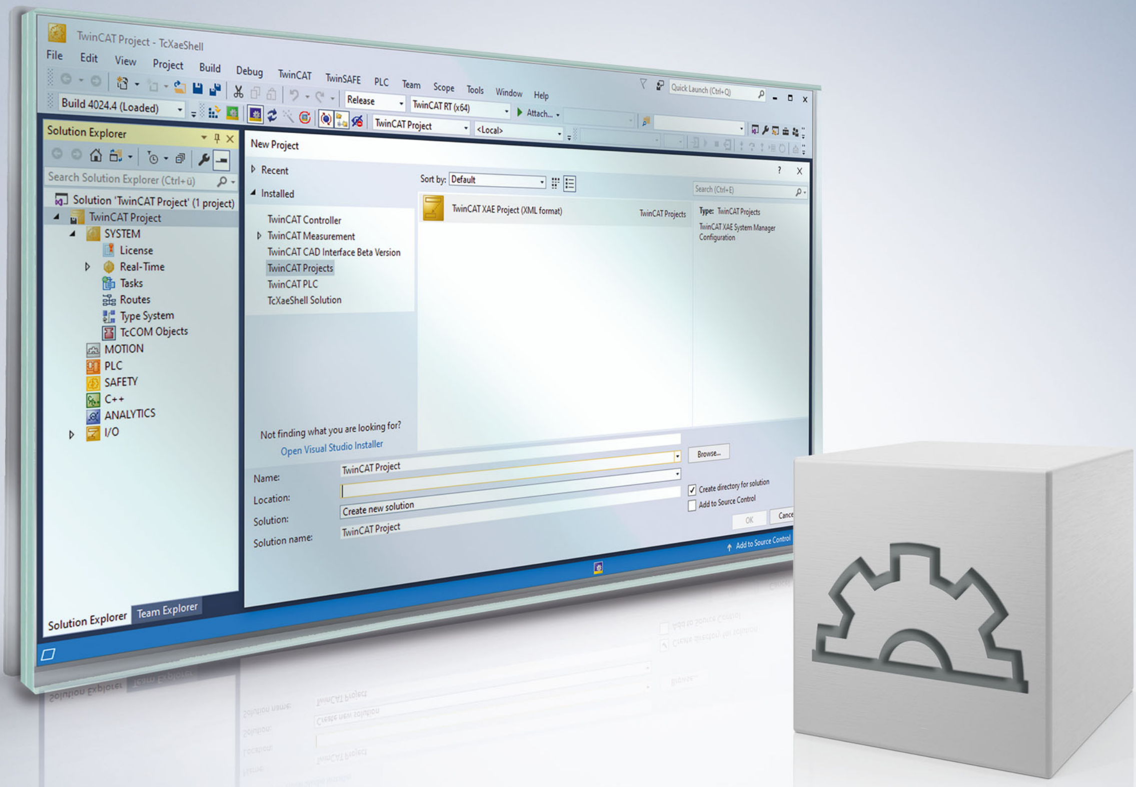
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc3_DriveMotionControl



2025-06-26 | Version: 1.5.3

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

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

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Copyright

Third-party trademarks

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

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

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

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Overview

The TwinCAT Motion Control PLC library Tc3_DriveMotionControl contains function blocks for programming simple machine applications based on Beckhoff servo terminal technology. It is based on the PLCopen specification for Motion Control function blocks V2.0 (www.PLCopen.org).

This library offers an alternative option for controlling simple movements without using TwinCAT NC PTP. The functionality is reduced compared to TwinCAT NC PTP.

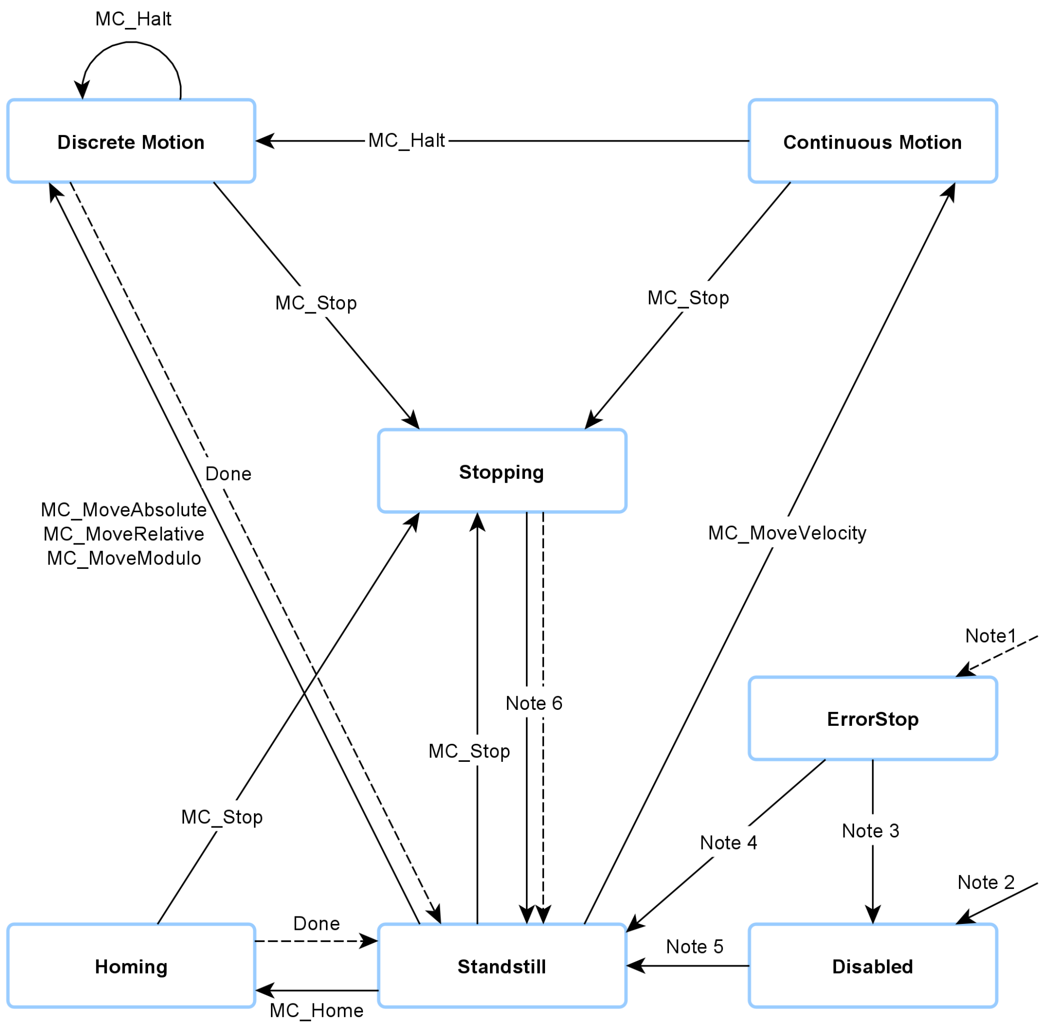
If the library is to be used in parallel with Tc2_MC2, set the option "*Qualified access only*" = TRUE for one of the libraries.

This library can then be addressed via the corresponding namespace, e.g. Tc2_MC2.MC_Power.

Motion commands can be commanded directly to a servo terminal in the usual PLCopen-compliant manner.

3 State diagram

The following state diagram defines the behavior of an axis.



Note 1	From undefined state in which an error occurs.
Note 2	From undefined state if MC_Power.Enable = FALSE. The axis has no fault.
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.

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 influence the status of the axis.

The "Stopping" state indicates that the axis is in a stop ramp. After the complete stop the state does not change to "Standstill" until MC_Stop is called with Execute = FALSE. Otherwise the axis remains locked for further motion commands.

4 General rules for MC function blocks

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

Exclusivity of the outputs

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

An exception is the motion command MC_Stop [▸ 37]. This sets the "Done" output to TRUE as soon as the axis is stopped. Nevertheless, the "Busy" and "Active" outputs remain TRUE because the axis is locked. The axis is unlocked and the "Busy" and "Active" outputs set to FALSE only after the "Execute" input is set to FALSE.

Initial state

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

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

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

Input parameters

The input parameters become active with a rising edge. To change the parameters, the command must be retriggered after it has finished. If an input parameter is not passed to the function block, the last value passed to this function block remains valid. Meaningful values should be parameterized at the first call.

