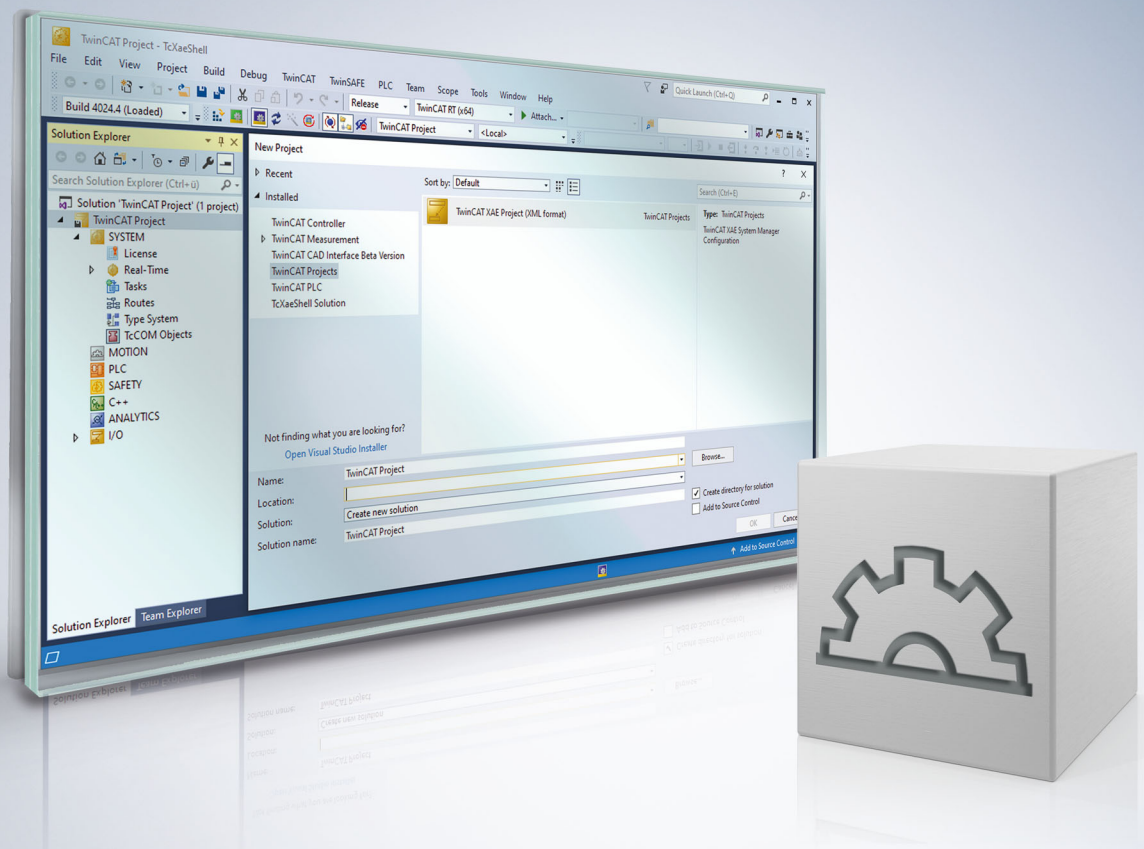
BECKHOFF New Automation Technology

Manual | EN

TF5410

TwinCAT 3 | Motion Collision Avoidance



2024-11-06 | Version: 2.6.0

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS®, and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

The EtherCAT Technology is covered by the following patent applications and patents, without this constituting an exhaustive list:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

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

Copyright

Third-party brands

Third-party trademarks and wordmarks are used in this documentation. The trademark endorsements can be found at: 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 Introduction

TwinCAT 3 Motion Collision Avoidance is an optional package for collision avoidance when operating multiple axes with TwinCAT 3 NC PTP in linear and/or translational dependency. The underlying algorithm ensures a minimum distance to the predecessor axis. This means that active collision avoidance can be implemented with TwinCAT 3 Motion Collision Avoidance if several motors share a rail, for example. In addition to active collision avoidance, the TF5410 can also be used to accumulate axes in a controlled manner, for example in linear movements such as the XTS (eXtended Transport System).

