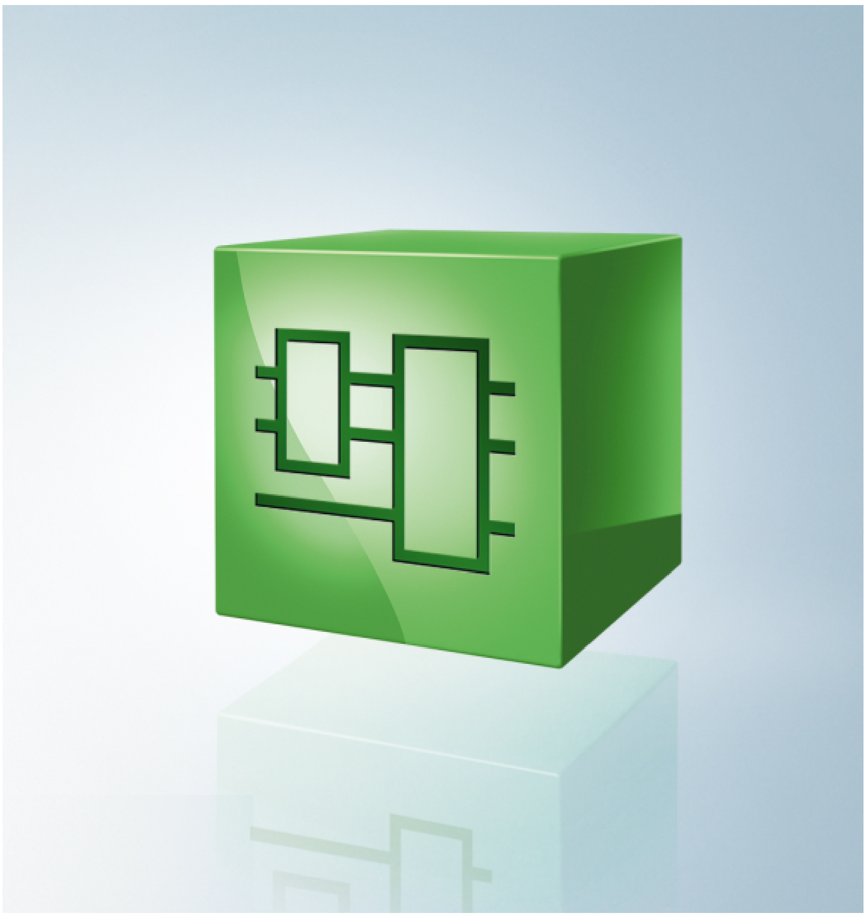


Manual

PLC Lib: Tc2_MC2

TwinCAT 3

Version: 1.8
Date: 2018-05-16

BECKHOFF

Table of contents

1 Foreword ... 7

1.1 Notes on the documentation ... 7

1.2 Safety instructions ... 8

2 Overview ... 9

3 State diagram ... 10

4 General rules for MC function blocks ... 13

5 Organization blocks ... 16

5.1 Axis functions ... 16

5.1.1 MC_Power ... 16

5.1.2 MC_Reset ... 17

5.1.3 MC_SetPosition ... 18

5.2 Status and parameter ... 19

5.2.1 MC_ReadActualPosition ... 19

5.2.2 MC_ReadActualVelocity ... 20

5.2.3 MC_ReadAxisComponents ... 21

5.2.4 MC_ReadAxisError ... 22

5.2.5 MC_ReadBoolParameter ... 23

5.2.6 MC_ReadParameter ... 24

5.2.7 MC_ReadParameterSet ... 25

5.2.8 MC_ReadStatus ... 26

5.2.9 MC_WriteBoolParameter ... 27

5.2.10 MC_WriteParameter ... 28

5.2.11 MC_WriteBoolParameterPersistent ... 29

5.2.12 MC_WriteParameterPersistent ... 30

5.3 Touch probe ... 31

5.3.1 MC_TouchProbe ... 31

5.3.2 MC_AbortTrigger ... 34

5.4 External set value generator ... 35

5.4.1 MC_ExtSetPointGenEnable ... 35

5.4.2 MC_ExtSetPointGenDisable ... 36

5.4.3 MC_ExtSetPointGenFeed ... 37

5.5 Special extensions ... 38

5.5.1 MC_PowerStepper ... 38

5.5.2 Notes on the MC_PowerStepper ... 39

5.5.3 MC_OverrideFilter ... 44

5.5.4 MC_SetOverride ... 45

5.5.5 MC_SetEncoderScalingFactor ... 46

5.5.6 MC_ReadDriveAddress ... 47

5.5.7 MC_PositionCorrectionLimiter ... 48

5.5.8 MC_SetAcceptBlockedDriveSignal ... 49

6 Motion function blocks ... 50

6.1 Point to point motion ... 50

6.1.1 MC_MoveAbsolute ... 50

6.1.2 MC_MoveRelative ... 51

6.1.3 MC_MoveAdditive ... 53

6.1.4 MC_MoveModulo ... 55

6.1.5 Notes on modulo positioning ... 57

6.1.6 MC_MoveVelocity ... 61

6.1.7 MC_MoveContinuousAbsolute ... 63

6.1.8 MC_MoveContinuousRelative ... 65
6.1.9 MC_Halt ... 66
6.1.10 MC_Stop ... 68

6.2 Superposition ... 69
6.2.1 MC_MoveSuperimposed ... 69
6.2.2 Application examples for MC_MoveSuperimposed ... 72
6.2.3 MC_AbortSuperposition ... 75

6.3 Homing ... 76
6.3.1 MC_Home ... 76

6.4 Manual motion ... 78
6.4.1 MC_Jog ... 78

6.5 Axis coupling ... 80
6.5.1 MC_GearIn ... 80
6.5.2 MC_GearInDyn ... 81
6.5.3 MC_GearOut ... 83
6.5.4 MC_GearInMultiMaster ... 84

6.6 Phasing ... 86
6.6.1 MC_HaltPhasing ... 86
6.6.2 MC_PhasingAbsolute ... 87
6.6.3 MC_PhasingRelative ... 89

7 Data types ... 91

7.1 Axis interface ... 91
7.1.1 AXIS_REF ... 91
7.1.2 NCTOPLC_AXIS_REF ... 92
7.1.3 NCTOPLC_AXIS_REF_OPMODE ... 94
7.1.4 NCTOPLC_AXIS_REF_STATE ... 95
7.1.5 NCTOPLC_AXIS_REF_STATE2 ... 96
7.1.6 NCTOPLC_AXIS_REF_STATE2_FLAGS ... 97
7.1.7 NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE ... 97
7.1.8 PLCTONC_AXIS_REF ... 98
7.1.9 PLCTONC_AXIS_REF_CTRL ... 99

7.2 Motion ... 99
7.2.1 E_JogMode ... 99
7.2.2 E_SuperpositionMode ... 100
7.2.3 MC_AxisStates ... 101
7.2.4 MC_BufferMode ... 101
7.2.5 MC_Direction ... 103
7.2.6 MC_HomingMode ... 104
7.2.7 ST_AxisOpModes ... 104
7.2.8 ST_SuperpositionOptions ... 105

7.3 Status and parameter ... 106
7.3.1 E_ReadMode ... 106
7.3.2 MC_AxisParameter ... 106
7.3.3 ST_AxisParameterSet ... 108
7.3.4 ST_AxisStatus ... 109
7.3.5 ST_DriveAddress ... 110
7.3.6 ST_PowerStepperStruct ... 111
7.3.7 E_AxisPositionCorrectionMode ... 111

7.4 External set value generator ... 111
7.4.1 E_PositionType ... 111

7.5 Touch probe ... 112
7.5.1 TRIGGER_REF ... 112
7.5.2 MC_TouchProbeRecordedData ... 113

8 Global constants 115
8.1 Library version 115

9 Appendix 116
9.1 Summary of examples 116
9.2 Support and Service 117

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared 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 prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

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

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER	**Serious risk of injury!** Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.
⚠ WARNING	**Risk of injury!** Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.
⚠ CAUTION	**Personal injuries!** Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.
❗ Attention	**Damage to the environment or devices** Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.
ℹ Note	**Tip or pointer** This symbol indicates information that contributes to better understanding.

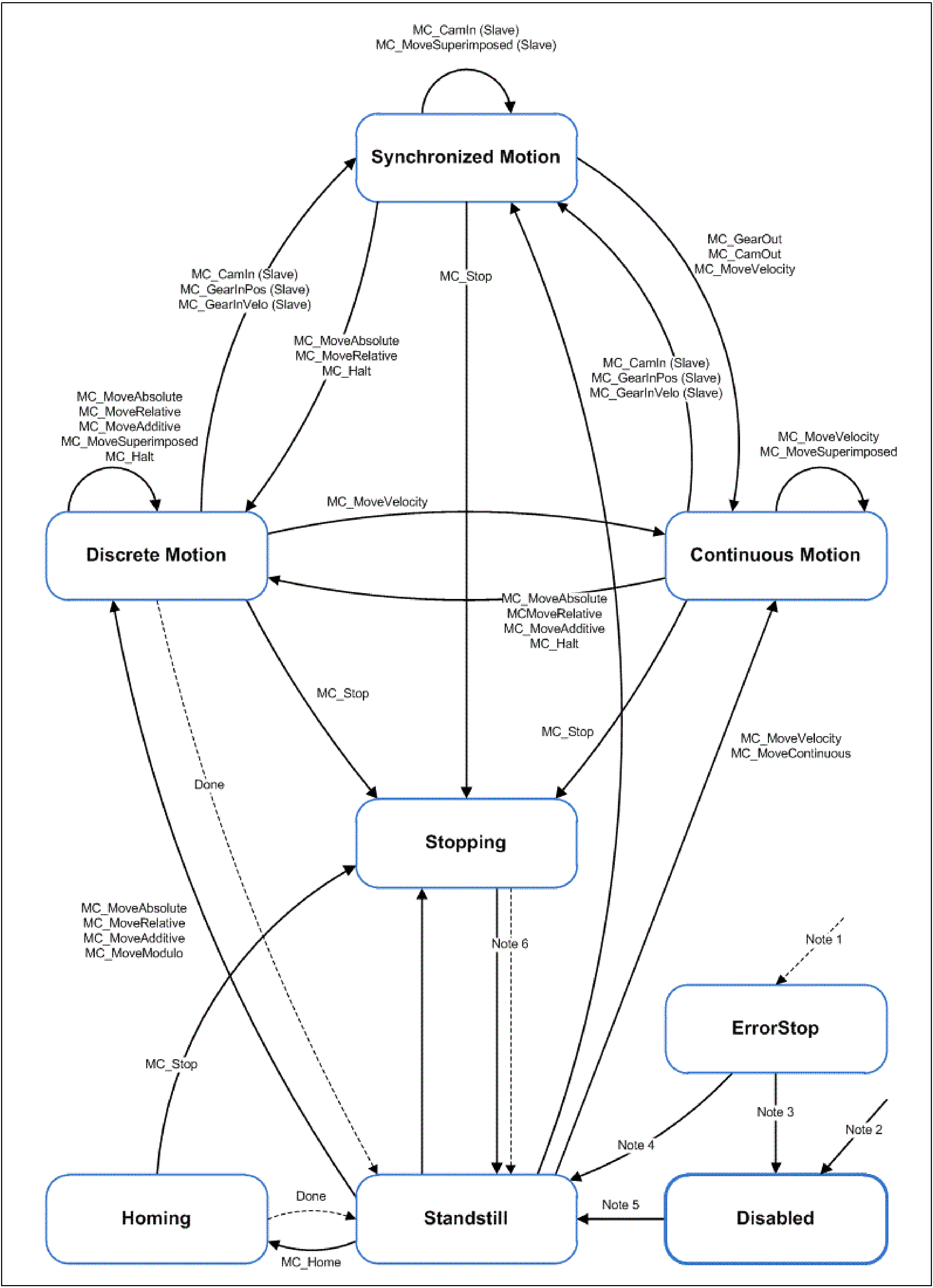
2 Overview

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).

Note	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 complex 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 "Continous 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 status 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. Furthermore, only one of the outputs "Active", "Error", "Done" and "CommandAborted" can be TRUE at the same time.

An exception is the motion command MC_Stop [▶ 68]. The command sets "Done" to TRUE as soon as the axis is stopped. However, "Busy" and "Active" remain TRUE since the axis is locked. The axis is only unlocked and "Busy" and "Active" are set to FALSE when Execute 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.

Resetting "Execute" during command execution 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 "Execute" is triggered more than once while a command is executed, the function block will not execute further commands and will not provide any feedback.

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. "Distance", in contrast, is a relative measurement, 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/s2
Deceleration	mm/s2
Jerk	mm/s3

Error handling

All function blocks have two error outputs for indicating errors during command execution. "Error" indicates the error and "ErrorID" returns a supplementary error number. The outputs "Done", "InVelocity", "InGear" and "InSync" denote a successful command execution and are not set if "Error" becomes 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 [▸ 17].
- 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 [▸ 17]. 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

"CommandAborted" is set if a command is interrupted through 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 with a positive edge at the "Execute" input if "Busy" 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 "Busy" is TRUE, the function block must be called cyclically for the command to be executed.

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 [▸ 26] 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, as long as "Valid" is TRUE, the data can be constantly updated.

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 [▶ 101]

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 [▶ 61] 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 [▶ 15]), 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.

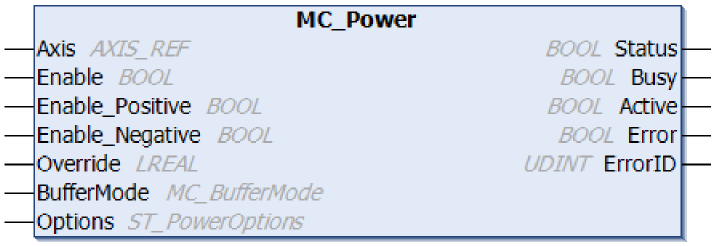
Slave axes

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 [▶ 50] 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



MC_Power activates software enable for 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.

Note	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 *)
END_VAR
```

Enable: General software enable for the axis.

Enable_Positive: Advance movement enable in positive direction. Only takes effect if Enable = TRUE.

Enable_Negative: Advance movement enable in negative direction. Only takes effect if Enable = TRUE.

Override: Velocity override in % for all motion commands. (0 ≤ Override ≤ 100.0)

BufferMode: This is evaluated when "Enable" is reset (see also MC_BufferMode [▸ 101]). 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.

See also: General rules for MC function blocks [▸ 13]

Outputs

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

Status: TRUE when the axis is ready for operation.

Busy: TRUE, as long as the function block is called with Enable = TRUE.

Active: Indicates that the command is executed.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▸ 13]

Inputs/outputs

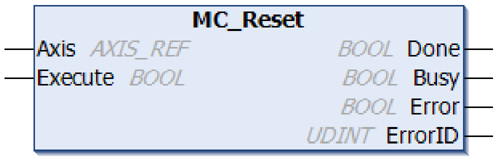
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

5.1.2 MC_Reset



MC_Reset resets 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
```

Execute: The command is executed with a positive edge.

Outputs

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

Done: TRUE if the reset was successfully executed.

Busy: 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 command. At the same time, one of the outputs "Done" or "Error" is set.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

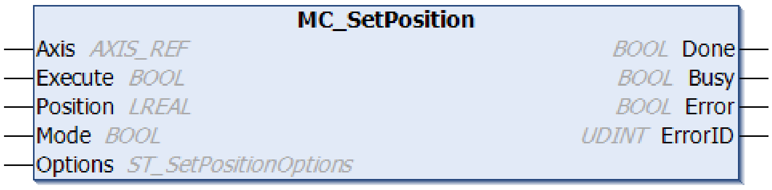
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

5.1.3 MC_SetPosition



MC_SetPosition 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 such that any lag error that may exist is retained. The switch Options.ClearPositionLag can be used to clear the lag error.

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

Inputs

```
VAR_INPUT
    Execute  : BOOL;
    Position : LREAL;
    Mode     : BOOL; (* RELATIVE=True, ABSOLUTE=False (Default) *)
    Options  : ST_SetPositionOptions;
END_VAR
```

Execute: The command is executed with a positive edge.

Position: Position value to which the axis position is to be set. In absolute mode the actual position is set to this value, in relative mode it is shifted by this value.

Mode: 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

- **ClearPositionLag:** Can optionally be used to set the set and actual positions to the same value. In this case the lag error is cleared.
- **SelectEncoderIndex:** Can optionally be set if an axis with several encoders is used and the position of a particular encoder is to be set (Options.EncoderIndex).
- **EncoderIndex:** Indicates the encoder (0..n) if SelectEncoderIndex = TRUE.

See also: General rules for MC function blocks [▸ 13]

Outputs

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

Done: TRUE if the position was set successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

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

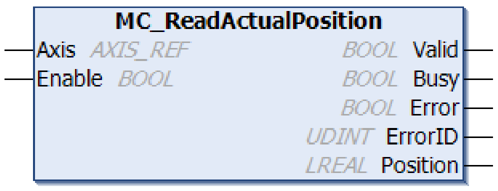
Axis: Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2 Status and parameter

5.2.1 MC_ReadActualPosition



The current axis position can be read with the MC_ReadActualPosition function block.

Inputs

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

Enable: TRUE as long as the command is executed.

Outputs

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

Valid: TRUE if the output "Position" has a valid value.

Busy: Indicates that the function block is active.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Position: Current axis position

Inputs/outputs

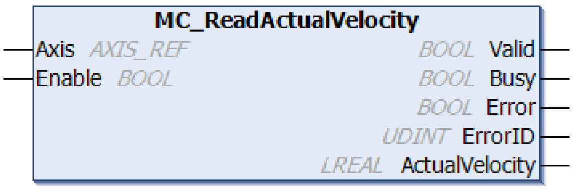
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.2 MC_ReadActualVelocity



The current axis velocity can be read with the function block MC_ReadActualVelocity.

Inputs

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

Enable: TRUE as long as the command is executed.

Outputs

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

Valid: TRUE if the output "ActualVelocity" has a valid value.

Busy: Indicates that the function block is active.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

ActualVelocity: Current axis velocity

Inputs/outputs

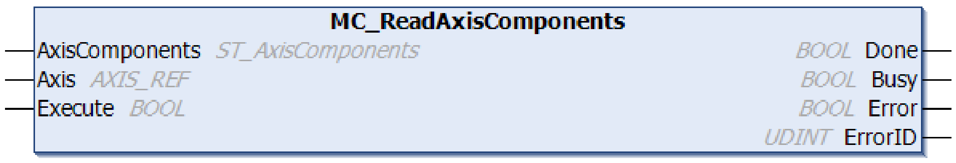
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.3 MC_ReadAxisComponents



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

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

Inputs

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

Execute: The command is executed with a positive edge.