Position and Distance

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

Dynamic parameters

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

Velocity	mm/s
Acceleration	mm/s2
Deceleration	mm/s2

Error handling

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

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

Error types

- Function block errors only concern the function block, not the axis (e.g. incorrect parameterization). Function block errors do not have to be reset explicitly. They are reset automatically when the "Execute" input is reset.
- Communication errors (e.g. the function block cannot address the axis) usually indicate incorrect configuration or parameterization. A Reset is not possible. The function block can be retriggered after the configuration has been corrected.
- In many cases, drive errors (controller) can be reset via the motion command MC_Reset [▶ 14].

Behavior of the Done output

The output "Done" (or alternatively "InVelocity") is set if a command was executed successfully.

Behavior of the CommandAborted output

The output "CommandAborted" is set if a command is interrupted.

Behavior of the Busy output

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

Behavior of the Active output

The "Active" output of a function block indicates that the function block has control over the axis.

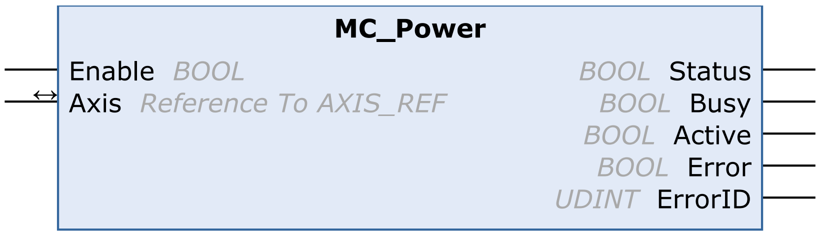
Option input

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

5 Organization blocks

5.1 Axis functions

5.1.1 MC_Power



The function block MC_Power switches the software enabling of an axis. At Status output operational readiness of the axis is indicated.

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;
END_VAR
```

Name	Type	Description
Enable	BOOL	General software enable for the axis.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

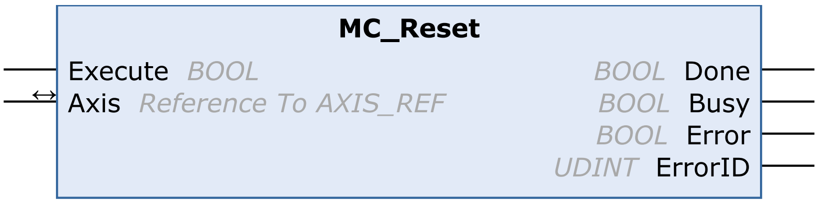
```
VAR_OUTPUT
    Status  : BOOL;
    Busy    : BOOL;
    Active  : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

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

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.1.2 MC_Reset



The function block 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
```

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

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

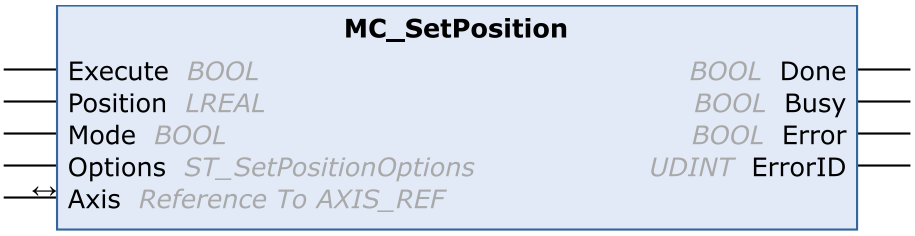
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

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

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.1.3 MC_SetPosition



The function block 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
```

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	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	BOOL	The axis position is set to an absolute value set if Mode = FALSE. Otherwise the axis position is changed relative to the specified Position value. Relative mode can be used for changing the position of an axis during motion.
Options	ST_SetPositionOptions	Not used at present

See also: General rules for MC function blocks [▶ 11]

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

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

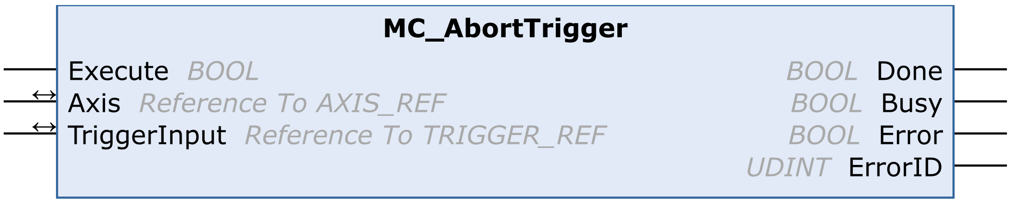
See also: General rules for MC function blocks [▶ 11]

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.2 Touch probe

5.2.1 MC_AbortTrigger



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

Inputs

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

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

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure
TriggerInput	TRIGGER_REF [▶ 45]	Data structure for describing the trigger source. This data structure must be parameterized before the function block is called for the first time.

Outputs

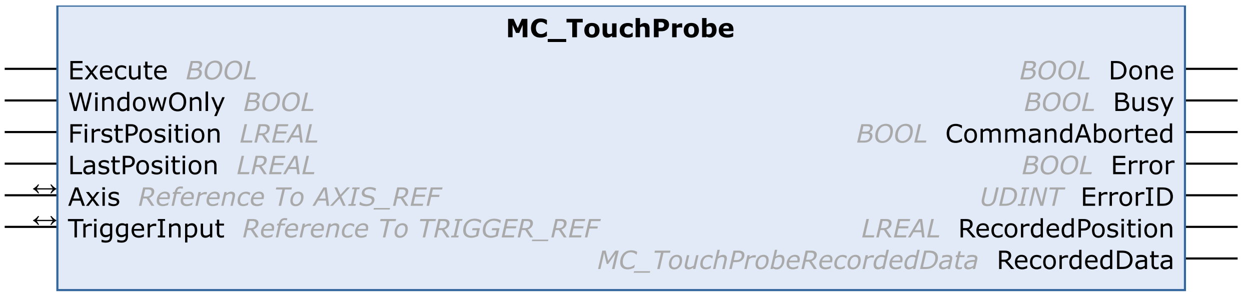
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

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

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.2.2 MC_TouchProbe

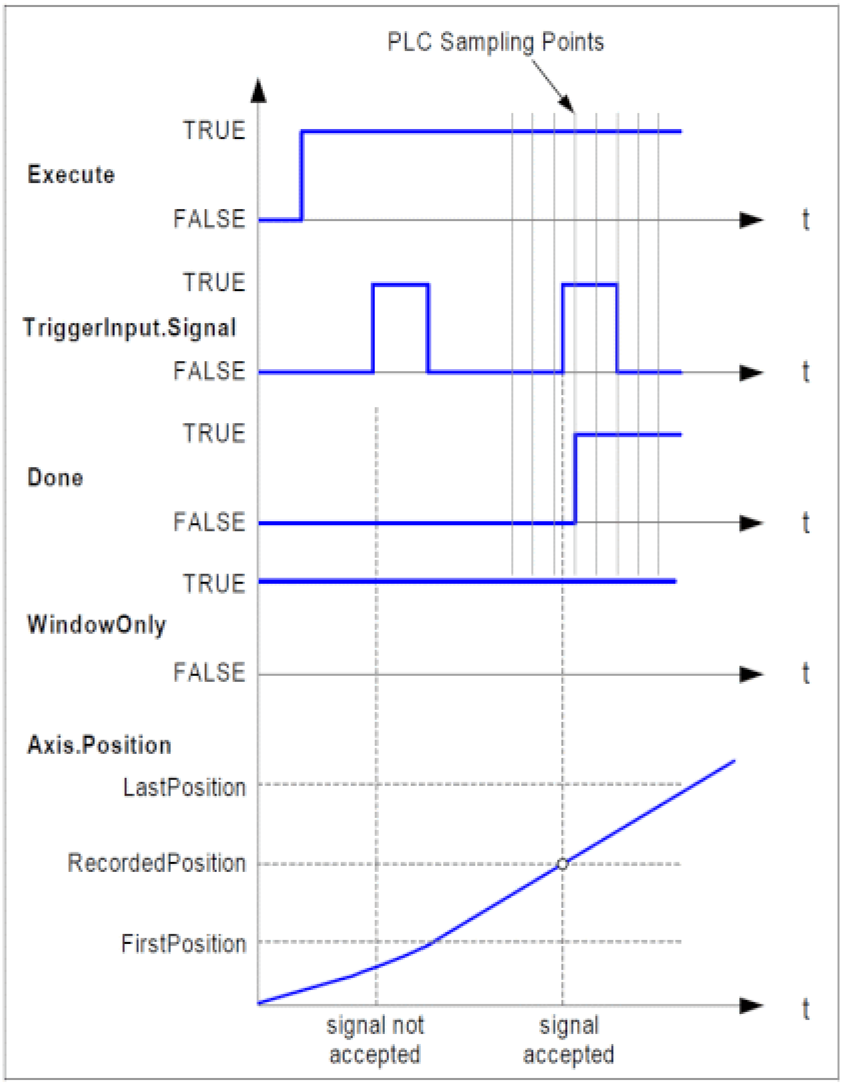


The function block MC_TouchProbe records an axis position at the time of a digital signal (touch probe function). The position is captured via an external hardware latch and is therefore highly accurate and independent of the cycle time. The function block controls this mechanism and determines the externally recorded position.

