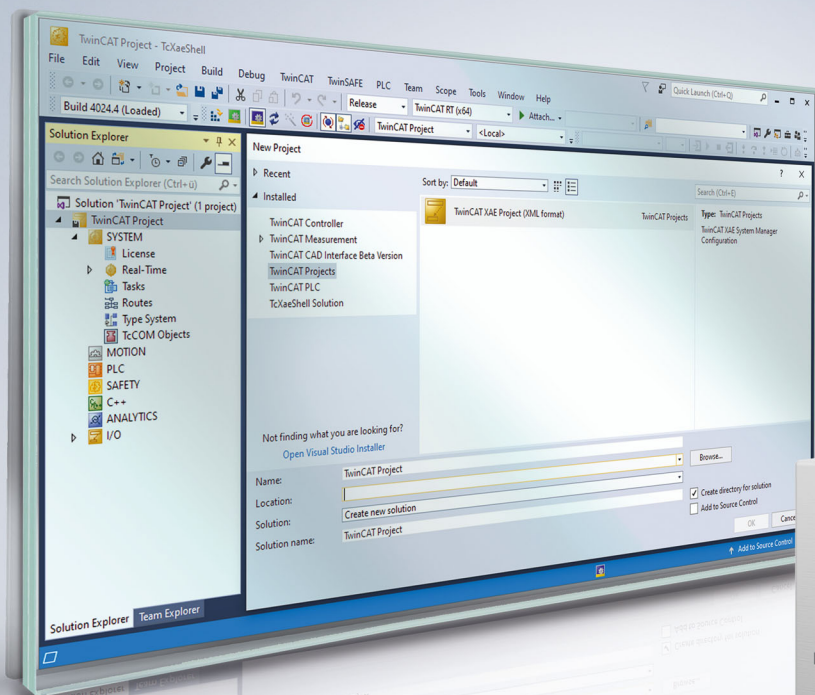


# BECKHOFF New Automation Technology

Manual | EN

# TE1000

TwinCAT 3 | PLC Library: Tc2\_MC2





# Table of contents

<b>1</b>	<b>Foreword</b> .....	<b>7</b>
1.1	Notes on the documentation .....	7
1.2	For your safety .....	8
1.3	Notes on information security.....	9
<b>2</b>	<b>Overview</b> .....	<b>10</b>
<b>3</b>	<b>State diagram</b> .....	<b>11</b>
<b>4</b>	<b>General rules for MC function blocks</b> .....	<b>14</b>
<b>5</b>	<b>Organization blocks</b> .....	<b>17</b>
5.1	Axis functions .....	17
5.1.1	MC_Power .....	17
5.1.2	MC_Reset .....	18
5.1.3	MC_SetPosition .....	19
5.2	Status and parameter.....	21
5.2.1	MC_ReadActualPosition .....	21
5.2.2	MC_ReadActualVelocity .....	22
5.2.3	MC_ReadAxisComponents.....	23
5.2.4	MC_ReadAxisError .....	24
5.2.5	MC_ReadBoolParameter .....	25
5.2.6	MC_ReadParameter .....	27
5.2.7	MC_ReadParameterSet.....	28
5.2.8	MC_ReadStatus.....	29
5.2.9	MC_WriteBoolParameter .....	31
5.2.10	MC_WriteParameter .....	32
5.2.11	MC_WriteBoolParameterPersistent .....	33
5.2.12	MC_WriteParameterPersistent .....	34
5.3	Touch probe .....	35
5.3.1	MC_TouchProbe .....	35
5.3.2	MC_AbortTrigger.....	39
5.4	External set value generator .....	40
5.4.1	MC_ExtSetPointGenEnable.....	40
5.4.2	MC_ExtSetPointGenDisable .....	42
5.4.3	MC_ExtSetPointGenFeed.....	43
5.4.4	MC_ExtSetPointGenFeedWithTorque .....	44
5.5	Special extensions .....	45
5.5.1	DriveOperationMode .....	45
5.5.2	MC_BacklashCompensation.....	47
5.5.3	MC_CalcDynamicsByRampTime .....	49
5.5.4	MC_OverrideFilter.....	51
5.5.5	MC_PositionCorrectionLimiter .....	52
5.5.6	MC_ReadDriveAddress .....	54
5.5.7	MC_SelectControlLoop.....	55
5.5.8	MC_SetAcceptBlockedDriveSignal .....	56
5.5.9	MC_SetEncoderScalingFactor.....	57

5.5.10	MC_SetOverride .....	58
5.5.11	MC_WriteNcIoOutput .....	59
5.5.12	MC_TableBasedPositionCompensation .....	60
5.5.13	MC_ConvertIncPosToPos .....	63
5.5.14	MC_ConvertPosToIncPos .....	64
<b>6</b>	<b>Motion function blocks .....</b>	<b>65</b>
6.1	Point to point motion .....	65
6.1.1	MC_MoveAbsolute .....	65
6.1.2	MC_MoveRelative .....	67
6.1.3	MC_MoveAdditive .....	69
6.1.4	MC_MoveModulo .....	71
6.1.5	Notes on modulo positioning .....	74
6.1.6	MC_MoveVelocity .....	79
6.1.7	MC_MoveContinuousAbsolute .....	81
6.1.8	MC_MoveContinuousRelative .....	83
6.1.9	MC_Halt .....	85
6.1.10	MC_Stop .....	87
6.2	Superposition .....	89
6.2.1	MC_MoveSuperImposed .....	89
6.2.2	Application examples for MC_MoveSuperImposed .....	91
6.2.3	MC_AbortSuperposition .....	95
6.3	Homing .....	96
6.3.1	MC_Home .....	96
6.4	Manual motion .....	98
6.4.1	MC_Jog .....	98
6.5	Axis coupling .....	101
6.5.1	MC_GearIn .....	101
6.5.2	MC_GearInDyn .....	102
6.5.3	MC_GearOut .....	104
6.5.4	MC_GearInMultiMaster .....	106
6.6	Phasing .....	109
6.6.1	MC_HaltPhasing .....	109
6.6.2	MC_PhasingAbsolute .....	111
6.6.3	MC_PhasingRelative .....	113
6.7	Torque Control .....	115
6.7.1	MC_TorqueControl .....	115
<b>7</b>	<b>Data types .....</b>	<b>118</b>
7.1	Axis interface .....	118
7.1.1	ST_AdAddress .....	118
7.1.2	AXIS_REF .....	118
7.1.3	NCTOPLC_AXIS_REF .....	119
7.1.4	NCTOPLC_AXIS_REF_OPMODE .....	122
7.1.5	NCTOPLC_AXIS_REF_STATE .....	123
7.1.6	NCTOPLC_AXIS_REF_STATE2 .....	124
7.1.7	NCTOPLC_AXIS_REF_STATE2_FLAGS .....	124

7.1.8	NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE .....	125
7.1.9	PLCTONC_AXIS_REF .....	125
7.1.10	PLCTONC_AXIS_REF_CTRL .....	126
7.2	DriveOperationMode .....	127
7.2.1	E_DriveOperationMode .....	127
7.3	External setpoint generator .....	127
7.3.1	ST_ExtSetPointEnableOptions .....	127
7.4	Gearing .....	128
7.4.1	E_GearInMultiMasterSyncMode .....	128
7.4.2	ST_GearInDynOptions .....	128
7.4.3	ST_GearInMultiMasterOptions .....	128
7.4.4	ST_GearInOptions .....	129
7.5	Homing .....	129
7.5.1	E_EncoderReferenceMode .....	129
7.5.2	MC_HomingMode .....	130
7.5.3	ST_HomingOptions .....	130
7.6	Motion .....	131
7.6.1	E_JogMode .....	131
7.6.2	E_PositionType .....	132
7.6.3	MC_BufferMode .....	132
7.6.4	MC_Direction .....	135
7.6.5	ST_MoveOptions .....	135
7.7	Nc Process Data .....	136
7.7.1	E_NcloDevice .....	136
7.7.2	E_NcloOutput .....	136
7.8	Phasing .....	136
7.8.1	E_PhasingType .....	136
7.9	Position Correction .....	137
7.9.1	E_AxisPositionCorrectionMode .....	137
7.9.2	ST_PositionCompensationTableElement .....	137
7.9.3	ST_PositionCompensationTableParameter .....	137
7.10	SelectControlLoop .....	138
7.10.1	E_SelectControlLoopType .....	138
7.11	Status and parameter .....	138
7.11.1	E_AxisErrorCodes .....	138
7.11.2	E_NcAxisType .....	138
7.11.3	E_NcDriveType .....	139
7.11.4	E_NcEncoderType .....	140
7.11.5	E_ReadMode .....	141
7.11.6	E_SetScalingFactorMode .....	141
7.11.7	E_WorkDirection .....	142
7.11.8	MC_AxisParameter .....	142
7.11.9	MC_AxisStates .....	144
7.11.10	ST_AxisComponents .....	144
7.11.11	ST_AxisOpModes .....	145
7.11.12	ST_AxisParameterSet .....	145

7.11.13	ST_AxisStatus.....	146
7.11.14	ST_DriveAddress.....	148
7.11.15	ST_McOutputs.....	148
7.11.16	ST_NcApplicationRequest.....	148
7.11.17	ST_SetEncoderScalingOptions.....	148
7.11.18	ST_SetPositionOptions.....	148
7.11.19	ST_NcPositionConversionOptions.....	149
7.12	Stepper.....	149
7.12.1	E_DestallDetectMode.....	149
7.12.2	E_DestallMode.....	149
7.12.3	ST_PowerStepperStruct.....	150
7.13	Superposition.....	150
7.13.1	E_SuperpositionAbortOption.....	150
7.13.2	E_SuperpositionMode.....	150
7.13.3	ST_SuperpositionOptions.....	152
7.14	Torque Control.....	153
7.14.1	ST_TorqueControlOptions.....	153
7.15	Touch probe.....	153
7.15.1	E_SignalEdge.....	153
7.15.2	E_SignalSource.....	154
7.15.3	MC_TouchProbeRecordedData.....	154
7.15.4	TRIGGER_REF.....	154
7.15.5	E_TouchProbeMode.....	156
7.15.6	E_Touch Probe.....	156
<b>8</b>	<b>Global constants.....</b>	<b>157</b>
8.1	Library version.....	157
<b>9</b>	<b>Examples.....</b>	<b>158</b>
<b>10</b>	<b>Support and Service.....</b>	<b>159</b>
<b>11</b>	<b>Appendix.....</b>	<b>160</b>
11.1	NC Backlash Compensation.....	160
11.1.1	Mechanical backlash.....	160
11.1.2	NC Implementing of the TwinCAT Position Correction.....	161
11.1.3	NC Implementing the TwinCAT Backlash Compensation.....	162

# 1 Foreword

## 1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.

The documentation and the following notes and explanations must be complied with when installing and commissioning the components.

The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

### Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without notice.

Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS®, and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.

If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

**EtherCAT** 

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

### Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: <https://www.beckhoff.com/trademarks>.

## 1.2 For your safety

### Safety regulations

Read the following explanations for your safety.

Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

### Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

### Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

### Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

#### Personal injury warnings

**⚠ DANGER**

Hazard with high risk of death or serious injury.

**⚠ WARNING**

Hazard with medium risk of death or serious injury.

**⚠ CAUTION**

There is a low-risk hazard that could result in medium or minor injury.

#### Warning of damage to property or environment

**NOTICE**

The environment, equipment, or data may be damaged.

#### Information on handling the product



This information includes, for example:  
recommendations for action, assistance or further information on the product.



## 1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <https://www.beckhoff.com/secguide>.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at <https://www.beckhoff.com/secinfo>.

## 2 Overview

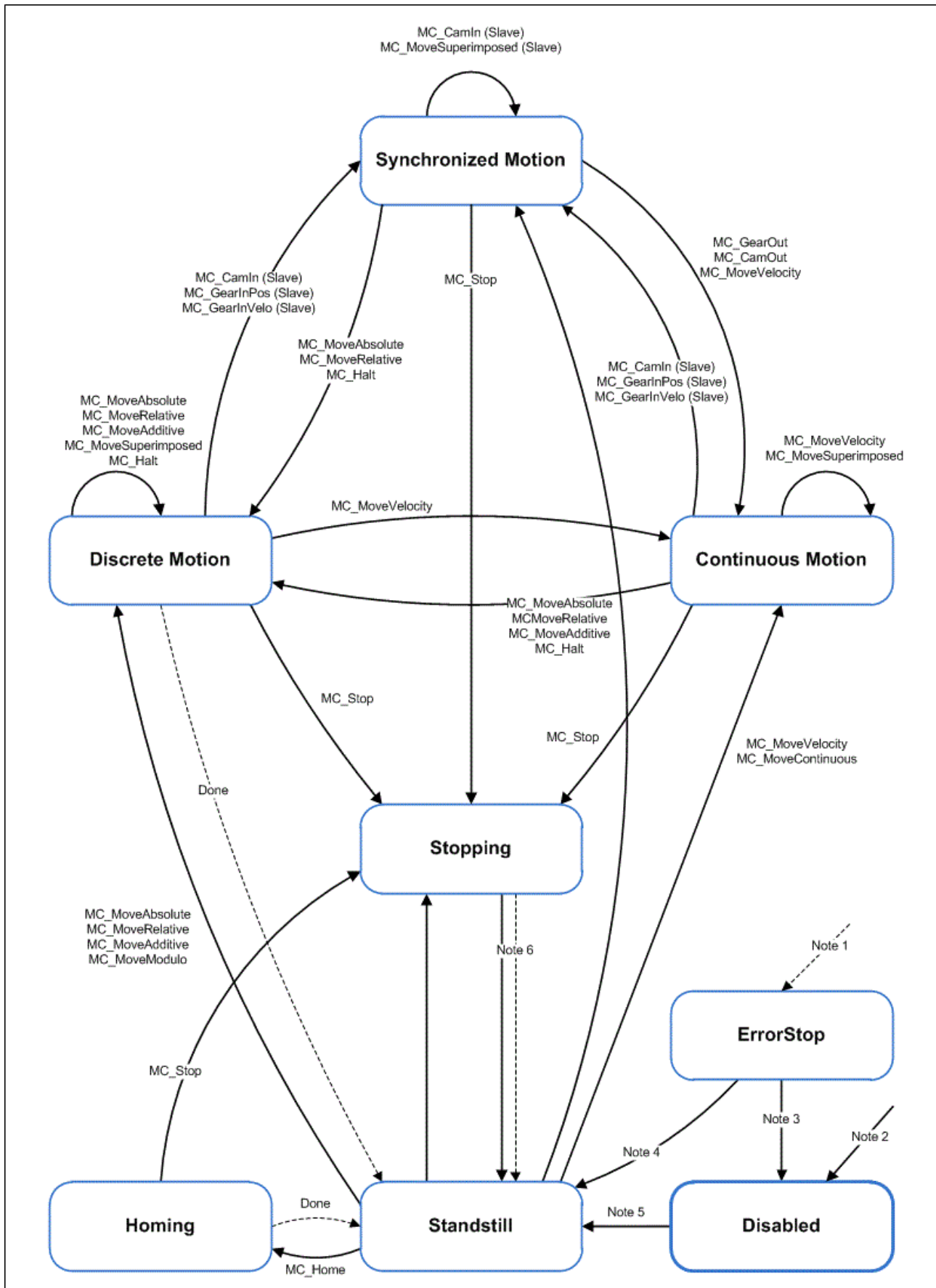
The TwinCAT Motion Control PLC library Tc2\_MC2 contains function blocks for programming machine applications. It is based on the PLCopen specification for Motion Control function blocks V2.0 ([www.PLCopen.org](http://www.PLCopen.org)).



This is a converted TwinCAT 2 TcMC2 library. Existing projects that still use the TwinCAT 2 TcMC library must first be adapted to the TcMC2 library before they can be converted for TwinCAT 3.

### 3 State diagram

The following state diagram defines the behavior of an axis in situations where several function blocks are simultaneously active for this axis. The combination of several function blocks is useful for generating more sophisticated motion profiles or for dealing with exceptional situations during program execution.



Note 1	From any state in which an error occurs
Note 2	From any state if MC_Power.Enable = FALSE and the axis has no error
Note 3	MC_Reset and MC_Power.Status = FALSE

Note 4	MC_Reset and MC_Power.Status = TRUE and MC_Power.Enable = TRUE
Note 5	MC_Power.Status = TRUE and MC_Power.Enable = TRUE
Note 6	MC_Stop.Done = TRUE and MC_Stop.Execute = FALSE

Motion commands are always processed sequentially. All commands operate in the described state diagram.

The axis is always in one of the defined states. Each motion command that causes a transition changes the state of the axis and thus the motion profile. The state diagram is an abstraction layer that reflects the real axis state, comparable to the process image for I/O points. The axis state changes immediately when the command is issued.

The state diagram refers to single axes. Multi-axis blocks such as MC\_CamIn or MC\_GearIn influence the states of several axes, which can always be traced back to individual axis states of the axes involved in the process. For example, a cam plate master can be in "Continuous Motion" state, while the associated slave is in "Synchronized Motion" state. Coupling of a slave has no influence on the state of the master.

The "Disabled" state is the default state of an axis. In this state can the axis cannot be moved through a function block. When the MC\_Power function block is called with Enable = TRUE, the axis changes to the "Standstill" state or, in the event of an error, to "ErrorStop" state. If the function block MC\_Power is called with Enable = FALSE, the state changes to "Disabled".

The purpose of "ErrorStop" state is to stop the axis and then block further commands, until a reset was triggered. The "Error" state transition only refers to actual axis errors and not to execution errors of a function block. Axis errors can also be displayed at the error output of a function block.

Function blocks that are not listed in the state diagram do not affect the state of the axis (MC\_ReadStatus, MC\_ReadAxisError, MC\_ReadParameter, MC\_ReadBoolParameter, MC\_WriteParameter, MC\_WriteBoolParameter, MC\_ReadActualPosition and MC\_CamTableSelect).

The "Stopping" state indicates that the axis is in a stop ramp. The state changes after the complete stop after "Standstill".

Motion commands such as MC\_MoveAbsolute that lead out of the "Synchronized Motion" state are possible only if they are explicitly permitted in the axis parameters. Uncoupling commands such as MC\_GearOut are possible independent of that.

## 4 General rules for MC function blocks

The rules described below apply to all MC function blocks. They ensure a defined processing by the PLC program.

### Exclusivity of the outputs

The outputs "Busy", "Done", "Error" and "CommandAborted" are mutually exclusive, i.e. only one of these outputs can be TRUE on a function block at the same time. When the "Execute" input becomes TRUE, one of the outputs must become TRUE. At the same time, however, only one of the outputs "Active", "Done", "Error" and "CommandAborted" can be TRUE.

An exception is the motion command [MC\\_Stop](#) [▶ 87]. This sets the "Done" output to TRUE as soon as the axis is stopped. Nevertheless, the "Busy" and "Active" outputs remain TRUE because the axis is locked. The axis is unlocked and the "Busy" and "Active" outputs set to FALSE only after the "Execute" input is set to FALSE.

### Initial state

If the function block is not active, the outputs "Done", "InGear", "InVelocity", "Error", "ErrorID" and "CommandAborted" are reset with a falling edge at input "Execute". However, the falling edge at input "Execute" does not affect the command execution.

If the "Execute" input is already reset during command execution, this ensures that one of the outputs is set at the end of the command for a PLC cycle. Only then are the outputs reset.

If the "Execute" input is triggered more than once during the execution of a command, the function block does not provide any feedback, nor does it execute any further commands.

### Input parameters

The input parameters become active with a positive edge. To change the parameters the command has to be triggered again once it is completed, or a second instance of the function block must be triggered with new parameters during command execution.

If an input parameter is not passed to the function block, the last value passed to this function block remains valid. A meaningful default value is used for the first call.

### Position and Distance

The "Position" input designates a defined value within a coordinate system. The "Distance" input, on the other hand, is a relative dimension, i.e. the distance between two positions. "Position" and "Distance" are specified in technical units, e.g. mm or °, according to the axis scaling.

### Dynamic parameters

The dynamic parameters for Move functions are specified in technical units with second as timebase. For example, if an axis is scaled in millimeters, the parameters have the following units:

Velocity	mm/s
Acceleration	mm/s <sup>2</sup>
Deceleration	mm/s <sup>2</sup>
Jerk	mm/s <sup>3</sup>

### Error handling

All function blocks have two error outputs for indicating errors during command execution. The "Error" output indicates the error and the "ErrorID" output contains a supplementary error number. The "Done", "InVelocity", "InGear" and "InSync" outputs indicate successful command execution and are not set if the "Error" output is TRUE.

Errors of different type are signaled at the function block output. The error type is not specified explicitly. It depends on the unique and global error number.

## Error types

- Function block errors only concern the function block, not the axis (e.g. incorrect parameterization). Function block errors do not have to be reset explicitly. They are reset automatically when the "Execute" input is reset.
- Communication errors (the function block cannot address the axis, for example). Communication errors usually indicate incorrect configuration or parameterization. A Reset is not possible. The function block can be retriggered after the configuration has been corrected.
- Axis errors (logical NC axis) usually occur during the motion (e.g. lag error). They cause the axis to switch to error state. An axis error must be reset with the motion command [MC\\_Reset \[► 18\]](#).
- Drive errors (controller) may cause an axis error, i.e. an error of the logical NC axis. In many cases, axis and drive errors can be reset together via the motion command [MC\\_Reset \[► 18\]](#). Depending on the drive controller, a separate reset mechanism may be required (e.g. connection of a reset line to the control device).

## Behavior of the Done output

The output "Done" (or alternatively "InVelocity", "InGear", "InSync" etc.) is set if a command was executed successfully. If several function blocks are used for an axis and the running command is interrupted through a further function block, the Done output for the first function block is not set.

## Behavior of the CommandAborted output

The "CommandAborted" output is set if a command is interrupted by another function block.

## Behavior of the Busy output

The "Busy" output indicates that the function block is active. The function block can only be triggered by a rising edge at the "Execute" input if the "Busy" output is FALSE. "Busy" is immediately set with the positive edge at the "Execute" input and is not reset until the command has been successfully or unsuccessfully terminated. As long as the "Busy" output is TRUE, the function block must be called cyclically in order to be able to execute the command.

## Behavior of the Active output

If the axis movement is controlled by several function blocks, the "Active" output of a function block indicates that the axis executes the command. The state Busy = TRUE and Active = FALSE means that the command has not yet been executed or is no longer executed.

## Enable input and Valid output

In contrast to the "Execute" input, the "Enable" input causes an action to be executed continuously and repeatedly, as long as "Enable" is TRUE. For example, the function block [MC\\_ReadStatus \[► 29\]](#) cyclically updates the status of an axis as long as "Enable" is TRUE. A function block with an "Enable" input indicates through the "Valid" output that the data indicated at the outputs are valid. However, the data can be updated continuously as long as the "Valid" output is TRUE.

## BufferMode

Some function blocks have a "BufferMode" input for controlling the command flow with several function blocks. For example, BufferMode can specify that a command interrupts another command (unbuffered mode) or that the following command is only executed after the previous command (buffered mode). BufferMode can be used to specify the movement transition from one command to the next. This is referred to as "Blending", which specifies the velocity at the transition point.

A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.

In unbuffered mode a subsequent command leads to termination of a running command. The previous command then sets the output "CommandAborted". In BufferMode a subsequent command waits until a running command is completed. Note that a continuous motion does not allow a buffered follow-on command. Buffered commands always lead immediately to an endless movement being aborted, as in unbuffered operation.

Only one command is buffered while another command is executed. If more than one command is triggered during a running command, then the last-started command to be buffered is rejected with an error (error 0x4292 Buffer Full). However, if the last command is started unbuffered, it becomes active in any case and interrupts the current and an already buffered command.

### BufferModes

- **Aborting:** Default mode without buffering. The command is executed immediately and interrupts any other command that may be running.
- **Buffered:** The command is executed once no other command is running on the axis. The previous movement continues until it has stopped. The following command is started from standstill.
- **BlendingLow:** The command is executed once no other command is running on the axis. In contrast to Buffered the axis does not stop at the previous target, but passes through this position with the lower velocity of two commands.
- **BlendingHigh:** The command is executed once no other command is running on the axis. In contrast to Buffered the axis does not stop at the previous target, but passes through this position with the higher velocity of two commands.
- **BlendingNext:** The command is executed once no other command is running on the axis. In contrast to Buffered the axis does not stop at the previous target, but passes through this position with the velocity of the last command.
- **BlendingPrevious:** The command is executed once no other command is running on the axis. In contrast to Buffered the axis does not stop at the previous target, but passes through this position with the velocity of the first command.

See also: [Diagram of the BufferModes](#) [► 132]

### Optional blending position

Blending in the different BufferModes takes place in each case at the target position of the currently running command. In the case of the motion command [MC\\_MoveVelocity](#) [► 79] no target position is defined, and in other cases it may make sense to change the blending position. For this purpose, a blending position can be defined via the "Options" input of the function block (see [Options input](#) [► 16]), which is then used for the new command. The optional blending position must be located before the target position of the previous command, otherwise the new command will be rejected with an error message (0x4296). If the optional blending position has already been passed, the new command is implemented instantaneously. In other words, the command behaves like an aborting command.

### Option input

Many function blocks have an "Options" input with a data structure containing additional, infrequently required options. To execute the basic function of the function block these options are often not required, so that the input can remain open. The user only has to occupy the Options data structure in cases where the documentation explicitly refers to certain options.

### Slave axes

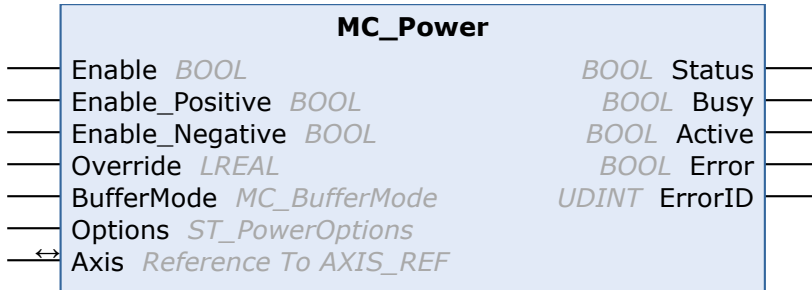
Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as [MC\\_MoveAbsolute](#) [► 65] then automatically leads to uncoupling of the axis, after which the command is executed. In this case the only available BufferMode is "Aborting".



## 5 Organization blocks

### 5.1 Axis functions

#### 5.1.1 MC\_Power



The function block is used to switch the software enable of an axis. Enable can be activated for both directions of travel or only one direction. At "Status" output operational readiness of the axis is indicated.

A velocity override influences the velocity of all travel commands by a specified percentage.

Depending on the drive type, "Status" also signals operational readiness of the drive. Digital drives provide feedback on operational readiness, while analog drives are unable to indicate their operational readiness. In the latter case Status only indicated operational readiness of the control side.



In addition to software enable it may be necessary to activate a hardware enable signal in order to enable a drive. This signal is not influenced by MC\_Power and must be activated separately by the PLC.

#### Inputs

```

VAR_INPUT
  Enable       : BOOL; (* B *)
  Enable_Positive : BOOL; (* E *)
  Enable_Negative : BOOL; (* E *)
  Override     : LREAL (* V *) := 100.0; (* in percent - Beckhoff proprietary input *)
  BufferMode   : MC_BufferMode; (* V *)
  Options     : ST_PowerOptions;
END_VAR
    
```

Name	Type	Description
Enable	BOOL	General software enable for the axis.
Enable_Positive	BOOL	Feed enable in positive direction. Only takes effect if Enable = TRUE.
Enable_Negative	BOOL	Feed enable in negative direction. Only takes effect if Enable = TRUE.
Override	LREAL	Velocity override in % for all movement commands. (0 ≤ Override ≤ 100.0)
BufferMode	MC_BufferMode [▶ 132]	This is evaluated when "Enable" is reset. MC_Aborting mode leads to immediate deactivation of the axis enable. Otherwise, e.g. in "MC_Buffered" mode, the function block waits until the axis no longer executes a command.
Options	ST_PowerOptions	Not implemented.

See also: [General rules for MC function blocks \[▶ 14\]](#)

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Status : BOOL; (* B *)
  Busy : BOOL; (* V *)
  Active : BOOL; (* V *)
  Error : BOOL; (* B *)
  ErrorID : UDINT; (* E *)
END_VAR
```

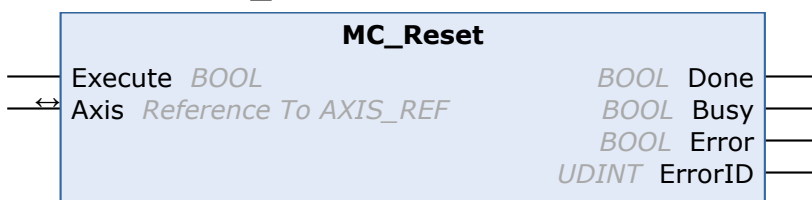
Name	Type	Description
Status	BOOL	TRUE when the axis is ready for operation.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Active	BOOL	Indicates that the command is executed.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[▶ 14\]](#)

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.1.2 MC\_Reset**



The function block is used to reset the NC axis. In many cases this also leads to a reset of a connected drive device. Depending on the bus system or drive types, in some cases a separate reset may be required for the drive device.

 **Inputs**

```
VAR_INPUT
  Execute : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

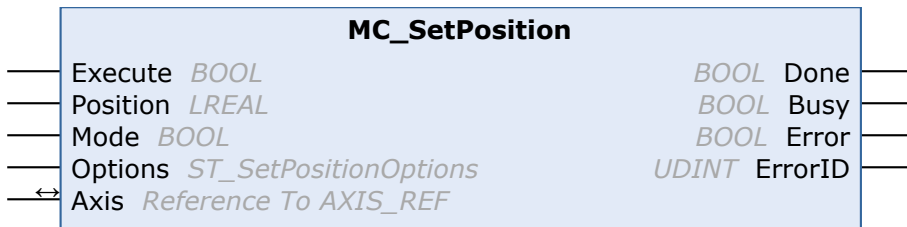
```
VAR_OUTPUT
  Done : BOOL;
  Busy : BOOL;
  Error : BOOL;
  ErrorID : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE, if the reset was executed successfully.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.1.3 MC\_SetPosition**



The function block sets the current axis position to a parameterizable value.

In absolute mode, the actual position is set to the parameterized absolute Position value. In relative mode, the actual position is offset by the parameterized Position value. In both cases, the set position of the axis is set so that any potential lag error is retained. The `Options.ClearPositionLag` switch can be used to clear the lag error.

Relative mode can be used to change the axis position during the motion.

As of TwinCAT 3.1.4024.51 and Tc2\_MC2 3.3.56, the position offset can be deleted and the position can be reset to the originally referenced coordinate system via the `Options.ClearPositionOffset` switch. From this version onwards, the offset can also be read via ADS and displayed in the UserData variable of the AXIS\_REF structure so that the value is available for the current cycle.

 **Inputs**

```
VAR_INPUT
  Execute : BOOL;
  Position : LREAL;
```

```

Mode      : BOOL; (* RELATIVE=True, ABSOLUTE=False (Default) *)
Options   : ST_SetPositionOptions;
END_VAR

```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Position value that the axis position will be set to. In absolute mode, the actual position is set to this value, in relative mode it is shifted by this value.
Mode	BOOL	The axis position is set to an absolute value set if Mode = FALSE. Otherwise the axis position is changed relative to the specified Position value. Relative mode can be used for changing the position of an axis during motion.
Options	<a href="#">ST_SetPositionOptions</a> [▶ 148]	Data structure containing additional rarely used parameters. The input can normally remain open. <ul style="list-style-type: none"> <li>• <b>ClearPositionLag:</b> can optionally be used to set the set and actual positions to the same value. In this instance, the following error is cleared.</li> <li>• <b>SelectEncoderIndex:</b> can optionally be set if an axis with several encoders is used and the position of a certain encoder is to be set (Options.EncoderIndex).</li> <li>• <b>EncoderIndex:</b> indicates the encoder (0..n) if SelectEncoderIndex is TRUE.</li> <li>• <b>ClearPositionOffset:</b> Can be set to delete the position offset selected with MC_SetPosition and thus reset the position to the originally referenced coordinate system; available from TwinCAT 3.1.4024.51 and Tc2_MC2 3.3.56.</li> </ul>

See also: [General rules for MC function blocks](#) [▶ 14]

 **Inputs/outputs**

```

VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR

```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```

VAR_OUTPUT
  Done      : BOOL;
  Busy      : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
END_VAR

```

Name	Type	Description
Done	BOOL	TRUE, if the reset was executed successfully.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks](#) [▶ 14]

### Reading the MC\_SetPosition offset

As of TwinCAT 3.1.4024.51, the position offset set by MC\_SetPosition can be read out via ADS:

<b>Function</b>	ADS Read	
<b>Port</b>	501 (dec)	
<b>Index Group</b>	0x4100 + axis ID	0x5100 + axis ID
<b>Index Offset</b>	0x00n10017	0x0017
<b>Data</b>	REAL64	

### Merging the MC\_SetPosition offset into the AXIS\_REF

The AXIS\_REF axis structure contains a UserData variable that can be configured so that the current value of the MC\_SetPosition offset is displayed here from TwinCAT version 3.1.4024.51 onwards. The value is then available for the current cycle.

The UserData variable must be configured as follows via ADS:

<b>Function</b>	ADS Write	
<b>Port</b>	501 (dec)	
<b>Index Group</b>	0x4000 + axis ID	
<b>Index Offset</b>	0x010D	
<b>Data</b>	0x00000000: Deactivated (default) 0x00010017: Activates the MC_SetPosition Offset value	

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 5.2 Status and parameter

### 5.2.1 MC\_ReadActualPosition



The actual axis position can be read with the function block. The actual position value is the position value that is returned from the drive amplifier to the axis.

#### Inputs

```
VAR_INPUT
    Enable : BOOL;
END_VAR
```

Name	Type	Description
Enable	BOOL	Reads the parameter once or cyclically depending on the ReadMode.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

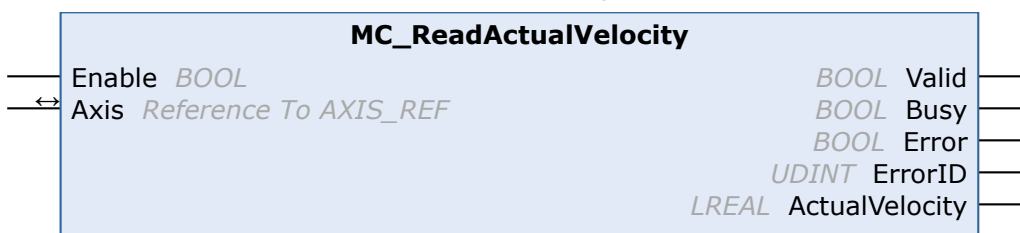
```
VAR_OUTPUT
  Valid : BOOL;
  Busy : BOOL;
  Error : BOOL;
  ErrorID : UDINT;
  Position : LREAL;
END_VAR
```

Name	Type	Description
Valid	BOOL	TRUE if the output "Position" has a valid value.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
Position	LREAL	Current axis position

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.2.2 MC\_ReadActualVelocity**



The actual axis velocity value can be read with the function block. The actual velocity value is the velocity value that is returned from the servo drive to the axis.

 **Inputs**

```
VAR_INPUT
  Enable : BOOL;
END_VAR
```

Name	Type	Description
Enable	BOOL	Reads the parameter once or cyclically depending on the ReadMode.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

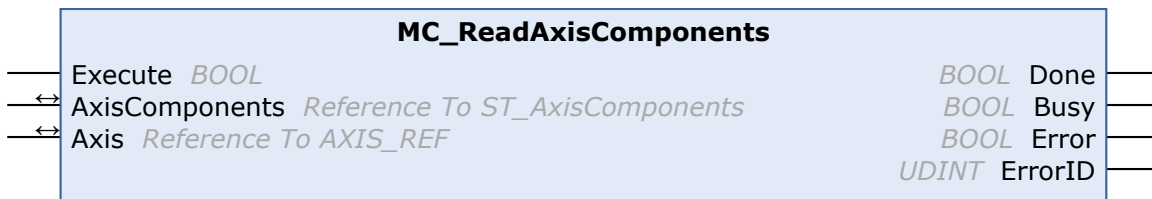
```
VAR_OUTPUT
  Valid      : BOOL;
  Busy       : BOOL;
  Error      : BOOL;
  ErrorID    : UDINT;
  ActualVelocity : LREAL;
END_VAR
```

Name	Type	Description
Valid	BOOL	Signals with TRUE that the value read at output Value is valid.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
ActualVelocity	LREAL	Current axis velocity

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.2.3 MC\_ReadAxisComponents**



The function block can be used to read information about the subelements encoder, drive and controller of an axis.



In this case "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

 **Inputs**

```
VAR_INPUT
  Execute : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.

 **Inputs/outputs**

```
VAR_IN_OUT
  AxisComponents : ST_AxisComponents;
  Axis           : AXIS_REF;
END_VAR
```

Name	Type	Description
AxisComponents	ST_AxisComponents [ <a href="#">▶ 144</a> ]	Data structure used to return the components (encoders, controllers and drives) of the axis.
Axis	AXIS_REF [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

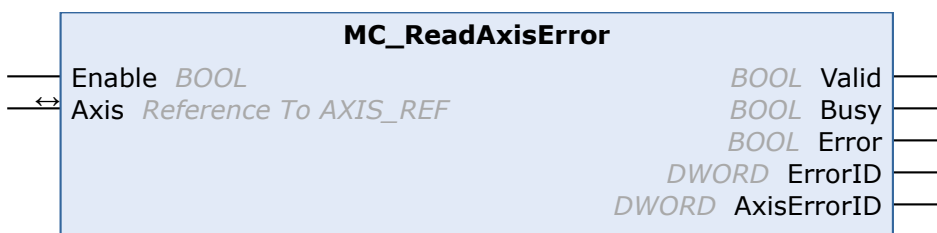
```
VAR_OUTPUT
  Done      : BOOL;
  Busy      : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the components have been read successfully.
Busy	BOOL	TRUE as soon as the Execute input of the function block has been set to TRUE and component reading has not yet been completed. Afterwards either the output is Done or Error TRUE and Busy is set to FALSE again.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.2.4 MC\_ReadAxisError**



The function block is used to read the axis error of an axis.

 **Inputs**

```
VAR_INPUT
  Enable : BOOL;
END_VAR
```

Name	Type	Description
Enable	BOOL	If Enable = TRUE, the axis error is output at the "AxisErrorID" output.

See also: [General rules for MC function blocks](#) [[▶ 14](#)]



 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Valid      : BOOL; (* B *)
  Busy       : BOOL; (* E *)
  Error      : BOOL; (* B *)
  ErrorID    : UDINT; (* B *)
  AxisErrorID : DWORD; (* B *)
END_VAR
```

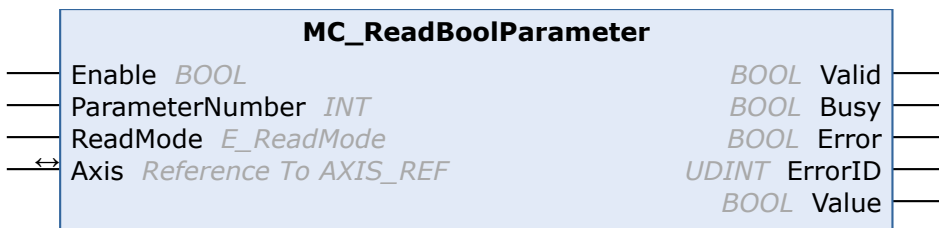
Name	Type	Description
Valid	BOOL	TRUE if the error signaled at the "AxisErrorID" output is valid.
Busy	BOOL	TRUE as soon as the command is started with Enable and as long as the command is processed. If Busy is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
AxisErrorID	DWORD	Error number for the axis

See also: [General rules for MC function blocks](#) [[▶ 14](#)]

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.2.5 MC\_ReadBoolParameter



The function block MC\_ReadBoolParameter is used to read a boolean axis parameter.



In this case "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

 **Inputs**

```
VAR_INPUT
  Enable       : BOOL;      (* B *)
  ParameterNumber : MC_AxisParameter; (* B *)
  ReadMode     : E_ReadMode (* V *)
END_VAR
```

Name	Type	Description
Enable	BOOL	Reads the parameter once or cyclically depending on the ReadMode.
ParameterNumber	<a href="#">MC_AxisParameter [▶ 142]</a>	Number of the parameter to be read
ReadMode	<a href="#">E_ReadMode [▶ 141]</a>	Read mode of the parameter to be read (once or cyclic)

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

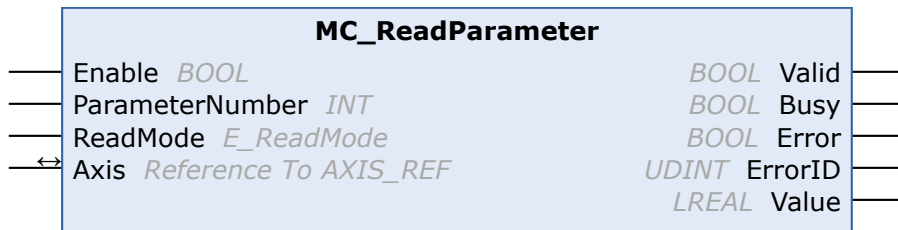
```
VAR_OUTPUT
  Valid   : BOOL; (* B *)
  Busy    : BOOL; (* E *)
  Error   : BOOL; (* B *)
  ErrorID : UDINT; (* E *)
  Value   : BOOL; (* B *)
END_VAR
```

Name	Type	Description
Valid	BOOL	Signals with TRUE that the value read at output Value is valid.
Busy	BOOL	TRUE as soon as the command is started with Enable and as long as the command is processed. If Busy is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
Value	BOOL	Shows the read Boolean value.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 5.2.6 MC\_ReadParameter



The function block MC\_ReadParameter is used to read an axis parameter.



In this case "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

### Inputs

```
VAR_INPUT
    Enable      : BOOL;      (* B *)
    ParameterNumber : MC_AxisParameter; (* B *)
    ReadMode    : E_ReadMode (* V *)
END_VAR
```

Name	Type	Description
Enable	BOOL	Reads the parameter once or cyclically depending on the ReadMode.
ParameterNumber	<a href="#">MC_AxisParameter [▶ 142]</a>	Number of the parameter to be read
ReadMode	<a href="#">E_ReadMode [▶ 141]</a>	Read mode of the parameter to be read (once or cyclic)

### Inputs/outputs

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
    Valid : BOOL; (* B *)
    Busy  : BOOL; (* E *)
    Error : BOOL; (* B *)
    ErrorID : DWORD; (* E *)
    Value : LREAL; (* B *)
END_VAR
```

Name	Type	Description
Valid	BOOL	Signals with TRUE that the value read at output Value is valid.
Busy	BOOL	TRUE as soon as the command is started with Enable and as long as the command is processed. If Busy is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	DWORD	If the error output is set, this parameter supplies the error number.

Name	Type	Description
Value	LREAL	Shows the read value.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.2.7 MC\_ReadParameterSet



The function block MC\_ReadParameterSet can be used to read the entire parameter set of an axis.



In this case "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

**Inputs**

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.