Outputs

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

Done: TRUE if the parameters were read successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

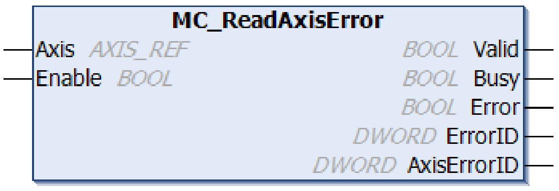
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.4 MC_ReadAxisError



MC_ReadAxisError reads the axis error of an axis.

Inputs

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

Enable: If Enable = TRUE, the axis error is output at the "AxisErrorID" output.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Valid: TRUE if the error signaled at the "AxisErrorID" output is valid.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

AxisErrorID: Error number for the axis

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

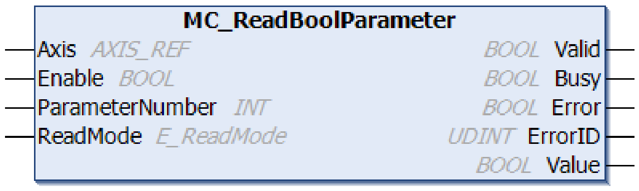
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.5 MC_ReadBoolParameter



The function block MC_ReadBoolParameter is used to read a boolean axis parameter.

NOTE! 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 *)
EEND_VAR
```

Enable: TRUE as long as the command is executed.

ParameterNumber: Number of the parameter to be read. (Type MC_AxisParameter [▶ 106])

ReadMode: Read mode of the parameter to be read (once or cyclic). (Type: E_ReadMode [▶ 106])

Outputs

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

Valid: TRUE if the value signaled at output "Value" is valid.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Value: Shows the read Boolean value.

Inputs/outputs

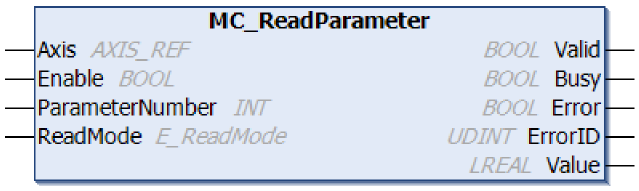
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.6 MC_ReadParameter



The function block MC_ReadParameter is used to read an axis parameter.

NOTE! 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
```

Enable: TRUE as long as the command is executed.

ParameterNumber: Number of the parameter to be read. (Type MC_AxisParameter [▸ 106])

ReadMode: Read mode of the parameter to be read (once or cyclic). (Type: E_ReadMode [▸ 106])

Outputs

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

Valid: TRUE if the value signaled at output "Value" is valid.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Value: Shows the read value.

Inputs/outputs

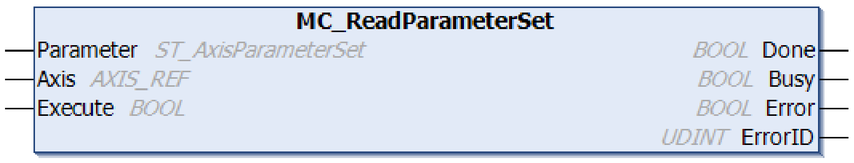
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

5.2.7 MC_ReadParameterSet



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

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

Inputs

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

Execute: The command is executed with a positive edge.

Inputs/outputs

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

Parameter: Parameter data structure into which the parameters are read. (Type: ST_AxisParameterSet [▶ 108])

Axis: 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. (Type: AXIS_REF [▶ 91])

Outputs

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

Done: TRUE if the parameters were read successfully.

Busy: 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.

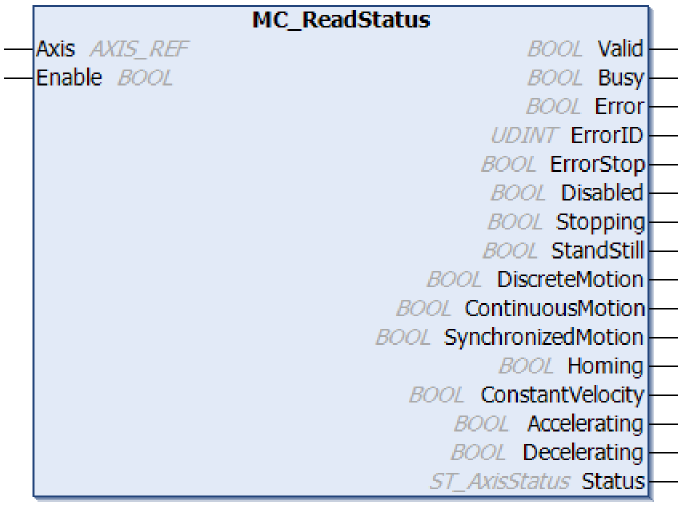
Error: TRUE, if an error occurs.

ErrorID: 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)	Tc2_MC2

5.2.8 MC_ReadStatus



MC_ReadStatus 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 [▸ 91]) 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
```

Enable: if Enable = TRUE, the axis operating state is updated with each call of the function block.

See also: General rules for MC function blocks [▸ 13]

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
```

Valid: Indicates that the axis operating state indicated at the other outputs is valid.

Busy: Indicates that the function block is active.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

ErrorStop: Axis status according to the PlcOpen state diagram [▶ 10]

Disabled: Axis status according to the PlcOpen state diagram [▶ 10]

Stopping: Axis status according to the PlcOpen state diagram [▶ 10]

StandStill: Axis status according to the PlcOpen state diagram [▶ 10]

DiscreteMotion: Axis status according to the PlcOpen state diagram [▶ 10]

ContinousMotion: Axis status according to the PlcOpen state diagram [▶ 10]

SynchronizedMotion: Axis status according to the PlcOpen state diagram [▶ 10]

Homing: Axis status according to the PlcOpen state diagram [▶ 10]

ConstantVelocity: The axis is moving with constant velocity.

Acceleration: The axis accelerates.

Decelerating: The axis decelerates.

Status: Extended status data structure [▶ 109] with additional status information. (Type: ST_AxisStatus [▶ 109])

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

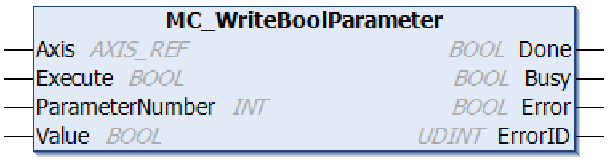
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.9 MC_WriteBoolParameter



Boolean parameters for the axis can be written with the function block MC_WriteBoolParameter.

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

Inputs

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

Execute: The command is executed with a positive edge.

ParameterNumber: Number of the parameter to be written. (Type MC_AxisParameter [▶ 106])

Value: Boolean value that is written.

Outputs

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

Done: TRUE if the parameters were written successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

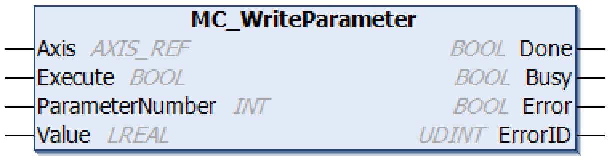
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.2.10 MC_WriteParameter



Axis parameters can be written with the function block MC_WriteParameter.

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

Inputs

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

Execute: The command is executed with a positive edge.

ParameterNumber: Number of the parameter to be written. (See MC_AxisParameter [▶ 106])

Value: LREAL value that is written.

Outputs

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

Done: TRUE if the parameters were written successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

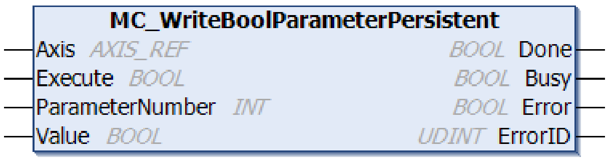
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

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

Inputs

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

Execute: The command is executed with a positive edge.

ParameterNumber: Number of the parameter to be written. (Type MC_AxisParameter [▶ 106])

Value: Boolean value that is written.

Outputs

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

Done: TRUE if the parameters were written successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

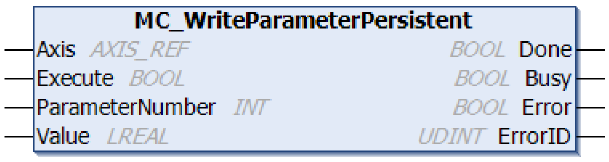
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

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

Inputs

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

Execute: The command is executed with a positive edge.

ParameterNumber: Number of the parameter to be written. (See MC_AxisParameter [▸ 106])

Value: LREAL value that is written.

Outputs

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

Done: TRUE if the parameters were written successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

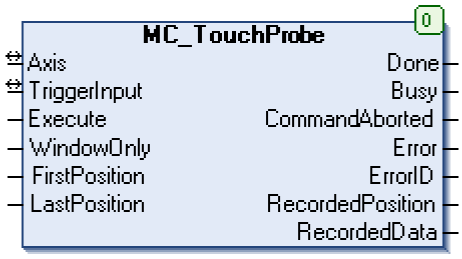
Axis: Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state. (Type: AXIS_REF [▸ 91])

Requirements

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

NOTE! 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.

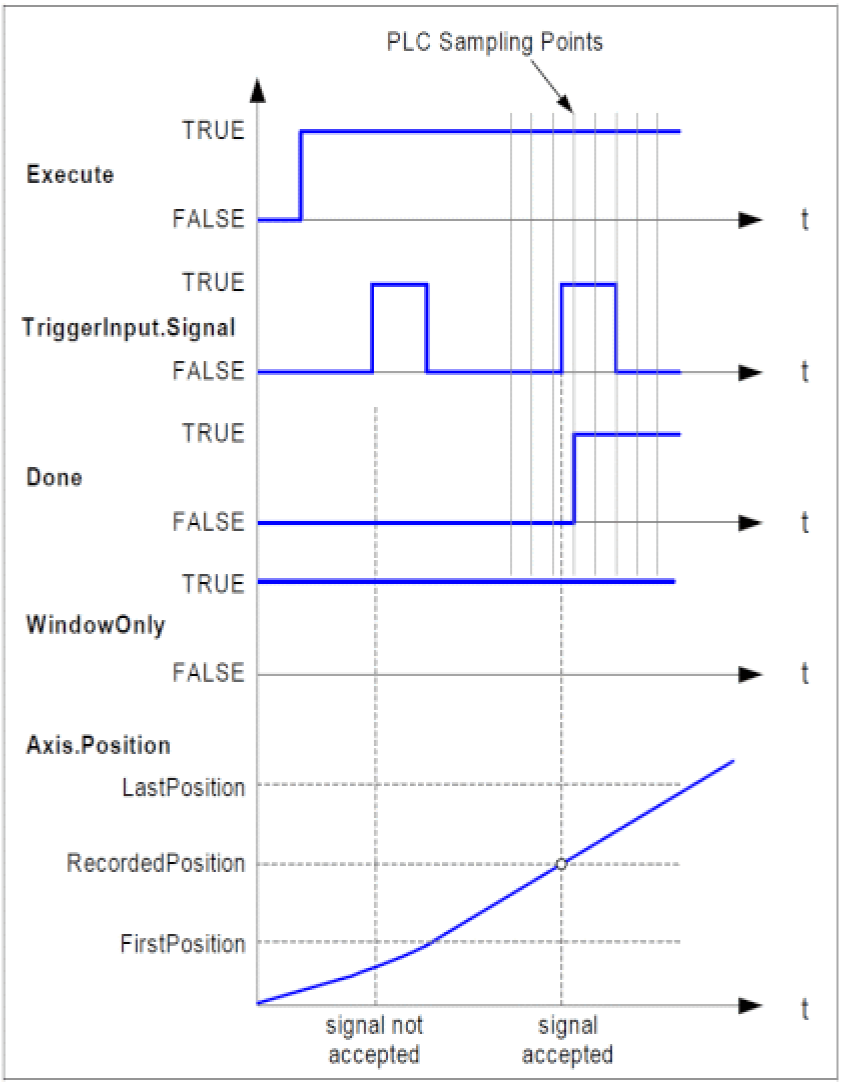
The digital trigger signal is wired into this hardware and, independently of the PLC cycle, triggers the recording of the current axis position.

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.

Note

After a measuring probe cycle has been initiated by a positive edge on the "Execute" input, this is only terminated if the outputs "Done", "Error" or" CommandAborted" become TRUE. If the operation is to be aborted in the meantime, the function block MC_AbortTrigger [▶ 34] must be called with the same TriggerInput [▶ 112] data structure. Otherwise no new cycle can be initiated.

Signal curve



Inputs

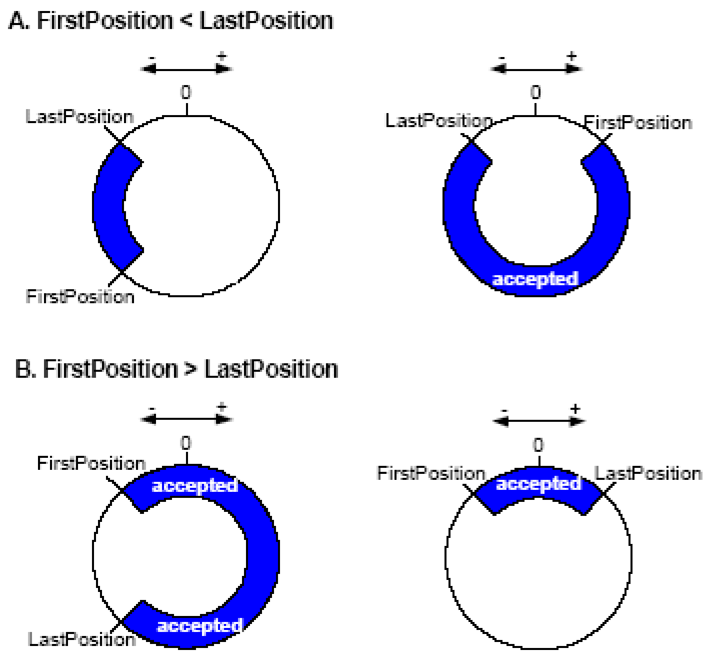
```
VAR_INPUT
    Execute       : BOOL;
    WindowOnly    : BOOL;
    FirstPosition : LREAL;
    LastPosition  : LREAL;
END_VAR
```

Execute: The command is executed with a positive edge, and the external position latch is activated.

WindowOnly: If WindowOnly = TRUE, only one position within the window between "FirstPosition" and "LastPosition" is captured. 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 TriggerInput [▸ 112] 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).

FirstPosition: 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 TriggerInput [▸ 112] data structure.

LastPosition: Final 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 TriggerInput [▸ 112] data structure.



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

Outputs

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

Done: TRUE if an axis position was successfully detected. The position is sent to the output "RecordedPosition".

Busy: TRUE as soon as the function block is active. FALSE if it is in the default state.

CommandAborted: Becomes TRUE if the process is interrupted by an external event, e.g. by the call up of MC_AbortTrigger [▸ 34].

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

RecordedPosition: Axis position recorded at the point in time of the trigger signal

RecordedData: 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
```

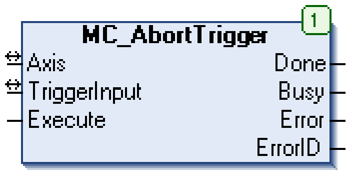
Axis: Axis data structure (type: AXIS_REF [▶ 91])

TriggerInput: Data structure for describing the trigger source (type: TRIGGER_REF [▶ 112]). This data structure must be parameterized before the function block is called for the first time.

Requirements

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

5.3.2 MC_AbortTrigger



The function block MC_AbortTrigger cancels a probe cycle started by MC_TouchProbe. MC_TouchProbe initiates a measuring 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 measuring probe cycle has completed successfully, it is not necessary to call up this function block.

Inputs

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

Execute: The command is executed with the positive edge and the external position latch is deactivated.

Outputs

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

Done: TRUE as soon as the measuring probe cycle has been successfully terminated.

Busy: TRUE as soon as the function block is active. FALSE if it is in the default state.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Axis: Axis data structure (type: AXIS_REF [▸ 91])

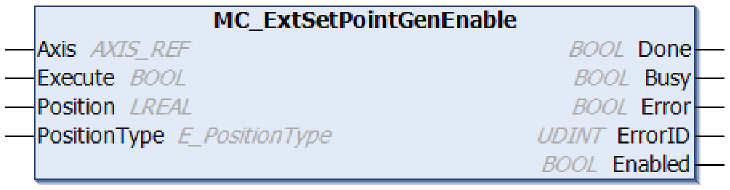
TriggerInput: Data structure for describing the trigger source (type: TRIGGER_REF [▸ 112]). This data structure must be parameterized before the function block is called for the first time.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86)	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 function block MC_ExtSetPointGenEnable. 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.

See also: MC_ExtSetPointGenDisable [▸ 36] and MC_ExtSetPointGenFeed [▸ 37]

Inputs

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

Execute: The command is executed with a positive edge.

Position: Position for target position monitoring. Setting of this position does not mean that the axis moves to this position, for which only the external setpoint generator is responsible. Instead, setting this position activates the target position monitoring. The flag InTargetPosition becomes TRUE when this position is reached.

PositionType: POSITIONTYPE_ABSOLUTE or POSITIONTYPE_RELATIVE (type: E_PositionType [▸ 111])

Outputs

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

Done: TRUE if the command was executed successfully.

Busy: TRUE as soon as the function block is active. FALSE if it is in the default state.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Enabled: Shows the current state of the external setpoint generator, regardless of the function execution.

Inputs/outputs

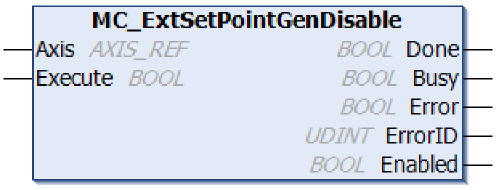
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.4.2 MC_ExtSetPointGenDisable



The external setpoint generator of an axis can be switched off with the function block MC_ExtSetPointGenDisable. 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.

See also: MC_ExtSetPointGenEnable [▶ 35] and MC_ExtSetPointGenFeed [▶ 37]

Inputs

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

Execute: The command is executed with a positive edge.

Outputs

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

Done: TRUE if the command was executed successfully.

Busy: TRUE as soon as the function block is active. FALSE if it is in the default state.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Enabled: Shows the current state of the external setpoint generator, regardless of the function execution.

Inputs/outputs

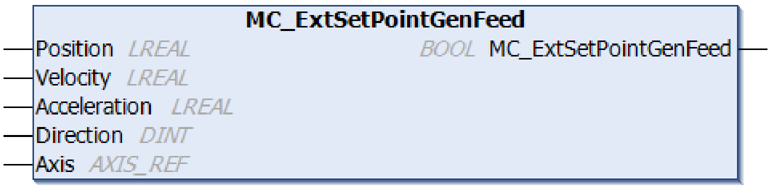
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.4.3 MC_ExtSetPointGenFeed



The MC_ExtSetPointGenFeed function is used to feed set values from an external setpoint generator into an axis. The function copies the data instantaneously into the cyclic axis interface (fExtSetPos, fExtSetVelo, fExtSetAcc and nExtSetDirection). The function result of MC_ExtSetPointGenFeed 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.

See also: MC_ExtSetPointGenEnable [▶ 35] and MC_ExtSetPointGenDisable [▶ 36]

Inputs

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

Position: Set position from an external setpoint generator

Velocity: Set velocity from an external setpoint generator

Acceleration: Set acceleration from an external setpoint generator

Direction: 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
```

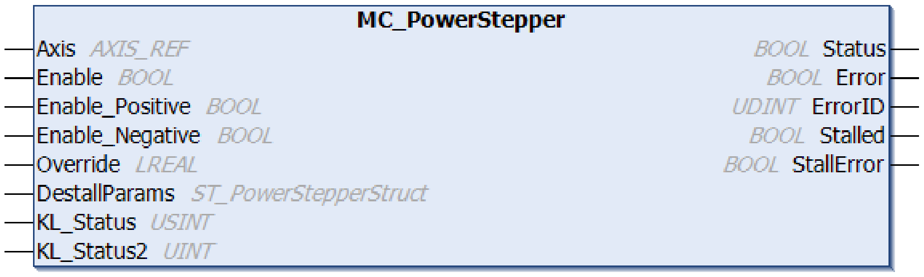
Axis: Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state. (Type: AXIS_REF [▶ 91])

Requirements

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

5.5 Special extensions

5.5.1 MC_PowerStepper



The enables for an axis are set with the function block MC_PowerStepper. Function block MC_Power is used internally for this purpose. The MC_PowerStepper also detects the stall situations that occur in stepper motors if they are overloaded, and offers suitable counter measures. The status bits of a KL2531 or KL2541 terminal are monitored, and the errors indicated there are reported to the NC.