The parameters involved may have to be set in the drive parameters. For the servo terminal the parameters can be found in the objects DMC Setting (0x8030) or DMC Features (0x8031).

After a measuring probe cycle has been initiated by a rising edge on the "Execute" input, this is only terminated if the outputs "Done", "Error" or" CommandAborted" become TRUE. If the operation is to be aborted in the meantime, the function block MC_AbortTrigger [▸ 16] must be called with the same TriggerInput [▸ 45] data structure. Otherwise no new cycle can be initiated.

Signal curve

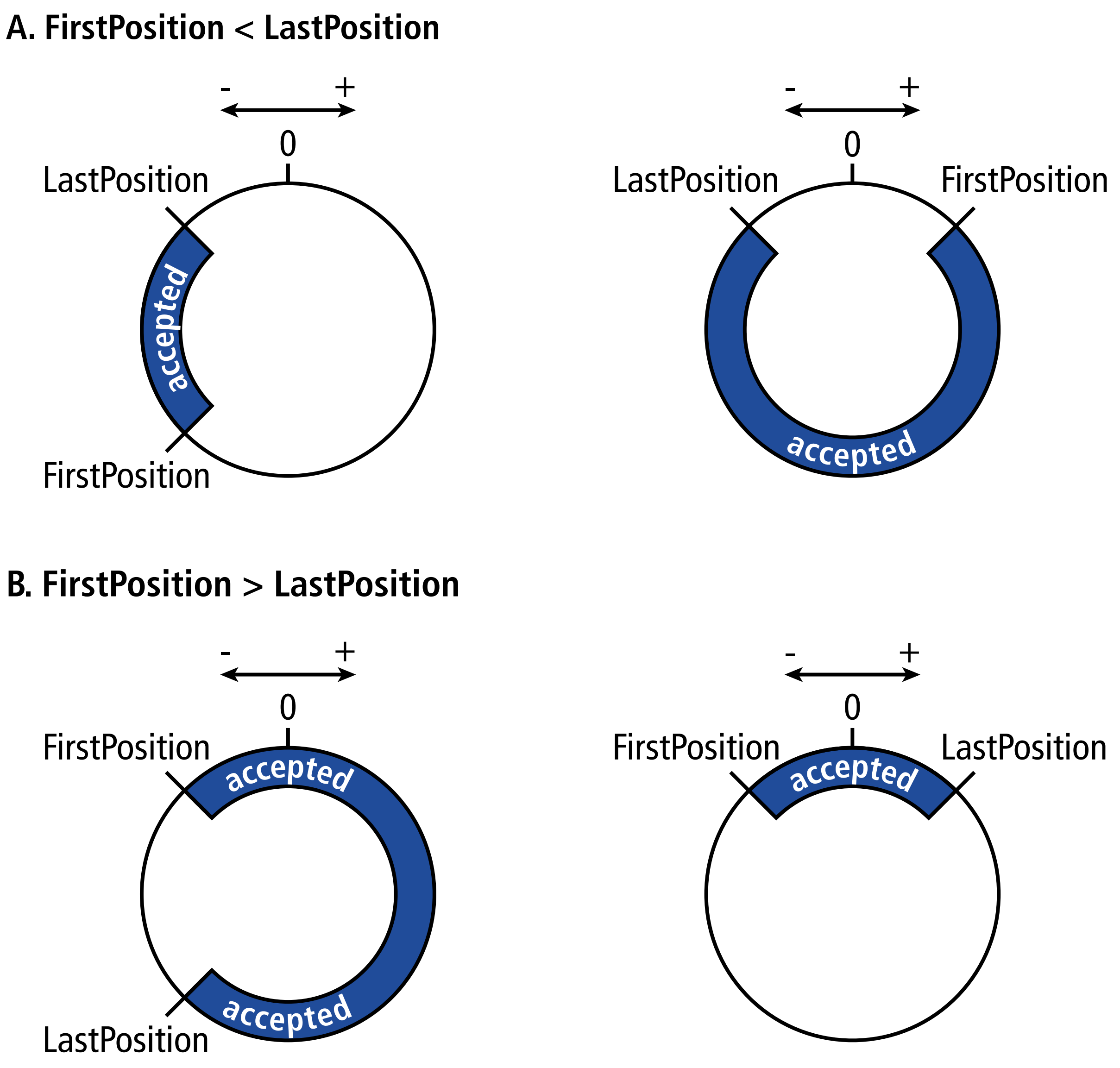


Inputs

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

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge, and the external position latch is activated.
WindowOnly	BOOL	If WindowOnly = TRUE, only one position within the window between "FirstPosition" and "LastPosition" is recorded. Positions outside the window are rejected and the external position latch is automatically newly activated. Only if the recorded position lies inside the window, "Done" becomes TRUE. The recording window can be interpreted in terms of absolute or modulo values. For this purpose, the "ModuloPositions" flag must be set accordingly in the TriggerInput [▶ 45] 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°).

Name	Type	Description
FirstPosition	LREAL	Initial position of the recording window, if "WindowOnly" is TRUE. This position can be interpreted as an absolute or modulo value. For this purpose, the "ModuloPositions" flag must be set accordingly in the TriggerInput [▶ 45] data structure.
LastPosition	LREAL	End position of the recording window, if "WindowOnly" is TRUE. This position can be interpreted as an absolute or modulo value. For this purpose, the "ModuloPositions" flag must be set accordingly in the TriggerInput [▶ 45] data structure.



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

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▸ 39]	Axis data structure
TriggerInput	TRIGGER_REF [▸ 45]	Data structure for describing the trigger source. This data structure must be parameterized before the function block is called for the first time.

Outputs

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

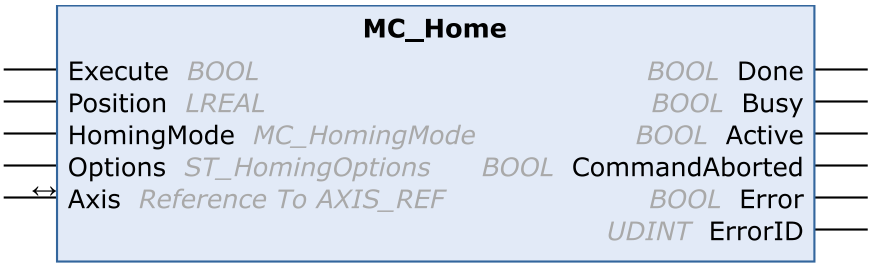
Name	Type	Description
Done	BOOL	TRUE if an axis position was successfully detected. The position is sent to the output "RecordedPosition".
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state.
CommandAborted	BOOL	TRUE if the process is interrupted by an external event, e.g. by the call up of MC_AbortTrigger [▸ 16].
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.
RecordedData	MC_TouchProbeRecordedData	Data structure with complementary information relating to the logged axis position at the time of the trigger signal

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6 Motion function blocks

6.1 MC_Home (Homing)



An axis reference run is carried out with the function block MC_Home.

Reference mode is set in the options with the parameter "ReferenceMode".

The parameters involved may have to be set in the drive parameters. For the servo terminal, the parameters can be found in the objects DMC Setting (0x8030) or DMC Features (0x8031).

