

Manual | EN

PLC Library: TcMC

TwinCAT 2 | TX1200

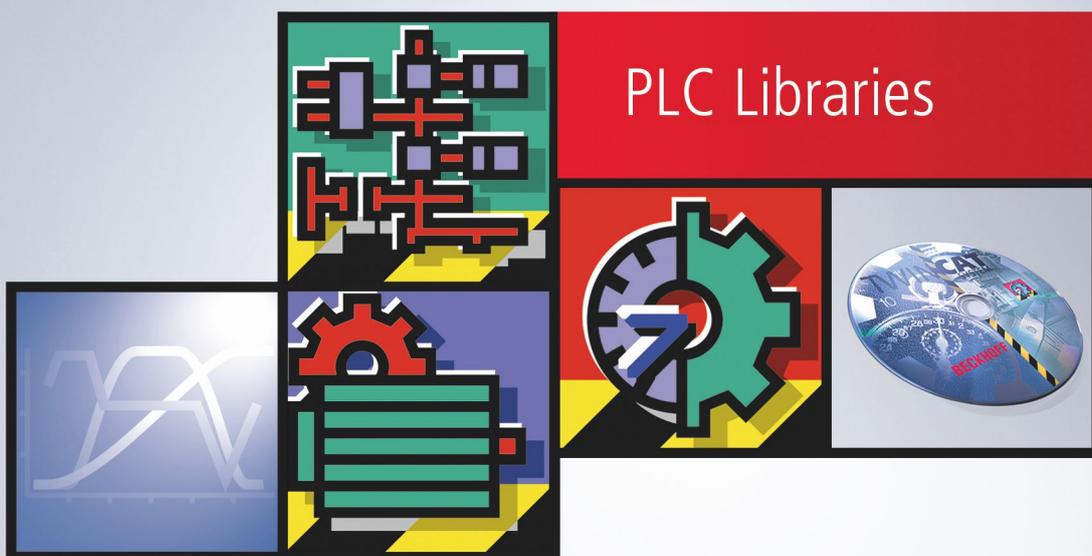


Table of contents

1 Foreword	5
1.1 Notes on the documentation	5
1.2 For your safety	5
1.3 Notes on information security.....	7
2 Overview	8
3 Function blocks - motion	9
3.1 MC_MoveAbsolute	9
3.2 MC_MoveRelative	10
3.3 MC_MoveVelocity	11
3.4 MC_MoveModulo	12
3.5 Notes on modulo positioning.....	13
3.6 MC_MoveAbsoluteOrRestart	19
3.7 MC_NewPos	20
3.8 MC_NewPosAndVelo.....	21
3.9 MC_MoveSuperImposed	22
3.10 MC_MoveSuperImposedExt	23
3.11 Application examples for MC_MoveSuperImposedExt	24
3.12 MC_AbortSuperposition	27
3.13 MC_Jog.....	28
3.14 MC_Home	30
3.15 MC_Stop	32
3.16 MC_OrientedStop	33
3.17 MC_GearIn.....	34
3.18 MC_GearInFloat.....	35
3.19 MC_GearInDyn	36
3.20 MC_GearOut.....	38
3.21 MC_GearOutExt.....	39
4 Function blocks - administration	41
4.1 MC_Power	41
4.2 MC_Reset	42
4.3 MC_ReadActualPosition	42
4.4 MC_ReadStatus	43
4.5 MC_ReadAxisError	44
4.6 MC_ReadParameter	45
4.7 MC_ReadBoolParameter	46
4.8 MC_ReadParameterSet.....	47
4.9 MC_ReadAxisComponents.....	48
4.10 MC_WriteParameter	49
4.11 MC_WriteBoolParameter	49
4.12 MC_SetActualPosition	50
4.13 MC_SetActualPositionOnTheFly.....	51
4.14 MC_SetOverride	52
4.15 MC_SetReferenceFlag.....	53

4.16	MC_ExtSetPointGenEnable	54
4.17	MC_ExtSetPointGenDisable	55
4.18	MC_ExtSetPointGenFeed	55
4.19	MC_OverrideFilter	56
4.20	MC_ChangeDynParam	57
4.21	MC_TouchProbe	59
4.22	MC_AbortTrigger	62
4.23	MC_PowerStepper	63
4.23.1	Notes on the MC_PowerStepper	64
5	Functions	69
5.1	F_GetVersionTcMC	69
6	Data types	70
6.1	MC_AxisPara	70
6.2	MC_Direction	71
6.3	MC_TouchProbe data structures	71
6.4	E_ReadMode	73
6.5	E_SuperpositionMode	73
6.6	ST_PowerStepperStruct	73
6.7	E_DestallMode	74
6.8	E_DestallDetectMode	74
7	Appendix	76
7.1	MC_Move.....Done	76

1 Foreword

1.1 Notes on the documentation

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

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

The trained specialists must always use the current valid documentation.

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

Disclaimer

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

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

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

Trademarks

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

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



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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

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

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

Third-party trademarks

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

1.2 For your safety

Safety regulations

Read the following explanations for your safety.

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings** DANGER**

Hazard with high risk of death or serious injury.

 WARNING

Hazard with medium risk of death or serious injury.

 CAUTION

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

Warning of damage to property or environment**NOTICE**

The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Overview

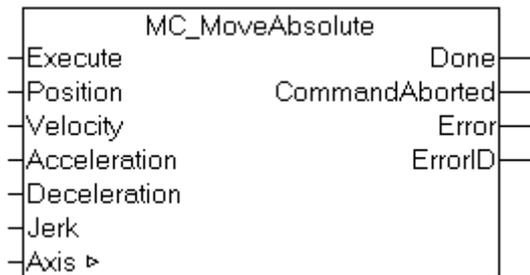
This library contains blocks that were standardised by PLCopen, the IEC61131 user organisation.

The specification can be found at www.PLCopen.org. The TcMC.lib contains the basic blocks. The camming blocks are supplied separately with the camming package.



3 Function blocks - motion

3.1 MC_MoveAbsolute



Absolute positioning is carried out with the function block MC_MoveAbsolute.

VAR_INPUT

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

Execute : The command is executed with rising 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). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

VAR_OUTPUT

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

Done : Becomes TRUE once the target position is reached. [More Information... \[▶ 76\]](#)

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop, or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

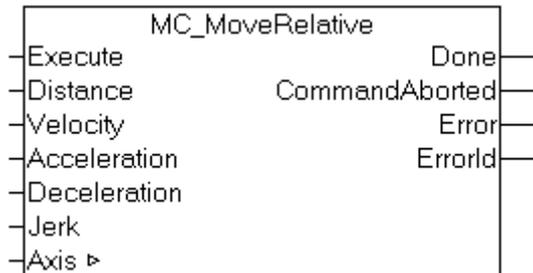
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.2 MC_MoveRelative



Relative positioning is carried out with the function block MC_MoveRelative.

VAR_INPUT

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

Execute : The command is executed with rising 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). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

VAR_OUTPUT

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

Done : Becomes TRUE once the target position is reached. [More Information...](#) [[▶ 76](#)]

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop, or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

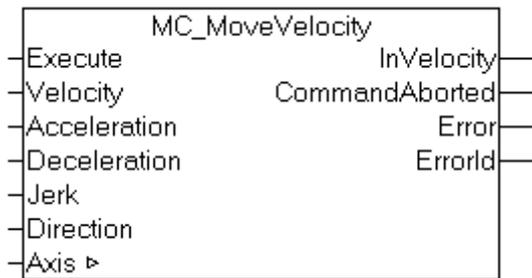
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.3 MC_MoveVelocity



Continuous travel is carried out with the function block MC_MoveVelocity.

VAR_INPUT

```
VAR_INPUT
  Execute      : BOOL;
  Velocity     : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
  Direction    : MC_Direction;
END_VAR
```

Execute : The command is executed with rising edge.

Velocity : Maximum velocity increase during travel (>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). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

Direction : [Direction \[► 71\]](#) for the motion.

VAR_OUTPUT

```
VAR_OUTPUT
  InVelocity      : BOOL;
  CommandAborted : BOOL;
  Error           : BOOL;
  ErrorID        : UDINT;
END_VAR
```

InVelocity : Becomes TRUE once the target velocity is reached.

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop, or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

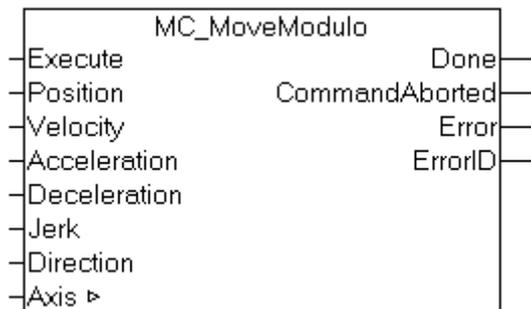
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.4 MC_MoveModulo



The MC_MoveModulo function block carries out a positioning referenced to the modulo position of an axis. Modulo rotation is based on the adjustable axis parameter *modulo factor* (e.g. 360).

Special cases: Special attention must be paid to the behavior when one or more complete modulo rotations are requested. 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 case, it may be more effective for complete modulo rotations to calculate the desired destination position based on the current absolute position, and then to position using the MC_MoveAbsolute block.

Note: Modulo positioning, like absolute positioning, is available for all axes, regardless of the *Modulo* setting in the TwinCAT System Manager. The current absolute position, *fPosSoll*, can be read for any axis from the cyclic axis interface NCTOPLC_AXLESTRUCT.

Important: [Detailed information about modulo positioning \[► 13\]](#).

VAR_INPUT

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

Execute : The command is executed with rising edge.

Position : Absolute modulo destination position to be reached.

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). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

Direction : Positioning direction [MC Direction \[► 71\]](#) (forwards, reverse or the shortest route)

VAR_OUTPUT

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

Done : Becomes TRUE once the target position is reached. [More Information... \[► 76\]](#)

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.5 Notes on modulo positioning

Modulo positioning ([MC MoveModulo \[► 121\]](#)) is possible irrespective of the axis type. It may be used both for linear or rotary axes, because TwinCAT makes no distinction between these types. A modulo axis has a consecutive absolute position in the range $\pm\infty$. The modulo position of the axis is simply a piece of additional information about the absolute axis position. Modulo positioning represents the required target position in a different way. Unlike absolute positioning, where the user specifies the target unambiguously, modulo positioning has potential pitfalls, because the required target position may be interpreted in different ways.

Settings in the TwinCAT System Manager

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



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

Special features of axis resets

Axis positioning always refers to the set position. The set position of an axis is normally the target position of the last travel command. An axis reset ([MC_Reset](#) [► 42], controller activation with [MC Power](#) [► 41]) 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 will reset any following error that may have occurred. If this possibility is not considered, subsequent positioning may lead to unexpected behaviour.

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 of once again being exactly 90° . If an axis reset is carried out at this stage, the set position may happen to be somewhat smaller or greater. 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.

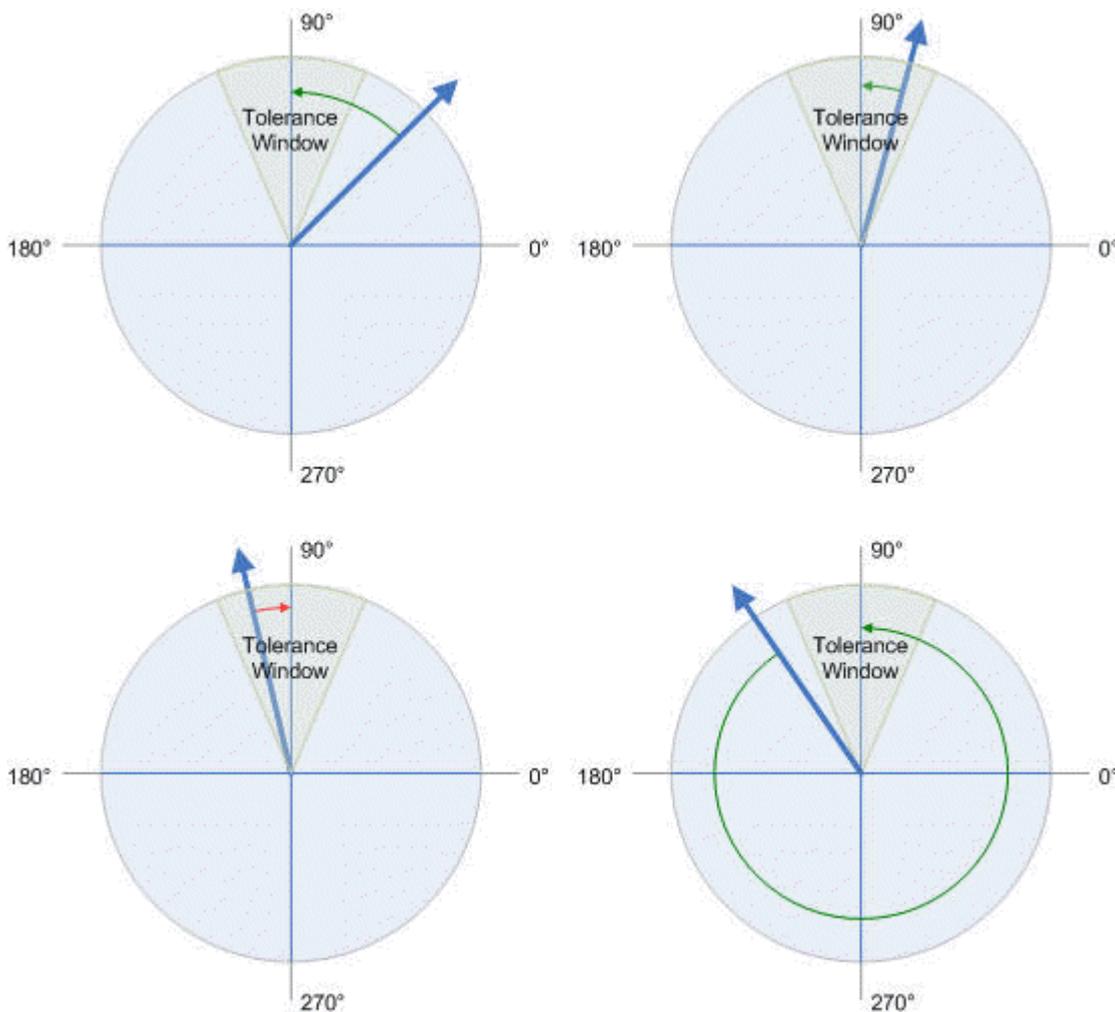


Fig. 1: Figure: Effect of the modulo tolerance window - modulo target position 90° in positive direction

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 |► 711). 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.