The programming of the positioning commands from the PLC is done via the library Tc3_McCollisionAvoidance [▶ 33], which is based on the library Tc2_MC2 and has been extended by the "Gap" input. With TwinCAT 3 Motion Collision Avoidance, for example, all axes can be started to the same target position. The algorithm then ensures that only the first axis moves to the target position. The remaining axes automatically maintain their minimum distance and line up. In this way, dynamic buffers can be created to accumulate products without further programming.
The administrative function blocks are contained in the library Tc3_McCoordinatedMotion [▶ 56].

In addition, TF5410 contains a geo-compensation for the XTS, with which the reference point of the path dynamics can be shifted from the XTS motor path to the center of mass of the tool/product on the XTS mover. In curve segments, the two path dynamics differ, so that unexpected forces can act without geo-compensation. The library Tc3_McCompensations [▶ 52] is available in the PLC for this purpose.

Installation

The TF5410 TwinCAT 3 Motion Collision Avoidance software package is installed together with the TF5400 software package.

Target System

Windows XP or Windows 7/8/10 platform level 40 or higher

Additional licensing requirements

TF5410 TwinCAT 3 Motion Collision Avoidance requires the TC1250 license.

3 Overview of the new functions

As of TF5400 V3.3.19

- New: CA group [▶ 20] parameter Active Gap Establishing
  with the new default setting that axes are only braked to avoid/minimize gap violations in the future, but do not actively move apart, as is the default behavior up to and including V3.2.
- New: The gap control direction "mcGapCtrlDirectionBoth" can be used in connection with the gap control mode mcGapCtrlModeFast.
- Requires TwinCAT V3.1.4024.40 or higher

As of TF5400 V3.2.27:

- Optimizations to MC_GearInPosCA that prevent SAF cycle offset between master and slave axis.
- Optimizations to the gap controller when the axis is already in the target position and only the gap changes. If the adjacent mover is commanded, the new gap takes effect.
- Requires an x64 platform

As of TF5400 V3.1.10.63:

- Requires TwinCAT V3.1.4024.24 or higher

As of TF5400 V3.1.10.30:

- Behavioral change in modulo positioning. Additional turns are now to be commanded via the new parameter `ST_MoveAbsoluteCAOptions.AdditionalTurns`. Please refer to the notes on Modulo positioning [▶ 116].

As of TF5400 V3.1.10.1:

- Track management
- Revised state diagram
- Requires TwinCAT V3.1.4024.7 or higher

As of TF5400 V3.1.6.3:

- Geo compensation

As of TF5400 V3.1.4.4:

- New: As of TF5400 3.1.4.4 `MC_MAXIMUM` is supported as input value. For more detailed information please refer to the documentation for the respective function block.

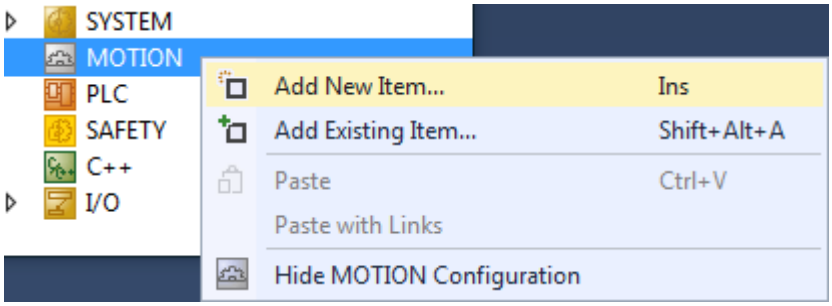
As of TF5400 V3.1.2.47:

