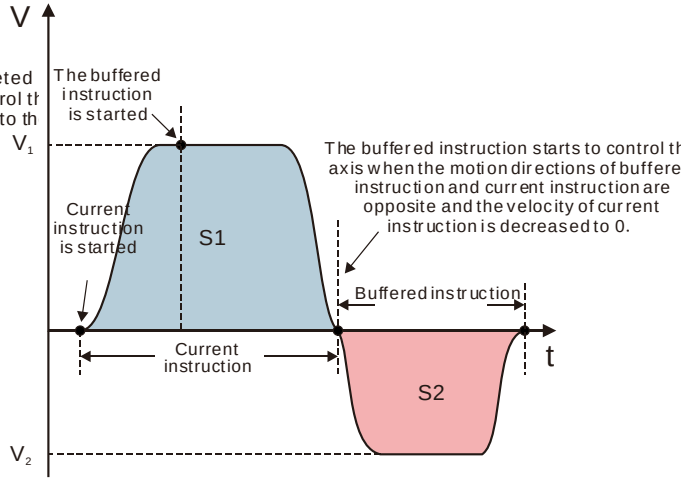
Case 1 : When the target velocity of the current instruction is less than that of the buffered instruction.



Case 2 : When the target velocity of the current instruction is greater than that of the buffered instruction.



Case 3 : When the velocity direction of the current instruction is opposite to that of the buffered instruction

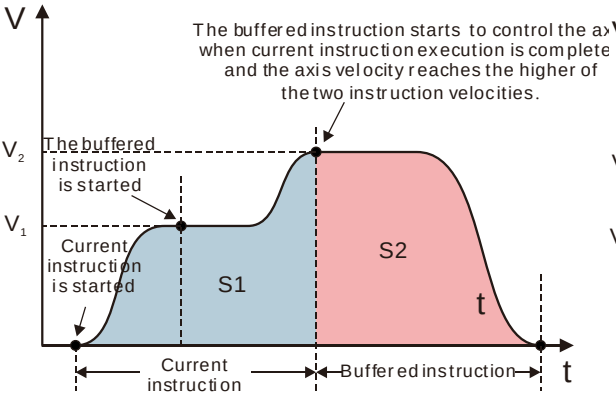


Explanation of BufferMode

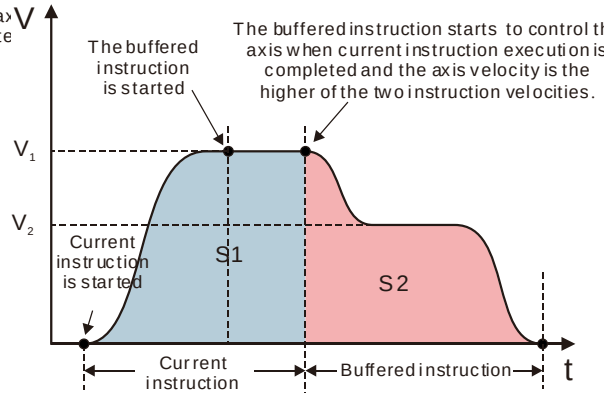
Buffer_Mode=mcBlendingHigh

- When *BufferMode* of the buffered instruction is *mcBufferedHigh*, the instruction will just start to control the axis after the current instruction reaches the target position and **the higher of the target velocities of the current instruction and buffered instruction is taken as the transit velocity.**

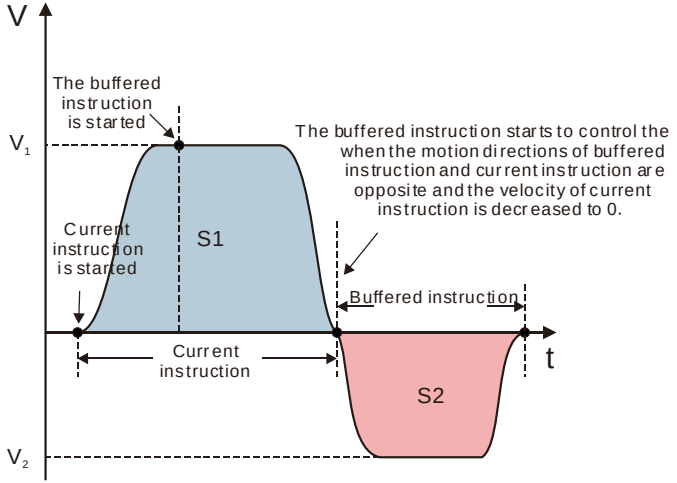
Case 1: When the target velocity of the current instruction is less than that of the buffered instruction



Case 2: When the target velocity of the current instruction is greater than that of the buffered instruction



Case 3: When the velocity direction of the current instruction is opposite to that of the buffered instruction

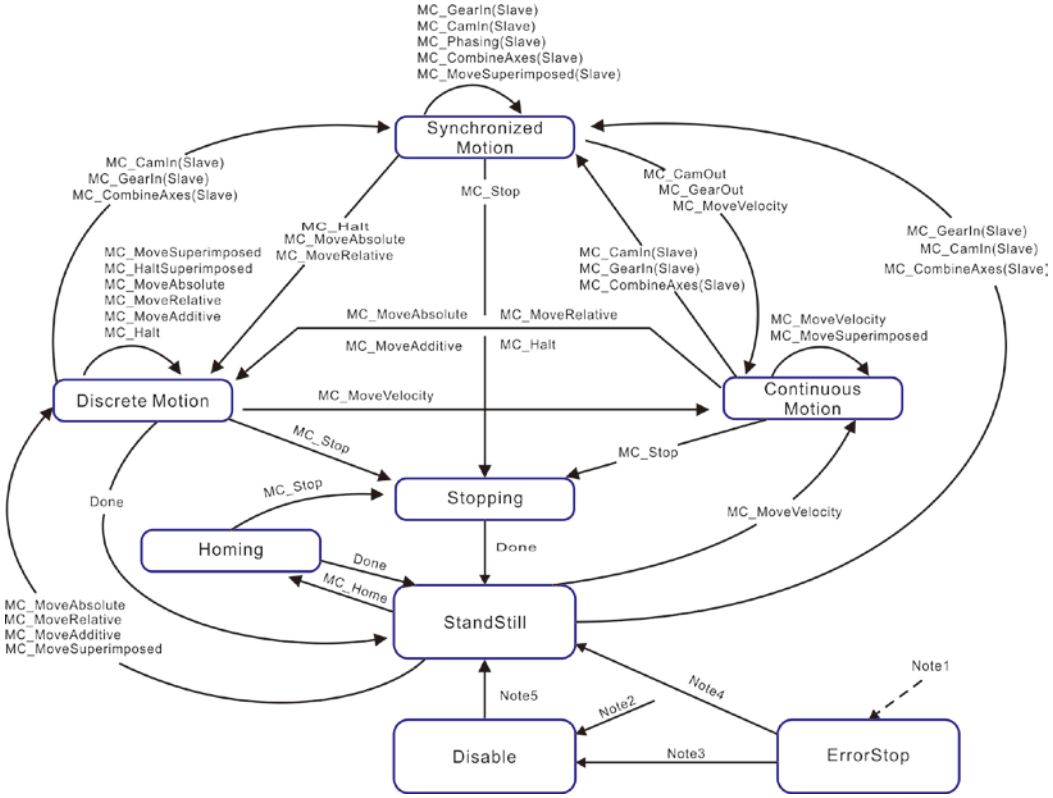


Motion Control Principle

State Machine

- The state machine shows the current state of an axis and manages the execution of motion instructions.
- Every axis has its own state machine.
- Motion instructions can be executed only in the allowed axis state. Otherwise, an error will occur.
- The axis state may change when an instruction is executed.
- The MC_ReadStatus instruction can be used to get the state of an axis.

Axis state	Indication
Disable	No-execution state
StandStill	Pre-execution state
ErrorStop	Error state
Stopping	Stop state
Homing	Homing state
Discrete Motion	Discrete motion state
Continuous Motion	Continuous motion state
Synchronized Motion	Synchronized motion state



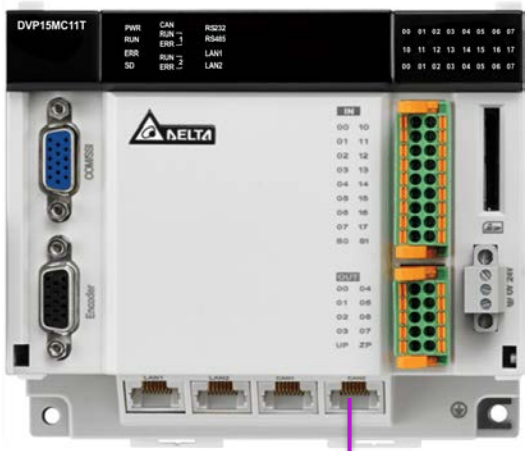
DVP15MC11T

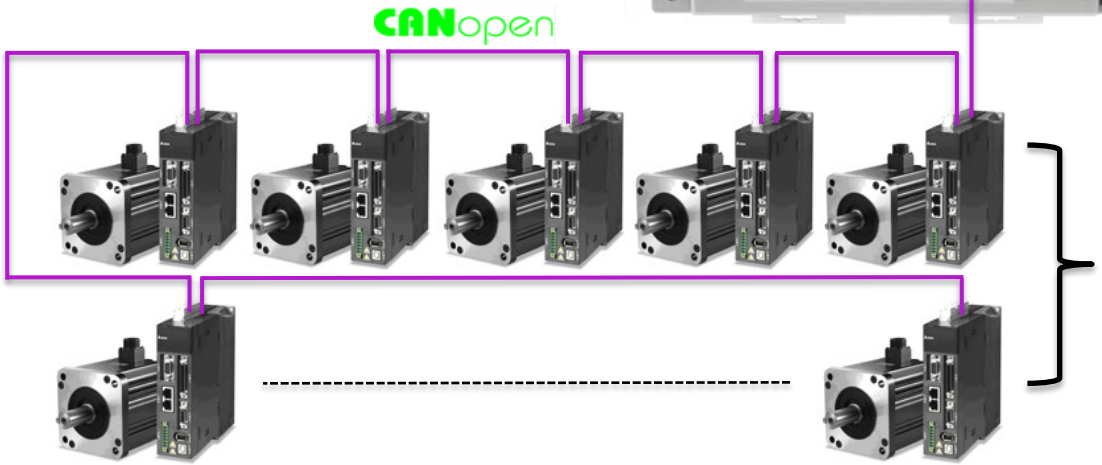
Motion Network Configuration

Motion Network Construction

Motion Network Construction

- Delta servo drive: Only ASD-A2-****-M series are connectable.
- Maximum 24 servo drives (24 real axes) are allowed to connect to 15MC. The range of CANopen station address for servo drives: 1~24.
- **Please do use Delta standard CAN cable!**
- Connect the resistors of 120Ω between CAN_H and CAN_L of the two ends of the network respectively. Model of Delta CAN terminal resistor : TAP-TR01. CAN2 port has a built-in terminal resistor and thus only one terminal resistor is needed to connect at the other end of the network.





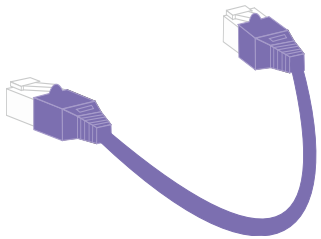
Transmission speed & Distance

Transmission speed (bit/second)	500K	1M
Max. communication distance (meter)	100	25

Motion Network Construction

Delta's CAN Network Cables

- **Please do use Delta standard CAN cable!**



Model	Length	Diameter（AWG）
UC-DN01Z-01A	305M	2#15，2#18 SHLD PVC（Thick cable）
UC-DN01Z-02A	305M	2#22，2#24 SHLD PVC（Thin cable）
UC-CMC003-01A	0.3M	4#26，1#24 PVC（Thin cable）
UC-CMC005-01A	0.5M	4#26，1#24 PVC（Thin cable）
UC-CMC010-01A	1.0M	4#26，1#24 PVC（Thin cable）
UC-CMC015-01A	1.5M	4#26，1#24 PVC（Thin cable）
UC-CMC020-01A	2.0M	4#26，1#24 PVC（Thin cable）
UC-CMC030-01A	3.0M	4#26，1#24 PVC（Thin cable）
UC-CMC050-01A	5.0M	4#26，1#24 PVC（Thin cable）
UC-CMC100-01A	10.0M	4#26，1#24 PVC（Thin cable）
UC-CMC200-01A	20.0M	4#26，1#24 PVC（Thin cable）

Motion Network Configuration

Setting Servo Parameters

Function	Parameter	Setting value	Description	
CANopen node address	P3-00	1~32	The node addresses of devices in the network are **different**.	The parameters must be set.
CANopen transmission speed	P3-01	403（1M） or 203（500K）	The transmission speeds of all devices in the network must be the **same**.	
Control mode	P1-01	B（CANopen mode）		
Emergency stop	P2-10~P2-17	Set to ON or OFF according to need.		The parameters are set or not according to actual demand.
Negative limit				
Positive limit				

DVP50MC11T

Motion Network Configuration

Motion Network Construction

Motion Network Construction

- A maximum of 24 ASD-A2-****-E servo drives are connectable（24 real axes）;
- There is a strict topology requirement to the EtherCAT network. It must strictly follow the rule that the input port of a servo should be connected to the output port of the next servo.
- When multiple servos are connected in series, the max. distance between servo drives is 50m;
- **Please do use Delta standard EtherCAT cable.**