Important: 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 start position	Modulo target position	Relative travel	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° must 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 start position	Modulo target position	Relative travel	Absolute end position	Modulo end position	Remark
Positive direction	90.00	90.00	0.00	90.00	90.00	
Positive direction	90.90	90.00	-0.90	90.00	90.00	
Positive direction	91.10	90.00	358.90	450.00	90.00	outside TW
Positive direction	89.10	90.00	0.90	90.00	90.00	
Positive direction	88.90	90.00	1.10	90.00	90.00	outside TW
Positive direction	90.00	450.00	360.00	450.00	90.00	
Positive direction	90.90	450.00	359.10	450.00	90.00	
Positive direction	91.10	450.00	718.90	810.00	90.00	outside TW
Positive direction	89.10	450.00	360.90	450.00	90.00	
Positive direction	88.90	450.00	361.10	450.00	90.00	outside TW
Positive direction	90.00	810.00	720.00	810.00	90.00	
Positive direction	90.90	810.00	719.10	810.00	90.00	

Direction (modulo start type)	Absolute start position	Modulo target position	Relative travel	Absolute end position	Modulo end position	Remark
Positive direction	91.10	810.00	1078.90	1170.00	90.00	outside TW
Positive direction	89.10	810.00	720.90	810.00	90.00	
Positive direction	88.90	810.00	721.10	810.00	90.00	outside TW
Negative direction	90.00	90.00	0.00	90.00	90.00	
Negative direction	90.90	90.00	-0.90	90.00	90.00	
Negative direction	91.10	90.00	-1.10	90.00	90.00	outside TW
Negative direction	89.10	90.00	0.90	90.00	90.00	
Negative direction	88.90	90.00	-358.90	-270.00	90.00	outside TW
Negative direction	90.00	450.00	-360.00	-270.00	90.00	
Negative direction	90.90	450.00	-360.90	-270.00	90.00	
Negative direction	91.10	450.00	-361.10	-270.00	90.00	outside TW
Negative direction	89.10	450.00	-359.10	-270.00	90.00	
Negative direction	88.90	450.00	-718.90	-630.00	90.00	outside TW
Negative direction	90.00	810.00	-720.00	-630.00	90.00	
Negative direction	90.90	810.00	-720.90	-630.00	90.00	
Negative direction	91.10	810.00	-721.10	-630.00	90.00	outside TW
Negative direction	89.10	810.00	-719.10	-630.00	90.00	
Negative direction	88.90	810.00	-1078.90	-990.00	90.00	outside TW

Modulo calculations within the PLC program

In TwinCAT NC, all axis positioning tasks are executed based on the set position. The current actual position is only used for control purposes. If a PLC program is to calculate a new target position based on the current position, the current set position of the axis has to be used in the calculation. In contrast to the absolute set position *fPosSoll*, the modulo set position is not explicitly made available in the cyclic axis interface NCTOPLC_AXLESTRUCT. The modulo set position and the number of modulo rotations can be calculated using library functions from the *TcMath.lib* library.

```
ModuloSetPosition := MODABS( NcToPlc.fPosSoll, 360 );
```

```
ModuloSetTurns := MODTURNS( NcToPlc.fPosSoll, 360 );
```

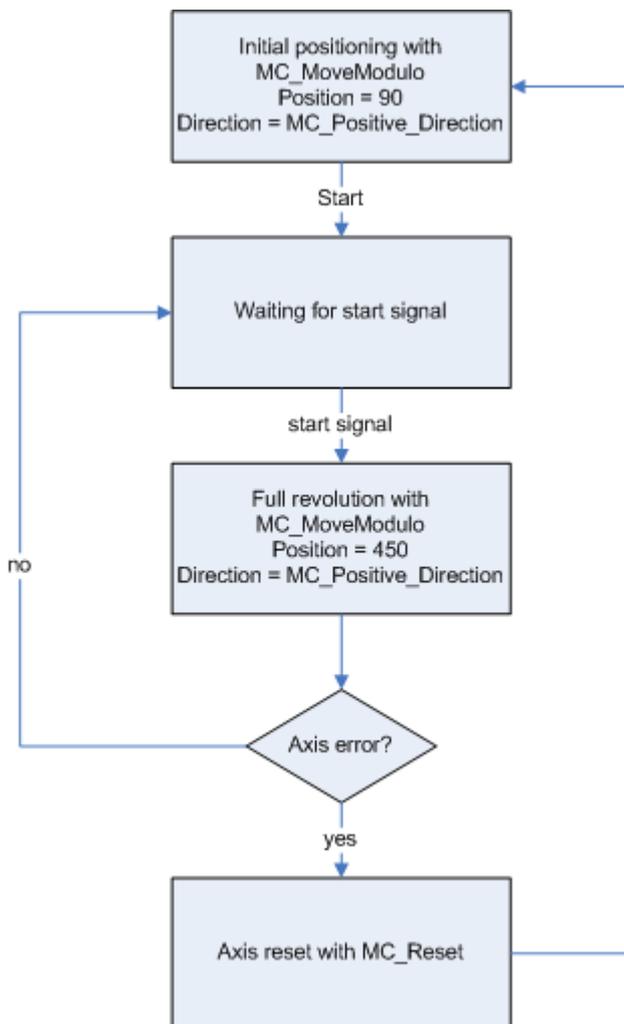
Task calculations based on the modulo actual position, which is available in the cyclic axis interface (*fModuloPos1st* and *nModuloTurns*), are not recommended, because axis control deviations may lead to errors in the programmed process, such as undesirable rotations.

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.

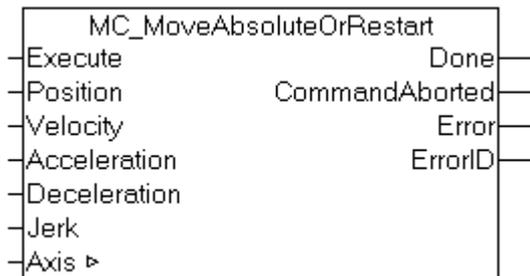
The modulo tolerance window is set to 1.5° in the System Manager. This ensures that undesirable axis turns after an axis reset are avoided. Since the axis may only be positioned in positive direction, the command MC_MoveModulo [► 12] with modulo start type *positive direction* (*MC_Positive_Direction*) 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 [► 12]) 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 [► 42] and subsequently (at the start of the step sequence) moved to its valid starting position. In this case, 90° is specified as the target position to enable this position to be reached as quickly as possible. No motion occurs if the axis is already at the starting position.



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

3.6 MC_MoveAbsoluteOrRestart



Along with the Function block MC_MoveAbsoluteOrRestart a movement from the standstill, as along with the block [MC_MoveAbsolute \[► 9\]](#) can be started. On the other hand target position and velocity are changed, if the building block during a movement is triggered.

VAR_INPUT

```

VAR_INPUT
Execute      : BOOL;    (* Start axis. Toggle while axis is moving to restart axis *)
Position     : LREAL;   (* target position *)
Velocity     : LREAL;   (* velocity *)
Acceleration : LREAL;   (* optional on axis start. not used for restart *)
Deceleration : LREAL;   (* optional on axis start. not used for restart *)
Jerk        : LREAL;   (* optional on axis start. not used for restart *)
END_VAR
    
```

Execute : The command is executed with rising edge. The function block can be retriggered while the axis is in motion.

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. The acceleration will only be used with a start from standstill.

Deceleration : Deceleration (≥0). If the value is 0, the standard deceleration from the axis configuration in the System Manager is used. The deceleration will only be used with a start from standstill.

Jerk : Jerk (≥0). If the value is 0, the standard jerk from the axis configuration in the System Manager is used. The jerk will only be used with a start from standstill.

VAR_OUTPUT

```

VAR_OUTPUT
Done          : BOOL;    (* move completed *)
CommandAborted : BOOL;   (* move aborted *)
Error         : BOOL;
ErrorID       : UDINT;
END_VAR
    
```

Done : Becomes TRUE once the target position is reached. The Done signal is set at the earliest after the end of the logical setpoint positioning (setpoint generator, HasJob flag). Depending on switched on monitoring functions, such as target position monitoring, Done is only set when the actual position is also in a defined target position window (position control error). See [more \[► 76\]](#)

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

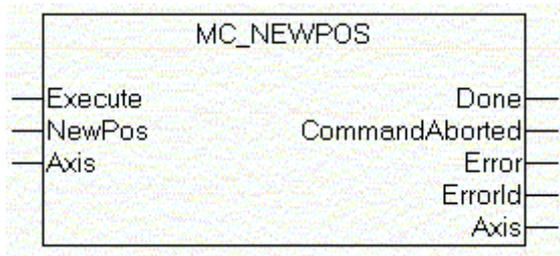
VAR_IN_OUT

```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development Environment	Target System	PLC libraries to include
TwinCAT v2.8 Build	PC (i386)	TcMC.Lib

3.7 MC_NewPos

A new target position for an axis in motion can be specified with the function block MC_NewPos.

VAR_INPUT

```
VAR_INPUT
  Execute : BOOL;
  NewPos  : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

NewPos : New target position. The new target position can be greater or smaller than the old target position.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

CommandAborted : Becomes TRUE if the task is interrupted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

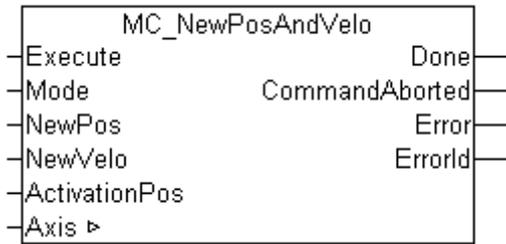
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.8 MC_NewPosAndVelo



A new target position with new velocity for an axis in motion can be specified with the function block MC_NewPosAndVelo.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    Mode         : E_CmdTypeNewTargPosAndVelo;
    NewPos       : LREAL;
    NewVelo      : LREAL;
    ActivationPos : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

Mode : Mode for the new target position/velocity

NewPos : New target position. The new target position can be greater or smaller than the old target position.

NewVelo : New velocity.

ActivationPos : Position at which the new position or velocity is activated for non-instantaneous take-over. When using the mode REACH_VELO_AT_POS, the new velocity will have been reached at this position. The change will then be initiated before.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

CommandAborted : Becomes TRUE if the task is interrupted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

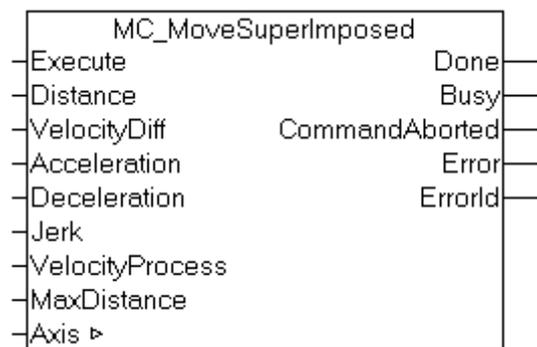
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.9 MC_MoveSuperImposed



Relative positioning is carried out with the function block MC_MoveSuperImposed.

MC_MoveSuperImposedExt [► 23] is an extended Version of this function block.

VAR_INPUT

```

VAR_INPUT
  Execute      : BOOL;
  Distance     : LREAL;
  VelocityDiff : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk         : LREAL;
  VelocityProcess : LREAL;
  MaxDistance  : LREAL;
END_VAR
  
```

Execute: The command is executed with rising edge.

Distance: Relative distance to catch up (<>0).

VelocityDiff: Maximum velocity increase during travel (absolute value > 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 : The jerk value is not used.

VelocityProcess: Mean process velocity during the constant velocity phase (absolute value > 0).

MaxDistance: Distance over which the catching up process may occur (absolute value > 0).

VAR_OUTPUT

```

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

Done: Becomes TRUE once the superposition move finished.

Busy: Becomes TRUE during catching up travel.

CommandAborted: Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: Supplies the error number when the Error output is set.

Note: The block returns error 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 error message within his application as a proper error or merely as a warning.

VAR_IN_OUT

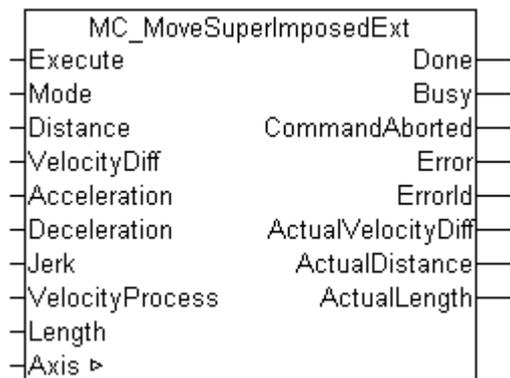
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis: Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.10 MC_MoveSuperImposedExt



Relative positioning is carried out with the function block MC_MoveSuperImposedExt.

The function block is an extended version of the [MC_MoveSuperImposed \[► 22\]](#) block.

[Application samples with MC MoveSuperimposed \[► 24\]](#).

VAR_INPUT

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

Execute: The command is executed with rising edge.

Mode: [Mode \[► 73\]](#) determines the type of the superimposed motion.

Distance: Relative distance to catch up (<>0).

VelocityDiff: Maximum velocity increase during travel (absolute value > 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 : The jerk value is not used.

VelocityProcess: Mean process velocity during the constant velocity phase (absolute value > 0).

Length: Distance over which the catching up process may occur (absolute value > 0). [Mode \[► 73\]](#) defines the interpretation of *Length*.

VAR_OUTPUT

```
VAR_OUTPUT
  Done           : BOOL;
  Busy           : BOOL;
  CommandAborted : BOOL;
  Error          : BOOL;
  ErrorId        : UDINT;
  ActualVelocityDiff : LREAL;
  ActualDistance  : LREAL;
  ActualLength    : LREAL;
END_VAR
```

Done: Becomes TRUE once the superposition move finished.

Busy: Becomes TRUE during catching up travel.

CommandAborted: Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: Supplies the error number when the Error output is set.

Note: The block returns error 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 error 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 block tries to reach the full *Distance* as specified. This distance may not be reached fully, depending on the parameterization (*VelocityDiff*, *Acceleration*, *Deceleration*, *Length*, *Mode*). In this case the maximum possible distance is superimposed. ($ActualDistance = Distance$).

ActualLength: Actual travel during superimposed motion ($ActualLength = Length$).

VAR_IN_OUT

```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

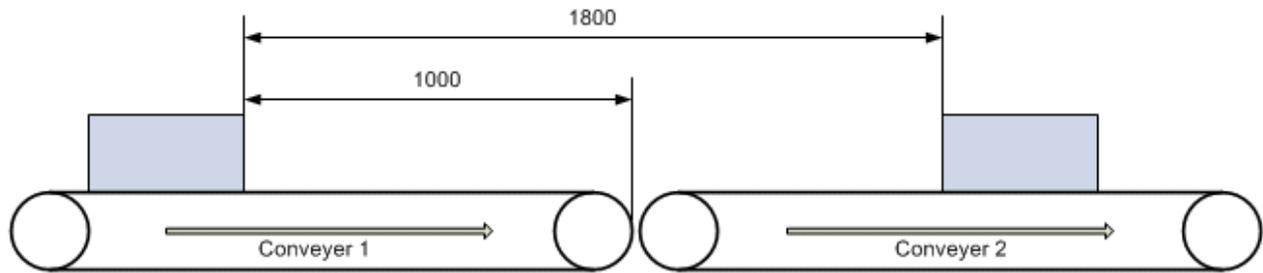
Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9 from build 1025	PC (i386)	TcMC.Lib

3.11 Application examples for MC_MoveSuperImposedExt

The function block [MC_MoveSuperImposedExt \[► 23\]](#) starts a higher-level motion 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_MoveSuperImposedExt](#) [▶ 23] 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

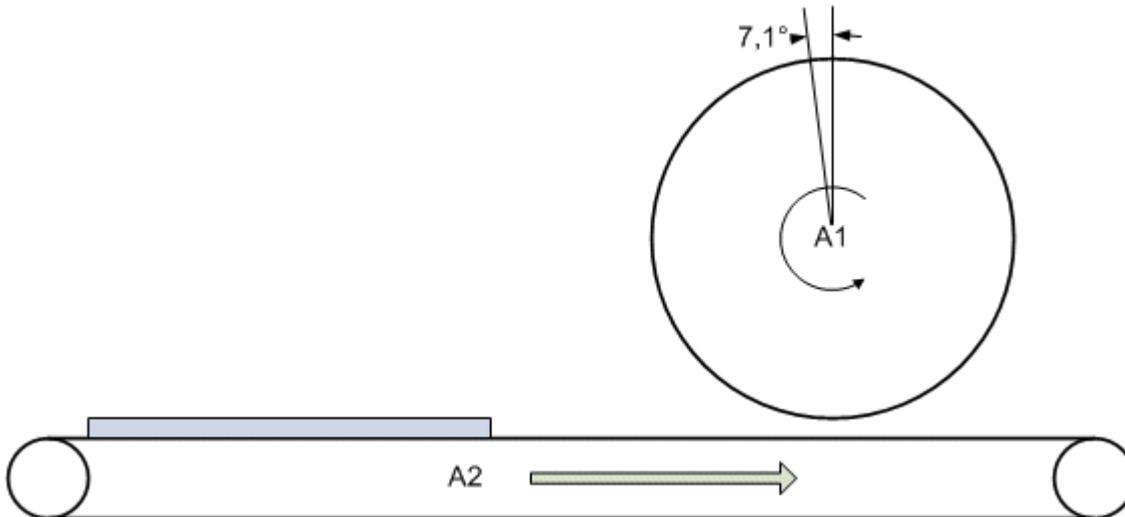
VelocityDiff = 500 mm/s

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 negative and the available correction distance *Length* is the distance between the right-hand package 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. Alternatively a correction distance can be defined within which the correction can occur, e.g. a complete roller revolution. This leads to the following possible parameterizations for function block [MC_MoveSuperImposedExt](#) [▶ 23]:

1. Fast correction:

Distance = 7.1°

Length = 360° (maximum possible correction distance)

Mode = SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION

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

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

Alternatively SUPERPOSITIONMODE_LENGTHREDUCTION_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.