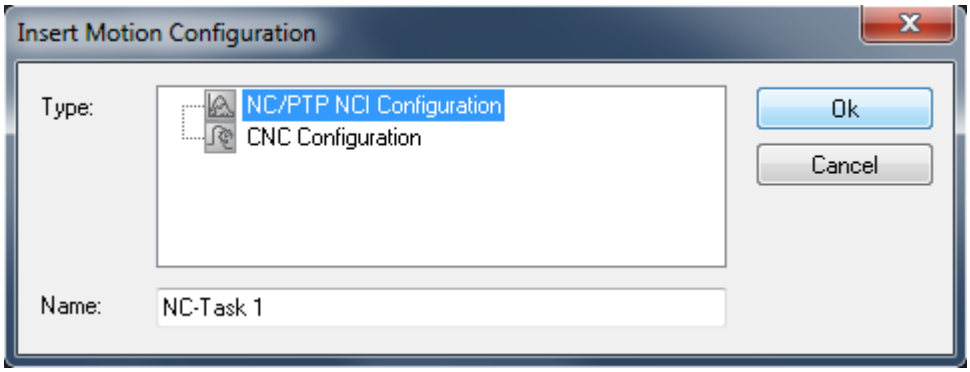
- New input MC_GAP_CONTROL_MODE [▶ 51] at each motion function block.
- New flag MC_GearInPosCAOptions [▶ 46]. OverrideSlaveDynamicRestrictions to improve the behavior when coupled to a master with non-constant velocity (e.g. encoder axis).
- New CA group parameter `GapControlModeDirection` defines the direction of gap monitoring.

4 Configuring the CA-Group for Collision Avoidance

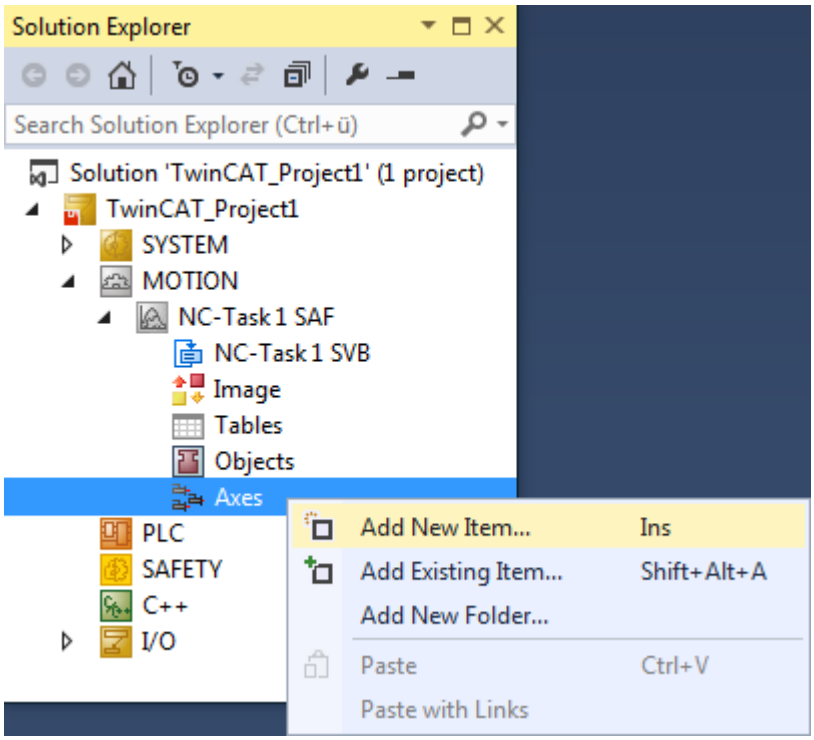
In principle, the configuration described here applies to all motion objects in the Advanced Motion Pack.