Motion Network Construction

Delta EtherCAT Cable Model

- **Please do use Delta standard EtherCAT cable!**

EtherCAT cable model	Length	Diameter（AWG）
UC-EMC003-02A	0.3M	4#22 PVC
UC-EMC005-02A	0.5M	4#22 PVC
UC-EMC010-02A	1.0M	4#22 PVC
UC-EMC020-02A	2.0M	4#22 PVC
UC-EMC050-02A	5.0M	4#22 PVC
UC-EMC100-02A	10.0M	4#22 PVC
UC-EMC200-02A	20.0M	4#22 PVC

DVP15MC11T
DVP50MC11T

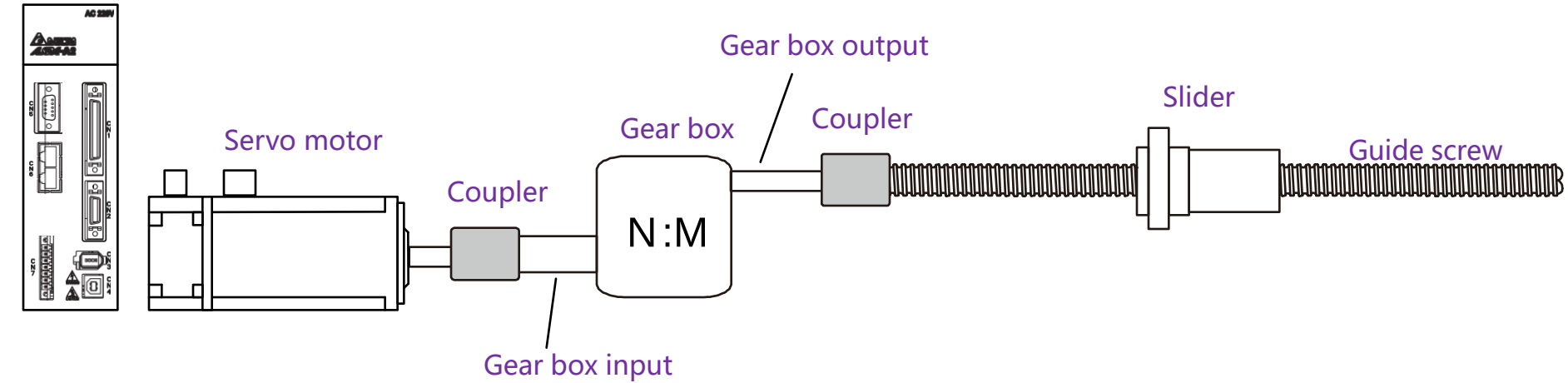
Motion Network Configuration

Motion Network Configuration

Definition of An Axis

- The servo drive, servo motor and actuator driven by the servo motor are collectively called an axis.

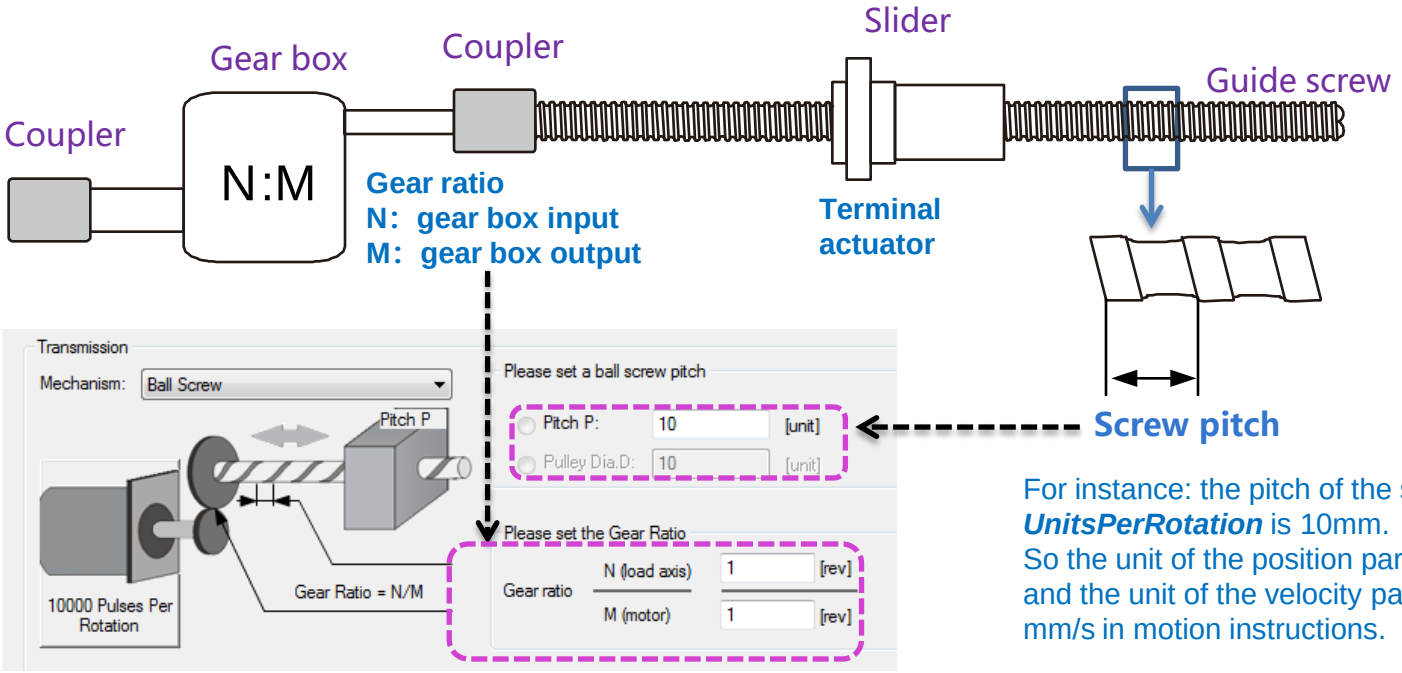
An Illustration of Guide-Screw Mechanism Axis



Motion Network Configuration

Axis Parameter

- **InputRotation & OutputRotation:** The parameter is set in accordance with the gear ratio of the gear box (gear reducer) in the actual mechanism.
- **UnitsPerRotation:** The number of units which the terminal actuator moves while output shaft of the gear box rotates for a circle. If it is a guide-screw mechanism, the parameter means the pitch of the guide screw.



For instance: the pitch of the screw is 10mm, ***UnitsPerRotation*** is 10mm.
So the unit of the position parameter is mm and the unit of the velocity parameter is mm/s in motion instructions.

DVP15MC11T
DVP50MC11T
Motion Control Instructions

Overview of Motion Instructions

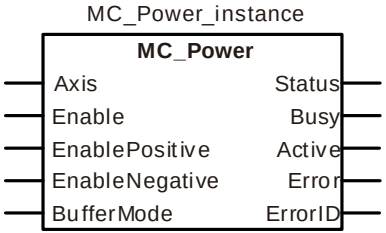
Instruction set		Instruction	Function
Single-axis instruction	Administration	MC_Power	Power Servo
		MC_Home	Homing
		MC_SetPosition	Set an axis position
		MC_SetOverride	Set override factors
		MC_Reset	Reset
		MC_ReadAxisError	Read axis errors
		MC_ReadActualPosition	Read actual axis position
		MC_ReadStatus	Read axis status
		MC_ReadMotionState	Read axis' motion status
		DMC_ReadParameter_CAN2	Read a servo parameter
		DMC_WriteParameter_CAN2	Write a servo parameter
	Velocity control	MC_MoveVelocity	Velocity
		MC_Halt	Halt
		MC_Stop	Stop

Instruction set		Instruction	Function
Single-axis instruction	Position control	MC_MoveRelative	Relative positioning
		MC_MoveAdditive	Additive positioning
		MC_MoveAbsolute	Absolute positioning
		MC_MoveSuperimposed	Superimposed positioning
		MC_HaltSuperimposed	Halt superimposing
	Torque control	DMC_SetTorque	Torque control
	Homing	MC_Home	Homing
Multi-axis instruction	E-cam	MC_CamIn	Start cam operation
		MC_CamOut	End cam operation
	E-gear	MC_GearIn	Start gear operation
		MC_GearOut	End gear operation
		MC_CombineAxes	Two-master-axis-combined gear
Application instruction	Rotary cut	APF_RotaryCut_Init	Initialize rotary cut
		APF_RotaryCut_In	Rotary cut in
		APF_RotaryCut_Out	Rotary cut out

Enable or Disable

The MC_Power instruction is used to enable or disable an axis.

- When *Enable* is TRUE, the controlled axis is powered on. When *Enable* is FALSE, the controlled axis is powered off.
- When *Status* is TRUE, it means the axis is enabled. When *Status* is FALSE, it means the axis is disabled.
- Apart from administration instructions, other motion control instructions must be executed while power on.



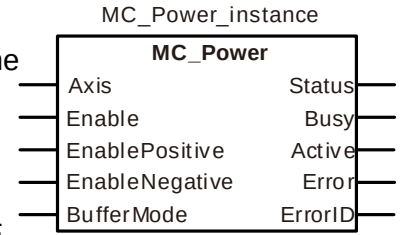
Forward/Reverse Rotation Setting

- When ***EnablePositive* is TRUE, the axis is allowed to run forward.** When ***EnablePositive*** is set to **FALSE and the axis is prohibited to run forward**, the instruction execution will be aborted and an error will occur if the instruction has controlled axis to run forward.
- When ***EnableNegative* is TRUE, the axis is allowed to run reversely.** When ***EnableNegative*** is set to **FALSE** and **the axis is prohibited to run reversely**, the instruction execution will be aborted and an error will occur if the instruction has controlled the axis to run reversely.
- If MC_Home or MC_SetTorque is executed, executing MC_Power has no effect on the forward and reverse motion of the axis.

Power Servo Axis

Precautions

- Please keep in mind the rule: one MC_Power instruction is for one axis during programming. If several MC_Power instructions are executed on the same axis, only the last executed MC_Power is effective.
- Positive or negative rotation must not be performed during the axis motion. The axis will decelerate at the deceleration rate of the instruction which is controlling the axis till the velocity is decreased to 0.
- When the forward motion is prohibited during the forward motion and the reverse motion is prohibited during the reverse motion, the axis should decelerate at the deceleration rate of the instruction which is controlling the axis motion till the velocity of the axis is decreased to 0.
- The motion mode of the axis is determined by *BufferMode* parameter when *Enable* changes from TRUE to FASLE during the axis motion.

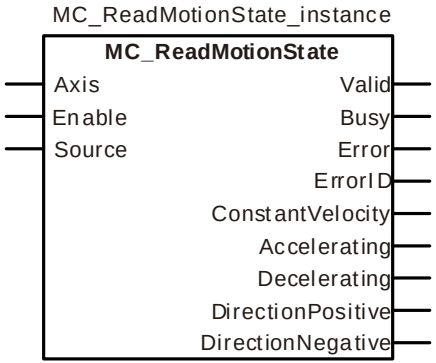


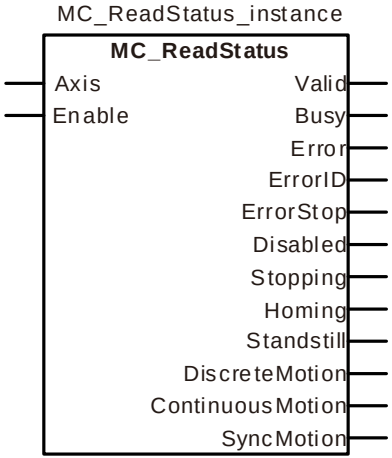
Buffermode	Function
0: Aborting	When *Enable* changes from TRUE to FALSE, the axis will stop moving immediately and enter the Disabled state. Please note that the operation may lead to the injury to persons or damage to the device!
1: Buffered	When *Enable* changes from TRUE to FALSE, **the axis will not enter the Disabled state immediately**. The state machine will go into the Standstill state only when the axis stops moving. One program cycle later, it will just go into the Disabled state.

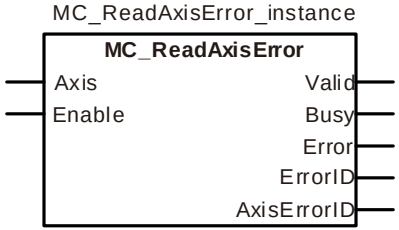
Read Axis Status

Introduction

- **MC_ReadMotionState** is used for obtaining current **motion status** (such as acceleration, deceleration, constant motion) and **motion direction** (forward/reverse).
- **MC_ReadStatus** is used for obtaining **axis status** (in the state machine).
- **MC_ReadAxisError** is used for obtaining detailed **error codes** (alarm codes that the servo drive reports) when the axis **is in ErrorStop**.