2. Slow correction:

Distance = 7.1°

Length = 360° (correction distance)

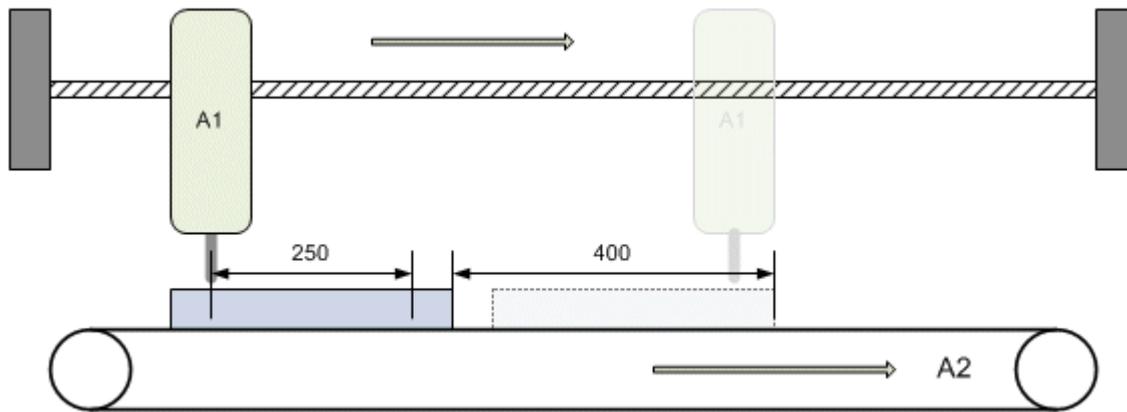
Mode = SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION

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

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

Drilling unit

A drilling unit should drill two holes in a moving workpiece. Synchronization for the first hole is assumed to be achieved via the flying saw (MC_GearInPos) and is not 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_MoveSuperImposedExt \[► 23\]](#):

1. 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 device can travel a larger distance since *Length* refers to the workpiece plus a relative change in position.

2. Slow correction:

Distance = 250 mm

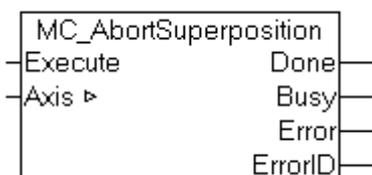
Length = 400 mm

Mode = SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION

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

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). 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*).

3.12 MC_AbortSuperposition



The MC_AbortSuperposition function block interrupts a overlaid motion started by MC_MoveSuperImposed [▶ 22] or MC_MoveSuperImposedExt [▶ 23] without stopping the basic motion.

A complete axis stop can be initiated with MC_Stop [▶ 32]. Calling MC_AbortSuperposition is not necessary in this case.

VAR_INPUT

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

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

VAR_OUTPUT

```
VAR_OUTPUT
  Done       : BOOL;
  Busy       : BOOL; (* function block is currently busy *)
  Error      : BOOL; (* Signals that an error has occurred within Function Block *)
  ErrorID    : UDINT; (* Error identification *)
END_VAR
```

Done : Becomes TRUE, as soon as the overlaid motion has been interrupted successfully.

Busy: Becomes TRUE as soon as the function block is active, and becomes FALSE when it has returned to its initial state.

Error : Becomes TRUE, as soon as an error occurs.

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

VAR_IN_OUT

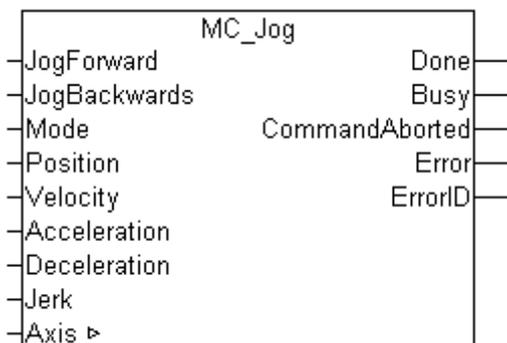
```
VAR_IN_OUT
  Axis      : NCTOPLC_AXLESTRUCT; (* Identifies the axis which position should be recorded at
a defined event at the trigger input *)
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
from TwinCAT v2.10	PC (i386)	TcMC.Lib

3.13 MC_Jog



The MC_Jog function block 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 mode.

VAR_INPUT

```
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 rising edge and the axis moved in positive direction of travel.

Depending on the mode the axis moves as long as the signal remains TRUE or stops automatically after a specified distance. During the motion no further signal edges are accepted (this includes the JogBackwards input). If signal edges occur simultaneously at the JogForward and JogBackwards inputs, JogForward has priority.

JogBackwards : The command is executed with rising edge and the axis moved in negative direction of travel.

JogForward and JogBackwards should be triggered alternatively, although they are also mutually locked internally.

Mode : The mode input specifies the mode for manual operation.

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 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 mode the position, velocity and dynamics data specified in the function block have no effect.

MC_JOGMODE_CONTINUOUS

The axis moves as long as the signal at one of the jog inputs is TRUE. The velocity and dynamics data specified by the user are used. The position has no effect.

MC_JOGMODE_INCHING

With rising edge at one of the jog inputs the axis is moved by a certain distance which is specified via the position input. The axis stop automatically, irrespective of the state of the jog inputs. A new movement step is only executed once a further rising edge is encountered. With each start the velocity and dynamics data specified by the user are used.

Position : Relative distance for movements in *MC_JOGMODE_INCHING* 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). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

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 do not have to be specified.

VAR_OUTPUT

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

Done : Becomes TRUE if a movement is completed successfully.

Busy : Becomes TRUE as soon as the function block is active, and becomes FALSE when it has returned to its initial state. Only then can a further edge be accepted at the jog inputs.

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be a [stop command](#) [► 32].

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

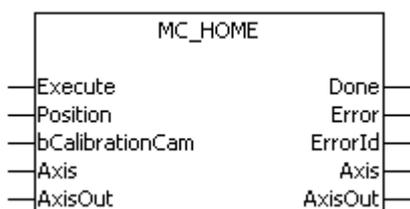
```
VAR_IN_OUT
  Axis           : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10 from build 1303	PC (i386)	TcMC.Lib

3.14 MC_Home



Calibration of the axis (referencing) is carried out with the function block *MC_Home*.

The reference mode is set in the TwinCAT SystemManager in the encoder *Incremental* dialog (also see Reference mode for incremental encoder).

VAR_INPUT

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

Execute : The command is executed with rising edge.

Position : Position when the reference cam is reached. The constant `DEFAULT_HOME_POSITION` can be used to make use of the *Calibration Value* which is defined in the TwinCAT SystemManger.

bCalibrationCam : Reference cam.

VAR_OUTPUT

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

Done : Becomes TRUE once referencing is complete.

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

```
VAR_IN_OUT
  Axis      : NCTOPLC_AXLESTRUCT;
  AxisOut   : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

AxisOut : Axis structure from PLC to NC.

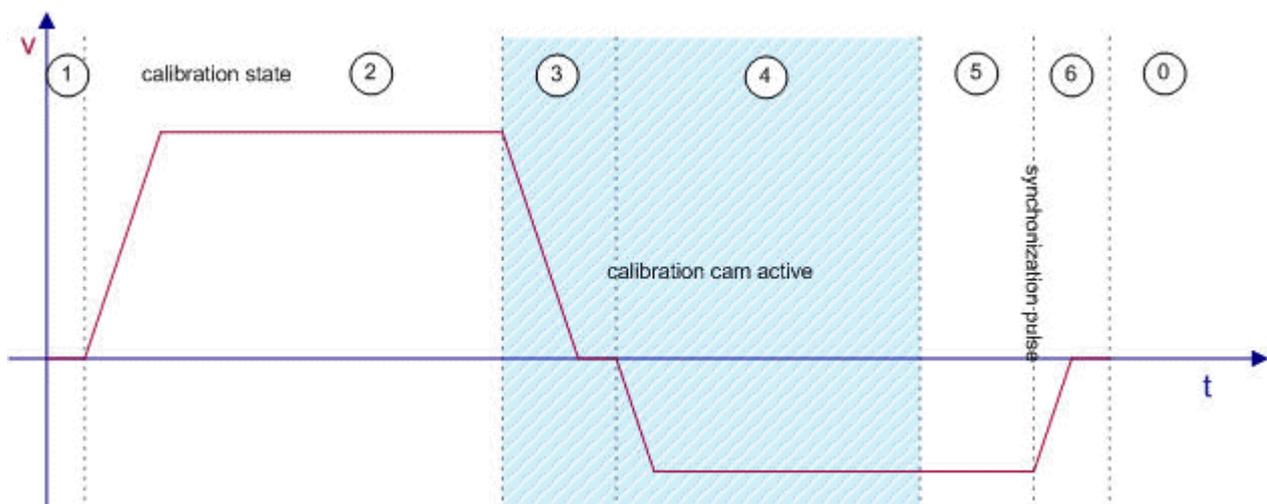
Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

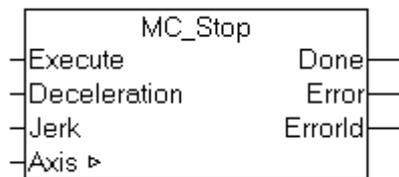
Note

The referencing process has several phases. The referencing state (calibration state) is signalled in the cyclic interface of the axis (`Axis.nCalibrationState`). The following diagram illustrates the individual process phases after starting of the `MC_Home` block.

If an axis is to be referenced without reference cam, i.e. only based on the sync pulse of the sensor, the reference cam can be simulated via the PLC program. The `bCalibrationCam` signal is initially activated and then cancelled, if `Axis.nCalibrationState` is equal or greater 4.



3.15 MC_Stop



An axis is stopped with the function block MC_Stop.

VAR_INPUT

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

Execute : The command is executed with rising edge.

Deceleration : Deceleration during brake travel.

If the value is 0, the deceleration from the move command is used. A non zero value will only be accepted if a *Jerk* value is passed as well.

Jerk : Jerk during brake travel.

If the value is 0, the jerk from the move command is used. A non zero value will only be accepted if a *Deceleration* value is passed as well.

Note:

The start command is used to specify the dynamic parameters (deceleration and jerk) for an axis. If no parameters are specified, the default values from the dynamics menu of the axis (TwinCAT System Manager) are used. For software versions up to TwinCAT version 2.10 Build 1242, MC_Stop can only be used to increase the deceleration dynamics, i.e. both deceleration and jerk must be equal or greater than the current dynamic values.

From TwinCAT 2.10 Build 1243 a new set value generator (7 phases optimized) was introduced that can be selected in the global axis parameters. This means that weaker deceleration dynamics can be specified with an MC_Stop command, in order to decelerate more slowly than originally specified. For safety reasons the following exceptions apply, in which case the weaker parameters are not used:

- If the target position would be exceeded
- If a stop with more rigorous dynamics had already been initiated
- If the axis is in an acceleration phase, and the velocity would exceed the parameterised travel speed due to weaker dynamic parameters (acceleration would reduce too slowly)
- If the axis is in a braking phase, and the direction of travel would reverse due to weaker dynamic parameters (negative acceleration cannot be reduced in time)

In all cases the axis is stopped safely and without error, even if weaker dynamic parameters are not accepted for the cases described above, i.e. if the current, more rigorous parameters are retained.

VAR_OUTPUT

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

Done : Becomes TRUE, once axis has finished its job and stands still.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

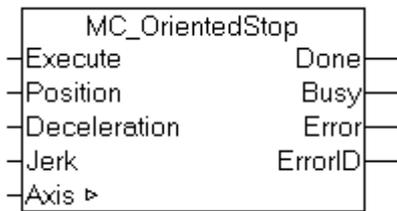
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.16 MC_OrientedStop



The function block MC_OrientedStop is used to affect an "oriented" axis stop at the specified modulo position. Depending on the speed and the dynamic parameters of the axis, it will stop at the next possible oriented position. The stopping distance compared to a standard stop may be up to one modulo period longer.

Note: The error output must be analysed, 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.

VAR_INPUT

```
VAR_INPUT
  Execute      : BOOL;
  Position     : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

Position: Absolute modulo destination position at which the axis should stop. The position must be in the range $[0 \leq \text{position} < \text{modulo period}]$. With a modulo period of 360 degrees, for example, 360 is therefore not a valid position. A value of 0 has to be specified instead.

Deceleration : Deceleration during brake travel. This input is currently not supported. The deceleration specified at the start of the axis is used for stopping.

Jerk: Jerk during brake travel. This input is currently not supported. The jerk specified at the start of the axis is used for stopping.

VAR_OUTPUT

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

Done : Becomes TRUE, once axis has finished its job and stands still.

Busy : Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Execute and becomes FALSE again after the block finished or aborted its operation. A rising edge at Execute will then be accepted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

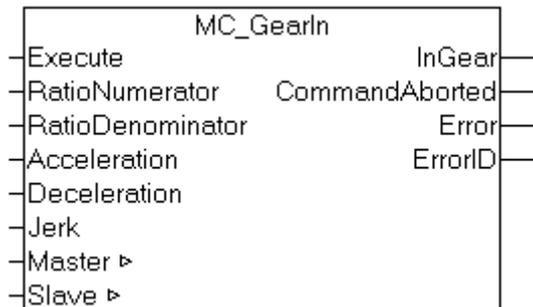
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9 from build 1003	PC (i386)	TcMC.Lib

3.17 MC_GearIn



The function block `MC_GearIn` activates a linear master-slave coupling (gear coupling). The block accepts a fixed gear ratio in a numerator denominator format. Alternatively the blocks `MC_GearInFloat` [▶ 35] with a LREAL gear ratio and `MC_GearInDyn` [▶ 36] with a dynamically changeable gear ratio are available.

Note:

The coupling can be activated only in standstill. The inputs `Acceleration`, `Deceleration` and `Jerk` are not used. Moving axes can be coupled using the optional *Flying Saw* functionality (`MC_GearInPos`, `MC_GearInVelo`).

VAR_INPUT

```
VAR_INPUT
  Execute       : BOOL;
  RatioNumerator : INT;
  RatioDenominator : UINT;
  Acceleration  : LREAL;
  Deceleration  : LREAL;
  Jerk          : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

RatioNumerator : Gearing factor numerator

RatioDenominator : Gearing factor denominator.

Acceleration : Acceleration (not used)

Deceleration : Deceleration (not used)

Jerk : Jerk (not used)

For a ratio of 1:3, the numerator has to be 1 and the denominator 3. The numerator may also be negative.

VAR_OUTPUT

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

InGear : Becomes TRUE, if the coupling was successful.

CommandAborted : Becomes TRUE, if the coupling could not be carried out.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

```
VAR_IN_OUT
  Master : NCTOPLC_AXLESTRUCT;
  Slave  : NCTOPLC_AXLESTRUCT;
END_VAR
```

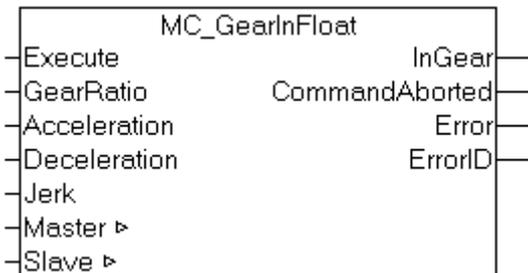
Master : Axis structure of the master.