**Inputs/outputs**

```
VAR_IN_OUT
    Parameter : ST_AxisParameterSet;
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Parameter	ST_AxisParameterSet [ <a href="#">▶ 145</a> ]	Parameter data structure into which the parameters are read.
Axis	AXIS_REF [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Outputs**

```
VAR_OUTPUT
    Done : BOOL;
    Busy : BOOL;
    Error : BOOL;
    ErrorID : UDINT;
END_VAR
```

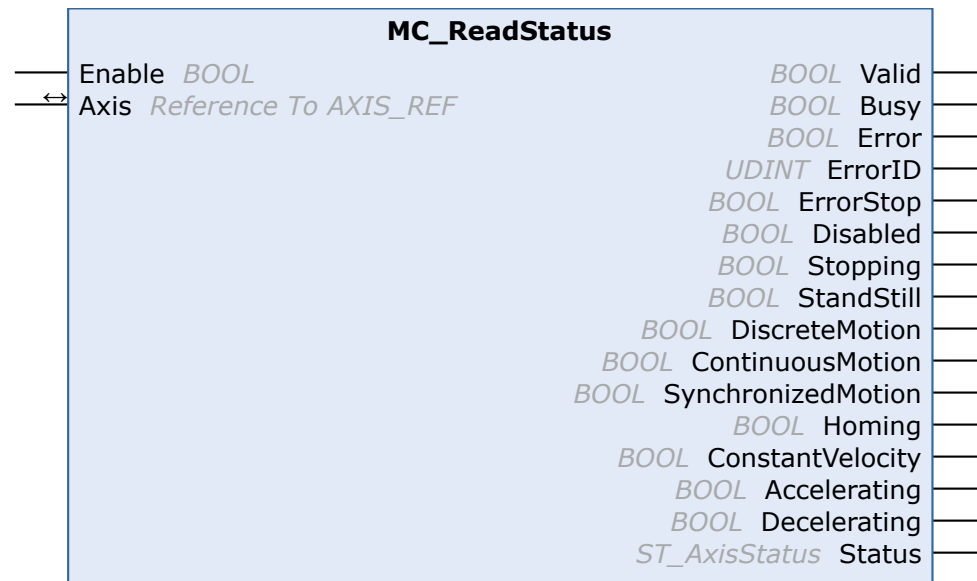
Name	Type	Description
Done	BOOL	TRUE if the parameters were read successfully.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Error	BOOL	TRUE, if an error occurs.

Name	Type	Description
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.2.8 MC\_ReadStatus



The MC\_ReadStatus function block determines the current operating state of an axis and signals it at the function block outputs.

The updated operating state is additionally stored in the Status output data structure and in the Axis.Status axis data structure. This means the operating state only has to be read once at the start of each PLC cycle and can then be accessed via Axis.Status.

The Axis variable (type [AXIS\\_REF \[► 118\]](#)) already includes an instance of the function block MC\_ReadStatus. This means that the operating state of an axis can be updated at the start of a PLC cycle by calling up Axis.ReadStatus.

**Sample:**

```
PROGRAM MAIN
VAR
  Axis1 : AXIS_REF
END_VAR

(* call the read status function *)
Axis1.ReadStatus;
```

**Inputs**

```
VAR_INPUT
  Enable : BOOL;
END_VAR
```

Name	Type	Description
Enable	BOOL	If Enable = TRUE, the axis operating state is updated with each call of the function block.

See also: [General rules for MC function blocks \[► 14\]](#)

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Valid          : BOOL;
  Busy           : BOOL;
  Error          : BOOL;
  ErrorId        : UDINT;
  (* motion control statemachine states: *)
  ErrorStop      : BOOL;
  Disabled       : BOOL;
  Stopping       : BOOL;
  StandStill     : BOOL;
  DiscreteMotion : BOOL;
  ContinuousMotion : BOOL;
  SynchronizedMotion : BOOL;
  Homing         : BOOL;
  (* additional status *)
  ConstantVelocity : BOOL;
  Accelerating   : BOOL;
  Decelerating   : BOOL;
  (* status data structure *)
  Status         : ST_AxisStatus;
END_VAR
```

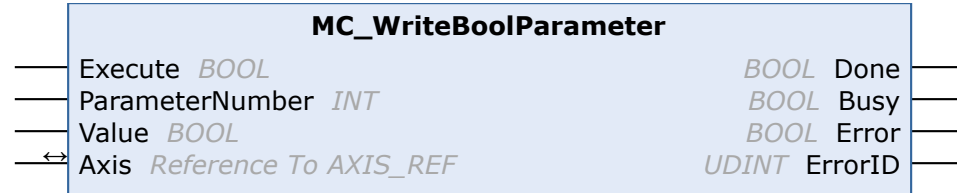
Name	Type	Description
Valid	BOOL	Indicates that the axis operating state indicated at the other outputs is valid.
Busy	BOOL	TRUE, as long as the function block is called with Enable = TRUE.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
ErrorStop	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
Disabled	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
Stopping	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
StandStill	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
DiscreteMotion	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
ContinuousMotion	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
SynchronizedMotion	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
Homing	BOOL	Axis status according to the <a href="#">PlcOpen state diagram [► 11]</a>
ConstantVelocity	BOOL	The axis is moving with constant velocity.
Accelerating	BOOL	The axis accelerates.
Decelerating	BOOL	The axis decelerates.
Status	ST_AxisStatus	Extended <a href="#">status data structure [► 146]</a> with additional status information. (Type: <a href="#">ST_AxisStatus [► 146]</a> )

See also: [General rules for MC function blocks \[► 14\]](#)

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 5.2.9 MC\_WriteBoolParameter



Boolean parameters for the axis can be written with the function block MC\_WriteBoolParameter.



In this case "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

### Inputs

```
VAR_INPUT
    Execute      : BOOL;
    ParameterNumber : MC_AxisParameter;
    Value        : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ParameterNumber	<a href="#">MC_AxisParameter [▶ 142]</a>	Number of the parameter to be written.
Value	BOOL	Boolean value that is written.

### Inputs/outputs

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
    Done      : BOOL;
    Busy      : BOOL;
    Error     : BOOL;
    ErrorID   : UDINT;
END_VAR
```

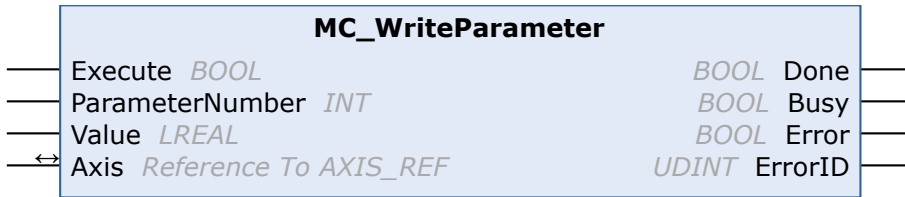
Name	Type	Description
Done	BOOL	TRUE if the parameters were written successfully.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done" or "Error" is set.

Name	Type	Description
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.2.10 MC\_WriteParameter**



Axis parameters can be written with the function block MC\_WriteParameter.



In this case "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

**Inputs**

```
VAR_INPUT
    Execute      : BOOL;
    ParameterNumber : MC_AxisParameter;
    Value        : LREAL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ParameterNumber	<a href="#">MC_AxisParameter [▶ 142]</a>	Number of the parameter to be written.
Value	LREAL	LREAL value that is written.

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Outputs**

```
VAR_OUTPUT
    Done      : BOOL;
    Busy      : BOOL;
    Error     : BOOL;
    ErrorID   : UDINT;
END_VAR
```



Name	Type	Description
Done	BOOL	TRUE if the parameters were written successfully.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done" or "Error" is set.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Sample**

```

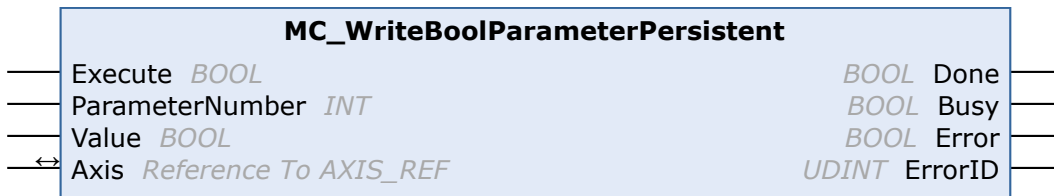
VAR
  Axis1      : Axis_REF;
  fbWriteParameter: MC_WriteParameter;
END_VAR

fbWriteParameter(
  Execute := TRUE;
  Axis:= Axis1 ,
  ParameterNumber:= MC_AxisParameter.SwLimitPos,
  Value:= 2000
);
    
```

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.2.11 MC\_WriteBoolParameterPersistent**



Boolean axis parameters can be written persistently with the function block MC\_WriteBoolParameterPersistent.

The persistent parameter to be written is stored in an initialization list. At system startup, the system initially starts with the originally configured values and overwrites these with the persistent data from the initialization list before the start of the task. The initialization list is cleared when a new system configuration is registered. The system then starts with the unchanged data from the new configuration.



In this case, "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

**Inputs**

```

VAR_INPUT
  Execute      : BOOL;
  ParameterNumber : MC_AxisParameter;
  Value       : BOOL;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ParameterNumber	MC_AxisParameter [▶_142]	Number of the parameter to be written.
Value	BOOL	Boolean value that is written.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

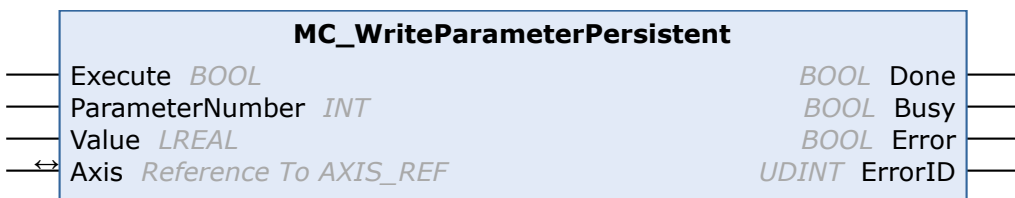
```
VAR_OUTPUT
  Done : BOOL;
  Busy : BOOL;
  Error : BOOL;
  ErrorID : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the parameters were written successfully.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done" or "Error" is set.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.2.12 MC\_WriteParameterPersistent**



Axis parameters can be written persistently with the function block MC\_WriteParameterPersistent.

The persistent parameter to be written is stored in an initialization list. At system startup, the system initially starts with the originally configured values and overwrites these with the persistent data from the initialization list before the start of the task. The initialization list is cleared when a new system configuration is registered. The system then starts with the unchanged data from the new configuration.



In this case, "axis" refers to the TwinCAT NC axis and its parameters, and not the drive.

 **Inputs**

```
VAR_INPUT
  Execute : BOOL;
  ParameterNumber : MC_AxisParameter;
  Value : LREAL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ParameterNumber	MC_AxisParameter [▶ 142]	Number of the parameter to be written.
Value	LREAL	LREAL value that is written.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Done : BOOL;
  Busy : BOOL;
  Error : BOOL;
  ErrorID : UDINT;
END_VAR
```

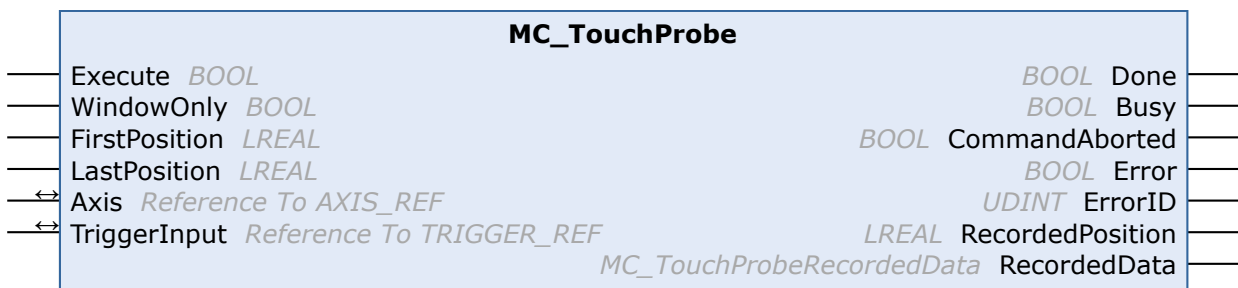
Name	Type	Description
Done	BOOL	TRUE if the parameters were written successfully.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done" or "Error" is set.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 5.3 Touch probe

### 5.3.1 MC\_TouchProbe



The function block MC\_TouchProbe records an axis position at the time of a digital signal (touch probe function). The position is usually not recorded directly in the PLC environment, but via an external hardware latch and is therefore highly accurate and independent of the cycle time. The function block controls this mechanism and determines the externally recorded position.



The function block was extended, compared to TwinCAT 2. It has the same functionality as the existing function block MC\_TouchProbe\_V2.

---

### Requirements

The prerequisite for the position detection is suitable encoder hardware that is able to latch the recorded position. Support is offered for:

- SERCOS drives  
In contrast to MC\_TouchProbe, the drive must be configured with an extended interface, in which the parameters S 0 0405 and S-0 0406 are included in the process image.
- EtherCAT SoE drives (E.g. AX5000)  
In contrast to MC\_TouchProbe, the drive must be configured with an extended interface, in which the parameters S 0 0405 and S-0 0406 are included in the process image.
- EtherCAT CoE drives  
The drive must be configured with the parameter 0x60B9 (touch probe status) in the process image.
- EL5101, KL5101  
Latching of the C track and the digital input is possible. This hardware can only record one signal or edge at a time. Continuous mode is not supported.

The digital trigger signal is wired into this hardware and, independently of the PLC cycle, triggers the recording of the current axis position.

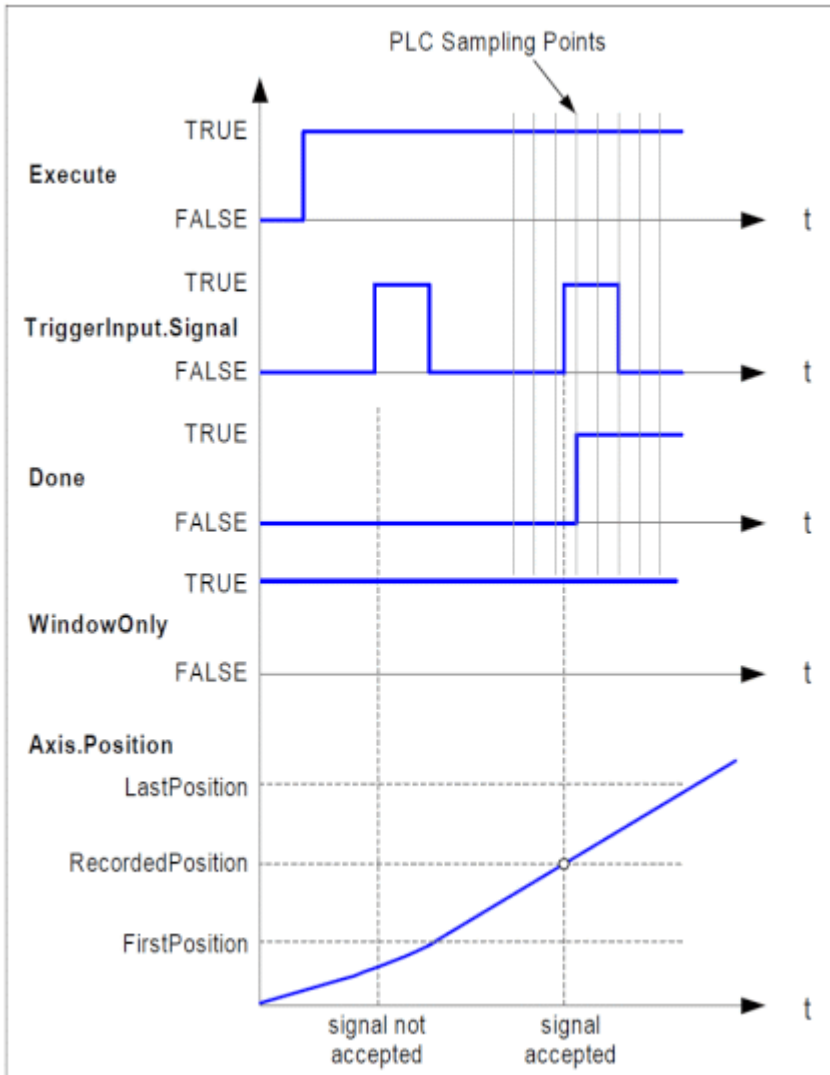
These end devices have to be configured to some extent so that a position detection is possible. Beckhoff EtherCAT drives can be configured with the System Manager. Note that the probe unit has to be configured with the "Extended NC Probe Unit" interface.



After a touch probe cycle has been initiated by a rising edge on the "Execute" input, this is only terminated if the outputs "Done", "Error" or "CommandAborted" become TRUE. If the process is to be interrupted at an intermediate point in time, the function block [MC\\_AbortTrigger \[▶ 39\]](#) with the same [TriggerInput \[▶ 154\]](#) data structure must be called up. Otherwise no new cycle can be initiated.

---

Signal curve



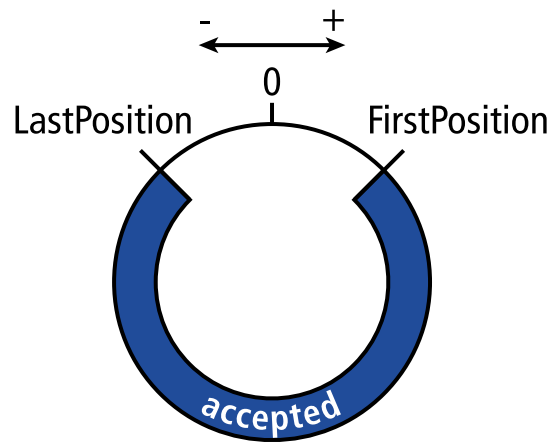
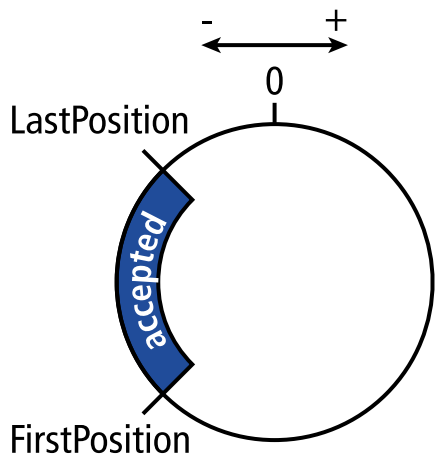
Inputs

```
VAR_INPUT
  Execute      : BOOL;
  WindowOnly   : BOOL;
  FirstPosition : LREAL;
  LastPosition  : LREAL;
END_VAR
```

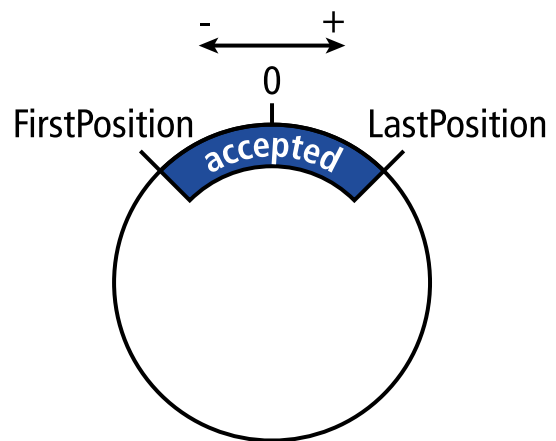
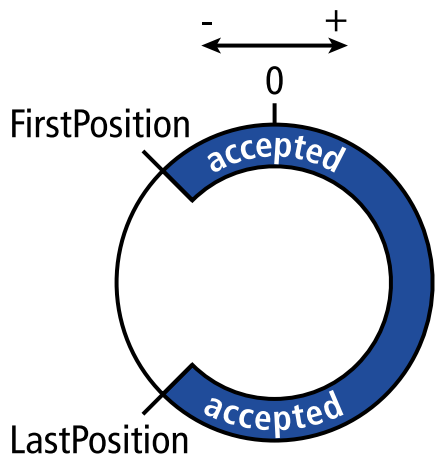
Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
WindowOnly	BOOL	If WindowOnly = TRUE, only one position within the window between "FirstPosition" and "LastPosition" is recorded. Positions outside the window are rejected and the external position latch is automatically newly activated. Only if the recorded position lies inside the window, "Done" becomes TRUE. The recording window can be interpreted in terms of absolute or modulo values. For this purpose, the "ModuloPositions" flag must be set accordingly in the <a href="#">TriggerInput [► 154]</a> data structure. In the case of absolute value positions there is exactly one window. With modulo positions, the window is repeated within the modulo cycle defined in the axis parameters (e.g. 0 to 360 degrees).

Name	Type	Description
FirstPosition	LREAL	Initial position of the recording window, if "WindowOnly" is TRUE. This position can be interpreted as an absolute or modulo value. For this purpose, the "ModuloPositions" flag must be set accordingly in the <a href="#">TriggerInput [▶ 154]</a> data structure.
LastPosition	LREAL	End position of the recording window, if "WindowOnly" is TRUE. This position can be interpreted as an absolute or modulo value. For this purpose, the "ModuloPositions" flag must be set accordingly in the <a href="#">TriggerInput [▶ 154]</a> data structure.

**A. FirstPosition < LastPosition**



**B. FirstPosition > LastPosition**



examples of windows, where trigger events are accepted (for modulo axes)

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**Outputs**

```

VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorId        : UDINT;
  RecordedPosition : LREAL;
  RecordedData    : MC_TouchProbeRecordedData;
END_VAR
    
```

Name	Type	Description
Done	BOOL	TRUE if an axis position was successfully detected. The position is sent to the output "RecordedPosition".
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state.
CommandAborted	BOOL	TRUE if the process is interrupted by an external event, e.g. by the call up of <code>MC_AbortTrigger</code> [▶ 39].
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
RecordedPosition	LREAL	Axis position recorded at the point in time of the trigger signal
RecordedData	<code>MC_TouchProbeRecordedData</code> [▶ 154]	Data structure with complementary information relating to the logged axis position at the time of the trigger signal

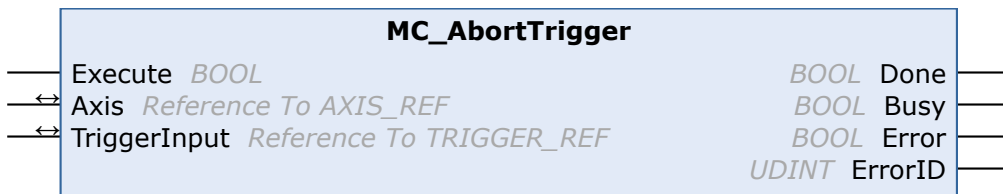
**Inputs/outputs**

```

VAR_IN_OUT
  Axis       : AXIS_REF;
  TriggerInput : TRIGGER_REF;
END_VAR
    
```

Name	Type	Description
Axis	<code>AXIS_REF</code> [▶ 118]	Axis data structure
TriggerInput	<code>TRIGGER_REF</code> [▶ 154]	Data structure for describing the trigger source. This data structure must be parameterized before the function block is called for the first time.

**5.3.2 MC\_AbortTrigger**



The function block `MC_AbortTrigger` is used to abort a touch probe cycle started by `MC_TouchProbe`. `MC_TouchProbe` initiates a touch probe cycle by activating a position latch in external encoder or drive hardware. The function block `MC_AbortTrigger` can be used to terminate the procedure before the trigger signal has activated the position latch. If the touch probe cycle has completed successfully, it is not necessary to call up this function block.

**Inputs**

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with the rising edge and the external position latch is disabled.

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
    TriggerInput : TRIGGER_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [▶ 118]	Axis data structure
TriggerInput	TRIGGER_REF [▶ 154]	Data structure for describing the trigger source. This data structure must be parameterized before the function block is called for the first time.

**Outputs**

```
VAR_OUTPUT
    Done : BOOL;
    Busy : BOOL;
    Error : BOOL;
    ErrorID : UDINT;
END_VAR
```

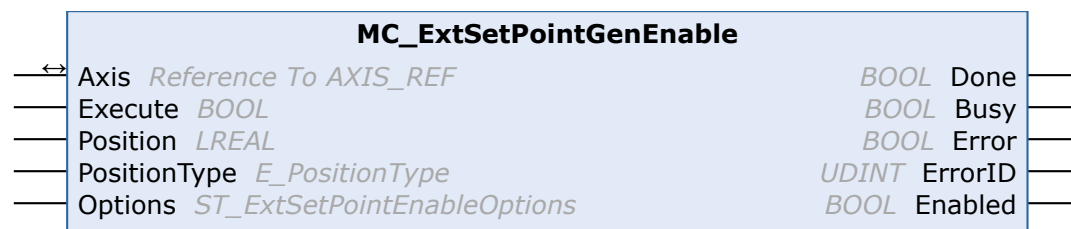
Name	Type	Description
Done	BOOL	TRUE as soon as the touch probe cycle has been successfully terminated.
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 5.4 External set value generator

### 5.4.1 MC\_ExtSetPointGenEnable





The external setpoint generator of an axis can be switched on with the MC\_ExtSetPointGenEnable function block. The axis then adopts the set value specifications from its cyclic axis interface (Axis.PlcToNc.ExtSetPos, ExtSetVelo, ExtSetAcc, and ExtSetDirection).

An external setpoint generator is usually a PLC function block that calculates cyclic set values for an axis and can therefore substitute the internal setpoint generator in an NC axis.

Additional information can be found under [MC\\_ExtSetPointGenDisable \[▶ 42\]](#) and [MC\\_ExtSetPointGenFeed \[▶ 43\]](#).

 **Inputs**

```
VAR_INPUT
  Execute      : BOOL;
  Position     : LREAL;
  PositionType : E_PositionType;
  Options      : ST_ExtSetPointEnableOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Position for target position monitoring. Setting this position does not mean that the axis moves to this position, only the external setpoint generator is responsible for this movement. Instead, setting this position activates the target position monitoring. The InTargetPosition flag becomes TRUE when this position is reached.
PositionType	<a href="#">E_PositionType [▶ 132]</a>	Position type: POSITION TYPE_ABSOLUTE or POSITION TYPE_RELATIVE
Options	ST_ExtSetPointEnableOptions	<b>UseTorqueOffset:</b> Must be set to TRUE so that the TorqueOffset is also transmitted to the drive controller cyclically when the MC_ExtSetPointGenFeedWithTorque is used.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Done      : BOOL;
  Busy      : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
  Enabled   : BOOL;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the command was executed successfully.
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

Name	Type	Description
Enabled	BOOL	Shows the current state of the external setpoint generator, regardless of the function execution.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.4.2 MC\_ExtSetPointGenDisable



The external setpoint generator of an axis can be switched off with the MC\_ExtSetPointGenDisable function block. The axis then no longer adopts the set value specifications from its cyclic axis interface (Axis.PlcToNc.ExtSetPos, ExtSetVelo, ExtSetAcc, and ExtSetDirection).

An external setpoint generator is usually a PLC function block that calculates cyclic set values for an axis and can therefore substitute the internal setpoint generator in an NC axis.

Additional information can be found under [MC\\_ExtSetPointGenEnable \[▶ 40\]](#) and [MC\\_ExtSetPointGenFeed \[▶ 43\]](#).

**Inputs**

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Outputs**

```
VAR_OUTPUT
    Done : BOOL;
    Busy : BOOL;
    Error : BOOL;
    ErrorID : UDINT;
    Enabled : BOOL;
END_VAR
```

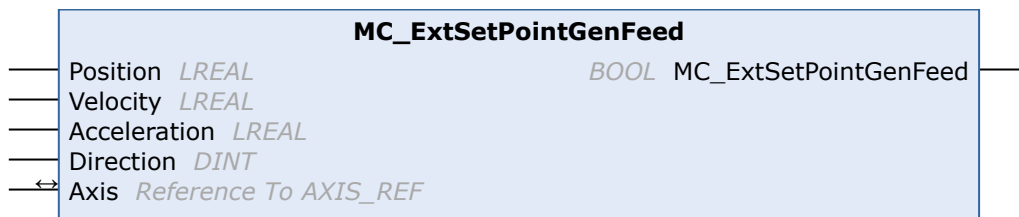
Name	Type	Description
Done	BOOL	TRUE if the command was executed successfully.

Name	Type	Description
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
Enabled	BOOL	Shows the current state of the external setpoint generator, regardless of the function execution.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.4.3 MC\_ExtSetPointGenFeed



The MC\_ExtSetPointGenFeed function is used to feed set values from an external setpoint generator to an axis. The function copies the data into the cyclic axis interface instantaneously (fExtSetPos, fExtSetVelo, fExtSetAcc and nExtSetDirection). The MC\_ExtSetPointGenFeed function result is not used and therefore always FALSE.

An external setpoint generator is usually a PLC function block that calculates cyclic set values for an axis and can therefore substitute the internal setpoint generator in an NC axis.

Additional information can be found under [MC\\_ExtSetPointGenEnable \[▶ 40\]](#) and [MC\\_ExtSetPointGenDisable \[▶ 42\]](#).

 **Inputs**

```
VAR_INPUT
    Position      : LREAL;
    Velocity      : LREAL;
    Acceleration  : LREAL;
    Direction     : DINT;
END_VAR
```

Name	Type	Description
Position	LREAL	Set position from an external setpoint generator
Velocity	LREAL	Set velocity from an external setpoint generator
Acceleration	LREAL	Set acceleration from an external setpoint generator
Direction	DINT	Set direction from an external setpoint generator. (-1 = negative direction, 0 = standstill, 1 = positive direction)

 **Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.4.4 MC\_ExtSetPointGenFeedWithTorque



The MC\_ExtSetPointGenFeedWithTorque function is used to feed set values from an external setpoint generator to an axis. Compared to the MC\_ExtSetPointGenFeed function, this function is extended by the transfer of a TorqueOffset. To ensure that the TorqueOffset is also transferred cyclically from the NC to the drive, this must be activated explicitly when the FB\_ExtSetPointGenEnable is called via Options.UseTorqueOffset. The function copies the data to the cyclic axis interface instantaneously (ExtSetPos, ExtSetVelo, ExtSetAcc, ExtSetDirection, and ExtTorque) of the axis. The MC\_ExtSetPointGenFeedWithTorque function result is unused and therefore always FALSE.

An external setpoint generator is usually a PLC function block that calculates cyclic set values for an axis and can therefore substitute the internal setpoint generator in an NC axis.

Additional information can be found under [MC\\_ExtSetPointGenEnable](#) [► 40] and [MC\\_ExtSetPointGenDisable](#) [► 42].

**Inputs**

```
VAR_INPUT
    Position      : LREAL;
    Velocity      : LREAL;
    Acceleration  : LREAL;
    TorqueOffset  : LREAL;
    Direction     : DINT;
END_VAR
```

Name	Type	Description
Position	LREAL	Set position from an external setpoint generator
Velocity	LREAL	Set velocity from an external setpoint generator
Acceleration	LREAL	Set acceleration from an external setpoint generator
TorqueOffset	LREAL	TorqueOffset from an external setpoint generator
Direction	DINT	Set direction from an external setpoint generator. (-1 = negative direction, 0 = standstill, 1 = positive direction)

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [► 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

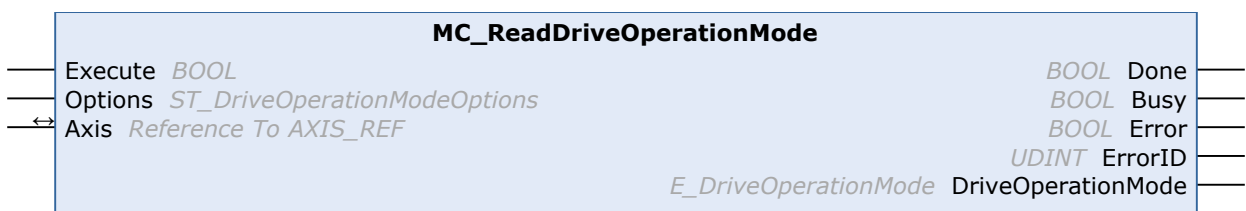
**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86 or x64)	Tc2_MC2 (from v3.3.68.0)

## 5.5 Special extensions

### 5.5.1 DriveOperationMode

#### 5.5.1.1 MC\_ReadDriveOperationMode



The function block MC\_ReadDriveOperationMode reads the currently active operation mode of the drive device linked to the NC axis.

**Inputs**

```
VAR_INPUT
    Execute          : BOOL;
    Options          : ST_DriveOperationModeOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Options	ST_DriveOperationModeOptions	Not implemented.

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [► 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Outputs**

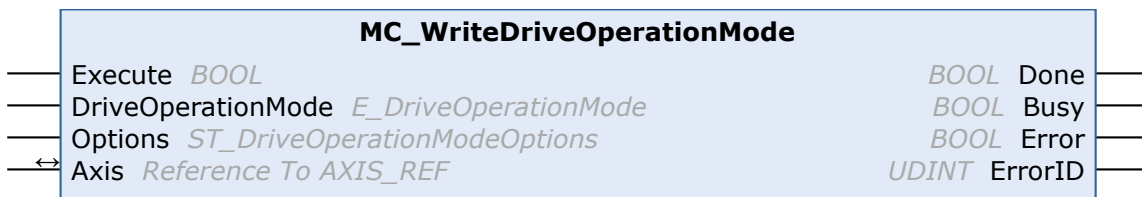
```
VAR_OUTPUT
    Done          : BOOL;
    Busy          : BOOL;
    Error         : BOOL;
    ErrorID       : UDINT;
    DriveOperationMode : E_DriveOperationMode;
END_VAR
```

Name	Type	Description
Done	BOOL	Becomes TRUE, if the command was completed successfully.
Busy	BOOL	The Busy output becomes TRUE when the command is started with Execute and remains TRUE as long as the command is processed. If Busy becomes FALSE again, the function block is ready for a new order. At the same time one of the outputs, Done, CommandAborted or Error, is set.
Error	BOOL	Becomes TRUE, as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
DriveOperationMode	E_DriveOperationMode	Currently active operation mode of the connected drive device ( <a href="#">E_DriveOperationMode</a>  > 127]).

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.5.1.2 MC\_WriteDriveOperationMode**



The function block MC\_WriteDriveOperationMode initiates the change to the parameterized operation mode for a drive device linked to the NC axis.

**Inputs**

```
VAR_INPUT
    Execute          : BOOL;
    DriveOperationMode : E_DriveOperationMode;
    Options          : ST_DriveOperationModeOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
DriveOperationMode	E_DriveOperationMode  > 127]	Operation mode which is to be activated for the connected drive device.
Options	ST_DriveOperationModeOptions	Not implemented.

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF  > 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**🔌 Outputs**

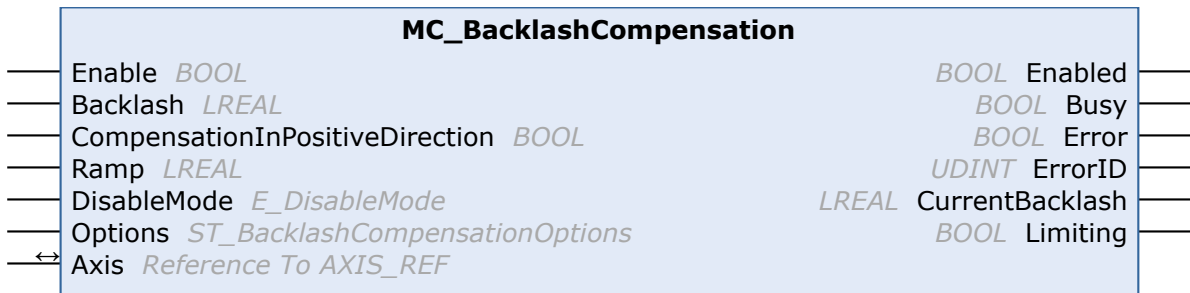
```
VAR_OUTPUT
  Done          : BOOL;
  Busy          : BOOL;
  Error         : BOOL;
  ErrorID      : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the command was executed without errors.
Busy	BOOL	The Busy output becomes TRUE when the command is started with Execute and remains TRUE as long as the command is processed. If Busy becomes FALSE again, the function block is ready for a new order.
Error	BOOL	Becomes TRUE, as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.5.2 MC\_BacklashCompensation**



The MC\_BacklashCompensation function block is used together with the axis parameter *Position Correction* to compensate backlash. It is possible to compensate negative as well as positive backlash.

**● Instructions for use and necessary settings in TwinCAT:**

- i** The functionality described here works with the parameter *Position Correction*. This parameter must be activated in TwinCAT XAE or via ADS (see also [TwinCAT 3 NC PTP Axes](#)). The Backlash Compensation function offered in the XAE is only still present for compatibility reasons. Do not use it on new projects.
- If the user also wants to use the function *Position Correction* in parallel to the function block *MC\_BacklashCompensation* (whose implementation is based on the function *Position Correction*) for other purposes, then this "double use" of the parameter can be solved by a workaround in the PLC by feeding in the sum of both requirements using the function block *MC\_PositionCorrectionLimiter*.

**🔌 Inputs**

```
VAR_INPUT
  Enable          : BOOL;
  Backlash       : LREAL;
  CompensationInPositiveDirection : BOOL;
  Ramp           : LREAL;
  DisableMode    : E_DisableMode;
  Options        : ST_BacklashCompensationOptions;
END_VAR
```

Name	Type	Description
Enable	BOOL	The command is executed as long as Enable is active. Backlash compensation is hereby enabled.
Backlash	LREAL	Compensation value with sign [mm]. If the default value is used, the internal NC backlash value is read via ADS and used in this function block.
CompensationInPositiveDirection	BOOL	Compensation takes place in the selected working direction. FALSE (default): Backlash compensation must be executed when moving in negative direction. TRUE: Backlash compensation must be executed when moving in positive direction.
Ramp	LREAL	Velocity limit for entered backlash compensation (constant velocity and linear position as subprofiles for backlash compensation [mm/s]).
DisableMode	E_DisableMode	Disable mode: (0)=HOLD, (1)=RESET
Options	ST_BacklashCompensationOptions	Optional parameters (not implemented)

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Enabled          : BOOL;
  Busy             : BOOL;
  Error            : BOOL;
  ErrorId          : UDINT;
  CurrentBacklash : LREAL;
  Limiting         : BOOL;
END_VAR
```

Name	Type	Description
Enabled	BOOL	This output becomes TRUE, if the backlash compensation has been enabled without error.
Busy	BOOL	This output becomes TRUE when the command is started with Enable and remains so as long as the function block executes the command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).
CurrentBacklash	LREAL	Current compensation value [mm].
Limiting	BOOL	This output becomes TRUE if the function block currently limits the backlash compensation.

**TYPE E\_DisableMode:**

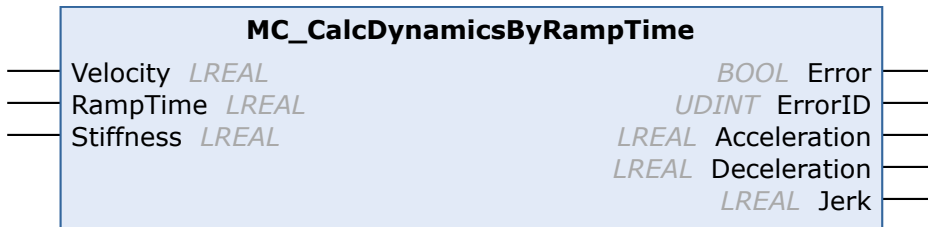
```
(
  DisableModeHold      :=0, (* hold on the last output value (default) *)
  DisableModeReset     :=1 (* reset the output value to zero (jump to zero) *);
END_TYPE
```



Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.5.3 MC\_CalcDynamicsByRampTime



The MC\_CalcDynamicsByRampTime function block is used to calculate the acceleration, deceleration and jerk dynamic parameters, which are required to achieve a specified velocity in a defined time.

The function block assumes that the specified velocity can actually be achieved. If the calculated dynamic parameters are used on a short travel distance, the achieved velocity and the associated acceleration time may be smaller. The function block calculates the acceleration ramp and outputs an identical value as deceleration.

The calculated acceleration, deceleration and jerk dynamic parameters can be used with all MC\_Move... function blocks, as well as MC\_Halt and MC\_Stop.

**Inputs**

```
VAR_INPUT
    Velocity      : LREAL;
    RampTime     : LREAL;
    Stiffness    : LREAL;
END_VAR
```

Name	Type	Description
Velocity	LREAL	Velocity to be achieved
RampTime	LREAL	Time required to achieve the velocity.
Stiffness	LREAL	Stiffness of the acceleration profile [0..1]. A large value means higher jerk in the profile.

**Outputs**

```
VAR_OUTPUT
    Error        : BOOL;
    ErrorID     : UDINT;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk        : LREAL;
END_VAR
```

Name	Type	Description
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
Acceleration	LREAL	Calculated acceleration
Deceleration	LREAL	Calculated deceleration (Deceleration=Acceleration)
Jerk	LREAL	Calculated jerk

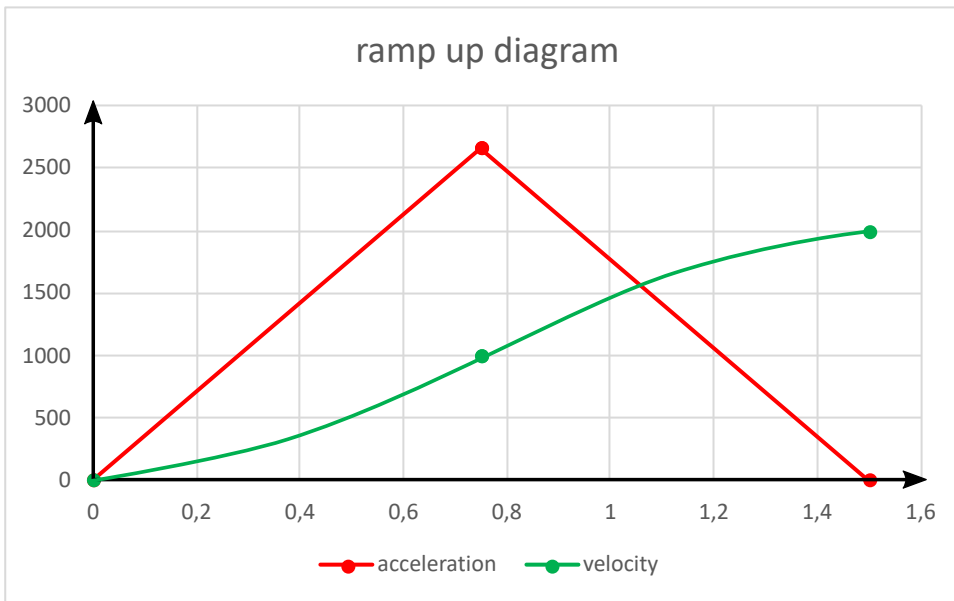
**Examples**

t<sub>1</sub>: internal acceleration ramp time / jerk time

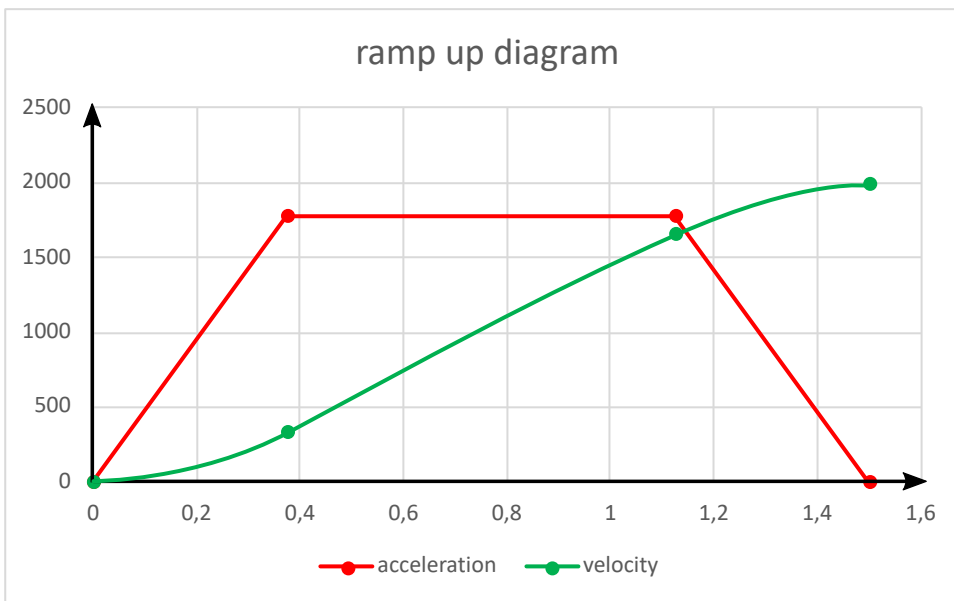
t<sub>2</sub>: internal constant acceleration time

$t$  : acceleration ramp time,  $t = 2 t_1 + t_2$

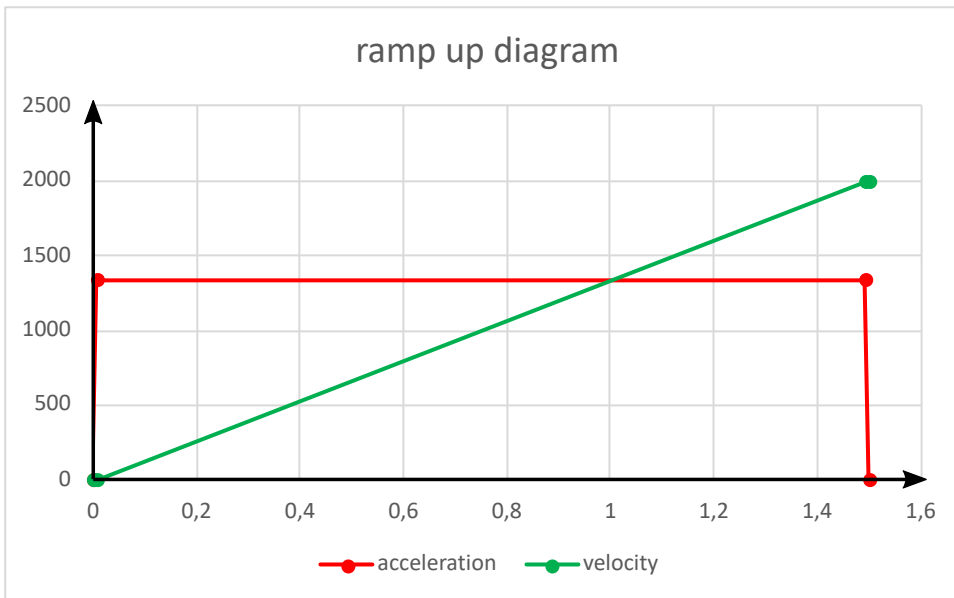
$c_s = 0\%$  →  $t_1 = \frac{1}{2} t$   $t_2 = 0$



$c_s = 50\%$  →  $t_1 = t_2 = \frac{1}{3} t$



$c_s = 99\%$  →  $t_1 = 0$   $t_2 = t$



**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.42	PC or CX (x86 or x64)	Tc2_MC2

**5.5.4 MC\_OverrideFilter**



The function block MC\_OverrideFilter can be used to convert an unfiltered override value consisting of digits (e.g. a voltage value of an analog input terminal) into a filtered override value that matches the cyclic axis interface (PlcToNc) (DWORD in the range 0...1000000). This filtered override is also available in percent (LREAL in the range 0...100%).