See also: Notes on the MC_PowerStepper [▶ 39]

Inputs

```
VAR_INPUT
    Enable          : BOOL;
    Enable_Positive : BOOL;
    Enable_Negative : BOOL;
    Override        : LREAL;
    DestallParams   : ST_PowerStepperStruct;
    KL_Status       : USINT;
    KL_Status2      : UINT;
END_VAR
```

Enable: NC controller enable for the axis.

Enable_Positive: NC feed enable in positive direction.

Enable_Negative: NC feed enable in negative direction.

Override: Override value in percent (e.g. 68.123%)

DestallParams: Structure in which the functions of the function block are enabled and their work rules are defined. (Type: ST_PowerStepperStruct [▶ 111])

KL_Status: Status byte of a terminal of type KL2531 or KL2541.

KL_Status2: Status word of a terminal of type KL2531 or KL2541.

Outputs

```
VAR_OUTPUT
    Status     : BOOL;
    Error      : BOOL;
    ErrorID    : UDINT;
    Stalled    : BOOL;
    StallError : BOOL;
END_VAR
```

Status: TRUE if the enables were set successfully.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Stalled: No description

StallError: No description

Inputs/outputs

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

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.5.2 Notes on the MC_PowerStepper

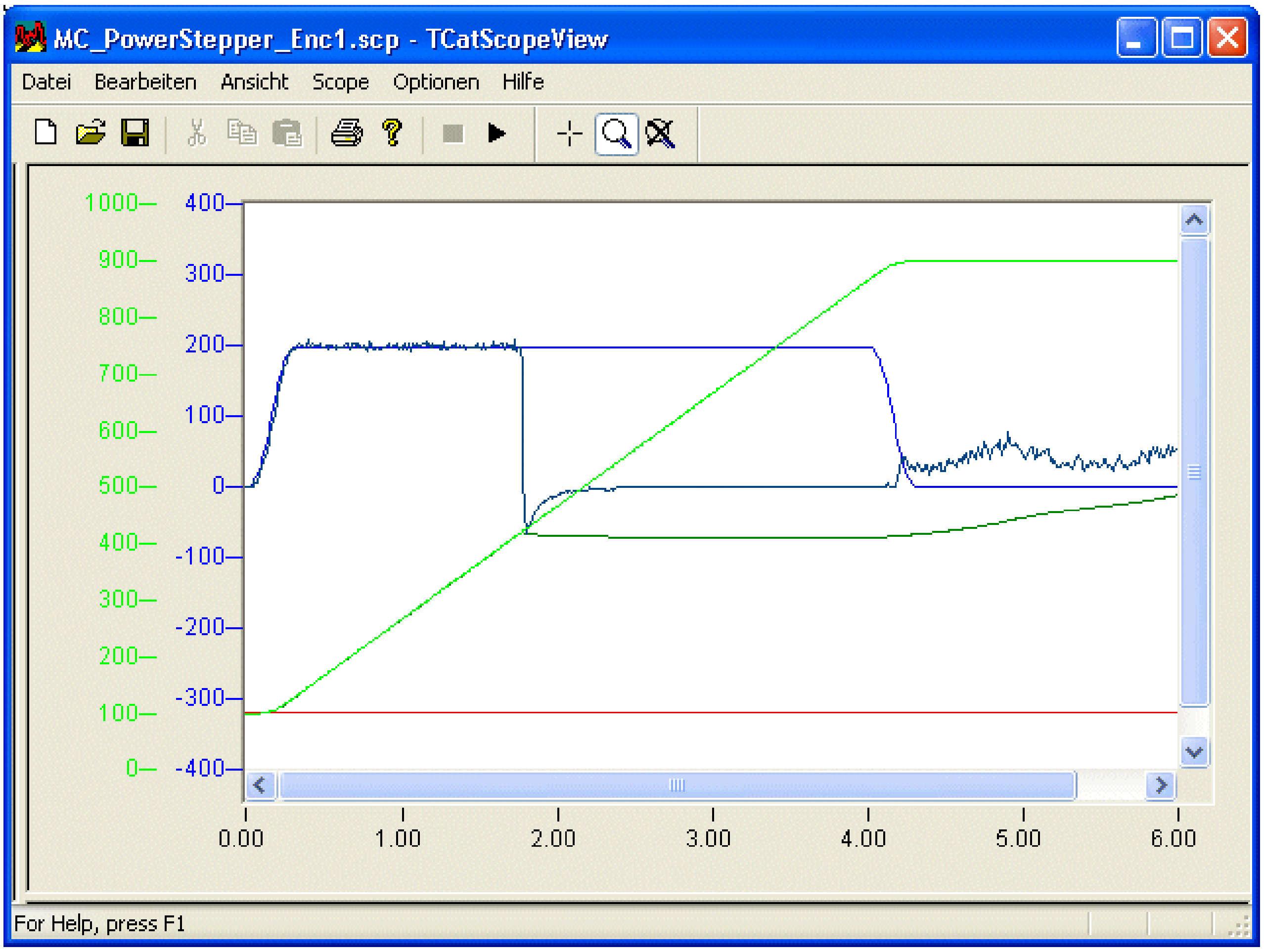
The enables and the override for an axis are set with the function block MC_PowerStepper [▶ 38]. Function block MC_Power [▶ 16] is used internally for this purpose. The MC_PowerStepper also detects the stall situations that occur in stepper motors if they are overloaded, and offers suitable counter measures. The status bits of a KL2531 or KL2541 terminal are monitored, and the errors indicated there are reported to the NC.

Stepper motor and synchronous servo: similarities and differences

Both types of motor use an electromagnetic field and the field of a permanent magnet in order to generate a driving force through their interaction. Whereas, however, the servomotor makes use of an expensive system of sensors in order to make specific adjustments to the alignments of the fields (current supplied dependent on the rotor position), this position-dependent control is not used for the stepper motor. This can lead to significant cost savings. However, the motor can be moved by an external force beyond the point of maximum torque generated by it. Since the electromagnetically generated field does not take this into account, it generates a falling restoring torque as the deflection continues to rise. As a result of this, if the excursion is more than the one half of one pole step then the corrective torque will change sign, pushing the motor on in the direction of the next pole position. Depending on the prevailing conditions, the motor can latch in the new orientation (the so-called step loss has occurred) or repeat the same procedure. The latter is called "stalling" and occurs mainly when the motor is supplied with current at typical frequencies of the active motion.

Example:

A stepper motor fitted with an encoder is operated with the NC PTP using the parameters typical for servos.

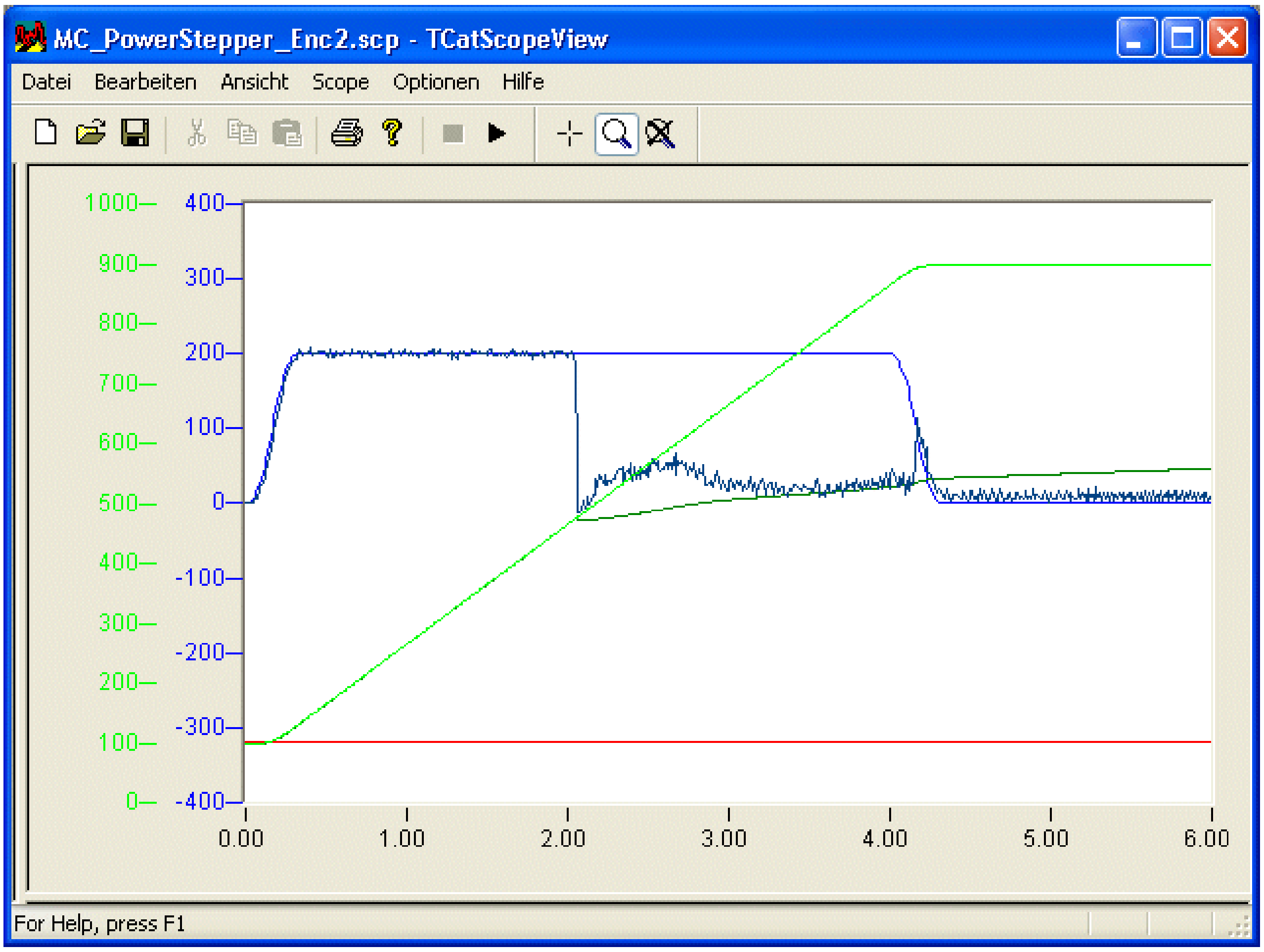


🟥 bStalled
🟦 SetVelo
🟩 ActualPos
🟦 ActualVelo
🟩 SetPos

After about 1.8 seconds, the axis is briefly blocked by an obstacle. Although the axis is then able to move freely, it is unable to follow the set value of the velocity, but will remain stationary, making considerable noise but without generating any detectable torque. Only after the profile generator has reached its target does the total of the set and correction velocities fall. The motor starts moving unevenly. Even a low load torque can prevent this. The only solution is to execute the command MC_Reset [▶ 17] and an appropriate settling time. The axis must then be restarted by the application. Various status bits in the axis interface respond. This has to be taken into account in the application, as this can lead to faulty responses in the machine cycle.

First corrective step: Controller limitation

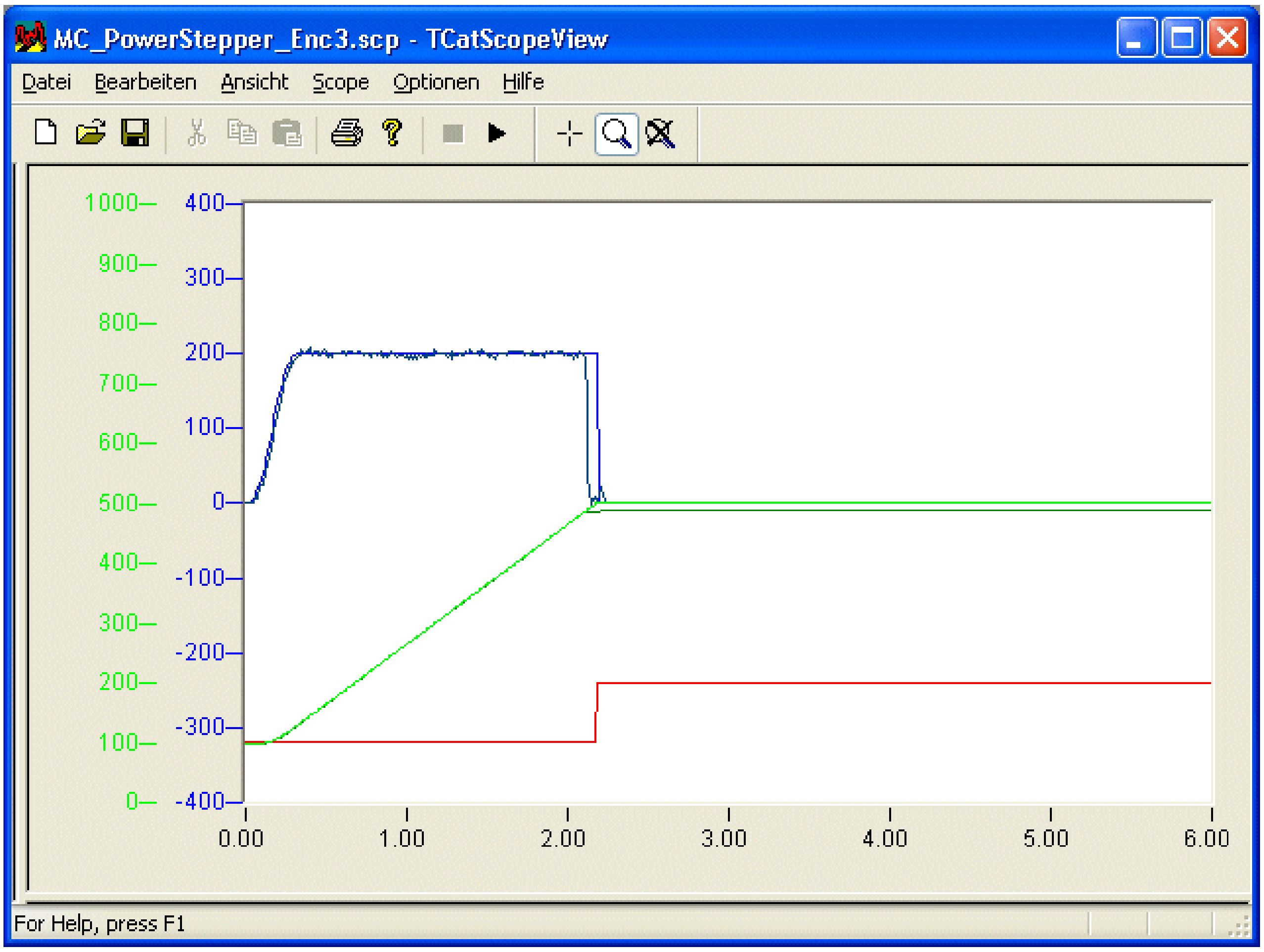
If, in the situation described above, the output of the position controller is limited to a sufficiently small value such as, for instance, 2%, the following pattern results.