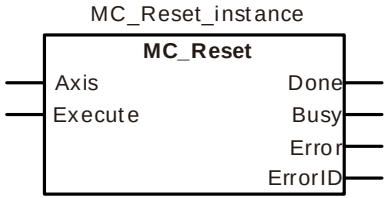




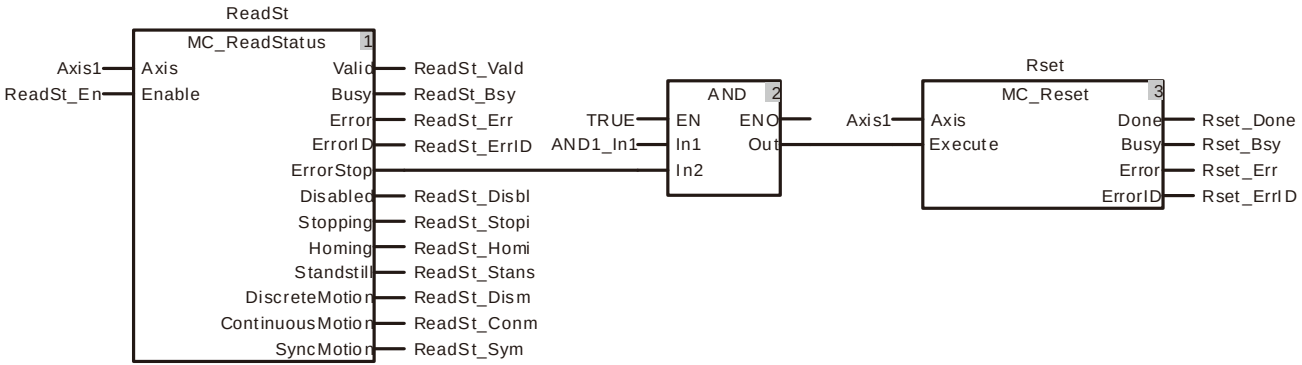
Axis Reset

Introduction

- MC_Reset resets the axis in ErrorStop state **including the axis state of the axis and the alarm state of the corresponding servo drive**.
- When *Done* is TRUE, it means the error is cleared successfully. Some alarm states of the servo drive may not be cleared via MC_Reset and should be cleared according to the servo drive manual.
- After MC_Reset is executed, MC_Power and the servo drive jointly determine whether the axis enters the StandStill or Disable state.



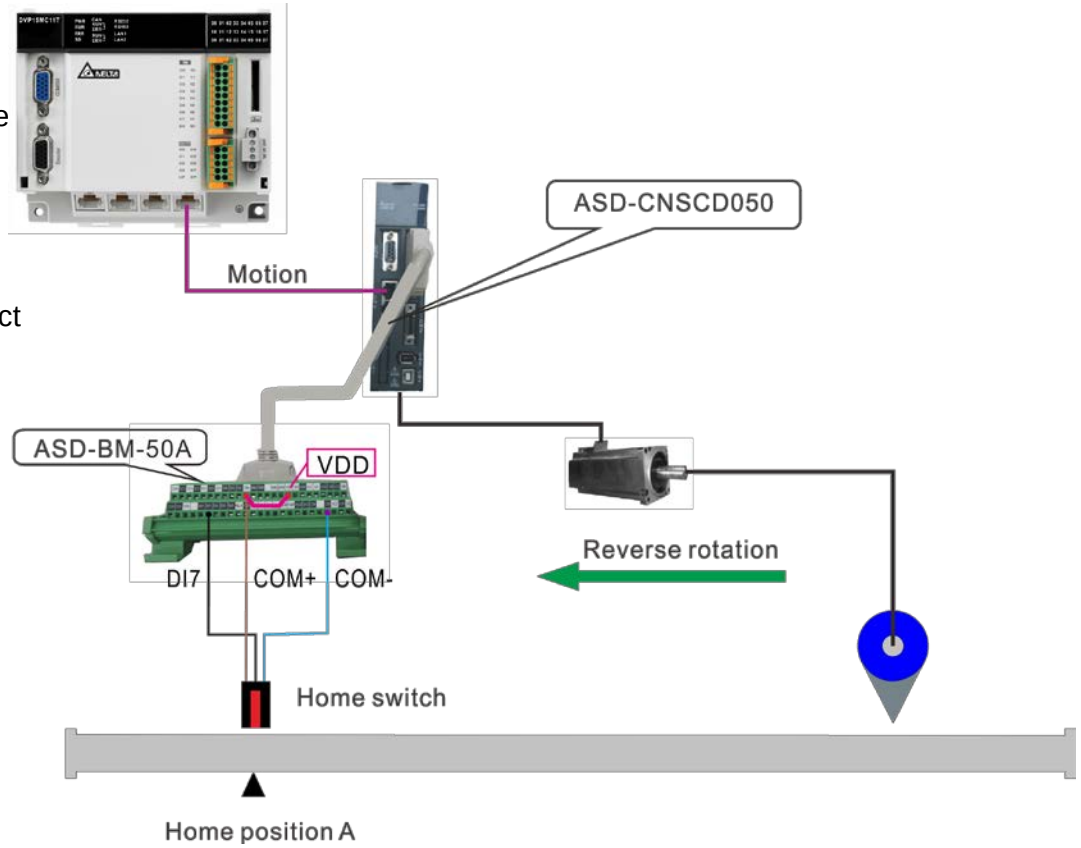
Example



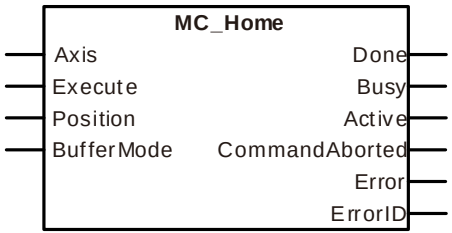
Homing

MC_Home

- MC_Home controls the axis to return to the home position.
- Supports 35 modes of homing, which can be set through the programming software for DVP15MC11T.
- For some modes, the servo drive needs to connect homing-related signals (Negative limit switch, positive limit switch and home switch).



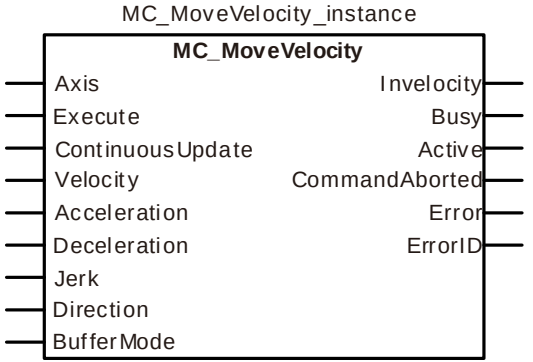
MC_Home_instance



Motion exercise 1

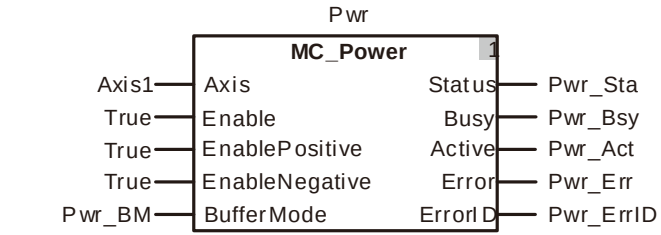
MC_MoveVelocity

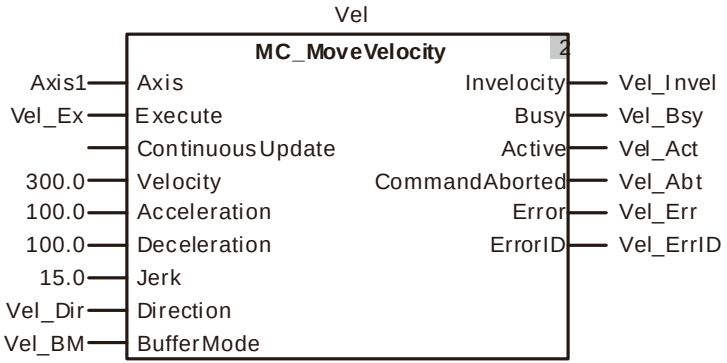
- MC_MoveVelocity controls the axis to constantly move at the target speed specified by *Velocity* in the direction specified by *Direction*.
- Due to the effect of MC_MoveVelocity, the axis motion is affected by the parameters *Acceleration*, *Deceleration and Jerk* during the change from current speed to target speed.
- When the axis reaches the set target speed, *Invelocity* changes to TRUE. *Invelocity* can be used to check if the axis reaches the target velocity.
- **The instruction is re-triggered for execution.** At any time while MC_MoveVelocity is being executed, the instruction can be re-triggered via *Execute* and re-executed according to current parameters. The parameters such as *Velocity, Acceleration, Deceleration* and *Jerk* can be valid again.
- MC_MoveVelocity is affected by MC_SetOverride. After the execution of MC_MoveVelocity is finished (that is, *InVelocity* changes from FALSE to TRUE), *InVelocity* will always keep TRUE even if the target velocity is changed via MC_SetOverride. When the execution of MC_MoveVelocity is not finished (that is *InVelocity* is FALSE), *InVelocity* will not changes from FALSE to TRUE until the new target velocity is reached after the target velocity is changed via MC_SetOverride.

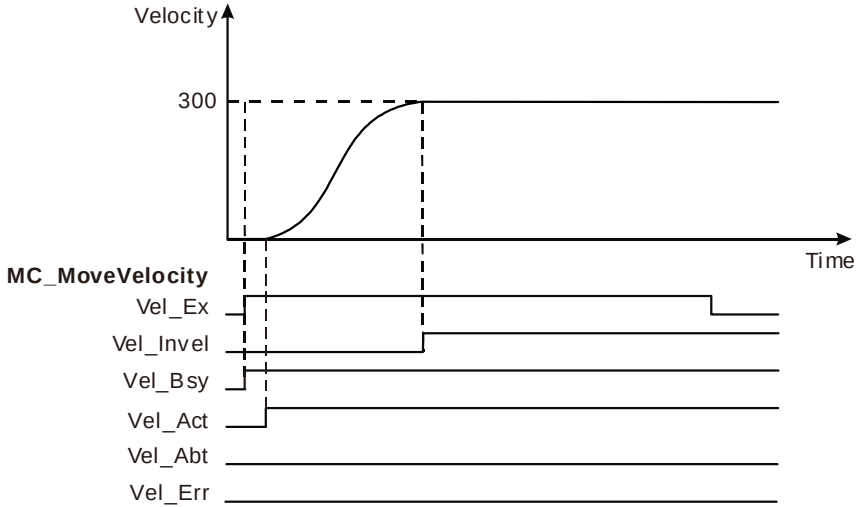


Velocity Control

Example





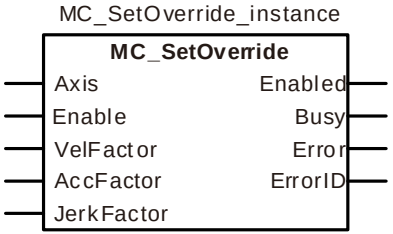


Motion exercise 2

Set Velocity Override Factors

MC_SetOverride

- MC_SetOverride is used for changing target velocities of instructions based on the set factor. The new target velocity after modification= Target velocity of currently executed instruction x Velocity override factor.
- The unit of *VelFactor* is %. “100” indicates “100%”. The valid range of *VelFactor* is between 0 and 500. An error will occur if the MC_SetOverride instruction is executed when *VelFactor* value exceeds the valid range.
- During execution of MC_SetOverride, the value of *VelFactor* can be changed dynamically and the new value will take effect immediately.
- The value of *VelFactor* is 100% when the execution of MC_SetOverride stops.



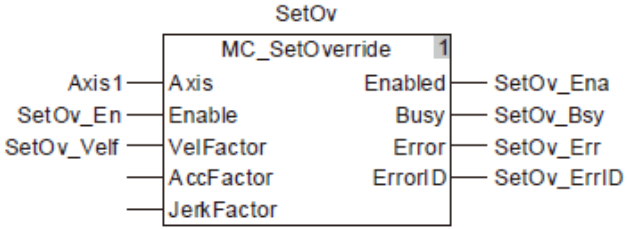
Instructions affected by MC_SetOverride

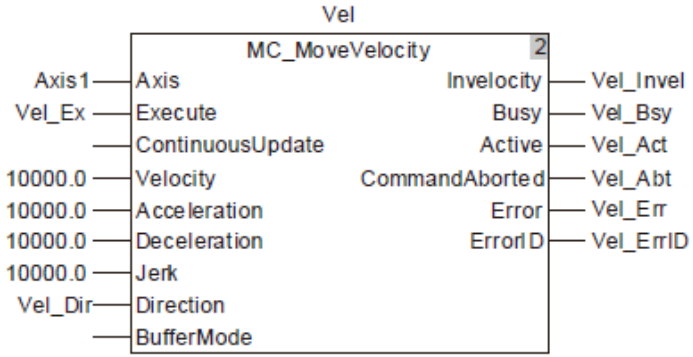
Instruction	Name
MC_MoveVelocity	Velocity instruction
MC_MoveRelative	Relative positioning
MC_MoveAbsolute	Absolute positioning
MC_MoveAdditive	Additive positioning
MC_MoveSuperimposed	Superimposed positioning