Inputs

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

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Absolute reference position to which the axis is set after homing. Alternatively, the constant DEFAULT_HOME_POSITION can be used here. In this case, the "Reference position for homing" specified in the TwinCAT System Manager is used.
HomingMode	MC_HomingMode	Determines how the calibration is performed. (Type: MC_HomingMode [▶ 40]) • MC_DefaultHoming Initiates default homing. • MC_Direct Sets the axis position directly to Position without executing a movement. • MC_Block Performs homing to a mechanical end stop. • MC_ForceCalibration Enforces the "axis is calibrated" state. No movement takes place, and the position remains unchanged. • MC_ResetCalibration Resets the calibration state of the axis. No movement takes place, and the position remains unchanged.
Options	ST_HomingOptions	Data structure containing additional parameters. • SearchDirection: Direction in which the referencing cam is to be searched • SearchVelocity: Velocity at which the referencing cam is to be searched

Name	Type	Description
		• SyncDirection: Direction in which the falling edge of the referencing cam is searched after the referencing cam has been detected • SyncVelocity: Velocity at which the falling edge of the referencing cam is searched for after the referencing cam has been detected • ReferenceMode: Referencing mode (currently only ENCODERREFERENCEMODE_CAMATDIGITALINPUT) • Acceleration: Acceleration for homing • Deceleration: Deceleration for homing The signal of a referencing cam must be routed to a digital terminal input (HomingMode = MC_DefaultHoming).

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.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▸ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

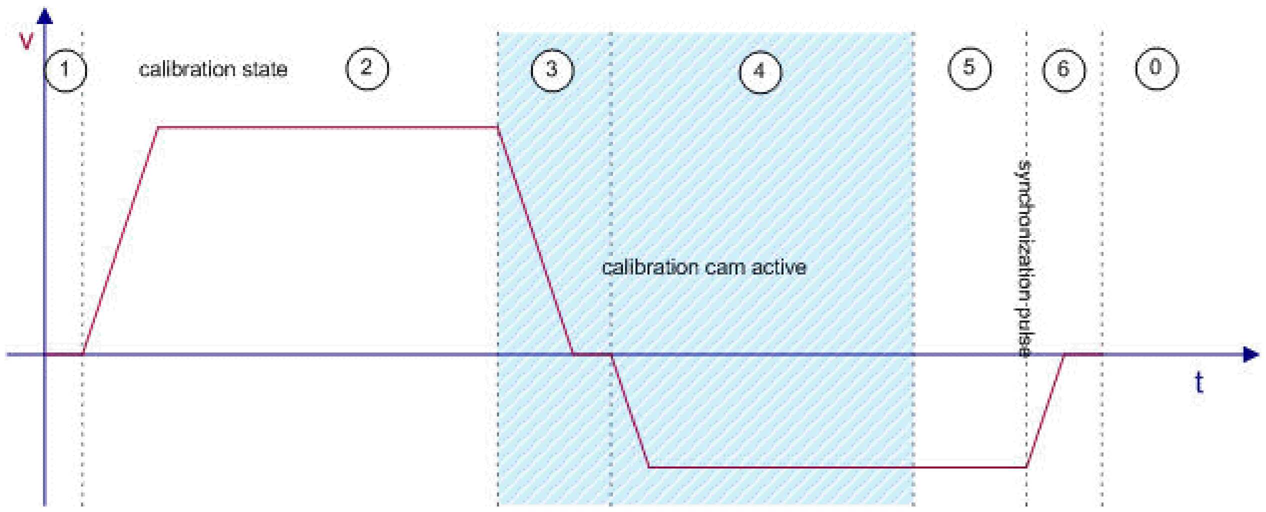
```
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	TRUE when the axis has been calibrated and the movement is complete.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted" or "Error" is set.
Active	BOOL	Currently not implemented - "Active" indicates that the command is running. If the command has been buffered, it may only become active after a running command has been terminated.

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

Note

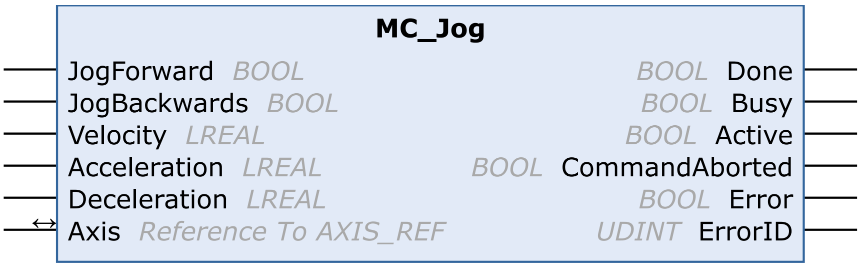
The referencing process has several phases. The following diagram illustrates the sequence after starting function block MC_Home with the individual phases for the case HomingMode = MC_DefaultHoming.



Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.2 MC_Jog (Manual Motion)



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.

Inputs