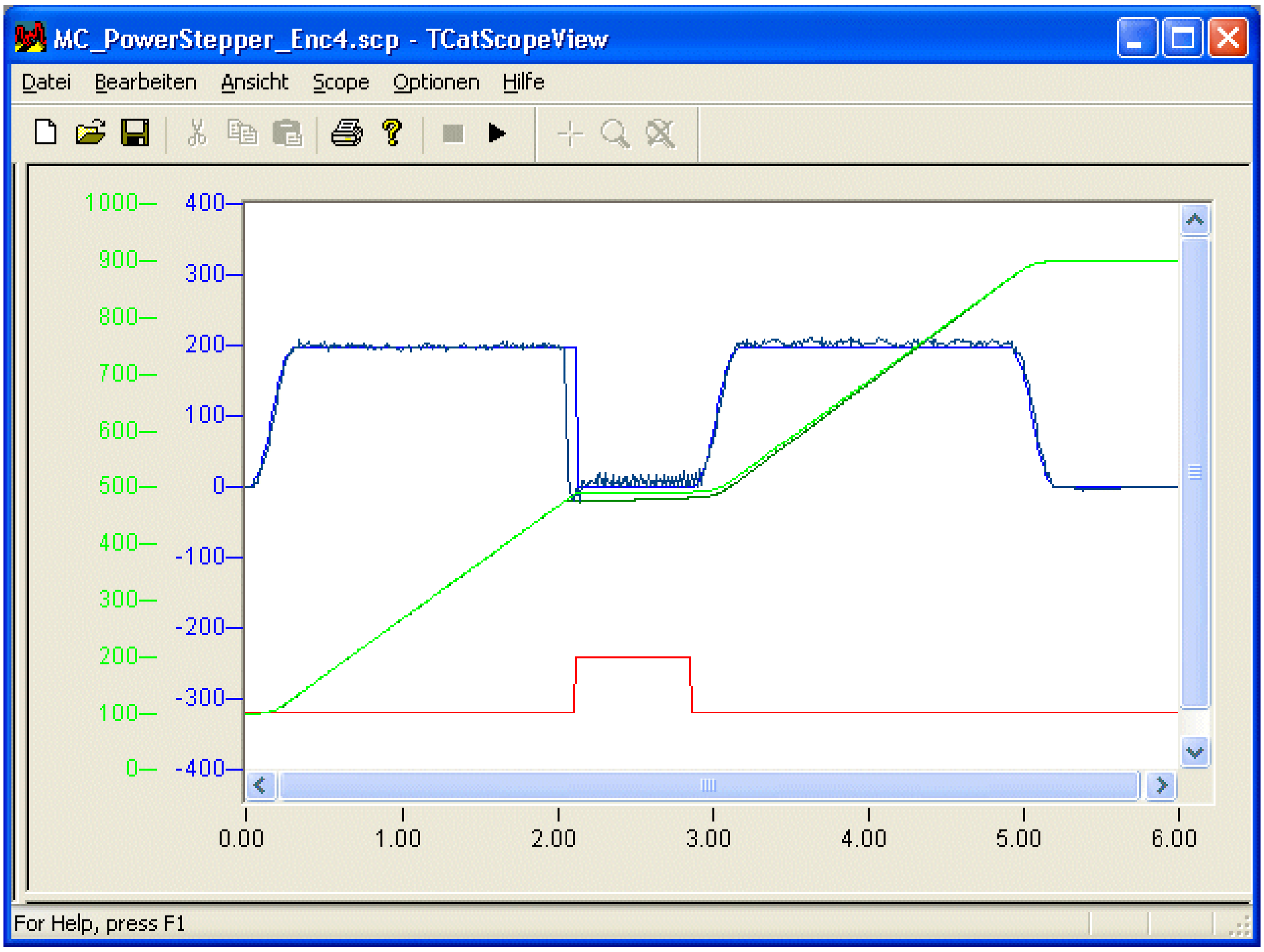
Here again, for the remaining period of profile generation, the set speed is too high for the stepper motor to be able to follow the set movement properly. Once the end of the set profile has been reached, the position controller guides the stepper motor to the target at a low operating frequency, which it can follow without a ramp. During this process it generates a relatively high torque. The time required for this corrective measure is, however, very long.

Second measure: Detection and handling of stall situations using an encoder

In order to be able to take appropriate counter-measures, it is first necessary to detect the problem. The following pattern results if an MC_PowerStepper function block is used. It has a parameter structure of type ST_PowerStepperStruct [▶ 111], in which "PwStDetectMode_Lagging" is entered as the "DestallDetectMode". The function block uses the lag error of the axis as a basis for decision making, and the limit value and the filter time of the lag monitoring to be deactivated here from the NC axis data. In this example, "PwStMode_SetError" is entered as "DestallMode". Initially, the only difference from the lag error alarm is the different error code.



If "PwStMode_UseOverride" is entered as the "DestallMode", the function block MC_PowerStepper uses the override to stop the profile immediately. Because, however, this halt does not abort the profile, yet does at the same time prevent the end of the profile from being reached, there is no effect on any status bits. The controller output, limited here to 2%, brings the axis to within the lag error threshold of the current set position of the profile. Then the override is then returned to the value specified by the application.



As a result, a considerably larger part of the overall profile is moved at the specified velocity, and the target position is reached correctly. The status information for the axis is generated correctly.

Combinations of stall detection and handling

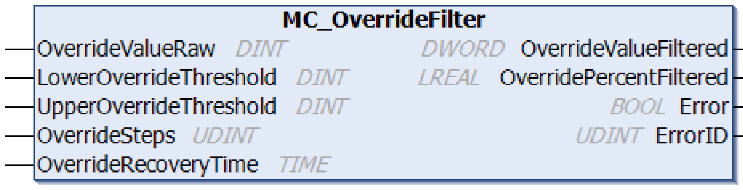
The following table illustrates the combinations of the supported modes for stall detection and handling.

	PwStMode_SetError	**PwStMode_SetErrNonRef**	**PwStMode_UseOverride**
PwStDetectMode_ Encoderless	Useful for axes without encoder that are not referenced.	Useful for axes without encoder that are referenced with the aid of the terminal's pulse counter and, for instance, an external sensor.	not suitable
PwStDetectMode_ Lagging	The resultant behavior largely corresponds to lag monitoring.	not useful	Useful for axes with encoder.

Requirements

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

5.5.3 MC_OverrideFilter


MC_OverrideFilter
OverrideValueRaw DINT
LowerOverrideThreshold DINT
UpperOverrideThreshold DINT
OverrideSteps UDINT
OverrideRecoveryTime TIME
DWORD OverrideValueFiltered
LREAL OverridePercentFiltered
BOOL Error
UDINT ErrorID

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.

Note Due to the stepping of the override output value ("OverrideValueFiltered"), the filtered override may become zero for 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
```

OverrideValueRaw: Raw, unfiltered override value (e.g. a voltage value of an analog input terminal).

LowerOverrideThreshold: Lower bound that limits the raw override value.

UpperOverrideThreshold: Upper bound that limits the raw override value.

OverrideSteps: The predefined steps (override resolution)

OverrideRecoveryTime: 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
```

OverrideValueFiltered: Filtered override value in digits (the data type matches the override in the cyclic axis interface 0..1000000).

OverridePercentFiltered: Filtered override value in percent (0..100%).

Error: TRUE, if an error occurs.

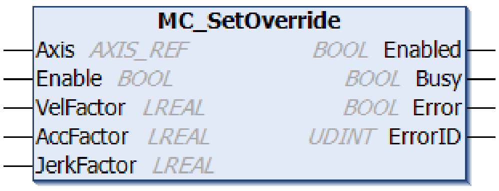
ErrorID: If the error output is set, this parameter supplies the error number.

Possible error number	Possible causes
MC_ERROR_PARAMETER_NOT_CORRECT	• OverrideSteps <= 1 • LowerOverrideThreshold >= UpperOverrideThreshold

Requirements

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

5.5.4 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
```

Enable: TRUE as long as the command is executed.

VelFactor: Velocity override factor

AccFactor: Not supported

JerkFactor: Not supported

Outputs

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

Enabled: The parameterized override is set.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

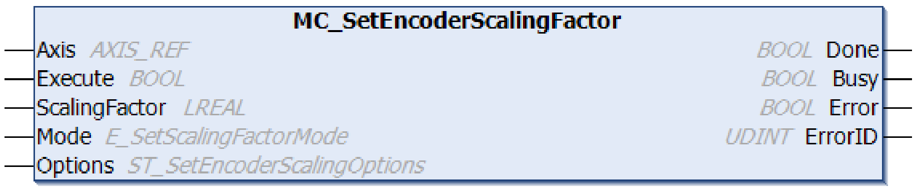
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.5.5 MC_SetEncoderScalingFactor


MC_SetEncoderScalingFactor
Axis AXIS_REF
Execute BOOL
ScalingFactor LREAL
Mode E_SetScalingFactorMode
Options ST_SetEncoderScalingOptions
BOOL Done
BOOL Busy
BOOL Error
UDINT ErrorID

MC_SetEncoderScalingFactor changes the scaling factor for 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
```

Execute: The command is executed with a positive edge.

ScalingFactor: 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: The scaling factor can be set in absolute or relative mode (ENCODERSCALINGMODE_ABSOLUTE, ENCODERSCALINGMODE_RELATIVE).
In absolute mode counting starts at the origin of the axis coordinate system, resulting in a position discontinuity if 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

- **SelectEncoderIndex:** Can be set if an axis with several encoders is used and the position of a particular encoder is to be set (Options.EncoderIndex).
- **EncoderIndex:** Indicates the encoder (0..n) if "SelectEncoderIndex" is TRUE.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE if the position was set successfully.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

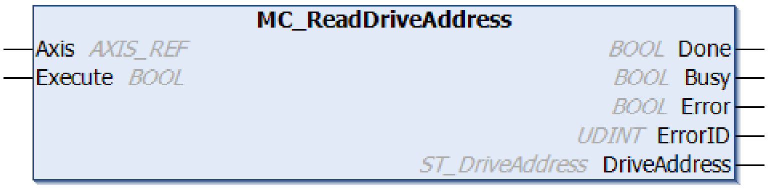
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.5.6 MC_ReadDriveAddress



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
```

Execute: The command is executed with a positive edge.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE if the command was executed without errors.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

DriveAddress: ADS access data of a drive device connected to the axis. (Type: ST_DriveAddress [▶ 110])

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

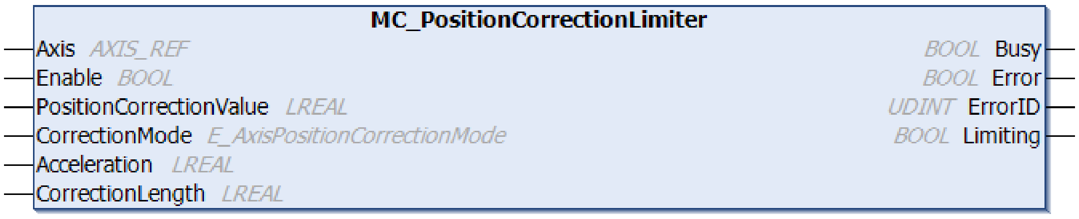
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

5.5.7 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.

NOTE! To use this function block successfully, the Position Correction parameter must be activated in the System Manager.

Inputs

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

Enable: Activates continuous writing of the correction value "PositionCorrectionValue". It must be TRUE as long as new correction values are to be accepted.

PositionCorrectionValue: Correction value to be added to the actual value of the axis

CorrectionMode: Depending on this mode, the correction value "PositionCorrectionValue" is written either directly or filtered. For a detailed description see E_AxisPositionCorrectionMode [▸ 111].

Acceleration: Depending on the "CorrectionMode" the maximum acceleration to reach the new correction value is specified here. In the case of PositionCorrectionMode_Fast [▸ 111] this value has a direct effect on the position delta by PLC-tick.
Maximum permissible correction value Position delta = acceleration * (PLC cycle time)^2.
If acceleration is parameterized as 0.0, the position correction is not limited.

CorrectionLength: Becomes active when the "CorrectionMode" matches PositionCorrectionMode_FullLength [▸ 111]. A change in the "PositonCorrectionValue" is distributed over this correction length.

Outputs

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

Busy: TRUE as soon as the function block is active. FALSE when it returns to its original state.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Limiting: TRUE if the required correction value "PositionCorrectionValue" is not yet fully accepted.

Inputs/outputs

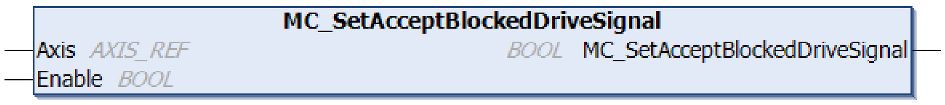
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

5.5.8 MC_SetAcceptBlockedDriveSignal


MC_SetAcceptBlockedDriveSignal
Axis AXIS_REF
Enable BOOL
BOOL MC_SetAcceptBlockedDriveSignal

There are situations in which a drive no longer follows the set NC values, 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 set NC values.

Notes:

- See also bit 8 of the ControlDWord in AXIS_REF.
- 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
```

Enable: NC controller enable for the axis.

Inputs/outputs

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

Axis: 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. (Type: AXIS_REF [▶ 91])

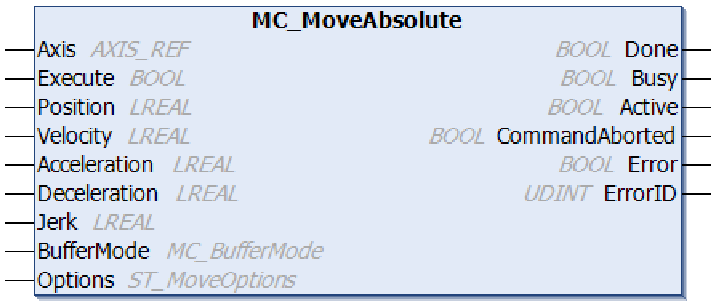
Requirements

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

6 Motion function blocks

6.1 Point to point motion

6.1.1 MC_MoveAbsolute



MC_MoveAbsolute starts positioning to an absolute target 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.

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 [▶ 55] 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 uncoupling 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
```

Execute: The command is executed with a positive edge.

Position: Absolute target position to be used for positioning.

Velocity: Maximum travel velocity (>0).

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

BufferMode: Is evaluated if the axis already executes another command (type: MC_BufferMode [▶ 101]). 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, only the BufferMode "Aborting" is possible. 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE when the target position has been reached.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

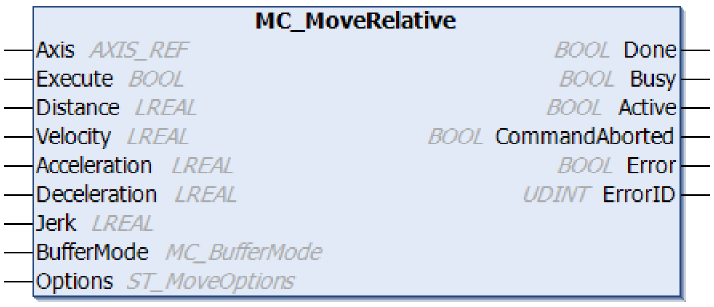
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

6.1.2 MC_MoveRelative



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 uncoupling 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
```

Execute: The command is executed with a positive edge.

Distance: Relative distance to be used for positioning.

Velocity: Maximum travel velocity (>0).

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

BufferMode: Is evaluated if the axis already executes another command (type: MC_BufferMode [▶ 101]). 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, only the BufferMode "Aborting" is possible. 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE when the target position has been reached.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

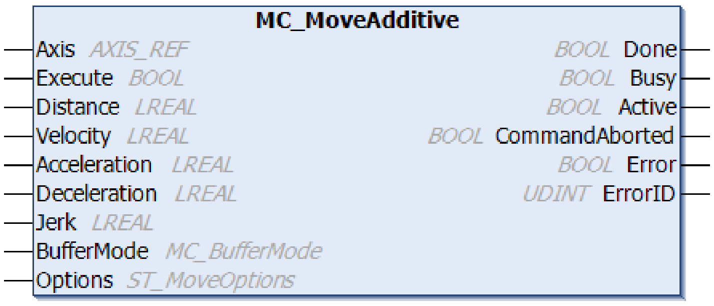
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

6.1.3 MC_MoveAdditive



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.

NOTE! 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
```

Execute: The command is executed with a positive edge.

Distance: Relative distance to be used for positioning.

Velocity: Maximum travel velocity (>0).

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

BufferMode: Is evaluated if the axis already executes another command (see MC_BufferMode [▶ 101]). 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE when the target position has been reached.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

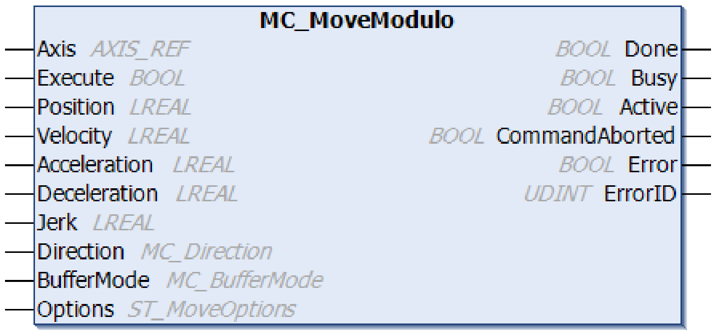
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86)	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 uncoupling 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 direction of movement cannot be reversed by MC_MoveModulo, i.e. the target can only be reached in the current direction. 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 MC_MoveAbsolute [▸ 50] function block.

Note	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 [▸ 92].

See also: Further information on modulo movements [▶ 57]

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
```

Execute: The command is executed with a positive edge.

Position: 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: Maximum travel velocity (>0).

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

Direction: Positive or negative direction of travel (type: MC_Direction [▶ 103]). If the axis is started during a motion, the direction may not be reversed.

BufferMode: Is evaluated if the axis already executes another command (see MC_BufferMode [▶ 101]). MC_MoveModulo 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE when the target position has been reached.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▸ 13]

Inputs/outputs

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

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

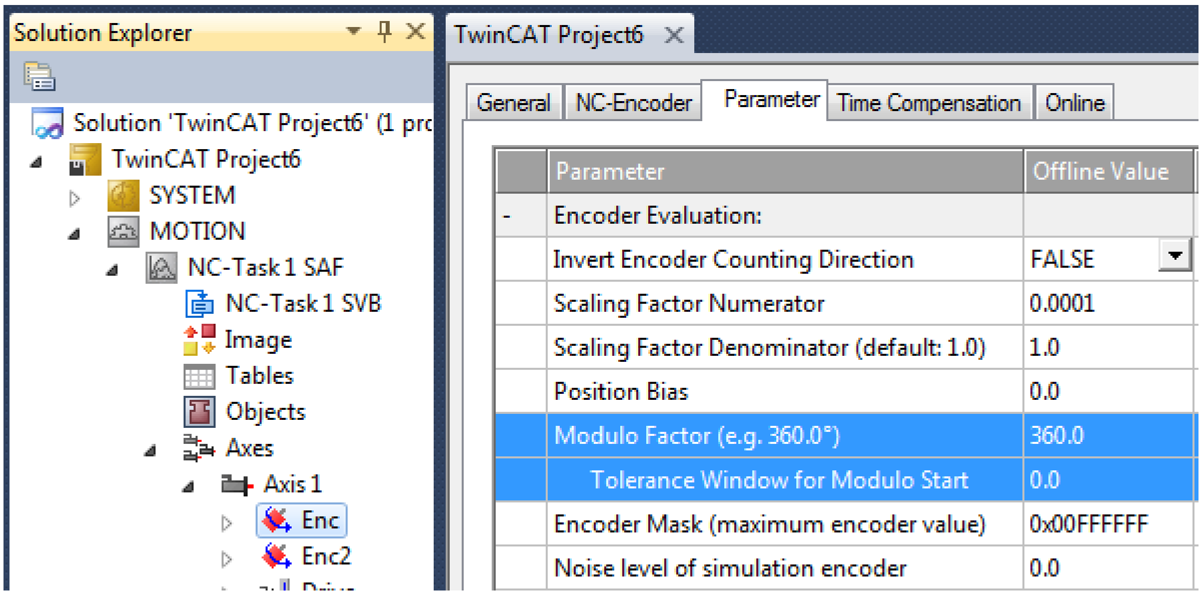
Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86)	Tc2_MC2

6.1.5 Notes on modulo positioning

Modulo positioning (MC_MoveModulo [▸ 55]) is possible irrespective of the axis type. If 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 ±∞. 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.