The raw input value is limited to a validity range by "LowerOverrideThreshold" and "UpperOverrideThreshold", and implemented as parameterizable steps (resolution) ("OverrideSteps"). After each override change at the output of the FB, the system internally waits for a minimum recovery time ("OverrideRecoveryTime") before a new override value can be applied. The only exceptions are the override values 0% and 100%, which are always implemented without delay for safety reasons.

**i** Due to the gradation of the output override value ("OverrideValueFiltered"), the filtered override can become zero with very small override input values ("OverrideValueRaw"). A zero override leads to standstill of the axis. If total standstill is undesired, "OverrideValueRaw" should not fall below the smallest level.

**Inputs**

```

VAR_INPUT
  OverrideValueRaw      : DINT;
  LowerOverrideThreshold : DINT := 0; (* 0...32767 digits *)
  UpperOverrideThreshold : DINT := 32767; (* 0...32767 digits *)
  OverrideSteps         : UDINT := 200; (* 200 steps => 0.5 percent *)
  OverrideRecoveryTime  : TIME := T#150ms; (* 150 ms *)
END_VAR
    
```

Name	Type	Description
OverrideValueRaw	DINT	Raw, unfiltered override value (e.g. a voltage value of an analog input terminal).
LowerOverrideThreshold	DINT	Lower threshold that limits the raw override value.
UpperOverrideThreshold	DINT	Upper threshold that limits the raw override value.
OverrideSteps	UDINT	The specified steps (override resolution).
OverrideRecoveryTime	TIME	Minimum recovery time, after which a new filtered override value is applied to the output. However, the override values 0% and 100% are implemented without delay.

**Outputs**

```

VAR_OUTPUT
  OverrideValueFiltered : DWORD; (* 0...1000000 counts *)
  OverridePercentFiltered : LREAL; (* 0...100 % *)
  Error : BOOL;
  ErrorId : UDINT;
END_VAR
    
```

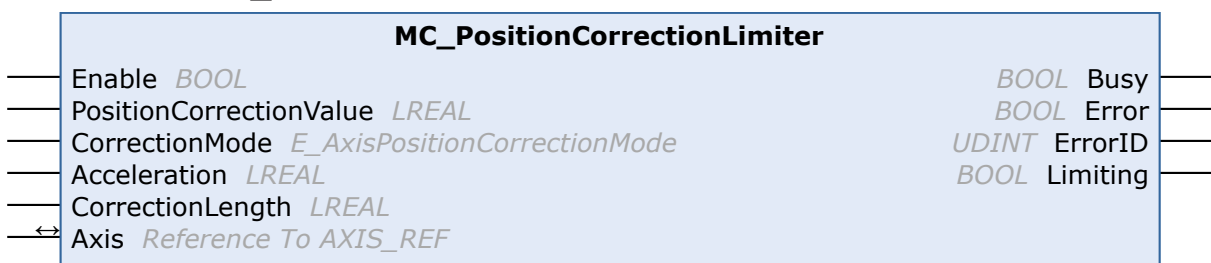
Name	Type	Description
OverrideValueFiltered	DWORD	The filtered override value in digits (the data type matches the override in the cyclic axis interface 0..1000000).
OverridePercentFiltered	LREAL	Filtered override value in percent (0..100%).
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

Possible error number	Possible causes
MC_ERROR_PARAMETER_NOT_CORRECT	<ul style="list-style-type: none"> <li>OverrideSteps &lt;= 1</li> <li>LowerOverrideThreshold &gt;= UpperOverrideThreshold</li> </ul>

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.5.5 MC\_PositionCorrectionLimiter**



The function block MC\_PositionCorrectionLimiter writes a correction value (PositionCorrectionValue) at the actual position of an axis. Depending on the correction mode the data are fed either directly or filtered to the axis.



To use this function block successfully, the "Position Correction" parameter must be activated in the System Manager. The function block should only be executed on enabled axes.

 **Inputs**

```
VAR_INPUT
  Enable           :   BOOL;
  PositionCorrectionValue : LREAL;
  CorrectionMode   :   E_AxisPositionCorrectionMode;
  Acceleration     :   LREAL;
  CorrectionLength :   LREAL;
END_VAR
```

Name	Type	Description
Enable	BOOL	Activates continuous writing of the correction value "PositionCorrectionValue". It must be TRUE as long as new correction values are to be accepted.
PositionCorrectionValue	LREAL	Correction value to be added to the actual value of the axis.
CorrectionMode	E_AxisPositionCorrectionMode <a href="#">▶ 137</a>	Depending on this mode, the correction value "PositionCorrectionValue" is written either directly or filtered.
Acceleration	LREAL	Depending on the "CorrectionMode" the maximum acceleration to reach the new correction value is specified here. In the case of PositionCorrectionMode_Fast this value has a direct effect on the position delta by PLC-tick.  Maximum permissible correction value Position delta = acceleration * (PLC cycle time) <sup>2</sup> .  If acceleration is parameterized as 0.0, the position correction is not limited.
CorrectionLength	LREAL	Becomes active when the "CorrectionMode" matches <a href="#">PositionCorrectionMode_FullLength ▶ 137</a> . A change in the "PositionCorrectionValue" is distributed over this correction length.

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF ▶ 118</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

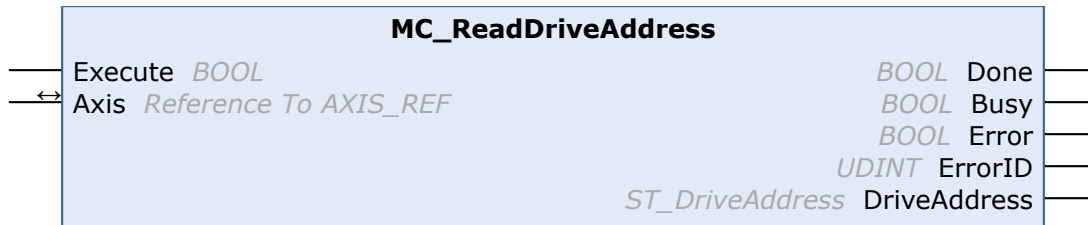
```
VAR_OUTPUT
  Busy       :   BOOL;
  Error      :   BOOL;
  ErrorID    :   UDINT;
  Limiting   :   BOOL;
END_VAR
```

Name	Type	Description
Busy	BOOL	TRUE as soon as the function block is active. FALSE when it returns to its original state.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
Limiting	BOOL	TRUE if the required correction value "PositionCorrectionValue" is not yet fully accepted.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.5.6 MC\_ReadDriveAddress**



The function block MC\_ReadDriveAddress reads the ADS access data for a drive device connected to the axis. This information is required for accessing the device, e.g. for special parameterization.

**Inputs**

```
VAR_INPUT
    Execute : BOOL; (* B *)
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.

See also: [General rules for MC function blocks \[▶ 14\]](#)

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [▶ 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Outputs**

```
VAR_OUTPUT
    Done      : BOOL; (* B *)
    Busy      : BOOL; (* E *)
    Error      : BOOL; (* B *)
    ErrorID    : DWORD; (* B *)
    DriveAddress : ST_DriveAddress; (* B *)
END_VAR
```

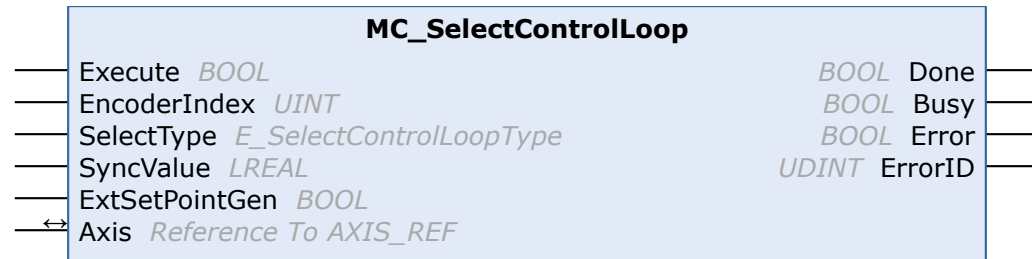
Name	Type	Description
Done	BOOL	TRUE if the command was executed without errors.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	DWORD	If the error output is set, this parameter supplies the error number.
DriveAddress	<a href="#">ST_DriveAddress [▶ 148]</a>	ADS access data of a drive device connected to the axis.

See also: [General rules for MC function blocks \[▶ 14\]](#)

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.5.7 MC\_SelectControlLoop**



The function block MC\_SelectControlLoop is used to switch between control loops of an axis. The prerequisite is that two or more control loops have been created below the axis in the system configuration.

See example "[Control loop switching in an AX5000 with two existing encoders \[► 158\]](#)".

**Inputs**

```
VAR_INPUT
Execute : BOOL;
EncoderIndex : UINT;
SelectType : E_SelectControlLoopType;
SyncValue : LREAL;
ExtSetPointGen : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
EncoderIndex	UINT	The sequential number [0..9] of the axis control loop to be activated.
SelectType	<a href="#">E_SelectControlLoopType [► 138]</a>	Defines how the control loop switching is performed. Currently only applicable with the SelectControlLoopType_Standard value.
SyncValue	LREAL	Currently not applicable.
ExtSetPointGen	BOOL	Currently not applicable.

**Inputs/outputs**

```
VAR_IN_OUT
Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [► 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Outputs**

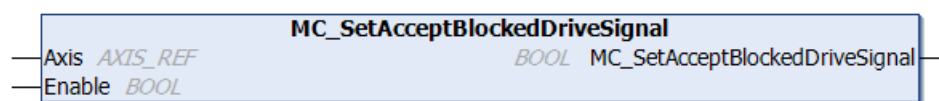
```
VAR_OUTPUT
Done : BOOL;
Busy : BOOL;
Error : BOOL;
ErrorID : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the command was executed without errors.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86)	Tc2_MC2

### 5.5.8 MC\_SetAcceptBlockedDriveSignal



There are situations in which a drive no longer follows the NC setpoints, e.g. if an axis reaches a limit switch. The NC interprets such a situation as an error, and the drive is stopped. In some cases the user may want to provoke such a situation deliberately, e.g. in order to move to a limit switch for a reference run.

The function MC\_SetAcceptBlockedDriveSignal can be used to temporarily prevent the NC axis generating an error in situations where the drive no longer follows the NC setpoints.

- See also bit 8 of the ControlDWord in [AXIS\\_REF \[► 118\]](#).
- A SERCOS/SoE drive reports "Drive follows the command values" via status bit 3 of drive status word S-0-0135.
- A CANopen/CoE drive reports "Drive follows the command values" via status bit 12 of object 6041h.

**FUNCTION MC\_SetAcceptBlockedDriveSignal: BOOL**

**Inputs**

```
VAR_INPUT
    Enable : BOOL;
END_VAR
```

Name	Type	Description
Enable	BOOL	NC controller enable for the axis

**Inputs/outputs**

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

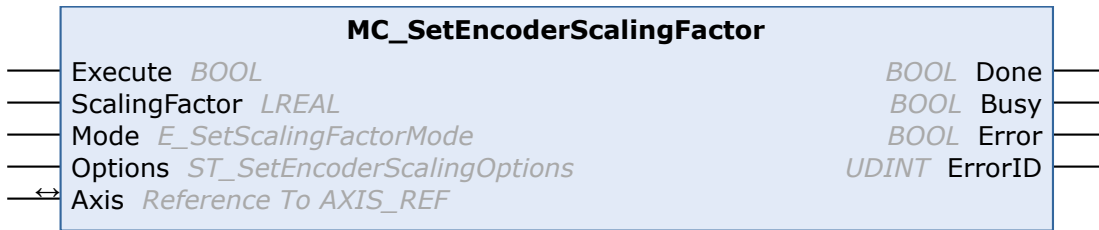
Name	Type	Description
Axis	<a href="#">AXIS_REF [► 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2



### 5.5.9 MC\_SetEncoderScalingFactor



The function block MC\_SetEncoderScalingFactor changes the scaling factor of the active encoder of an axis, either at standstill or in motion.

The change can be absolute or relative. This mode is only suitable at standstill, since in absolute mode the change in scaling factor leads to a position discontinuity. In relative mode an internal position offset is adapted at the same time such that no discontinuity occurs. Note that an intervention during the motion results in a change in the actual axis velocity, while the real velocity remains constant. Therefore only small changes can be implemented during the motion.

#### Inputs

```
VAR_INPUT
    Execute      : BOOL;
    ScalingFactor : LREAL;
    Mode         : E_SetScalingFactorMode;
    Options      : ST_SetEncoderScalingOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ScalingFactor	LREAL	Scaling factor of the active encoder of an axis. The scaling factor is specified in physical positioning units [u] divided by the number of encoder increments.
Mode	<a href="#">E_SetScalingFactorMode</a> ▶ <a href="#">141</a>	The scaling factor can be set in absolute or relative mode (ENCODERSCALINGMODE_ABSOLUTE, ENCODERSCALINGMODE_RELATIVE). In absolute mode, counting starts at the zero point of the axis coordinate system, therefore a position jump results when the scaling factor is changed. In relative mode the actual position of the axis does not change. This mode is therefore also suitable for changes during motion.
Options	<a href="#">ST_SetEncoderScalingOptions</a> ▶ <a href="#">148</a>	Data structure containing additional, rarely used parameters. The input can normally remain open. <ul style="list-style-type: none"> <li><b>SelectEncoderIndex:</b> can be set if an axis with several encoders is used and the position of a certain encoder is to be set (Options.EncoderIndex).</li> <li><b>EncoderIndex:</b> indicates the encoder (0..n) if "SelectEncoderIndex" is TRUE.</li> </ul>

See also: [General rules for MC function blocks](#) ▶ [14](#)

#### Inputs/outputs

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> ▶ <a href="#">118</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Done      : BOOL;
  Busy      : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
END_VAR
```

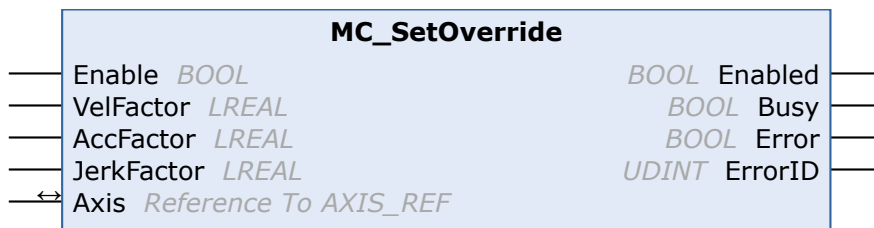
Name	Type	Description
Done	BOOL	TRUE if the position was set successfully.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done" or "Error" is set.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[▶ 14\]](#)

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 5.5.10 MC\_SetOverride



The override for an axis can be specified with the function block MC\_SetOverride.

 **Inputs**

```
VAR_INPUT
  Enable      : BOOL; (* B *)
  VelFactor   : LREAL (* B *) := 1.0; (* 1.0 = 100% *)
  AccFactor   : LREAL (* E *) := 1.0; (* 1.0 = 100% *) (* not supported *)
  JerkFactor  : LREAL (* E *) := 1.0; (* 1.0 = 100% *) (* not supported *)
END_VAR
```

Name	Type	Description
Enable	BOOL	TRUE as long as the command is executed.
VelFactor	LREAL	Velocity override factor
AccFactor	LREAL	Not supported
JerkFactor	LREAL	Not supported

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**🔌 Outputs**

```
VAR_OUTPUT
  Enabled : BOOL;
  Busy    : BOOL;
  Error   : BOOL;
  ErrorID : UDINT;
END_VAR
```

Name	Type	Description
Enabled	BOOL	The parameterized override is set.
Busy	BOOL	TRUE as soon as the command is started with Enable and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**5.5.11 MC\_WriteNcIoOutput**

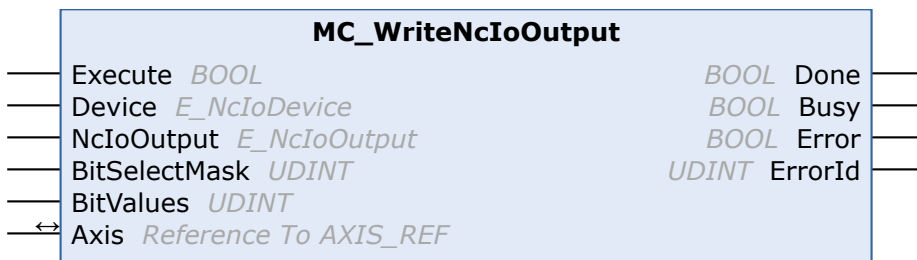


Fig. 1:

The MC\_WriteNcIoOutput function block can be used to write to unused IO outputs of the axis.

**🔌 Inputs**

```
VAR_INPUT
  Execute      : BOOL;
  Device       : E_NcIoDevice := E_NcIoDevice.NcIoDeviceDrive;
  NcIoOutput   : E_NcIoOutput := E_NcIoOutput.NcIoOutputnCtrl1;
  BitSelectMask : UDINT := 16#0;
  BitValues    : UDINT;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Device	<a href="#">E_NcIoDevice</a> [► 136]	Selection of the Nc axis component whose IO object is to be modified (encoder or drive)
NcIoOutput	<a href="#">E_NcIoOutput</a> [► 136]	Selection of the subobject whose value is to be modified (e.g. nCtrl1)
BitSelectMask	UDINT	Mask for selecting which bits are to be modified

Name	Type	Description
BitValues	UDINT	Values of the corresponding bits to be modified

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

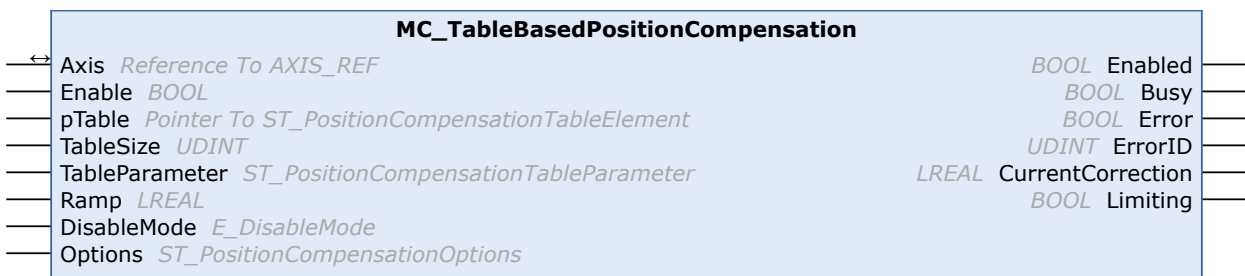
```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the command was executed without errors.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86)	Tc2_MC2

**5.5.12 MC\_TableBasedPositionCompensation**



The *MC\_TableBasedPositionCompensation* function block is used to correct the axis position with a correction factor depending on the current axis position. The correction values must be stored in an equidistant, monotonically increasing table.

 **Inputs**

```
VAR_INPUT
  Enable           : BOOL;
  pTable           : POINTER To ST_PositionCompensationTableElement;
  TableSize        : UDINT;
  TableParameter   : ST_PositionCompensationTableParameter;
  Ramp             : LREAL;
END_VAR
```

```

DisableMode      : E_DisableMode;
Options          : ST_PositionCompensationOptions;
END_VAR

```

Name	Type	Description
Enable	BOOL	The command is executed as long as Enable is active.
pTable	POINTER To <a href="#">ST_PositionCompensationTableElement</a> [ <a href="#">▶ 137</a> ]	Pointer to the compensation table, which is an array of the type <a href="#">St_PositionCompensationTableElement</a> .
TableSize	UDINT	Size of the compensation table
TableParameter	<a href="#">ST_PositionCompensationTableParameter</a> [ <a href="#">▶ 137</a> ]	Data structure with additional parameters for the compensation table.
Ramp	LREAL	Velocity limit for the entered compensation (constant velocity and linear position as subprofiles for table compensation [mm/s]).
DisableMode	E_DisableMode	Disable mode: DisableModeHold = last correction is retained DisableModeReset = last correction is set to 0
Options	ST_PositionCompensationOptions	Optional parameters (not implemented)

 **Inputs/outputs**

```

VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR

```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```

VAR_OUTPUT
  Enabled      : BOOL;
  Busy         : BOOL;
  Error        : BOOL;
  ErrorID      : UDINT;
  CurrentCorrection : LREAL;
  Limiting     : BOOL;
END_VAR

```

Name	Type	Description
Enabled	BOOL	This output becomes TRUE when the table compensation was enabled without errors.
Busy	BOOL	This output becomes TRUE when the command is started with Enable and remains so as long as the function block executes the command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorID	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).
CurrentCorrection	LREAL	Current compensation value, in the unit of the axis.
Limiting	BOOL	This output is TRUE if the correction value associated with the position has not yet been fully applied.

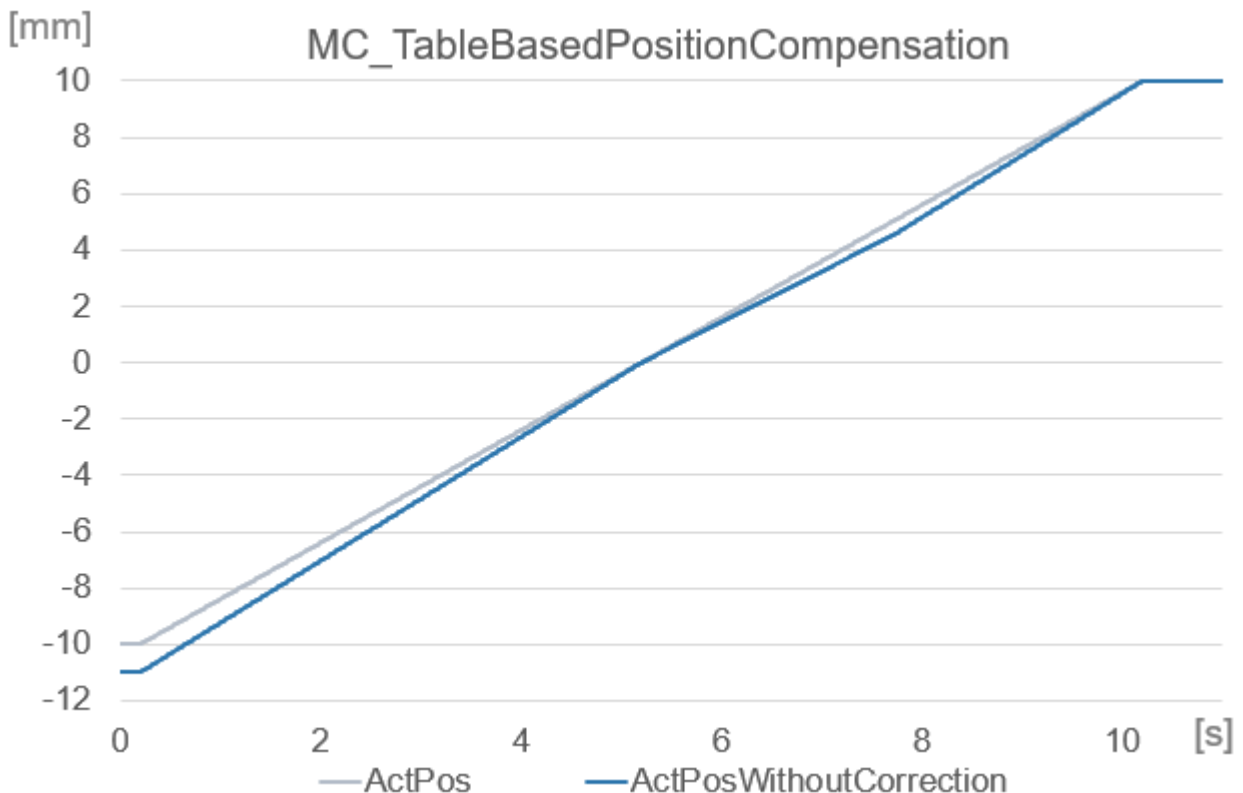
**Further information**

The following is an example of a position table and the corresponding table parameters:

```

VAR_INPUT
...
  stParameter : ST_PositionCompensationTableParameter
               := (MinPosition := -10.0, MaxPosition := 10.0,
                  NoOfTableElements := 21, Direction := WorkDirectionBoth);
  stPosTable : ARRAY[0..20] OF ST_PositionCompensationTableElement
              := [ ( Position := -10.0, Compensation := 1.0 ),
                  ( Position := -9.0, Compensation := 0.9 ),
                  ( Position := -8.0, Compensation := 0.8 ),
                  ( Position := -7.0, Compensation := 0.7 ),
                  ( Position := -6.0, Compensation := 0.6 ),
                  ( Position := -5.0, Compensation := 0.5 ),
                  ( Position := -4.0, Compensation := 0.4 ),
                  ( Position := -3.0, Compensation := 0.3 ),
                  ( Position := -2.0, Compensation := 0.2 ),
                  ( Position := -1.0, Compensation := 0.1 ),
                  ( Position := 0.0, Compensation := 0.0 ),
                  ( Position := 1.0, Compensation := 0.1 ),
                  ( Position := 2.0, Compensation := 0.2 ),
                  ( Position := 3.0, Compensation := 0.3 ),
                  ( Position := 4.0, Compensation := 0.4 ),
                  ( Position := 5.0, Compensation := 0.5 ),
                  ( Position := 6.0, Compensation := 0.4 ),
                  ( Position := 7.0, Compensation := 0.3 ),
                  ( Position := 8.0, Compensation := 0.2 ),
                  ( Position := 9.0, Compensation := 0.1 ),
                  ( Position := 10.0, Compensation := 0.0 )
                ];
...
END_VAR
    
```

As shown graphically, the table results in the following correction behavior:



**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86/x64)	Tc2_MC2

### 5.5.13 MC\_ConvertIncPosToPos



The function block MC\_ConvertIncPosToPos converts incremental encoder positions into a corresponding NC axis position.

The incremental position is scaled with the scaling factor and the currently valid position offset is applied. The determined NC axis position is in the same coordinate system as the actual position of the axis.

Please note: For the incremental position, bear in mind that the incremental encoder position regularly overflows. The conversion does not take past or future overflows into account and the result is based on current encoder feedback.

#### Inputs

```
VAR_INPUT
  Axis      : AXIS_REF;
  IncPos    : UDINT;
  Options   : ST_NcPositionConversionOptions;
END_VAR
```

Name	Type	Description
Axis	<u>AXIS_REF</u>	Axis data structure that unambiguously addresses an axis in the system. It contains the current axis status, including position, velocity or error state, among other parameters.
InPos	UDINT	Incremental encoder position
Options	<u>ST_NcPositionConversionOptions</u> [ <a href="#">▶_149</a> ]	Optional parameters

#### Outputs

```
VAR_OUTPUT
  Valid      : BOOL;
  Position   : LREAL;
  Error      : BOOL;
  ErrorID    : UDINT;
END_VAR
```

Name	Type	Description
Valid	BOOL	Supplies the NC axis position corresponding to dcTime. This is an NC axis position that has been scaled and provided with an offset, with, for instance, physical units: degrees or millimeters.
Position	LREAL	Position of the NC axis
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	Returns the error number if an error occurs, e.g. error 0x4012 (axis ID is not allowed, or axis does not exist within the system).

#### Prerequisites

Library	Version
Tc2_MC2	3.3.59

### 5.5.14 MC\_ConvertPosToIncPos



The function block MC\_ConvertPosToIncPos converts an NC axis position into a corresponding incremental encoder position.

The incremental position is scaled with the scaling factor and the currently valid position offset is applied. The determined NC axis position is in the same coordinate system as the actual position of the axis.

Please note: For the incremental position, bear in mind that the incremental encoder position regularly overflows. If the NC axis position is too far away from the actual position of the NC axis, the function block returns an error.

#### Inputs

```
VAR_INPUT
    Axis      : AXIS_REF;
    Position  : LREAL;
    Options   : ST_NcPositionConversionOptions;
END_VAR
```

Name	Type	Description
Axis	<u>AXIS_REF</u>	Axis data structure that unambiguously addresses an axis in the system. It contains the current axis status, including position, velocity or error state, among other parameters.
Position	LREAL	Position of the NC axis
Options	<u>ST_NcPositionConversionOptions</u> [ <a href="#">▶_149</a> ]	Optional parameters

#### Outputs

```
VAR_OUTPUT
    Valid      : BOOL;
    IncPos     : UDINT;
    Error      : BOOL;
    ErrorID    : UDINT;
END_VAR
```

Name	Type	Description
Valid	BOOL	Supplies the NC axis position corresponding to dcTime. This is an NC axis position that has been scaled and provided with an offset, with, for instance, physical units: degrees or millimeters.
InPos	UDINT	Incremental encoder position
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	Returns the error number if an error occurs, e.g. error 0x4012 (axis ID is not allowed, or axis does not exist within the system).

#### Prerequisites

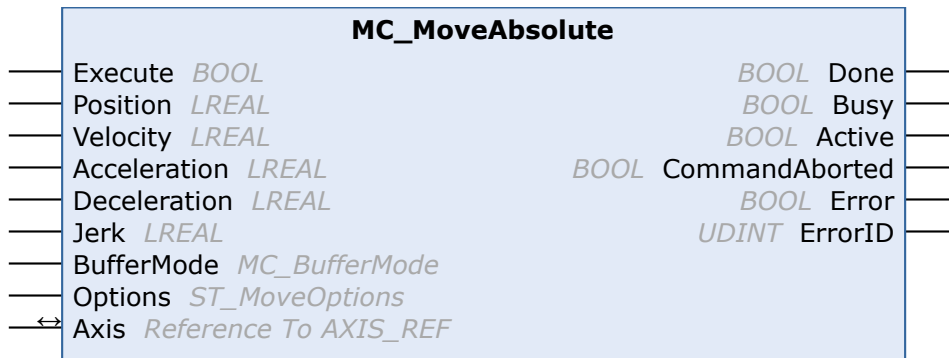
Library	Version
Tc2_MC2	3.3.59



## 6 Motion function blocks

### 6.1 Point to point motion

#### 6.1.1 MC\_MoveAbsolute



The function block MC\_MoveAbsolute starts the positioning to an absolute target position and monitors the axis movement over the entire travel path. The "Done" output is set once the target position has been reached. Otherwise, the output "CommandAborted" or, in case of an error, the output "Error" is set.

MC\_MoveAbsolute is predominantly used for linear axis systems. For modulo axes the position is not interpreted as a modulo position, but as an absolute position in continuous absolute coordinate system. Alternatively, the function block [MC\\_MoveModulo \[► 71\]](#) can be used for modulo positioning.

Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as MC\_MoveAbsolute then automatically leads to decoupling of the axis, after which the command is executed. In this case the only available BufferMode is "Aborting".

#### Inputs

```
VAR_INPUT
  Execute      : BOOL;
  Position     : LREAL;
  Velocity     : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  BufferMode   : MC_BufferMode;
  Options     : ST_MoveOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Absolute target position to be used for positioning.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.
Deceleration	LREAL	Deceleration (≥0) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk (≥0) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

Name	Type	Description
BufferMode	MC_BufferMode [► 132]	Is evaluated if the axis is already executing another command. MC_MoveAbsolute becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. If the command is applied to a coupled slave axis, the only available BufferMode is "Aborting". A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.
Options	ST_MoveOptions [► 135]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks \[► 14\]](#)

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done      : BOOL;
  Busy      : BOOL;
  Active    : BOOL;
  CommandAborted : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE when the target position has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[► 14\]](#)

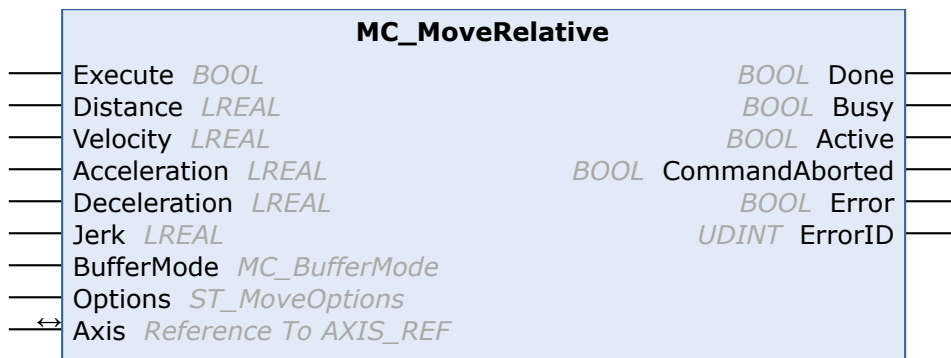


- "Target Position Monitoring" is activated (standard case): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- "Position Range Monitoring" is activated ("Target Position Monitoring" is not active): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- Neither "Target Position Monitoring" nor "Position Range Monitoring" is activated: the Done output is set immediately when the logical positioning end is reached (the NC setpoint generation is ended and the HasJob bit is FALSE).

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.1.2 MC\_MoveRelative**



The function block MC\_MoveRelative starts a relative positioning procedure based on the current set position and monitors the axis movement over the whole travel path. The "Done" output is set once the target position has been reached. Otherwise, the output "CommandAborted" or, in case of an error, the output "Error" is set.

Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as MC\_MoveRelative then automatically leads to decoupling of the axis, after which the command is executed. In this case the only available BufferMode is "Aborting".

**Inputs**

```

VAR_INPUT
    Execute      : BOOL;
    Distance     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk        : LREAL;
    BufferMode   : MC_BufferMode;
    Options     : ST_MoveOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Distance	LREAL	Relative distance to be used for positioning.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Name	Type	Description
Deceleration	LREAL	Deceleration ( $\geq 0$ ) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk ( $\geq 0$ ) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.
BufferMode	<a href="#">MC_BufferMode [► 132]</a>	Is evaluated if the axis is already executing another command. MC_MoveRelative becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. If the command is applied to a coupled slave axis, the only available BufferMode is "Aborting". A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.
Options	<a href="#">ST_MoveOptions [► 135]</a>	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks \[► 14\]](#)

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF [► 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE when the target position has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[► 14\]](#)

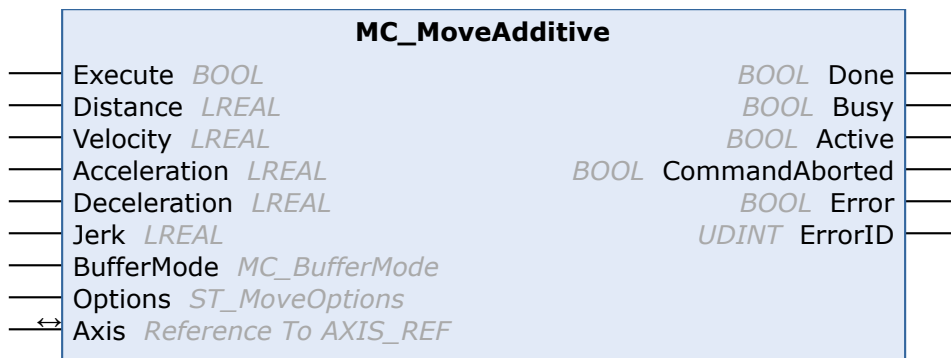


- "Target Position Monitoring" is activated (standard case): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- "Position Range Monitoring" is activated ("Target Position Monitoring" is not active): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- Neither "Target Position Monitoring" nor "Position Range Monitoring" is activated: the Done output is set immediately when the logical positioning end is reached (the NC setpoint generation is ended and the HasJob bit is FALSE).

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.1.3 MC\_MoveAdditive**



The function block MC\_MoveAdditive starts relative positioning procedure based on the last target position instruction, irrespective of whether this was reached. The "Done" output is set once the target position has been reached. Otherwise, the output "CommandAborted" or, in case of an error, the output "Error" is set.

If no last target position is known or the axis is moving continuously, the movement is executed based on the current set position for the axis.



MC\_MoveAdditive is not implemented for high/low speed axes.

**Inputs**

```

VAR_INPUT
  Execute      : BOOL;
  Distance     : LREAL;
  Velocity     : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  BufferMode   : MC_BufferMode;
  Options     : ST_MoveOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Distance	LREAL	Relative distance to be used for positioning.
Velocity	LREAL	Maximum travel velocity (>0).

Name	Type	Description
Acceleration	LREAL	Acceleration ( $\geq 0$ ) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.
Deceleration	LREAL	Deceleration ( $\geq 0$ ) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk ( $\geq 0$ ) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.
BufferMode	<a href="#">MC_BufferMode</a> [ <a href="#">▶ 132</a> ]	Is evaluated if the axis is already executing another command. MC_MoveAdditive becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.
Options	<a href="#">ST_MoveOptions</a> [ <a href="#">▶ 135</a> ]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks](#) [[▶ 14](#)]

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE when the target position has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[► 14\]](#)

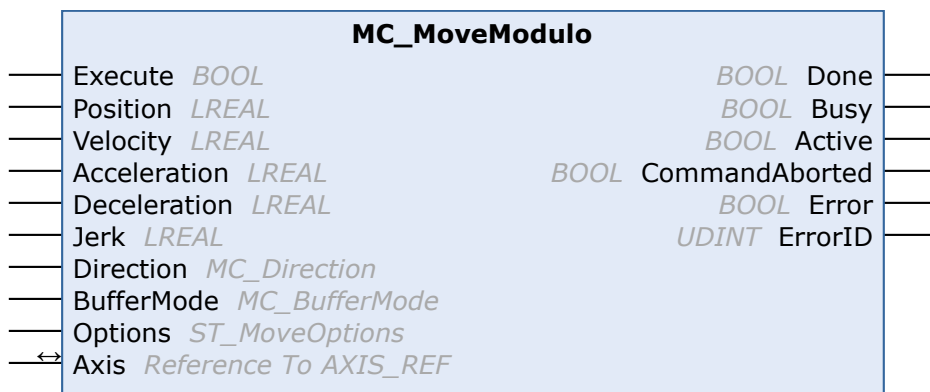


- "Target Position Monitoring" is activated (standard case): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- "Position Range Monitoring" is activated ("Target Position Monitoring" is not active): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- Neither "Target Position Monitoring" nor "Position Range Monitoring" is activated: the Done output is set immediately when the logical positioning end is reached (the NC setpoint generation is ended and the HasJob bit is FALSE).

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.1.4 MC\_MoveModulo**



The function block MC\_MoveModulo is used to execute a positioning operation, which refers to the modulo position of an axis. A modulo rotation is based on the adjustable axis parameter modulo factor (e.g. 360°). A distinction is made between three possible start types, depending on the "Direction" input.

- Positioning in positive direction
- Positioning in negative direction
- Positioning along shortest path

Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as MC\_MoveModulo then automatically leads to decoupling of the axis, after which the command is executed. In this case the only available BufferMode is "Aborting".

**Starting an axis from standstill**

If an axis is started from standstill with MC\_MoveModulo, it is possible to specify positions greater than or equal to 360°, in order to perform additional full turns. The same applies to a start with the BufferMode "MC\_Buffered".

**Starting an axis during motion**

If an axis is already in motion, certain special considerations apply.

The user is not able to specify the number of additional turns. The system automatically calculates how the axis can be moved to the target position on the shortest possible path.

The error output must be analyzed, because under certain conditions an oriented stop is not possible. For example, a standard stop may have been triggered just before, or an oriented stop would cause an active software limit switch to be exceeded. For all fault conditions, the axis is stopped safely, but it may subsequently not be at the required oriented position.

### Special cases

Particular attention should be paid to the behavior when requesting one or more complete modulo rotations. If the axis is located at an exact set position, such as 90 degrees, and if positioning to 90 degrees is required, no movement is carried out. If required to turn 450 degrees in a positive direction, it will perform just one rotation. The behavior can be different following an axis reset, because the reset will cause the current actual position to be adopted as the set position. The axis will then no longer be exactly at 90 degrees, but will be a little under or over. These cases will give rise either to a minimum positioning to 90 degrees, or on the other hand a complete rotation.

Depending on the particular case, it may be more effective for complete modulo rotations to calculate the desired target position on the basis of the current absolute position, and then to position using the function block `MC_MoveAbsolute` [► 65].

**i** Modulo positioning and absolute positioning are available for all axes, irrespective of the modulo setting in the TwinCAT System Manager. For each axis, the current absolute position "SetPos" can be read from the cyclic axis interface data type `NCTOPLC_AXIS_REF` [► 119].