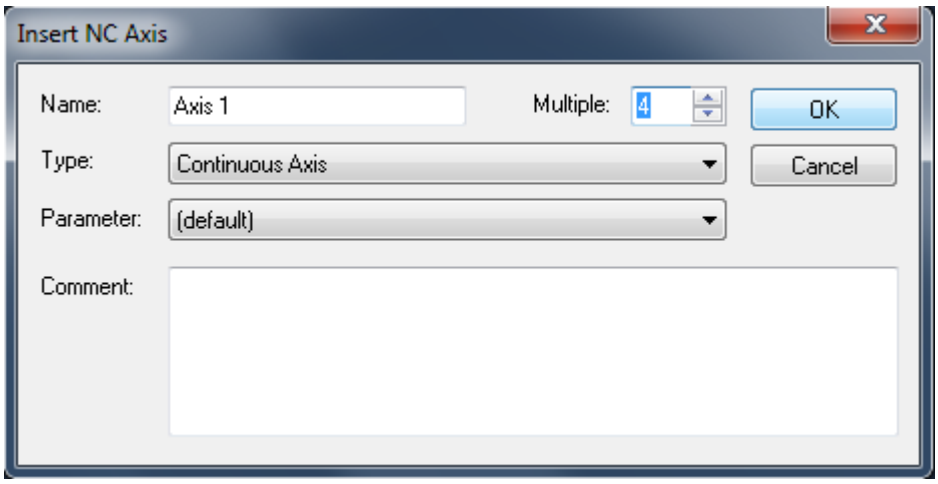
1. Add a new **NC/PTP NCI configuration** in the **Motion** section.