Parameter	Offline Value
Encoder Evaluation:	
Invert Encoder Counting Direction	FALSE
Scaling Factor Numerator	0.0001
Scaling Factor Denominator (default: 1.0)	1.0
Position Bias	0.0
Modulo Factor (e.g. 360.0°)	360.0
Tolerance Window for Modulo Start	0.0
Encoder Mask (maximum encoder value)	0x00FFFFFF
Noise level of simulation encoder	0.0

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 ± tolerance value). A detailed description of the tolerance window is provided below.

Special features of axis resets

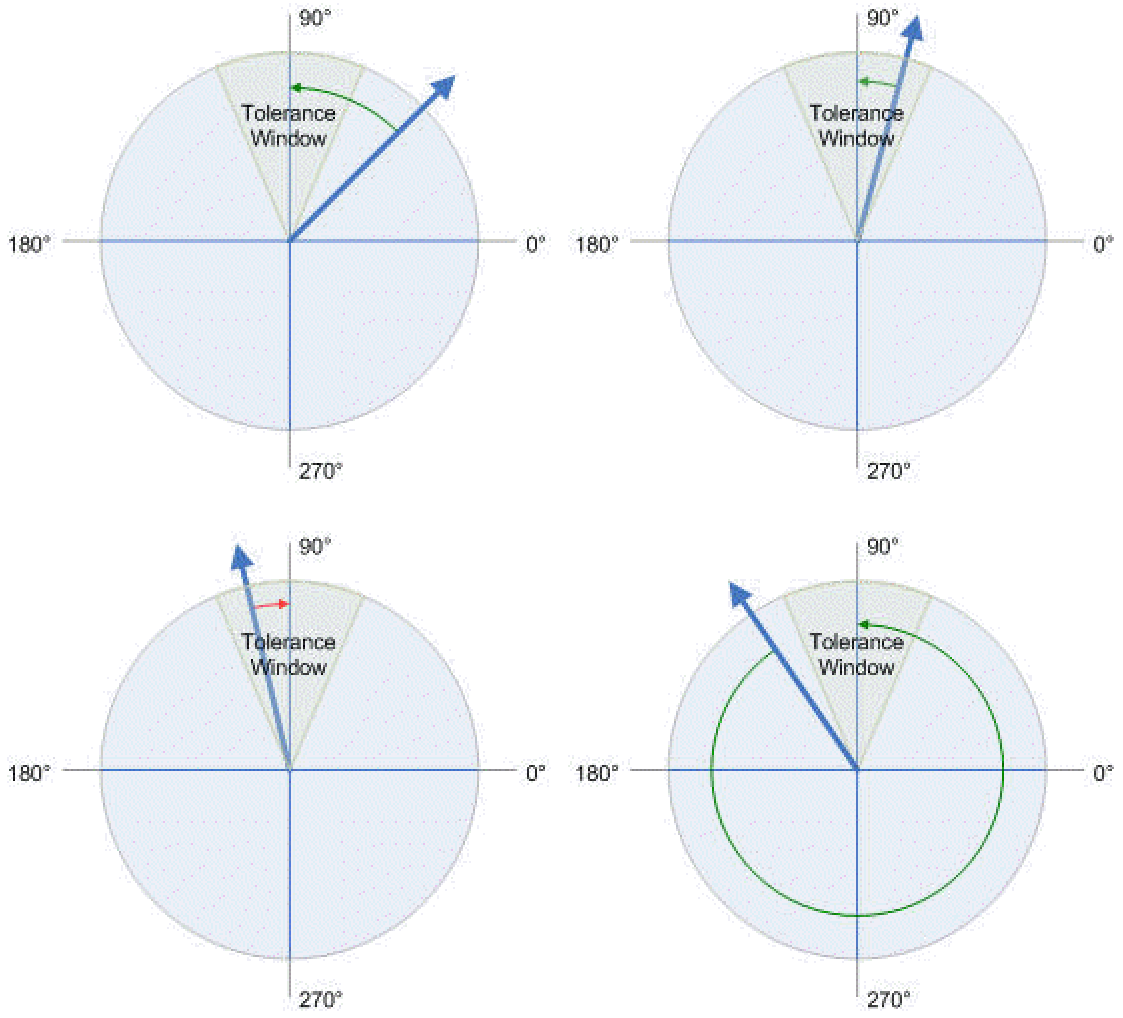
Axis positioning always refers to the set position. The set position of an axis is normally the target position of the last travel command. An axis reset (MC_Reset [▸ 17], controller activation with MC_Power [▸ 16]) can lead to a set position that is different from that expected by the user, because 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 ≤; 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 [▸ 103]). 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.

NOTE! 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 TF
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 TF
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 TF
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 TF
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	1170.00	90.00	outside TF

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 TF
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 TF
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 TF
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 TF
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 TF
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 TF
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 TF

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).

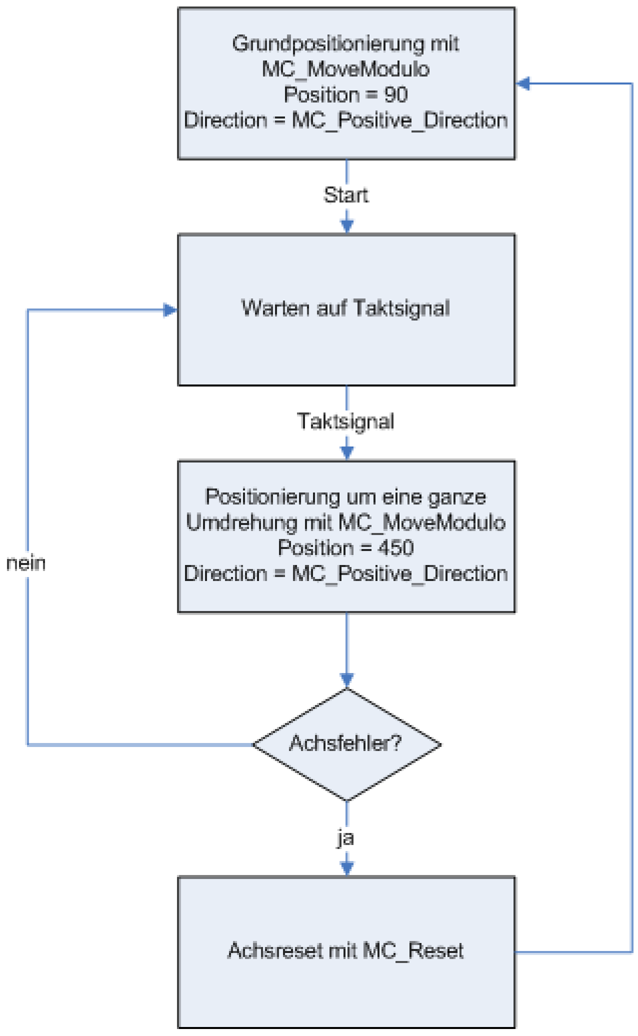
It is not advisable to perform order calculations based on the actual modulo position, which is available in the cyclical axis interface (ModuloActPos and ModuloActTurns). Due to the greater or lesser control deviation of the axis, errors in the programmed sequence, such as undesired rotations, could 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 [▸ 55] 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 [▸ 55]) 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 [▸ 17] 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.

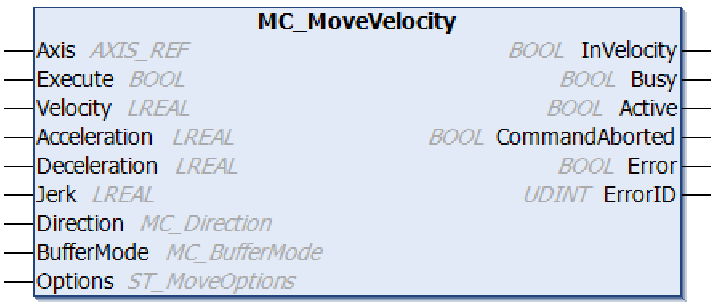

Grundpositionierung mit MC_MoveModulo Position = 90 Direction = MC_Positive_Direction
Start
Warten auf Taktsignal
Taktsignal
Positionierung um eine ganze Umdrehung mit MC_MoveModulo Position = 450 Direction = MC_Positive_Direction
nein
Achsfehler?
ja
Achsreset mit MC_Reset

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)	Tc2_MC2

6.1.6 MC_MoveVelocity



MC_MoveVelocity starts a continuous movement with 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 uncoupling 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
```

Execute: The command is executed with a positive edge.

Velocity: Maximum travel speed (>0).

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

Direction: Positive or negative direction of travel (type: MC_Direction [▸ 103]).

BufferMode: Is evaluated if the axis already executes another command (type: MC_BufferMode [▸ 101]). 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, only the BufferMode "Aborting" is possible. 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▸ 13]

Outputs

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

InVelocity: TRUE as soon as the constant velocity is reached and FALSE if the velocity deviates.

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

Busy: 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 "InVelocity", "CommandAborted" or "Error" is set.

Active: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

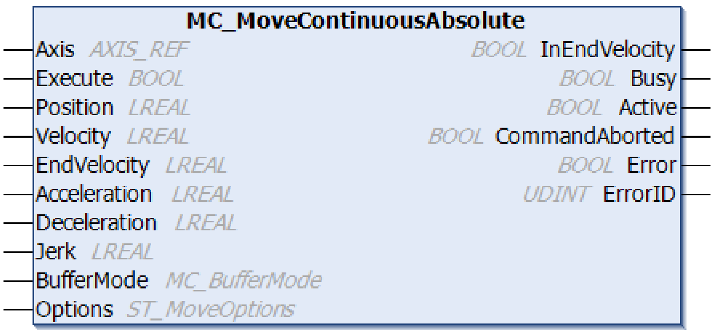
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

6.1.7 MC_MoveContinuousAbsolute



MC_MoveContinuousAbsolute starts positioning to an absolute target position and monitors the axis movement over the whole 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.

NOTE! 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
```

Execute: The command is executed with a positive edge.

Position: Absolute target position

Velocity: Maximum velocity for the movement to the target position (>0).

EndVelocity: End velocity to be maintained once the target position has been reached.

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

BufferMode: Is evaluated if the axis already executes another command (type: MC_BufferMode [▸ 101]). 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▸ 13]

Outputs

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

InEndVelocity: TRUE when the target position has been reached.

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

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▸ 13]

Inputs/outputs

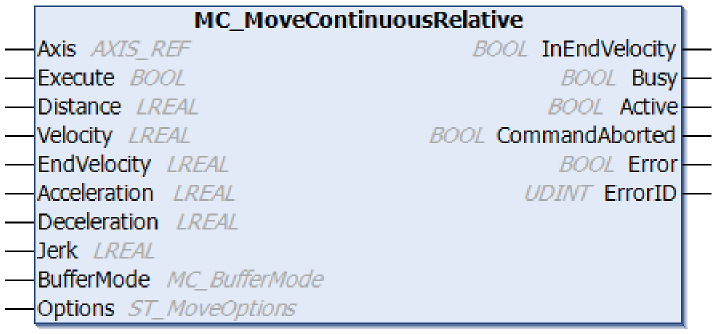
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

6.1.8 MC_MoveContinuousRelative



MC_MoveContinuousRelative starts positioning by a relative distance and monitors the axis movement over the whole 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.

NOTE! 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
```

Execute: The command is executed with a positive edge.

Distance: Relative distance to be used for positioning.

Velocity: Maximum velocity for the movement over the "Distance" (>0).

EndVelocity: End velocity to be maintained after the relative "Distance".

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

BufferMode: Is evaluated if the axis already executes another command (type: MC_BufferMode [▸ 101]). 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: Data structure containing additional, rarely used parameters. The input can normally remain open.

See also: General rules for MC function blocks [▸ 13]

Outputs

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

InEndVelocity: TRUE when the target position has been reached.

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

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▸ 13]

Inputs/outputs

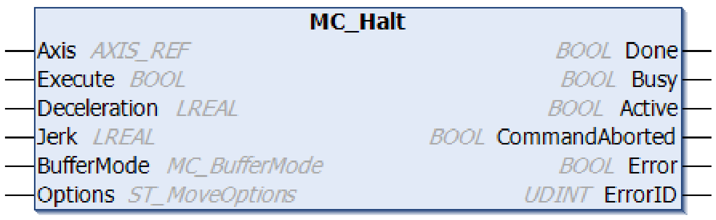
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Requirements

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

6.1.9 MC_Halt



MC_Halt stops an axis with a defined braking ramp.

In contrast to MC_Stop [▸ 68], the axis is not locked against further movement commands. The axis can therefore be restarted through a further command during the braking 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 uncoupling 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
```

Execute: The command is executed with a positive edge.

Deceleration: Deceleration. If the value is ≤ 0, the deceleration parameterized with the last Move command is used. For safety reasons MC_Halt and MC_Stop [▶ 68] cannot be executed with weaker dynamics than the currently active travel command. The parameterization is adjusted automatically, if necessary.

Jerk: Jerk. If the value is ≤ 0, the jerk parameterized with the last Move command is used.
For safety reasons MC_Halt and MC_Stop [▶ 68] cannot be executed with weaker dynamics than the currently active travel command. The parameterization is adjusted automatically, if necessary.

BufferMode: Is evaluated if the axis already executes another command (see MC_BufferMode [▶ 101]). 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" has no effect, if the command is executed when the system is at a standstill. The blending modes "MC_BlendingNext" and "MC_BlendingLow" do not change the last target position, although they can result in 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: Currently not implemented

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE if the axis has been stopped and is stationary.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: Becomes TRUE, if the command could not be fully executed. The running command may have been followed by a Move command.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

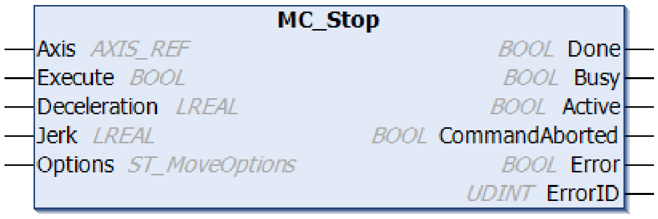
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

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

6.1.10 MC_Stop



MC_Stop stops an axis with a defined braking ramp and locks it 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 completely. A few cycles are required to release the axis after a negative 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 [▶ 17].

Alternatively, the axis can be stopped with MC_Halt [▶ 66] without locking. MC_Halt is preferable for normal movements.

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_MoveStop then automatically leads to uncoupling of the axis, after which the command is executed.

Inputs

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

Execute: The command is executed with a positive 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 completely.

Deceleration: Deceleration. If the value is ≤ 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: Jerk. If the value is ≤ 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: Currently not implemented

See also: General rules for MC function blocks [▶ 13]

Outputs

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

Done: TRUE if the axis has been stopped and is stationary.

Busy: 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: 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: TRUE if the command could not be executed completely.

Error: TRUE, if an error occurs

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

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

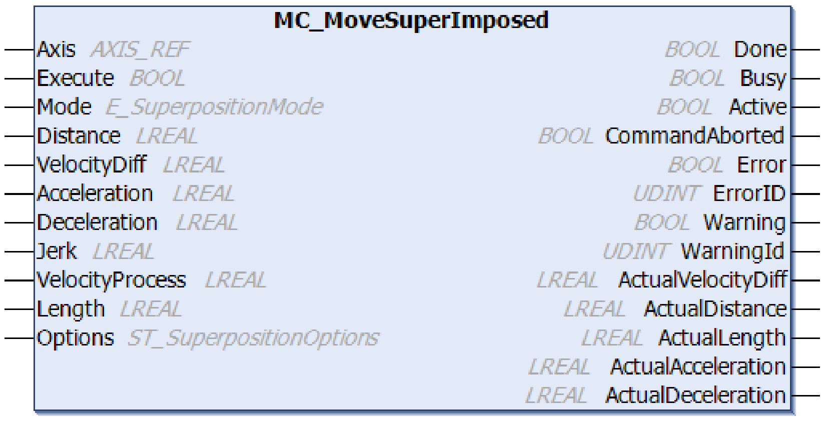
Axis: Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state. (Type: AXIS_REF [▶ 91])

Requirements

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

6.2 Superposition

6.2.1 MC_MoveSuperimposed



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 will precede or follow the other axis as determined 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 [▶ 72]

Inputs

```
VAR_INPUT
    Execute         : BOOL;  (* B *)
    Mode            : E_SuperpositionMode;
    Distance        : LREAL;  (* B *)
    VelocityDiff    : LREAL;  (* E *)
    Acceleration    : LREAL;  (* E *)
    Deceleration    : LREAL;  (* E *)
    Jerk            : LREAL;  (* E *)
    VelocityProcess : LREAL;  (* V *)
    Length          : LREAL;  (* V *)
    Options         : ST_SuperpositionOptions;  (* V *)
END_VAR
```

Execute: The command is executed with a positive edge.

Mode: Determines the type of the superimposed motion. (Type: E_SuperpositionMode [▶ 100])

Distance: 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: 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":

- Distance > 0 (axis accelerates)
  VelocityDiff = maximum velocity – basic velocity
- Distance < 0 (axis decelerates)
  VelocityDiff = basic velocity

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

VelocityProcess: Mean process velocity in the axis (>0). If the basic velocity during superposition is constant, the set axis velocity can be specified.

Length: Distance over which the superimposed movement is available. The "Mode" parameter defines how this distance is interpreted.

Options: Data structure containing additional, rarely used parameters. The input can normally remain open. (Type: ST_SuperpositionOptions [▶ 105])

- **AbortOption:** Defines the behavior when the subordinate motion stops. The superimposed movement can be aborted or continued later.