See also: [Notes on modulo positioning](#) [► 74]

### Inputs

```
VAR_INPUT
  Execute      : BOOL;
  Position     : LREAL;
  Velocity     : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  Direction    : MC_Direction;
  BufferMode   : MC_BufferMode;
  Options     : ST_MoveOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Modulo target position to be used for positioning. If the axis is started from standstill, positions greater than 360° result in additional turns. Negative positions are not permitted.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.
Deceleration	LREAL	Deceleration (≥0) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk (≥0) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.
Direction	<a href="#">MC_Direction</a> [► 135]	Positive or negative direction of travel. If the axis is started during a motion, the direction may not be reversed.
BufferMode	<a href="#">MC_BufferMode</a> [► 132]	Is evaluated if the axis is already executing another command. <code>MC_MoveModulo</code> becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. A second function block is always



Name	Type	Description
		required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.
Options	<a href="#">ST_MoveOptions</a> [▶ 135]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks](#) [▶ 14]

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE when the target position has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks](#) [▶ 14]



- "Target Position Monitoring" is activated (standard case): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- "Position Range Monitoring" is activated ("Target Position Monitoring" is not active): when the logical positioning end is reached (the NC setpoint generation is finished and the HasJob bit is FALSE) AND afterwards the InPositionArea bit has become TRUE, then the Done output is set to TRUE.
- Neither "Target Position Monitoring" nor "Position Range Monitoring" is activated: the Done output is set immediately when the logical positioning end is reached (the NC setpoint generation is ended and the HasJob bit is FALSE).

## Requirements

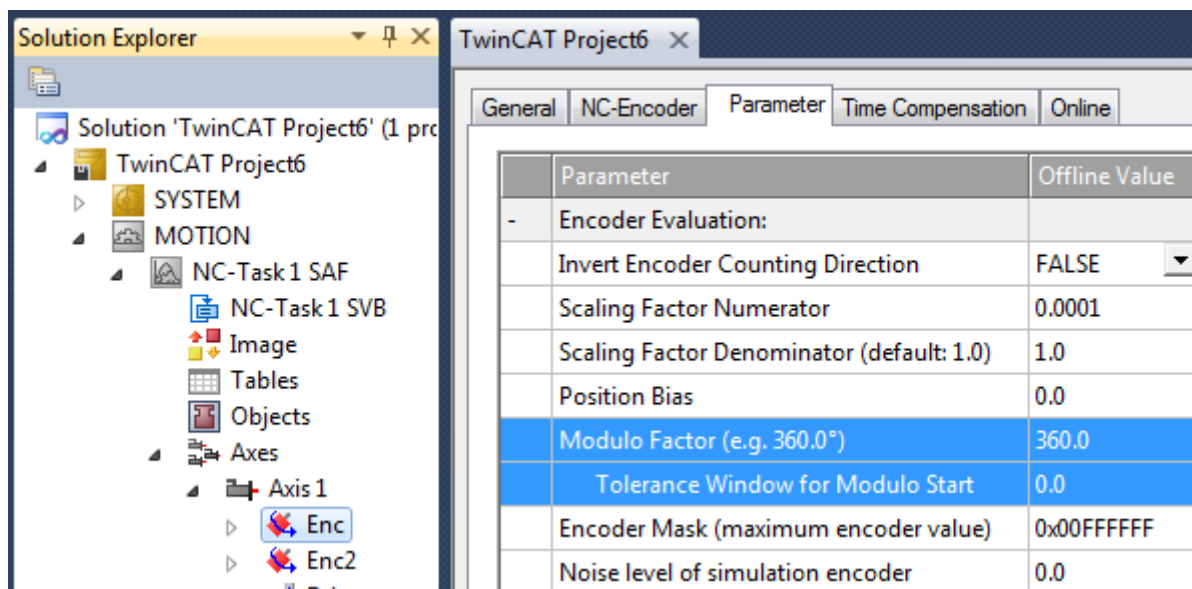
Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 6.1.5 Notes on modulo positioning

Modulo positioning ([MC\\_MoveModulo](#) [► 71]) is possible irrespective of the axis type. It may be used both for linear or rotary axes, because TwinCAT makes no distinction between these types. A modulo axis has a consecutive absolute position in the range  $\pm\infty$ . The modulo position of the axis is simply a piece of additional information about the absolute axis position. Modulo positioning represents the required target position in a different way. Unlike absolute positioning, where the user specifies the target unambiguously, modulo positioning has some risks, because the required target position may be interpreted in different ways.

### Settings in the TwinCAT System Manager

Modulo positioning refers to a modulo period that can be set in the TwinCAT System Manager. The examples on this page assume a rotary axis with a modulo period of 360 degrees.



The modulo tolerance window defines a position window around the current modulo set position of the axis. The window width is twice the specified value (set position  $\pm$  tolerance value). A detailed description of the tolerance window is provided below.

### Special features of axis resets

Axis positioning always refers to the set position. The set position of an axis is normally the target position of the last travel command. An axis reset ([MC\\_Reset](#) [► 18], controller activation with [MC\\_Power](#) [► 17]) can lead to a set position that is different from that expected by the user, because in this case the current actual position is used as the set position. The axis reset resets any lag error that may have occurred as a result of this procedure. If this possibility is not considered, subsequent positioning may lead to unexpected behavior.

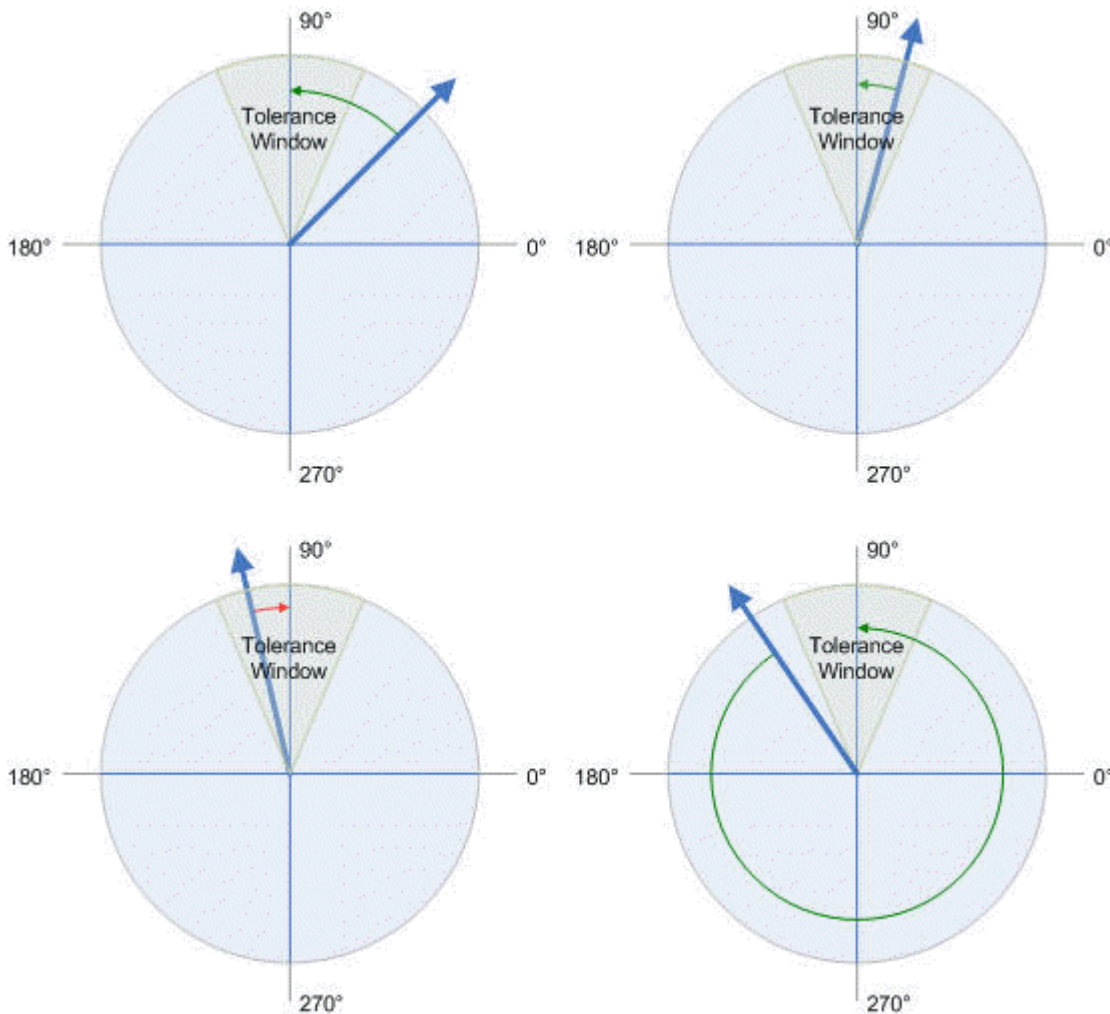
#### Example:

An axis is positioned to 90°, with the result that subsequently the set position of the axis is exactly 90°. A further modulo travel command to 450° in positive direction results in a full turn, with the subsequent modulo position of the axis is once again exactly 90°. If an axis reset is then carried out, the set position can randomly be somewhat smaller or slightly larger than 90°. The new value depends on the actual value of the axis at the time of the reset. However, the next travel command will lead to different results. If the set position is slightly less than 90°, a new travel command to 90° in positive direction only leads to minimum motion. The deviation created by the reset is compensated, and the subsequent set position is once again exactly 90°. However, if the set position after the axis reset is slightly more than 90°, the same travel

command leads to a full turn to reach the exact set position of 90°. This problem occurs if complete turns by 360° or multiples of 360° were initiated. For positioning to an angle that is significantly different from the current modulo position, the travel command is unambiguous.

To solve the problem, a modulo tolerance window can be parameterized in the TwinCAT System Manager. This ensures that small deviations from the position that are within the window do not lead to different axis behavior. If, for example, a window of 1° is parameterized, in the case described above the axis will behave identically, as long the set position is between 89° and 91°. If the set position exceeds 90° by less than 1°, the axis is re-positioned in positive direction at a modulo start. In both cases, a target position of 90° therefore leads to minimum movement to exactly 90°. A target position of 450° leads to a full turn in both cases.

**Effect of the modulo tolerance window: Modulo target position 90° in positive direction**



For values that are within the window range, the modulo tolerance window can therefore lead to movements against the specified direction. For small windows this is usually not a problem, because system deviations between set and actual position are compensated in both directions. This means that the tolerance window may also be used for axes that may only be moved in one direction due to their construction.

**Modulo positioning by less than one turn**

Modulo positioning from a starting position to a non-identical target position is unambiguous and requires no special consideration. A modulo target position in the range  $[0 \leq \text{position} < 360]$  reaches the required target in less than one whole turn. No motion occurs if target position and starting position are identical. Target positions of more than 360 degrees lead to one or more full turns before the axis travels to the required target position.

For a movement from 270° to 0°, a modulo target position of 0° (not 360°) should therefore be specified, because 360° is outside the basic range and would lead to an additional turn.

For modulo positioning, a distinction is made between three different directions, i.e. positive direction, negative direction and along shortest path ([MC\\_Direction](#) [[▶ 135](#)]). For positioning along the shortest path, target positions of more than 360° are not sensible, because the movement towards the target is always direct. In contrast to positive or negative direction, it is therefore not possible to carry out several turns before the axis moves to the target.



For modulo positioning with start type "along shortest path", only modulo target positions within the basic period (e.g. less than 360°) are permitted, otherwise an error is returned.

The following table shows some positioning examples:

Direction (modulo start type)	Absolute starting position	Modulo target position	Relative travel path	Absolute end position	Modulo end position
Positive direction	90.00	0.00	270.00	360.00	0.00
Positive direction	90.00	360.00	630.00	720.00	0.00
Positive direction	90.00	720.00	990.00	1080.00	0.00
Negative direction	90.00	0.00	-90.00	0.00	0.00
Negative direction	90.00	360.00	-450.00	-360.00	0.00
Negative direction	90.00	720.00	-810.00	-720.00	0.00
Along shortest path	90.00	0.00	-90.00	0.00	0.00

### Modulo positioning with full turns

In principle, modulo positioning by one or full turns are no different than positioning to an angle that differs from the starting position. No motion occurs if target position and starting position are identical. For a full turn, 360° has to be added to the starting position.

The reset behavior described above shows that positioning with full turns requires particular attention. The following table shows positioning examples for a starting position of approximately 90°. The modulo tolerance window (TW) is set to 1°. Special cases for which the starting position is outside this window are identified.

Direction (modulo start type)	Absolute starting position	Modulo target position	Relative travel path	Absolute end position	Modulo end position	Note
Positive direction	90.00	90.00	0.00	90.00	90.00	
Positive direction	90.90	90.00	-0.90	90.00	90.00	
Positive direction	91.10	90.00	358.90	450.00	90.00	outside TW
Positive direction	89.10	90.00	0.90	90.00	90.00	
Positive direction	88.90	90.00	1.10	90.00	90.00	outside TW
Positive direction	90.00	450.00	360.00	450.00	90.00	
Positive direction	90.90	450.00	359.10	450.00	90.00	
Positive direction	91.10	450.00	718.90	810.00	90.00	outside TW
Positive direction	89.10	450.00	360.90	450.00	90.00	
Positive direction	88.90	450.00	361.10	450.00	90.00	outside TW
Positive direction	90.00	810.00	720.00	810.00	90.00	
Positive direction	90.90	810.00	719.10	810.00	90.00	
Positive direction	91.10	810.00	1078.90	1,170.00	90.00	outside TW
Positive direction	89.10	810.00	720.90	810.00	90.00	
Positive direction	88.90	810.00	721.10	810.00	90.00	outside TW
Negative direction	90.00	90.00	0.00	90.00	90.00	
Negative direction	90.90	90.00	-0.90	90.00	90.00	
Negative direction	91.10	90.00	-1.10	90.00	90.00	outside TW
Negative direction	89.10	90.00	0.90	90.00	90.00	
Negative direction	88.90	90.00	-358.90	-270.00	90.00	outside TW

Direction (modulo start type)	Absolute starting position	Modulo target position	Relative travel path	Absolute end position	Modulo end position	Note
Negative direction	90.00	450.00	-360.00	-270.00	90.00	
Negative direction	90.90	450.00	-360.90	-270.00	90.00	
Negative direction	91.10	450.00	-361.10	-270.00	90.00	outside TW
Negative direction	89.10	450.00	-359.10	-270.00	90.00	
Negative direction	88.90	450.00	-718.90	-630.00	90.00	outside TW
Negative direction	90.00	810.00	-720.00	-630.00	90.00	
Negative direction	90.90	810.00	-720.90	-630.00	90.00	
Negative direction	91.10	810.00	-721.10	-630.00	90.00	outside TW
Negative direction	89.10	810.00	-719.10	-630.00	90.00	
Negative direction	88.90	810.00	-1078.90	-990.00	90.00	outside TW

**Modulo calculations within the PLC program**

In TwinCAT NC, all axis positioning tasks are executed based on the set position. The current actual position is only used for control purposes. If a PLC program is to calculate a new target position based on the current position, the current set position of the axis has to be used in the calculation (Axis.NcToPlc.ModuloSetPos and Axis.NcToPlc.ModuloSetTurns).

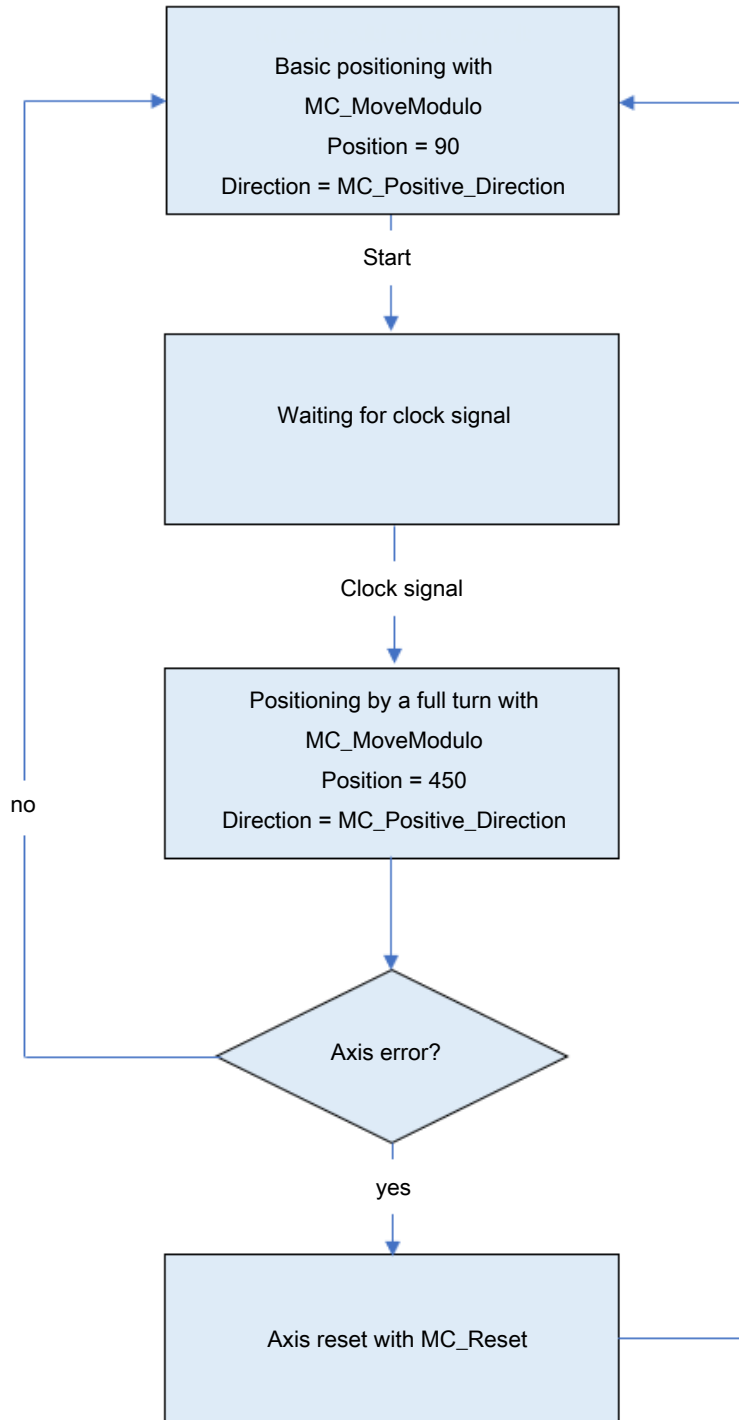
It is not advisable to perform order calculations based on the actual modulo position, which is available in the cyclical axis interface (ModuloActPos and ModuloActTurns). Due to the greater or lesser control deviation of the axis, errors in the programmed sequence, such as undesired rotations, could occur.

**Application example**

Within a system, a rotational axis carries out an operation. The starting position for each operation is 90°, and with each cycle the axis is to be positioned by 360° in positive direction. Reverse positioning is not permitted for mechanical reasons. Small reverse positioning is acceptable as part of position control movements.

The modulo tolerance window is set to 1.5° in the System Manager. This prevents undesired axis rotations after an axis reset. Since the axis may only be pre-positioned, the motion command [MC\\_MoveModulo \[► 71\]](#) with the modulo startup type "positive direction" (MC\_Positive\_Direction) is used. The modulo target position is specified as 450°, since the original orientation is to be reached again after a full turn by 360°. A modulo target position of 90° would not lead to any motion.

The process starts with a basic positioning movement ([MC\\_MoveModulo \[► 71\]](#)) to ensure that the starting position is accurate. The step sequence then changes into an execution cycle. In the event of a fault, the axis is reset with [MC\\_Reset \[► 18\]](#) and subsequently (at the start of the step sequence) moved to its valid starting position. In this case, 90° is specified as the target position to enable this position to be reached as quickly as possible. No motion occurs if the axis is already at the starting position.

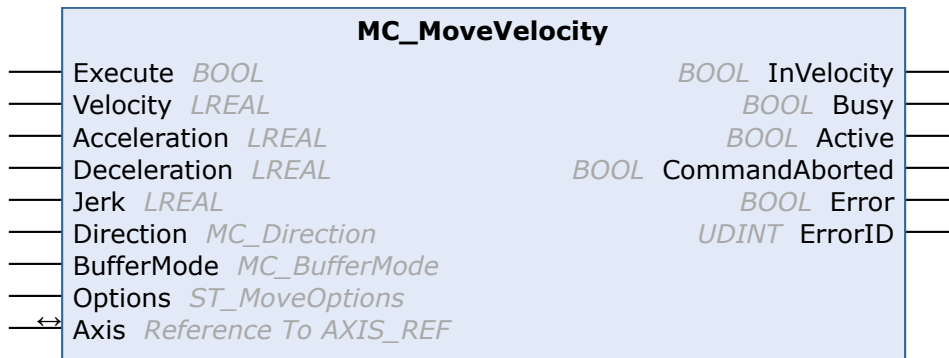


Alternatively, the reset step may be carried out at the start of the step sequence, so that the axis is initialized at the start of the process.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 6.1.6 MC\_MoveVelocity



The function block MC\_MoveVelocity is used to start an endless travel with a specified velocity and direction. The movement can be stopped through a Stop command.

The InVelocity output is set once the constant velocity is reached. Once constant velocity has been reached, the block function is complete, and no further monitoring of the movement takes place. If the command is aborted during the acceleration phase, the output "CommandAborted" or, in the event of an error, the output "Error" is set.

Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as MC\_MoveAbsolute then automatically leads to decoupling of the axis, after which the command is executed. In this case the only available BufferMode is "Aborting".

#### Inputs

```

VAR_INPUT
    Execute      : BOOL; (* B *)
    Velocity     : LREAL; (* E *)
    Acceleration : LREAL; (* E *)
    Deceleration : LREAL; (* E *)
    Jerk         : LREAL; (* E *)
    Direction    : MC_Direction := MC_Positive_Direction; (* E *)
    BufferMode    : MC_BufferMode; (* E *)
    Options      : ST_MoveOptions; (* V *)
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.
Deceleration	LREAL	Deceleration (≥0) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk (≥0) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.
Direction	<a href="#">MC_Direction [► 135]</a>	Positive or negative direction of travel.
BufferMode	<a href="#">MC_BufferMode [► 132]</a>	Is evaluated if the axis is already executing another command. MC_MoveVelocity becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. If the command is applied to a coupled slave axis, the only available BufferMode is "Aborting". A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.

Name	Type	Description
Options	ST_MoveOptions [► 135]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks \[► 14\]](#)

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  InVelocity      : BOOL; (* B *)
  Busy            : BOOL; (* E *)
  Active          : BOOL; (* E *)
  CommandAborted : BOOL; (* E *)
  Error           : BOOL; (* B *)
  ErrorID        : UDINT; (* E *)
END_VAR
```

Name	Type	Description
InVelocity	BOOL	After the axis acceleration, the InVelocity output assumes the value TRUE once the requested target velocity has been reached. The InVelocity output remains TRUE until the axis velocity is changed by another command. As soon as a velocity deviation is determined, the output signals FALSE.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the function block is active. If "Busy" is FALSE, the function block is ready for a new order. At the same time one of the outputs "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

#### InVelocity and Override

**i** If the override value is changed, the InVelocity output signals the value FALSE. At the same time, after a change in velocity, the CommandAborted output assumes the value TRUE and the function block no longer displays the status "busy".

#### InVelocity during a velocity overlay with MC\_MoveSuperImposed

**i** A change in velocity due to the function block MC\_MoveSuperImposed is permitted and does not lead to a cancellation of the activity of the function block. Although the velocity changes, the InVelocity output remains TRUE because the total velocity is made up of the basic velocity, which has not changed, and a velocity superimposed on this basic velocity (Superposition).



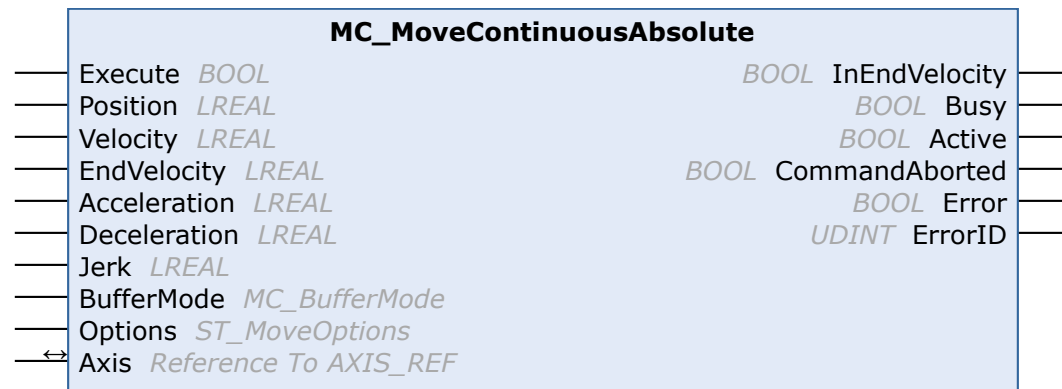
The function block remains "Busy" and "Active" until a new command is issued.

See also: [General rules for MC function blocks \[► 14\]](#)

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 6.1.7 MC\_MoveContinuousAbsolute



The function block MC\_MoveContinuousAbsolute starts the positioning to an absolute target position and monitors the axis movement over the entire travel path. At the target position a constant end velocity is reached, which is maintained. The "InEndVelocity" output is set once the target position was reached. Otherwise, the output "CommandAborted" or, in case of an error, the output "Error" is set.

Once the target position has been reached, the function of the function block is terminated and the axis is no longer monitored.



MC\_MoveContinuousAbsolute is not implemented for high/low speed axes.

**Inputs**

```

VAR_INPUT
  Execute      : BOOL;
  Position     : LREAL;
  Velocity     : LREAL;
  EndVelocity  : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  BufferMode   : MC_BufferMode;
  Options     : ST_MoveOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Absolute target position
Velocity	LREAL	Maximum velocity for the movement to the target position (>0).
EndVelocity	LREAL	End velocity to be maintained once the target position has been reached.
Acceleration	LREAL	Acceleration (≥0) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Name	Type	Description
Deceleration	LREAL	Deceleration ( $\geq 0$ ) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk ( $\geq 0$ ) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.
BufferMode	<a href="#">MC_BufferMode</a> [► 132]	Is evaluated if the axis is already executing another command. MC_MoveContinuousAbsolute becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.
Options	<a href="#">ST_MoveOptions</a> [► 135]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks](#) [► 14]

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  InEndVelocity : BOOL;
  Busy          : BOOL;
  Active        : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorID       : UDINT;
END_VAR
```

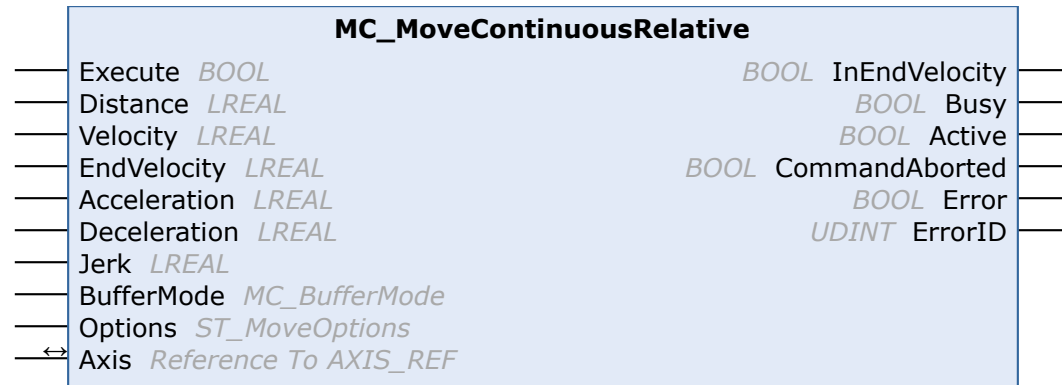
Name	Type	Description
InEndVelocity	BOOL	TRUE when the target position has been reached. The function block remains "Busy" and "Active" until a new command is issued.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time one of the outputs "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks](#) [► 14]

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.1.8 MC\_MoveContinuousRelative**



The function block MC\_MoveContinuousRelative is used to start a positioning by a relative distance and to monitor the axis movement over the entire travel path. At the target position a constant end velocity is reached, which is maintained. The "InEndVelocity" output is set once the target position was reached. Otherwise, the output "CommandAborted" or, in case of an error, the output "Error" is set.

Once the target position has been reached, the block function is complete and the axis is no longer monitored.



MC\_MoveContinuousRelative is not implemented for fast/slow axes.

**Inputs**

```

VAR_INPUT
  Execute      : BOOL;
  Distance     : LREAL;
  Velocity     : LREAL;
  EndVelocity  : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  BufferMode   : MC_BufferMode;
  Options     : ST_MoveOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Distance	LREAL	Relative distance to be used for positioning.
Velocity	LREAL	Maximum velocity for the movement over the "Distance" (>0).
EndVelocity	LREAL	End velocity to be maintained after the relative "Distance".
Acceleration	LREAL	Acceleration (≥0) If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.
Deceleration	LREAL	Deceleration (≥0) If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Name	Type	Description
Jerk	LREAL	Jerk ( $\geq 0$ ) At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.
BufferMode	MC_BufferMode [► 132]	Is evaluated if the axis is already executing another command. MC_MoveContinuousRelative becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.
Options	ST_MoveOptions [► 135]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks \[► 14\]](#)

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  InEndVelocity : BOOL;
  Busy          : BOOL;
  Active        : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorID       : UDINT;
END_VAR
```

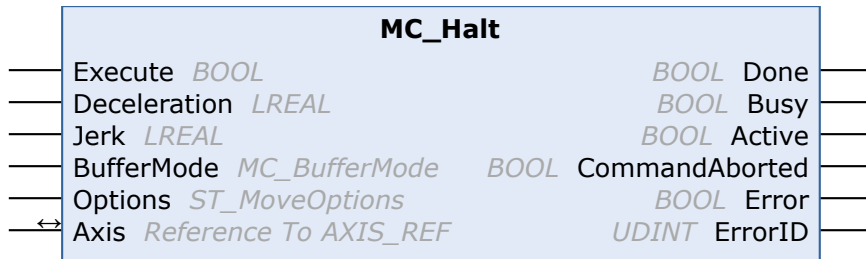
Name	Type	Description
InEndVelocity	BOOL	TRUE when the target position has been reached. The function block remains "Busy" and "Active" until a new command is issued.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time one of the outputs "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[► 14\]](#)

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.1.9 MC\_Halt**



The function block MC\_Halt is used to stop an axis with a defined deceleration ramp.

In contrast to [MC\\_Stop \[► 87\]](#), the axis is not locked against further movement commands. The axis can therefore be restarted through a further command during the deceleration ramp or after it has come to a halt.

Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as MC\_Halt then automatically leads to decoupling of the axis, after which the command is executed. In this case the only available BufferMode is "Aborting".

**Inputs**

```

VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode    : MC_BufferMode;
    Options      : ST_MoveOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Deceleration	LREAL	Deceleration If the value is $\leq 0$ , the deceleration parameterized with the last Move command is used. For safety reasons MC_Halt and <a href="#">MC_Stop [► 87]</a> cannot be executed with weaker dynamics than the currently active motion command. The parameterization is adjusted automatically, if necessary.
Jerk	LREAL	Jerk If the value is $\leq 0$ , the jerk parameterized with the last Move command takes effect. For safety reasons MC_Halt and <a href="#">MC_Stop [► 87]</a> cannot be executed with weaker dynamics than the currently active motion command. The parameterization is adjusted automatically, if necessary.
BufferMode	<a href="#">MC_BufferMode [► 132]</a>	Is evaluated if the axis is already executing another command. MC_Halt becomes active after the current command or aborts it. Transition conditions from the current to the next command are also determined by the BufferMode. If the command is applied to a coupled slave axis, only the BufferMode "Aborting" is possible. Special characteristics of MC_Halt: the BufferMode "Buffered" shows no effect, because the command is executed only at standstill. The blending modes "MC_BlendingNext" and "MC_BlendingLow" do not change the last target position, although they can result in

Name	Type	Description
		a change in dynamics (deceleration) of the stop ramp. The modes "MC_BlendingPrevious" and "MC_BlendingHigh" extend the travel to the original target position. The stop ramp is only initiated when this position is reached (defined braking point).
Options	<a href="#">ST_MoveOptions</a> [► 135]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks](#) [► 14]

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [► 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
END_VAR
```

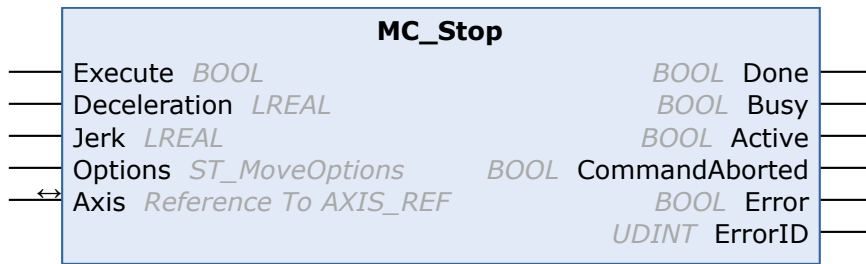
Name	Type	Description
Done	BOOL	TRUE if the axis has been stopped and is stationary.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the function block is active. If "Busy" is FALSE, the function block is ready for a new order. At the same time one of the outputs "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks](#) [► 14]

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 6.1.10 MC\_Stop



The function block MC\_Stop is used to stop an axis with a defined deceleration ramp and to lock the axis against other motion commands. The function block is therefore suitable for stops in special situations, in which further axis movements are to be prevented.

At the same time the axis is blocked for other motion commands. The axis can only be restarted once the Execute signal has been set to FALSE after the axis has stopped. A few cycles are required to release the axis after a falling edge of Execute. During this phase the "Busy" output remains TRUE, and the function block has to be called until "Busy" becomes FALSE.

The locking of the axis is canceled with an [MC\\_Reset](#) [[18](#)].

Alternatively, the axis can be stopped with [MC\\_Halt](#) [[85](#)] without locking. MC\_Halt is preferable for normal movement sequences.

Motion commands can be applied to coupled slave axes, if this option was explicitly activated in the axis parameters. A motion command such as MC\_Stop then automatically leads to decoupling of the axis, after which the command is executed.

#### Inputs

```
VAR_INPUT
  Execute      : BOOL;
  Deceleration : LREAL;
  Jerk         : LREAL;
  Options      : ST_MoveOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge. The axis is locked during the stop. The axis can only be restarted once the Execute signal has been set to FALSE after the axis has stopped.
Deceleration	LREAL	Deceleration If the value is $\leq 0$ , the deceleration parameterized with the last Move command is used. For safety reasons MC_Stop and MC_Halt cannot be executed with weaker dynamics than the currently active motion command. The parameterization is adjusted automatically, if necessary.
Jerk	LREAL	Jerk If the value is $\leq 0$ , the jerk parameterized with the last Move command is used. For safety reasons MC_Stop and MC_Halt cannot be executed with weaker dynamics than the currently active motion command. The parameterization is adjusted automatically, if necessary.
Options	<a href="#">ST_MoveOptions</a> [ <a href="#">135</a> ]	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks](#) [[14](#)]

 **Inputs/outputs**

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

```
VAR_OUTPUT
  Done          : BOOL;
  Busy          : BOOL;
  Active        : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorID       : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the axis has been stopped and is stationary.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. "Busy" remains TRUE as long as the axis is locked. The axis is only unlocked and "Busy" becomes FALSE when Execute is set to FALSE.
Active	BOOL	Indicates that the function block controls the axis. Remains TRUE as long as the axis is locked. The axis is only unlocked and "Active" becomes FALSE when Execute is set to FALSE.
CommandAborted	BOOL	TRUE if the command could not be executed completely.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks](#) [[▶ 14](#)]

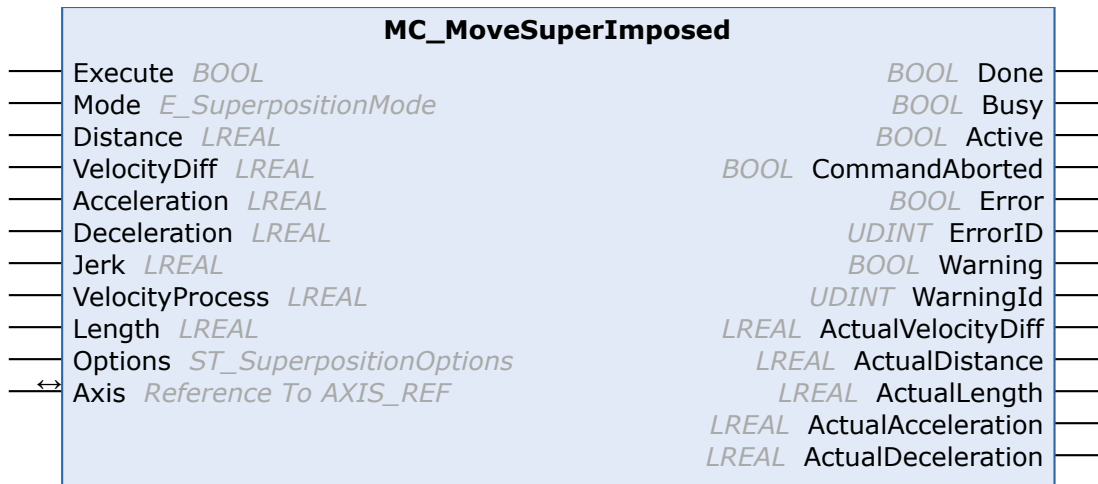
**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2



## 6.2 Superposition

### 6.2.1 MC\_MoveSuperImposed



The function block MC\_MoveSuperImposed starts a relative superimposed movement while the axis is already moving. The current movement is not interrupted. The "Done" output is set once the superimposed movement is completed. The original subordinate movement may continue to be active and is monitored by the associated Move function block.

The function of the superimposition becomes clear when considering two axes that operate with the same velocity. If one of the axes is superimposed by MC\_MoveSuperImposed, it moves ahead or behind by the "Distance" parameter. Once the superimposed movement is completed, the "Distance" between the two axes is maintained.

MC\_MoveSuperImposed can be executed on single axes as well as on master or slave axes. In a slave axis, the superimposed movement acts solely on the slave axis. If the function is applied to a master axis, the slave mimics the superimposed movement of the master due to the axis coupling.

Since MC\_MoveSuperImposed executes a relative superimposed movement, the target position for the subordinate travel command changes by Distance.

The superimposed movement depends on the position of the main movement. This means that a velocity change of the main movement also results in a velocity change in the superimposed movement, and that the superimposed movement is inactive if the main movement stops. The "Options" parameter can be used to specify whether the superimposed movement is to be aborted or continued if the main movement stops.

See also: [Application examples for MC\\_MoveSuperImposed \[► 91\]](#)

#### Inputs

```

VAR_INPUT
  Execute          : BOOL;
  Mode             : E_SuperpositionMode;
  Distance         : LREAL;
  VelocityDiff    : LREAL;
  Acceleration     : LREAL;
  Deceleration    : LREAL;
  Jerk            : LREAL;
  VelocityProcess : LREAL;
  Length          : LREAL;
  Options         : ST_SuperpositionOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Mode	E_SuperpositionMode	Determines the type of the superimposed motion. (Type: E_SuperpositionMode [► 150])

Name	Type	Description
Distance	LREAL	Relative distance to catch up. A positive value means an increase in velocity by an amount required to cover the additional distance, compared with the unaffected movement. A negative value results in braking and falling back by this distance.
VelocityDiff	LREAL	Maximum velocity difference to the current velocity (basic velocity) of the axis (>0). For this parameter a distinction may have to be made, depending on the superimposition direction (acceleration or deceleration). If, for example, a direction reversal is not permitted, the maximum available acceleration corresponds to the maximum velocity, and the maximum deceleration to stop. Therefore, there are two possible maximum values for "VelocityDiff": <ul style="list-style-type: none"> <li>Distance &gt; 0 (axis accelerates) VelocityDiff = maximum velocity -basic velocity</li> <li>Distance &lt; 0 (axis decelerates) VelocityDiff = basic velocity</li> </ul>
Acceleration	LREAL	Acceleration ( $\geq 0$ ). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.
Deceleration	LREAL	Deceleration ( $\geq 0$ ). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	The Jerk parameter is not implemented.
VelocityProcess	LREAL	Mean process velocity in the axis (>0). If the basic velocity during superposition is constant, the set axis velocity can be specified.
Length	LREAL	Distance over which the superimposed movement is available. The "Mode" parameter defines how this distance is interpreted.
Options	<a href="#">ST_SuperpositionOptions</a> [▶ 152]	Data structure containing additional, rarely used parameters. The input can normally remain open. <ul style="list-style-type: none"> <li><b>AbortOption:</b> defines the behavior when the subordinate movement stops. The superimposed movement can be aborted or continued later.</li> </ul>

See also: [General rules for MC function blocks](#) [▶ 14]

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	<a href="#">AXIS_REF</a> [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
  Warning        : BOOL;
  WarningID      : UDINT;
```