```
VAR_INPUT
    JogForward   : BOOL;
    JogBackwards : BOOL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
END_VAR
```

Name	Type	Description
JogForward	BOOL	The command is executed with a rising edge, and the axis is moved in positive direction of travel. During the motion no further signal edges are accepted (this includes the "JogBackwards" input).
JogBackwards	BOOL	The command is executed with a rising edge, and the axis is moved in negative direction of travel. "JogForward" and "JogBackwards" should be triggered alternatively, although they are also mutually locked internally.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0).
Deceleration	LREAL	Deceleration (≥0).

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

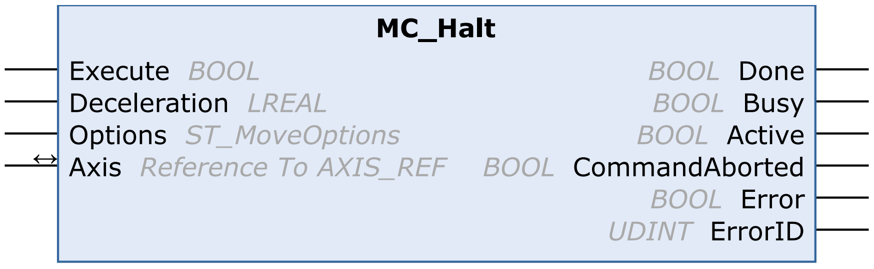
Name	Type	Description
Done	BOOL	TRUE if a movement was successfully completed.
Busy	BOOL	TRUE as soon as the function block is active. FALSE if it is in the default state. Only then can a further edge be accepted at the jog inputs.
Active	BOOL	Indicates that the axis is moved via the jog function.
CommandAborted	BOOL	TRUE if the process is interrupted by an external event, e.g. by the call up of MC_Stop [▶ 37].
Error	BOOL	TRUE, if an error occurs.
ErrorID	UDINT	If the error output is set, this parameter supplies the error number.

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3 Point to point motion

6.3.1 MC_Halt



The function block MC_Halt stops an axis with a defined deceleration ramp.

In contrast to MC_Stop [▶ 37], the axis is not locked against further movement commands. The axis can therefore be started by another command after the stop.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Deceleration	LREAL	Deceleration If the value is ≤ 0, the deceleration parameterized with the last Move command is used. For safety reasons MC_Halt and MC_Stop [▶ 37] cannot be executed with weaker dynamics than the currently active motion command. The parameterization is adjusted automatically, if necessary.
Options	ST_MoveOptions	Data structure (ST_MoveOptions [▶ 41]), which contains additional, rarely required parameters. The input can normally remain unused.

See also: General rules for MC function blocks [▶ 11].

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

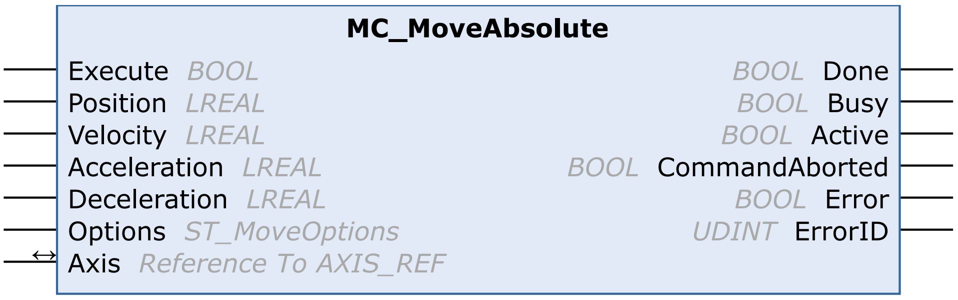
Name	Type	Description
Done	BOOL	TRUE if the axis has been stopped and is stationary.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done", "CommandAborted"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

See also: General rules for MC function blocks [▶ 11].

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.2 MC_MoveAbsolute



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

MC_MoveAbsolute is predominantly used for linear axis systems. For modulo axes the position is not interpreted as a modulo position, but as an absolute position in continuous absolute coordinate system. Alternatively, the function block MC_MoveModulo [▶ 28] can be used for modulo positioning.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Position	LREAL	Absolute target position to be used for positioning.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (>0)
Deceleration	LREAL	Deceleration (>0)
Options	ST_MoveOptions	Data structure (ST_MoveOptions [▶ 41]) containing additional, rarely used parameters. The input can normally remain unused.

See also: General rules for MC function blocks [▶ 11].

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

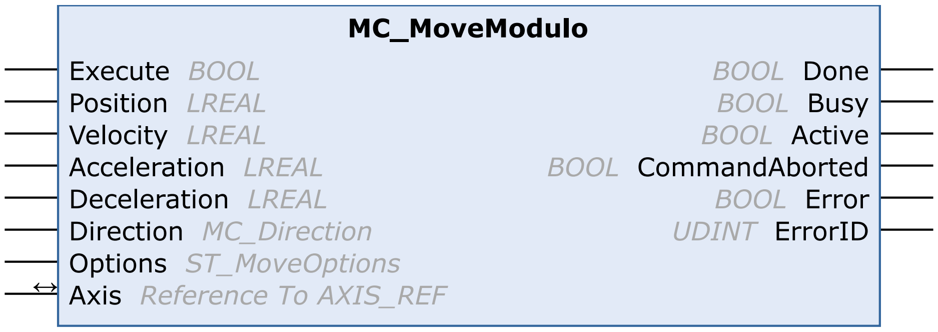
Name	Type	Description
Done	BOOL	TRUE when the target position has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done",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

See also: General rules for MC function blocks [▶ 11].

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.3 MC_MoveModulo



The function block MC_MoveModulo is used to execute a positioning operation, which refers to the modulo position of an axis. The basis for a modulo rotation is the adjustable parameter "Modulo Factor" of the AXIS_REF structure (Axis.Parameter.ModuloFactor, 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

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.

Special cases

Particular attention should be paid to the behavior when requesting one or more complete modulo rotations. No movement is performed if the axis is at an exact set position of 90° and is positioned at 90°, for example. If 450° in positive direction is requested, the axis performs 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. This means that the axis is no longer exactly at 90°, but slightly below or above this value. These cases will give rise either to a minimum positioning to 90 degrees, or on the other hand a complete rotation.

Depending on the use case, it may be more effective for complete modulo rotations to calculate the desired target position on the basis of the current absolute position, and then to position using the function block MC_MoveAbsolute [▶ 26].

See also: Modulo positioning [▶ 30]

Inputs

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

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

Name	Type	Description
Position	LREAL	Modulo target position to be used for positioning. If the axis is started from standstill, positions greater than 360° result in additional turns. Negative positions are not permitted.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0)
Deceleration	LREAL	Deceleration (≥0)
Direction	MC_Direction	Positive or negative direction of travel (type MC_Direction [▶ 41]). If the axis is started during a motion, the direction may not be reversed.
Options	ST_MoveOptions	Data structure (ST_MoveOptions [▶ 41]) containing additional, rarely used parameters. The input can normally remain unused.

See also: General rules for MC function blocks [▶ 11].

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▶ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

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

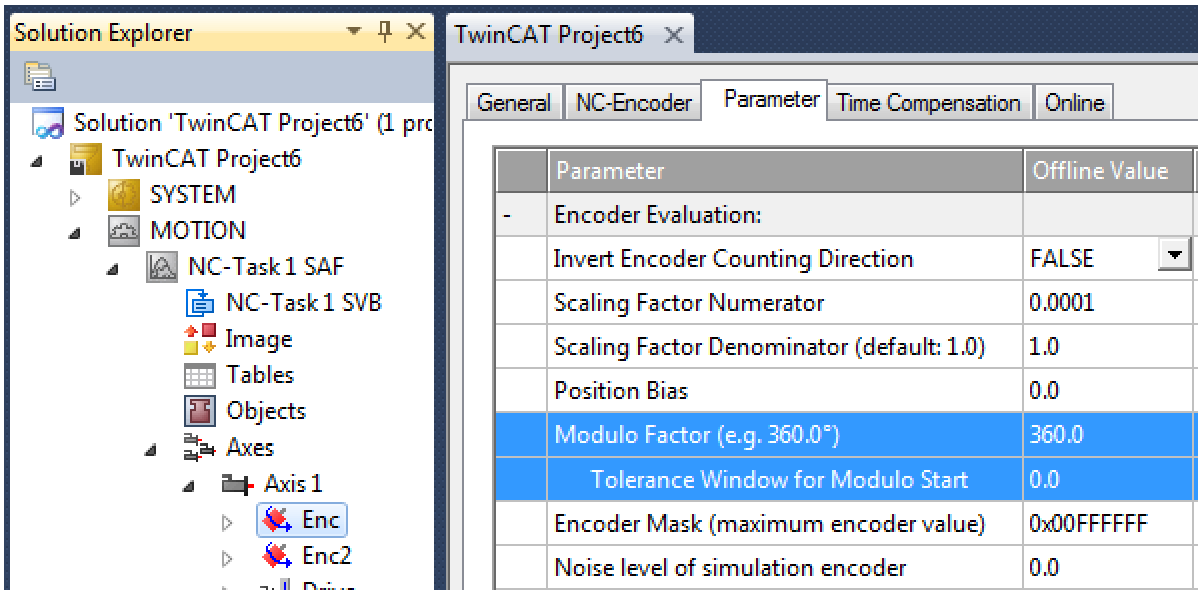
See also: General rules for MC function blocks [▸ 11].

6.3.4 Modulo positioning

Modulo positioning (MC_MoveModulo [▸ 28]) 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 AXIS_REF parameters

Modulo positioning generally refers to a modulo period to be set under *Axis Parameter. ModuloFactor* of the corresponding AXIS_REF [▸ 39]. In addition, appropriate settings are required in the drive parameters (e.g. EL72xx). The examples on this page are given based on a rotary axis with a modulo period of 360°.



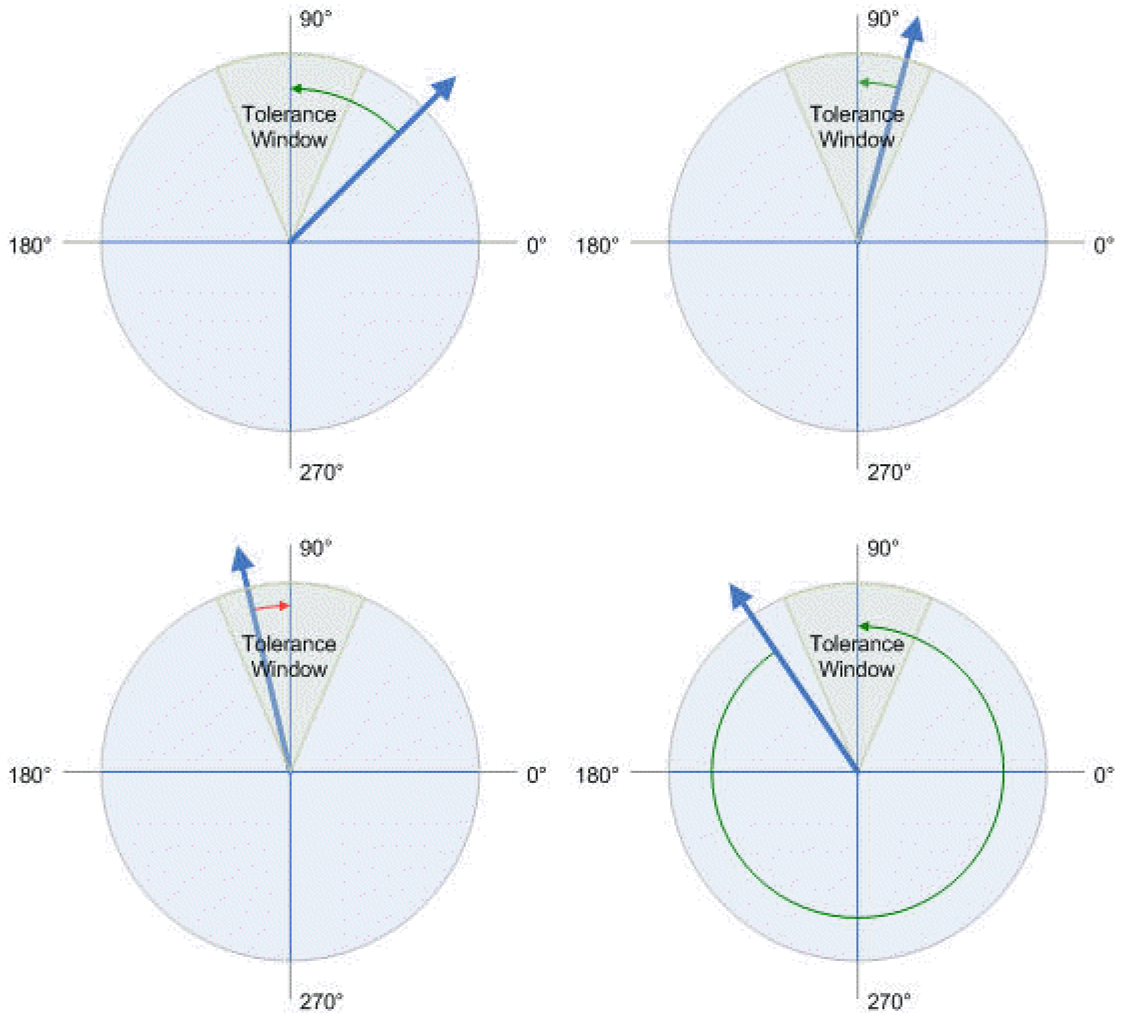
	Parameter	Offline Value
-	Encoder Evaluation:	
	Invert Encoder Counting Direction	FALSE
	Scaling Factor Numerator	0.0001
	Scaling Factor Denominator (default: 1.0)	1.0
	Position Bias	0.0
	Modulo Factor (e.g. 360.0°)	360.0
	Tolerance Window for Modulo Start	0.0
	Encoder Mask (maximum encoder value)	0x00FFFFFF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

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 [▸ 14], controller enable with MC_Power [▸ 13]) 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.



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° 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 [▶ 41]). For positioning along the shortest path, target positions of more than 360° are not sensible, because the movement towards the target is always direct. In contrast to positive or negative direction, it is therefore not possible to carry out several turns before the axis moves to the target.

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

The following table shows some positioning examples:

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

Direction (modulo start type)	Absolute starting position	Modulo target position	Relative travel path	Absolute end position	Modulo end position
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°.

Direction (modulo start type)	Absolute starting position	Modulo target position	Relative travel path	Absolute end position	Modulo end position
Positive direction	90.00	90.00	0.00	90.00	90.00
Positive direction	91.10	90.00	358.90	450.00	90.00
Positive direction	88.90	90.00	1.10	90.00	90.00
Positive direction	90.00	450.00	360.00	450.00	90.00
Positive direction	91.10	450.00	718.90	810.00	90.00
Positive direction	88.90	450.00	361.10	450.00	90.00
Positive direction	90.00	810.00	720.00	810.00	90.00
Positive direction	91.10	810.00	1078.90	1,170.00	90.00
Positive direction	88.90	810.00	721.10	810.00	90.00
Negative direction	90.00	90.00	0.00	90.00	90.00
Negative direction	91.10	90.00	-1.10	90.00	90.00
Negative direction	88.90	90.00	-358.90	-270.00	90.00
Negative direction	90.00	450.00	-360.00	-270.00	90.00
Negative direction	91.10	450.00	-361.10	-270.00	90.00
Negative direction	88.90	450.00	-718.90	-630.00	90.00
Negative direction	90.00	810.00	-720.00	-630.00	90.00
Negative direction	91.10	810.00	-721.10	-630.00	90.00
Negative direction	88.90	810.00	-1078.90	-990.00	90.00

Modulo calculations within the PLC program

All positioning jobs on an axis are executed based of the set position. The current actual position is only used for control purposes. If a new target position is to be calculated in a PLC program based on the current position, this calculation must be performed with the current set position of the axis (Axis.Status.ModuloSetPos and Axis.Status.ModuloSetTurns).

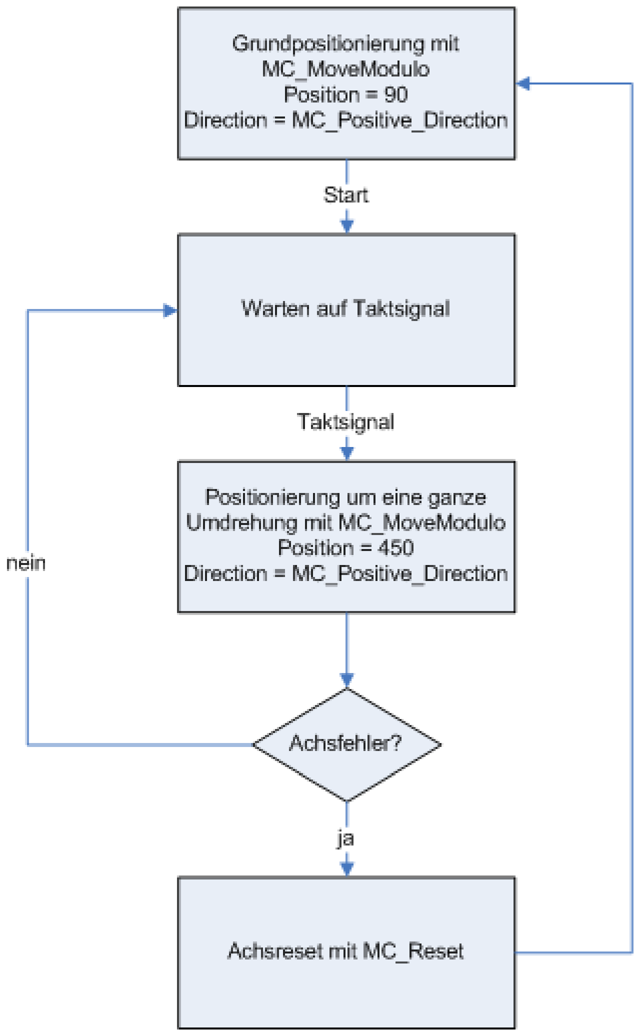
It is not advisable to perform order calculations based on the actual modulo position, which is available in the axis status (ModuloActPos and ModuloActTurns), Due to the greater or lesser control deviation of the axis, errors in the programmed sequence, such as undesired rotations, could occur.

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.

Since the axis may only be pre-positioned, the motion command MC_MoveModulo [▶ 28] 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 [▶ 28]) 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 [▶ 14] 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 so that this position is approached as quickly as possible. No motion occurs if the axis is already at the starting position.

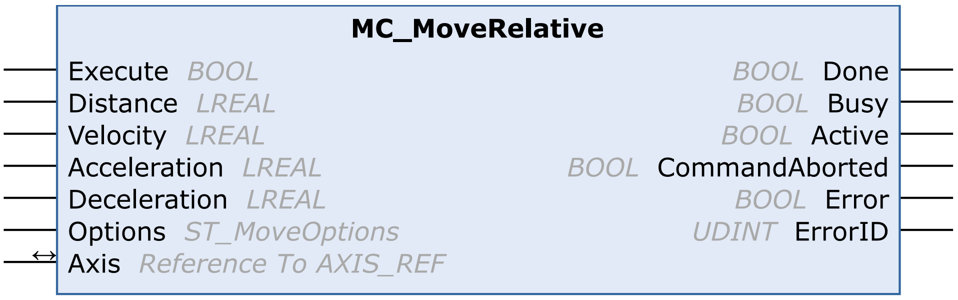

Grundpositionierung mit MC_MoveModulo Position = 90 Direction = MC_Positive_Direction
Start
Warten auf Taktsignal
Taktsignal
Positionierung um eine ganze Umdrehung mit MC_MoveModulo Position = 450 Direction = MC_Positive_Direction
Achsfehler?
nein
ja
Achsreset mit MC_Reset

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	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.5 MC_MoveRelative



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

Inputs

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

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge.
Distance	LREAL	Relative distance to be used for positioning.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0)
Deceleration	LREAL	Deceleration (≥0)
Options	ST_MoveOptions	Data structure (ST_MoveOptions [▸ 41]) containing additional, rarely used parameters. The input can normally remain unused.

See also: General rules for MC function blocks [▸ 11].

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▸ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

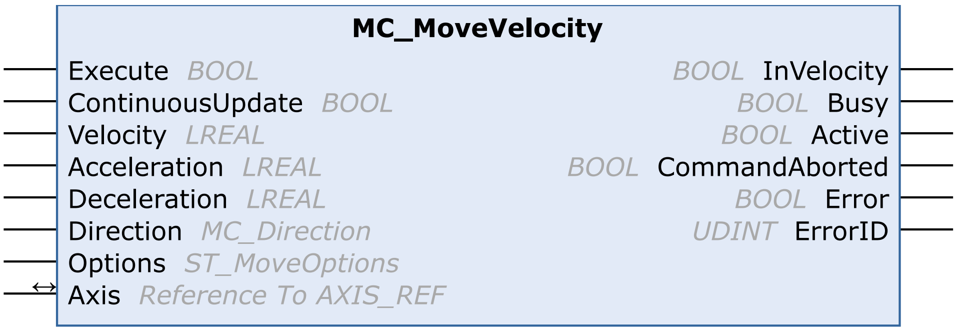
Name	Type	Description
Done	BOOL	TRUE when the target position has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the movement command is processed. If "Busy" is FALSE, the function block is ready for a new order. At the same time, one of the outputs "Done",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

See also: General rules for MC function blocks [▶ 11].

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.6 MC_MoveVelocity



The function block MC_MoveVelocity starts an endless travel with a specified velocity and direction. The movement can be stopped through a Stop command.

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

Inputs