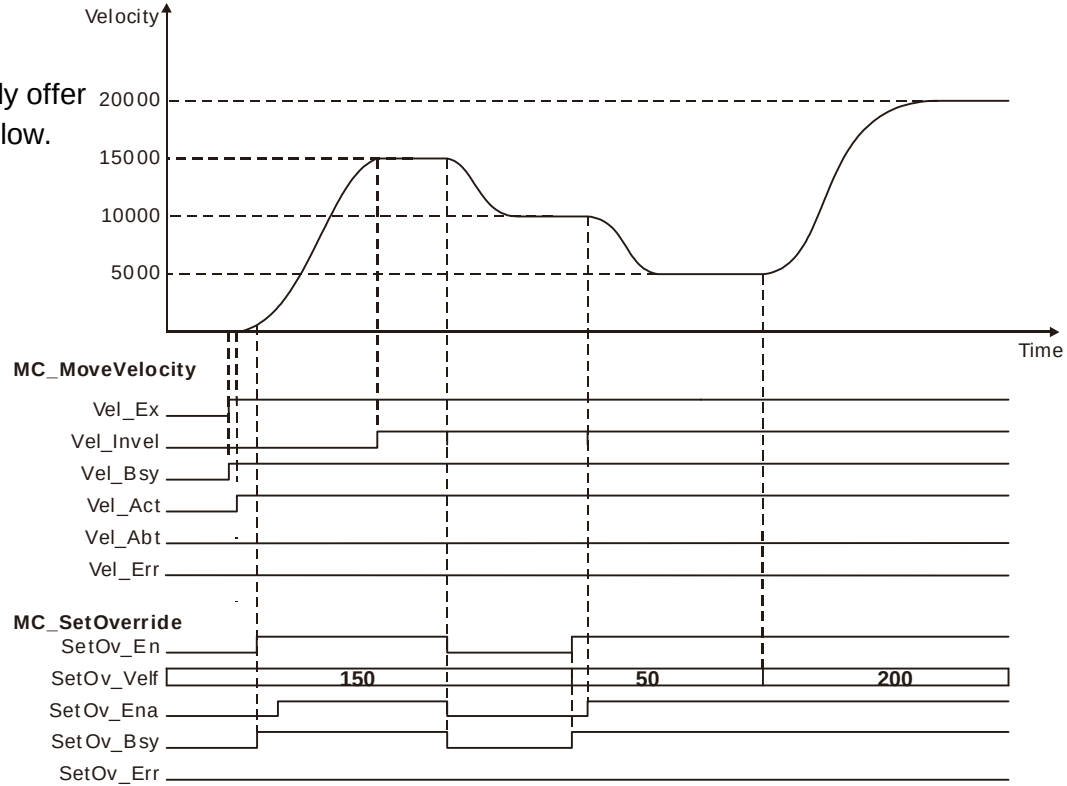
Set Velocity Override Factors

Example

- The values of *VelFactor*: 150, 50 and 200 respectively offer the impact on the MC_MoveVelocity instruction as below.



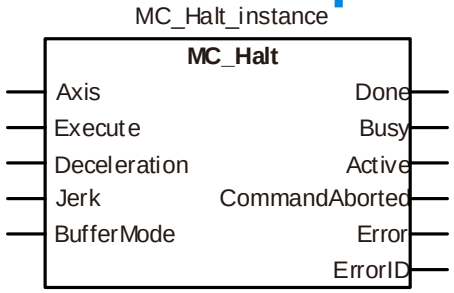




Motion exercise 3

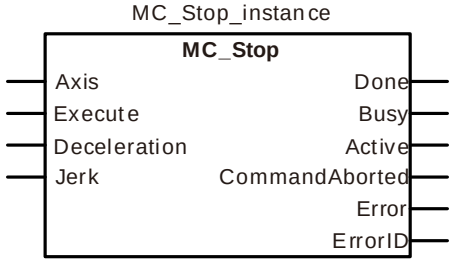
Halt & Stop

MC_Halt



- MC_Halt controls an axis to decelerate at the set deceleration rate till the axis halts.
- An axis is in DiscreteMotion state during the execution of MC_Halt. When the axis velocity is decreased to 0, *Done* changes to TRUE and meanwhile the axis enters Standstill state.

MC_Stop



- MC_Stop controls an axis to decelerate at the set deceleration rate till the axis stops.
- An axis is in Stopping state during the execution of MC_Stop. When the instruction execution is finished, the axis will still be in Stopping state if *Execute* is TRUE. The axis state will not change from Stopping to Standstill until *Execute* changes from TRUE to FALSE.

Difference Comparison

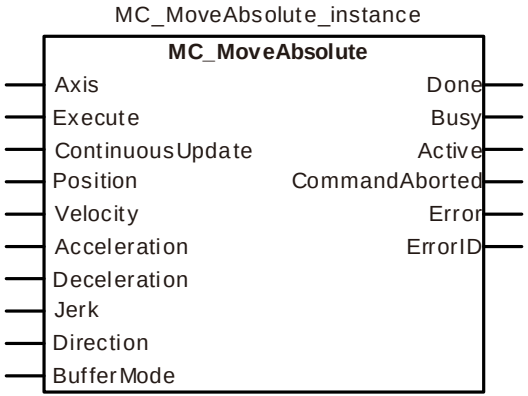
Motion exercise 4

Instruction	Function	During execution	After execution is finished
MC_Halt	Decelerate an axis to a halt	Axis status: DiscreteMotion The instruction can be aborted by other instruction.	While an axis is in Standstill state, other instruction can be executed on the axis.
MC_Stop	Decelerate an axis to a stop	Axis status: Stopping The instruction can be aborted only by MC_Stop.	*Execute* is TRUE: While an axis is in Stopping state, motion instructions except administration instructions can not be executed on it. *Execute* is FALSE: While an axis is in Standstill state, other motion instructions can be executed on it.

Absolute Positioning

MC_MoveAbsolute

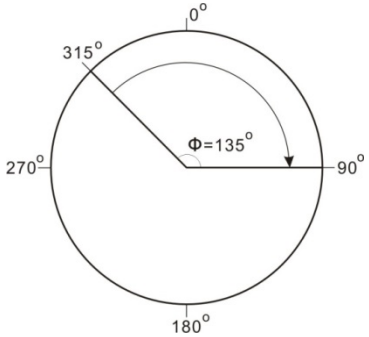
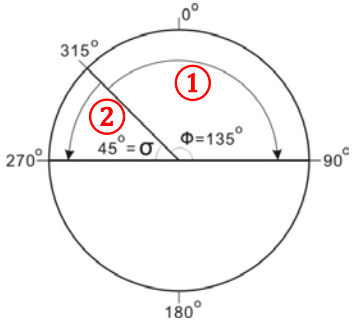
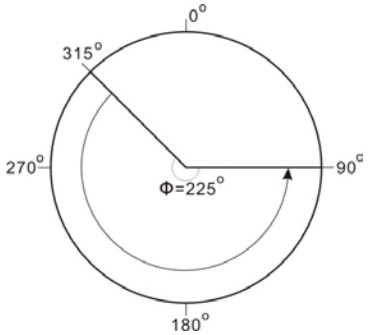
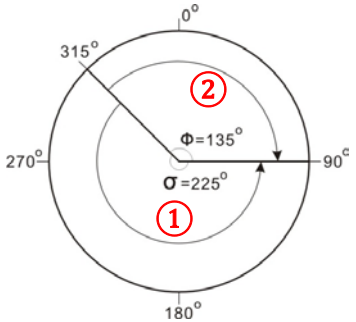
- MC_MoveAbsolute controls an axis to move to the target position which is **relative to zero point** at the set velocity, acceleration, deceleration and jerk.
- The value of *Position* should be a nonnegative number less than modulo if the axis is in the mode of rotary axis. Otherwise, an error will occur.
- *Direction,* used for selection of the motion direction, is just valid when the axis is a rotary axis.



Absolute Positioning

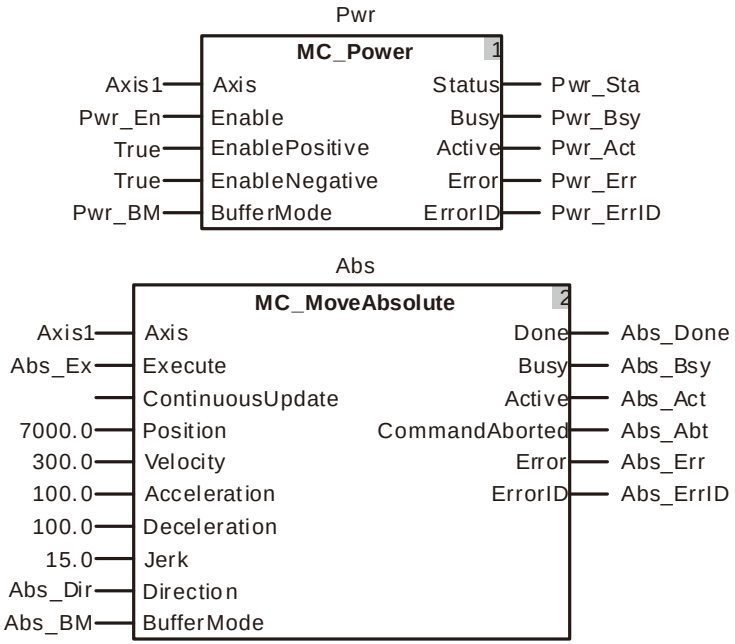
Direction

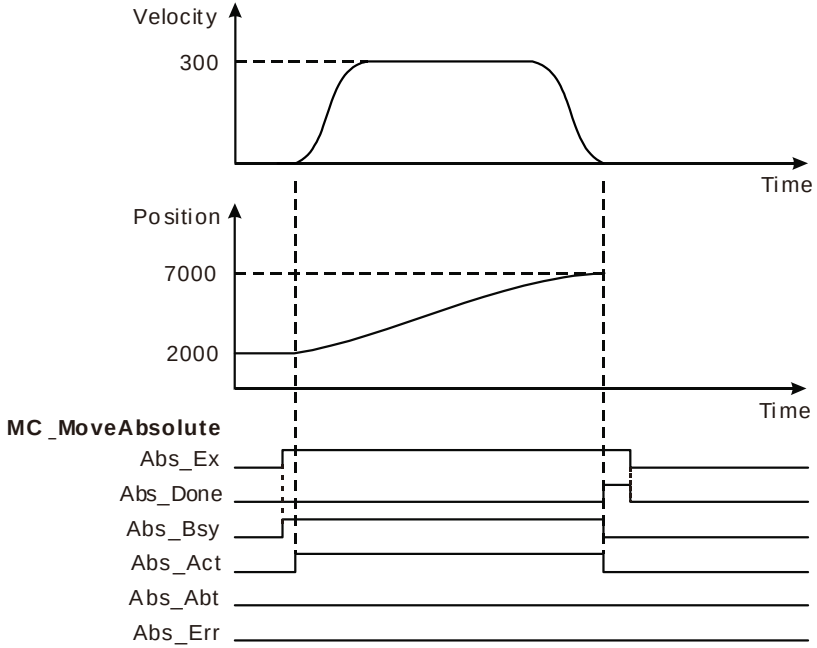
- *Direction*, used for selection of the motion direction, is just valid when the axis is a rotary axis.

Direction：1（Positive direction） **Current position: 315°** **Target position: 90°** **Movement angle: 135°**		**Direction：2（Shortest way）**	**Current position：315°** **Target position: 90°** **Movement angle: 135°（①）** **Current position: 315°** **Target position: 270°** **Movement angle: 45°（②）**	
Direction：3（Negative direction） **Current position：315°** **Target position: 90°** **Movement angle: 225°**		**Direction：4（Current direction）**	**Current direction: reverse rotation** **Current position: 315°** **Target position: 90°** **Movement angle: 225°（①）** **Current direction: Standstill or forward rotation** **Current position: 315°** **Target position: 90°** **Movement angle: 135°（②）**	

Absolute Positioning

Example



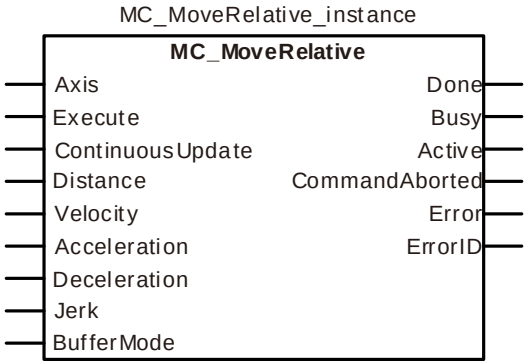


Motion exercise 5

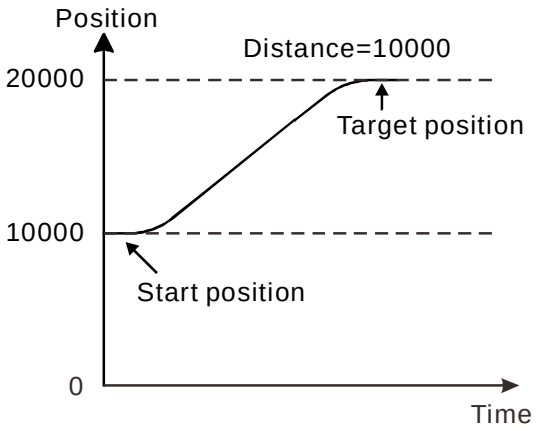
Relative Positioning

MC_MoveRelative

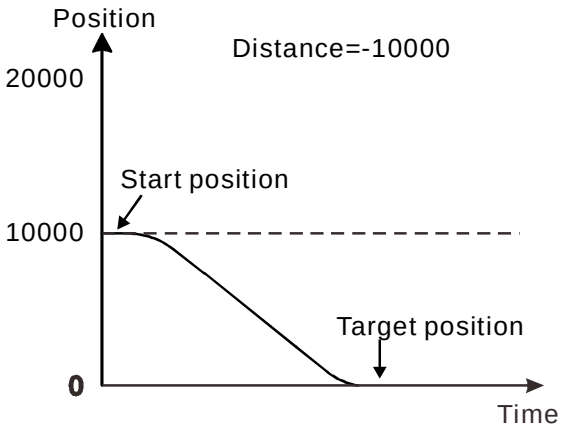
- MC_MoveRelative is used to make the axis move a given distance which takes the current axis position as the reference start point at a given speed, acceleration and deceleration and Jerk.
- The value of Distance can be a negative number or a positive number. **Target position = Start position + *Distance***