```

ActualVelocityDiff : LREAL;
ActualDistance     : LREAL;
ActualLength       : LREAL;
ActualAcceleration : LREAL;
ActualDeceleration : LREAL;
END_VAR

```

Name	Type	Description
Done	BOOL	TRUE if the superimposed movement was successfully completed.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed.
CommandAborted	BOOL	Becomes TRUE, if the command was aborted by another command and could therefore not be completed.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
Warning	BOOL	TRUE if the action cannot be executed completely.
WarningID	UDINT	The function block returns warning 4243 <sub>hex</sub> (16963) if the compensation was incomplete due to the parameterization (distance, velocity, etc.). In this case compensation is implemented as far as possible. The user has to decide whether to interpret this warning message within his application as a proper error or merely as a warning.
ActualVelocityDiff	LREAL	Actual velocity difference during the superimposed motion (ActualVelocityDiff ≤ VelocityDiff).
ActualDistance	LREAL	Actual superimposed distance. The function block tries to reach the full "Distance" as specified. Depending on the parameterization ("VelocityDiff", "Acceleration", "Deceleration", "Length", "Mode") this distance may not be fully reached. In this case the maximum possible distance is superimposed. (ActualDistance ≤ Distance).
ActualLength	LREAL	Actual travel during superimposed motion (ActualLength ≤ Length).
ActualAcceleration	LREAL	Actual acceleration of the superimposed movement (ActualAcceleration ≤ Acceleration).
ActualDeceleration	LREAL	Actual deceleration of the superimposed movement (ActualDeceleration ≤ Deceleration).

See also: [General rules for MC function blocks \[► 14\]](#)

**Requirements**

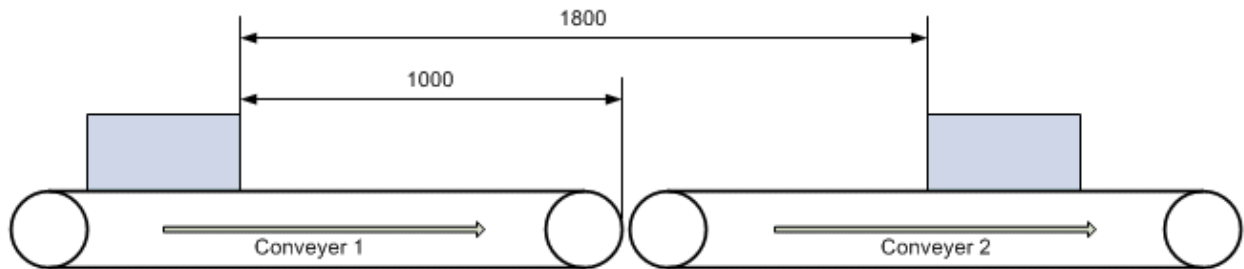
Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.2.2 Application examples for MC\_MoveSuperImposed**

The function block [MC\\_MoveSuperImposed \[► 89\]](#) starts a superimposed movement on an axis that is already moving. For this superposition various applications are available that are described below.

### Distance correction for products on a conveyor belt

A conveyor belt consists of individual segments, each driven by an axis. The conveyor belt is used for transporting packages, the spacing of which is to be corrected. To this end a conveying segment must briefly run faster or slower relative to a following segment.



The measured distance is 1800 mm and is to be reduced to 1500 mm. Conveyor belt 1 should be accelerated in order to reduce the distance. The correction must be completed by the time the end of belt 1 is reached in order to prevent the package being pushed onto the slower belt 2.

Since in this situation conveyor 1 has to be accelerated the drive system requires a velocity reserve, assumed to be 500 mm/s in this case. In practice this value can be determined from the difference between the maximum conveyor velocity and the current set velocity.

For parameterization of function block [MC\\_MoveSuperImposed](#) [► 89] this means:

Distance = 1800 mm - 1500 mm = 300 mm (distance correction)

Length = 1000 mm (available distance up to the end of belt 1)

Mode = SUPERPOSITIONMODE\_VELOREDUCTION\_LIMITEDMOTION

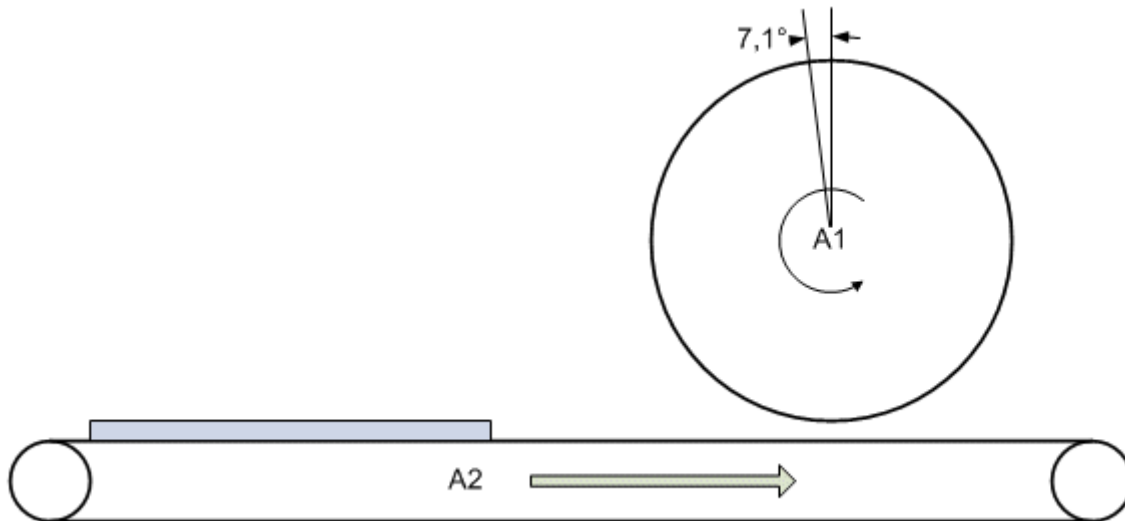
VelocityDiff = 500 mm/s

The "mode" defines that the distance "Length" up to the end of the conveyor belt is used for the correction and that the correction is completed at this point. The system uses the internally calculated velocity as degree of freedom. "VelocityDiff" therefore is the upper limit for the velocity change in this case.

Alternatively the correction could be achieved by decelerating belt 2. In this case, "Distance" must be specified as a negative value, and the available correction distance "Length" is the distance between the package on the right and the end of the belt. The maximum possible velocity change "VelocityDiff" corresponds to the current set velocity. Belt 2 could therefore be decelerated down to zero, if necessary.

### Phase shift of a print roller

A print roller rotates with constant peripheral velocity at the same velocity as conveyor belt on which a workpiece to be printed is transported. For synchronization with the workpiece the print roller is to be advanced by a certain angle (phase shift).



The phase shift can be implemented in two ways:

- The angle can be corrected as quickly as possible, resulting in a short-term strong increase in the velocity of the print roller.
- A correction distance can be defined within which the correction can take place, e.g. a full roller revolution. This results in the following possibilities for parameterization of the function block [MC\\_MoveSuperImposed](#) [► 89].

#### Fast correction:

Distance =  $7.1^\circ$

Length =  $360^\circ$  (maximum possible correction distance)

Mode = SUPERPOSITIONMODE\_LENGTHREDUCTION\_LIMITEDMOTION

VelocityDiff =  $30^\circ/\text{s}$  (velocity reserve)

The "mode" specifies that the correction distance should be as short as possible. The stated value for "Length" therefore is an upper limit that can be chosen freely (but not too small).

Alternatively SUPERPOSITIONMODE\_VELOREDUCTION\_ADDITIVEMOTION can be used as "Mode". In this case the whole correction distance would be up to  $367.1^\circ$ . Since the distance should be as short as possible both modes are equivalent in this case.

#### Slow correction:

Distance =  $7.1^\circ$

Length =  $360^\circ$  (correction distance)

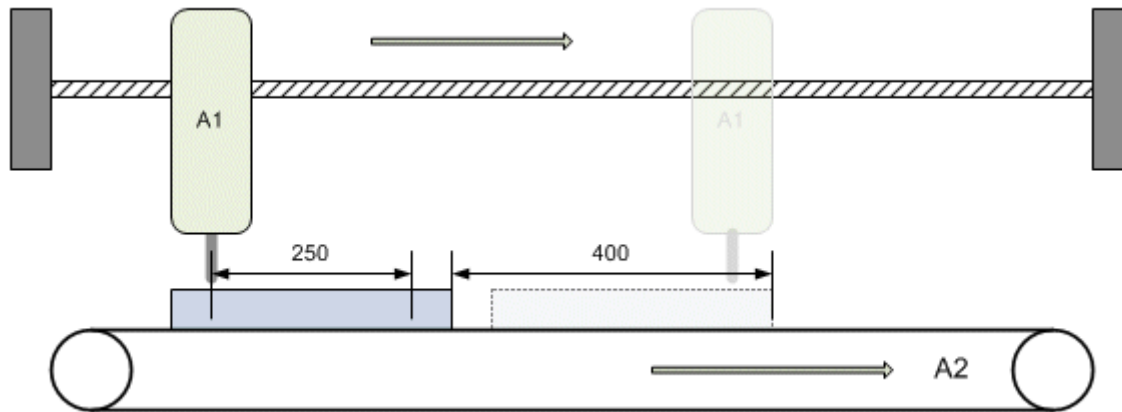
Mode = SUPERPOSITIONMODE\_VELOREDUCTION\_LIMITEDMOTION

VelocityDiff =  $30^\circ/\text{s}$  (velocity reserve)

The mode specifies that the correction distance should be utilized fully and the velocity change should be kept as small as possible. The stated value for "VelocityDiff" therefore is an upper limit that can be chosen freely (but not too small).

#### Drilling unit

A drilling unit should drill two holes in a moving workpiece. Synchronization for the first hole is assumed to be achieved via the flying saw (MC\_GearInPos) and is not be considered here. After the first operation the device must be moved by certain distance relative to the moving workpiece.



The drilling unit is to be advanced by 250 mm relative to the workpiece after the first hole has been drilled. Meanwhile the workpiece covers a distance of 400 mm. From this position the drilling unit is once again synchronous with the workpiece and the second hole can be drilled.

Here too two options are available that differ in terms of the velocity change of the drilling device and therefore in the mechanical strain.

Parameterization of function block [MC\\_MoveSuperImposed](#) [▶ 89]:

#### Fast correction:

Distance = 250 mm

Length = 400 mm

Mode = SUPERPOSITIONMODE\_LENGTHREDUCTION\_ADDITIVEMOTION

VelocityDiff = 500 mm/s (velocity reserve of the drilling device)

The mode specifies that the correction distance should be as short as possible. The stated value for "Length" therefore is an upper limit that can be chosen freely (but not too small). The drilling unit can travel a larger distance since "Length" refers to the workpiece plus a relative change in position.

#### Slow correction:

Distance = 250 mm

Length = 400 mm

Mode = SUPERPOSITIONMODE\_VELOREDUCTION\_ADDITIVEMOTION

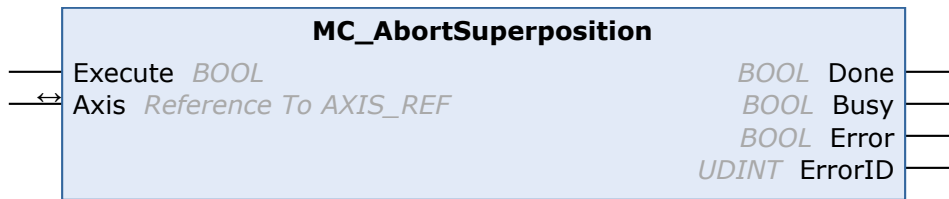
VelocityDiff = 500 mm/s (velocity reserve of the drilling device)

"Mode" specifies that the correction distance should be utilized fully and the velocity change should be kept as small as possible. The stated value for "VelocityDiff" therefore is an upper limit that can be chosen freely (but not too small). During the change in position the workpiece covers the distance "Length", the drilling unit travels 650 mm due to the additional correction distance (Length + Distance).

#### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 6.2.3 MC\_AbortSuperposition



The function block MC\_AbortSuperposition terminates a superimposed movement started by MC\_MoveSuperImposed [▶ 89], without stopping the subordinate axis movement.

A full axis stop can be achieved with MC\_Stop [▶ 87] or MC\_Halt [▶ 85], if necessary. In this case MC\_AbortSuperposition does not have to be called.

#### Inputs

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge and the superimposed movement is completed.

#### Inputs/outputs

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [▶ 118]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

#### Outputs

```
VAR_OUTPUT
    Done : BOOL;
    Busy : BOOL;
    Error : BOOL;
    ErrorID : UDINT;
END_VAR
```

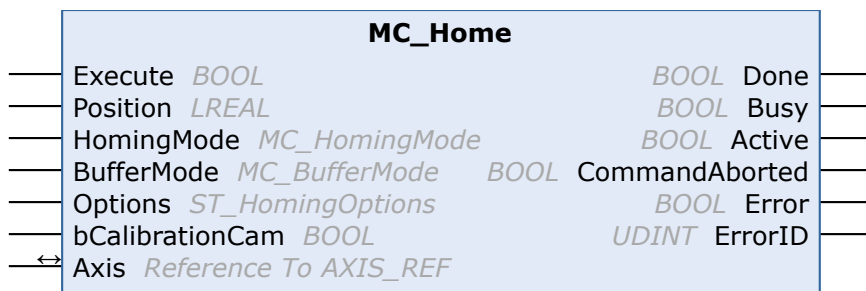
Name	Type	Description
Done	BOOL	TRUE if the superimposed movement was successfully completed.
Busy	BOOL	TRUE as soon as the function block is active. FALSE when it returns to its initial state.
Error	BOOL	TRUE as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

#### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 6.3 Homing

### 6.3.1 MC\_Home



An axis reference run is carried out with the function block MC\_Home.

The referencing mode is set in the TwinCAT System Manager with the encoder parameter "Reference Mode". Depending on the connected encoder system, different sequences are possible (see also Reference mode for incremental encoders in the TwinCAT 3 ADS Interface NC documentation).

#### Inputs

```

VAR_INPUT
  Execute      : BOOL;
  Position     : LREAL      := DEFAULT_HOME_POSITION;
  HomingMode   : MC_HomingMode;
  BufferMode   : MC_BufferMode;
  Options     : ST_HomingOptions;
  bCalibrationCam : BOOL;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Absolute reference position to which the axis is set after homing. Alternatively, the constant DEFAULT_HOME_POSITION can be used here. In this case, the "Reference position for homing" specified in the TwinCAT System Manager is used.
HomingMode	<a href="#">MC_HomingMode [► 130]</a>	Determines how the calibration is performed. <ul style="list-style-type: none"> <li>• MC_DefaultHoming Initiates default homing.</li> <li>• MC_Direct Sets the axis position directly to Position without executing a movement.</li> <li>• MC_ForceCalibration Enforces the "axis is calibrated" state. No movement takes place, and the position remains unchanged.</li> <li>• MC_ResetCalibration Resets the calibration state of the axis. No movement takes place, and the position remains unchanged.</li> </ul>
BufferMode	MC_BufferMode	Currently not implemented.
Options	<a href="#">ST_HomingOptions [► 130]</a>	Data structure containing additional, rarely used parameters. The input can normally remain open. <ul style="list-style-type: none"> <li>• <b>ClearPositionLag:</b> is only effective in "MC_Direct" mode. Can optionally be used to set the set and actual positions to the same value. In this case the following error is cleared.</li> </ul>



Name	Type	Description
bCalibrationCam	BOOL	Reflects the signal of a referencing cam that can be applied to the controller via a digital input.

**i** Since the reference position is generally set during the motion, the axis will not stop exactly at this position. The standstill position differs by the braking distance of the axis, although the calibration is nevertheless exact.

See also: [General rules for MC function blocks](#) [▶ 14]

 **Inputs/outputs**

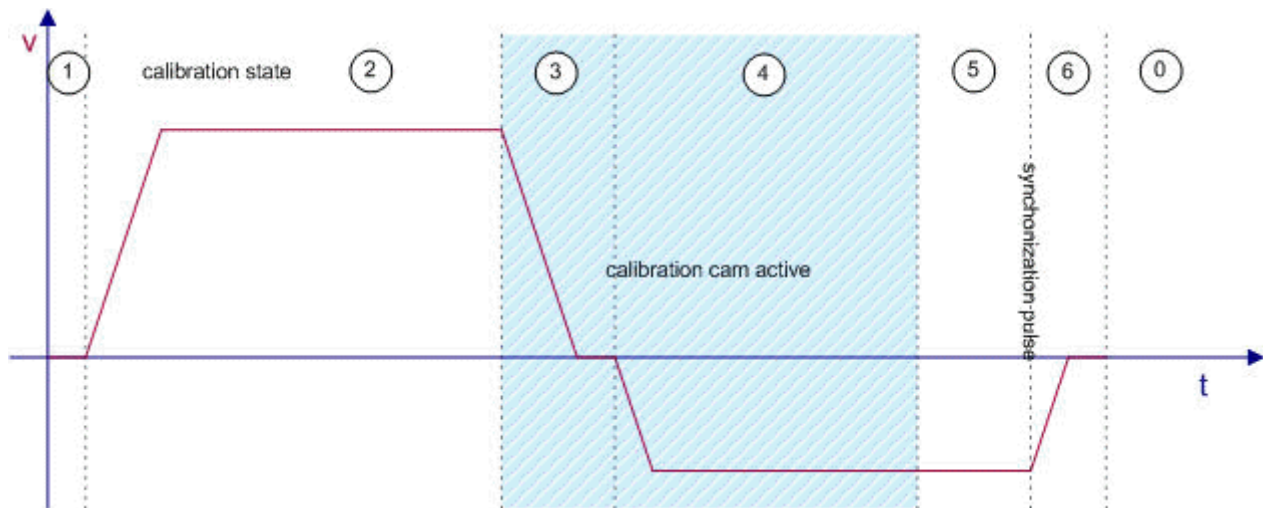
```
VAR_IN_OUT
  Axis      : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF	Axis data structure of the type <a href="#">AXIS_REF</a> [▶ 118], which uniquely addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**Referencing process**

The referencing process has several phases. The calibration state is signaled in the cyclic interface of the axis (Axis.NcToPlc.HomingState). The following diagram shows the process flow after starting function block MC\_Home with the individual phases.

If an axis is to be referenced without referencing cam, i.e. only based on the sync pulse of the encoder, the referencing cam can be simulated via the PLC program. The "bCalibrationCam" signal is initially activated and then canceled, if [Axis.NcToPlc.HomingState](#) [▶ 119] is equal or greater 4.



 **Outputs**

```
VAR_OUTPUT
  Done      : BOOL;
  Busy      : BOOL;
  Active    : BOOL;
  CommandAborted : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE when the axis has been calibrated and the movement is complete.

Name	Type	Description
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Currently not implemented - "Active" indicates that the command is running. If the command has been buffered, it may only become active after a running command has been terminated.
CommandAborted	BOOL	TRUE if the command could not be executed completely.
Error	BOOL	TRUE as soon as an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

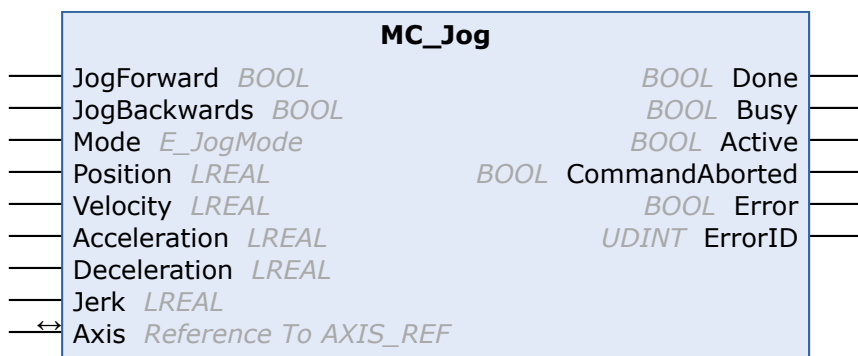
See also: [General rules for MC function blocks](#) [► 14]

## Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 6.4 Manual motion

### 6.4.1 MC\_Jog



The function block MC\_Jog enables an axis to be moved via manual keys. The key signal can be linked directly with the "JogForward" and "JogBackwards" inputs. The required operation mode is specified via the "mode" input. An inching mode for moving the axis by a specified distance whenever the key is pressed is also available. The velocity and dynamics of the motion can be specified depending on the operation mode.

#### Inputs

```

VAR_INPUT
  JogForward   : BOOL;
  JogBackwards : BOOL;
  Mode         : E_JogMode;
  Position     : LREAL;
  Velocity     : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
END_VAR

```

Name	Type	Description
JogForward	BOOL	The command is executed with a rising edge, and the axis is moved in positive direction of travel. Depending on the operation mode (see "Mode" input), the axis moves as long as the signal remains TRUE, or it stops automatically after a specified distance. During the motion no further signal edges are accepted (this includes the "JogBackwards" input). If a simultaneous signal edge occurs at the inputs "JogForward" and "JogBackwards", "JogForward" has priority.
JogBackwards	BOOL	The command is executed with a rising edge, and the axis is moved in negative direction of travel. "JogForward" and "JogBackwards" should be triggered alternatively, although they are also mutually locked internally.
Mode	<a href="#">E JogMode  &gt; 131</a>	<p>Determines the operation mode in which the manual function is executed.</p> <ul style="list-style-type: none"> <li>• MC_JOGMODE_STANDARD_SLOW the axis moves as long as the signal at one of the jog inputs is TRUE. The "low velocity for manual functions" specified in the TwinCAT System Manager and standard dynamics are used. In this operation mode the position, velocity and dynamics data specified in the function block have no effect.</li> <li>• MC_JOGMODE_STANDARD_FAST the axis moves as long as the signal at one of the jog inputs is TRUE. The "high velocity for manual functions" specified in the TwinCAT System Manager and standard dynamics are used. In this operation mode the position, velocity and dynamics data specified in the function block have no effect.</li> <li>• MC_JOGMODE_CONTINUOUS The axis moves as long as the signal at one of the jog inputs is TRUE. The velocity and dynamics data specified by the user are used. The position has no effect.</li> <li>• MC_JOGMODE_INCHING With a rising edge at one of the jog inputs the axis is moved by a certain distance which is specified via the "Position" input. The axis stops automatically, irrespective of the state of the jog inputs. A new movement step is only executed once a further rising edge is encountered. With each start the velocity and dynamics data specified by the user are used.</li> <li>• MC_JOGMODE_INCHING_MODULO With a rising edge at one of the jog inputs the axis is moved by a certain distance which is specified via the "Positions" input. The axis position will snap to an integer multiple of the position parameter. The axis stops automatically, irrespective of the state of the jog inputs. A new movement step is only executed once a further rising edge is encountered. With each start the velocity and dynamics data specified by the user are used.</li> </ul>
Position	LREAL	Relative distance for movements in MC_JOGMODE_INCHING operation mode.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Name	Type	Description
Deceleration	LREAL	Deceleration ( $\geq 0$ ). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.
Jerk	LREAL	Jerk ( $\geq 0$ ). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.



The parameters "Position", "Velocity", "Acceleration", "Deceleration" and "Jerk" are not used in the operation modes MC\_JOGMODE\_STANDARD\_SLOW and MC\_JOGMODE\_STANDARD\_FAST and can remain unassigned.

### Inputs/outputs

```
VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR
```

Name	Type	Description
Axis	AXIS_REF [ <a href="#">▶ 118</a> ]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done          : BOOL;
  Busy          : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorID      : UDINT;
END_VAR
```

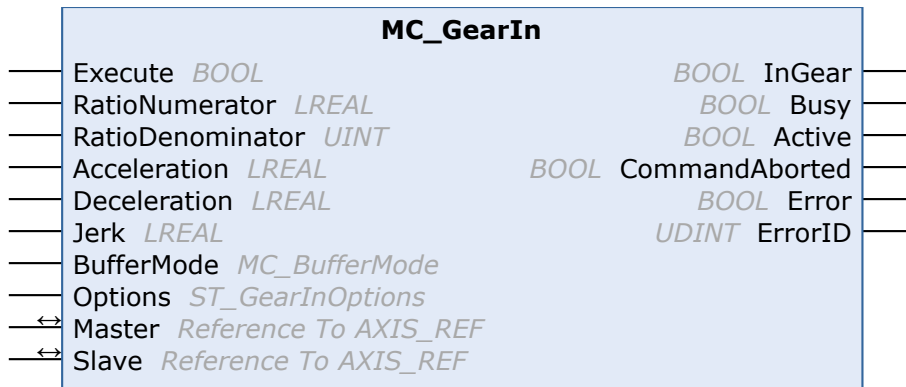
Name	Type	Description
Done	BOOL	TRUE if a movement was successfully completed.
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state. Only then can a further edge be accepted at the jog inputs.
Active	BOOL	Indicates that the axis is moved via the jog function.
CommandAborted	BOOL	TRUE if the process is interrupted by an external event, e.g. by the call up of <a href="#">MC_Stop</a> [ <a href="#">▶ 87</a> ].
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 6.5 Axis coupling

### 6.5.1 MC\_GearIn



The function block MC\_GearIn activates a linear master-slave coupling (gear coupling). The function block accepts a fixed gear ratio in numerator/denominator format.

The slave axis can be coupled to the master axis at standstill. With this function block it is not possible to synchronize to a moving master axis. The flying saw function blocks MC\_GearInVelo or MC\_GearInPos can be used for this purpose.

The slave axis can be decoupled with the function block [MC\\_GearOut \[► 104\]](#). If the slave is decoupled while it is moving, then it retains its velocity and can be halted using [MC\\_Stop \[► 87\]](#) or [MC\\_Halt \[► 85\]](#).

Alternatively, the function block [MC\\_GearInDyn \[► 102\]](#) with dynamically changeable gear ratio is available.

#### Inputs

```
VAR_INPUT
    Execute          : BOOL;
    RatioNumerator   : LREAL;
    RatioDenominator : UINT;
    Acceleration     : LREAL;
    Deceleration     : LREAL;
    Jerk             : LREAL;
    BufferMode        : MC_BufferMode;
    Options          : ST_GearInOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
RatioNumerator	LREAL	Gear ratio numerator. Alternatively, the gear ratio can be specified in the numerator as a floating point value, if the denominator is 1.
RatioDenominator	UINT	Gear ratio denominator
Acceleration	LREAL	Acceleration (≥0). (Currently not implemented)
Deceleration	LREAL	Deceleration (≥0). (Currently not implemented)
Jerk	LREAL	Jerk (≥0). (Currently not implemented)
BufferMode	MC_BufferMode	Currently not implemented
Options	ST_GearInOptions	Currently not implemented

For a 1:4 ratio the RatioNumerator must be 1, the RatioDenominator must be 4. Alternatively, the RatioDenominator may be 1, and the gear ratio can be specified as floating point number 0.25 under RatioNumerator. The RatioNumerator may be negative.

 **Inputs/outputs**

```
VAR_IN_OUT
  Master : AXIS_REF;
  Slave  : AXIS_REF;
END_VAR
```

Name	Type	Description
Master	AXIS_REF	Axis data structure of the master
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF \[118\]](#) addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

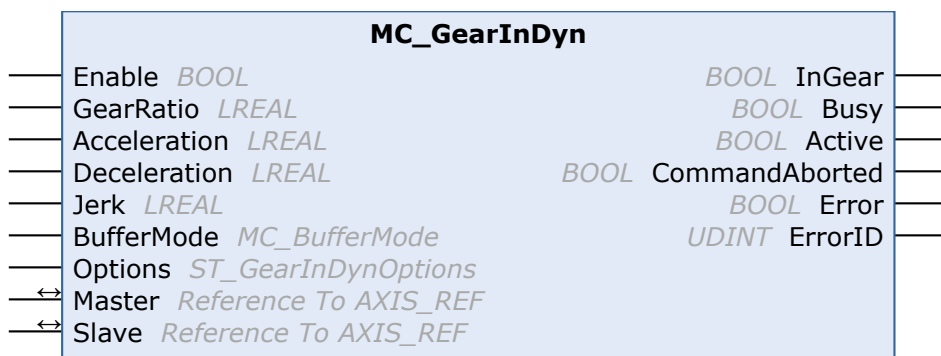
```
VAR_OUTPUT
  InGear      : BOOL;
  Busy        : BOOL;
  Active       : BOOL;
  CommandAborted : BOOL;
  Error        : BOOL;
  ErrorID      : UDINT;
END_VAR
```

Name	Type	Description
InGear	BOOL	TRUE if the coupling was successful.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "InGear", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. (Currently Active = Busy)
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.5.2 MC\_GearInDyn**



The function block MC\_GearInDyn activates a linear master-slave coupling (gear coupling). The gear ratio can be adjusted dynamically during each PLC cycle. Hence a controlled master/slave coupling can be build up. The "Acceleration" parameter has a limiting effect in situations with large gear ratio variations.

The slave axis can be decoupled with the function block [MC\\_GearOut \[► 104\]](#). If the slave is decoupled while it is moving, then it retains its velocity and can be halted using [MC\\_Stop \[► 87\]](#) or [MC\\_Halt \[► 85\]](#).

Alternatively, the function block [MC\\_GearIn \[► 101\]](#) with fixed gear ratio is available.

 **Inputs**

```
VAR_INPUT
  Enable      : BOOL;
  GearRatio   : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  BufferMode   : MC_BufferMode;
  Options     : ST_GearInDynOptions;
END_VAR
```

Name	Type	Description
Enable	BOOL	If "Enable" is TRUE, the function block establishes a coupling.  If an MC_GearOut is executed while the MC_GearInDyn continues to be called with "Enable" TRUE, it is only decoupled for a short time and then immediately attempts to re-establish the coupling.  As long as "Enable" is TRUE, the gear factor can be changed cyclically. If "Enable" becomes FALSE after coupling, the command is terminated. The gear factor is frozen at its last value, but the slave is not decoupled.
GearRatio	LREAL	Gear ratio as floating point value. The gear ratio can be changed cyclically as long as "Enable" is TRUE. If "Enable" is FALSE, the gear ratio remains unchanged.
Acceleration	LREAL	Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used. The parameter limits the acceleration of the slave in situations with large gear ratio variations. The maximum acceleration is only reached at the maximum master velocity. Otherwise the slave acceleration is below this value when the gear ratio changes significantly.
Deceleration	LREAL	Deceleration (≥0). (Not implemented)
Jerk	LREAL	Jerk (≥0). (Not implemented)
BufferMode	MC_BufferMode	Currently not implemented
Options	ST_GearInDynOptions	Currently not implemented

 **Inputs/outputs**

```
VAR_IN_OUT
  Master : AXIS_REF;
  Slave  : AXIS_REF;
END_VAR
```

Name	Type	Description
Master	AXIS_REF	Axis data structure of the master
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF \[► 118\]](#) addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**🔌 Outputs**

```

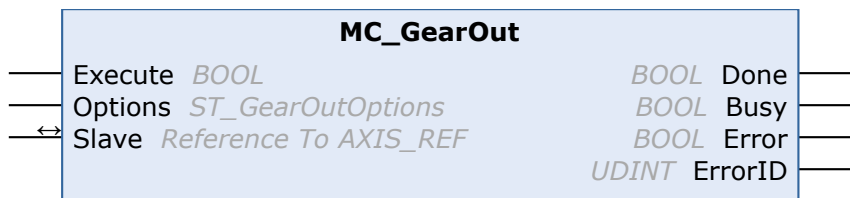
VAR_OUTPUT
  InGear      : BOOL;
  Busy        : BOOL;
  Active      : BOOL;
  CommandAborted : BOOL;
  Error       : BOOL;
  ErrorID     : UDINT;
END_VAR
    
```

Name	Type	Description
InGear	BOOL	TRUE if the coupling was successful.
Busy	BOOL	TRUE as soon as the command is started and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "InGear", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. (Currently Active = Busy)
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**6.5.3 MC\_GearOut**



The function block MC\_GearOut disables a master-slave coupling.

**⚠ WARNING**

**No standstill of the axis due to decoupling**

When a slave axis is uncoupled during the movement, it is not stopped automatically but reaches a constant velocity at which it continues to travel infinitely.

You can stop the axis with the function blocks [MC\\_Halt \[▶ 85\]](#) or [MC\\_Stop \[▶ 87\]](#).

**● Setpoint generator type**

**i** If the setpoint generator type of the axis is set to "7 phases (optimized)", the slave axis assumes an acceleration-free state after decoupling and continues to move with the resulting constant velocity. There is no positioning based on the master travel path calculated with the coupling factor. Instead, the behavior matches the behavior after a MC\_MoveVelocity command. In TwinCAT 2.10, the setpoint generator type can be selected by the user. From TwinCAT 2.11, the setpoint generator type is set to "7 phases (optimized)". The behavior described here is the result of a project update from TwinCAT 2.10 to TwinCAT 2.11. Depending on the circumstances, an update of existing applications to version 2.11 may necessitate an adaptation of the PLC program.



 **Inputs**

```
VAR_INPUT
    Execute : BOOL;
    Options : ST_GearOutOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Options	ST_GearOutOptions	Currently not implemented

 **Inputs/outputs**

```
VAR_IN_OUT
    Slave : AXIS_REF;
END_VAR
```

Name	Type	Description
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF](#) [▶ 118] addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

 **Outputs**

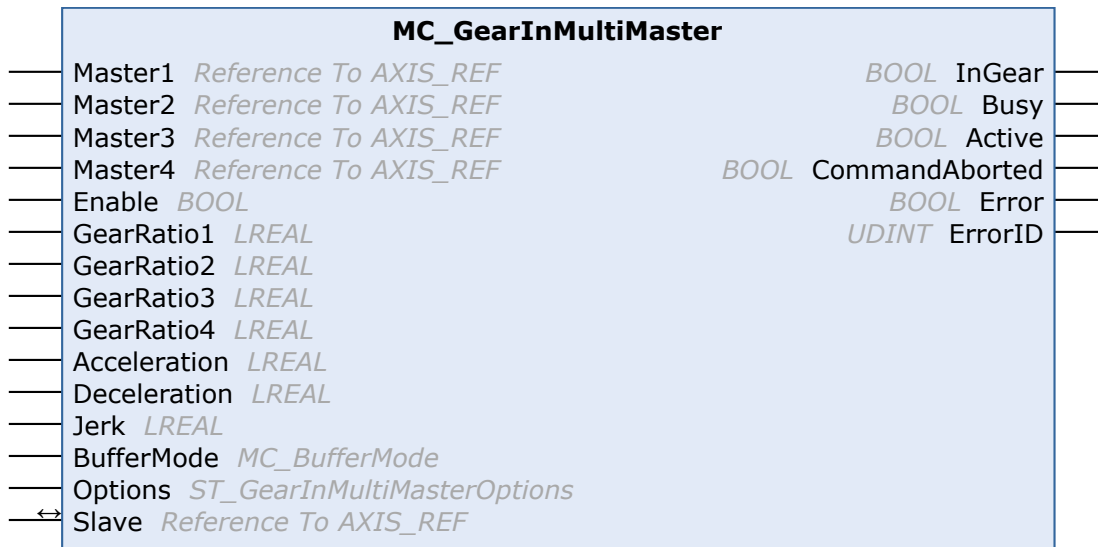
```
VAR_OUTPUT
    Done : BOOL;
    Busy : BOOL;
    Error : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if the axis has been successfully decoupled.
Busy	BOOL	TRUE as soon as the command is started and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done" or "Error" is set.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 6.5.4 MC\_GearInMultiMaster



The function block MC\_GearInMultiMaster is used to activate linear master/slave coupling (gear coupling) for up to four different master axes. The gear ratio can be adjusted dynamically during each PLC cycle. The slave movement is determined by the superimposed master movements. The "Acceleration" parameter has a limiting effect in situations with large gear ratio variations.

The slave axis can be decoupled with the function block MC\_GearOut [▶ 104]. If the slave is decoupled while it is moving, then it retains its velocity and can be halted using MC\_Stop [▶ 87].

If less than four masters are used, an empty data structure can be transferred for each of the parameters "Master2" to "Master4" (the axis ID must be 0).

#### Inputs

```

VAR_INPUT
  Master1      : Reference To AXIS_REF;
  Master2      : Reference To AXIS_REF;
  Master3      : Reference To AXIS_REF;
  Master4      : Reference To AXIS_REF;
  Enable       : BOOL;
  GearRatio1   : LREAL;
  GearRatio2   : LREAL;
  GearRatio3   : LREAL;
  GearRatio4   : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk         : LREAL;
  BufferMode    : MC_BufferMode;
  Options      : ST_GearInMultiMasterOptions;
END_VAR
    
```

Name	Type	Description
Master1	Reference To <u>AXIS_REF</u> [▶ 118]	Axis data structure of the first master
Master2	Reference To <u>AXIS_REF</u> [▶ 118]	Axis data structure of the second master
Master3	Reference To <u>AXIS_REF</u> [▶ 118]	Axis data structure of the third master
Master4	Reference To <u>AXIS_REF</u> [▶ 118]	Axis data structure of the fourth master
Enable	BOOL	If "Enable" is TRUE, the function block establishes a coupling.

Name	Type	Description
		<p>If an MC_GearOut is executed while the MC_GearInMultiMaster continues to be called with "Enable" TRUE, it is only decoupled briefly and then immediately attempts to re-establish the coupling.</p> <p>As long as "Enable" is TRUE, the gear factors can be changed cyclically. If "Enable" becomes FALSE after coupling, the command is terminated. The gear factors are frozen at their last values, but the slave is not decoupled.</p>
GearRatio1	LREAL	Gear ratio as floating point value for the first master axis. The gear ratio can be changed cyclically as long as "Enable" is TRUE. If "Enable" is FALSE, the gear ratio remains unchanged.
GearRatio2	LREAL	Gear ratio as floating point value for the second master axis. The gear ratio can be changed cyclically as long as "Enable" is TRUE. If "Enable" is FALSE, the gear ratio remains unchanged.
GearRatio3	LREAL	Gear ratio as floating point value for the third master axis. The gear ratio can be changed cyclically as long as "Enable" is TRUE. If "Enable" is FALSE, the gear ratio remains unchanged.
GearRatio4	LREAL	Gear ratio as floating point value for the fourth master axis. The gear ratio can be changed cyclically as long as "Enable" is TRUE. If "Enable" is FALSE, the gear ratio remains unchanged.
Acceleration	LREAL	Acceleration ( $\geq 0$ ). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used. The parameter limits the acceleration of the slave in situations with large gear ratio variations.
Deceleration	LREAL	Deceleration ( $\geq 0$ ). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used. The parameter limits the deceleration of the slave in situations with large gear ratio variations. <b>Used only for the option "AdvancedSlaveDynamics".</b>
Jerk	LREAL	Jerk ( $\geq 0$ ). If the value is 0, the standard jerk from the axis configuration in the System Manager is used. The parameter limits the jerk of the slave in situations with large gear ratio variations. <b>Used only for the option "AdvancedSlaveDynamics".</b>
BufferMode	MC_BufferMode	Currently not implemented
Options	<a href="#">ST_GearInMultiMasterOptions</a> <a href="#">▶ 128</a>	<ul style="list-style-type: none"> <li>• <b>AdvancedSlaveDynamics:</b> swaps the internal algorithm of the function block. This makes it possible to synchronize to masters already in motion. In this case, "Acceleration" and "Deceleration" should only be parameterized symmetrically. If jerk presets are too large, this is reduced to the extent that a change from zero to the parameterized acceleration / deceleration can take place in one NC cycle. The resolution of the acceleration / deceleration thus depends directly on the suitable parameterization of the jerk value.</li> <li>• <b>SyncMode (available from TC3.1.4024.11):</b> the SyncMode specifies how the axis coupling behaves if the movement of the slave axis is limited in the AdvancedSlaveDynamics mode (limitation of velocity, acceleration and jerk). The following parameters can be set:</li> </ul>

Name	Type	Description
		<ul style="list-style-type: none"> <li>◦ VELOSYNC: The slave axis and the master axes move with a synchronized velocity. In the case of a dynamic limitation, the slave establishes a position difference that is not caught up again.</li> <li>◦ POSSYNC1: The slave axis moves to the master axes with synchronized velocity and position. In the case of a dynamic limitation, the slave establishes a position difference that is then caught up at a later time. Changes in the gear ratios during the journey are executed taking into account the dynamics. The position difference between master and slave resulting from this is not caught up.</li> <li>◦ POSSYNC2: The slave axis moves to the master axes with synchronized velocity and position. In the case of a dynamic limitation, the slave establishes a position difference that is then caught up at a later time. Changes in the gear ratios during the journey are immediately calculated internally and the slave continuously synchronizes itself to the new position.</li> </ul>

**Overview:** Which user command results in a permanent position difference or catches it up depending on the SyncMode?

		User commands		
		Coupling during the journey	Change Gearing factor	Dynamic limitation (Velo, Acc, Jerk)
SyncMode	VELOSYNC (default)	Permanent position difference	Permanent position difference	Permanent position difference
	POSSYNC1	Permanent position difference	Permanent position difference	Catch up position difference
	POSSYNC2	Catch up position difference	Catch up position difference	Catch up position difference



If coupling takes place with a constant gear ratio with the axes at a standstill (and the gear ratio no longer changes during the course of coupling), then the two SyncModes POSSYNC1 and POSSYNC2 behave identically.

**Inputs/outputs**

```
VAR_IN_OUT
  Slave : AXIS_REF;
END_VAR
```

Name	Type	Description
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF](#) [118] addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

**🚀 Outputs**

```
VAR_OUTPUT
  InGear      : BOOL;
  Busy        : BOOL;
  Active      : BOOL;
  CommandAborted : BOOL;
  Error       : BOOL;
  ErrorID     : UDINT;
END_VAR
```

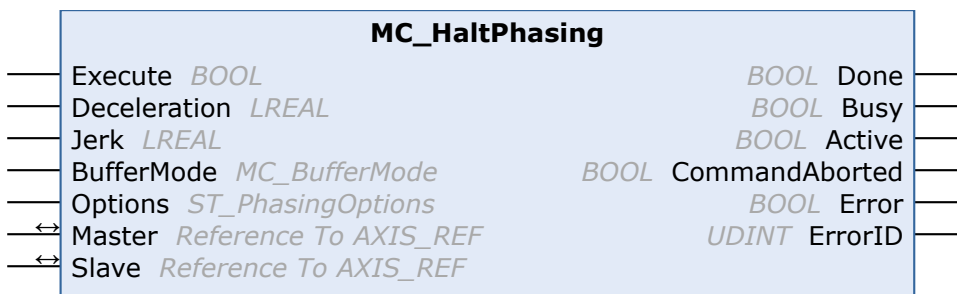
Name	Type	Description
InGear	BOOL	TRUE if the coupling was successful.
Busy	BOOL	TRUE as soon as the command is started with "Enable" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "InGear", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. (Currently Active = Busy)
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis may have become decoupled during the coupling process (simultaneous command execution).
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 6.6 Phasing

### 6.6.1 MC\_HaltPhasing



The function block MC\_HaltPhasing is used to bring about a controlled stop of the phase shift of a slave axis relative to the master axis. The "Halt" is always jerk-limited, based on the constant jerk value for the braking delay set in the "Jerk" parameter. MC\_HaltPhasing terminates a superimposed movement through MC\_PhasingAbsolute or MC\_PhasingRelative.

**🚀 Inputs**

```
VAR_INPUT
  Execute      : BOOL;
  Deceleration : LREAL;
  Jerk         : LREAL;
  BufferMode    : MC_BufferMode;
  Options      : ST_PhasingOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Deceleration	LREAL	Maximum deceleration value
Jerk	LREAL	Maximum jerk value
BufferMode	MC_BufferMode	MC_Aborting only
Options	ST_PhasingOptions	Not implemented

### Inputs/outputs

```
VAR_IN_OUT
  Master : AXIS_REF;
  Slave  : AXIS_REF;
END_VAR
```

Name	Type	Description
Master	AXIS_REF	Axis data structure of the master
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF](#) [► 118] addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

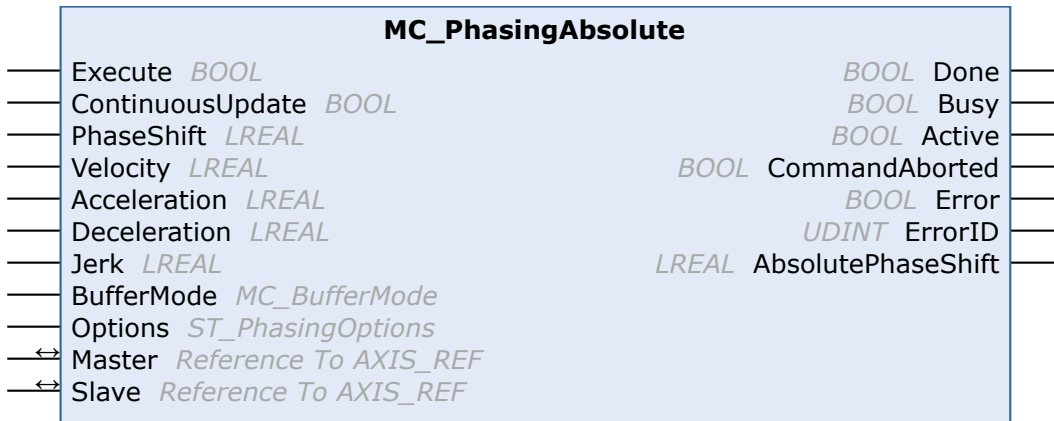
```
VAR_OUTPUT
  Done       : BOOL;
  Busy       : BOOL;
  Active     : BOOL;
  CommandAborted : BOOL;
  Error      : BOOL;
  ErrorId    : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE if velocity = 0 is reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 6.6.2 MC\_PhasingAbsolute



The function block MC\_PhasingAbsolute can be used to set a phase shift between a master axis and a slave axis. The function block executes a superimposed movement of the slave axis and thus sets a position difference "PhaseShift" between master and slave.

The dynamic values "velocity", "acceleration" and "deceleration" refer to the superimposed movement with which the phase shift is carried out. The movement is always jerk-limited, based on the constant jerk value set in the Jerk parameter. This value applies both to "Acceleration" and "Deceleration".

The phase shift can be used for a simple MC\_GearIn [▶\_101] coupling, as well as for a coupling with dynamic coupling factor. In the latter case, note that MC\_GearInMultimaster [▶\_106] is used with Options.AdvancedSlaveDynamics = TRUE (gearing) or MC\_CamIn\_V2 (camming).

MC\_GearInDyn [▶\_102] is not supported.

#### Inputs

```

VAR_INPUT
    Execute           : BOOL;
    ContinuousUpdate : BOOL;
    PhaseShift        : LREAL;
    Velocity           : LREAL;
    Acceleration       : LREAL;
    Deceleration       : LREAL;
    Jerk               : LREAL;
    BufferMode          : MC_BufferMode;
    Options            : ST_PhasingOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ContinuousUpdate	BOOL	If this input is TRUE at the rising edge at the "Execute" input, the "PhaseShift", "Velocity", "Acceleration", "Deceleration" and "Jerk" inputs can be changed during the execution of the command and made effective as quickly as possible.
PhaseShift	LREAL	Phase shift to be set between master and slave axis
Velocity	LREAL	Maximum velocity that may be reached during the phase shift (>=0.01).
Acceleration	LREAL	Maximum acceleration value
Deceleration	LREAL	Maximum deceleration value
Jerk	LREAL	Maximum jerk value
BufferMode	MC_BufferMode	MC_Aborting only
Options	ST_PhasingOptions	Not implemented

### Inputs/outputs

```
VAR_IN_OUT
  Master : AXIS_REF;
  Slave  : AXIS_REF;
END_VAR
```

Name	Type	Description
Master	AXIS_REF	Axis data structure of the master
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF \[► 118\]](#) addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorId        : UDINT;
  AbsolutePhaseShift : LREAL;
END_VAR
```

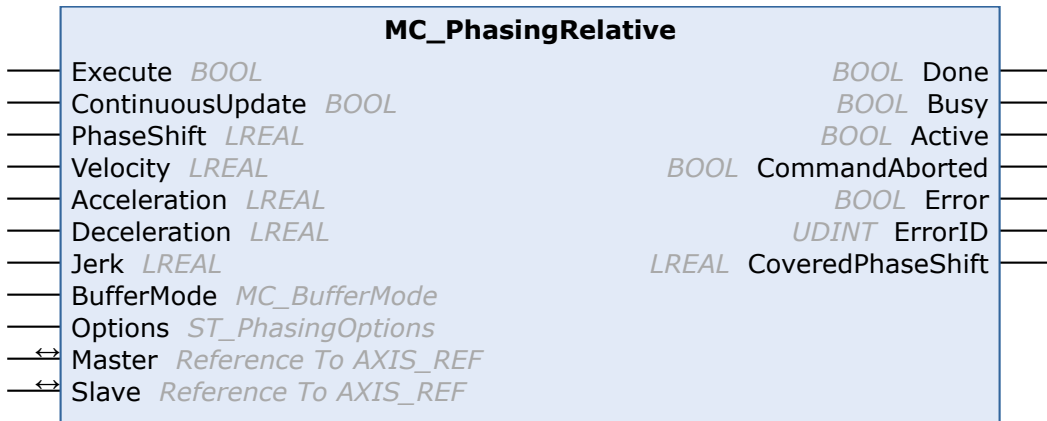
Name	Type	Description
Done	BOOL	TRUE when the absolute phase shift is established.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the phase shift occurs. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
AbsolutePhaseShift	LREAL	Absolute phase shift

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2



### 6.6.3 MC\_PhasingRelative



The function block MC\_PhasingRelative can be used to set a phase shift between a master and a slave axis. The function block executes a superimposed movement of the slave axis, thereby changing the position difference between master and slave by the distance "PhaseShift".