```
VAR_INPUT
    Execute      : BOOL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Direction    : MC_Direction := MC_Positive_Direction;
    Options      : ST_MoveOptions;
END_VAR
```

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

Name	Type	Description
ContinuousUpdat e	BOOL	If ContinuousUpdate is TRUE, the dynamics can be changed according to the Velocity, Acceleration and Deceleration inputs during the processing of the command with a rising edge at the Execute input and brought into effect as quickly as possible.
Velocity	LREAL	Maximum travel velocity (>0).
Acceleration	LREAL	Acceleration (≥0)
Deceleration	LREAL	Deceleration (≥0)
Direction	MC_Direction	Positive or negative direction of travel (type MC_Direction [▸ 41])
Options	ST_MoveOptions	Data structure (ST_MoveOptions [▸ 41]) containing additional, rarely used parameters. The input can normally remain unused.

See also: General rules for MC function blocks [▸ 11].

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▸ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

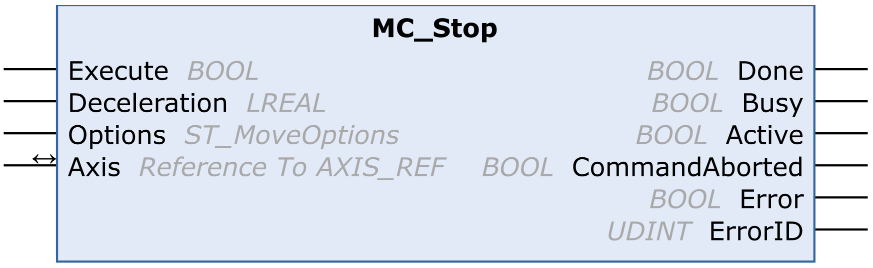
Name	Type	Description
InVelocity	BOOL	After the axis acceleration, the InVelocity output assumes the value TRUE once the requested target velocity has been reached.
Busy	BOOL	TRUE as soon as the command is started with "Execute" and as long as the function block is active. If "Busy" is FALSE, the function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

See also: General rules for MC function blocks [▶ 11].

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.7 MC_Stop



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

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

The locking of the axis is canceled with an MC_Reset [▶ 14].

Alternatively, the axis can be stopped with MC_Halt [▶ 25] without locking. MC_Halt is preferable for normal movements.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is executed with a rising edge. The axis is locked during the stop. The axis can only be restarted once the Execute signal has been set to FALSE after the axis has stopped.
Deceleration	LREAL	Deceleration If the value is 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.
Options	ST_MoveOptions	Data structure (ST_MoveOptions [▶ 41]) containing additional, rarely used parameters. The input can normally remain unused.

See also: General rules for MC function blocks [▶ 11].

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF [▸ 39]	Axis data structure that unambiguously addresses an axis in the system. Among other parameters it contains the current axis status, including position, velocity or error state.

Outputs

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

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

See also: General rules for MC function blocks [▸ 11].

Requirements

Development environment	PLC libraries to include
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

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 DRIVE. It is added to MC function blocks as axis reference.