Slave : Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.18 MC_GearInFloat



The function block MC_GearInFloat activates a linear master-slave coupling (gear coupling). The block accepts a fixed gear ratio in a LREAL format. Alternatively the blocks [MC_GearIn](#) [34] with a numerator-denominator gear ratio and [MC_GearInDyn](#) [36] with a dynamically changeable gear ratio are available.

Note:

The coupling can be activated only in standstill. The inputs Acceleration, Deceleration and Jerk are not used.

Moving axes can be coupled using the optional *Flying Saw* functionality (MC_GearInPos, MC_GearInVelo).

VAR_INPUT

```
VAR_INPUT
  Execute      : BOOL;
  GearRatio    : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk         : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

GearRatio : Gearing factor.

Acceleration : Acceleration (not used)

Deceleration : Deceleration (not used)

Jerk : Jerk (not used)

VAR_OUTPUT

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

InGear : Becomes TRUE, if the coupling was successful.

CommandAborted : Becomes TRUE, if the coupling could not be carried out.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

```
VAR_IN_OUT
  Master : NCTOPLC_AXLESTRUCT;
  Slave  : NCTOPLC_AXLESTRUCT;
END_VAR
```

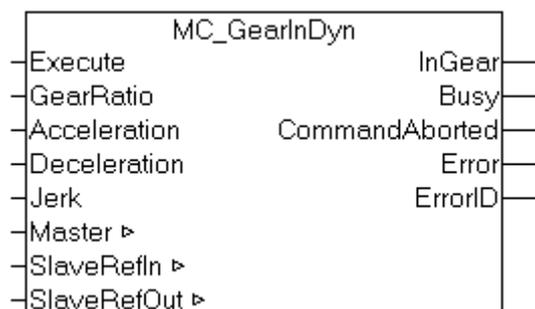
Master : Axis structure of the master.

Slave : Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.19 MC_GearInDyn



The function block `MC_GearIn` activates a linear master-slave coupling (gear coupling). The gear ratio can be modified in every PLC cycle. The block is therefore suitable for velocity controlled master slave couplings. The acceleration parameter limits the slaves acceleration in case of high gear ratio changes.

Note:

The coupling can be activated only in standstill. Moving axes can be coupled using the optional *Flying Saw* functionality (`MC_GearInPos`, `MC_GearInVelo`).

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    GearRatio    : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
END_VAR
```

Execute : The command is executed with rising edge. The GearRatio is accepted in every PLC cycle while Execute is TRUE. The command finishes its execution after a falling edge.

GearRatio : Gearing factor. The gear ratio may be changed in every PLC cycle.

Acceleration : Acceleration. The acceleration value limits the slaves acceleration in case of high gear ratio changes. The maximum acceleration will be reached when the master is moving at its maximum velocity. The slave acceleration may be lower if the master is moving at slower speed.

Deceleration : Deceleration. (not used)

Jerk : Jerk. (not used)

VAR_OUTPUT

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

InGear : Becomes TRUE, if the coupling was successful.

Busy : Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Execute and becomes FALSE again after the block finished or aborted its operation. A rising edge at Execute will then be accepted.

CommandAborted : Becomes TRUE, if the slave is decoupled while Execute is TRUE.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

```
VAR_IN_OUT
    Master : NCTOPLC_AXLESTRUCT;
    Slave  : NCTOPLC_AXLESTRUCT;
END_VAR
```

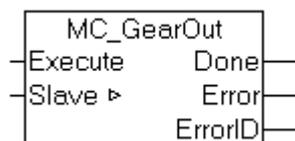
Master : Axis structure of the master.

Slave : Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8 B748 or higher	PC (i386)	TcMC.Lib
TwinCAT v2.9 B1000 or higher		

3.20 MC_GearOut



The function block MC_GearOut deactivates a linear master-slave coupling (gear coupling).

Note: If a slave axis is disconnected in motion, it is not stopped automatically, but it reaches a constant speed with which it continues to move endlessly. The axis can be stopped with a stop command. (see also [MC_GearOutExt \[► 39\]](#))

NOTICE

Migrating to TwinCAT 2.11

If the Set Point Generator Type is set to '*7 Phases (optimized)*', the slave axis will reduce its acceleration to zero after it is being decoupled and it will then continue moving endless at constant velocity. The decoupled axis will not be positioned to any target position.

The behavior is comparable to a move commanded by MC_MoveVelocity.

With TwinCAT 2.10 the set point generator type is selectable.

From TwinCAT 2.11 the setting is fixed to '*7 Phases (optimized)*'.

If a project is upgraded from TwinCAT 2.10 to TwinCAT 2.11, the behavior will be as described here. After updating an existing application to TwinCAT 2.11, it might be necessary to adapt the PLC program.

VAR_INPUT

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

Execute : The command is executed with rising edge.

VAR_OUTPUT

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

Done : Becomes TRUE, if decoupling was successful.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

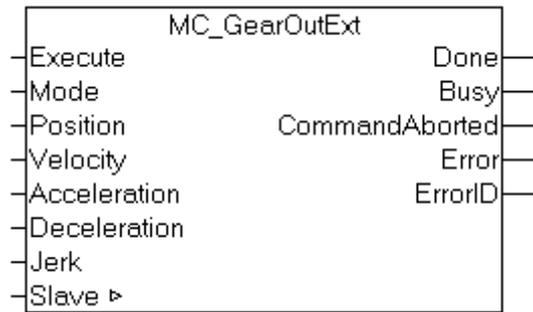
```
VAR_IN_OUT
  Slave : NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.21 MC_GearOutExt



NOTICE

From TwinCAT 2.11 onwards, the slave axis is moved without acceleration when it is uncoupled and continues to move at the constant speed that is set. There is no positioning around the master travel distance converted with the coupling factor, but a behavior as after a MC_MoveVelocity is established.

MC_GearOutExt is an extended universal decoupling command. With this command the slave axis is feed with a sequence command directly after decoupling. Therefore, a smooth transition from a coupled movement to a discrete movement is possible.

Remark: The sequence command can only be executed if the slave axis is decoupled in the movement. In the standstill, no sequence command is possible.

NOTICE

Migrating to TwinCAT 2.11

If the Set Point Generator Type is set to '7 Phases (optimized)', the slave axis will reduce its acceleration to zero after it is being decoupled and it will then continue moving endless at constant velocity. The decoupled axis will not be positioned to any target position. The behavior is comparable to a move commanded by MC_MoveVelocity. With TwinCAT 2.10 the set point generator type is selectable. From TwinCAT 2.11 the setting is fixed to '7 Phases (optimized)'. If a project is upgraded from TwinCAT 2.10 to TwinCAT 2.11, the behavior will be as described here. After updating an existing application to TwinCAT 2.11, it might be necessary to adapt the PLC program.

VAR_INPUT

```
VAR_INPUT
  Execute      : BOOL;
  Mode         : E_DecoupleMode;
  Position     : LREAL;
  Velocity     : LREAL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk        : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

Mode: describes the sequence command, that is executed directly after the decoupling of the slave axis.

```
TYPE E_DecoupleMode :
(
  MC_DECOUPLEMODE_STOP,           (* 0 Stop after decoupling *)
  MC_DECOUPLEMODE_ORIENTEDSTOP,  (* 1 Oriented stop after decoupling *)
  MC_DECOUPLEMODE_ENDLESS,       (* 2 endless motion at actual velocity after decoupling *)
  MC_DECOUPLEMODE_ENDLESS_NEWVELO, (* 3 endless motion at parameterized velocity after decoupling *)
*)
  MC_DECOUPLEMODE_NEWPOS,        (* 4 stop at parameterized position after decoupling *)
  MC_DECOUPLEMODE_NEWPOSANDVELO, (* 5 move at parameterized velocity after decoupling and stop
at parameterized position *)
  MC_DECOUPLEMODE_INSTANTANEOUSSTOP (* 6 instantaneous stop (jump to velocity zero) *)
);
END_TYPE
```

Position : Absolute target position to be used for positioning. If the mode *OrientedStop* is used, the *Position* is a modulo position.

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). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

VAR_OUTPUT

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

Done : Becomes TRUE, if decoupling was successful. And if according to the following command the target position or the constant velocity was reached.

Busy : Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Execute and becomes FALSE again after the block finished or aborted its operation. A rising edge at Execute will then be accepted.

CommandAborted : Becomes TRUE, if the command could not be fully executed. The reason may be an error, a stop or a superimposed travel command.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set

VAR_IN_OUT

```
VAR_IN_OUT
  Slave          : NCTOPLC_AXLESTRUCT;
END_VAR
```

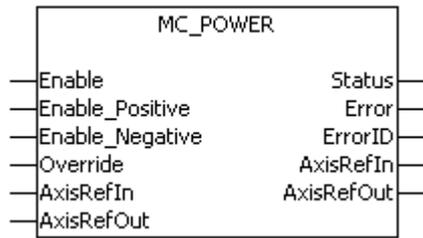
Slave : Axis structure of the slave.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10 Build 1313	PC (i386)	TcMC.Lib

4 Function blocks - administration

4.1 MC_Power



The enablings for an axis are set with the function block MC_Power.

VAR_INPUT

```
VAR_INPUT
  Enable       : BOOL;
  Enable_Positive : BOOL;
  Enable_Negative : BOOL;
  Override     : LREAL;
END_VAR
```

Enable : The command is executed as long as the input is TRUE.

Enable_Positive : .

Enable_Negative :

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

VAR_OUTPUT

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

Status : Becomes TRUE once all enablings were set successfully. This signal includes the drive's *ready* signal when available.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

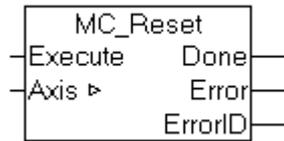
```
VAR_IN_OUT
  AxisRefIn  : NCTOPLC_AXLESTRUCT;
  AxisRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.2 MC_Reset



An axis reset is carried out with the function block MC_Reset.

VAR_INPUT

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

Execute : The command is executed with a rising edge.

VAR_OUTPUT

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

Done : Becomes TRUE if the reset command is executed without error feedback.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

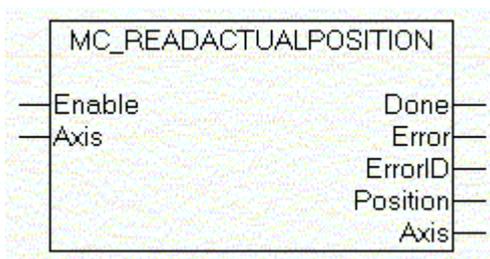
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.3 MC_ReadActualPosition



The actual axis position can be read with the function block MC_ActualPosition.

VAR_INPUT

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

Enable : The command is executed as long as the input is TRUE.

VAR_OUTPUT

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

Done : Becomes TRUE, if the states were determined successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

Position : Current axis position

VAR_IN_OUT

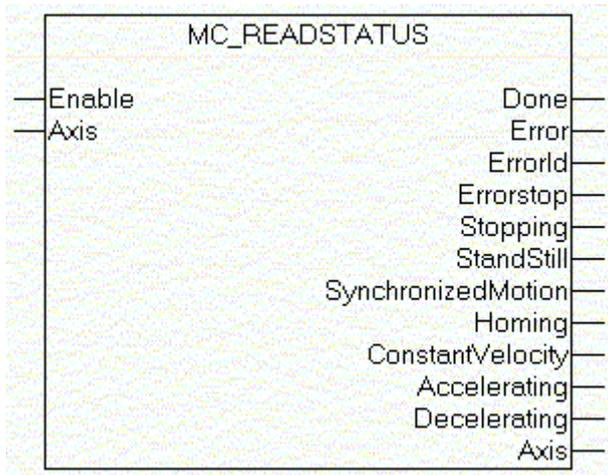
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.4 MC_ReadStatus



The states of an axis are displayed with the function block MC_ReadStatus.

VAR_INPUT

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

Enable : The command is executed as long as the input is TRUE.

VAR_OUTPUT

```
VAR_OUTPUT
  Done           : BOOL;
  Error          : BOOL;
  ErrorID        : UDINT;
  Errorstop      : BOOL;
  Stopping       : BOOL;
  StandStill     : BOOL;
  DiscreteMotion : BOOL;
```

```

ContinuousMotion : BOOL;
SynchronizedMotion : BOOL;
Homing : BOOL;
ConstantVelocity : BOOL;
Accelerating : BOOL;
Decelerating : BOOL;
END_VAR

```

- Done** : Becomes TRUE, if the states were determined successfully.
- Error**: Becomes TRUE, as soon as an error occurs.
- ErrorID** : Supplies the error number when the Error output is set.
- Errorstop** : Becomes TRUE as soon as an axis error occurs.
- Stopping** : Becomes TRUE as soon as the axis is in the braking phase.
- StandStill** : Becomes TRUE as soon as the axis is stationary.
- DiscreteMotion** : Becomes TRUE as soon as the axis is moved by a discrete motion.
- ContinuousMotion** : Becomes TRUE as soon as the axis is moved by a continuous motion.
- SynchronizedMotion** : Becomes TRUE as soon as the axis is coupled.
- Homing** : Becomes TRUE as soon as the axis is referenced.
- ConstantVelocity** : Becomes TRUE as soon as the axis travels with constant velocity.
- Acceleration** : Becomes TRUE as soon as the axis accelerates.
- Deceleration** : Becomes TRUE as soon as the axis decelerates.

VAR_IN_OUT

```

VAR_IN_OUT
Axis : NCTOPLC_AXLESTRUCT;
END_VAR

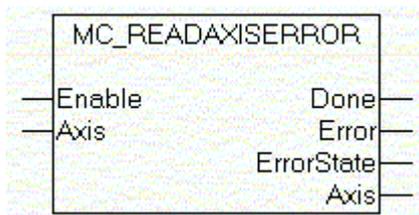
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.5 MC_ReadAxisError



The error code for an axis is displayed with the function block MC_ReadAxisError.

VAR_INPUT

```

VAR_INPUT
Enable : BOOL;
END_VAR

```

Enable : The command is executed as long as the input is TRUE.

VAR_OUTPUT

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

Done : Becomes TRUE, if the states were determined successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

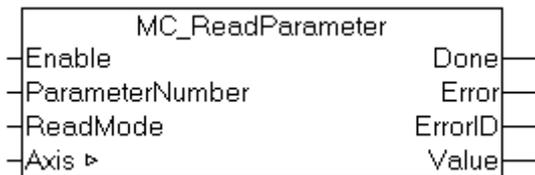
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.6 MC_ReadParameter



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

Note: "Axis" is referencing the TwinCAT NC axis parameters in this case, not the drive parameters!

VAR_INPUT

```
VAR_INPUT
  Enable      : BOOL;
  ParameterNumber : INT;
  ReadMode    : E_ReadMode;
END_VAR
```

Enable : The command is executed as long as the input is TRUE.

ParameterNumber : Number [▶ 70] of the parameter to be read.

ReadMode : ReadMode [▶ 73] of the parameter to be read (once or cyclic).

VAR_OUTPUT

```
VAR_OUTPUT
  Done      : BOOL;
  Error     : BOOL;
  ErrorID   : UDINT;
  Value     : LREAL;
END_VAR
```

Done : Becomes TRUE, if the states were determined successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

Value : Displays the read value.

VAR_IN_OUT

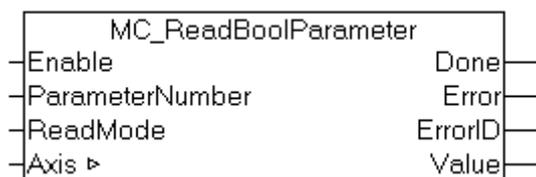
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.7 MC_ReadBoolParameter



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

Note: "Axis" is referencing the TwinCAT NC axis parameters in this case, not the drive parameters!

VAR_INPUT

```
VAR_INPUT
  Enable       : BOOL;
  ParameterNumber : INT;
  ReadMode     : E_ReadMode;
END_VAR
```

Enable : The command is executed as long as the input is TRUE.

ParameterNumber : Number [▶ 70] of the parameter to be read.

ReadMode : ReadMode [▶ 73] of the parameter to be read (once or cyclic).