See also: General rules for MC function blocks [▶ 13]

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
```

Done: TRUE if the superimposed movement was successfully completed.

Busy: 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: Indicates that the command is executed.

CommandAborted: Becomes TRUE, if the command was aborted by another command and could therefore not be completed.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Warning: TRUE if the action cannot be executed completely.

WarningID: The function block returns warning 4243_{hex} (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: Actual velocity difference during the superimposed motion (ActualVelocityDiff ≤ VelocityDiff).

ActualDistance: 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: Actual travel during superimposed motion (ActualLenght ≤ Length).

ActualAcceleration: Actual acceleration of the superimposed movement (ActualAcceleration ≤ Acceleration).

ActualDeceleration: Actual deceleration of the superimposed movement (ActualDeceleration ≤ Deceleration).

See also: General rules for MC function blocks [▶ 13]

Inputs/outputs

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

Axis: 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. (Type: AXIS_REF [▶ 91])

Requirements

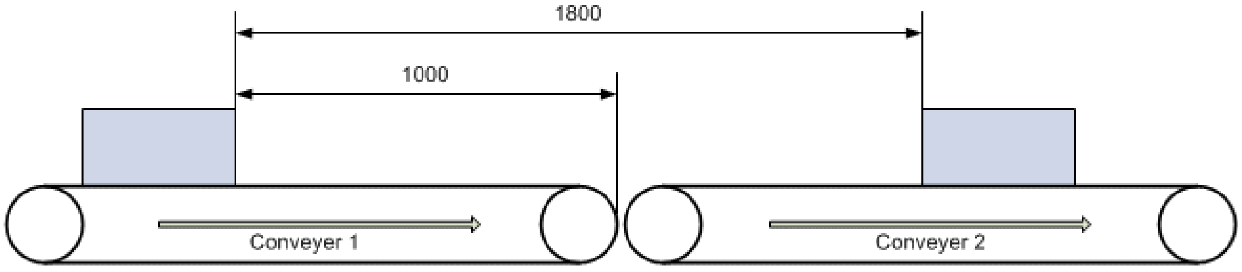
Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86)	Tc2_MC2

6.2.2 Application examples for MC_MoveSuperimposed

The function block MC_MoveSuperimposed [▶ 69] 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 speed and the current set velocity.

For parameterization of function block MC_MoveSuperimposed [▶ 69] 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

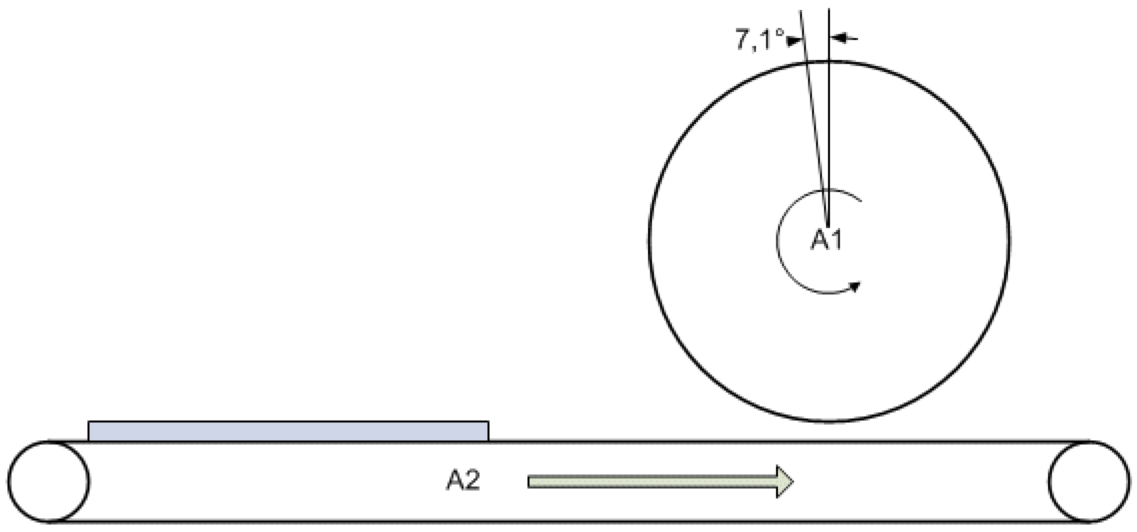
VelocityDiff = 500 mm/s

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 packet 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 speed 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 [▶ 69].

Fast correction:

Distance = 7.1°

Length = 360° (maximum possible correction distance)

Mode = SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION

VelocityDiff = 30°/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°. Since the distance should be as short as possible both modes are equivalent in this case.

Slow correction:

Distance = 7.1°

Length = 360° (correction distance)

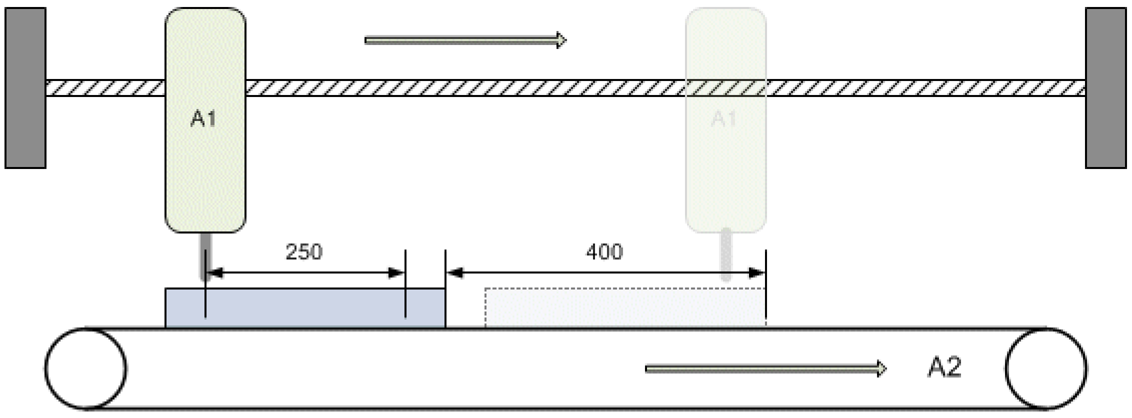
Mode = SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION

VelocityDiff = 30°/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 [▶ 69]:

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

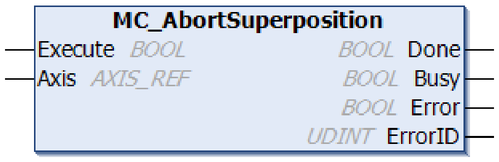
VelocityDiff = 500 mm/s (velocity reserve of the drilling device)

"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)	Tc2_MC2

6.2.3 MC_AbortSuperposition



The MC_AbortSuperposition function block terminates a superimposed movement started by MC_MoveSuperImposed [▶ 69], without stopping the subordinate axis movement.

A full axis stop can be achieved with MC_Stop [▶ 68] or MC_Halt [▶ 66], if necessary. In this case MC_AbortSuperposition does not have to be called.

Inputs

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

Execute: The command is executed with a positive edge and the superimposed movement is completed.

Outputs

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

Done: TRUE when the superimposed movement was successfully aborted.

Busy: TRUE as soon as the function block is active. FALSE when it returns to its initial state.

Error: TRUE as soon as an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

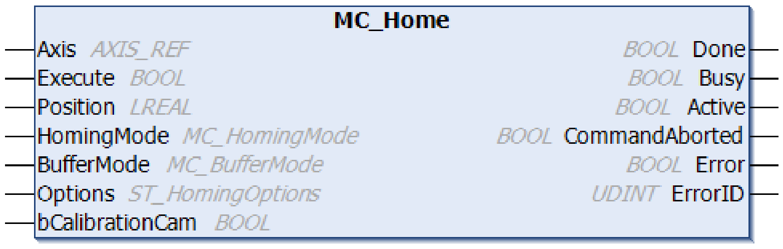
Axis: Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state. (Type: AXIS_REF [▶ 91])

Requirements

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

Execute: The command is executed with a positive edge.

Position: 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. **NOTE! 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.**

HomingMode: Determines how the calibration is performed. (Type: MC_HomingMode [▸ 104])

- MC_DefaultHoming
  Executes the standard 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.

BufferMode: Currently not implemented.

Options: Data structure containing additional, rarely used parameters. The input can normally remain open.

- **ClearPositionLag:** Only works in "MC_Direct" mode. Can optionally be used to set the set and actual positions to the same value. In this case the lag error is cleared.

bCalibrationCam: Reflects the signal of a reference cam that can be applied to the controller via a digital input.

See also: General rules for MC function blocks [▸ 13]

Outputs

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

Done: TRUE when the axis has been calibrated and the movement is complete.

Busy: 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: 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: TRUE if the command could not be executed completely.

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

See also: General rules for MC function blocks [▸ 13]

Inputs/outputs

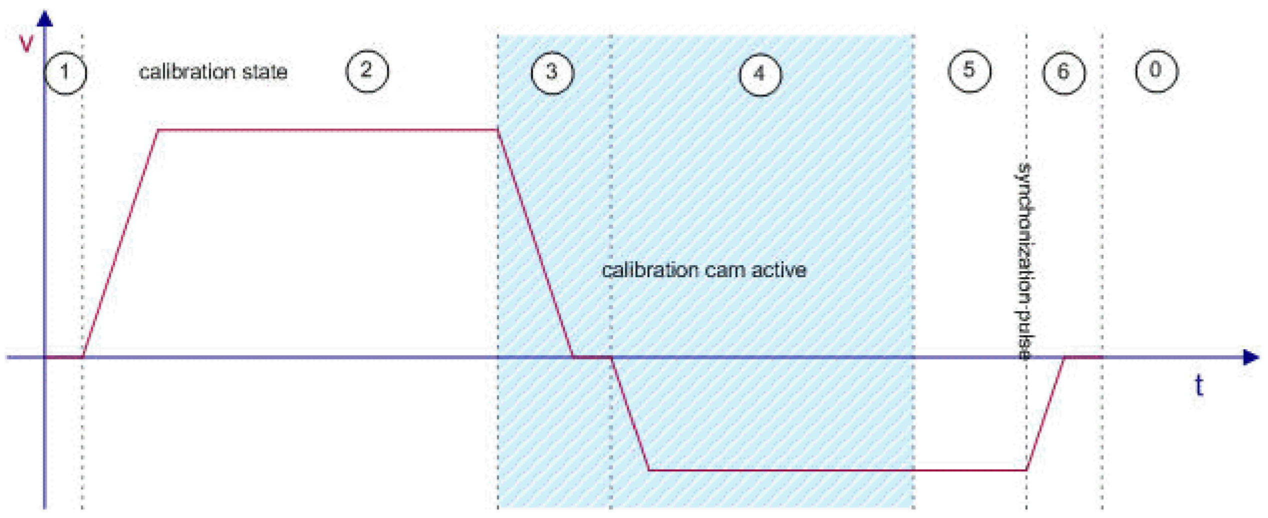
```
VAR_IN_OUT
    Axis      : AXIS_REF;
END_VAR
```

Axis: 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. (Type: AXIS_REF [▸ 91])

Note

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 reference cam, i.e. only based on the sync pulse of the encoder, the reference cam can be simulated via the PLC program. The "bCalibrationCam" signal is initially activated and then canceled, if Axis.NcToPlc.HomingState [▸ 92] is equal or greater 4.

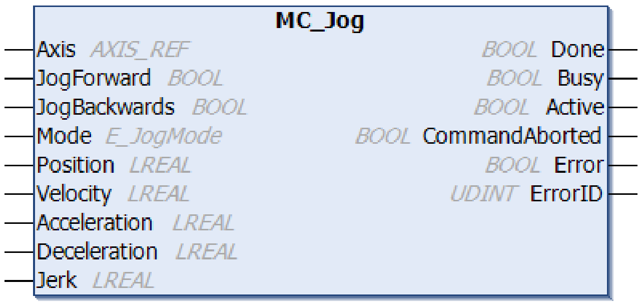


Requirements

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

JogForward: The command is executed with a positive edge, and the axis is moved in positive direction of travel. Depending on the operating 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: The command is executed with a positive 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: Determines the operating mode in which the manual function is executed. (Type: E_JogMode [▶ 99])

- 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_CONTINOUS
  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 positive 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 positive 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 positive edge at one of the jog inputs by a certain distance, which is defined via the "Position" 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 positive edge is encountered. With each start the velocity and dynamics data specified by the user are used.

Position: Relative distance for movements in MC_JOGMODE_INCHING operation mode.

Velocity: Maximum travel velocity (>0).

Acceleration: Acceleration (≥0). If the value is 0, the standard acceleration from the axis configuration in the System Manager is used.

Deceleration: Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used.

Jerk: Jerk (≥0). At a value of 0, the standard jerk from the axis configuration in the System Manager is applied.

NOTE! The parameters "Position", "Velocity", "Acceleration", "Deceleration" and "Jerk" are not used in the operating modes MC_JOGMODE_STANDARD_SLOW and MC_JOGMODE_STANDARD_FAST and can remain unassigned.

Outputs

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

Done: TRUE if a movement was successfully completed.

Busy: 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: Indicates that the axis is moved via the jog function.

CommandAborted: TRUE if the operation was aborted from the outside, e.g. by calling MC_Stop [▶ 68].

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

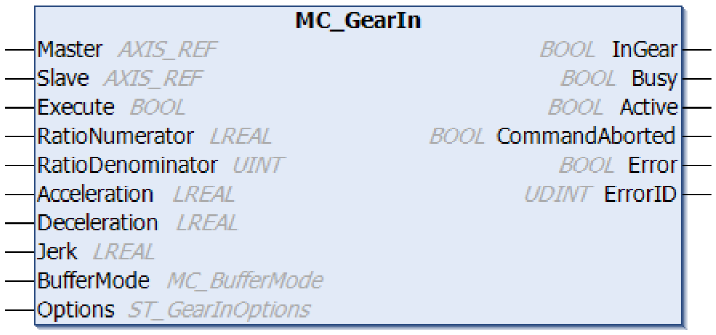
Axis: Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state. (Type: AXIS_REF [▶ 91])

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86)	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 [▸ 83]. If the slave is decoupled while it is moving, then it retains its velocity and can be halted using MC_Stop [▸ 68]or MC_Halt [▸ 66].

Alternatively, the function block MC_GearInDyn [▸ 81] 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
```

Execute: The command is executed with a positive edge.

RatioNumerator: Gear ratio numerator. Alternatively, the gear ratio can be specified in the numerator as a floating point value, if the denominator is 1.

RatioDenominator: Gear ratio denominator

Acceleration: Acceleration (≥0). (Currently not implemented)

Deceleration: Deceleration (≥0). (Currently not implemented)

Jerk: Jerk (≥0). (Currently not implemented)

BufferMode: Currently not implemented

Options: 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.

Outputs

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

InGear: TRUE if the coupling was successful.

Busy: 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: Indicates that the command is executed. (Currently Active = Busy)

CommandAborted: TRUE if the command could not be executed completely. The axis may have become decoupled during the coupling process (simultaneous command execution).

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Master: Axis data structure of the master.

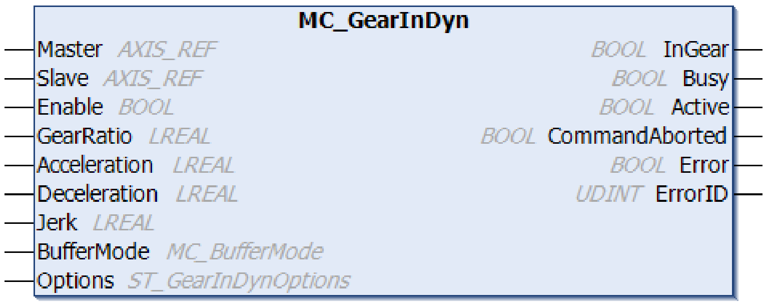
Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF [▸ 91] addresses an axis unambiguously within 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)	Tc2_MC2

6.5.2 MC_GearInDyn



The function block MC_GearIn 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 [▸ 83]. If the slave is decoupled while it is moving, then it retains its velocity and can be halted using MC_Stop [▸ 68]or MC_Halt [▸ 66].

Alternatively, the function block MC_GearIn [▸ 80] with dynamically variable 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
```

Enable: The coupling is executed with a positive edge. The gear ratio can be changed cyclically as long as "Enable" is TRUE. The command is terminated if "Enable" becomes FALSE after coupling. The gear ratio is frozen at its last value, but the slave is not decoupled.

GearRatio: Gear ratio as floating point value. The gear ratio can be changed cyclically as long as "Enable" is TRUE. If "Enable" isFALSE, the gear ratio remains unchanged.

Acceleration: 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: Deceleration (≥0). (Not implemented)

Jerk: Jerk (≥0). (Not implemented)

BufferMode: Currently not implemented

Options: Currently not implemented

Outputs

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

InGear: TRUE if the coupling was successful.

Busy: 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: Indicates that the command is executed. (Currently Active = Busy)

CommandAborted: TRUE if the command could not be executed completely. The axis may have become decoupled during the coupling process (simultaneous command execution).

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Master: Axis data structure of the master.

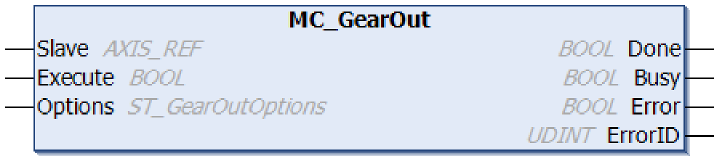
Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF [▶ 91] addresses an axis unambiguously within 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)	Tc2_MC2

6.5.3 MC_GearOut



The function block MC_GearOut deactivates a master-slave coupling.

Note	**No automatic stop of the slave axis** 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. The axis can be stopped with a MC_Halt [▶ 66] or MC_Stop [▶ 68] command.

Note	**Setpoint generator type** If the setpoint generator type of the axis is set to "7 phases (optimized)", the slave axis assumes an acceleration-free state after uncoupling and continues to move with the resulting constant speed. 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
```

Execute: The command is executed with a positive edge.

Options: Currently not implemented

Outputs

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

Done: TRUE if the axis has been successfully decoupled.

Busy: 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: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

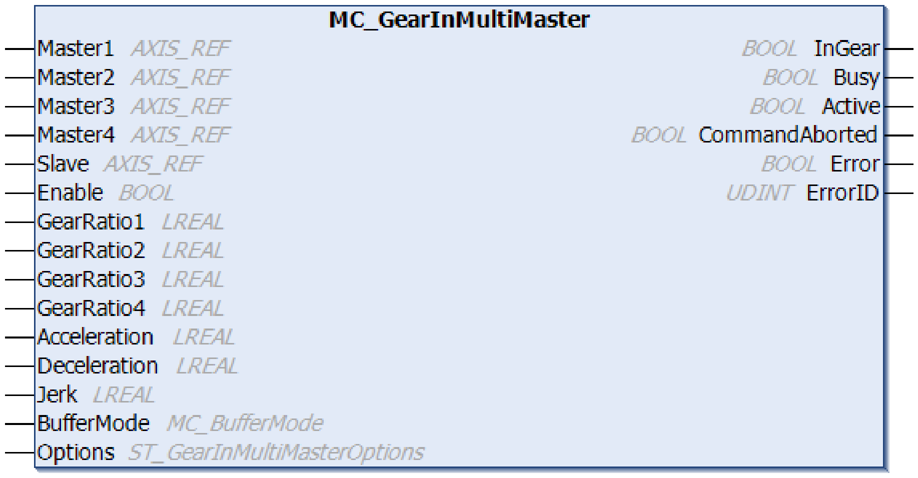
Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF [▸ 91] addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v3.0.0	PC or CX (x86)	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 [▸ 83]. If the slave is decoupled while it is moving, then it retains its velocity and can be halted using MC_Stop [▸ 68].