```
TYPE AXIS_REF :
VAR_INPUT
    PlcToDrive AT %Q* : PLCTODRIVE_AXIS_REF;
    Parameter         : ST_AxisParameters;
END_VAR
VAR_OUTPUT
    DriveToPlc AT %I* : DRIVETOPLC_AXIS_REF;
    WcState    AT %I* : DRIVETOPLC_WCSTATE;
    InfoData   AT %I* : DRIVETOPLC_IFODATA;
    Status            : ST_AxisStatus;
END_VAR
END_TYPE
```

For the library function, it is essential to parameterize the `EncoderScalingFactor` and `MaxVelocity parameters` of the ST_AxisParameters data type.

AXIS_REF elements

Name	Description
PlcToDrive	Data structure that is cyclically exchanged between PLC and DRIVE. The MC function blocks communicate with the DRIVE via this data structure and send control information and commands from the PLC to the DRIVE. The data structure is automatically placed in the output process image of the PLC and must be connected to the input process image of a DRIVE in the TwinCAT development environment.
Parameter	Data structure for parameterization of the axis (type: ST_AxisParameters [▸ 42]).
DriveToPlc	Data structure that is cyclically exchanged between PLC and DRIVE. The MC function blocks communicate with the DRIVE via this data structure and receive status information from the DRIVE. The data structure is automatically placed in the input process image of the PLC and must be connected to the output process image of a DRIVE in the TwinCAT System Manager. The DRIVETOPLC 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.
WcState	Data structure that is cyclically exchanged between PLC and DRIVE and contains the WcState of the DRIVE. The data structure is automatically placed in the input process image of the PLC and must be connected to the output process image of a DRIVE in the TwinCAT development environment (type: DRIVETOPLC_WCSTATE [▸ 40]).
InfoData	Data structure that is cyclically exchanged between PLC and DRIVE and contains ADS information for accessing DRIVE parameters. The data structure is automatically placed in the input process image of the PLC and must be connected to the output process image of a DRIVE in the TwinCAT development environment (type: DRIVETOPLC_INFODATA [▸ 40]).

Name	Description
Status	Data structure containing additional or processed status information for an axis (type: ST_AxisStatus [▶ 42]). This data structure is not refreshed cyclically, but has to be updated through the PLC program. For this purpose MC_ReadStatus or alternatively the action "ReadStatus" of AXIS_REF can be called (see sample).

Sample:

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

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

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

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

7.1.2 DRIVETOPLC_INFODATA

The structure DRIVETOPLC_INFODATA is part of the structure AXIS_REF [▶ 39].

```
TYPE DRIVETOPLC_INFODATA :
    State       : UINT;
    AdsAddr     : AMSADDR;
END_TYPE
```

7.1.3 DRIVETOPLC_WCSTATE

The structure DRIVETOPLC_WCSTATE is part of the structure AXIS_REF [▶ 39].

```
TYPE DRIVETOPLC_WCSTATE :
    WcState      : BIT;
    InputToggle : BIT;
END_TYPE
```

7.2 Homing

7.2.1 E_EncoderReferenceMode

```
TYPE E_EncoderReferenceMode :
(
    EncoderReferenceMode_CamAtDigitalInput,    (* Cam is connected to  digital input*)
);
END_TYPE
```

7.2.2 MC_HomingMode

This data type is used to parameterize the function block MC_Home [▶ 21].

```
TYPE MC_HomingMode :
(
    MC_DefaultHoming       :=0,   (* default homing as defined in the SystemManager encoder parameter
s *)
    MC_Direct              :=4,   (* Static Homing forcing position from user reference *)
    MC_Block               :=6,   (* Homing against hardware parts blocking movement *)
    MC_ForceCalibration    :=7,   (* set the calibration flag without perfomring any motion or changi
ng the position *)
    MC_ResetCalibration    :=8    (* resets the calibration flag without perfomring any motion or cha
nging the position *)
);
END_TYPE
```

7.2.3 ST_HomingOptions

```
TYPE ST_HomingOptions :
STRUCT
(
    SearchDirection    : MC_Direction := MC_Direction.MC_Undefined_Direction;
                (* search direction *)
    SearchVelocity     : LREAL;
                (* search velocity *)
    SyncDirection      : MC_Direction := MC_Direction.MC_Undefined_Direction;
                (* synchronization direction *)
    SyncVelocity       : LREAL;
                (* synchronization velocity *)
    ReferenceMode      : E_EncoderReferenceMode := E_EncoderReferenceMode.ENCODERREFERENCEMODE_CAMATD
IGITALINPUT;    (* Mode of reference sequence *)

    Acceleration       : LREAL;
                (* user defined acceleration *)
    Deceleration       : LREAL;
                (* user defined deceleration *)
);
END_TYPE
```

Name	Data type	Description
SearchDirection	MC_Direction	Direction in which the referencing cam is to be searched.
SearchVelocity	LREAL	Velocity at which the referencing cam is to be searched.
SyncDirection	MC_Direction	Direction in which the falling edge of the referencing cam is searched after the referencing cam has been detected.
ReferenceMode	E_EncoderReferenceMode	Referencing mode (currently only ENCODERREFERENCEMODE_CAMATDIGITALINPUT).
Acceleration	LREAL	Acceleration for the reference run
Deceleration	LREAL	Deceleration for the reference run

7.3 Motion

7.3.1 MC_Direction

This enumeration type contains the possible directions of movement for the function blocks MC_MoveVelocity [▸ 35] and MC_MoveModulo [▸ 28].