VAR_OUTPUT

```
VAR_OUTPUT
  Done       : BOOL;
  Error      : BOOL;
  ErrorID    : UDINT;
  Value      : LREAL;
END_VAR
```

Done : Becomes TRUE, if the states were determined successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

Value : Displays the boolean value that was read.

VAR_IN_OUT

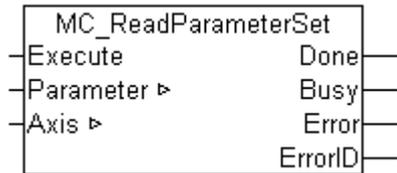
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.8 MC_ReadParameterSet



The function block MC_ReadParameterSet is used to read the complete axis parameter set.

Note: "Axis" is referencing the TwinCAT NC axis parameters in this case, not the drive parameters!

VAR_INPUT

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

Execute : The command is executed with rising edge.

VAR_OUTPUT

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

Done : Becomes TRUE, if the states were determined successfully.

Busy : Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Execute and becomes FALSE again after the block finished or aborted its operation. A rising edge at Execute will then be accepted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

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

Parameter : Complete axis parameter set of data type *ST_AxisParameterSet*.

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.9 MC_ReadAxisComponents



The function block MC_ReadAxisComponents is used to read information about the elements of an axis as encoder, drive and controller.

Note: "Axis" is referencing the TwinCAT NC axis parameters in this case, not the drive!

VAR_INPUT

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

Execute : The command is executed with rising edge.

VAR_OUTPUT

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

Done : Becomes TRUE, if the states were determined successfully.

Busy : Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Execute and becomes FALSE again after the block finished or aborted its operation. A rising edge at Execute will then be accepted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

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

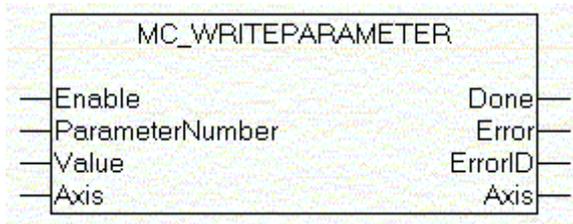
AxisComponents : axis parameter set of data type *ST_AxisComponents*.

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10 Build 1251	PC (i386)	TcMC.Lib

4.10 MC_WriteParameter



Axis parameters can be written with the function block MC_WriteParameter.

Note: "Axis" is referencing the TwinCAT NC axis parameters in this case, not the drive parameters!

VAR_INPUT

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

Enable : The command is executed as long as the input is TRUE.

ParameterNumber : Number [▶ 70] of the parameter to be read.

Value : This LREAL value is written.

VAR_OUTPUT

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

Done : Becomes TRUE, if the states were determined successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

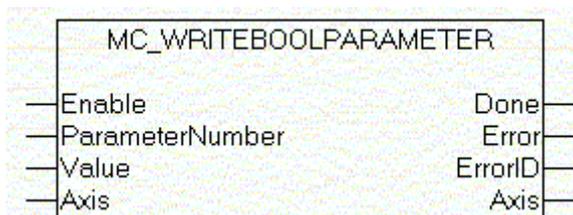
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.11 MC_WriteBoolParameter



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

Note: "Axis" is referencing the TwinCAT NC axis parameters in this case, not the drive parameters!

VAR_INPUT

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

Enable : The command is executed as long as the input is TRUE.

ParameterNumber : Number [▶ 70] of the parameter to be read.

Value : This BOOL value is written.

VAR_OUTPUT

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

Done : Becomes TRUE, if the states were determined successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

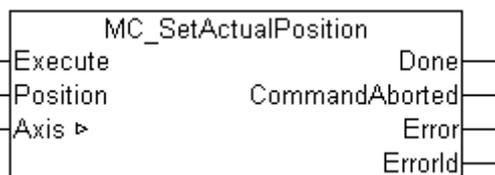
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.12 MC_SetActualPosition



The actual position of an axis can be set with the function block MC_SetActualPosition.

VAR_INPUT

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

Execute : The command is executed with rising edge.

Position : Actual position to be set.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

CommandAborted : Becomes TRUE if the task is interrupted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

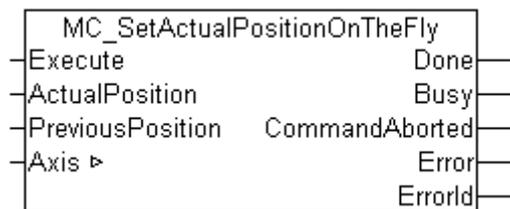
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.13 MC_SetActualPositionOnTheFly



The actual position of an axis can be set with the function block MC_SetActualPositionOnTheFly while the axis is moving. The active travel command is adapted in order to prevent the original target being overrun.

This block is used for special reference travels, for example. A precisely measured position is detected via a hardware latch, for example. The latch supplies the current axis position at this point. Without stopping the axis, the axis position can be set to this point at a later stage by specifying the latch position as the previous reference position (*PreviousPosition*), in addition to the required position (*ActualPosition*)

Note:

Pay attention to the fact that the current set position and the target position of the current move will be shifted by the same position difference ($ActualPosition - PreviousPosition$). The move will cover exactly the same distance but due to the shifted coordinate system the final position will differ from the commanded target position.

VAR_INPUT

```
VAR_INPUT
  Execute          : BOOL;
  ActualPosition   : LREAL;
  PreviousPosition : LREAL;
NEND_VAR
```

Execute : The command is executed with rising edge.

ActualPosition : Actual position to be set.

PreviousPosition : Previous position that corresponds with the new actual position.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

Busy : Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Execute and becomes FALSE again after the block finished or aborted its operation. A rising edge at Execute will then be accepted.

CommandAborted : Becomes TRUE if the task is interrupted.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

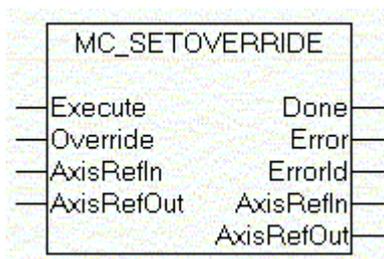
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.10 Build 1251	PC (i386)	TcMC.Lib

4.14 MC_SetOverride



The override for an axis can be specified with the function block MC_SetOverride.

VAR_INPUT

```
VAR_INPUT
  Execute : BOOL;
  Override : LREAL;
END_VAR
```

Execute : The command is executed with rising edge.

Override: New override value.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

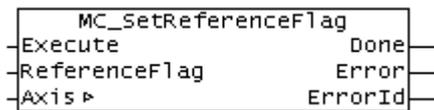
```
VAR_IN_OUT
  AxisRefIn : NCTOPLC_AXLESTRUCT;
  AxisRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.15 MC_SetReferenceFlag



The MC_SetReferenceFlag function block changes the current *axis is homed* state of an axis. This function block is required when not the MC_Home function block but any other mechanisms are used to home the axis (e.g. move to a mechanical stop). In this case the home position can be set using the [MC_SetActualPosition \[► 50\]](#) function block.

VAR_INPUT

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

Execute : The command is executed with rising edge.

ReferenceFlag : New referenced state of the axis. TRUE = axis is homed. FALSE = axis is not homed

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

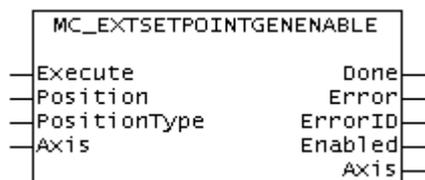
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.16 MC_ExtSetPointGenEnable



The external set value generator of an axis can be switched on with the function block MC_ExtSetPointGenEnable. Subsequently, the axis takes over the set value specifications from its cyclic axis interface (fExtSetPos, fExtSetVelo, fExtSetAcc and nExtSetDirection)

VAR_INPUT

```

VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    PositionType : E_TargPosType;
END_VAR

```

Execute : The command is executed with rising 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 set value generator is responsible. Setting of this position activates target position monitoring, and the flag In target position becomes TRUE, as soon as this position is reached.

PositionType : Position type

VAR_OUTPUT

```

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

```

Done : Becomes TRUE, if the command was issued successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

Enabled : Enabled shows the current state of the external set value generator, independent of the function execution.

VAR_IN_OUT

```

VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR

```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.17 MC_ExtSetPointGenDisable



The external set value generator of an axis can be switched off with the function block MC_ExtSetPointGenDisable. Subsequently, the axis no longer takes over the set value specifications from its cyclic axis interface (fExtSetPos, fExtSetVelo, fExtSetAcc and nExtSetDirection)

VAR_INPUT

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

Execute : The command is executed with rising edge.

VAR_OUTPUT

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

Done : Becomes TRUE, if the command was issued successfully.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

Enabled : Enabled shows the current state of the external set value generator, independent of the function execution.

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

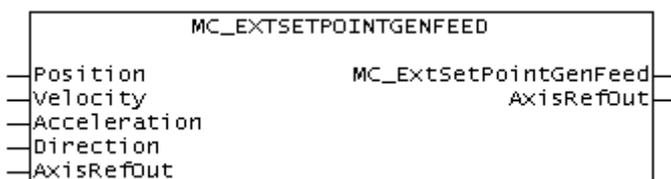
Axis : Axis structure.

Example

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.18 MC_ExtSetPointGenFeed



With the function MC_ExtSetPointGenFeed, the set values of an external set value generator are fed into an axis. The function copies the data instantan into the cyclical axis interface (fExtSetPos, fExtSetVelo, fExtSetAcc und nExtSetDirection) of the axis. The function result MC_ExtSetPointGenFeed is unused and therefore always FALSE.

VAR_INPUT

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

Position : Set position from an external set value generator

Velocity : Set velocity from an external set value generator

Acceleration : Set acceleration from an external set value generator

Direction : Set direction from an external set value generator. (-1 = negative direction, 0 = standstill, 1 = positive direction)

VAR_IN_OUT

```
VAR_IN_OUT
    AxisRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.19 MC_OverrideFilter

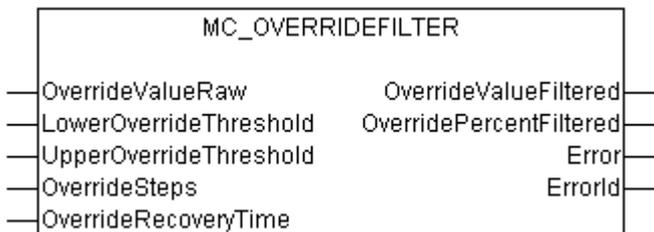


Fig. 2: MC_OverrideFilter

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

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

VAR_INPUT

```
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: The lower threshold for the raw override value.

UpperOverrideThreshold: The upper threshold for the raw override value.

OverrideSteps: The specified steps (override resolution).

OverrideRecoveryTime: Minimum recovery time, after which a new filtered override value is placed on the output. The override values 0% and 100% are implemented without delay.

VAR_OUTPUT

```

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

```

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

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

Error: Becomes TRUE, as soon as an error occurs.

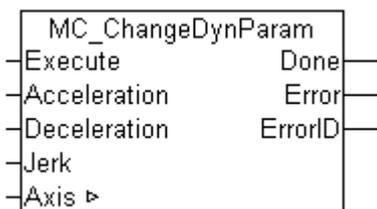
ErrorID : If the error output is set, this parameter supplies the error number.

Possible error number	Possible causes
MC_ERROR_PARAMETER_NOT_CORRECT	<ul style="list-style-type: none"> • OverrideSteps <= 1 • LowerOverrideThreshold >= UpperOverrideThreshold

Requirements

Development Environment	Target System	PLC Libraries to include
TwinCAT v2.8 Build > 739	PC (i386)	TcMC.Lib

4.20 MC_ChangeDynParam



MC_ChangeDynParam changes the acceleration and deceleration parameters for an active positioning process.

Requirements

The current dynamics can be modified if the set value generator type of the axis is set to "7 phases (optimized)" (global axis parameters). Otherwise only very limited modification is possible.

Restrictions

In some situations no new dynamic parameters can be accepted for safety reasons, and the current dynamic parameters are retained:

- If the target position would be exceeded (error 16#427F)
- If a stop with more rigorous dynamics had already been initiated (error 16#425D)
- If the axis is in a acceleration phase, and the velocity would exceed the parameterised travel speed due to weaker dynamic parameters (acceleration would reduce too slowly) (error 16#423A)
- If the axis is in a braking phase, and the direction of travel would reverse due to weaker dynamic parameters (negative acceleration cannot be reduced in time) (error 16#4289)
- Special case: internal optimisation is not possible (error 16#427E)

VAR_INPUT

```
VAR_INPUT
  Execute      : BOOL;
  Acceleration : LREAL;
  Deceleration : LREAL;
  Jerk         : LREAL;
END_VAR
```

Execute : The command is executed with rising edge. The block can be re-triggered during the movement.

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 acceleration from the axis configuration in the System Manager is used.

Jerk : Jerk (=0). If the value is 0, the standard jerk from the axis configuration in the System Manager is used.

Note: The Jerk can be changed if the set point generator type "7 Phases (optimized)" is configured. (from TwinCAT 2.10 Build 1251 or TwinCAT 2.9 Build 1032)

VAR_OUTPUT

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

Done : Becomes TRUE when the axis is at its destination and all monitoring activities (such as destination position monitoring) are positive.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID : Supplies the error number when the Error output is set.

VAR_IN_OUT

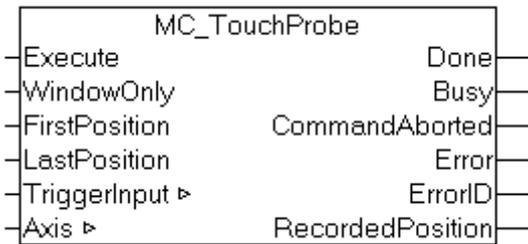
```
VAR_IN_OUT
  Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8 Build	PC (i386)	TcMC.Lib

4.21 MC_TouchProbe



The MC_TouchProbe function block records an axis position at the point in 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 thus very accurate and independent of cycle time. The function block controls this mechanism and determines the externally recorded position.

Requirements

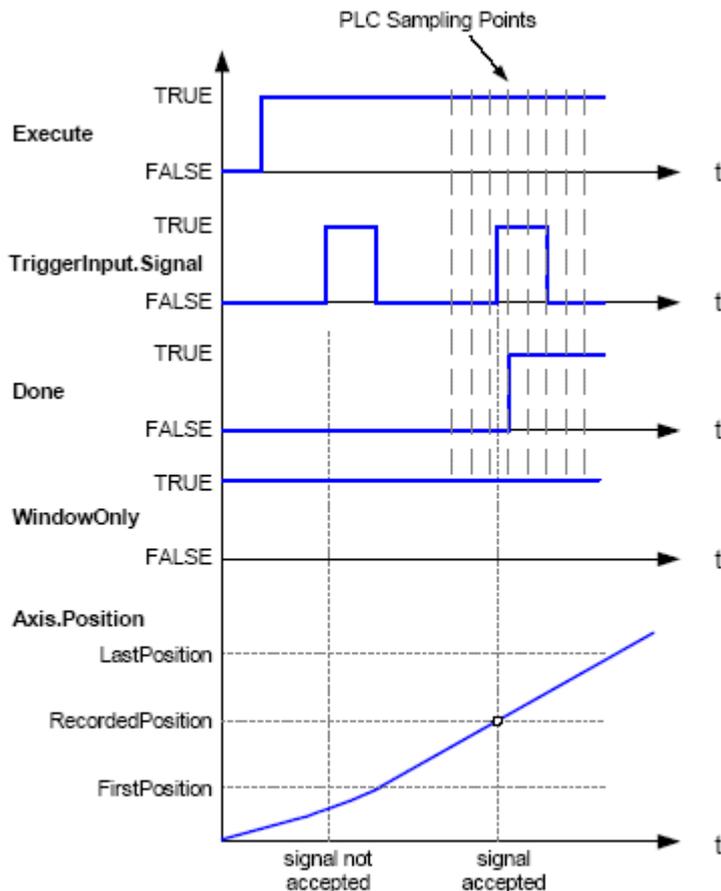
The prerequisite for the position acquisition is suitable encoder hardware that is able to latch the recorded position. The following equipment is supported, for example: SERCOS drives, the Beckhoff AX2000 with SERCOS and Lightbus interfaces and the Beckhoff KL5101 Encoder Bus Terminal. The digital trigger 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 recording is possible. On this subject read *Measuring probe evaluation with AX2xxx-B200 (Lightbus)* and *Measuring probe evaluation with AX2xxx-B750 (SERCOS)*.