Distance>0



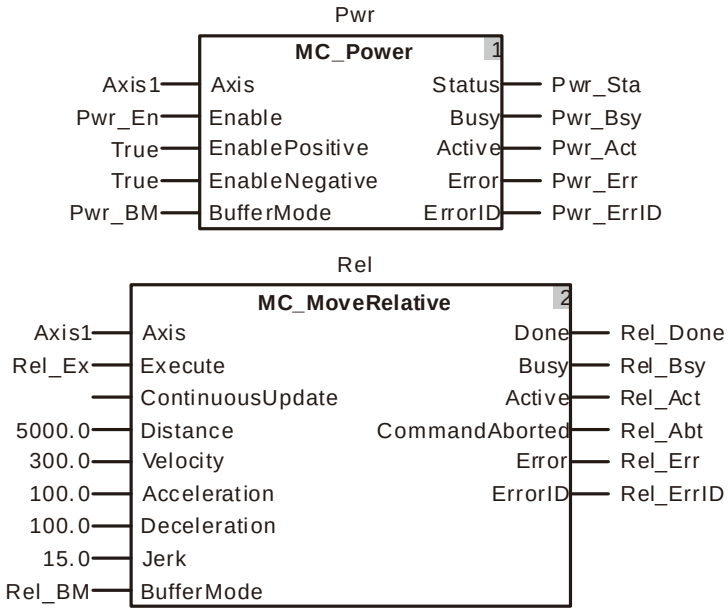
Distance<0

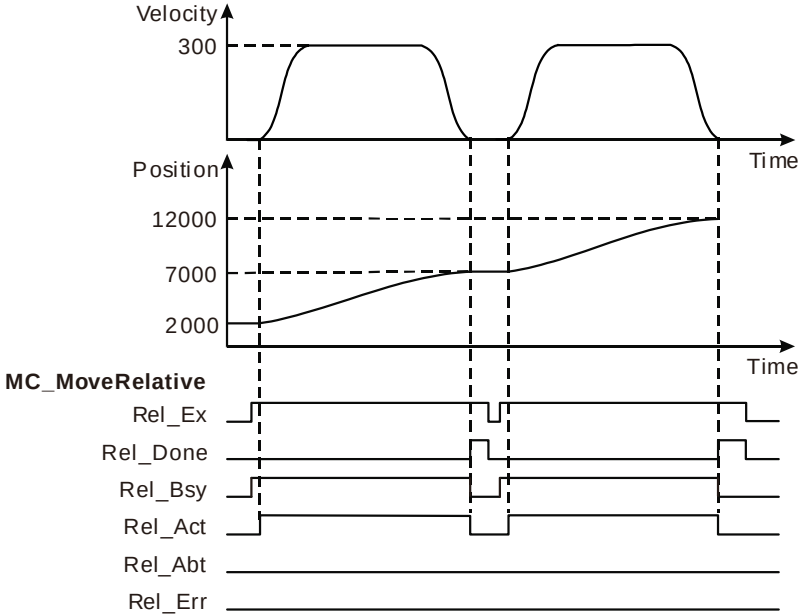


Relative Positioning

Example

- Repetitively trigger MC_MoveRelative to execute.



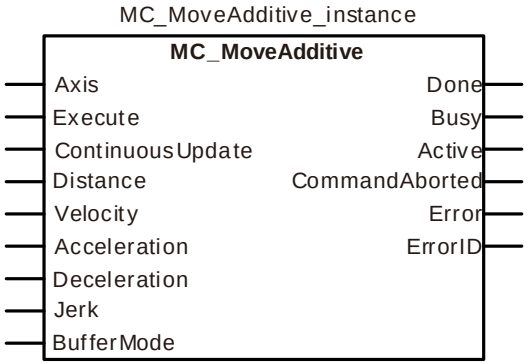


Motion exercise 6

Additive Positioning

MC_MoveAdditive

- MC_MoveAdditive is used to make the axis move an additive distance at a given speed, acceleration, deceleration and jerk.
- If **MC_MoveAdditive** is executed during the execution of a positioning-related instruction, the being executed instruction will be aborted and the uncompleted distance will be finished by **MC_MoveAdditive**. The distance that the axis which **MC_MoveAdditive** controls should move is the sum of the distance uncompleted by the aborted instruction and the additive distance specified by **MC_MoveAdditive.**
- The velocity instruction will be aborted if MC_MoveAdditive is executed during the execution of the velocity instruction.

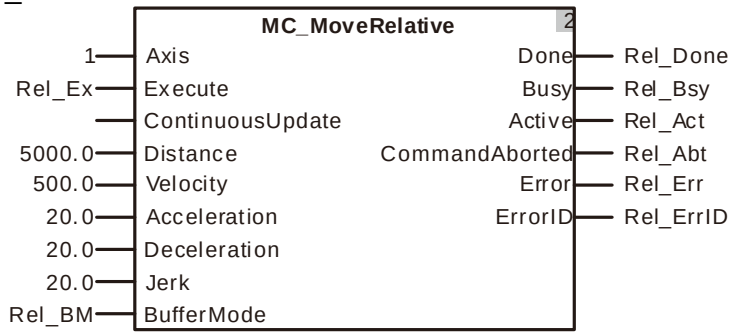


Additive Positioning

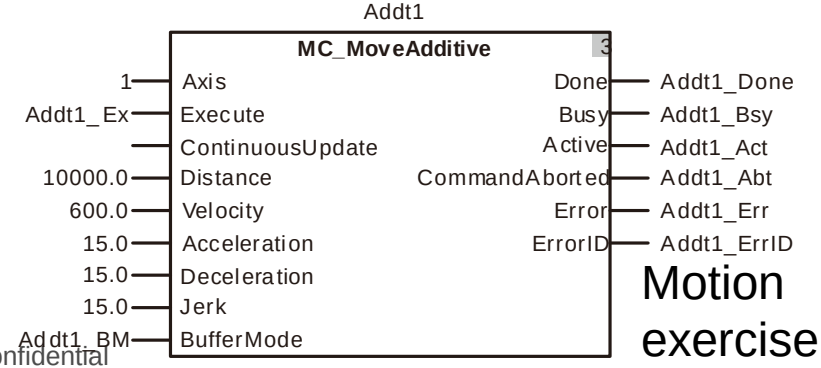
Example

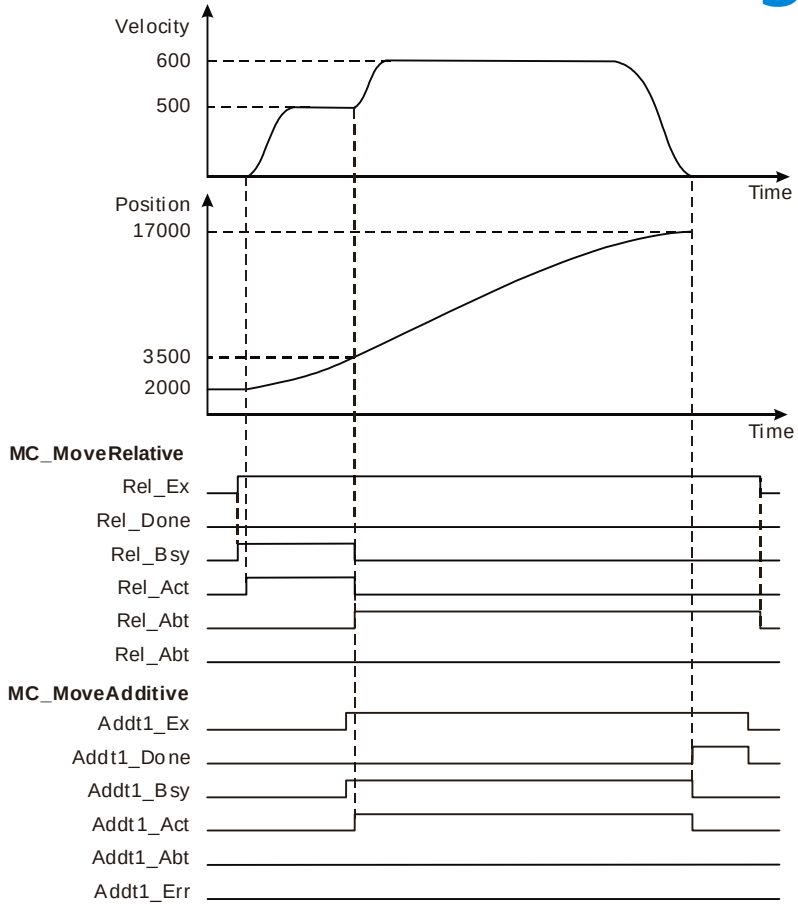
- Execute MC_MoveAdditive during the execution of MC_MoveRelative.

Rel



Addt1





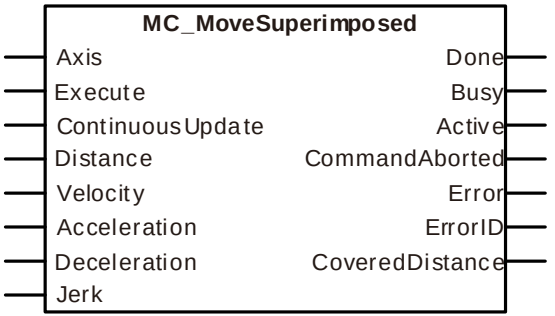
Motion exercise 7

Superimposed Positioning& Halt Superimposing

MC_MoveSuperimposed

- Based on the current motion state, MC_MoveSuperimposed controls an axis to **superimpose** the set distance according to the set velocity, acceleration and deceleration.
- Executing MC_MoveSuperimposed will not abort the being executed motion instruction excluding MC_MoveSuperimposed and MC_HaltSuperimposed. The two instructions will be in execution together and the **distances, velocities, accelerations and decelerations** will be added up respectively.

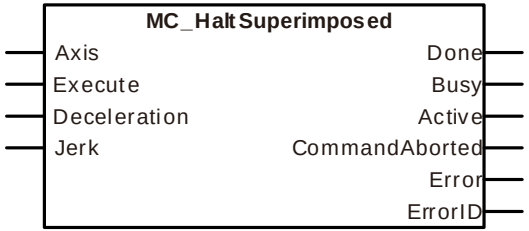
MC_MoveSuperimposed_instance



MC_HaltSuperimposed

- MC_HaltSuperimposed is only used for aborting the execution of MC_MoveSuperimposed.
- MC_HaltSuperimposed can not be executed when no MC_MoveSuperimposed is executed
- MC_HaltSuperimposed and MC_MoveSuperimposed can be aborted by each other.

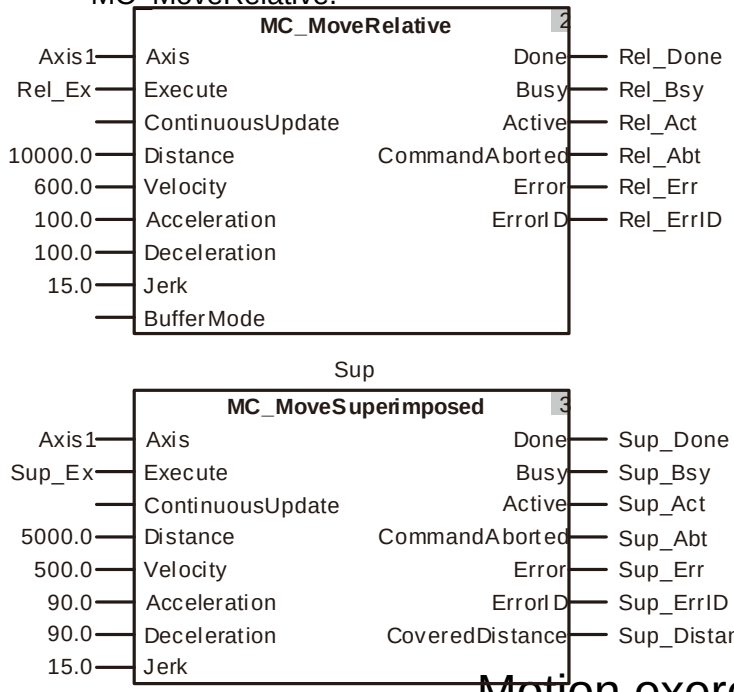
MC_HaltSuperimposed_instance

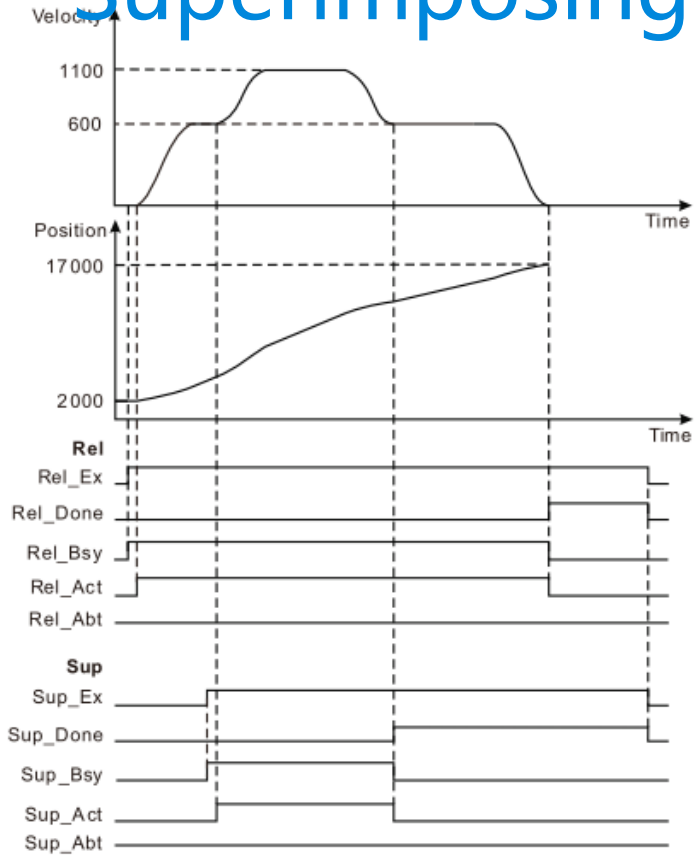


Superimposed Positioning& Halt Superimposing

Example

- Execute **MC_MoveSuperimposed** during the execution of MC_MoveRelative.