The dynamic values "velocity", "acceleration" and "deceleration" refer to the superimposed movement with which the phase shift is carried out. The movement is always jerk-limited, based on the constant jerk value set in the Jerk parameter. This value applies both to "Acceleration" and "Deceleration".

The phase shift can be used for a simple MC\_GearIn [▶\_101] coupling, as well as for a coupling with dynamic coupling factor. In the latter case, note that MC\_GearInMultimaster [▶\_106] is used with Options.AdvancedSlaveDynamics = TRUE (gearing) or MC\_CamIn\_V2 (camming).

MC\_GearInDyn [▶\_102] is not supported.

#### Inputs

```

VAR_INPUT
    Execute           : BOOL;
    ContinuousUpdate : BOOL;
    PhaseShift        : LREAL;
    Velocity           : LREAL;
    Acceleration       : LREAL;
    Deceleration       : LREAL;
    Jerk               : LREAL;
    BufferMode          : MC_BufferMode;
    Options            : ST_PhasingOptions;
END_VAR
    
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ContinuousUpdate	BOOL	If this input is TRUE at the rising edge at the Execute input, the "PhaseShift", "Velocity", "Acceleration", "Deceleration" and "Jerk" inputs can be changed during the execution of the command and made effective as quickly as possible.
PhaseShift	LREAL	Amount by which the phase shift between master and slave axis is changed.
Velocity	LREAL	Maximum velocity that may be reached during the phase shift (>=0.01).
Acceleration	LREAL	Maximum acceleration value
Deceleration	LREAL	Maximum deceleration value
Jerk	LREAL	Maximum jerk value
BufferMode	MC_BufferMode	MC_Aborting only
Options	ST_PhasingOptions	Not implemented

### Inputs/outputs

```
VAR_IN_OUT
  Master : AXIS_REF;
  Slave  : AXIS_REF;
END_VAR
```

Name	Type	Description
Master	AXIS_REF	Axis data structure of the master
Slave	AXIS_REF	Axis data structure of the slave

The axis data structure of type [AXIS\\_REF \[► 118\]](#) addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  Active         : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorId        : UDINT;
  CoveredPhaseShift : LREAL;
END_VAR
```

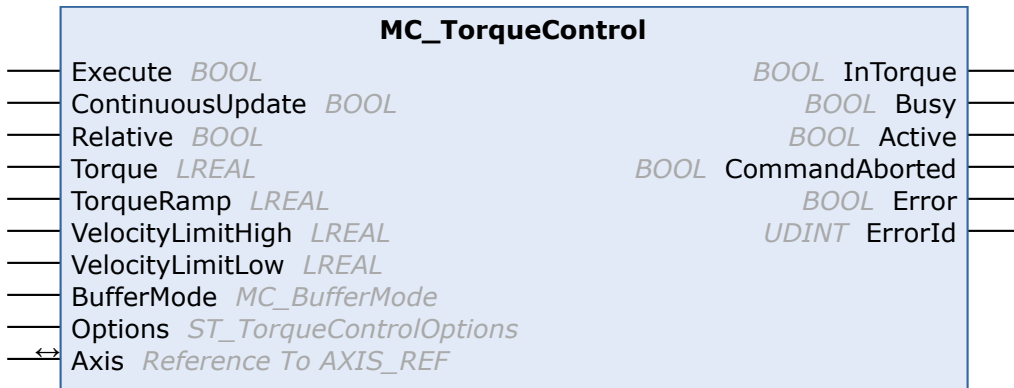
Name	Type	Description
Done	BOOL	TRUE when the absolute phase shift is established.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the phase shift occurs. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.
CommandAborted	BOOL	TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
CoveredPhaseShift	LREAL	Absolute phase shift

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 6.7 Torque Control

### 6.7.1 MC\_TorqueControl



The MC\_TorqueControl function block switches an axis under NC control to the "Cyclic Synchronous Torque Mode" (CST) and sets a torque setpoint for it. In this operation mode, the NC also provides the parameterized "VelocityLimitHigh" & "VelocityLimitLow" as linkable objects to the drive controller (Axis->Drive->Outputs->nDataOut5/nDataOut6) in the cyclic interface.

To change the operation mode without jerking, the dead time compensation must be activated in the NC axis.

#### ⚠ DANGER

#### Danger to life or risk of serious injury or damage to property due to unintentional movements of the axis

When using the function block, the axis is switched to CST mode. After using the function block (especially after error situations), the axis may still be in CST mode. This can lead to sudden and unplanned movements (especially with lifting axes) when the axis is released.

- Ensure that there is no hazard as defined by the risk assessment.
- Check the current operation mode via the function block [MC\\_ReadDriveOperationMode \[▶ 45\]](#).
- If the axis is not in a position-related operation mode (CSV/CSP), transfer it before an enable:
  - *directly* with [MC\\_WriteDriveOperationMode \[▶ 46\]](#) into the desired position-related operation mode (CSV/CSP) or
  - *indirectly* with [MC\\_Halt \[▶ 85\]](#) / [MC\\_Stop \[▶ 87\]](#) into the desired position-related operation mode (CSV/CSP) (from TwinCAT 3.1.4024.40)

Other function blocks that switch the axis indirectly into a position-related operation mode can only do this to a limited extent and are therefore not to be used for a deliberate operation mode change.

⇒ Subsequently, it is necessary to check again whether the axis is really in a position-related operation mode (CSV/CSP), if not, an abort with error handling is required.

In order for this module to be used sensibly, the drive controller must support switching of the operating modes on the fly, as well as speed limitation in torque mode.

The relevant parameters are available for scaling the actual and setpoint torque in the Drive area in the NC axis parameters.

Notes on parameterization for NC-guided position- and torque-based mixed operation are in preparation and will be published soon.

#### Inputs

```

VAR_INPUT
    Execute           : BOOL;
    ContinuousUpdate : BOOL;
    Relative          : BOOL;
    Torque            : LREAL;
    TorqueRamp       : LREAL;
    
```

```

VelocityLimitHigh : LREAL;
VelocityLimitLow  : LREAL;
BufferMode        : MC_BufferMode;
Options           : ST_TorqueControlOptions;
END_VAR

```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
ContinuousUpdate	BOOL	If this input is TRUE on the rising edge at the "Execute" input, the "Torque", "TorqueRamp", "VelocityLimitHigh" and "VelocityLimitLow" inputs can be changed during the execution of the command and made effective as quickly as possible.
Relative	BOOL	If this input is TRUE, the torque value is changed relatively by the specified value "Torque".
Torque	LREAL	Torque setpoint with which the drive is operated ("Relative" = FALSE) or by which the torque is to be changed ("Relative" = TRUE). (e.g. %)  Here, a positive torque setting means torque in a logically positive direction of movement.
TorqueRamp	LREAL	Rate of change of the torque setpoint ( e.g. %/s)
VelocityLimitHigh	LREAL	Upper velocity limit for limitation in CST mode (e.g. mm/s). This limitation must be configured accordingly in the process image.
VelocityLimitLow	LREAL	Lower velocity limit for limitation in CST mode (e.g. mm/s). This limitation must be configured accordingly in the process image.
BufferMode	<a href="#">MC_BufferMode [► 132]</a>	Currently only "Aborting" is supported.
Options	<a href="#">ST_TorqueControlOptions [► 153]</a>	Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: [General rules for MC function blocks \[► 14\]](#)

### Inputs/outputs

```

VAR_IN_OUT
  Axis : AXIS_REF;
END_VAR

```

Name	Type	Description
Axis	<a href="#">AXIS_REF [► 118]</a>	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

### Outputs

```

VAR_OUTPUT
  InTorque      : BOOL;
  Busy          : BOOL;
  Active        : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorID       : UDINT;
END_VAR

```

Name	Type	Description
InTorque	BOOL	TRUE, if the axis has built up the set torque and no velocity limitation is active.
Busy	BOOL	TRUE as soon as the command is started with Execute and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order.

Name	Type	Description
Active	BOOL	Indicates that the function block controls the axis.
CommandAborted	BOOL	TRUE if the command could not be executed completely.
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

See also: [General rules for MC function blocks \[► 14\]](#)

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86)	Tc2_MC2

## 7 Data types

### 7.1 Axis interface

#### 7.1.1 ST\_AdsAddress

ADS data structure used in the [AXIS\\_REF](#) [► 118] and containing the ADS communication parameters of an axis needed for direct ADS communication. Normally this structure does not have to be populated. The user can use it to stored information for controlling an axis on another target system or via a special port number.

```
TYPE ST_AdsAddress
STRUCT
  NetId   : STRING(23);
  Port    : UINT;
  Channel : UINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
NetId	STRING(23)	AmsNetId of the target system
Port	UINT	Port number
Channel	UINT	Channel number

#### 7.1.2 AXIS\_REF

The [AXIS\\_REF](#) data type contains axis information. [AXIS\\_REF](#) is an interface between the PLC and the NC. It is added to MC function blocks as axis reference.

```
TYPE AXIS_REF :
VAR_INPUT
  PlcToNc AT %Q* : PLCTONC_AXIS_REF;
END_VAR
VAR_OUTPUT
  NcToPlc AT %I* : NCTOPLC_AXIS_REF;
  ADS         : ST_AdsAddress;
  Status      : ST_AxisStatus;
  DriveAddress : ST_DriveAddress;
END_VAR
END_TYPE
```

AXIS_REF elements	Description
PlcToNc	Data structure which is cyclically exchanged between PLC and NC. Via this data structure the MC function blocks communicate with the NC and send control information from the PLC to the NC. This data structure is automatically placed in the output process image of the PLC and must be linked in TwinCAT System Manager with the input process image of an NC axis. (Type: <a href="#">PLCTONC_AXIS_REF</a> [► 125])
NcToPlc	Data structure which is cyclically exchanged between PLC and NC. Via this data structure the MC function blocks communicate with the NC and receive status information from the NC. This data structure is automatically placed in the input process image of the PLC and must be linked in TwinCAT System Manager with the output process image of an NC axis. The <a href="#">NCTOPLC</a> structure contains all main state data for an axis such as position, velocity and instruction state. Since data exchange takes place cyclically, the PLC can access the current axis state at any time without additional communication effort. (Type: <a href="#">NCTOPLC_AXIS_REF</a> [► 119])
ADS	The ADS data structure contains the ADS communication parameters for an axis that are required for direct ADS communication. Normally this structure does not have to be populated. The user can use it to stored information for controlling an axis on another target system or via a special port number.

AXIS_REF elements	Description
Status	Data structure containing additional or processed status information for an axis (type: <code>ST_AxisStatus</code> [▶ 146]). This data structure is not refreshed cyclically, but has to be updated through the PLC program. For this purpose <code>MC_ReadStatus</code> [▶ 29] or alternatively the action "ReadStatus" of <code>AXIS_REF</code> can be called:
DriveAddress	Data structure containing the ADS access data of a drive device. These data are not filled until the function block <code>MC_ReadDriveAddress</code> [▶ 54] has been called implicitly or explicitly.

Sample:

```

VAR
  Axis1 : AXIS_REF (* axis data structure for Axis-1 *)
END_VAR

(* program code at the beginning of each PLC cycle *)
Axis1.ReadStatus();

(* alternative program code at the beginning of each PLC cycle *)
Axis1();

```

The call of "ReadStatus" or "Axis1" should be made once at the beginning of each PLC cycle. The current status information can then be accessed in `AXIS_REF` from the whole PLC program. Within a cycle the status does not change.

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**7.1.3 NCTOPLC\_AXIS\_REF**

The data structure `NCTOPLC_AXIS_REF` is part of the `AXIS_REF` [▶ 118] data structure and is automatically updated by the NC, so that updated information is available during each PLC cycle. `NCTOPLC_AXIS_REF` is also referred to as axis interface between NC and PLC.

```

TYPE NCTOPLC_AXIS_REF
STRUCT
  StateDWord          : NCTOPLC_AXIS_REF_STATE; (* Status double word *)
  ErrorCode           : DWORD; (* Axis error code *)
  AxisState           : DWORD; (* Axis moving status *)
  AxisModeConfirmation : DWORD; (* Axis mode confirmation (feedback from NC) *)
  HomingState         : DWORD; (* State of axis calibration (homing) *)
  CoupleState         : DWORD; (* Axis coupling state *)
  SvbEntries          : DWORD; (* SVB entries/orders (SVB = Set preparation task) *)
  SafEntries          : DWORD; (* SAF entries/orders (SAF = Set execution task) *)
  AxisId              : DWORD; (* Axis ID *)
  OpModeDWord         : NCTPPLC_AXIS_REF_OPMODE; (* Current operation mode *)
  ActPos              : LREAL; (* Actual position (absolut value from NC) *)
  ModuloActPos        : LREAL; (* Actual modulo position *)
  ActiveControlLoopIndex : WORD; (* Active control loop index *)
  ControlLoopIndex    : WORD; (* Axis control loop index (0, 1, 2, when multiple control
loops are used) *)
  ModuloActTurns      : DINT; (* Actual modulo turns *)
  ActVelo             : LREAL; (* Actual velocity *)
  PosDiff             : LREAL; (* Position difference (lag distance) *)
  SetPos              : LREAL; (* Setpoint position *)
  SetVelo             : LREAL; (* Setpoint velocity *)
  SetAcc              : LREAL; (* Setpoint acceleration *)
  TargetPos           : LREAL; (* Estimated target position *)
  ModuloSetPos        : LREAL; (* Setpoint modulo position *)
  ModuloSetTurns      : DINT; (* Setpoint modulo turns *)
  CmdNo               : WORD; (* Continuous actual command number *)
  CmdState            : WORD; (* Command state *)
  SetJerk             : LREAL;
  SetTorque           : LREAL;
  ActTorque           : LREAL;
  StateDWord2         : NCTOPLC_AXIS_REF_STATE2;
  StateDWord3         : DWORD;
  TouchProbeState     : DWORD;
  TouchProbeCounter   : DWORD;

```

```

CamCouplingState      : ARRAY [0..7] OF NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE;
CamCouplingTableID    : ARRAY [0..7] OF UINT;
ActTorqueDerivative    : LREAL;
SetTorqueDerivative    : LREAL;
AbsPhasingPos         : LREAL;
TorqueOffset          : LREAL;
ActPosWithoutPosCorrection : LREAL;
ActAcc                : LREAL;
DcTimeStamp           : UDINT;
{attribute 'hide'}
_reserved2             : ARRAY [1..4] OF USINT;
UserData              : LREAL;

```

END\_TYPE

Variable name	Data type	Definition range	Description
StateDWord	NCTOPLC_AXIS_REF_STATE [► 123]	-	State double word.
ErrorCode	DWORD	≥0	Axis error code
AxisState	DWORD	ENUM [► 121]	Present state of the axis movement
AxisModeConfirmation	DWORD	ENUM	Axis operating mode (feedback from the NC)
HomingState	DWORD	ENUM [► 121]	Reference status of the axis ("calibration status")
CoupleState	DWORD	ENUM [► 122]	Axis coupling state
SvbEntries	DWORD	≥0	SVB entries/tasks
SafEntries	DWORD	≥0	SAF entries/tasks (NC interpreter, FIFO group)
AxisId	DWORD	>0	axis ID
OpModeDWord.	NCTOPLC_AXIS_REF_OPMODE [► 122]	-	Axis operation mode double word
ActPos	LREAL	±∞	Actual position (calculated absolute value)
ModuloActPos	LREAL	≥0	Modulo actual position (calculated value in, for example, degrees)
ActiveControlLoopIndex	WORD	≥0	Active axis control loop index
ControlLoopIndex	WORD	≥0	Axis control loop index (0, 1, 2, etc. if more than one axis control loop is used)
ModuloActTurns	DINT	±∞	Modulo actual rotations
ActVelo	LREAL	±∞	Actual velocity (optional)
PosDiff	LREAL	±∞	Lag error (position)
SetPos	LREAL	±∞	Set position (calculated absolute value)
SetVelo	LREAL	±∞	Set velocity
SetAcc	LREAL	±∞	Set acceleration
TargetPos	LREAL	±∞	Estimated target position of the axis
ModuloSetPos	LREAL	≥0	Modulo set position (calculated value in, for example, degrees)
ModuloSetTurns	DINT	≥0	Modulo set rotations
CmdNo	WORD	≥0	Command number of the active axis job (see BufferMode)
CmdState	WORD	≥0	Command status information (see BufferMode)
SetJerk	LREAL		Set jerk
SetTorque	LREAL		Set torque
ActTorque	LREAL		Actual torque



Variable name	Data type	Definition range	Description
StateDWord2	NCTOPLC_AXIS_REF_STATE2 [▶ 124]		State double word 2
StateDWord3	DWORD		State double word 3
TouchProbeState	DWORD		TouchProbe status
TouchProbeCounter	DWORD		TouchProbe counter
CamCouplingState	ARRAY [0..7] OF NCTOPLC_AXIS_REF_CAMCOUPLING_STATE [▶ 125]		Cam coupling information for multitables (from TwinCAT 3.1.4020.0)
CamCouplingTableId	ARRAY [0..7] OF UINT		Cam coupling ID for multitables (from TwinCAT 3.1.4020.0)
ActTorqueDerivative	LREAL		First derivative of the actual torque
SetTorqueDerivative	LREAL		First derivative of the set torque
AbsPhasingPos	LREAL		Absolute phasing offset
TorqueOffset	LREAL		Additive torque part
ActPosWithoutPosCorrection	LREAL		Actual position without position correction
ActAcc	LREAL		Actual acceleration
DcTimeStamp	UDINT		Current NC timestamp
UserData	LREAL		Configurable axis state parameter

Define	Master: motion state / drive phase of the continuous master axis (servo) (AxisState)
0	Setpoint generator not active (INACTIVE)
1	Setpoint generator active (RUNNING)
2	Velocity override is zero (OVERRIDE_ZERO)
3	Constant velocity (PHASE_VELOCONST)
4	Acceleration phase (PHASE_ACCPOS)
5	Deceleration phase (PHASE_ACCNEG)
Define	Master: motion state / drive phase of the discrete master axis (fast/creep) (AxisState)
0	Setpoint generator not active
1	Moving phase (rapid or slow traverse)
2	Switchover delay from rapid to slow traverse
3	Creep motion (within the creep distance)
4	Deceleration time (starting from the braking distance in front of the target)
Define	Slave: motion state / drive phase of the continuous slave axis (servo) (AxisState)
0	Slave generator not active (INACTIVE)
11	Slave is in a movement pre-phase (PRE-PHASE)
12	Slave is synchronizing (SYNCHRONIZING)
13	Slave is synchronized and moves synchronously (SYNCHRON)

Define	Calibration state of the axis (HomingState)
0	Referencing process completed (READY)
1	Continuous start in the direction of the referencing cam. <b>If the cam is occupied at the beginning, then the program starts directly with calibration state 3.</b>
2	Wait for positive edge of the referencing cam and initiate axis stop
3	Wait until the axis is stopped (check whether cam is still occupied) and then endless start of the referencing cam in the direction of the sync pulse

Define	Calibration state of the axis (HomingState)
4	Wait for falling edge of the referencing cam
5	Activate latch, wait until latch has become valid and then initiate axis stop
6	If axis has stopped, then set actual position (actual position = reference position + braking distance)

See also function block description and remarks for [MC\\_Home](#) [► 96]

Define	Coupling state of the axis (CoupleState)
0	Single axis that is neither a master nor a slave (SINGLE)
1	Master axis with any number of slaves (MASTER)
2	Slave axis that is the master of another slave (MASTERSLAVE)
3	Just a slave axis (SLAVE)

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.1.4 NCTOPLC\_AXIS\_REF\_OPMODE

The structure NCTOPLC\_AXIS\_REF\_OPMODE is part of the structure [NCTOPLC\\_AXIS\\_REF](#) [► 119].

```
TYPE NCTOPLC_AXIS_REF_OPMODE :
    DWORD;
END_TYPE
```

The individual items of information are provided in the status structure of the AXIS\_REF at the following points:

Bit	Variable name	Description
0	Status.Opmode.PositionAreaMonitoring	Position range monitoring
1	Status.Opmode.TargetPositionMonitoring	Target position window monitoring
2	Status.Opmode.LoopMode	Loop movement
3	Status.Opmode.MotionMonitoring	Physical movement monitoring
4	Status.Opmode.PEHTimeMonitoring	PEH time monitoring
5	Status.Opmode.BacklashCompensation	Backlash compensation
6	Status.Opmode.DelayedErrorReaction	Delayed error reaction of the NC
7	Status.Opmode.Modulo	Modulo axis (modulo display)
8	Status.Opmode.SimulationAxis	Simulation axis
9-11		
12	Status.Opmode.StopMonitoring	Standstill monitoring
13-15		
16	Status.Opmode.PositionLagMonitoring	Lag monitoring - position
17	Status.Opmode.VeloLagMonitoring	Lag monitoring - velocity
18	Status.Opmode.SoftLimitMinMonitoring	End position monitoring min.
19	Status.Opmode.SoftLimitMaxMonitoring	End position monitoring max.
20	Status.Opmode.PositionCorrection	Position correction ("Measuring system error compensation")
21	Status.Opmode.AllowSlaveCommands	Allow motion commands to slave axes
22	Status.Opmode.AllowExtSetAxisCommands	Allow motion commands to an axis that is fed by an external setpoint generator
23	Status.NcApplicationRequest	Request bit for the application software (PLC code), e.g. for an "ApplicationHomingRequest"
24-31	Status.NcCycleCounter	NC cycle counter

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

**7.1.5 NCTOPLC\_AXIS\_REF\_STATE**

The structure NCTOPLC\_AXIS\_REF\_STATE is part of the structure [NCTOPLC\\_AXIS\\_REF](#) [▶ 119].

```
TYPE NCTOPLC_AXIS_REF_STATE :
    DWORD;
END_TYPE
```

The individual items of information are provided in the status structure of the AXIS\_REF at the following points:

Bit	Variable name	Description
0	Status.Operational	Axis is ready for operation
1	Status.Homed	Axis is referenced ("axis calibrated")
2	Status.NotMoving	Axis is logically stationary ("Axis not moving")
3	Status.InPositionArea	Axis is in position window (physical feedback)
4	Status.InTargetPosition	Axis is at target position (PEH) (physical feedback)
5	Status.Protected	Axis in protected operation mode (e.g. slave axis)
6	Status.ErrorPropagationDelayed	Axis signals a preliminary error warning (from TC 2.11)
7	Status.HasBeenStopped	Axis has been stopped or is presently executing a stop
8	Status.HasJob	Axis has job, is executing job
9	Status.PositiveDirection	Axis moving to logically larger values
10	Status.NegativeDirection	Axis moving to logically smaller values
11	Status.HomingBusy	Axis is referencing ("axis is being calibrated")
12	Status.ConstantVelocity	Axis has reached its constant velocity or rotary speed
13	Status.Compensating	Section compensation passive[0]/active[1] (see MC_MoveSuperImposed)
14	Status.ExtSetPointGenEnabled	Enable external setpoint generation
15		Operation mode not yet executed (Busy). Not yet released!
16	Status.ExternalLatchValid	External latch value or measuring probe has become valid
17	Status.NewTargetPos	Axis has received a new end position or a new velocity
18		Axis not in target position or cannot/will not reach it (e.g. axis stop). Not yet released!
19	Status.ContinuousMotion	Axis executing endless positioning task
20	Status.ControlLoopClosed	Axis ready to operate and axis control loop closed (e.g. position control)
21	Status.CamTableQueued	New table ready for "Online Change" and waiting for activation
22	Status.CamDataQueued	Table data (MF) ready for "Online Change" and waiting for activation
23	CamScalingPending	Table scalings ready for "Online Change" and waiting for activation
24	Status.CmdBuffered	Follow-up command is available in the command buffer (see BufferMode) (from TwinCAT V2.10 Build 1311)
25	Status.PTPmode	Axis in PTP operating mode (no slave, no NCI axis, no FIFO axis) (from TC 2.10 Build 1326)
26	Status.SoftLimitMinExceeded	Software minimum end position is active/occupied (from TC 2.10 Build 1327)
27	Status.SoftLimitMaxExceeded	Software maximum end position is active/occupied (from TC 2.10 Build 1327)

Bit	Variable name	Description
28	Status.DriveDeviceError	Drive hardware has an error (no warning); interpretation possible only if drive is in I/O data exchange. e.g. EtherCAT "OP" state (from TC 2.10 Build 1326)
29	Status.MotionCommandsLocked	Axis is blocked for motion commands (TcMc2)
30	Status.IoDataInvalid	IO data invalid (e.g. "WcState" or "CdlState" of the fieldbus)
31	Error	Axis is in an error state

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.1.6 NCTOPLC\_AXIS\_REF\_STATE2

The structure `NCTOPLC_AXIS_REF_STATE2` is part of the structure `NCTOPLC_AXIS_REF` [► 119].

```

TYPE NCTOPLC_AXIS_REF_STATE2 :
UNION
    Value      : DWORD;
    Flags      : NCTOPLC_AXIS_REF_STATE2_FLAGS;
END_TYPE

```

See also: `NCTOPLC_AXIS_REF_STATE2_FLAGS` [► 124]

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.1.7 NCTOPLC\_AXIS\_REF\_STATE2\_FLAGS

The structure `NCTOPLC_AXIS_REF_STATE2_FLAGS` is part of the structure `NCTOPLC_AXIS_REF_STATE2` [► 124].

```

TYPE NCTOPLC_AXIS_REF_STATE2_FLAGS :
STRUCT
    AvoidingCollision      : BIT;
    {attribute 'hide'}
    _reserved1             : BIT;
    {attribute 'hide'}
    _reserved2             : BIT;
    {attribute 'hide'}
    _reserved3             : BIT;
    {attribute 'hide'}
    _reserved4             : BIT;
    {attribute 'hide'}
    _reserved5             : BIT;
    {attribute 'hide'}
    _reserved6             : BIT;
    {attribute 'hide'}
    _reserved7             : BIT;
    {attribute 'hide'}
    _reserved8             : ARRAY [1..3] OF USINT;
END_STRUCT;
END_TYPE

```

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 7.1.8 NCTOPLC\_AXIS\_REF\_CAMCOUPLINGSTATE

```

TYPE NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE :
STRUCT
  CamActivationPending      : BIT;
  CamDeactivationPending   : BIT;
  CamActive                 : BIT;
  {attribute 'hide'}
  _reserved1               : BIT;
  {attribute 'hide'}
  _reserved2               : BIT;
  {attribute 'hide'}
  _reserved3               : BIT;
  CamDataQueued            : BIT;
  CamScalingPending        : BIT;
END_STRUCT
END_TYPE

```

Bit	Variable name	Description
0	CamActivationPending	Table waiting for activation
1	CamDeactivationPending	Table waiting for deactivation
2	CamActive	Table is active
3-5		RESERVE
6	CamDataQueued	Table data (MF) ready for "Online Change" and waiting for activation
7	CamScalingPending	Table scalings ready for "Online Change" and waiting for activation

#### Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4020	PC or CX (x86)	Tc2_MC2

### 7.1.9 PLCTONC\_AXIS\_REF

The data structure PLCTONC\_AXIS\_REF is part of the [AXIS\\_REF \[▶ 118\]](#) data structure and cyclically transfers information to the NC. PLCTONC\_AXIS\_REF is also referred to as axis interface between PLC and NC.

```

TYPE PLCTONC_AXIS_REF
STRUCT
  ControlDWord              : PLCTONC_AXIS_REF_CTRL; (* Control double word *)
  Override                  : UDINT; (* Velocity override *)
  AxisModeRequest           : UDINT; (* Axis operating mode (PLC request) *)
  AxisModeDWord             : UDINT; (* optional mode parameter *)
  AxisModeLReal             : LREAL; (* optional mode parameter *)
  PositionCorrection        : LREAL; (* Correction value for current position *)
  ExtSetPos                 : LREAL; (* external position setpoint *)
  ExtSetVelo                : LREAL; (* external velocity setpoint *)
  ExtSetAcc                 : LREAL; (* external acceleration setpoint *)
  ExtSetDirection          : DINT; (* external direction setpoint *)
  {attribute 'hide'}
  _reserved1               : UDINT; (* reserved *)
  ExtControllerOutput       : LREAL; (* external controller output *)
  GearRatio1                : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
  GearRatio2                : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
  GearRatio3                : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
  GearRatio4                : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
  MapState                  : BOOL; (* reserved - internal use *)
  PlcCycleControl           : BYTE;
  PlcCycleCount             : BYTE;
  {attribute 'hide'}
  _reserved2               : ARRAY [1..5] OF USINT;
  ExtTorque                 : LREAL;
  {attribute 'hide'}
  _reserved3               : ARRAY [1..8] OF USINT;
END_STRUCT
END_TYPE

```

Variable name	Data type	Definition range	Description
ControlDWord	PLCTONC_AXIS_REF_CTRL L [▶ 126]	0/1	Control double word
Override	UDINT	0...1000000	Velocity override (0% to 100%)
AxisModeRequest	UDINT		Axis operating mode. Only provided for internal use!
AxisModeDWord	UDINT		Only provided for internal use!
AxisModeLReal	LREAL		Only provided for internal use!
PositionCorrection	LREAL		Actual position correction value
ExtSetPos	LREAL		External set position
ExtSetVelo	LREAL		External set velocity
ExtSetAcc	LREAL		External set acceleration
ExtSetDirection	DINT		External set travel direction [-1,0,1]
ExtControllerOutput	LREAL		External controller output. Not yet released!
GearRatio1	LREAL	$\pm\infty$	Gear ratio (coupling factor) 1
GearRatio2	LREAL	$\pm\infty$	Gear ratio (coupling factor) 2
GearRatio3	LREAL	$\pm\infty$	Gear ratio (coupling factor) 3
GearRatio4	LREAL	$\pm\infty$	Gear ratio (coupling factor) 4
MapState	BOOL		Internal use only
PlcCycleControl	BYTE		Internal use only
PlcCycleCount	BYTE		Internal use only
ExtTorque	LREAL		Torque for MC_TorqueControl with "ContinuousUpdate"

## Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.1.10 PLCTONC\_AXIS\_REF\_CTRL

The structure PLCTONC\_AXIS\_REF\_CTRL is part of the structure NCTOPLC\_AXIS\_REF [▶ 125].

```
TYPE PLCTONC_AXIS_REF_CTRL :
    DWORD;
END_TYPE
```

Bit	Variable name	Description
0	Enable	Enable controller
1	FeedEnablePlus	Feed enable plus
2	FeedEnableMinus	Feed enable minus
3-4	-	RESERVE
5	HomingSensor	Referencing cam or referencing sensor
6-7	-	RESERVE
8	AcceptBlockedDrive	Accept blocking of the drive setpoint adoption (e.g. hardware end positions) from TwinCAT V2.10 Build 1311
9	BlockedDriveDetected	User signal "Axis is blocked" (e.g. mechanical fixed stop). Not yet released!
10-29	-	RESERVE
30	PlcDebugFlag	PLC debug function. For internal use only!

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.2 DriveOperationMode

### 7.2.1 E\_DriveOperationMode

This data type is used in conjunction with the function blocks [MC\\_ReadDriveOperationMode \[▶ 45\]](#) and [MC\\_WriteDriveOperationMode \[▶ 46\]](#), to read out or set the currently active operation mode.

```

TYPE E_ReadMode :
(
  DriveOperationMode_Default := 0, // 0x0000: default => reactivate the nc
  default op mode (if default mode is known!?)

  // general operation modes (e.g. CANopen/CoE)
  DriveOperationMode_Torque := 1, // 0x0001: torque control
  DriveOperationMode_Velo1 := 2, // 0x0002: velocity control with feedback
k 1
  DriveOperationMode_Velo2 := 3, // 0x0003: velocity control with feedback
k 2
  DriveOperationMode_Pos1 := 4, // 0x0004: pos control with feedback 1,
lag less
  DriveOperationMode_Pos2 := 5, // 0x0005: pos control with feedback 2,
lag less
  DriveOperationMode_TorqueAndCommutationAngle := 6, // 0x0006: torque control with commutati
on angle

  DriveOperationMode_VelocityMode := 34, // 0x0022: DS402/
CoE specific drive controlled velocity mode (VL)
  DriveOperationMode_DriveBasedHoming := 38, // 0x0026: DS402/
CoE specific drive controlled homing mode

  // special indirect index based operation modes (only for SERCOS/
SoE). In most cases the general operation modes shall be used
  DriveOperationMode_Index0 := 100, // 100: SERCOS/
SoE operation mode index 0 (primary operation mode 0, s. S-0-0032)
  DriveOperationMode_Index1 := 101, // 101: SERCOS/
SoE operation mode index 1 (secondary operation mode 1, s. S-0-0033)
  DriveOperationMode_Index2 := 102, // 102: SERCOS/
SoE operation mode index 2 (secondary operation mode 2, s. S-0-0034)
  DriveOperationMode_Index3 := 103, // 103: SERCOS/
SoE operation mode index 3 (secondary operation mode 3, s. S-0-0035)
) DINT;
END_TYPE

```

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.3 External setpoint generator

### 7.3.1 ST\_ExtSetPointEnableOptions

```

TYPE ST_ExtSetPointEnableOptions
STRUCT
  UseTorqueOffset : BOOL;
END_STRUCT
END_TYPE

```

Name	Type	Description
UseTorqueOffset	BOOL	Must be set to TRUE if a TorqueOffset is to be transferred to the drive controller cyclically using the MC_ExtSetPointGenFeedWithTorque function.

## 7.4 Gearing

### 7.4.1 E\_GearInMultiMasterSyncMode

E\_GearInMultiMasterSyncMode defines the possible synchronization modes for axis coupling at [MC\\_GearInMultiMaster \[► 106\]](#), if the slave axis motion is limited in AdvancedSlaveDynamics mode (limitation of velocity, acceleration and jerk).

```
TYPE E_GearInMultiMasterSyncMode :
(
  VELOSYNC := 0,
  POSSYNC1 := 1,
  POSSYNC2 := 2
) INT;
END_TYPE
```

E_GearInMultiMasterSyncMode	Description
VELOSYNC	The slave axis and the master axes move with a synchronized velocity. In the case of a dynamic limitation, the slave establishes a position difference that is not caught up again. (Standard)
POSSYNC1	The slave axis and the master axes move with synchronized velocity and position. In the case of a dynamic limitation, the slave establishes a position difference that is then caught up at a later time. Changes in the gear ratios during the journey are executed taking into account the dynamics. The position difference between master and slave resulting from this is not caught up.
POSSYNC2	The slave axis and the master axes move with synchronized velocity and position. In the case of a dynamic limitation, the slave establishes a position difference that is then caught up at a later time. Changes in the gear ratios during the journey are immediately calculated internally and the slave continuously synchronizes itself to the new position.

### 7.4.2 ST\_GearInDynOptions

```
TYPE ST_GearInDynOptions
STRUCT
  CCVmode : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
CCVmode	BOOL	Constant circumference velocity mode

### 7.4.3 ST\_GearInMultiMasterOptions

The data structure extends the function block [MC\\_GearInMultiMaster \[► 106\]](#) with additional input parameters.



```

TYPE ST_GearInMultiMasterOptions :
STRUCT
  AdvancedSlaveDynamics : BOOL;
  SyncMode                : E_GearInMultiMasterSyncMode;
END_STRUCT
END_TYPE
    
```

Name	Type	Description
AdvancedSlaveDynamics	BOOL	Swaps the internal algorithm of the function block. This makes it possible to synchronize to masters already in motion. In this case, "Acceleration" and "Deceleration" should only be parameterized symmetrically. If jerk presets are too large, this is reduced to the extent that a change from zero to the parameterized acceleration / deceleration can take place in one NC cycle. The resolution of the acceleration / deceleration thus depends directly on the suitable parameterization of the jerk value.
SyncMode	<a href="#">E_GearInMultiMasterSyncMode</a> [▶ 128]	The SyncMode specifies how the axis coupling behaves if the movement of the slave axis is limited in the AdvancedSlaveDynamics mode (limitation of velocity, acceleration and jerk). (available from TwinCAT 3.1.4024.11)

### 7.4.4 ST\_GearInOptions

```

TYPE ST_GearInOptions
STRUCT
  SlaveType : _E_TcNC_SlaveTypes;
END_STRUCT
END_TYPE
    
```

Name	Type	Description
SlaveType	<a href="#">_E_TcNC_SlaveTypes</a>	not used

## 7.5 Homing

### 7.5.1 E\_EncoderReferenceMode

This data type is used in conjunction with the [MC\\_Home](#) [▶ 96] function block and the ... structure to select a reference mode other than the one set in the configuration on the encoder, if required.

```

Type E_EncoderReferenceMode :
(
  ENCODERREFERENCEMODE_DEFAULT           := 0,
  ENCODERREFERENCEMODE_PLCCAM           := 1,
  ENCODERREFERENCEMODE_HARDWARESYNC     := 2,
  ENCODERREFERENCEMODE_HARDWARELATCHPOS := 3,
  ENCODERREFERENCEMODE_HARDWARELATCHNEG := 4,
  ENCODERREFERENCEMODE_SOFTWARESYNC     := 5,
  ENCODERREFERENCEMODE_SOFTDRIVELATCHPOS := 6,
  ENCODERREFERENCEMODE_SOFTDRIVELATCHNEG := 7,
) UDINT;
END_TYPE
    
```

## 7.5.2 MC\_HomingMode

This data type is used to parameterize the function block [MC\\_Home](#) [[▶ 96](#)].

```

TYPE MC_HomingMode :
(
  MC_DefaultHoming, (* default homing as defined in the SystemManager encoder parameters *)
  MC_Direct, (* Static Homing forcing position from user reference *)
  MC_ForceCalibration, (* set the calibration flag without performing any motion or changing the p
osition *)
  MC_ResetCalibration (* resets the calibration flag without performing any motion or changing th
e position *)
);
END_TYPE

```

MC_HomingMode	Description
MC_DefaultHoming	Executes the default homing.
MC_Direct	Sets the axis position directly to Position without executing a movement.
MC_ForceCalibration	Forces the state "Axis is calibrated". No movement takes place, and the position remains unchanged.
MC_ResetCalibration	Resets the calibration status of the axis. No movement takes place, and the position remains unchanged.

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.5.3 ST\_HomingOptions

The data structure extends the function block [MC\\_Home](#) [[▶ 96](#)] with additional input parameters.

```

TYPE ST_HomingOptions :
STRUCT
  ClearPositionLag : BOOL;
  SearchDirection : MC_Direction := MC_Direction.MC_Undefined_Direction;
  SearchVelocity : LREAL;
  SyncDirection : MC_Direction := MC_Direction.MC_Undefined_Direction;
  SyncVelocity : LREAL;
  ReferenceMode : E_EncoderReferenceMode :=
E_EncoderReferenceMode.ENCODERREFERENCEMODE_DEFAULT;
END_STRUCT
END_TYPE

```

Name	Type	Default	Description
ClearPositionLag	BOOL		Only works if <a href="#">MC_HomingMode</a> [ <a href="#">▶ 130</a> ] = MC_Direct. Can optionally be used to set the set and actual positions to the same value. In this case the lag error is cleared.
SearchDirection	<a href="#">MC_Direction</a> [ <a href="#">▶ 135</a> ]	MC_Undefined_Direction	The search of the calibration cam can be done in positive or negative direction of travel. If nothing is specified, the search is performed in the positive direction of travel.
SearchVelocity	LREAL		Axis velocity at which the calibration cam is searched.
SyncDirection	<a href="#">MC_Direction</a> [ <a href="#">▶ 135</a> ]	MC_Undefined_Direction	The search of the sync pulse can be done in positive or negative direction of travel. If nothing is specified, the search is performed in the positive direction of travel.

Name	Type	Default	Description
SyncVelocity	LREAL		Axis velocity at which the sync pulse is searched for.
ReferenceMode	E_EncoderReferenceMode [▶ 129]		Parameterizes which signal is used for the sync pulse search.

## 7.6 Motion

### 7.6.1 E\_JogMode

This data type is used in conjunction with the function block [MC\\_Jog](#) [▶ 98].

```

TYPE E_JogMode :
(
  MC_JOGMODE_STANDARD_SLOW, (* motion with standard jog parameters for slow motion *)
  MC_JOGMODE_STANDARD_FAST, (* motion with standard jog parameters for fast motion *)
  MC_JOGMODE_CONTINUOUS, (* axis moves as long as the jog button is pressed using parameterized dynamics *)
  MC_JOGMODE_INCHING, (* axis moves for a certain relative distance *)
  MC_JOGMODE_INCHING_MODULO (* axis moves for a certain relative distance - stop position is rounded to the distance value *)
);
END_TYPE
    
```

E_JogMode	Description
MC_JOGMODE_STANDARD_SLOW	The axis moves as long as the signal at one of the jog inputs is TRUE. The "low velocity for manual functions" specified in the TwinCAT System Manager and standard dynamics are used. In this operation mode the position, velocity and dynamics data specified in the function block have no effect.
MC_JOGMODE_STANDARD_FAST	The axis moves as long as the signal at one of the jog inputs is TRUE. The "high velocity for manual functions" specified in the TwinCAT System Manager and standard dynamics are used. In this operation mode the position, velocity and dynamics data specified in the function block have no effect.
MC_JOGMODE_CONTINUOUS	The axis moves as long as the signal at one of the jog inputs is TRUE. The velocity and dynamics data specified by the user are used. The position has no effect.
MC_JOGMODE_INCHING	The axis is moved with a rising edge at one of the jog inputs by a certain distance, which is defined via the "Position" input. The axis stops automatically, irrespective of the state of the jog inputs. A new movement step is only executed once a further rising edge is encountered. With each start the velocity and dynamics data specified by the user are used.
MC_JOGMODE_INCHING_MODULO	The axis is moved with a rising edge at one of the jog inputs by a certain distance, which is defined via the "Positions" input. The axis position will snap to an integer multiple of the position parameter. The axis stops automatically, irrespective of the state of the jog inputs. A new movement step is only executed once a further rising edge is encountered. With each start the velocity and dynamics data specified by the user are used.

#### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.6.2 E\_PositionType

E\_PositionType is used for external setpoint generation [MC\\_ExtSetPointGenEnable \[► 40\]](#) to define how the given position is to be interpreted.

```
TYPE E_PositionType :
(
  POSITIONTYPE_ABSOLUTE := 1, (*Absolute position*)
  POSITIONTYPE_RELATIVE  (*Relative position*)
);
END_TYPE
```

E_PositionType	Description
POSITIONTYPE_ABSOLUTE	Absolute position
POSITIONTYPE_RELATIVE	Relative position

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.6.3 MC\_BufferMode

This data type is used with various function blocks of the Motion Control library. BufferMode is used to specify how successive motion commands are to be processed.

```
TYPE MC_BufferMode :
(
  MC_Aborting,
  MC_Buffered,
  MC_BlendingLow,
  MC_BlendingPrevious,
  MC_BlendingNext,
  MC_BlendingHigh
);
END_TYPE
```

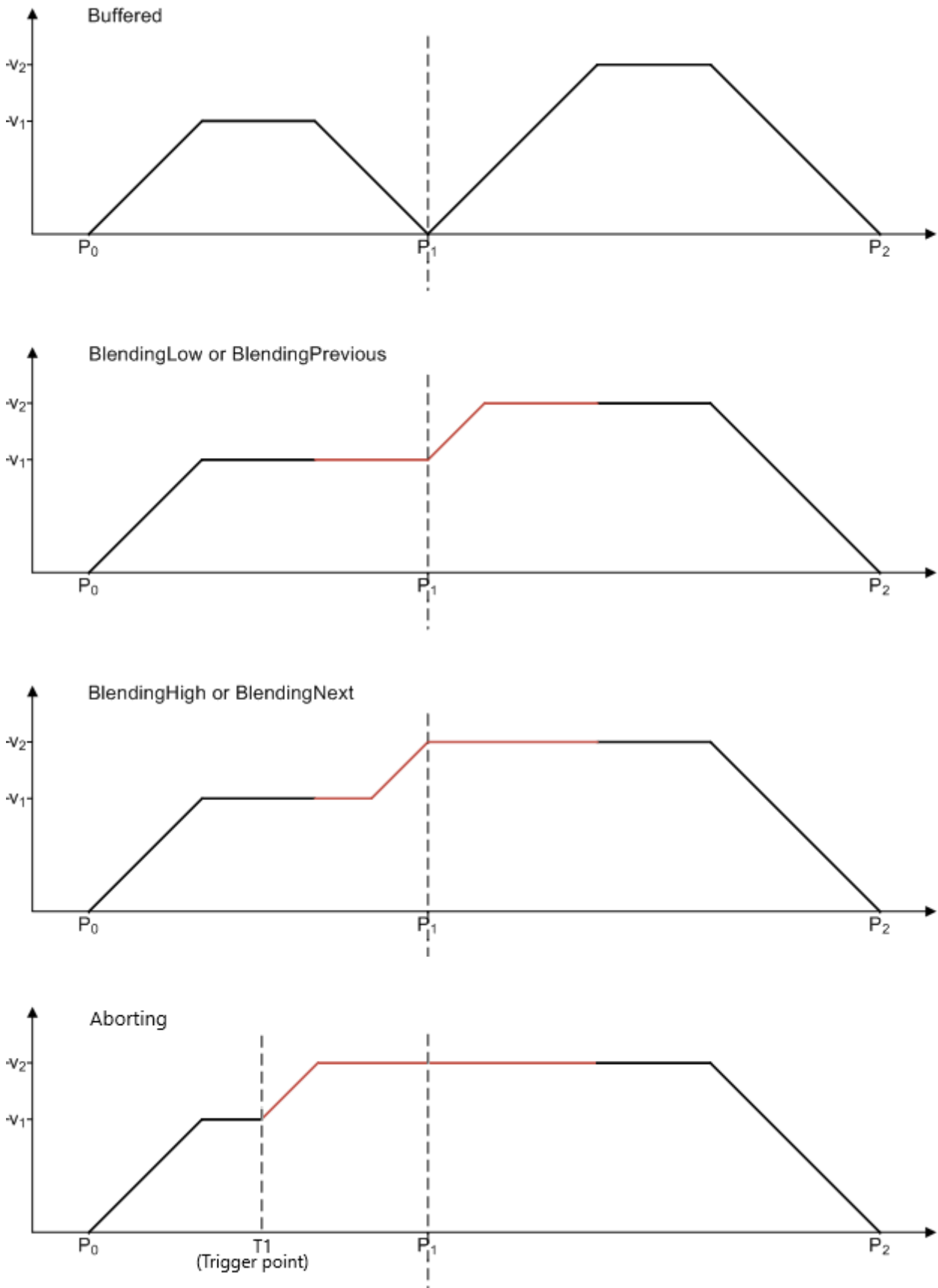


A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.

See also: [General rules for MC function blocks \[► 14\]](#)

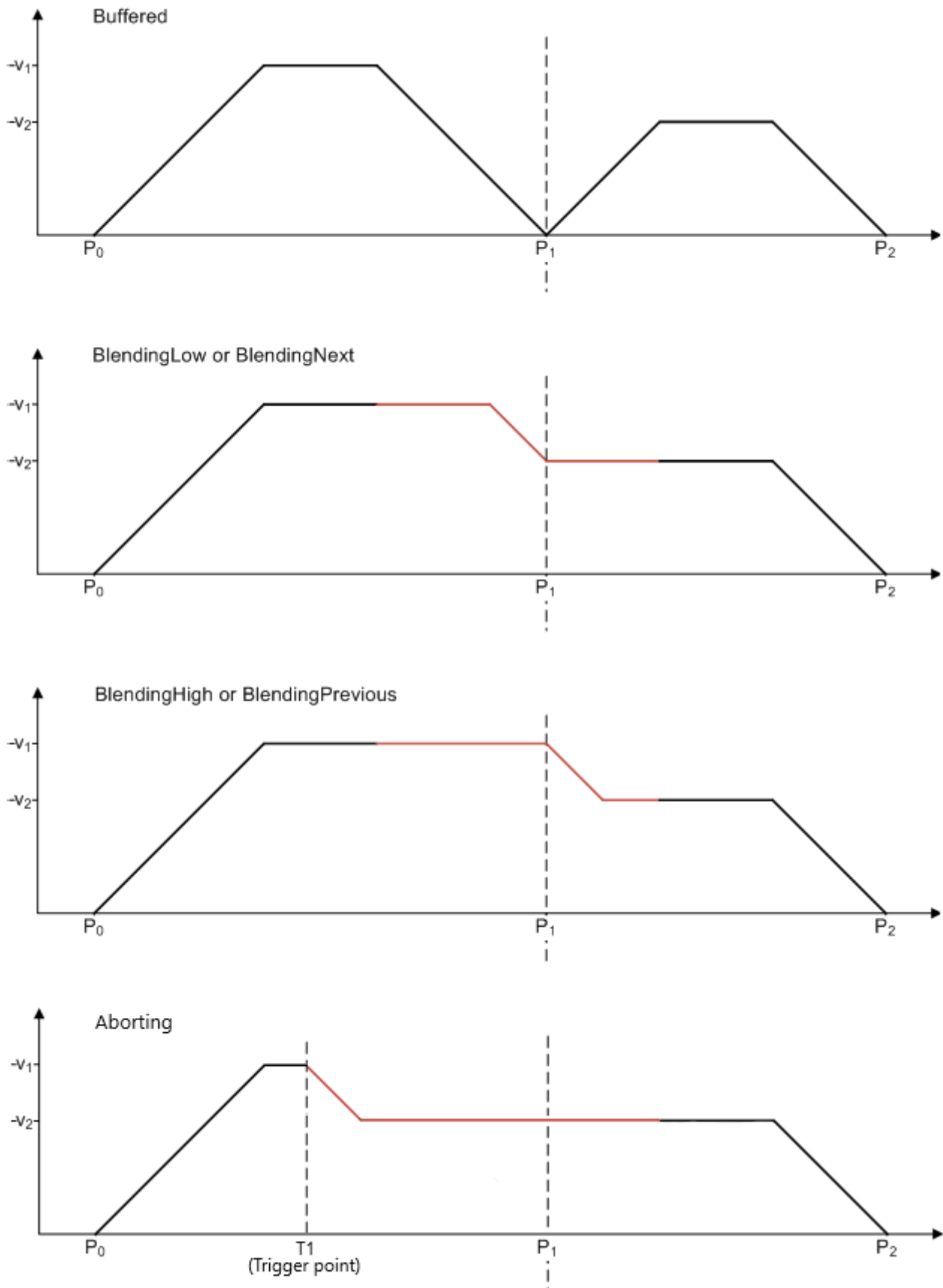
### Examples:

In the following example, a move command is used to move an axis from position  $P_0$  to  $P_1$  and then to  $P_2$ . The second command is issued during the movement to  $P_1$ , but before the braking ramp with different BufferModes. The reference point for the different velocity profiles is always  $P_1$ . The mode specifies the velocity  $v_1$  or  $v_2$  at this point.



Since the speed of the first command is lower than the second, the modes BlendingLow/BlendingPrevious and BlendingHigh/BlendingNext have the same result.

The difference in the next example is that the speed of the second command is lower than the first. Now, the modes BlendingLow/BlendingNext and BlendingHigh/BlendingPrevious are equivalent.



The velocity profiles described here assume that the following command is issued in time, i.e. before the braking ramp of the first command. Otherwise, blending is implemented as best as possible.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 7.6.4 MC\_Direction

This enumeration type contains the possible directions of movement for the function blocks [MC\\_MoveVelocity \[► 79\]](#) and [MC\\_MoveModulo \[► 71\]](#).

```

TYPE MC_Direction :
(
  MC_Positive_Direction := 1,
  MC_Shortest_Way ,
  MC_Negative_Direction,
  MC_Current_Direction,
  MC_Undefined_Direction := 128
);
END_TYPE
    
```

MC_Direction	Description
MC_Positive_Direction	Positive direction of movement
MC_Shortest_Way	Shortest way
MC_Negative_Direction	Negative direction of movement
MC_Current_Direction	Current direction of movement is maintained
MC_Undefined_Direction	Undefined direction of movement

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 7.6.5 ST\_MoveOptions

This data type contains optional settings for travel commands such as [MC\\_MoveAbsolute \[► 65\]](#) or [MC\\_Halt \[► 85\]](#).

```

TYPE ST_MoveOptions :
STRUCT
// Command activation at defined ActivationPosition
// Extends the buffer mode when enabled
  EnableBlendingPosition : BOOL;
  BlendingPosition : LREAL;

// PositionAreaMonitoring, TargetPositionMonitoring
// and StopMonitoring can be ignored using this flag
  IgnorePositionMonitoring : BOOL;

// PositionAreaMonitoring, TargetPositionMonitoring
// can be enabled for MC_Stop and MC_Halt commands
  EnableStopPositionMonitoring : BOOL;
END_STRUCT
END_TYPE
    
```

ST_MoveOptions	Type	Description
EnableBlendingPosition	BOOL	This flag can be activated in order to define a different blending position for a command. In the normal case, the blending position is the target of the running command. With an active <a href="#">MC_MoveVelocity [► 79]</a> , no target is defined at first.
BlendingPosition	LREAL	Blending position used with active EnableBlendingPosition flag.

ST_MoveOptions	Type	Description
IgnorePositionMonitoring	BOOL	The target window monitor can be switched off for a single travel command with this flag. The target window monitor waits until the actual value is within a defined target window after reaching the target position. The target window monitor can be activated in the axis parameters.
EnableStopPositionMonitoring	BOOL	MC_Stop [▶ 87] and MC_Halt [▶ 85] commands do not define a target position. However, a target window monitor can still be activated for these commands with this flag. The travel command then waits until the actual position of the axis is also within a defined window at the stopping position. The target window monitor must be activated in the axis parameters for this.

## Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86)	Tc2_MC2

## 7.7 Nc Process Data

### 7.7.1 E\_NcIoDevice

```

TYPE E_NcIoDevice :
(
  NcIoDeviceDrive := 0,
  NcIoDeviceEncoder := 1
) INT;
END_TYPE

```

### 7.7.2 E\_NcIoOutput

```

TYPE E_NcIoOutput :
(
  NcIoOutputnCtrl1 := 0,
  NcIoOutputnCtrl2 := 1,
  NcIoOutputnCtrl3 := 2,
  NcIoOutputnCtrl4 := 3,
  NcIoOutputnCtrl5 := 4,
  NcIoOutputnCtrl6 := 5,
  NcIoOutputnCtrl7 := 6,
  NcIoOutputnCtrl8 := 7,
  NcIoOutputnDataOut1 := 8,
  NcIoOutputnDataOut2 := 9,
  NcIoOutputnDataOut3 := 10,
  NcIoOutputnDataOut4 := 11,
  NcIoOutputnDataOut5 := 12,
  NcIoOutputnDataOut6 := 13
);
END_TYPE

```

## 7.8 Phasing

### 7.8.1 E\_PhasingType

```

TYPE E_PhasingType :
(
  NC_PHASINGTYPE_ABSOLUTE := 1,
  NC_PHASINGTYPE_RELATIVE := 2,
  NC_PHASINGTYPE_STOP := 4096
) UINT;
END_TYPE

```



## 7.9 Position Correction

### 7.9.1 E\_AxisPositionCorrectionMode

```

TYPE E_PositionCorrectionMode:
(
  POSITIONCORRECTION_MODE_UNLIMITED, (* no limitation - pass correction immediately *)
  POSITIONCORRECTION_MODE_FAST,      (* limitation to maximum position change per cycle *)
  POSITIONCORRECTION_MODE_FULLENGTH (* limitation uses full length to adapt to correction in small steps *)
);
END_TYPE
    
```

E_AxisPositionCorrectionMode	Description
POSITIONCORRECTION_MODE_UNLIMITED	No filtering, the correction is executed immediately. Note that large changes in the correction value can lead to high accelerations.
POSITIONCORRECTION_MODE_FAST	The position correction is limited to the extent that a maximum acceleration is not exceeded. However, the correction is completely executed as fast as possible.
POSITIONCORRECTION_MODE_FULLENGTH	The position correction is accomplished distributed over a distance of the axis (CorrectionLength). This results in smaller changes per time unit.

### 7.9.2 ST\_PositionCompensationTableElement

```

TYPE ST_PositionCompensationTableElement :
STRUCT
  Position      : LREAL;
  Compensation  : LREAL;
END_STRUCT
END_TYPE
    
```

Name	Type	Description
Position	LREAL	Uncorrected position value (e.g. SetPos)
Compensation	LREAL	Additive compensation value

### 7.9.3 ST\_PositionCompensationTableParameter

```

TYPE ST_PositionCompensationTableParameter :
STRUCT
  MinPosition      : LREAL;
  MaxPosition      : LREAL;
  NoOfTableElements : UDINT;
  Direction        : E_WorkDirection := WorkDirectionBoth;
  Modulo           : BOOL;
END_STRUCT
END_TYPE
    
```

Name	Type	Default	Description
MinPosition	LREAL		Start position for position compensation
MaxPosition	LREAL		End position for position compensation
NoOfTableElements	UDINT		Number of table entries
Direction	<a href="#">E_WorkDirection</a>  > 142]	WorkDirectionBoth	The position compensation is performed only in the selected direction.
Modulo	BOOL		FALSE: linear axis TRUE: modulo axis with cyclic period

## 7.10 SelectControlLoop

### 7.10.1 E\_SelectControlLoopType

Defines how the function block `MC_SelectControlLoop` [► 55] performs the control loop switching.

```
Type E_SelectControlLoop :
(
  SelectControlLoopType_Undefined,
  SelectControlLoopType_Standard,
  SelectControlLoopType_InternalSyncValue,
  SelectControlLoopType_UserSyncValue
) UDINT;
END_TYPE
```

E_SelectControlLoop	Description
SelectControlLoopType_Undefined	Not defined.
SelectControlLoopType_Standard	Simple switching (similar to an axis reset)
SelectControlLoopType_InternalSyncValue	Switching/synchronization by means of I/D-part of the controller to an internal initial value (jerk-free/smooth).
SelectControlLoopType_UserSyncValue	Switching/synchronization by means of I/D-part of the controller to a parameterizable initial value.

## 7.11 Status and parameter

### 7.11.1 E\_AxisErrorCodes

Axis error codes and error codes that can be returned by the function blocks of `Tc2_MC2` are defined in the enum `E_AxisErrorCodes`. A description of the `NC error codes` can be found here.

### 7.11.2 E\_NcAxisType

TwinCAT supports different axis types, which are defined in the enum `E_NcAxisType`.

```
TYPE E_NcAxisType
(
  NcAxisType_undefined           := 0,
  NcAxisType_Continuous         := 1,
  NcAxisType_Discrete_TwoSpeed  := 2,
  NcAxisType_LowCostStepper_DigIO := 3,
  NcAxisType_Encoder            := 5,
  NcAxisType_Hydraulic          := 6,
  NcAxisType_TimeGenerator      := 7,
  NcAxisType_Specific           := 100
) DWORD;
END_TYPE
```

E_NcAxisType	Description
NcAxisType_undefined	
NcAxisType_Continuous	Continuous axis (also SERCOS)
NcAxisType_Discrete_TwoSpeed	Discrete axis (high/low speed)
NcAxisType_LowCostStepper_DigIO	Stepper motor axis (without PWM terminal KL2502/30 and without pulse train KL2521)
NcAxisType_Encoder	Encoder axis
NcAxisType_Hydraulic	Continuous axis with operation mode switching for position/pressure control
NcAxisType_TimeGenerator	Time Base Generator
NcAxisType_Specific	

### 7.11.3 E\_NcDriveType

TwinCAT supports different drives, these are defined in the enum E\_NcDriveType.

```

TYPE E_NcDriveType :
(
  NcDriveType_undefined           := 0,
  NcDriveType_M2400_DAC1         := 1,
  NcDriveType_M2400_DAC2         := 2,
  NcDriveType_M2400_DAC3         := 3,
  NcDriveType_M2400_DAC4         := 4,
  NcDriveType_KL4xxx              := 5,
  NcDriveType_KL4xxx_NonLinear    := 6,
  NcDriveType_Discete_TwoSpeed    := 7,
  NcDriveType_Stepper             := 8,
  NcDriveType_Sercos              := 9,
  NcDriveType_KL5051             := 10,
  NcDriveType_AX2000_B200        := 11,
  NcDriveType_ProfilDrive         := 12,
  NcDriveType_Universal          := 13,
  NcDriveType_NcBackplane        := 14,
  NcDriveType_CANopen_Lenze      := 15,
  NcDriveType_CANopen_DS402_MDP742 := 16,
  NcDriveType_AX2000_B900        := 17,
  NcDriveType_KL2531_Stepper     := 20,
  NcDriveType_KL2532_DC          := 21,
  NcDriveType_TCOM               := 22,
  NcDriveType_MDP_733            := 23,
  NcDriveType_MDP_703            := 24
) DWORD;
END_TYPE
    
```

E_NcDriveType	Description
NcDriveType_undefined	
NcDriveType_M2400_DAC1	
NcDriveType_M2400_DAC2	
NcDriveType_M2400_DAC3	
NcDriveType_M2400_DAC4	
NcDriveType_KL4xxx	MDP 252/253: KL4xxx, PWM KL2502_30K (Frq-Cnt pulse mode), KL4132 (16-bit), Pulse-Train KL2521, IP2512
NcDriveType_KL4xxx_NonLinear	MDP 252/253: analog type for non-linear characteristics
NcDriveType_Discete_TwoSpeed	
NcDriveType_Stepper	
NcDriveType_Sercos	
NcDriveType_KL5051	MDP 510: BiSSI Drive KL5051 with 32 bits (see KL4xxx)
NcDriveType_AX2000_B200	AX2000-B200 Lightbus, incremental with 32 bits (AX2000)
NcDriveType_ProfilDrive	Incremental with 32 Bit
NcDriveType_Universal	Variable bit mask (max. 32 bits, signed value)
NcDriveType_NcBackplane	Variable bit mask (max. 32 bits, signed value)
NcDriveType_CANopen_Lenze	CANopen Lenze (max. 32 bits, signed value)
NcDriveType_CANopen_DS402_MDP742	MDP 742 (DS402): CANopen and EtherCAT (AX2000-B510, AX2000-B1x0, EL7201, AX8000)
NcDriveType_AX2000_B900	AX2000-B900 Ethernet (max. 32 bits, signed value)
NcDriveType_KL2531_Stepper	Stepper motor terminal KL2531/KL2541
NcDriveType_KL2532_DC	2-channel DC motor stage KL2532/KL2542, 2-channel PWM DC motor stage KL2535/KL2545
NcDriveType_TCOM	TCOM Drive -> Interface to Soft Drive
NcDriveType_MDP_733	MDP 733: Modular Device Profile MDP 733 for DC (e.g. EL7332/EL7342)

E_NcDriveType	Description
NcDriveType_MDP_703	MDP 703: Modular Device Profile MDP 703 for stepper (e.g. EL7031/EL7041)

## 7.11.4 E\_NcEncoderType

TwinCAT supports different encoders, these are defined in the enum E\_NcEncoderType.

```

TYPE E_NcEncoderType :
(
  NcEncoderType_undefined,
  NcEncoderType_Simulation,
  NcEncoderType_ABS_M3000,
  NcEncoderType_INC_M31X0,
  NcEncoderType_INC_KL5101,
  NcEncoderType_ABS_KL5001_SSI,
  NcEncoderType_INC_KL5051,
  NcEncoderType_ABS_KL30XX,
  NcEncoderType_INC_Sercos_P,
  NcEncoderType_INC_Sercos_PV,
  NcEncoderType_INC_Binary,
  NcEncoderType_ABS_M2510,
  NcEncoderType_ABS_FOX50,
  NcEncoderType_HYDRAULIC_FORCE,
  NcEncoderType_AX2000_B200,
  NcEncoderType_PROFIDRIVE,
  NcEncoderType_UNIVERSAL,
  NcEncoderType_NCBACKPLANE,
  NcEncoderType_CANOPEN_LENZE,
  NcEncoderType_CANOPEN_DS402_MDP513_MDP742,
  NcEncoderType_AX2000_B900,
  NcEncoderType_KL5151,
  NcEncoderType_IP5209,
  NcEncoderType_KL2531_Stepper,
  NcEncoderType_KL2532_DC,
  NcEncoderType_TIMEBASEGENERATOR,
  NcEncoderType_INC_TCOM,
  NcEncoderType_CANOPEN_MDP513_64BIT,
  NcEncoderType_SPECIFIC
) DWORD;
END_TYPE

```

E_NcEncoderType	Description
NcEncoderType_undefined	
NcEncoderType_Simulation	Simulation
NcEncoderType_ABS_M3000	Absolute, with 24 or 25 bits, and 12 and 13 bits single turn encoders (M3000)
NcEncoderType_INC_M31X0	Incremental, with 24 bits (M31x0, M3200, M3100, M2000)
NcEncoderType_INC_KL5101	MDP 511: Incremental with 16 bits and latch (MDP511: EL7041, EL5101, EL5151, EL2521, EL5021(SinCos); KL5101, IP5109, KL5111)
NcEncoderType_ABS_KL5001_SSI	MDP 500/501: Absolute SSI with 24 bits (KL5001, IP5009) (MDP 501: EL5001)
NcEncoderType_INC_KL5051	MDP 510: Absolute/Incremental BiSSI with 16 bits (KL5051, PWM KL2502_30K(Frq-Cnt pulse mode), Pulse-Train KL2521, IP2512)
NcEncoderType_ABS_KL30XX	Absolute analog input with 16 bits (KL30xx)
NcEncoderType_INC_Sercos_P	SERCOS "Encoder" position
NcEncoderType_INC_Sercos_PV	SERCOS "Encoder" position and velocity
NcEncoderType_INC_Binary	Binary incremental encoders (0/1)
NcEncoderType_ABS_M2510	Absolute analog input with 12 bits (M2510)
NcEncoderType_ABS_FOX50	T&R Fox 50 module (24 bits absolute (SSI))
NcEncoderType_HYDRAULIC_FORCE	MMW type: force acquisition from Pa, Pb, Aa, Ab

E_NcEncoderType	Description
NcEncoderType_AX2000_B200	Incremental AX2000-B200 Lightbus with 16/20 bits (AX2000)
NcEncoderType_PROFIDRIVE	Incremental with 32 Bit
NcEncoderType_UNIVERSAL	Incremental with variable bit mask (max. 32 bits)
NcEncoderType_NCBACKPLANE	Incremental NC backplane
NcEncoderType_CANOPEN_LENZE	Incremental CANopen Lenze
NcEncoderType_CANOPEN_DS402_MDP513_MDP742	MDP 513 / MDP 742 (DS402): CANopen and EtherCAT (AX2000-B510, AX2000-B1x0, EL7201, EL5032/32 Bit)
NcEncoderType_AX2000_B900	Incremental AX2000-B900 Ethernet
NcEncoderType_KL5151	Incremental with 16-bit counter and int. + ext 32-bit latch (KL5151_0000) (only switchable), the 2-channel KL5151_0050 has no latch.
NcEncoderType_IP5209	Incremental with 16-bit counter and int. 32-bit latch (IP5209)
NcEncoderType_KL2531_Stepper	Incremental with 16-bit counter and int. + ext. 15-bit latch (only switchable) (stepper motor terminal KL2531/ KL2541)
NcEncoderType_KL2532_DC	Incremental with 16-bit counter and ext. 16-bit latch (only switchable) (2-channel DC motor output stage KL2532/ KL2542), 2-channel PWM DC motor output stage KL2535/ KL2545
NcEncoderType_TIMEBASEGENERATOR	Time Base Generator
NcEncoderType_INC_TCOM	TCOM Encoder -> Interface to Soft Drive Encoder
NcEncoderType_CANOPEN_MDP513_64BIT	MDP 513 (DS402, EnDat2.2, 64 Bit): EL5032/64 Bit
NcEncoderType_SPECIFIC	

### 7.11.5 E\_ReadMode

This data type is used in conjunction with the function blocks [MC\\_ReadParameter \[▶ 27\]](#) and [MC\\_ReadBoolParameter \[▶ 25\]](#) to specify a single or cyclic operation.

```

TYPE E_ReadMode :
(
    READMODE_ONCE := 1,
    READMODE_CYCLIC
);
END_TYPE
    
```

#### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 7.11.6 E\_SetScalingFactorMode

The E\_SetScalingFactorMode enum is used in conjunction with the function block [MC\\_SetEncoderScalingFactor \[▶ 57\]](#) to change the scaling factor of the active encoder.

```

TYPE E_SetScalingFactorMode :
(
    ENCODERSCALINGMODE_ABSOLUTE,
    ENCODERSCALINGMODE_RELATIVE
) UINT;
END_TYPE
    
```

E_SetScalingFactorMode	Description
ENCODERSCALINGMODE_ABSOLUTE	

E_SetScalingFactorMode	Description
ENCODERSCALINGMODE_RELATIVE	

## 7.11.7 E\_WorkDirection

The enum E\_WorkDirection is used to specify a work direction for a command.

```

TYPE E_WorkDirection :
(
  WorkDirectionNone      := 0,
  WorkDirectionBoth      := 1,
  WorkDirectionPositive  := 2,
  WorkDirectionNegative  := 3
) INT;
END_TYPE

```

E_WorkDirection	Description
WorkDirectionNone	No direction specification
WorkDirectionBoth	Positive and negative work direction
WorkDirectionPositive	Positive work direction
WorkDirectionNegative	Negative work direction

## 7.11.8 MC\_AxisParameter

This data type is used in conjunction with function blocks for reading and writing axis parameters.

```

TYPE MC_AxisParameter : (
(* PLCopen specific parameters *) (* Index-Group 0x4000 + ID*)
  CommandedPosition := 1,          (* lreal *) (* taken from NcToPlc *)
  SWLimitPos,                (* lreal *) (* IndexOffset= 16#0001_000E *)
  SWLimitNeg,                (* lreal *) (* IndexOffset= 16#0001_000D *)
  EnableLimitPos,            (* bool *) (* IndexOffset= 16#0001_000C *)
  EnableLimitNeg,            (* bool *) (* IndexOffset= 16#0001_000B *)
  EnablePosLagMonitoring,    (* bool *) (* IndexOffset= 16#0002_0010 *)
  MaxPositionLag,            (* lreal *) (* IndexOffset= 16#0002_0012 *)
  MaxVelocitySystem,         (* lreal *) (* IndexOffset= 16#0000_0027 *)
  MaxVelocityAppl,           (* lreal *) (* IndexOffset= 16#0000_0027 *)
  ActualVelocity,            (* lreal *) (* taken from NcToPlc *)
  CommandedVelocity,         (* lreal *) (* taken from NcToPlc *)
  MaxAccelerationSystem,     (* lreal *) (* IndexOffset= 16#0000_0101 *)
  MaxAccelerationAppl,       (* lreal *) (* IndexOffset= 16#0000_0101 *)
  MaxDecelerationSystem,     (* lreal *) (* IndexOffset= 16#0000_0102 *)
  MaxDecelerationAppl,       (* lreal *) (* IndexOffset= 16#0000_0102 *)
  MaxJerkSystem,             (* lreal *) (* IndexOffset= 16#0000_0103 *)
  MaxJerkAppl,               (* lreal *) (* IndexOffset= 16#0000_0103 *)

(* Beckhoff specific parameters *) (* Index-Group 0x4000 + ID*)
  AxisId := 1000,             (* lreal *) (* IndexOffset= 16#0000_0001 *)
  AxisVeloManSlow,           (* lreal *) (* IndexOffset= 16#0000_0008 *)
  AxisVeloManFast,           (* lreal *) (* IndexOffset= 16#0000_0009 *)
  AxisVeloMax,                (* lreal *) (* IndexOffset= 16#0000_0027 *)
  AxisAcc,                    (* lreal *) (* IndexOffset= 16#0000_0101 *)
  AxisDec,                    (* lreal *) (* IndexOffset= 16#0000_0102 *)
  AxisJerk,                   (* lreal *) (* IndexOffset= 16#0000_0103 *)
  MaxJerk,                    (* lreal *) (* IndexOffset= 16#0000_0103 *)
  AxisMaxVelocity,           (* lreal *) (* IndexOffset= 16#0000_0027 *)
  AxisRapidTraverseVelocity, (* lreal *) (* IndexOffset= 16#0000_000A *)
  AxisManualVelocityFast,     (* lreal *) (* IndexOffset= 16#0000_0009 *)
  AxisManualVelocitySlow,     (* lreal *) (* IndexOffset= 16#0000_0008 *)
  AxisCalibrationVelocityForward, (* lreal *) (* IndexOffset= 16#0000_0006 *)
  AxisCalibrationVelocityBackward, (* lreal *) (* IndexOffset= 16#0000_0007 *)
  AxisJogIncrementForward,    (* lreal *) (* IndexOffset= 16#0000_0018 *)
  AxisJogIncrementBackward,  (* lreal *) (* IndexOffset= 16#0000_0019 *)
  AxisEnMinSoftPosLimit,     (* bool *) (* IndexOffset= 16#0001_000B *)
  AxisMinSoftPosLimit,       (* lreal *) (* IndexOffset= 16#0001_000D *)
  AxisEnMaxSoftPosLimit,     (* bool *) (* IndexOffset= 16#0001_000C *)
  AxisMaxSoftPosLimit,       (* lreal *) (* IndexOffset= 16#0001_000E *)
  AxisEnPositionLagMonitoring, (* bool *) (* IndexOffset= 16#0002_0010 *)
  AxisMaxPosLagValue,        (* lreal *) (* IndexOffset= 16#0002_0012 *)
  AxisMaxPosLagFilterTime,   (* lreal *) (* IndexOffset= 16#0002_0013 *)
  AxisEnPositionRangeMonitoring, (* bool *) (* IndexOffset= 16#0000_000F *)

```

AxisPositionRangeWindow,	(* lreal *)	(* IndexOffset= 16#0000_0010 *)	
AxisEnTargetPositionMonitoring,	(* bool *)	(* IndexOffset= 16#0000_0015 *)	
AxisTargetPositionWindow,	(* lreal *)	(* IndexOffset= 16#0000_0016 *)	
AxisTargetPositionMonitoringTime,	(* lreal *)	(* IndexOffset= 16#0000_0017 *)	
AxisEnInTargetTimeout,	(* bool *)	(* IndexOffset= 16#0000_0029 *)	
AxisInTargetTimeout,	(* lreal *)	(* IndexOffset= 16#0000_002A *)	
AxisEnMotionMonitoring,	(* bool *)	(* IndexOffset= 16#0000_0011 *)	
AxisMotionMonitoringWindow,	(* lreal *)	(* IndexOffset= 16#0000_0028 *)	
AxisMotionMonitoringTime,	(* lreal *)	(* IndexOffset= 16#0000_0012 *)	
AxisDelayTimeVeloPosition,	(* lreal *)	(* IndexOffset= 16#0000_0104 *)	
AxisEnLoopingDistance,	(* bool *)	(* IndexOffset= 16#0000_0013 *)	
AxisLoopingDistance,	(* lreal *)	(* IndexOffset= 16#0000_0014 *)	
AxisBacklash,	(* lreal *)	(* IndexOffset= 16#0000_002C *)	
AxisEnDataPersistence,	(* bool *)	(* IndexOffset= 16#0000_0030 *)	
AxisRefVeloOnRefOutput,	(* lreal *)	(* IndexOffset= 16#0003_0101 *)	
AxisOverrideType,	(* lreal *)	(* IndexOffset= 16#0000_0105 *)	
(* new since 4/2007 *)			
AxisEncoderScalingFactor,	(* lreal *)	(* IndexOffset= 16#0001_0006 *)	
AxisEncoderOffset,	(* lreal *)	(* IndexOffset= 16#0001_0007 *)	
AxisEncoderDirectionInverse,	(* bool *)	(* IndexOffset= 16#0001_0008 *)	
AxisEncoderMask,	(* dword *)	(* IndexOffset= 16#0001_0015 *)	
AxisEncoderModuloValue,	(* lreal *)	(* IndexOffset= 16#0001_0009 *)	
AxisModuloToleranceWindow,	(* lreal *)	(* IndexOffset= 16#0001_001B *)	
AxisEnablePosCorrection,	(* bool *)	(* IndexOffset= 16#0001_0016 *)	
AxisPosCorrectionFilterTime,	(* lreal *)	(* IndexOffset= 16#0001_0017 *)	
(* new since 1/2010 *)			
AxisUnitInterpretation,	(* lreal *)	(* IndexOffset= 16#0000_0026 *)	
AxisMotorDirectionInverse,	(* bool *)	(* IndexOffset= 16#0003_0006 *)	
(* new since 1/2011 *)			
AxisCycleTime,	(* lreal *)	(* IndexOffset= 16#0000_0004 *)	
(* new since 5/2011 *)			
AxisFastStopSignalType,	(* dword *)	(* IndexOffset= 16#0000_001E *)	
AxisFastAcc,	(* lreal *)	(* IndexOffset= 16#0000_010A *)	
AxisFastDec,	(* lreal *)	(* IndexOffset= 16#0000_010B *)	
AxisFastJerk,	(* lreal *)	(* IndexOffset= 16#0000_010C *)	
(* new since 1/2012 *)			
AxisEncoderScalingNumerator,	(* lreal *)	(* IndexOffset= 16#0001_0023 - available in Tc3 *)	
AxisEncoderScalingDenominator,	(* lreal *)	(* IndexOffset= 16#0001_0024 - available in Tc3 *)	
(* new since 7/2016 *)			
AxisMaximumAcceleration,	(* lreal *)	(* IndexOffset= 16#0000_00F1 - available in Tc3 *)	
AxisMaximumDeceleration,	(* lreal *)	(* IndexOffset= 16#0000_00F2 - available in Tc3 *)	
AxisVeloJumpFactor,	(* lreal *)	(* IndexOffset= 16#0000_0106 *)	
AxisToleranceBallAuxAxis,	(* lreal *)	(* IndexOffset= 16#0000_0108 *)	
AxisMaxPositionDeviationAuxAxis,	(* lreal *)	(* IndexOffset= 16#0000_0109 *)	
AxisErrorPropagationMode,	(* dword *)	(* IndexOffset= 16#0000_001A *)	
AxisErrorPropagationDelay,	(* lreal *)	(* IndexOffset= 16#0000_001B *)	
AxisCoupleSlaveToActualValues,	(* bool *)	(* IndexOffset= 16#0000_001C *)	
AxisAllowMotionCmdToSlaveAxis,	(* bool *)	(* IndexOffset= 16#0000_0020 *)	
AxisAllowMotionCmdToExtSetAxis,	(* bool *)	(* IndexOffset= 16#0000_0021 *)	
AxisEncoderSubMask,	(* dword *)	(* IndexOffset= 16#0001_0108 *)	
AxisEncoderReferenceSystem,	(* dword *)	(* IndexOffset= 16#0001_0019 *)	
AxisEncoderPositionFilterPT1,	(* lreal *)	(* IndexOffset= 16#0001_0010 *)	
AxisEncoderVelocityFilterPT1,	(* lreal *)	(* IndexOffset= 16#0001_0011 *)	
AxisEncoderAccelerationFilterPT1,	(* lreal *)	(* IndexOffset= 16#0001_0012 *)	
AxisEncoderMode,	(* dword *)	(* IndexOffset= 16#0001_000A *)	
AxisEncoderHomingInvDirCamSearch,	(* bool *)	(* IndexOffset= 16#0001_0101 *)	
AxisEncoderHomingInvDirSyncSearch,	(* bool *)	(* IndexOffset= 16#0001_0102 *)	
AxisEncoderHomingCalibValue,	(* lreal *)	(* IndexOffset= 16#0001_0103 *)	
AxisEncoderReferenceMode,	(* dword *)	(* IndexOffset= 16#0001_0107 *)	
AxisRefVeloOutputRatio,	(* lreal *)	(* IndexOffset= 16#0003_0102 *)	
AxisDrivePositionOutputScaling,	(* lreal *)	(* IndexOffset= 16#0003_0109 *)	
AxisDriveVelocityOutputScaling,	(* lreal *)	(* IndexOffset= 16#0003_0105 *)	
AxisDriveVelocityOutputDelay,	(* lreal *)	(* IndexOffset= 16#0003_010D *)	
AxisDriveMinOutputLimitation,	(* lreal *)	(* IndexOffset= 16#0003_000B *)	
AxisDriveMaxOutputLimitation,	(* lreal *)	(* IndexOffset= 16#0003_000C *)	
AxisTorqueInputScaling,	(* lreal *)	(* IndexOffset= 16#0003_0031 - available in Tc3 *)	
AxisTorqueInputFilterPT1,	(* lreal *)	(* IndexOffset= 16#0003_0032 - available in Tc3 *)	
AxisTorqueDerivationInputFilterPT1,	(* lreal *)	(* IndexOffset= 16#0003_0033 - available in Tc3 *)	
AxisTorqueOutputScaling,	(* lreal *)	(* IndexOffset= 16#0003_010B *)	
AxisTorqueOutputDelay,	(* lreal *)	(* IndexOffset= 16#0003_010F *)	
AxisAccelerationOutputScaling,	(* lreal *)	(* IndexOffset= 16#0003_010A *)	
AxisAccelerationOutputDelay,	(* lreal *)	(* IndexOffset= 16#0003_010E *)	
AxisDrivePosOutputSmoothFilterType,	(* dword *)	(* IndexOffset= 16#0003_0110 *)	
AxisDrivePosOutputSmoothFilterTime,	(* lreal *)	(* IndexOffset= 16#0003_0111 *)	
AxisDrivePosOutputSmoothFilterOrder,	(* dword *)	(* IndexOffset= 16#0003_0112 *)	
AxisDriveMode,	(* dword *)	(* IndexOffset= 16#0003_000A *)	
AxisDriftCompensationOffset,	(* lreal *)	(* IndexOffset= 16#0003_0104 *)	
AxisPositionControlKv,	(* lreal *)	(* IndexOffset= 16#0002_0102 *)	
AxisCtrlVeloCityPreCtrlWeight,	(* lreal *)	(* IndexOffset= 16#0002_000B *)	
AxisControllerMode,	(* dword *)	(* IndexOffset= 16#0002_000A *)	

```

AxisCtrlAutoOffset,          (* bool *) (* IndexOffset= 16#0002_0110 *)
AxisCtrlAutoOffsetTimer,    (* lreal *) (* IndexOffset= 16#0002_0115 *)
AxisCtrlAutoOffsetLimit,    (* lreal *) (* IndexOffset= 16#0002_0114 *)
AxisSlaveCouplingControlKcp, (* lreal *) (* IndexOffset= 16#0002_010F *)
AxisCtrlOutputLimit,        (* lreal *) (* IndexOffset= 16#0002_0100 *)

(* Beckhoff specific axis status information - READ ONLY *) (* Index-Group 0x4100 + ID*)
AxisTargetPosition := 2000,  (* lreal *) (* IndexOffset= 16#0000_0013 *)
AxisRemainingTimeToGo,      (* lreal *) (* IndexOffset= 16#0000_0014 *)
AxisRemainingDistanceToGo,  (* lreal *) (* IndexOffset= 16#0000_0022, 16#0000_0042 *)
AxisMcSetPositionOffset,    (* lreal *) (* IndexOffset= 16#00n1_0017 *)

(* Beckhoff specific axis functions *)
(* read/write gear ratio of a slave *)
AxisGearRatio := 3000,      (* lreal *) (* read: IndexGroup=0x4100+ID, IdxOffset=16#0000_00
22, *)
(* write:IndexGroup=0x4200+ID, IdxOffset=16#0000_0042 *)

(* Beckhoff specific other parameters *)
(* new since 1/2011 *)
NcSafCycleTime := 4000,    (* lreal *) (* IndexOffset= 16#0000_0010 *)
NcSvbCycleTime           (* lreal *) (* IndexOffset= 16#0000_0012 *)
);
END_TYPE

```



The AxisGearRatio parameter can only be read or written if the axis is coupled as a slave. Only very small changes are allowed during motion.

## Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.11.9 MC\_AxisStates

This data type describes the operating states according to the [PlcOpen state diagram \[► 11\]](#).

```

TYPE MC_AxisStates :
(
  MC_AXISSTATE_UNDEFINED,
  MC_AXISSTATE_DISABLED,
  MC_AXISSTATE_STANDSTILL,
  MC_AXISSTATE_ERRORSTOP,
  MC_AXISSTATE_STOPPING,
  MC_AXISSTATE_HOMING,
  MC_AXISSTATE_DISCRETEMOTION,
  MC_AXISSTATE_CONTINUOUSMOTION,
  MC_AXISSTATE_SYNCHRONIZEDMOTION
);
END_TYPE

```

See also: [General rules for MC function blocks \[► 14\]](#).

## Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.11.10 ST\_AxisComponents

```

TYPE ST_AxisComponents
STRUCT
  EncoderIDs      : ARRAY[1..9] OF DWORD;
  ControllerIDs  : ARRAY[1..9] OF DWORD;
  DriveIDs       : ARRAY[1..9] OF DWORD;
END_STRUCT
END_TYPE

```



### 7.11.11 ST\_AxisOpModes

This data type contains information about the parameterization of the operation modes of an axis.

```

TYPE ST_AxisOpModes :
STRUCT
  PositionAreaMonitoring : BOOL; (* bit 0 - OpModeDWord *)
  TargetPositionMonitoring : BOOL; (* bit 1 - OpModeDWord *)
  LoopMode : BOOL; (* bit 2 - OpModeDWord - loop mode for two speed axes *)
  MotionMonitoring : BOOL; (* bit 3 - OpModeDWord *)
  PEHTimeMonitoring : BOOL; (* bit 4 - OpModeDWord *)
  BacklashCompensation : BOOL; (* bit 5 - OpModeDWord *)
  DelayedErrorReaction : BOOL; (* bit 6 - OpModeDWord *)
  Modulo : BOOL; (* bit 7 - OpModeDWord -
axis is parameterized as modulo axis *)
  SimulationAxis : BOOL; (* bit 8 - OpModeDWord *)
  PositionLagMonitoring : BOOL; (* bit 16 - OpModeDWord *)
  VelocityLagMonitoring : BOOL; (* bit 17 - OpModeDWord *)
  SoftLimitMinMonitoring : BOOL; (* bit 18 - OpModeDWord *)
  SoftLimitMaxMonitoring : BOOL; (* bit 19 - OpModeDWord *)
  PositionCorrection : BOOL; (* bit 20 - OpModeDWord *)
  AllowSlaveCommands : BOOL; (* bit 21 - OpModeDWord *)
  AllowExtSetAxisCommands : BOOL; (* bit 22 - OpModeDWord *)
END_STRUCT
END_TYPE

```

#### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 7.11.12 ST\_AxisParameterSet

This data type contains the entire parameter data set of an axis that can be read with the [MC\\_ReadParameterSet](#) [▶ 28] function block.

Individual parameters that can be changed during runtime can be written with [MC\\_WriteParameter](#) [▶ 32]. It is not possible to write back the parameter dataset as a whole.