Note

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 process is to be interrupted at an intermediate point in time, the function block [MC_AbortTrigger \[▸ 62\]](#) with the same [TriggerInput \[▸ 61\]](#) data structure must be called up. Otherwise no new cycle can be initiated.

Signal curve



Timing example TouchProbe

VAR_INPUT

```

VAR_INPUT
  Execute       : BOOL; (* Starts touch probe recording *)
  WindowOnly   : BOOL; (* Enables the monitoring window *)
  FirstPosition : LREAL; (* Beginning of the monitoring window *)
  LastPosition  : LREAL; (* Ending of the monitoring window *)
END_VAR

```

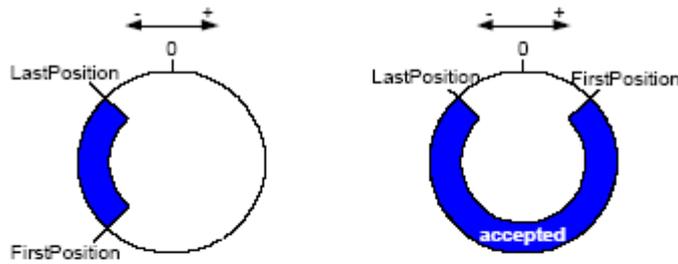
Execute : The command is executed with the rising edge and the external position latch is activated.

WindowOnly : If this option is active, only one position inside 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 does *Done* become TRUE. The recording window can be interpreted in terms of absolute or modulo values. In this connection the flag [ModuloPositions \[► 71\]](#) in the structure [TriggerInput \[► 71\]](#) is to be set accordingly. In the case of absolute value positions there is exactly one window. In the case of modulo value positions the window repeats itself within the modulo cycle defined in the axis parameters (e.g. 0 to 360 degrees).

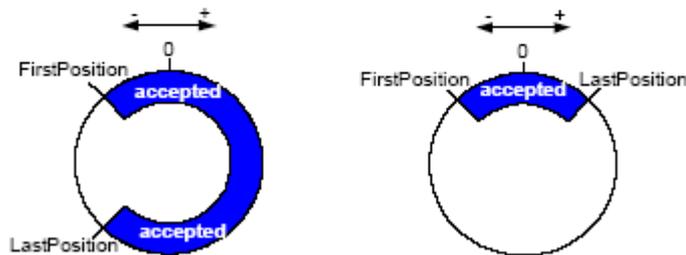
FirstPosition : Initial position of the recording window, if *WindowOnly* is TRUE. This position can be interpreted as an absolute or modulo value. In this connection the flag [ModuloPositions \[► 71\]](#) is to be set appropriately in the structure *TriggerInput* (see below).

LastPosition : Final position of the recording window, if *WindowOnly* is TRUE. This position can be interpreted as an absolute or modulo value. In this connection the flag [ModuloPositions \[► 71\]](#) is to be set appropriately in the structure *TriggerInput* (see below).

A. FirstPosition < LastPosition



B. FirstPosition > LastPosition



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

VAR_OUTPUT

```
VAR_OUTPUT
Done          : BOOL; (* move completed *)
Busy          : BOOL; (* function block is currently busy *)
CommandAborted : BOOL;
Error         : BOOL; (* Signals that an error has occurred within Function Block *)
ErrorID       : UDINT; (* Error identification *)
RecordedPosition : LREAL; (* Position where the trigger event occurred *)
END_VAR
```

Done : Becomes TRUE, if an axis position has been recorded successfully. The position is sent to the output *RecordedPosition*.

Busy: Becomes TRUE as soon as the function block is active, and becomes FALSE when it has returned to its initial state.

CommandAborted : Becomes TRUE if the process is interrupted by an external event, e.g. by the call up of *MC_AbortTrigger* [▶ 62].

Error : Becomes TRUE, as soon as 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

VAR_IN_OUT

```
VAR_IN_OUT
TriggerInput : MC_InputRef; (* Reference to the trigger signal source. *)
Axis         : NCTOPLC_AXLESTRUCT; (* Identifies the axis which position should be recorded at a defined event at the trigger input *)
END_VAR
```

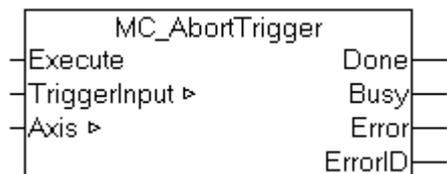
TriggerInput : Data structure of type *MC_InputRef* [▶ 71], that describes the trigger input for the recording of the axis position.

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
from TwinCAT v2.9 Build 1000	PC (i386)	TcMC.Lib

4.22 MC_AbortTrigger



The MC_AbortTrigger function block interrupts a measuring probe cycle initiated by [MC_TouchProbe \[► 59\]](#). MC_TouchProbe initiates a measuring probe cycle by activating a position latch in external encoder hardware. If the process is to be terminated before the trigger signal has activated the position latch, MC_AbortTrigger can be used for this purpose. If the measuring probe cycle has completed successfully, it is not necessary to call up this function block.

VAR_INPUT

```
VAR_INPUT
  Execute : BOOL; (* Starts touch probe recording *)
END_VAR
```

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

VAR_OUTPUT

```
VAR_OUTPUT
  Done : BOOL; (* move completed *)
  Busy : BOOL; (* function block is currently busy *)
  Error : BOOL; (* Signals that an error has occurred within Function Block *)
  ErrorID : UDINT; (* Error identification *)
END_VAR
```

Done : Becomes TRUE, as soon as the measuring probe cycle has been interrupted successfully.

Busy : Becomes TRUE as soon as the function block is active, and becomes FALSE when it has returned to its initial state.

Error : Becomes TRUE, as soon as an error occurs.

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

VAR_IN_OUT

```
VAR_IN_OUT
  TriggerInput : MC_InputRef; (* Reference to the trigger signal source. *)
  Axis : NCTOPLC_AXLESTRUCT; (* Identifies the axis which position should be recorded at
  a defined event at the trigger input *)
END_VAR
```

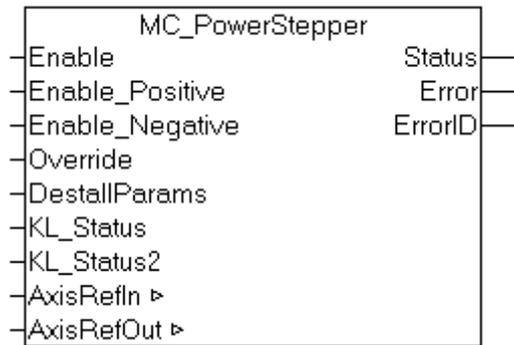
TriggerInput : Data structure of type [MC_InputRef \[► 71\]](#), that describes the trigger input for the recording of the axis position. Here the same data structure must be used as in the corresponding [MC_TouchProbe \[► 59\]](#) function block.

Axis : Axis structure.

Requirements

Development environment	Target system type	PLC libraries to be linked
from TwinCAT v2.9 Build 1000	PC (i386)	TcMC.Lib

4.23 MC_PowerStepper



The enables for an axis are set with the function block MC_PowerStepper. An MC_Power block 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.

There is more detailed explanation in the [Appendix \[▶ 64\]](#).

VAR_INPUT

```
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 advance movement enable in positive direction.

Enable_Negative: NC advance movement enable in negative direction.

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

DestallParams: The functions of the block are enabled here, and their [working rules \[▶ 73\]](#) are specified.

KL_Status: The status byte of a terminal of type KL2531 or KL2541.

KL_Status2: The status word of a terminal of type KL2531 or KL2541.

VAR_OUTPUT

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

Status: Becomes TRUE once all enables were set successfully.

Error: Becomes TRUE, as soon as an error occurs.

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

VAR_IN_OUT

```
VAR_IN_OUT
  AxisRefIn  : NCTOPLC_AXLESTRUCT;
  AxisRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

AxisRefIn: The axis structure provided by the NC.

AxisRefOut: The axis structure provided by the PLC.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9, build 1026 onwards	PC (i386)	TcMC.Lib

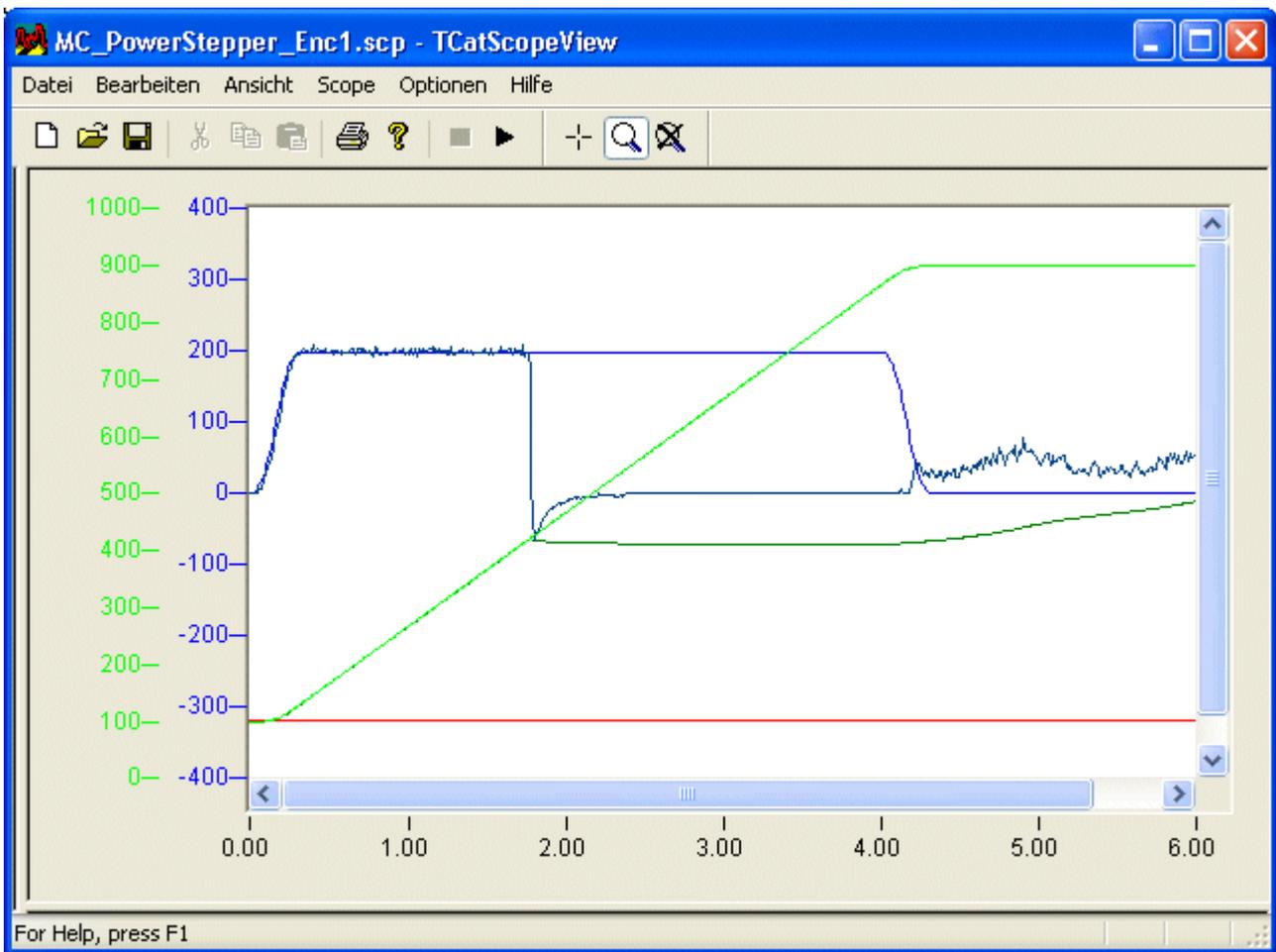
4.23.1 Notes on the MC_PowerStepper

The enables and the override for an axis are set with the [MC_PowerStepper \[► 63\]](#) function block. An MC_Power block 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 makes it possible to save considerable costs. There is, however, a possibility that some external force will push the motor beyond the position where it is able to generate the maximum torque. Because the electrically generated magnetic field does not take this into account, the restoring torque generated will fall as the excursion increases. 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 conditions that now apply, the motor may now latch into the new position (which means that a complete step has been lost), or the whole process may be repeated again here. The latter case is referred to as stalling, and is most likely to occur when current is fed to the motor at the typical frequency of the active drive operation.

Example 1: A stepper motor fitted with an encoder is operated with the NC PTP using the parameters typical for servos.

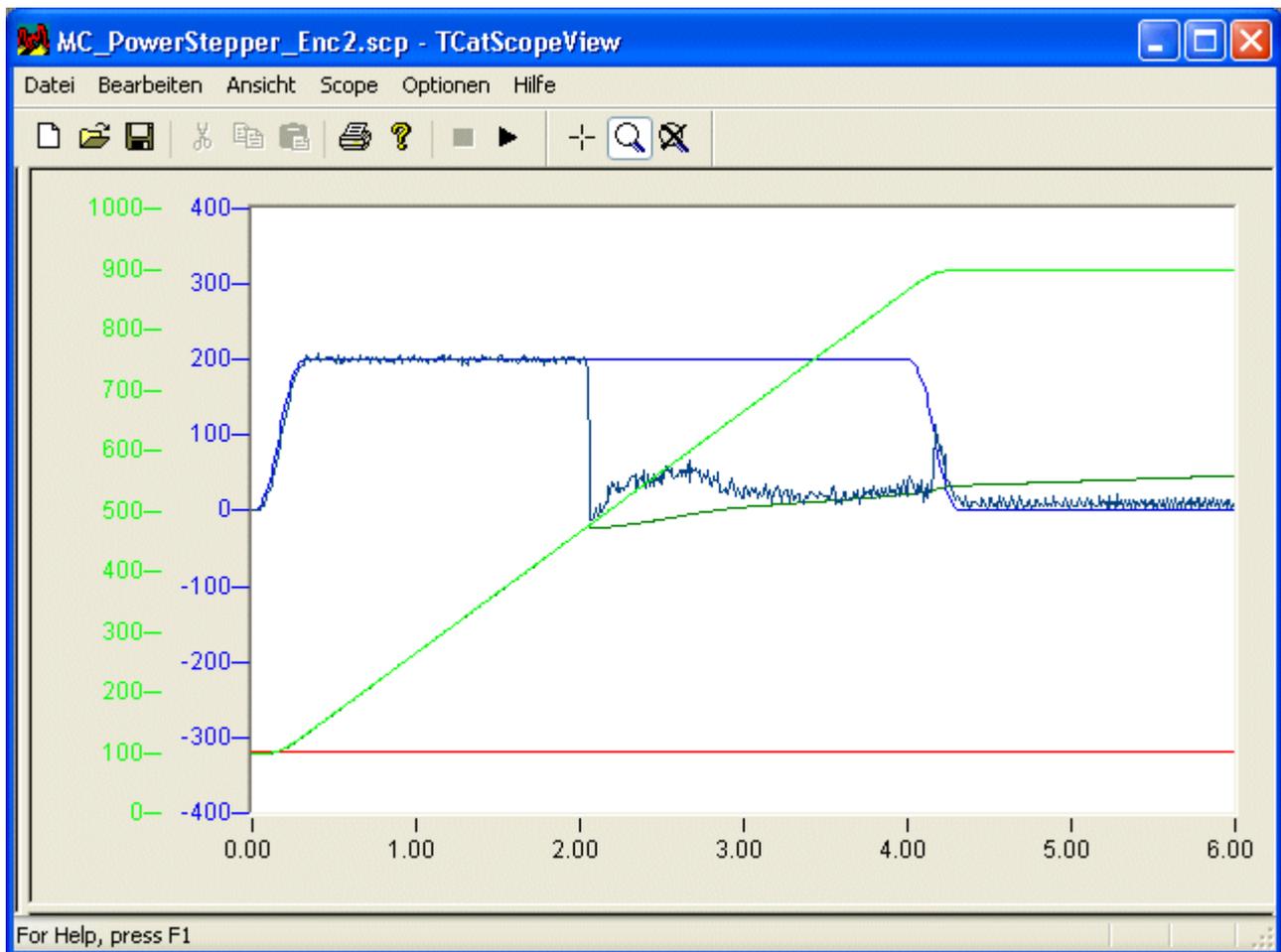


- bStalled
- SetVelo
- ActualPos
- ActualVelo
- SetPos

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 drive destination does the total of the set and correction velocities fall. In this example, the motor moves in an irregular manner. Even a small load torque will, however, prevent this. The only solution here would be to issue an [MC_Reset \[▶ 42\]](#) and to allow an appropriate settling time to pass. The axis would then have to be restarted by the application. A variety of state bits in the axis interface would react here. This must be appropriately considered in the application, as otherwise incorrect reactions may occur in the machine control process.