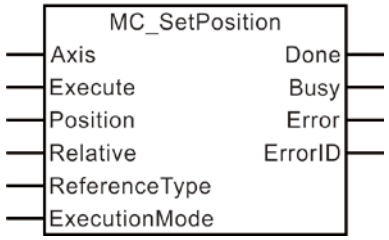


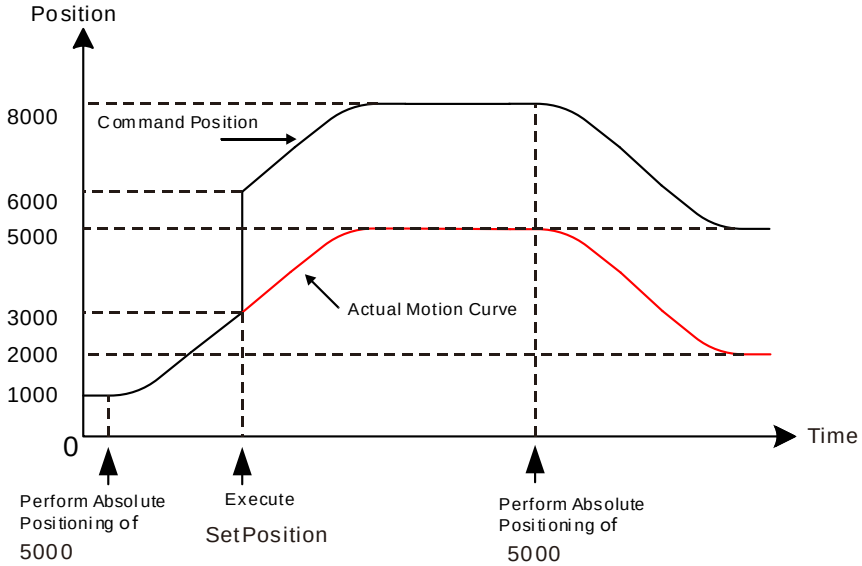
Position Setting & Reading

Introduction

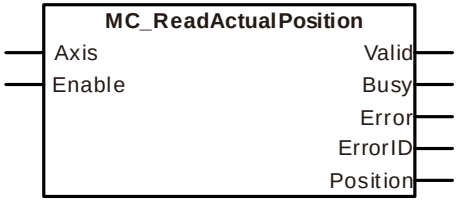
- **MC_SetPosition** sets the position of an axis to a given value and **no actual axis motion is brought.**
- The position of the axis after being set via MC_SetPosition can be read with **MC_ReadActualPosition**.
- Executing **MC_SetPosition** has no realistic impact on the motion which is carrying out. But it will affect the execution effect of other instruction which starts after the instruction execution is finished.

MC_SetPosition_instance





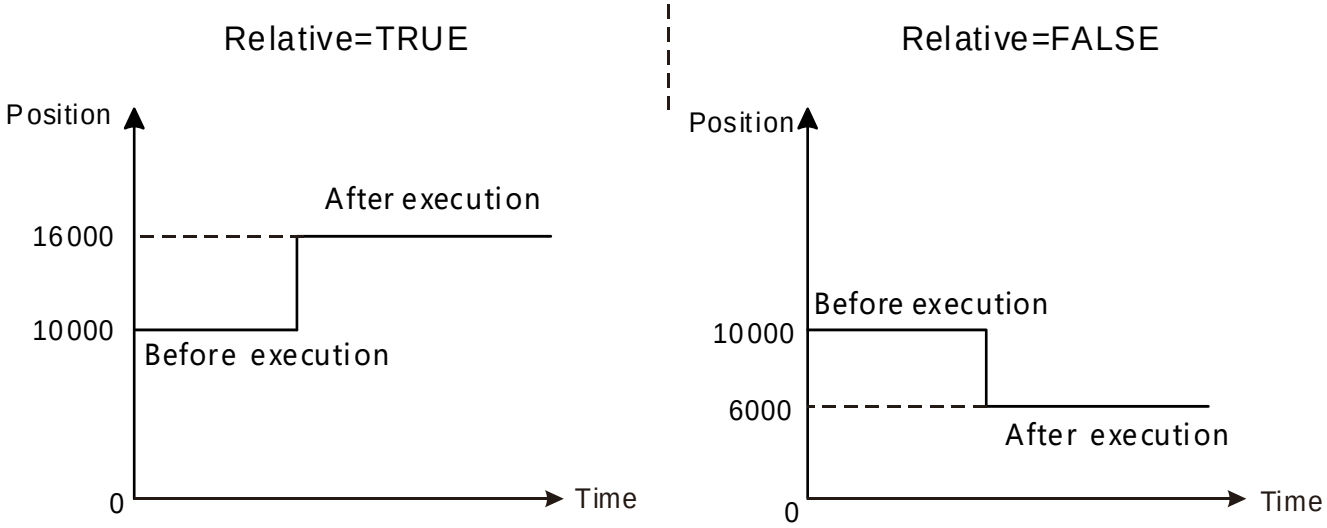
MC_ReadActualPosition_instance



Position Setting & Reading

Relative/Absolute Mode

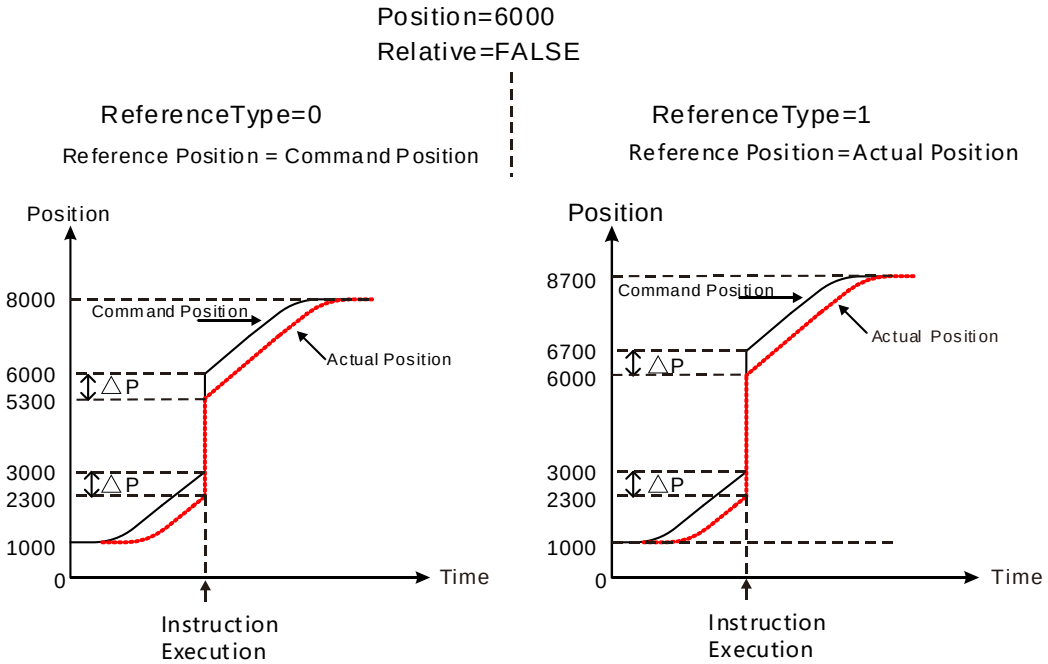
- When MC_SetPosition is executed, the relative/absolute relationship between the parameter *Position* and current position is determined by the parameter *Relative.* The execution effects are shown as below.



Position Setting & Reading

ReferenceType

- Parameter *ReferenceType* of MC_SetPosition is used to select the command position or actual position as the reference position and the execution effects are shown below.

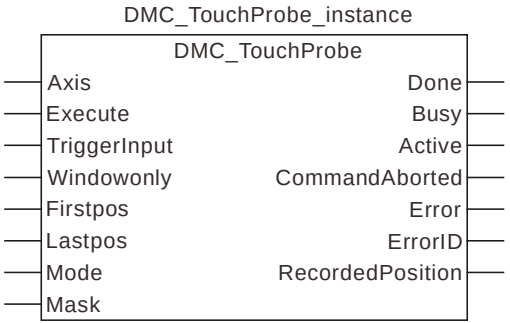


Motion exercise 9

Position Capture

DMC_TouchProbe

- DMC_TouchProbe is used for the position capture. The captured object includes **the encoder connected externally to the controller**, encoder of **the servo motor, pulses that CN1 of the servo drive receives and pulses that CN5 of the servo drive receives**.
- The capture value (*RecordedPosition)* is produced through converting the captured pulse number into the position value with Unit as the unit (of *RecordedPosition)* based on axis parameters, i.e., the unit of ***RecordedPosition*** is Unit.
- Capture mode 0~1 which DVP15MC11T and DVP50MC11T both support.
- Capture mode 2~4 which only DVP15MC11T supports and DVP50MC11T does not support.
- Capture mode 5~10 which only DVP50MC11T supports and DVP15MC11T does not support.



Position Capture

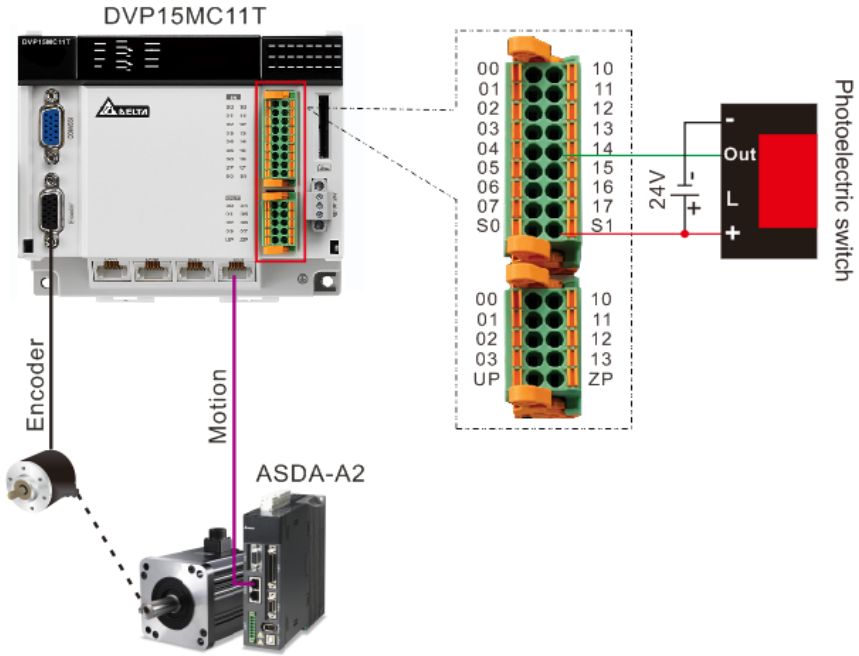
Explanation of Capture Modes

Mode	Data source	Trigger signal	Capture range (*RecordedPosition*)	Supportive controller
0	The encoder connected externally to a controller	Rising edge at the input points: I0~I7 and I10~I17 of a controller	After the number of pulses is out of the limits, the captured position can be accumulated according to the trend.	DVP15MC11T DVP50MC11T
1	The encoder connected externally to a controller	Falling edge at the input points: I0~I7 and I10~I17 of a controller		
2	The encoder of the servo motor	Rising edge at input point DI7 of CN1 port of A2-M servo drive		DVP15MC11T
3	CN1 port of the servo drive (pulses which pulse, /pule, sign, /sign or hpulse, /hpule, hsign, /hsign receives)	Rising edge at input point DI7 of CN1 port of A2-M servo drive	After the number of pulses is out of the limits, the captured position will not be accumulated.	
4	CN5 port of the servo drive (pulses which A , /A, B, /B receives)	Rising edge at input point DI7 of CN1 port of A2-M servo drive		
5	The encoder of the servo motor	Rising edge at input point DI13 of CN7 extension DI port of A2-E servo drive	After the number of pulses is out of the limits, the captured position will be accumulated based on the trend.	DVP50MC11T
6	The encoder of the servo motor	Falling edge at input point DI13 of CN7 extension DI port of A2-E servo drive		
7	The encoder of the servo motor	Rising edge and falling edge at input point DI13 of CN7 extension DI port of A2-E servo drive		
8	The encoder of the servo motor	Rising edge of Z phase signal from servo motor encoder		
9	The encoder of the servo motor	Falling edge of Z phase signal of servo motor encoder		
10	The encoder of the servo motor	Rising and falling edge of Z phase signal of servo motor encoder		