The individual parameters are described in the NC ADS documentation.

```

TYPE ST_AxisParameterSet :
STRUCT
  (* AXIS: *)
  AxisId : DWORD; (* 0x00000001 *)
  nAxisType : E_NcAxisType; (* 0x00000003 *)
  sAxisName : STRING(31); (* 0x00000002 *)
  fAxisCycleTime : LREAL; (* 0x00000004 *)
  bEnablePositionAreaControl : WORD; (* 0x0000000F *)
  fPositionAreaControlRange : LREAL; (* 0x00000010 *)
  bEnableMotionControl : WORD; (* 0x00000011 *)
  fMotionControlTime : LREAL; (* 0x00000012 *)
  bEnableLoop : WORD; (* 0x00000013 *)
  fLoopDistance : LREAL; (* 0x00000014 *)
  bEnableTargetPosControl : WORD; (* 0x00000015 *)
  fTargetPosControlRange : LREAL; (* 0x00000016 *)
  fTargetPosControlTime : LREAL; (* 0x00000017 *)
  fVeloMaximum : LREAL; (* 0x00000027 *)
  fRefVeloSearch : LREAL; (* 0x00000006 *)
  fRefVeloSync : LREAL; (* 0x00000007 *)
  fVeloSlowManual : LREAL; (* 0x00000008 *)
  fVeloFastManual : LREAL; (* 0x00000009 *)
  fMotionControlRange : LREAL; (* 0x00000028 *)
  bEnablePEHTimeControl : WORD; (* 0x00000029 *)
  fPEHControlTime : LREAL; (* 0x0000002A *)
  bEnableBacklashCompensation : WORD; (* 0x0000002B *)
  fBacklash : LREAL; (* 0x0000002C *)
  sAmsNetId : T_AmsNetId; (* 0x00000031 *)
  nPort : WORD; (* 0x00000031 *)
  nChnNo : WORD; (* 0x00000031 *)
  fAcceleration : LREAL; (* 0x00000101 *)
  fDeceleration : LREAL; (* 0x00000102 *)
  fJerk : LREAL; (* 0x00000103 *)

```

```

(* ENCODER: *)
nEncId          : DWORD;          (* 0x00010001 *)
nEncType        : E_NcEncoderType; (* 0x00010003 *)
sEncName        : STRING(31);     (* 0x00010002 *)
fEncScaleFactorNumerator : LREAL; (* 0x00010023 *)
fEncScaleFactorDenominator : LREAL; (* 0x00010024 *)
fEncScaleFactor : LREAL;          (* 0x00010006 *)
fEncOffset      : LREAL;          (* 0x00010007 *)
bEncIsInverse   : WORD;           (* 0x00010008 *)
fEncModuloFactor : LREAL;         (* 0x00010009 *)
nEncMode        : DWORD;          (* 0x0001000A *)
bEncEnableSoftEndMinControl : WORD; (* 0x0001000B *)
bEncEnableSoftEndMaxControl : WORD; (* 0x0001000C *)
fEncSoftEndMin  : LREAL;         (* 0x0001000D *)
fEncSoftEndMax  : LREAL;         (* 0x0001000E *)
nEncMaxIncrement : DWORD;        (* 0x00010015 *)
nEncRefSoftSyncMask : DWORD;     (* 0x00010108 *)
bEncEnablePosCorrection : WORD;  (* 0x00010016 *)
nEncReferenceSystem : DWORD;     (* 0x00010019 *)
fEncPosCorrectionFilterTime : LREAL; (* 0x00010017 *)
bEncRefSearchInverse : UINT;     (* 0x00010101 *)
bEncRefSyncInverse : UINT;      (* 0x00010102 *)
nEncRefMode     : UDINT;         (* 0x00010107 *)
fEncRefPosition : LREAL;        (* 0x00010103 *)

(* CONTROLLER: *)
nCtrlId        : DWORD;          (* 0x00020001 *)
nCtrlType      : DWORD;          (* 0x00020003 *)
sCtrlName      : STRING(31);     (* 0x00020002 *)
bCtrlEnablePosDiffControl : WORD; (* 0x00020010 *)
bCtrlEnableVeloDiffControl : WORD; (* 0x00020011 *)
fCtrlPosDiffMax : LREAL;        (* 0x00020012 *)
fCtrlPosDiffMaxTime : LREAL;    (* 0x00020013 *)
fCtrlPosKp     : LREAL;          (* 0x00020102 *)
fCtrlPosTn     : LREAL;          (* 0x00020103 *)
fCtrlPosTv     : LREAL;          (* 0x00020104 *)
fCtrlPosTd     : LREAL;          (* 0x00020105 *)
fCtrlPosExtKp  : LREAL;          (* 0x00020106 *)
fCtrlPosExtVelo : LREAL;        (* 0x00020107 *)
fCtrlAccKa     : LREAL;          (* 0x00020108 *)

(* DRIVE: *)
nDriveId       : DWORD;          (* 0x00030001 *)
nDriveType     : E_NcDriveType; (* 0x00030003 *)
sDriveName     : STRING(31);     (* 0x00030002 *)
bDriveIsInverse : WORD;         (* 0x00030006 *)
nDriveControlDWord : DWORD;     (* 0x00030010 *)
fDriveVeloReferenz : LREAL;     (* 0x00030101 *)
fDriveOutputReferenz : LREAL;   (* 0x00030102 *)
fDriveOutputScalingAcc : LREAL; (* 0x0003000A *)
fDriveOutputScalingTorque : LREAL; (* 0x0003000B *)
fDriveInputScalingTorque : LREAL; (* 0x00030031 *)
fDriveInputFilterTimeTorque : LREAL; (* 0x00030032 *)
fDriveInputFilterTimeTorqueDerivative : LREAL; (* 0x00030033 *)

fAccelerationMax : LREAL;       (* 0x000300F1 *)
fDecelerationMax : LREAL;       (* 0x000300F2 *)
END_STRUCT
END_TYPE

```

## Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

### 7.11.13 ST\_AxisStatus

This data type contains extensive status information about an axis. The data structure must be updated in each PLC cycle by calling `MC_ReadStatus` [► 29] or by calling the action `Axis.ReadStatus()` ([AXIS\\_REF](#) [► 118]).

```

TYPE ST_AxisStatus :
STRUCT
  UpdateTaskIndex      : BYTE; (* Task-Index of the task that updated this data set *)
  UpdateCycleTime      : LREAL; (* task cycle time of the task which calls the status function

```

```

*)
  CycleCounter          : UDINT; (* PLC cycle counter when this data set updated *)
  NcCycleCounter       : UDINT; (* NC cycle counter incremented after NC task updated NcToPlc
data structures *)

  MotionState          : MC_AxisStates; (* motion state in the PLCopen state diagram *)

  Error                : BOOL; (* axis error state *)
  ErrorId              : UDINT; (* axis error code *)

  (* PLCopen motion control statemachine states: *)
  ErrorStop            : BOOL;
  Disabled              : BOOL;
  Stopping              : BOOL;
  StandStill           : BOOL;
  DiscreteMotion       : BOOL;
  ContinuousMotion     : BOOL; (* StatedWord bit 19 *)
  SynchronizedMotion   : BOOL;
  Homing                : BOOL;

  (* additional status - (PLCopen definition)*)
  ConstantVelocity     : BOOL; (* StatedWord bit 12 *)
  Accelerating         : BOOL;
  Decelerating         : BOOL;

  (* Axis.NcToPlc.StateDWord *)
  Operational          : BOOL; (* StatedWord bit 0 *)
  ControlLoopClosed   : BOOL; (* StatedWord bit 20 -
operational and position control active *)
  HasJob               : BOOL; (* StatedWord bit 8 *)
  HasBeenStopped       : BOOL; (* StatedWord bit 7 *)
  NewTargetPosition   : BOOL; (* StatedWord bit 17 *-
* new target position commanded during move *)
  InPositionArea      : BOOL; (* StatedWord bit 3 *)
  InTargetPosition    : BOOL; (* StatedWord bit 4 *)
  ProtectedMode       : BOOL; (* StatedWord bit 5 *)
  Homed                : BOOL; (* StatedWord bit 1 *)
  HomingBusy          : BOOL; (* StatedWord bit 11 *)
  MotionCommandsLocked : BOOL; (* StatedWord bit 29 *- stop 'n hold *)
  SoftLimitMinExceeded : BOOL; (* StatedWord bit 26 *- reverse soft travel limit exceeded *)
  SoftLimitMaxExceeded : BOOL; (* StatedWord bit 27 *- forward soft travel limit exceeded *)

  Moving               : BOOL; (* StatedWord bit 9 or 10 *)
  PositiveDirection   : BOOL; (* StatedWord bit 9 *)
  NegativeDirection   : BOOL; (* StatedWord bit 10 *)
  NotMoving           : BOOL; (* StatedWord bit 2 *)
  Compensating        : BOOL; (* StatedWord bit 13 *- superposition - overlaid motion *)

  ExtSetPointGenEnabled : BOOL; (* StatedWord bit 14 *)
  PhasingActive       : BOOL; (* StatedWord bit 15 *)
  ExternalLatchValid  : BOOL; (* StatedWord bit 16 *)
  CamDataQueued       : BOOL; (* StatedWord bit 22 *)
  CamTableQueued      : BOOL; (* StatedWord bit 21 *)
  CamScalingPending   : BOOL; (* StatedWord bit 23 *)
  CmdBuffered         : BOOL; (* StatedWord bit 24 *)
  PTPmode             : BOOL; (* StatedWord bit 25 *)
  DriveDeviceError    : BOOL; (* StatedWord bit 28 *)
  IoDataInvalid       : BOOL; (* StatedWord bit 30 *)
  ErrorPropagationDelayed : BOOL; (* StatedWord bit 6 *)
  DriveLimitActive    : BOOL; (* StatedWord bit 18 *)

  (* Axis.NcToPlc.CoupleState *)
  Coupled              : BOOL;

  (* axis operation mode feedback from NcToPlc *)
  OpMode               : ST_AxisOpModes;
  NcApplicationRequest : BOOL; (* OpModeDWord bit 23 *)
END_STRUCT
END_TYPE

```

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.11.14 ST\_DriveAddress

This data type contains the ADS access data of a drive device. The data is read with the function block [MC\\_ReadDriveAddress](#) [► 54].

```

TYPE ST_DriveAddress :
STRUCT
  NetID          : T_AmsNetId;      (* AMS NetID of the drive as a string *)
  NetIdBytes     : T_AmsNetIdArr;   (* AMS NetID of the drive as a byte array (same information as NetID) *)
  SlaveAddress   : T_AmsPort;      (* slave address of the drive connected to a bus master *)
  Channel        : BYTE;           (* channel number of the drive *)
  (* new since 2013-04-04 - just available with versions after this date, otherwise zero *)
  NcDriveId      : DWORD;          (* ID [1..255] of the NC software drive of an axis *)
  NcDriveIndex   : DWORD;          (* index [0..9] of the NC software drive of an axis *)
  NcDriveType    : E_NcDriveType;  (* type enumeration of the NC software drive of an axis *)
  NcEncoderId    : DWORD;          (* ID [1..255] of the NC software encoder of an axis *)
  NcEncoderIndex : DWORD;          (* index [0..9] of the NC software encoder of an axis *)
  NcEncoderType  : E_NcEncoderType; (* type enumeration of the NC encoder drive of an axis *)
  NcAxisId       : DWORD;          (* ID [1..255] of the NC axis *)
  NcAxisType     : E_NcAxisType;   (* type enumeration of the NC axis *)
  (* new since 2016-04-11 - just available with versions after this date, otherwise zero *)
  TcDriveObjectId : DWORD;
  TcEncoderObjectId : DWORD;
  TcAxisObjectId  : DWORD;
END_STRUCT
END_TYPE

```

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.11.15 ST\_McOutputs

```

TYPE ST_McOutputs
STRUCT
  Done          : BOOL;
  Busy          : BOOL;
  Active        : BOOL;
  CommandAborted : BOOL;
  Error         : BOOL;
  ErrorID       : UDINT;
END_STRUCT
END_TYPE

```

## 7.11.16 ST\_NcApplicationRequest

```

TYPE ST_NcApplicationRequest
STRUCT
  bApplRequestBit : UINT;
  nApplRequestType : UINT;
  nApplRequestCounter : UDINT;
  nApplRequestVersion : UDINT;
END_STRUCT
END_TYPE

```

## 7.11.17 ST\_SetEncoderScalingOptions

```

TYPE ST_SetEncoderScalingOptions
STRUCT
  SelectEncoderIndex : BOOL;
  EncoderIndex       : UINT;
END_STRUCT
END_TYPE

```

## 7.11.18 ST\_SetPositionOptions

This data type contains the optional settings for the function block [MC\\_SetPosition](#) [► 19].

```

TYPE ST_SetPositionOptions
STRUCT
    ClearPositionLag      : BOOL;
    SelectEncoderIndex   : BOOL;
    EncoderIndex         : UINT;
    ClearPositionOffset  : BOOL;
END_STRUCT
END_TYPE
    
```

**ClearPositionOffset – deleting the MC\_SetPosition offset**

The position offset totaled by the [MC\\_SetPosition](#) [► 19] call can be deleted with this function block. To do this, the ClearPositionOffset switch is set in the options; the position transferred to the block is not relevant in this case.



This option is available from TwinCAT 3.1.4024.51 and Tc2\_MC2 3.3.56.

```

VAR
    mcSetPositionClear: MC_SetPosition;
END_VAR

mcSetPositionClear.Options.ClearPositionOffset := TRUE;
mcSetPositionClear (
    Axis:= Axis,
    Position:= , // not relevant
    Execute:= TRUE);
    
```

Homing the axis, e.g. with MC\_Home, produces a newly referenced coordinate system for the axis and also deletes the offset.

**7.11.19 ST\_NcPositionConversionOptions**

This data type contains the optional settings for the [MC\\_ConvertIncPosToPos](#) [► 63] and [MC\\_ConvertPosToIncPos](#) [► 64] function blocks.

```

TYPE ST_NcPositionConversionOptions
STRUCT
    SubIndex      : UINT;
    CtrlMask      : UINT;
END_STRUCT
END_TYPE
    
```

**7.12 Stepper**

**7.12.1 E\_DestallDetectMode**

```

TYPE E_DestallDetectMode :
(
    PwStDetectMode_None := 0,
    PwStDetectMode_Encoderless,
    PwStDetectMode_Lagging(* use encoder signals for stall detection *)
);
END_TYPE
    
```

E_DestallDetectMode	Description
PwStDetectMode_None	No blocking detection
PwStDetectMode_Encoderless	Use of status signals of the KL2531/KL2541 for blocking detection
PwStDetectMode_Lagging	Use of encoder signals for blocking detection

**7.12.2 E\_DestallMode**

```

TYPE E_DestallMode :
(
    PwStMode_None := 0,
    PwStMode_SetError,
    PwStMode_SetErrNonRef,
    );
END_TYPE
    
```

```

    PwStMode_UseOverride
);
END_TYPE

```

### 7.12.3 ST\_PowerStepperStruct

```

TYPE ST_PowerStepperStruct :
STRUCT
    DestallDetectMode : E_DestallDetectMode;
    DestallMode       : E_DestallMode;
    DestallEnable     : BOOL;
    StatusMonEnable   : BOOL;
    Retries           : INT;
    Timeout           : TIME;
END_STRUCT
END_TYPE

```

#### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.13 Superposition

### 7.13.1 E\_SuperpositionAbortOption

```

TYPE E_SuperpositionAbortOption :
(
    SUPERPOSITIONOPTION_ABORTATSTANDSTILL := 0,
    SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL,
    SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP
) UINT;
END_TYPE

```

E_SuperpositionAbortOption	Description
SUPERPOSITIONOPTION_ABORTATSTANDSTILL	Cancels the superimposed motion when the velocity of the subordinate motion profile is zero - default from <a href="#">MC_MoveSuperImposed [► 89]</a> (TcMC version 1)
SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL	Resumes the superimposed motion after a temporary standstill of the subordinate motion profile
SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP	Resumes the superimposed motion after a stop and restart of the subordinate motion profile - <code>ResumeAfterMotionStop</code> includes <code>ResumeAfterStandstill</code>

### 7.13.2 E\_SuperpositionMode

```

TYPE E_SuperpositionMode :
(
    SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION := 1,
    SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION,
    SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION,
    SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION,
    SUPERPOSITIONMODE_ACCREDUCTION_ADDITIVEMOTION, (from TwinCAT 2.11)
    SUPERPOSITIONMODE_ACCREDUCTION_LIMITEDMOTION (from TwinCAT 2.11)
);
END_TYPE

```

`E_SuperpositionMode` determines how a superimposed motion is carried out with the function block [MC\\_MoveSuperImposed \[► 89\]](#).

The modes referred to as "Veloreduction" execute a superimposed movement with minimum velocity change, preferentially over the full parameterized compensation section. Conversely, the modes referred to as "Lengthreduction" use the maximum possible velocity and therefore reduce the required distance. In both cases same distance is compensated.

In cases referred to as "Additivemotion", the superimposed axis executes a longer or shorter movement than indicated by "Length", with the difference described by Distance. These modes are used, for example, if the Length parameter refers to a reference axis and the superimposed axis may move by a longer or shorter distance in comparison.

In cases referred to as "Limitedmotion", the superposition is completed within the parameterized distance. These modes are used, for example, if the Length parameter refers to the superimposed axis itself. With these modes it should be noted that the superimposed Distance must be significantly shorter than the available "Length".

E_SuperpositionMode	Description
SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION	<p>The superimposed movement takes place over the whole "Length". The specified maximum change in velocity "VelocityDiff" is reduced in order to reach the required "Distance" over this length.</p> <p>"Length" refers to a reference axis without superimposed movement (e.g. master axis). The travel path of the axis affected by this compensation is Length + Distance.</p>
SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION	<p>The superimposed movement takes place over the whole "Length". The specified maximum change in velocity "VelocityDiff" is reduced in order to reach the required "Distance" over this length.</p> <p>The "Length" refers to the axis affected by the compensation. During compensation, the travel path of this axis is "Length".</p>
SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION	<p>The distance of the superimposed motion is as short as possible and the velocity is as high as possible. Although neither the maximum velocity change "VelocityDiff" nor the maximum "Length" are exceeded.</p> <p>"Length" refers to a reference axis without superimposed movement (e.g. master axis). The maximum travel path of the axis affected by this compensation is "Length + Distance".</p>
SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION	<p>The distance of the superimposed motion is as short as possible and the velocity is as high as possible. Although neither the maximum velocity change "VelocityDiff" nor the maximum "Length" are exceeded.</p> <p>The "Length" refers to the axis affected by the compensation. During compensation, the maximum travel path of this axis is "Length".</p>
SUPERPOSITIONMODE_ACCREDUCTION_ADDITIVEMOTION (from TwinCAT 2.11)	<p>The superimposed movement takes place over the whole "Length". The specified maximum acceleration (parameter "Acceleration" or "Deceleration") is reduced as far as possible, in order to reach the specified "Distance" on this path.</p> <p>"Length" refers to a reference axis without superimposed movement (e.g. master axis). The travel path of the axis affected by this compensation is "Length + Distance".</p>
SUPERPOSITIONMODE_ACCREDUCTION_LIMITEDMOTION (from TwinCAT 2.11)	<p>The superimposed movement takes place over the whole "Length". The specified maximum acceleration (parameter "Acceleration" or "Deceleration") is reduced as far as possible, in order to reach the specified "Distance" on this path.</p>

E_SuperpositionMode	Description
	The "Length" refers to the axis affected by the compensation. During compensation, the travel path of this axis is "Length".

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.13.3 ST\_SuperpositionOptions

```
TYPE ST_SuperpositionOptions :
STRUCT
  AbortOption : E_SuperpositionAbortOption;
END_STRUCT
END_TYPE
```

```
TYPE E_SuperpositionAbortOption :
(
  SUPERPOSITIONOPTION_ABORTATSTANDSTILL := 0,
  SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL,
  SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP
);
END_TYPE
```

AbortOption is an optional parameter of the function block [MC\\_MoveSuperImposed](#) [► 89], which determines the behavior of a superimposed movement at a standstill of the main movement.

AbortOption	Description
SUPERPOSITIONOPTION_ABORTATSTANDSTILL	The superimposed movement is aborted as soon as the subordinate movement leads to a standstill of the axis. The only exception to this is a standstill caused by a velocity override of zero. In this case the superimposed movement is also continued as soon as the override is not equal to zero. AbortAtStandstill is the default behavior if the option is not assigned by the user.
SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL	The superimposed movement is not aborted in the case of a temporary standstill of the main movement, but is continued as soon as the axis moves again. This can occur in particular in the case of a reversal of direction or with cam disc movements. The superimposed movement is terminated only if the target position of the axis has been reached or the axis has been stopped.
SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP	The superimposed movement is not aborted in the case of a standstill of the main movement, even if the axis has reached its target position or has been stopped. In this case, the superimposed movement is continued after the axis restarts.  This case is not of importance if the superimposed movement is applied to a slave axis, since this cannot be started or stopped actively. For slave axes, the operation modes RESUMEAFTERSTANDSTILL and RESUMEAFTERMOTIONSTOP are equivalent. The superimposed movement would thus also be continued after a restart of the master axis.

### Overview of the abort conditions for a superimposed movement (MC\_MoveSuperImposed)

	ABORTATSTANDSTILL	RESUMEAFTERSTANDSTILL	RESUMEAFTERMOTIONSTOP
1. override = 0%	resume	resume	resume



	ABORTATSTAND STILL	RESUMEAFTER STANDSTILL	RESUMEAFTER MOTIONSTOP
2. temporary standstill of the main movement	abort	resume	resume
3. motion reversal	abort	resume	resume
4. axis has reached the target position or is stopped	abort	abort	resume
5. axis reset or switch-off of the enable signal	abort	abort	abort
6. For slave axes: Decoupling	abort	abort	abort

**Requirements**

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.14 Torque Control

### 7.14.1 ST\_TorqueControlOptions

This data type contains optional settings for MC\_TorqueControl.

```

TYPE ST_TorqueControlOptions :
STRUCT
    EnableManualTorqueStartValue : BOOL;
    ManualTorqueStartValue       : LREAL;
END_STRUCT
END_TYPE
    
```

ST_TorqueControlOptions	Description
EnableManualTorqueStartValue	The flag can be enabled to specify a start value that differs from the current torque.
ManualTorqueStartValue	Start value for the torque that is used when the EnableManualTorqueStartValue flag is enabled.

**Requirements**

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024	PC or CX (x86)	Tc2_MC2

## 7.15 Touch probe

### 7.15.1 E\_SignalEdge

Edge defines whether the rising or falling edge of the trigger signal is evaluated.

```

TYPE E_SignalEdge :
(
    RisingEdge,
    FallingEdge
) UINT;
END_TYPE
    
```

Name	Description
RisingEdge	Rising signal edge
FallingEdge	Falling signal edge

## 7.15.2 E\_SignalSource

```

TYPE E_SignalSource :
(
  SignalSource_Default,      (* undefined or externally configured *)
  SignalSource_Input1,      (* digital drive input 1 *)
  SignalSource_Input2,      (* digital drive input 2 *)
  SignalSource_Input3,      (* digital drive input 3 *)
  SignalSource_Input4,      (* digital drive input 4 *)
  SignalSource_ZeroPulse := 128, (* encoder zero pulse *)
  SignalSource_DriveDefined (* defined by drive parameters - e. g. CAN object 0x60D0 *)
) UINT;
END_TYPE

```

Name	Description
SignalSource	Optionally defines the signal source, if it can be selected via the controller. In many cases the signal source is permanently configured in the drive and should then be set to the default value "SignalSource_Default".

## 7.15.3 MC\_TouchProbeRecordedData

```

TYPE MC_TouchProbeRecordedData :
STRUCT
  Counter          : LREAL;
  RecordedPosition : LREAL;
  AbsolutePosition : LREAL;
  ModuloPosition   : LREAL;
END_STRUCT
END_TYPE

```

MC_TouchProbeRecorded-Data	Data type	Description
Counter	LREAL	Counter indicating how many valid edges were detected in the last cycle. Detection of multiple edges is only implemented in mode TOUCHPROBEMODE_CONTINUOUS under SERCOS/SOE and must be supported explicitly by the hardware (e.g. AX5000).
RecordedPosition	LREAL	Axis position recorded at the point in time of the trigger signal. This corresponds to the absolute axis position or the modulo axis position, depending on the parameterization.
AbsolutePosition	LREAL	Absolute axis position detected at the point in time of the trigger signal.
ModuloPosition	LREAL	Modulo axis position recorded at the point in time of the trigger signal.

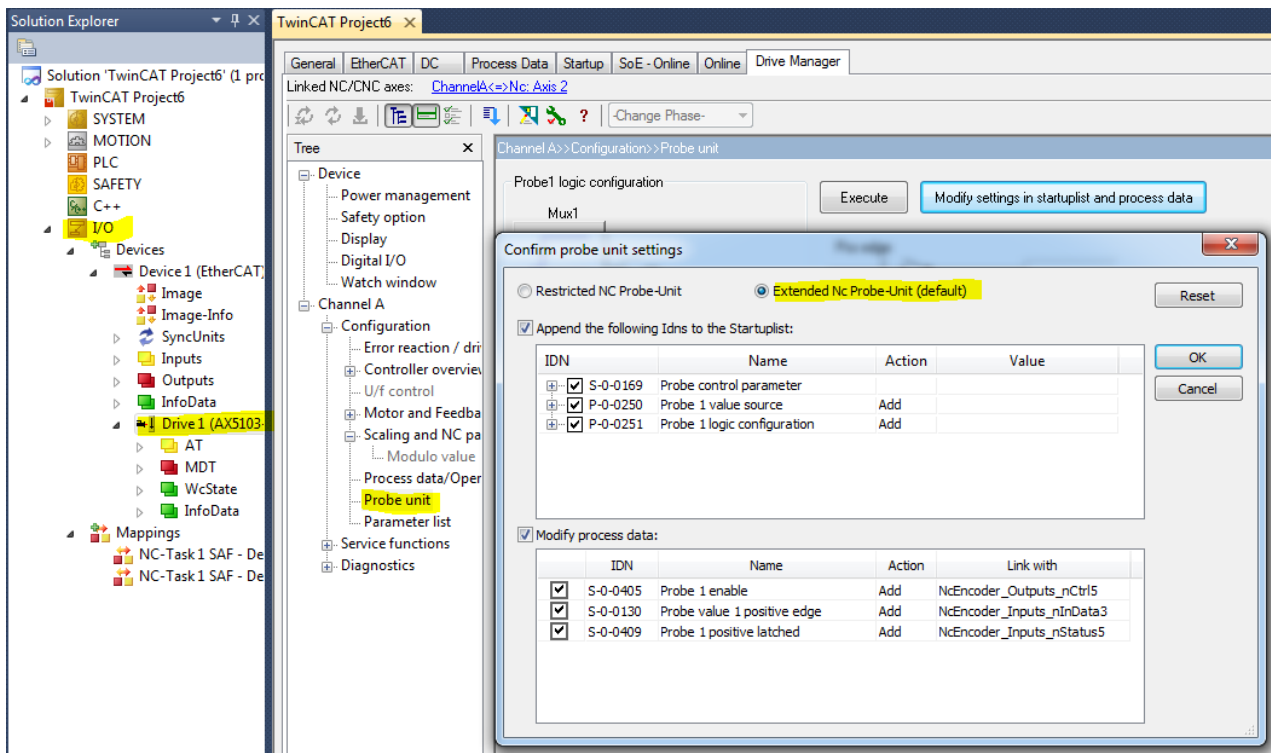
## 7.15.4 TRIGGER\_REF

```

TYPE TRIGGER_REF :
STRUCT
  EncoderID      : UDINT; (* 1..255 *)
  TouchProbe     : E_TouchProbe; (* probe unit definition *)
  SignalSource   : E_SignalSource; (* optional physical signal source used by the probe unit *)
  Edge          : E_SignalEdge; (* rising or falling signal edge *)
  Mode          : E_TouchProbeMode; (* single shot or continuous monitoring *)
  PlcEvent      : BOOL; (* PLC trigger signal input when TouchProbe signal source is set to 'PlcEvent' *)
  ModuloPositions : BOOL; (* interpretation of FirstPosition, LastPosition and RecordedPosition as modulo positions when TRUE *)
END_STRUCT
END_TYPE

```

Name	Data type	Description
EncoderID	UDINT	The ID of an encoder is indicated in the TwinCAT System Manager.
TouchProbe	E_TouchProbe [▶ 156]	Defines the latch unit (probe unit) within the encoder hardware used.
SignalSource	E_SignalSource [▶ 154]	Optionally defines the signal source, if it can be selected via the controller. In many cases the signal source is permanently configured in the drive and should then be set to the default value "SignalSource_Default".
Edge	E_SignalEdge [▶ 153]	Edge defines whether the rising or falling edge of the trigger signal is evaluated.
Mode	E_TouchProbeMode [▶ 156]	Specifies the operation mode of the latch unit. In single mode only the first edge is recorded. In continuous mode each PLC cycle edge is signaled.
PlcEvent	BOOL	If the signal source "TouchProbe" is set to the type "PlcEvent", a rising edge on these variables triggers the recording of the current axis position. "PlcEvent" is not a real latch function, but depends on the cycle time.
ModuloPositions	BOOL	If the variable "ModuloPositions" is FALSE, the axis position is interpreted in an absolute linear range from $-\infty$ to $+\infty$ . The "FirstPosition", "LastPosition" and "RecordedPosition" positions of the function block <a href="#">MC_TouchProbe [▶ 35]</a> are then also absolute. If "ModuloPositions" is TRUE, all positions are interpreted modulo in the modulo range of the axis used (e.g. 0..359.9999). At the same time this means that a defined trigger window repeats itself cyclically.



Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 7.15.5 E\_TouchProbeMode

Specifies the operation mode of the latch unit. In single mode only the first edge is recorded. In Continuous mode, each PLC cycle edge is signaled.

```

TYPE E_TouchProbeMode :
(
    TOUCHPROBEMODE_SINGLE_COMPATIBILITYMODE, (* for TwinCAT 2.10 and 2.11 before Build 2022 *)
    TOUCHPROBEMODE_SINGLE, (* multi probe interface - from 2.11 Build 2022 *)
    TOUCHPROBEMODE_CONTINUOUS (* multi probe interface - from 2.11 Build 2022 *)
) UINT;
END_TYPE

```



For the SINGLE or CONTINUOUS modes the probe unit must be configured as an "Extended Nc Probe Unit".

## 7.15.6 E\_Touch Probe

```

TYPE E_TouchProbe :
(
    TouchProbe1 := 1, (* 1st hardware probe unit with Sercos, CanOpen, KL5xxx and others *)
    TouchProbe2,    (* 2nd probe unit *)
    TouchProbe3,    (* currently not available *)
    TouchProbe4,    (* currently not available *)
    PlcEvent := 10 (* simple PLC signal TRUE/FALSE *)
) UINT;
END_TYPE

```

## 8 Global constants

### 8.1 Library version

All libraries have a certain version. The version is indicated in the PLC library repository, for example. A global constant contains the information about the library version:

#### Global\_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_MC2 : ST_LibVersion;
END_VAR
```

**stLibVersion\_Tc2\_MC2**: version information of the Tc2\_MC2 library (Typ: ST\_LibVersion).

To check whether the version you have is the version you need, use the function F\_CmpLibVersion (defined in Tc2\_System library).

All other options for comparing library versions, which you may know from TwinCAT 2, are outdated.

## 9 Examples

The sample programs use the Tc2\_MC2 library and run entirely in simulation mode.

Progress can be monitored in TwinCAT Scope View with the configuration provided.

### PTP – point to point movement

The sample program manages and moves an axis in PTP mode. The axis is moved with two instances of an MC\_MoveAbsolute function block in buffered mode over several intermediate positions and velocity levels.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386997515.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386997515.zip)

### Master/slave coupling

The sample program couples two axes and moves them together. The slave axis is uncoupled and positioned during the journey.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386995851.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386995851.zip)

### Dancer control

The sample program shows how the velocity of a slave axis can be controlled according to the position of a dancer.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386992523.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386992523.zip)

### Superimposed movement (superposition)

The sample shows the overlay of a movement while an axis is driving.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386999179.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386999179.zip)

### Compensation of the backlash of an axis

The sample program shows how the backlash of an axis can be compensated for.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386989195.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386989195.zip)

### External setpoint generation

The sample shows how an axis can be moved via the external setpoint generator. The movement of the NC axis "Axis" is generated as the sum of the individual movements of the other two Nc axes.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386994187.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386994187.zip)

### Control loop switching in an AX5000 with two existing encoders

The sample illustrates switching between two axis control loops. Suitable hardware is required for this sample.

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/2386990859.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/2386990859.zip)

### Writing outputs in the IO interface of the encoder or drive of an axis

The example shows how the IO outputs not used by the Nc can be written from the PLC

Download: [https://infosys.beckhoff.com/content/1033/TcPlcLib\\_Tc2\\_MC2/Resources/13977016971.zip](https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/13977016971.zip)

### Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86 or x64)	Tc2_MC2

## 10 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

### Download finder

Our [download finder](#) contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

### Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for [local support and service](#) on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: [www.beckhoff.com](http://www.beckhoff.com)

You will also find further documentation for Beckhoff components there.

### Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963-157  
e-mail: [support@beckhoff.com](mailto:support@beckhoff.com)

### Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963-460  
e-mail: [service@beckhoff.com](mailto:service@beckhoff.com)

### Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20  
33415 Verl  
Germany

Phone: +49 5246 963-0  
e-mail: [info@beckhoff.com](mailto:info@beckhoff.com)  
web: [www.beckhoff.com](http://www.beckhoff.com)

# 11 Appendix

## 11.1 NC Backlash Compensation

### 11.1.1 Mechanical backlash

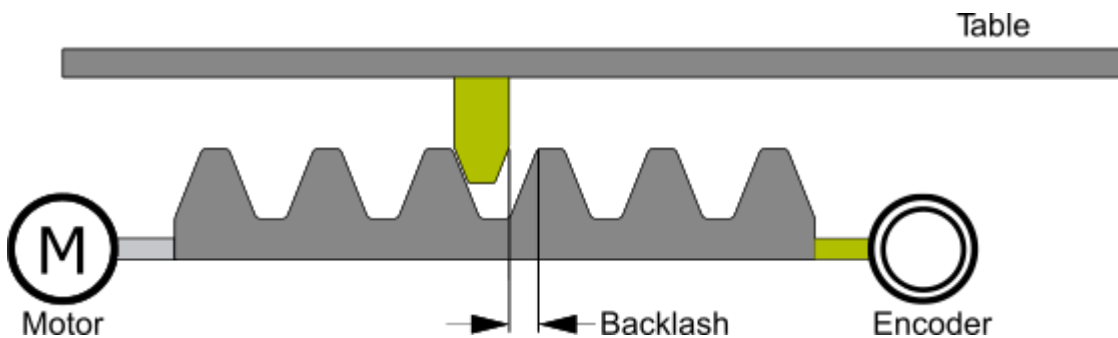
Mechanical backlash is the difference in position between a drive or an encoder and the load. Mechanical backlash arises due to mechanical tolerances in the drivetrain. This causes a difference between the required and the actual position of the load. This is especially important when the direction of motion is reversed.

There are three types of mechanical backlash:

#### Positive backlash

Positive backlash occurs in systems where the measuring system is directly coupled to the drive. In this case the backlash exists between the drive and the load. When the direction of motion is reversed, the measuring system will detect a change in position before the load has moved. So the encoder, which here measures the position of the load indirectly, leads the actual position of the load. As a consequence, the load will not reach the set position, it will be shortened by the length of the backlash.

In the figure below, a movement from left to right is defined as a positive movement.

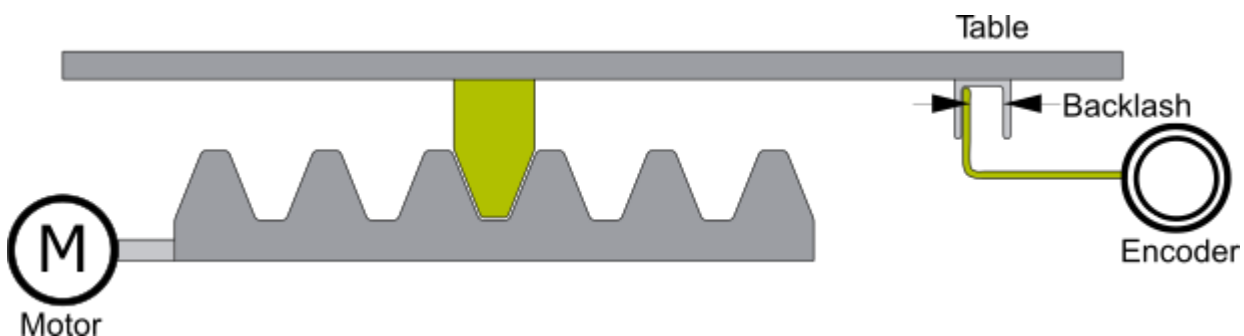


The encoder leads the load (e.g. machine table), so the measured encoder position, leads the actual position of the table. Therefore the table's movement will be too short.

In this case enter a positive correction value for the backlash (= normal case).

#### Negative backlash

Negative backlash occurs in systems where a mechanical tolerance exists between the drive and the measuring system. When the direction of motion is reversed the load immediately moves in the new direction and the measuring system won't detect the changed position. In this case the load will travel further than required. The encoder, which measures the position of the load directly, lags behind the actual position of the machine part.



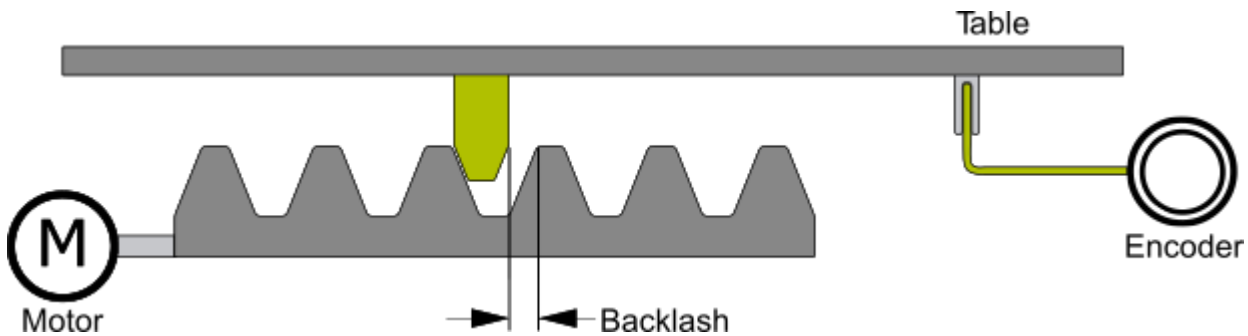
The encoder lags behind the actual position of the load (e.g. machine table). Therefore the table will travel too far.

In this case enter a negative correction value for the backlash.



**Neutral backlash**

In systems, neutral backlash is an exception. In this case the measuring system is directly coupled to the load and electrically connected with the drive. Here the encoder position and the load position are the same. Now when the drive's direction of motion is reversed, the backlash will be automatically compensated. The position control loop is closed around the drivetrain because the encoder is coupled to the load directly. The set position can be reached with no further compensation.



The encoder is directly coupled with the load (e.g. machine table) and ensuring steady state accuracy. No special settings are required.

**General Hints and Notes:**

- Implementation of positive or negative backlash compensation is the same in TwinCAT (only the sign of the backlash value differs). A positive backlash is parameterized as positive value, a negative as negative value.
- A negative backlash is undesirable, because an axis with a backlash in the encoder system is difficult to control (stationery vibrations / oscillations). Typically there are further steps necessary to solve this problem.
- It is not necessary to differentiate between position interface (position control in the drive) and velocity interface (position control in TwinCAT), because they have the same effect. This applies to all variants of backlashes.
- In case of neutral backlash there is no compensation action necessary, even though there is a mechanical backlash. The encoder system is coupled to the machine table therefore enforcing steady state accuracy.
- If a referencing (homing) of an axis is necessary, do the homing with backlash compensation deactivated also position correction deactivated. The last driving direction defines if the left or the right edge is the point of reference by defining a reference position (see [NC Implementing the TwinCAT Backlash Compensation](#) ▶ 162]).

**11.1.2 NC Implementing of the TwinCAT Position Correction**

TwinCAT position correction is used for the backlash compensation.

The following table shows a description of TwinCAT position correction for drives in cyclic position and velocity interfaces.

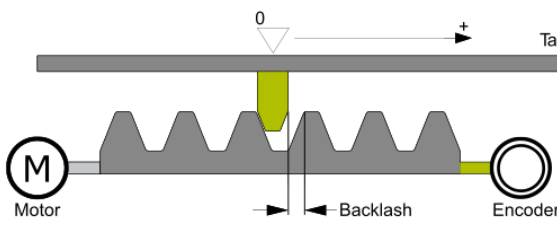
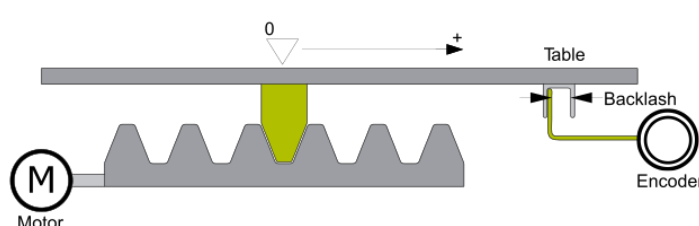
	<b>Implementation and effect of Position Correction resp. Backlash Compensation</b>
<b>Implementation</b>	1. The backlash correction is subtracted from the set position which is transmitted to the drive. 2. The backlash correction is added to the actual position which is transmitted from the encoder.
<b>Effect in the Position Interface</b> (position control in the Drive)	Description / Effect:

	<b>Implementation and effect of Position Correction resp. Backlash Compensation</b>
	<p>By manipulating the set position which is transmitted to the drive the backlash will be driven (subtraction of position correction, see case 1).</p> <p>To realize that the monitored actual value is correct, this backlash correction is subtracted from the transmitted actual value (addition of the position correction, see case 2).</p>
<p><b>Effect in the Velocity Interface</b> (position control in TwinCAT)</p>	<p>Description / Effect:</p> <p>In the velocity interface no set position is transmitted to the drive, so correcting the set position would have no effect.</p> <p>By correcting the actual position which is transmitted from the encoder to the drive a position difference is created. The closed loop position controller in TwinCAT controls, by means of this position difference ("lag error") that the backlash is driven (addition of the position correction, see case 2).</p>

### 11.1.3 NC Implementing the TwinCAT Backlash Compensation

Overview of the effect of the backlash compensation differentiating by type of backlash and reference position (left or right edge):

- **Positive backlash:** Drive resp. external encoder has no backlash – backlash value is positive.
- **Negative backlash:** Additional external encoder has a backlash – backlash value is negative.

	<b>Implementation and effect of backlash compensation</b>
<p><b>Reference position at the left edge</b></p> 	<p><b>Backlash compensation in negative direction</b></p> <p><b>Positive direction:</b></p> <ul style="list-style-type: none"> <li>• No manipulation</li> </ul> <p><b>Negative direction:</b></p> <ul style="list-style-type: none"> <li>• Manipulation of the actual position +backlash</li> <li>• Manipulation of the set position -backlash</li> </ul>
<p><b>Reference position at the right edge</b></p> 	<p><b>Backlash compensation in positive direction</b></p> <p><b>Positive direction:</b></p> <ul style="list-style-type: none"> <li>• Manipulation of the actual position -backlash</li> <li>• Manipulation of the set position +backlash</li> </ul> <p><b>Negative direction:</b></p> <ul style="list-style-type: none"> <li>• No manipulation</li> </ul>

## **Trademark statements**

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

## **Third-party trademark statements**

BiSS is a trademark of IC Haus GmbH.

EnDat is a trademark of Dr. Johannes Heidenhain GmbH.

More Information:  
**[www.beckhoff.com/te1000](http://www.beckhoff.com/te1000)**

Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)