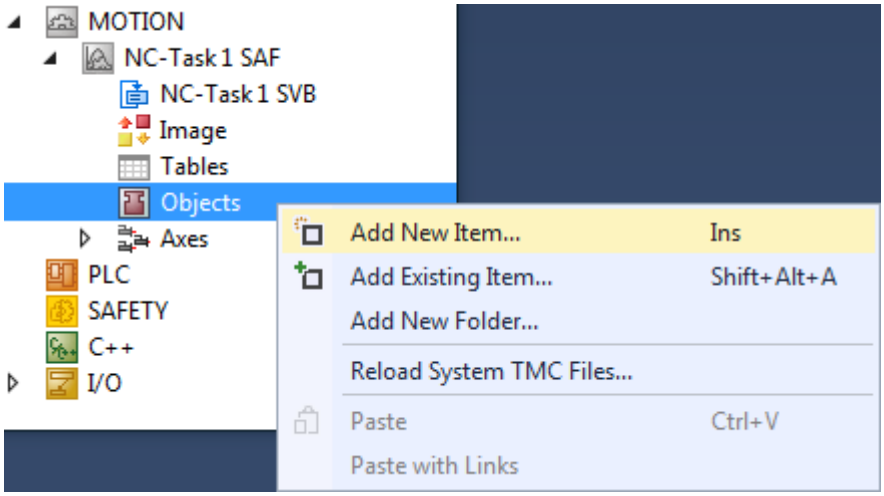


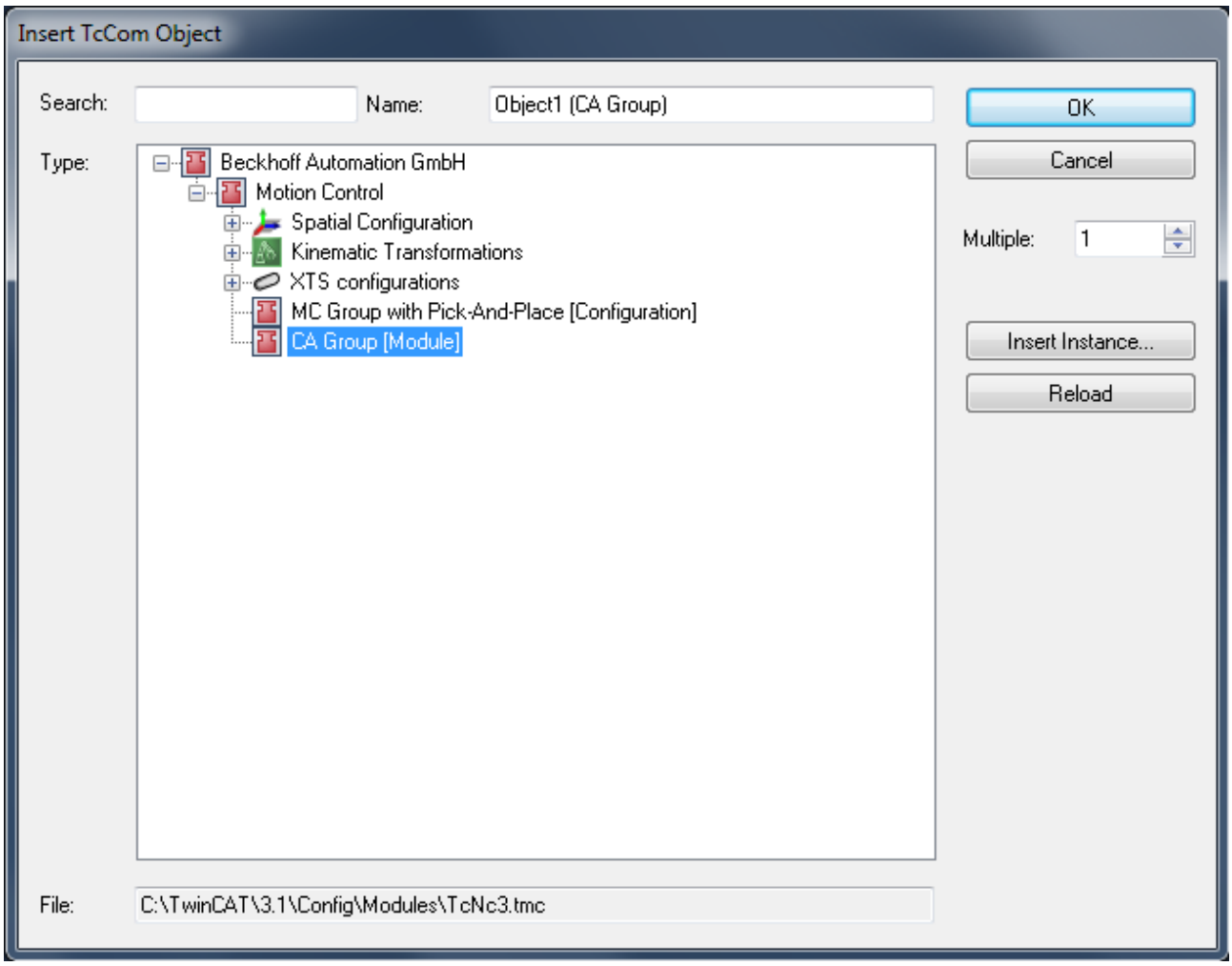
2. Add all axes to the NC configuration.



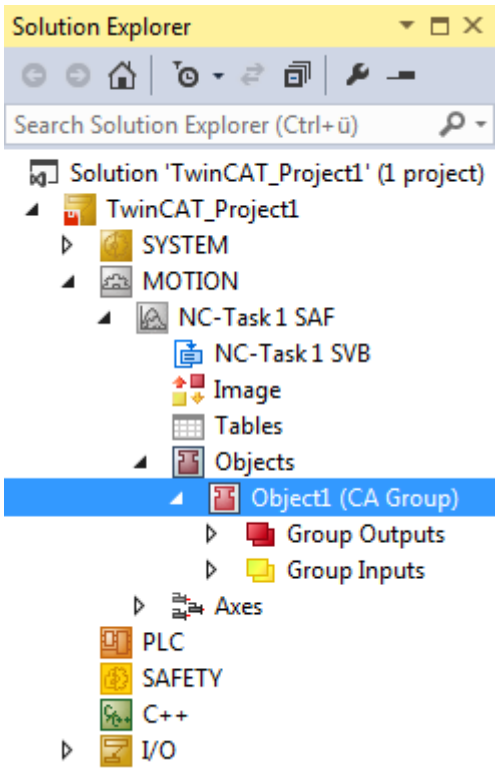


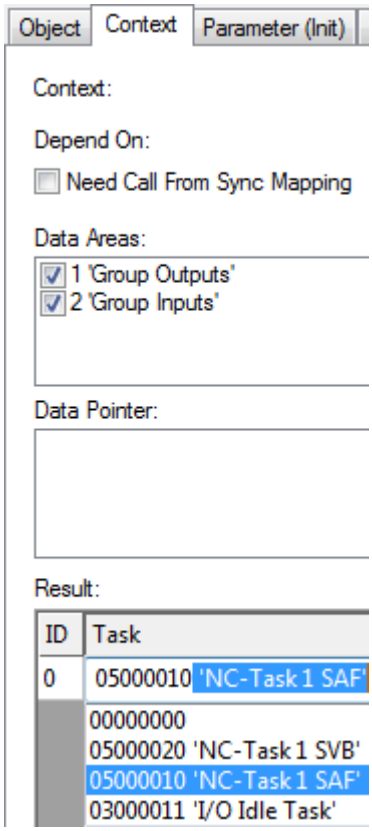
3. Add the corresponding group to the **Objects** entry in the NC configuration:
   For coordinated movement, multi-dimensional movements: CA Group (TF5410 TwinCAT 3 Collision Avoidance) [▶ 20].