Position Capture

Example of Wiring for Signal Capture in Mode 0 and Mode 1

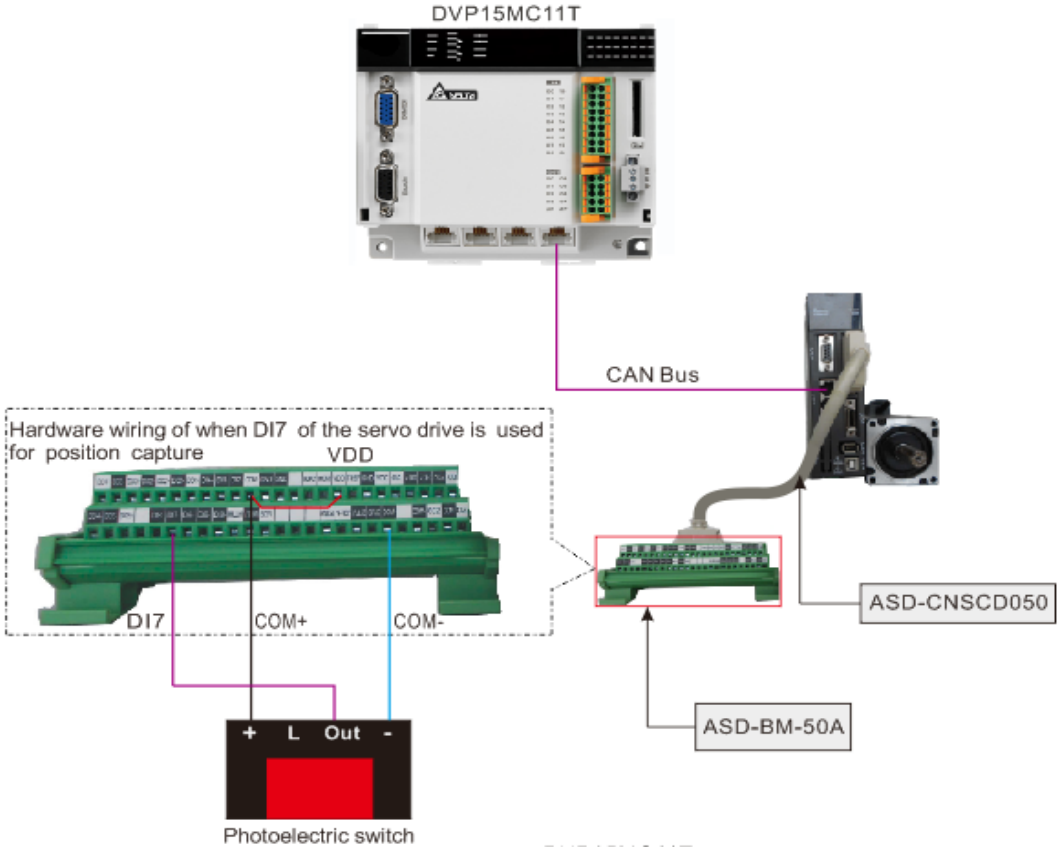
- Both of DVP15MC11T and DVP50MC11T support mode 0 and mode1.



Position Capture

Example of Wiring for Signal Capture in Mode 2

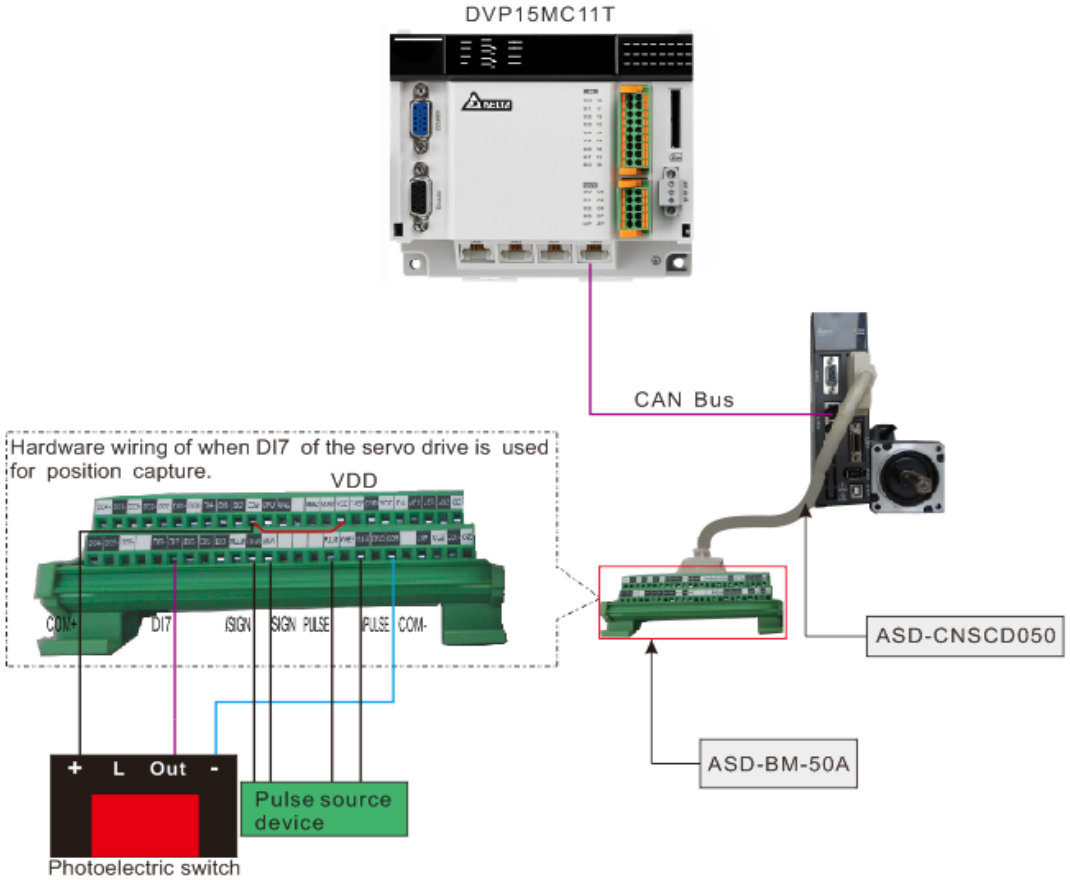
- Only DVP15MC11T supports mode 2.



Position Capture

Example of Wiring for Signal Capture in Mode 3

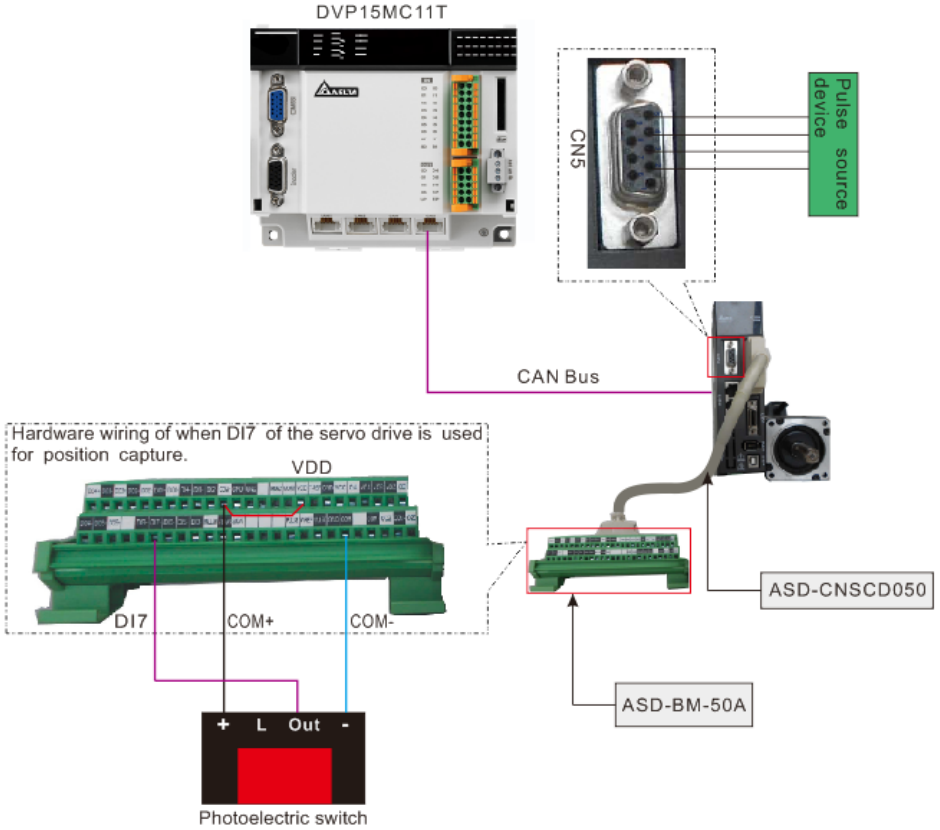
- Only DVP15MC11T supports mode 3.



Position Capture

Example of Wiring for Signal Capture in Mode 4

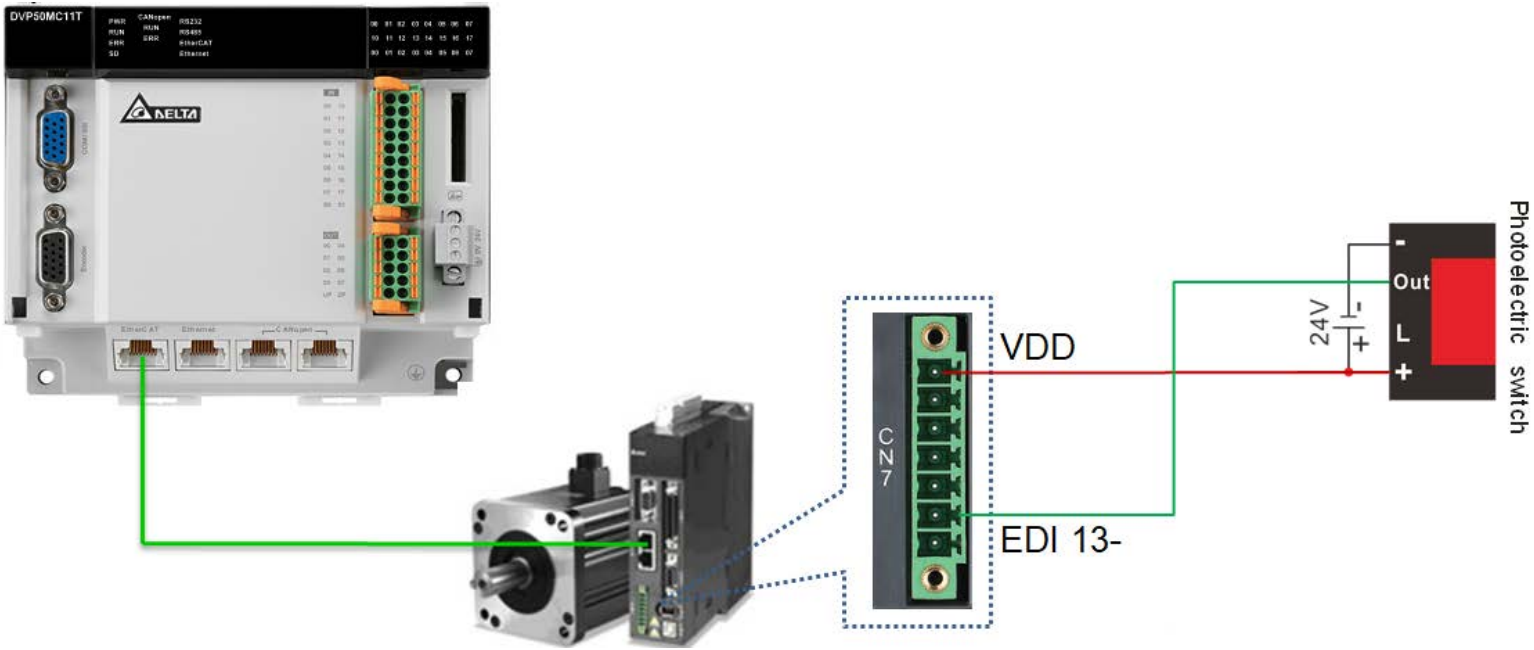
- Only DVP15MC11T supports mode 4.



Motion exercise 10

Example of Wiring for Signal Capture in Mode 5, 6 and 7

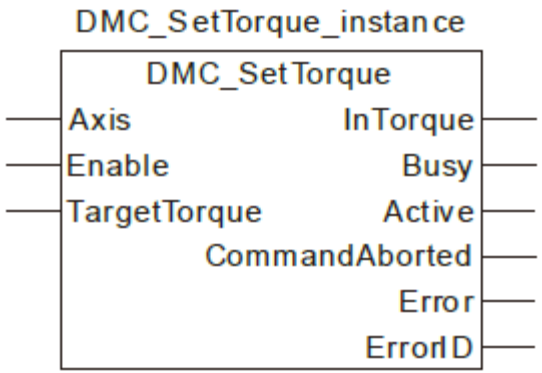
- Only DVP50MC11T supports mode 5, 6, and 7.