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
    Enable       : BOOL;
    GearRatio1   : LREAL;
    GearRatio2   : LREAL;
    GearRatio3   : LREAL;
    GearRatio4   : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    Options      : ST_GearInMultiMasterOptions;
END_VAR
```

Enable: The coupling is executed with a positive edge. The gear ratio can be changed cyclically as long as "Enable" is TRUE. The command is terminated if "Enable" becomes FALSE after coupling. The gear ratio is frozen at its last value, but the slave is not decoupled.

GearRatio1: 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" isFALSE, the gear ratio remains unchanged.

GearRatio2: 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" isFALSE, the gear ratio remains unchanged.

GearRatio3: 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" isFALSE, the gear ratio remains unchanged.

GearRatio4: 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" isFALSE, the gear ratio remains unchanged.

Acceleration: 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.

Deceleration: Deceleration (≥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. **Used only for the option "Advanced Slave Dynamics".**

Jerk: Jerk (≥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. **Used only for the option "Advanced Slave Dynamics".**

BufferMode: Currently not implemented

Options:

- **AdvancedSlaveDynamics:** 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.

Outputs

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

InGear: TRUE if the coupling was successful.

Busy: 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: Indicates that the command is executed. (Currently Active = Busy)

CommandAborted: TRUE if the command could not be executed completely. The axis may have become decoupled during the coupling process (simultaneous command execution).

Error: TRUE, if an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number.

Inputs/outputs

```
VAR_IN_OUT
    Master1 : AXIS_REF;
    Master2 : AXIS_REF;
    Master3 : AXIS_REF;
    Master4 : AXIS_REF;
    Slave   : AXIS_REF;
END_VAR
```

Master1: Axis data structure of the first master. (Type AXIS_REF [▶ 91])

Master2: Axis data structure of the second master. (Type AXIS_REF [▶ 91])

Master3: Axis data structure of the third master. (Type AXIS_REF [▶ 91])

Master4: Axis data structure of the fourth master. (Type AXIS_REF [▶ 91])

Slave: Axis data structure of the Slave. (Type AXIS_REF [▶ 91])

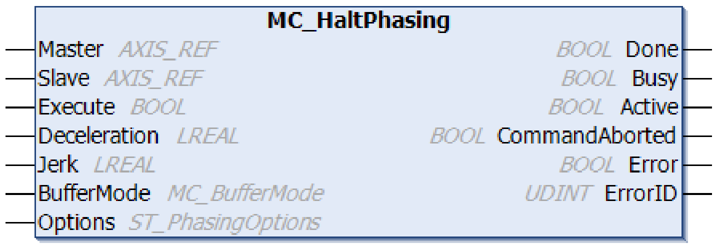
The axis data structure of type AXIS_REF [▶ 91] addresses an axis unambiguously within the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Requirements

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

6.6 Phasing

6.6.1 MC_HaltPhasing



The function block MC_HaltPhasing leads to 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
```

Execute: The command is executed with a positive edge.

Deceleration: Maximum deceleration value

Jerk: Maximum jerk value

BufferMode: Not implemented

Options: Data structure containing additional, rarely used parameters. The input can normally remain open.

Outputs

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

Done: TRUE if velocity = 0 is reached.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorId: If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

Master: Axis data structure of the master.

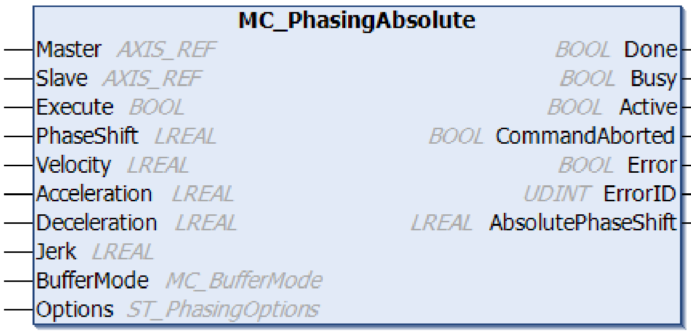
Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF [▶ 91] addresses an axis unambiguously within 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)	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".

Inputs

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

Execute: The command is executed with a positive edge.

PhaseShift: Phase shift to be set between master and slave axis

Velocity: Maximum velocity that may be reached during the phase shift.

Acceleration: Maximum acceleration value

Deceleration: Maximum deceleration value

Jerk: Maximum jerk value

BufferMode: Not implemented

Options: Data structure containing additional, rarely used parameters. The input can normally remain open.

Outputs

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

Done: TRUE when the absolute phase shift is established.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorId: If the error output is set, this parameter supplies the error number.

AbsolutePhaseShift: Absolute phase shift

Inputs/outputs

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

Master: Axis data structure of the master.

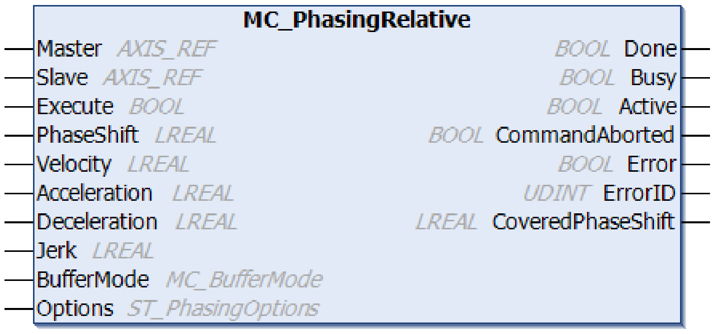
Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF [▸ 91] addresses an axis unambiguously within 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)	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".

Inputs

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

Execute: The command is executed with a positive edge.

PhaseShift: Amount by which the phase shift between master and slave axis is changed.

Velocity: Maximum velocity that may be reached during the phase shift.

Acceleration: Maximum acceleration value

Deceleration: Maximum deceleration value

Jerk: Maximum jerk value

BufferMode: Not implemented

Options: Data structure containing additional, rarely used parameters. The input can normally remain open.

Outputs

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

Done: TRUE when the relative phase shift is established.

Busy: 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: Indicates that the command is executed. If the command was buffered, it becomes active once a running command is completed.

CommandAborted: TRUE if the command could not be executed completely. The axis was stopped or the current command was replaced by another Move command.

Error: TRUE, if an error occurs.

ErrorId: If the error output is set, this parameter supplies the error number.

CoveredPhaseShift: Relative phase shift

Inputs/outputs

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

Master: Axis data structure of the master.

Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF [▶ 91] addresses an axis unambiguously within 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)	Tc2_MC2

7 Data types

7.1 Axis interface

7.1.1 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;
END_VAR
END_TYPE
```

AXIS_REF elements

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: PLCTONC_AXIS_REF [▸ 98])

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 NcToPlc 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: NCTOPLC_AXIS_REF [▸ 92])

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.

Status: Data structure containing additional or processed status information for an axis (type: ST_AxisStatus [▸ 109]). This data structure is not refreshed cyclically, but has to be updated through the PLC program. For this purpose MC_ReadStatus [▸ 26] or alternatively the action "ReadStatus" of AXIS_REF can be called:

Example:

```
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)	Tc2_MC2

7.1.2 NCTOPLC_AXIS_REF

The data structure NCTOPLC_AXIS_REF is part of the AXIS_REF [▶ 91] 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;
    {attribute 'hide'}
    _reserved1                 : ARRAY [1..16] OF USINT;
    ActPosWithoutPosCorrection : LREAL;
    ActAcc                     : LREAL;
    DcTimeStamp                : UDINT;
    {attribute 'hide'}
    _reserved2                 : ARRAY [1..12] OF USINT;
END_TYPE
```

Variable name	Data type	Definition range	Description
StateDWord	NCTOPLC_AXIS_REF_STATE [▶ 95]	-	State double word.
ErrorCode	DWORD	≥0	Axis error code
AxisState	DWORD	ENUM [▶ 94]	Present state of the axis movement
AxisModeConfirmation	DWORD	ENUM	Axis operating mode (feedback from the NC)
HomingState	DWORD	ENUM [▶ 94]	Reference status of the axis ("calibration status")
CoupleState	DWORD	ENUM [▶ 94]	Axis coupling state

Variable name	Data type	Definition range	Description
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 [▶ 94]	-	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
StateDWord2	NCTOPLC_AXIS_REF_STATE2 [▶ 96]		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_CAMCOUPLINGSTATE [▶ 97]		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
ActPosWithoutPosCorrection	LREAL		Actual position without position correction
ActAcc	LREAL		Actual acceleration
DcTimeStamp	UDINT		Current NC time stamp

Define	Master: Motion state / drive phase of the continuous master axis (servo) (Axis-State)
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) (Axis-State)
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 region)
4	Braking 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	Referencing state of the axis (HomingState)
0	Referencing process completed (READY)
1	Endless start in the direction of the referencing cam (note: if the cam is occupied at the start, then the process commences directly with referencing status 3)
2	Wait for rising 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
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 [▸ 76]

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)	Tc2_MC2

7.1.3 NCTOPLC_AXIS_REF_OPMODE

The structure NCTOPLC_AXIS_REF_OPMODE is part of the structure NCTOPLC_AXIS_REF [▸ 92].

```
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)	Tc2_MC2

7.1.4 NCTOPLC_AXIS_REF_STATE

The structure NCTOPLC_AXIS_REF_STATE is part of the structure NCTOPLC_AXIS_REF [▸ 92].

```
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)

Bit	Variable name	Description
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)
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)	Tc2_MC2

7.1.5 NCTOPLC_AXIS_REF_STATE2

The structure NCTOPLC_AXIS_REF_STATE2 is part of the structure NCTOPLC_AXIS_REF [▶ 92].

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

See also: NCTOPLC_AXIS_REF_STATE2_FLAGS [▸ 97]

Requirements

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

7.1.6 NCTOPLC_AXIS_REF_STATE2_FLAGS

The structure NCTOPLC_AXIS_REF_STATE2_FLAGS is part of the structure NCTOPLC_AXIS_REF_STATE2 [▸ 96].

```
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)	Tc2_MC2

7.1.7 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

7.1.8 PLCTONC_AXIS_REF

The data structure PLCTONC_AXIS_REF is part of the AXIS_REF [▶ 91] 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..21] OF USINT;
END_STRUCT
END_TYPE
```

Variable name	Data type	Definition range	Description
ControlDWord	PLCTONC_AXSI_REF_CTRL [▶ 99]	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	±∞	Gear ratio (coupling factor) 1
GearRatio2	LREAL	±∞	Gear ratio (coupling factor) 2
GearRatio3	LREAL	±∞	Gear ratio (coupling factor) 3
GearRatio4	LREAL	±∞	Gear ratio (coupling factor) 4
MapState	BOOL		Internal use only
PlcCycleControl	BYTE		Internal use only
PlcCycleCount	BYTE		Internal use only

Requirements

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

7.1.9 PLCTONC_AXIS_REF_CTRL

The structure PLCTONC_AXIS_REF_CTRL is part of the structure NCTOPLC_AXIS_REF [▸ 98].

```
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)	Tc2_MC2

7.2 Motion

7.2.1 E_JogMode

This data type is used in conjunction with the function block MC_Jog [▸ 78].

```
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_CONTINOUS, (* axis moves as long as the jog button is pressed using parameterized dy-
namics *)
    MC_JOGMODE_INCHING, (* axis moves for a certain relative distance *)
    MC_JOGMODE_INCHING_MODULO (* axis moves for a certain relative distance - stop posi-
tion is rounded to the distance value *)
);
END_TYPE
```

Requirements

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

7.2.2 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 [▶ 69].

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".

SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION

The superimposed motion 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.

"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.

SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION

The superimposed motion 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.

The "Length" refers to the axis affected by the compensation. During compensation, the travel path of this axis is "Length".

SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION

The distance of the superimposed motion is as short as possible and the speed is as high as possible. Although neither the maximum velocity change "VelocityDiff" nor the maximum "Length" are exceeded.

"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".

SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION

The distance of the superimposed motion is as short as possible and the speed is as high as possible. Although neither the maximum velocity change "VelocityDiff" nor the maximum "Length" are exceeded.

The "Length" refers to the axis affected by the compensation. During compensation, the maximum travel path of this axis is "Length".

SUPERPOSITIONMODE_ACCREDUCTION_ADDITIVEMOTION (from TwinCAT 2.11)

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.

"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".

SUPERPOSITIONMODE_ACCREDUCTION_LIMITEDMOTION (from TwinCAT 2.11)

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.

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)	Tc2_MC2

7.2.3 MC_AxisStates

This data type describes the operating states according to the PlcOpen state diagram [▶ 10].

```
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_CONTINOUSMOTION,
    MC_AXISSTATE_SYNCHRONIZEDMOTION
);
END_TYPE
```

See also: General rules for MC function blocks [▶ 13]

Requirements

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

7.2.4 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.

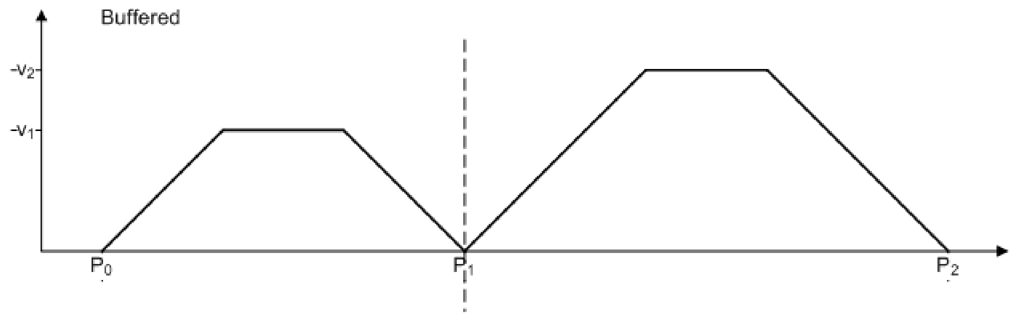
```
TYPE MC_BufferMode :
 (
    MC_Aborting,
    MC_Buffered,
    MC_BlendingLow,
    MC_BlendingPrevious,
    MC_BlendingNext,
    MC_BlendingHigh
 );
END_TYPE
```

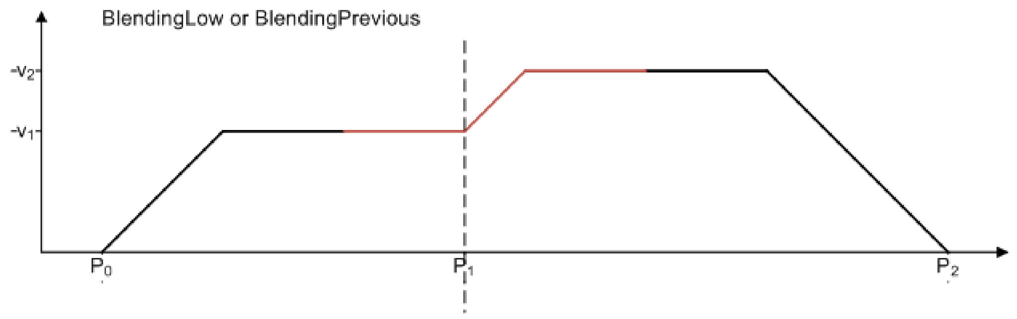
NOTE! A second function block is always required to use the BufferMode. It is not possible to trigger a move function block with new parameters while it is active.

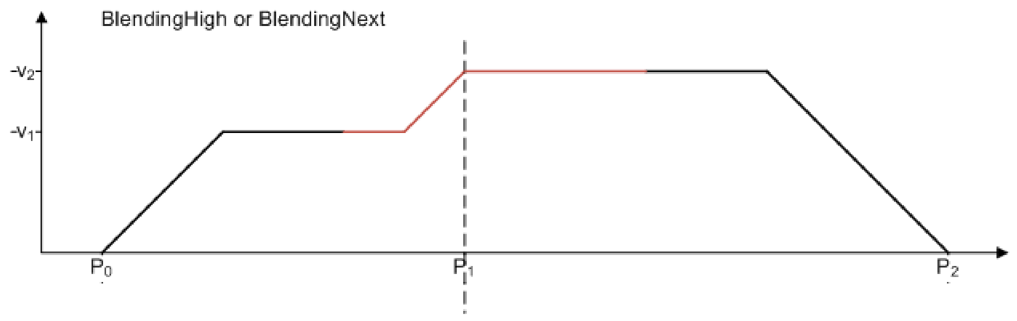
See also: General rules for MC function blocks [▶ 13]

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.