```
TYPE MC_Direction :
(
    MC_Undefined_Direction  := 0,
    MC_Positive_Direction   := 1,
    MC_Shortest_Way         := 2,
    MC_Negative_Direction   := 3
);
END_TYPE
```

7.3.2 ST_MoveOptions

This data type is intended for possible future optional motion commands settings such as MC_MoveAbsolute or MC_Halt.

```
TYPE ST_MoveOptions :
STRUCT
END_STRUCT
END_TYPE
```

7.4 Status and parameter

7.4.1 MC_AxisStates

This data type describes the operating states according to the PlcOpen state diagram [▸ 9].

```
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
);
END_TYPE
```

See also: General rules for MC function blocks [▸ 11].

7.4.2 ST_AxisParameters

This data type contains basic necessary axis parameters.

```
TYPE ST_AxisParameters :
STRUCT
    EncoderScalingFactor     : LREAL;
    MaxVelocity              : LREAL;
    ModuloFactor             : LREAL := 360.0;
END_STRUCT
END_TYPE
```

Formula for calculating the parameters:

$$EncoderScalingFactor = \frac{Feed\ constant}{Position\ Increments} = \frac{360°}{32\ Bit} = \frac{360°}{2^{32}} = 8.381903173490871 * 10^{-8}$$

$$MaxVelocity = Motor\ speed\ limitation * Feed\ constant = \frac{1615}{60}\frac{Revolutions}{s} * 360\frac{°}{Revolution} = 9690.0\frac{°}{s}$$

Motor speed Limitation

"Motor speed limitation" depends on the configured voltage and the motor used. For the servo terminals, this value can be read from CoE object 0x8011:1B, for example.

7.4.3 ST_AxisStatus

This data type contains extensive status information about an axis. The data structure must be updated in each PLC cycle by calling MC_ReadStatus or by calling the action Axis.ReadStatus() or Axis() (AXIS_REF [▸ 39]).

```
TYPE ST_AxisStatus :
STRUCT
    AxisId                : UDINT;
    AxisName              : STRING;

    ActPos                : LREAL;
    ModuloActPos          : LREAL;
    ActVelo               : LREAL;
    ActAcceleration       : LREAL;

    SetPos                : LREAL;
    ModuloSetPos          : LREAL;
    SetVelo               : LREAL;
    SetAcceleration       : LREAL;

    PosDiff               : LREAL;

    TargetPosition        : LREAL;
    TargetVelocity        : LREAL;
    TargetAcceleration    : LREAL;
    TargetDeceleration    : LREAL;

    InfoData1             : LREAL;
    InfoData2             : LREAL;
    InfoData3             : LREAL;

    DigitalInput1         : BOOL;
    DigitalInput2         : BOOL;

    CmdNo                 : UINT;
    CmdState              : UINT;

    MotionState           : MC_AxisStates; (* motion state in the PLCopen state diagram *)

    Error                 : BOOL;  (* axis error state *)
    ErrorId               : UDINT; (* axis error code *)

    (* statemachine states: *)
    ErrorStop             : BOOL;
    Disabled              : BOOL;
    Stopping              : BOOL;
    StandStill            : BOOL;
    DiscreteMotion        : BOOL;
    ContinuousMotion      : BOOL;
    Homing                : BOOL;

    (* additional status *)
    ConstantVelocity      : BOOL;
    Accelerating          : BOOL;
    Decelerating          : BOOL;

    (* Status *)
    Operational           : BOOL;
    ControlLoopClosed     : BOOL; (* operational and position control active *)
    HasJob                : BOOL;
    HasBeenStopped        : BOOL;
    InTargetPosition      : BOOL;
    Protected             : BOOL;
    Homed                 : BOOL;
    HomingBusy            : BOOL;
    MotionCommandsLocked  : BOOL; (* stop 'n hold *)

    Moving                : BOOL;
    PositiveDirection     : BOOL;
    NegativeDirection     : BOOL;
    NotMoving             : BOOL;

    PTPmode               : BOOL;
    DriveDeviceError      : BOOL;
    DrivePositioningError: BOOL;
    IoDataInvalid         : BOOL;
END_STRUCT
END_TYPE
```

7.5 Touch probe

7.5.1 E_SignalEdge

`Edge` defines whether the rising or falling edge of the trigger signal is evaluated.

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

Name	Description
RisingEdge	Rising signal edge
FallingEdge	Falling signal edge

7.5.2 E_SignalSource

```
TYPE E_SignalSource :
(
    SignalSource_Default,    (* undefined or externally configured *)
    SignalSource_ZeroPulse := 128, (* encoder zero pulse *)
);
END_TYPE
```

Name	Description
SignalSource	Optionally defines the signal source, if it can be selected via the controller. In many cases the signal source is permanently configured in the drive and should then be set to the default value "SignalSource_Default".

7.5.3 E_TouchProbe

```
TYPE E_TouchProbe :
(
    TouchProbe1 := 1, (* 1st hardware probe unit with Sercos, CanOpen, KL5xxx and others *)
    PlcEvent := 10    (* simple PLC signal TRUE/FALSE *)
);
END_TYPE
```

Name	Description
TouchProbe	Defines the latch unit (probe unit) within the encoder hardware used.
PlcEvent	If the signal source "TouchProbe" is set to the type "PlcEvent", a rising edge on these variables triggers the recording of the current axis position. "PlcEvent" is not a real latch function, but depends on the cycle time.

7.5.4 MC_TouchProbeRecordedData

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

Name	Data type	Description
Counter	LREAL	Counter indicating how many valid edges were detected in the last cycle. Detection of multiple edges is only implemented in mode TOUCHPROBEMODE_CONTINUOUS under SERCOS / SOE and must be supported explicitly by the hardware (e.g. AX5000).
RecordedPosition	LREAL	Axis position recorded at the point in time of the trigger signal. This corresponds to the absolute axis position or the modulo axis position, depending on the parameterization.
AbsolutePosition	LREAL	Absolute axis position detected at the point in time of the trigger signal.
ModuloPosition	LREAL	Modulo axis position recorded at the point in time of the trigger signal.

7.5.5 TRIGGER_REF

```
TYPE TRIGGER_REF :
STRUCT
    TouchProbe      : E_TouchProbe; (* probe unit definition *)
    SignalSource    : E_SignalSource; (* optional physical signal source used by the probe unit *)
    Edge            : E_SignalEdge; (* rising or falling signal edge *)
    Mode            : E_TouchProbeMode; (* single shot or continuous monitoring *)
    PlcEvent        : BOOL; (* PLC trigger signal input when TouchProbe signal source is set to 'Plc
Event' *)
    ModuloPositions : BOOL; (* interpretation of FirstPosition, LastPosition and RecordedPosition as
 modulo positions when TRUE *)
END_STRUCT
END_TYPE
```

```
TYPE E_TouchProbeMode :
(
    TOUCHPROBEMODE_SINGLE := 1
);
END_TYPE
```

Mode: Specifies the operation mode of the latch unit. In single mode only the first edge is recorded.

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 [▸ 17] are then also absolute.
If "ModuloPositions" is TRUE, all positions are interpreted in modulo mode 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.

See previous sections for further explanations.

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_Tc3_DriveMotionControl : ST_LibVersion;
END_VAR
```

stLibVersion_Tc3_DriveMotionControl: Version information of the Tc3_DriveMotionControl library (type: ST_LibVersion).

To check whether the version you have is the version you need, use the function F_CmpLibVersion (defined in the Tc2_System library).

Compare versions

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

9 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Trademark statements

More Information:
www.beckhoff.com/te1000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com