Torque Setting

DMC_SetTorque Instruction

- The instruction is used for controlling the servo axis to run in the torque mode according to the specified torque.
- ***TargetTorque* is the specified torque with the unit: permillage;**
- When TargetTorque value is positive, the axis is controlled to move in the positive direction. When TargetTorque value is negative, the axis is controlled to move in the negative direction.
- DMC_SetTorque can only be interrupted by MC_Stop. If the instruction execution need be stopped, change *Enable* parameter to FALSE.



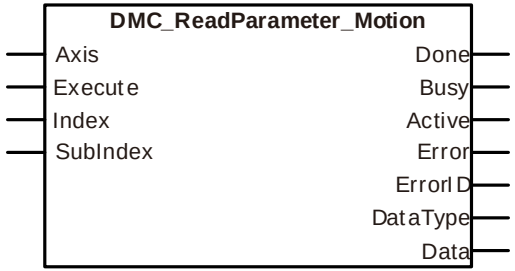
Read & Write Parameter

Introduction

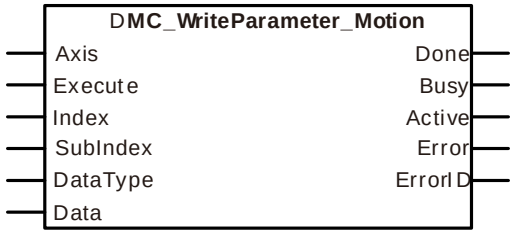
- **DMC_ReadParameter_Motion** **reads** a parameter of the slave (servo drive) in the Motion network.
- **DMC_WriteParameter_Motion** **sets** a parameter of the slave (servo drive) in the Motion network.

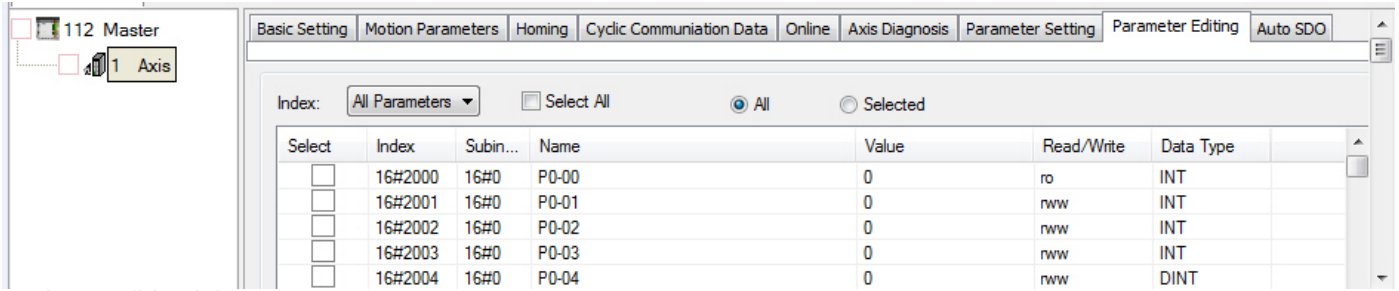
Instruction parameter	Explanation
Index	Prime index of a servo parameter （CANopen protocol） E.g. Index of servo parameter P6-10: 16#260A=16#2000+16#60A
Sub-Index	Sub-index of a servo parameter （CANopen protocol）
DataType	Data type of the servo parameter: Byte (1), Word (2) and Dword (4)
Data	The value of the servo parameter

DMC_ReadParameter_Motion_instance



DMC_WriteParameter_Motion_instance





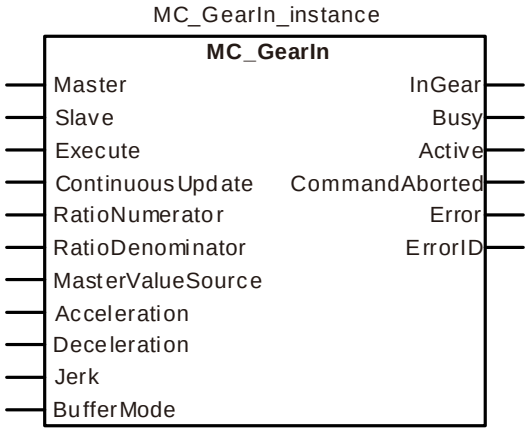
Motion exercise 11

Electronic Gear

MC_GearIn

- MC_GearIn controls the slave axis to follow the master axis for the synchronized motion based on the set gear ratio.
- If the gear ratio is a positive number, the directions of slave axis and master axis are the same. Otherwise, the directions of slave axis and master axis are opposite if the gear ratio is a negative number.

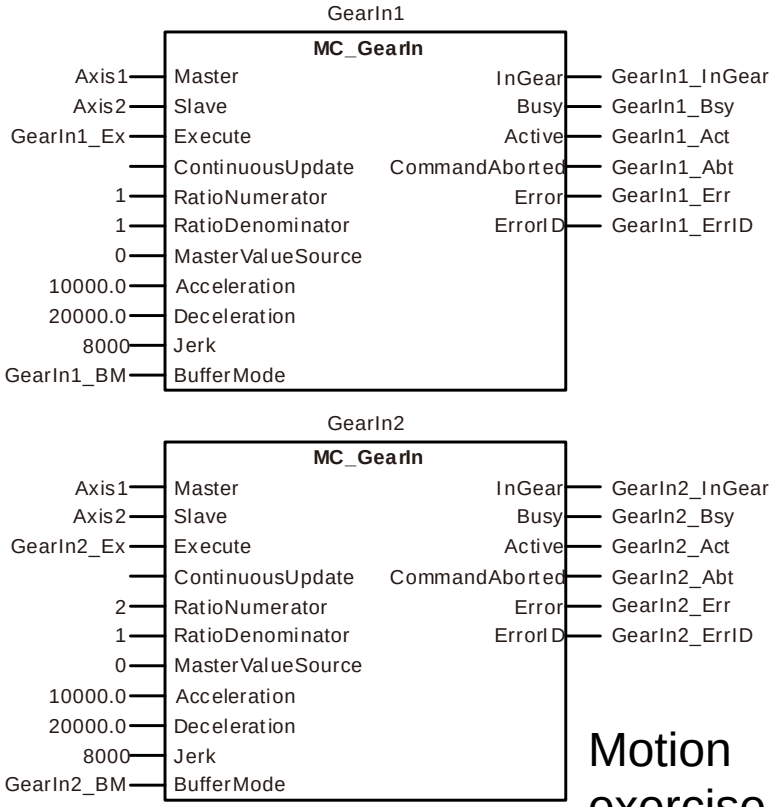
$$\text{Position increment of Slave axis} = \text{Position increment of Master axis} * \frac{\text{RatioNumerator}}{\text{RatioDenominator}}$$

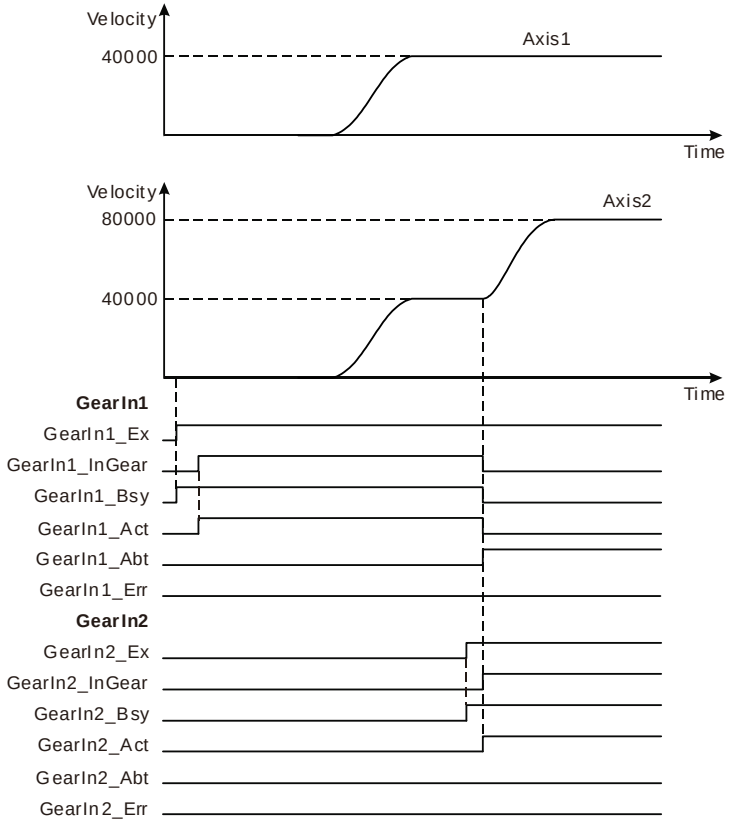


Electronic Gear

Example

The encoder axis can be used as the master axis.



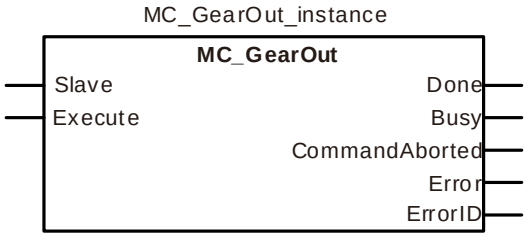


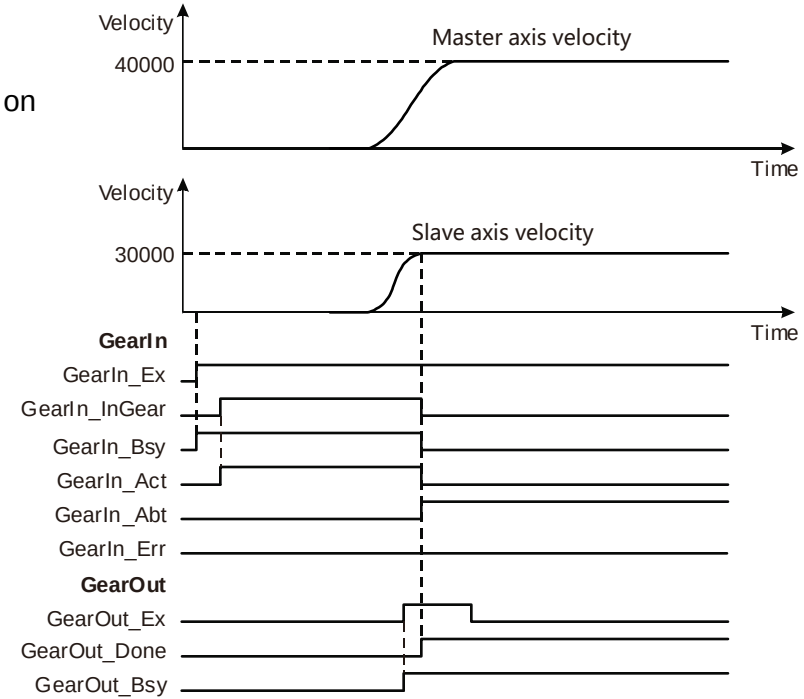
Motion exercise 12

Electronic Gear

MC_GearOut

- MC_GearOut ends the established e-gear relationship.
- After the e-gear relationship is disconnected, the slave axis will keep on moving at the velocity at the time of disconnection.

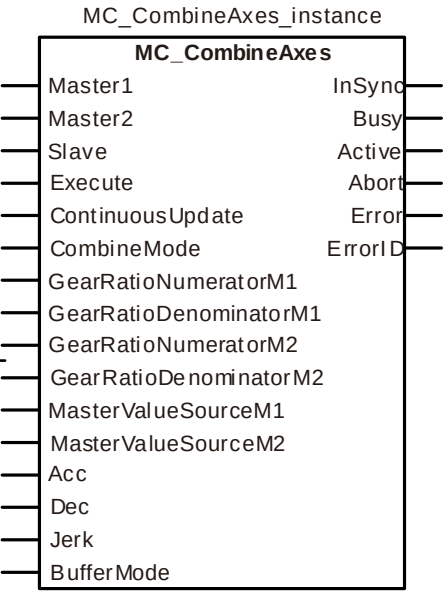




Two-Master-Axis-Combined Gear

MC_CombineAxes

- MC_CombineAxes controls the slave axis to follow two master axes for the synchronized gear motion.
- The effects the two master axes have on the slave axis can be added or subtracted.
- Executing MC_Stop on the slave axis can disconnect the established two-master-axis-combined gear relationship.



Addition Mode

$$\text{Position increment of Slave axis} = \text{Position increment of Master axis 1} * \frac{\text{GearRatioNumeratorM1}}{\text{GearRatioDenominatorM1}} + \text{Position increment of Master axis 2} * \frac{\text{GearRatioNumeratorM2}}{\text{GearRatioDenominatorM2}}$$

Subtraction Mode

$$\text{Position increment of Slave axis} = \text{Position increment of Master axis 1} * \frac{\text{GearRatioNumeratorM1}}{\text{GearRatioDenominatorM1}} - \text{Position increment of Master axis 2} * \frac{\text{GearRatioNumeratorM2}}{\text{GearRatioDenominatorM2}}$$

Motion exercise 13

Smarter. Greener. Together.

To learn more about Delta, please visit www.deltaww.com.