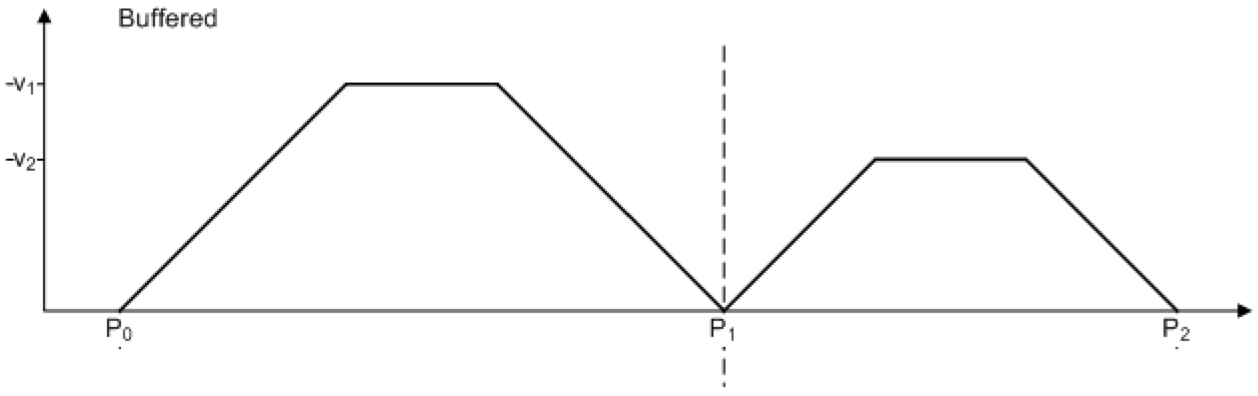


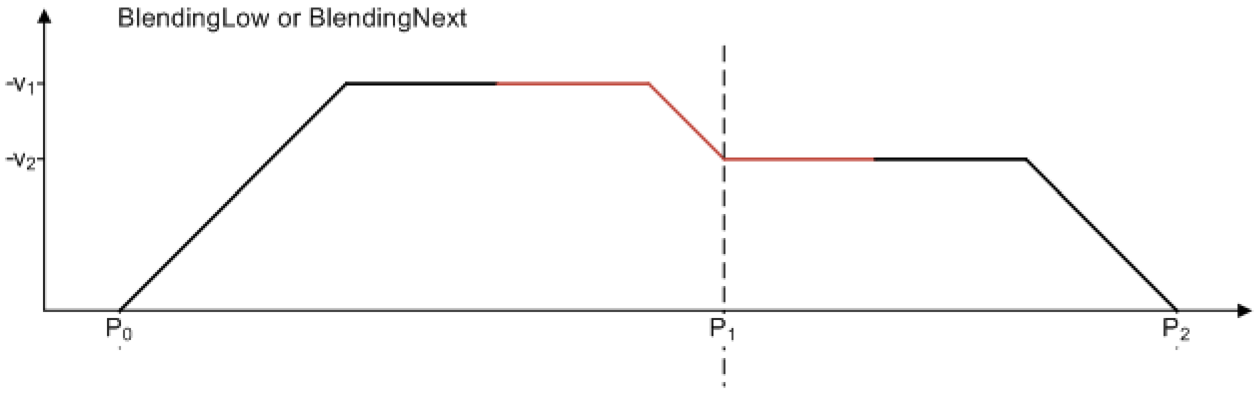


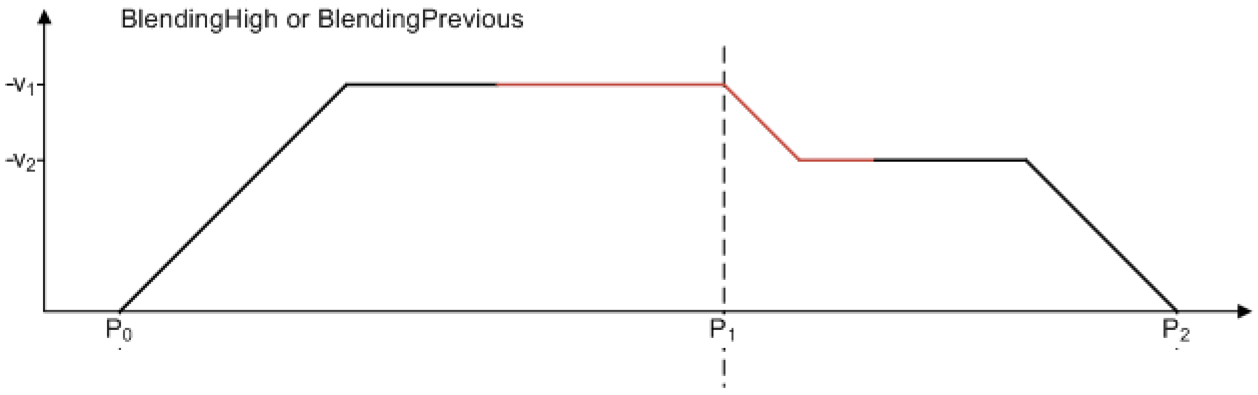


Since the speed of the first command is lower than the second, the modes BlendingLow/BlendingPrevious and BlendingHigh/BlendingNext have the same result.

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)	Tc2_MC2

7.2.5 MC_Direction

This enumeration type contains the possible directions of movement for the function blocks MC_MoveVelocity [▶ 61] and MC_MoveModulo [▶ 55].

```
TYPE MC_Direction :
(
    MC_Positive_Direction := 1,
    MC_Shortest_Way ,
    MC_Negative_Direction,
    MC_Current_Direction
);
END_TYPE
```

Requirements

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

7.2.6 MC_HomingMode

This data type is used to parameterize the function block MC_Home [▸ 76].

```
TYPE MC_HomingMode :
(
    MC_DefaultHoming,    (* default homing as defined in the SystemManager encoder parameters *)
    MC_AbsSwitch,        (* not implemented - Absolute Switch homing plus Limit switches *)
    MC_LimitSwitch,      (* not implemented - Homing against Limit switches *)
    MC_RefPulse,         (* not implemented - Homing using encoder Reference Pulse "Zero Mark" *)
    MC_Direct,           (* Static Homing forcing position from user reference *)
    MC_Absolute,         (* not implemented - Static Homing forcing position from absolute en-
coder *)
    MC_Block,            (* not implemented - Homing against hardware parts blocking movement *)
    MC_ForceCalibration, (* set the calibration flag without perfomring any motion or chang-
ing the position *)
    MC_ResetCalibration  (* resets the calibration flag without perfomring any motion or chang-
ing the position *)
);
END_TYPE
```

Requirements

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

7.2.7 ST_AxisOpModes

This data type contains information about the parameterization of the operating 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 *)
    Modulo                   : BOOL; (* bit 7 - OpModeDWord - axis is parameterized as mod-
ulo axis *)
    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 *)
END_STRUCT
END_TYPE
```

Requirements

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

7.2.8 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

AbortOption is an optional parameter of the function block MC_MoveSuperimposed [▶ 69], which determines the behavior of a superimposed movement at a standstill of the main movement.

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 speed 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 operating 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)

	ABORTATSTAND STILL	RESUMEAFTER STANDSTILL	RESUMEAFTER MOTIONSTOP
1. override = 0%	continued	continued	continued
2. temporary standstill of the main movement	Abort	continued	continued
3. motion reversal	Abort	continued	continued
4. axis has reached the target position or is stopped	Abort	Abort	continued
5. axis reset or switch-off of the enable signal	Abort	Abort	Abort
6. for slave axes: Uncoupling	Abort	Abort	Abort

Requirements

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

7.3 Status and parameter

7.3.1 E_ReadMode

This data type is used in conjunction with the function blocks MC_ReadBoolParameter [▸ 24] and MC_ReadBoolParameter [▸ 23] 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)	Tc2_MC2

7.3.2 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 *)
 AxisEnBacklashCompensation,         (* bool  *) (* IndexOffset= 16#0000_002B *)
 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 *)

(* Beckhoff specific axis functions *)
 (* read/write gear ratio of a slave *)
 AxisGearRatio := 3000,              (* lreal *) (* read: IndexGroup=0x4100+ID, IdxOff-
set=16#0000_0022, *)
 (* 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
```

NOTE! The AxisGearRatio parameter can only be read or written if the axis is coupled as a slave. During the motion only very small changes are allowed.

Requirements

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

7.3.3 ST_AxisParameterSet

This data type contains the entire parameter data set of an axis that can be read with the function block MC_ReadParameterSet [▸ 25].

Individual parameters that can be changed at runtime can be written with MC_WriteParameter [▸ 28]. 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 *)
    sAxisName                    : STRING(31);  (* 0x00000002 *)
    nAxisType                    : DWORD;       (* 0x00000003 *)
    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 *)
    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 *)
    sEncName                    : STRING(31);   (* 0x00010002 *)
    nEncType                    : DWORD;        (* 0x00010003 *)
    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 *)
    bEncEnablePosCorrection     : WORD;         (* 0x00010016 *)
    fEncPosCorrectionFilterTime : LREAL;        (* 0x00010017 *)

    (* CONTROLLER: *)
    nCtrlId                     : DWORD;        (* 0x00020001 *)
    sCtrlName                   : STRING(31);   (* 0x00020002 *)
    nCtrlType                   : DWORD;        (* 0x00020003 *)
    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 *)
    sDriveName                  : STRING(31);   (* 0x00030002 *)
    nDriveType                  : DWORD;        (* 0x00030003 *)
    bDriveIsInverse             : WORD;         (* 0x00030006 *)
    fDriveVeloReferenz          : LREAL;        (* 0x00030101 *)
    fDriveOutputReferenz        : LREAL;        (* 0x00030102 *)

    (* fill up *)
    arrReserved : ARRAY[455..512] OF BYTE; (* fill up to 512 bytes *)
END_STRUCT
END_TYPE
```

Requirements

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

7.3.4 ST_AxisStatus

This data type contains extensive status information about an axis. The data structure must be updated during each PLC cycle by calling MC_ReadStatus [▶ 26] or by calling the action Axis.ReadStatus (AXIS_REF [▶ 91]).

```
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 NcTo-
Plc 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;
    SynchronizedMotion    : BOOL;
    Homing                : BOOL;

    (* additional status - (PLCopen definition)*)
    ConstantVelocity      : BOOL;
    Accelerating          : BOOL;
    Decelerating          : BOOL;

    (* Axis.NcToPlc.StateDWord *)
    Operational           : BOOL;
    ControlLoopClosed     : BOOL; (* operational and position control active *)
    HasJob                : BOOL;
    HasBeenStopped        : BOOL;
    NewTargetPosition     : BOOL; (* new target position commanded during move *)
    InPositionArea        : BOOL;
    InTargetPosition      : BOOL;
    Protected             : BOOL;
    Homed                 : BOOL;
    HomingBusy            : BOOL;
    MotionCommandsLocked : BOOL; (* stop 'n hold *)
    SoftLimitMinExceeded : BOOL; (* reverse soft travel limit exceeded *)
    SoftLimitMaxExceeded : BOOL; (* forward soft travel limit exceeded *)

    Moving                : BOOL;
    PositiveDirection     : BOOL;
    NegativeDirection     : BOOL;
    NotMoving             : BOOL;
    Compensating          : BOOL; (* superposition - overlayed motion *)

    ExtSetPointGenEnabled: BOOL;
    ExternalLatchValid    : BOOL;
    CamDataQueued         : BOOL;
    CamTableQueued        : BOOL;
    CamScalingPending     : BOOL;
    CmdBuffered           : BOOL;
    PTPmode               : BOOL;
    DriveDeviceError      : BOOL;
    IoDataInvalid         : BOOL;

    (* Axis.NcToPlc.CoupleState *)
    Coupled               : BOOL;

    (* axis operation mode feedback from NcToPlc *)
    OpMode                : ST_AxisOpModes;
END_STRUCT
END_TYPE
```

Requirements

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

7.3.5 ST_DriveAddress

This data type contains the ADS access data of a drive device. The data is read with the function block MC_ReadDriveAddress [▶ 47].

```
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 informa-
tion 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   : DWORD;         (* 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   : DWORD;        (* type enumeration of the NC encoder drive of an axis *)
    NcAxisId        : DWORD;        (* ID [1..255] of the NC axis *)
    NcAxisType      : DWORD;        (* type enumeration of the NC axis *)
END_STRUCT
END_TYPE
```

Requirements

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

7.3.6 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)	Tc2_MC2

7.3.7 E_AxisPositionCorrectionMode

```
TYPE E_PositionCorrectionMode:
(
    POSITIONCORRECTION_MODE_UNLIMITED, (* no limitation - pass correction immediately *)
    POSITIONCORRECTION_MODE_FAST,      (* limitatation to maximum position change per cycle *)
    POSITIONCORRECTION_MODE_FULLENGTH  (* limitation uses full length to adapt to correc-
tion in small steps *)
);
END_TYPE
```

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_FULLLENGTH	The position correction is accomplished distributed over a distance of the axis (CorrectionLength). This results in smaller changes per time unit.

7.4 External set value generator

7.4.1 E_PositionType

```
TYPE E_PositionType :
(
    POSITIONTYPE_ABSOLUTE := 1, (*Absolute position*)
    POSITIONTYPE_RELATIVE,     (*Relative position*)
    POSITIONTYPE_MODULO := 5 (*Modulo position*)
);
END_TYPE
```

Requirements

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

7.5 Touch probe

7.5.1 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 sig-
nal source is set to 'PlcEvent' *)
    ModuloPositions : BOOL; (* interpretation of FirstPosition, LastPosition and RecordedPosi-
tion as modulo positions when TRUE *)
END_STRUCT
END_TYPE
```

EncoderID: The ID of an encoder is indicated in the TwinCAT System Manager.

TouchProbe: Defines the latch unit (probe unit) within the encoder hardware used.

```
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 *)
);
END_TYPE
```

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".

```
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 *)
);
END_TYPE
```

Edge: Defines whether the positive or negative edge of the trigger signal is evaluated.

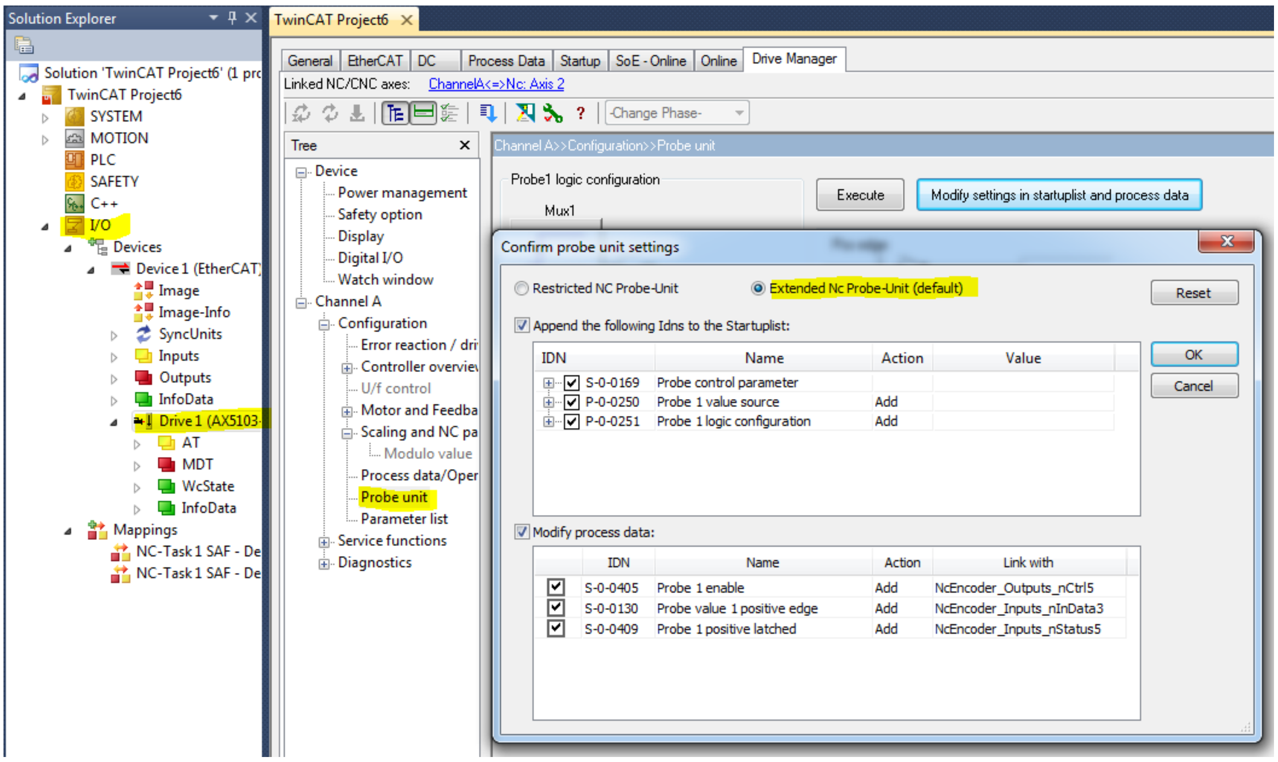
```
TYPE E_SignalEdge :
(
    RisingEdge,
    FallingEdge
);
END_TYPE
```

Mode: 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_CONTINOUS (* multi probe interface - from 2.11 Build 2022 *)
);
END_TYPE
```

Note regarding Beckhoff drives:

For the SINGLE or CONTINUOUS modes the probe unit must be configured as an "Extended Nc Probe Unit".


Confirm probe unit settings

Restricted NC Probe-Unit [✓ Extended Nc Probe-Unit (default)]

[✓] Append the following Idns to the Startuplist:

IDN	Name	Action	Value
✓ S-0-0169	Probe control parameter		
✓ P-0-0250	Probe 1 value source	Add	
✓ P-0-0251	Probe 1 logic configuration	Add	

[✓] Modify process data:

	IDN	Name	Action	Link with
✓	S-0-0405	Probe 1 enable	Add	NcEncoder_Outputs_nCtrl5
✓	S-0-0130	Probe value 1 positive edge	Add	NcEncoder_Inputs_nInData3
✓	S-0-0409	Probe 1 positive latched	Add	NcEncoder_Inputs_nStatus5

PlcEvent: If the signal source "TouchProbe" is set to the type "PlcEvent", a positive 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: If the variable "ModuloPositions" is FALSE, the axis position is interpreted in an absolute linear range from -∞ to +∞. The positions "FirstPosition", "LastPosition" and "RecordedPosition" of the function block MC_TouchProbe [▸ 31] are then also absolute.
If "ModuloPositions" is TRUE, all positions are interpreted as modulo values 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)	Tc2_MC2

7.5.2 MC_TouchProbeRecordedData

```
TYPE MC_TouchProbeRecordedData :
STRUCT
    Counter             : LREAL;
    RecordedPosition    : LREAL;
    AbsolutePosition    : LREAL;
    ModuloPosition      : LREAL;
END_STRUCT
END_TYPE
```

Counter: 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: 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: Absolute axis position detected at the time of the trigger signal.

ModuloPosition: Modulo axis position recorded at the time of the trigger signal.

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).

NOTE! All other options for comparing library versions, which you may know from TwinCAT 2, are outdated!

9 Appendix

9.1 Summary of 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/zip/9007201641738507.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/zip/9007201641736843.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/zip/9007201641733515.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/zip/9007201641740171.zip

Compensation of the backlash of an axis

The sample program shows how the backlash of an axis can be compensated

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_MC2/Resources/zip/9007201641730187.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/zip/9007201641735179.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/zip/9007201641731851.zip

Requirements

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

9.2 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.

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 her internet pages:
http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49(0)5246/963-0
Fax:	+49(0)5246/963-198
e-mail:	info@beckhoff.com

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(0)5246/963-157
Fax:	+49(0)5246/963-9157
e-mail:	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(0)5246/963-460
Fax:	+49(0)5246/963-479
e-mail:	service@beckhoff.com