First corrective step: controller limitation

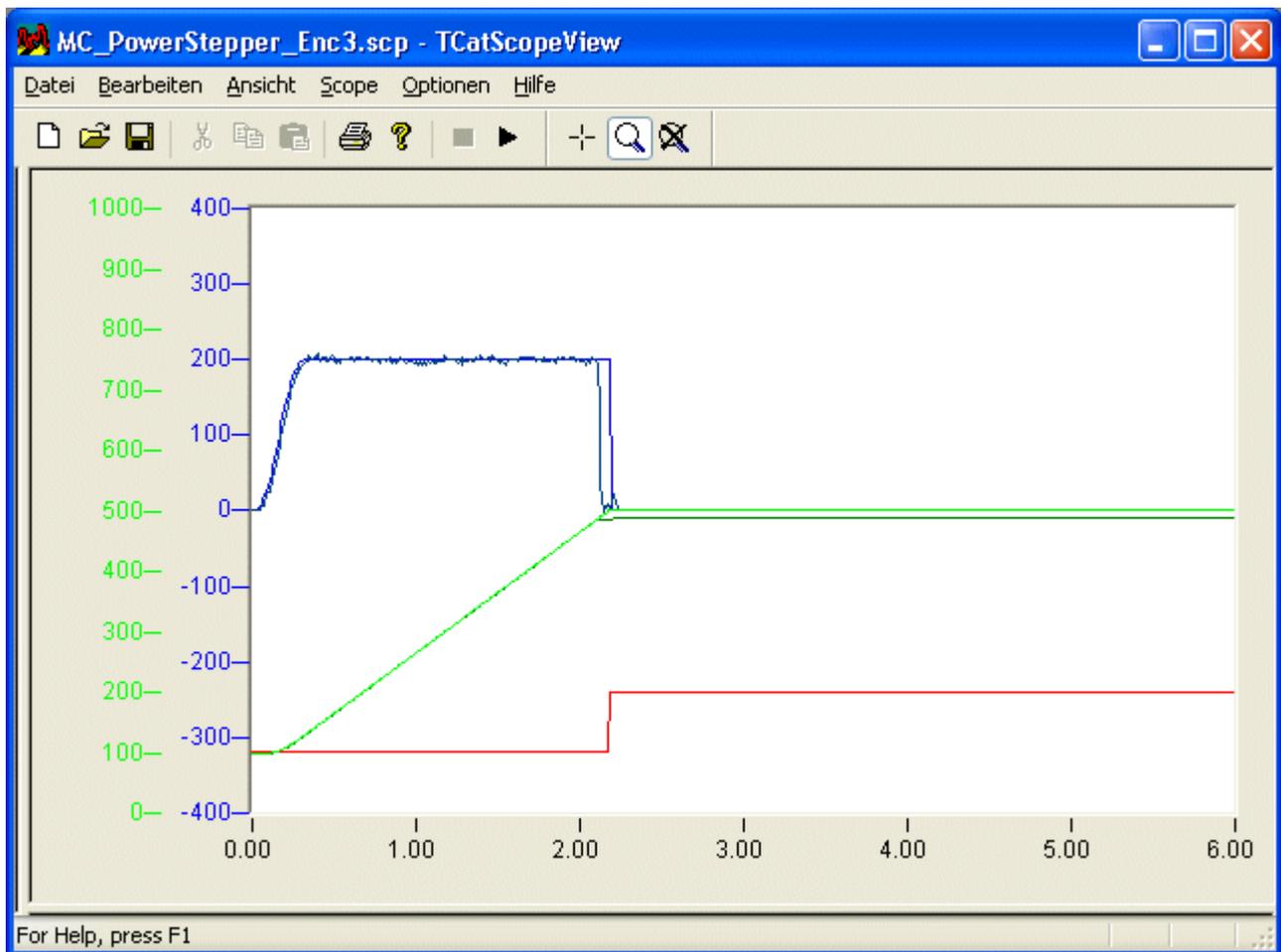
If, in the situation described above, the output of the position controller is limited to a sufficiently small value such as, for instance, 2%, the following pattern results.



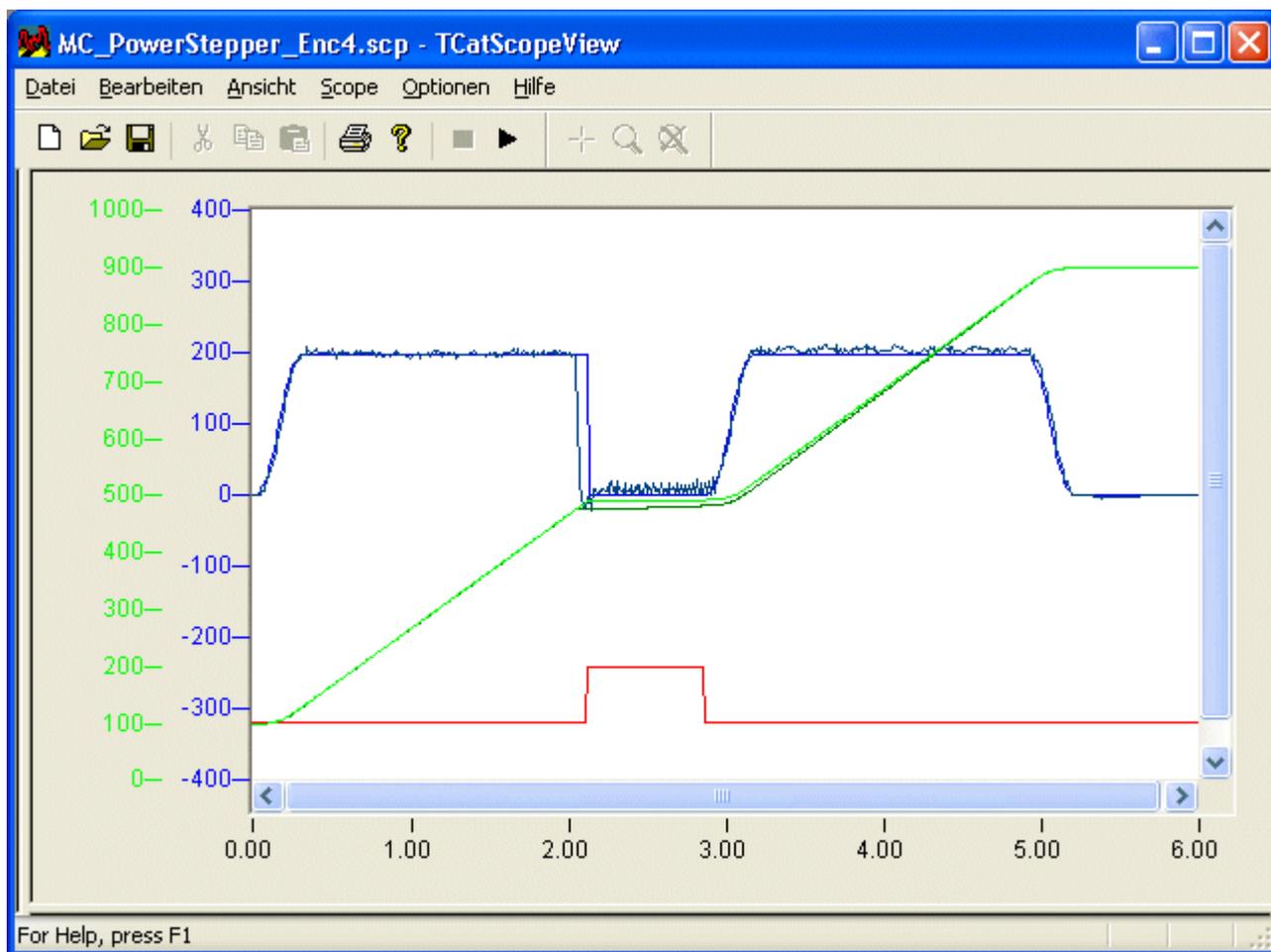
Here again, for the remaining period of profile generation, the set speed is too high for the stepper motor to be able to follow the set movement properly. When the end of the set profile has been reached, the stepper motor is now brought to its destination by the position controller, at a working frequency that it is able to follow without the ramp. It generates a very high torque as it does this. The time required for this corrective measure is, however, very long.

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 block is used. It has a parameter structure of type `ST_PowerStepperStruct [▶ 73]`, in which `PwStDetectMode_Lagging` is entered as the `DestallDetectMode`. The block uses the following error of the axis as the basis of its decision, making use of the threshold value and the filter time from the NC axis data for the following error monitoring that is to be deactivated here. In this example, `PwStMode_SetError` is entered as the `DestallMode`. Initially, the only difference from the following error alarm is the different error code.



If PwStMode_UseOverride is entered as the DestallMode, the MC_PowerStepper block uses the override to halt 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 following 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 significantly greater proportion of the overall profile is travelled at the specified speed, and the destination 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_SetErrNon-Ref	PwStMode_UseOverride
PwStDetectMode_Encoderless	Comment 1	Comment 2	not suitable
PwStDetectMode_Lagging	Comment 3	not useful	Comment 4

Comment 1: Useful for axes without encoder that are not referenced.

Comment 2: Useful for axes without encoder that are referenced with the aid of the terminal's pulse counter and, for instance, an external sensor.

Comment 3: The resultant behaviour largely corresponds to following error monitoring.

Comment 4: Useful for axes with encoder.

5 Functions

5.1 F_GetVersionTcMC



The function returns library version info.

FUNCTION F_GetVersionTcMC : UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT v2.8.0	PC (i386)	TcMC.Lib

6 Data types

6.1 MC_AxisPara

```

TYPE MC_AxisPara : (
(* PLCopen specific parameters *)
  CommandedPosition := 1,      (* lreal *)
  SWLimitPos,                (* lreal *)
  SWLimitNeg,                (* lreal *)
  EnableLimitPos,           (* bool *)
  EnableLimitNeg,           (* bool *)
  EnablePosLagMonitoring,   (* bool *)
  MaxPositionLag,           (* lreal *)
  MaxVelocitySystem,        (* lreal *)
  MaxVelocityAppl,          (* lreal *)
  ActualVelocity,           (* lreal *)
  CommandedVelocity,        (* lreal *)
  MaxAccelerationSystem,    (* lreal *)
  MaxAccelerationAppl,     (* lreal *)
  MaxDecelerationSystem,   (* lreal *)
  MaxDecelerationAppl,     (* lreal *)
  MaxJerk,                   (* lreal *)

(* Beckhoff specific parameters *)
  AxisId :=1000,              (* lreal *)
  AxisVeloManSlow,           (* lreal *)
  AxisVeloManFast,          (* lreal *)
  AxisVeloMax,               (* lreal *)
  AxisAcc,                   (* lreal *)
  AxisDec,                   (* lreal *)
  AxisJerk,                  (* lreal *)
  AxisMaxVelocity,          (* lreal *)
  AxisRapidTraverseVelocity, (* lreal *)
  AxisManualVelocityFast,   (* lreal *)
  AxisManualVelocitySlow,   (* lreal *)
  AxisCalibrationVelocityForward, (* lreal *)
  AxisCalibrationVelocityBackward, (* lreal *)
  AxisJogIncrementForward,  (* lreal *)
  AxisJogIncrementBackward, (* lreal *)
  AxisEnMinSoftPosLimit,   (* bool *)
  AxisMinSoftPosLimit,     (* lreal *)
  AxisEnMaxSoftPosLimit,   (* bool *)
  AxisMaxSoftPosLimit,     (* lreal *)
  AxisEnPositionLagMonitoring, (* bool *)
  AxisMaxPosLagValue,      (* lreal *)
  AxisMaxPosLagFilterTime, (* lreal *)
  AxisEnPositionRangeMonitoring, (* bool *)
  AxisPositionRangeWindow, (* lreal *)
  AxisEnTargetPositionMonitoring, (* bool *)
  AxisTargetPositionWindow, (* lreal *)
  AxisTargetPositionMonitoringTime, (* lreal *)
  AxisEnInTargetTimeout,   (* bool *)
  AxisInTargetTimeout,     (* lreal *)
  AxisEnMotionMonitoring,  (* bool *)
  AxisMotionMonitoringWindow, (* lreal *)
  AxisMotionMonitoringTime, (* lreal *)
  AxisDelayTimeVeloPosition, (* lreal *)
  AxisEnLoopingDistance,   (* bool *)
  AxisLoopingDistance,     (* lreal *)
  AxisEnBacklashCompensation, (* bool *)
  AxisBacklash,            (* lreal *)
  AxisEnDataPersistence,   (* bool *)
  AxisRefVeloOnRefOutput,  (* lreal *)
  AxisOverrideType,        (* lreal *)
  AxisEncoderScalingFactor, (* lreal *)
  AxisEncoderOffset,       (* lreal *)
  AxisEncoderDirectionInverse, (* bool *)
  AxisEncoderEncoderMask,  (* dword *)
  AxisEncoderModuloValue,  (* lreal *)
  AxisModuloToleranceWindow, (* lreal *)
  AxisEnActPosCorrection,  (* bool *)
  AxisActPosCorrectionFilterTime, (* lreal *)
  AxisUnitInterpretation,  (* lreal - Bit 2 of this word is the modulo flag *)

(* Beckhoff specific axis status information - READ ONLY *)
  AxisTargetPosition := 2000, (* lreal *)
  AxisRemainingTimeToGo,      (* lreal *)

```

```
AxisRemainingDistanceToGo      (* lreal *)
);
END_TYPE
```

In this enum, the parameters for the blocks MC_ReadParameter, MC_ReadBoolParameter, MC_WriteParameter and MC_WriteBoolParameter are specified. If the data types do not coincide, or if an attempt is made to read a non-existing parameter, an error is generated.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

6.2 MC_Direction

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

In this enum, the directions for the MC_MoveVelocity and MC_MoveModulo blocks are specified.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.8	PC (i386)	TcMC.Lib

6.3 MC_TouchProbe data structures

MC_InputRef

MC_InputRef

```
TYPE MC_InputRef :
STRUCT
  EncoderID      : UDINT;          (* 1..255 *)
  TouchProbe     : E_TouchProbe;  (* Signal source *)
  Edge           : E_SignalEdge;  (* rising or falling signal edge *)
  PlcEvent       : BOOL;          (* PLC trigger signal input when TouchProbe signal source
is set to 'PlcEvent' *)
  ProbeState     : E_TouchProbeState; (* internal state of the touch probe sequence *)
  ModuloPositions : BOOL;          (* interpretation of FirstPosition, LastPosition and
RecordedPosition as modulo positions when TRUE *)
END_STRUCT
END_TYPE
```

EncoderID : Encoder identification of the drive hardware used. The encoder ID can be read off in the TwinCAT SystemManager.

TouchProbe : Defines the signal source [▶ 72], within the encoder hardware used, which triggers the measuring probe function.

Edge : Defines whether the rising or falling edge [▶ 72] of the trigger signal is evaluated.

PlcEvent : If the signal source TouchProbe [▶ 71] is set to the type PlcEvent [▶ 71], a rising edge on these variables triggers the recording of the current axis position. The PlcEvent is not a true latch function, but is cycle-time dependent.

ProbeState : Internal state [▶ 72] of the measuring probe function. This variable must not be written to.

ModuloPositions : If the variable *ModuloPositions* is FALSE, the axis position is interpreted in an absolute linear range from $-\infty$ to $+\infty$. The positions *FirstPosition*, *LastPosition* und *RecordedPosition* of the [MC_TouchProbe](#) [[▶ 59](#)] function block 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.

E_TouchProbe

E_TouchProbe

The data type *E_TouchProbe* describes which signal from a signal source is used for the measuring probe function.

```
TYPE E_TouchProbe :
(
    TouchProbe1      := 1,
    TouchProbe2,
    TouchProbe3,
    TouchProbe4,
    PlcEvent         := 10
);
END_TYPE
```

TouchProbe1..TouchProbe4 : Signal source internal to the encoder hardware (e.g. drive AX2000, KL5001). Depending upon the hardware up to four trigger signals can be evaluated, but at the present time only the first signal, namely *TouchProbe1*, is supported.

PlcEvent : *PlcEvent* defines a trigger signal that is not switched using encoder hardware, but is generated directly in the PLC, and is thus a normal BOOL variable. This PLC trigger signal is stored in the data structure [MC_InputRef](#) [[▶ 71](#)] in the variable *PlcEvent* [[▶ 71](#)]. This leads to the axis position that is current at the time being recorded. The *PlcEvent* is not a true latch function, but is cycle-time dependent.

E_SignalEdge

E_SignalEdge

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

Rising or falling edge of the trigger signal.

E_TouchProbeState

E_TouchProbeState

```
TYPE E_TouchProbeState :
(
    TouchProbeInactive,
    TouchProbeActivated,
    TouchProbeAborted
);
END_TYPE
```

Internal state of the measuring probe function.

Requirements

Development environment	Target system type	PLC libraries to be linked
from TwinCAT v2.9 Build 1000	PC (i386)	TcMC.Lib

6.4 E_ReadMode

```

TYPE E_ReadMode :
(
    ReadMode_Once      := 1,
    ReadMode_Cyclic
);
END_TYPE
    
```

In this enum, the read modes for the blocks MC_ReadParameter and MC_ReadBoolParameter are specified.

Requirements

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

6.5 E_SuperpositionMode

```

TYPE E_SuperpositionMode :
(
    SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION:= 1,
    SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION,
    SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION,
    SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION
);
END_TYPE
    
```

The E_SuperpositionMode structure determines how a superimposed motion is carried out with the function block [MC_MoveSuperImposedExt](#) [▶ 23].

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. The *Length* is based on a reference axis without superimposed motion (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 change in velocity *VelocityDiff* or the maximum *Length* are exceeded. The *Length* is based on a reference axis without superimposed motion (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 change in velocity *VelocityDiff* or 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*.

Requirements

Development Environment	Target system	PLC Libraries to include
TwinCAT v2.9	PC (i386)	TcMC.Lib

6.6 ST_PowerStepperStruct

```

TYPE ST_PowerStepperStruct :
STRUCT
    DestallDetectMode : E_DestallDetectMode;
    DestallMode       : E_DestallMode;
    DestallEnable     : BOOL;
END_STRUCT
    
```

```
StatusMonEnable : BOOL;
END_STRUCT
END_TYPE
```

This structure specifies the parameters for the [MC_PowerStepper \[► 63\]](#) block.

DestallDetectMode: A coded [value \[► 74\]](#) is used here to specify how a stall situation is to be detected.

DestallMode: A coded [value \[► 74\]](#) is used here to specify how a stall situation that has been detected is to be handled.

DestallEnable: This parameter specifies whether stall detection and handling in accordance with the rules above is active.

StatusMonEnable: This parameter specifies whether the status bits of the I/O terminal are monitored and whether errors that occur are to be reported.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9, build 1026 onwards	PC (i386)	TcMC.Lib

6.7 E_DestallMode

```
TYPE E_DestallMode :
(
  PwStMode_None := 0,
  PwStMode_SetError,
  PwStMode_SetErrNonRef,
  PwStMode_UseOverride
);
END_TYPE
```

This enum specifies the way the [MC_PowerStepper \[► 63\]](#) block works.

PwStMode_None: Nothing is done about a stall situation that occurs.

PwStMode_SetError: If a stall situation occurs, the axis adopts the error state with code 16#4636. A message in plain text, such as "**MC_PowerStepper(AxId:=3) reporting error 0x4636 (stall error)!**" is entered into the log book, and appears in the Logger view if the System Manager is running.

PwStMode_SetErrNonRef: If a stall situation occurs, the axis is placed into an error state, as under the **PwStMode_SetError** setting, and a plain text message is issued. The axis is, moreover, placed into the non-referenced state.

PwStMode_UseOverride: If a stall situation occurs, the profile generator for the axis is stopped as quickly as possible by setting the override to zero. Once the position controller has brought the axis back again to the current set position, the override of the profile generator is set to one so that it will then execute the remaining distance.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9, build 1026 onwards	PC (i386)	TcMC.Lib

6.8 E_DestallDetectMode

```
TYPE E_DestallDetectMode :
(
  PwStDetectMode_None := 0,
  PwStDetectMode_Encoderless,
  PwStDetectMode_Lagging
);
END_TYPE
```

In this enum, the methods of stall detection are specified for the [MC_PowerStepper \[► 63\]](#) block.

PwStDetectMode_None: Stall detection is not carried out.

PwStDetectMode_Encoderless: Bits 1 to 3 of the status byte of a type KL2531 or KL2541 terminal are evaluated for the purposes of stall detection. The byte is to be passed to the [MC_PowerStepper](#) block as the

KL_Status.

PwStDetectMode_Lagging: The following error of the axis is evaluated for stall detection. The decision is made with the aid of the following error threshold and the following error filter time from the NC parameters.

Requirements

Development environment	Target system type	PLC libraries to be linked
TwinCAT v2.9, build 1026 onwards	PC (i386)	TcMC.Lib

7 Appendix

7.1 MC_Move.....Done

The successful completion of an axis move command is indicated by the “Done” signal of the corresponding function block. There are three conditions checked to indicate the successful completion of a move command. The first thing verified in order to set MC_Move...Done to TRUE is the flag AxisHasJob. If the option *Position Range Monitoring* AND/OR the option *Target Position Monitoring* in Global section of corresponding axis parameter is set to TRUE then the flags AxisInPositionWindow and AxisIsAtTargetPosition are checked respectively for their TRUE status, provided AxisHasJob has had a transition from TRUE to FALSE.

In short, the possible combinations checked for a successful completion of a move command are

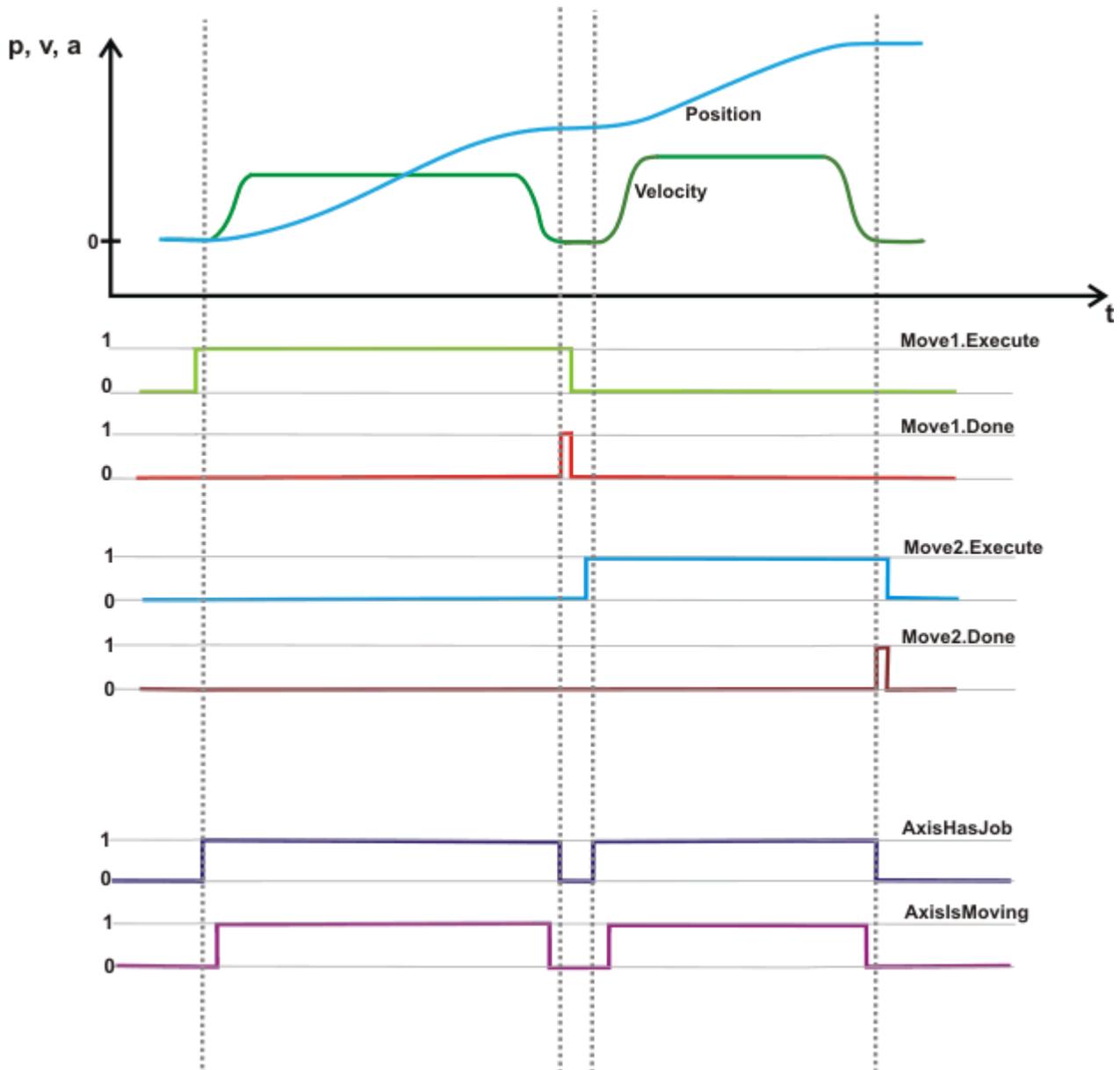
1. If *Position Range Monitoring* = TRUE and *Target Position Monitoring* = TRUE, then the condition checked for is FALSE status of AxisHasJob AND TRUE status of AxisIsAtTargetPosition (implicit checking for TRUE status of AxisInPositionWindow)
2. If *Position Range Monitoring* = TRUE and *Target Position Monitoring* = FALSE, then the condition checked for is FALSE status of AxisHasJob AND TRUE status of AxisInPositionWindow
3. If *Position Range Monitoring* = FALSE and *Target Position Monitoring* = FALSE, then the condition checked for is FALSE status of AxisHasJob

NOTE:

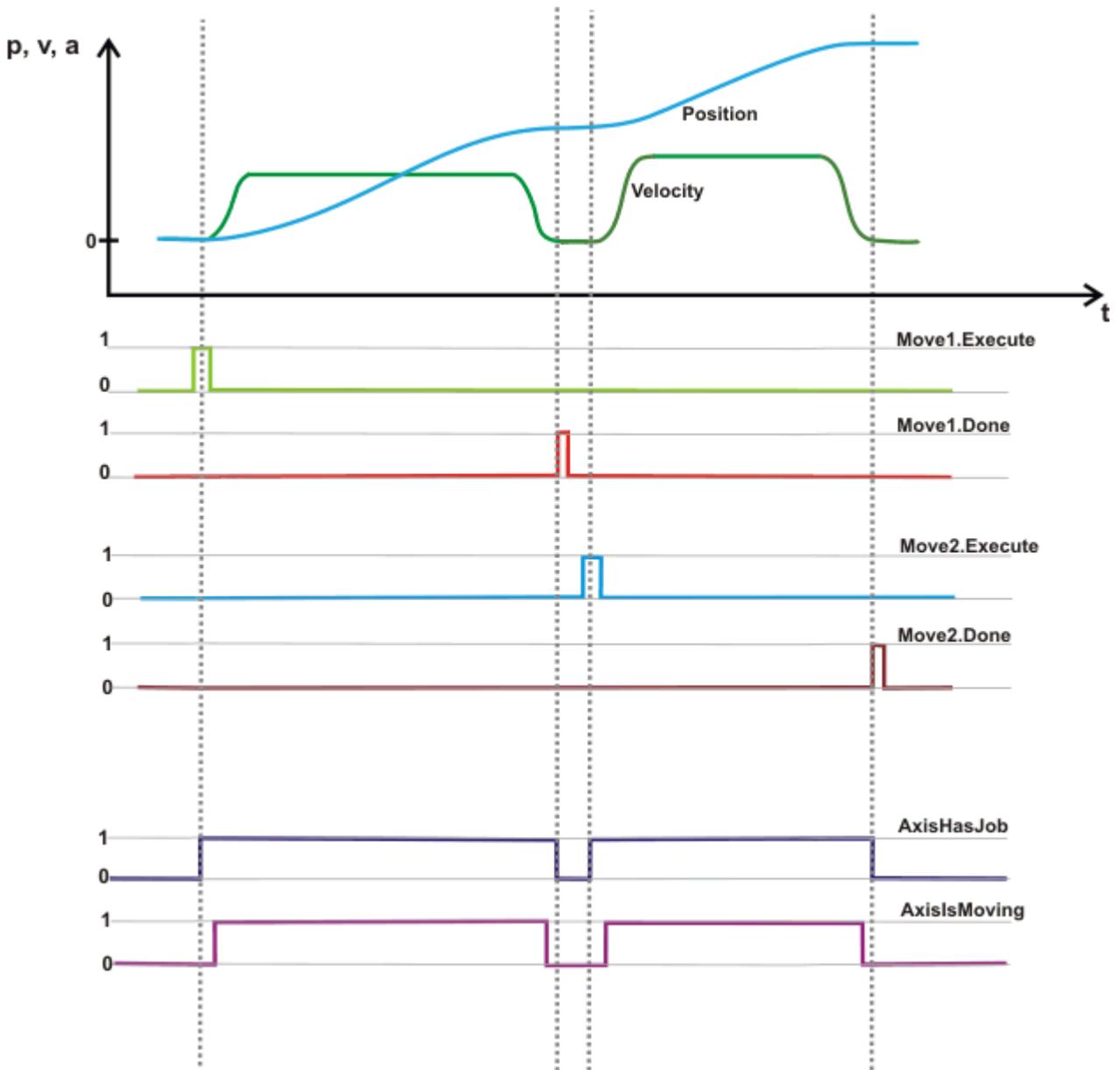
Position Range Monitoring and *Target Position Monitoring* are options in TwinCAT System Manager in the Global section of the corresponding axis parameters.

The figures below depict the behaviour of MC_Move...Done signal based on different types of MC_Move...Execute signal. The sequence designed here consists of two consecutive move commands.

In the following figure, MC_Move...Execute is set TRUE throughout the positioning period of the axis and is set to FALSE only when MC_Move...Done signal is seen to be TRUE. The MC_Move...Done remains TRUE until MC_Move...Execute is TRUE indicating that the command has been successfully completed.



In the following figure the MC_Move...Execute remains TRUE only for first few cycles after the positioning command for an axis is issued. The MC_Move...Done remains TRUE for one cycle indicating that the command has been successfully completed.



Trademark statements

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

More Information:
www.beckhoff.com/tx1200

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