4. Check the execution task in the group. This must always be set to "NC-Task 1 SAF".

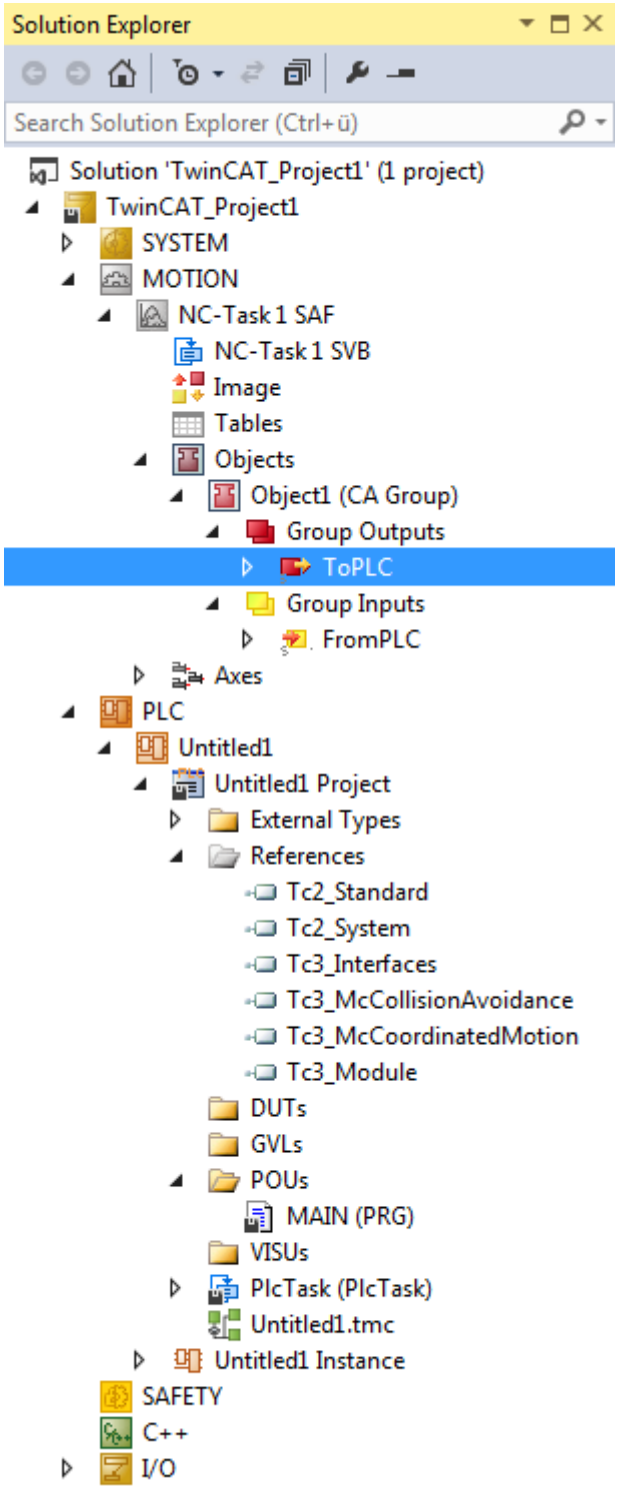


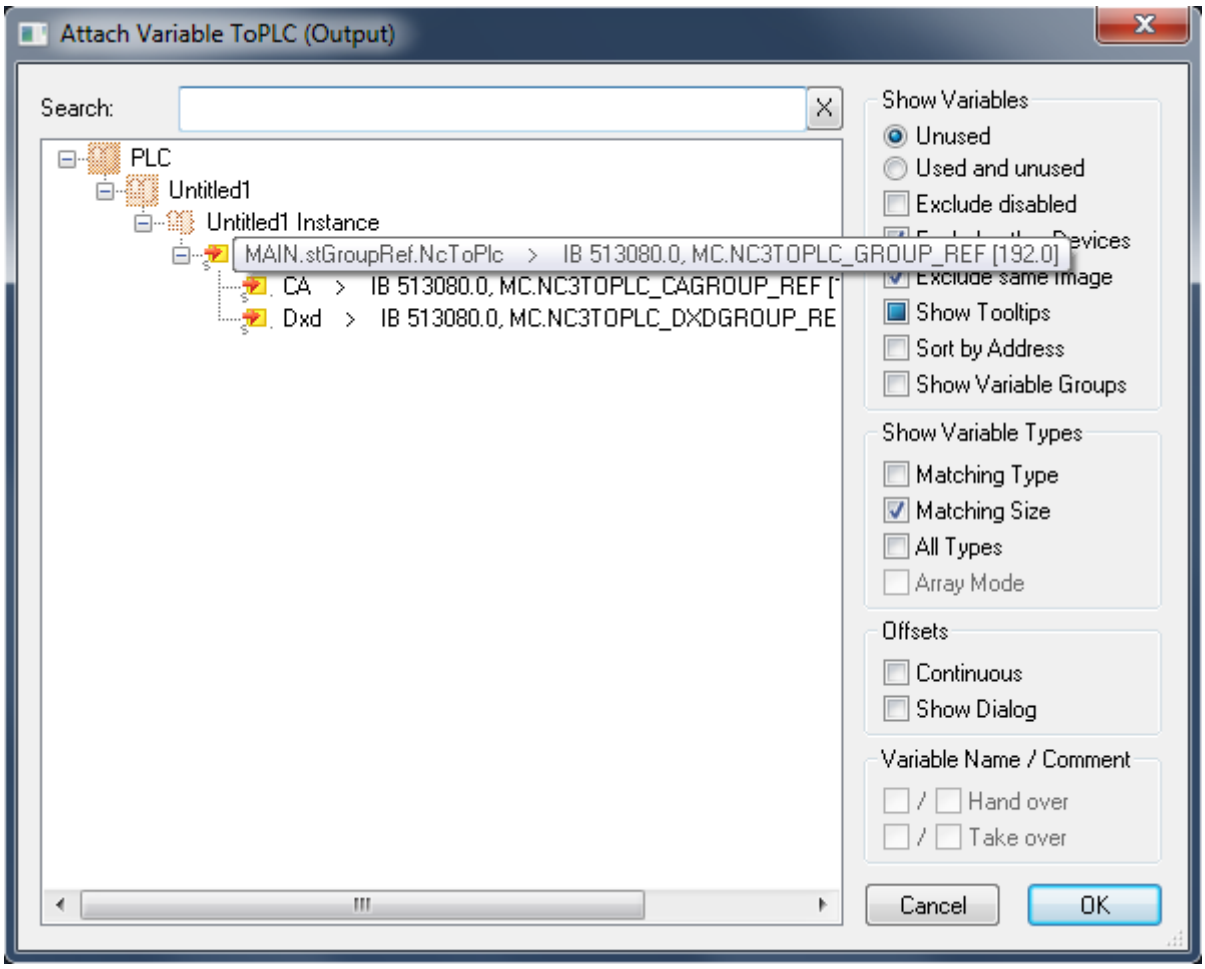


5. Configure the group parameters according to the desired application.
   For further explanation of the group parameters, see CA Group (TF5410 TwinCAT 3 Collision Avoidance) [▶ 20].
6. To address the group from the PLC, a cyclic interface must be declared and linked to the I/Os of the group (see PLC library Tc3_McCoordinatedMotion [▶ 56]). To address and enable the axes, the library Overview must be added to the project.

⇨ A new NC/PTP NCI configuration has been created.

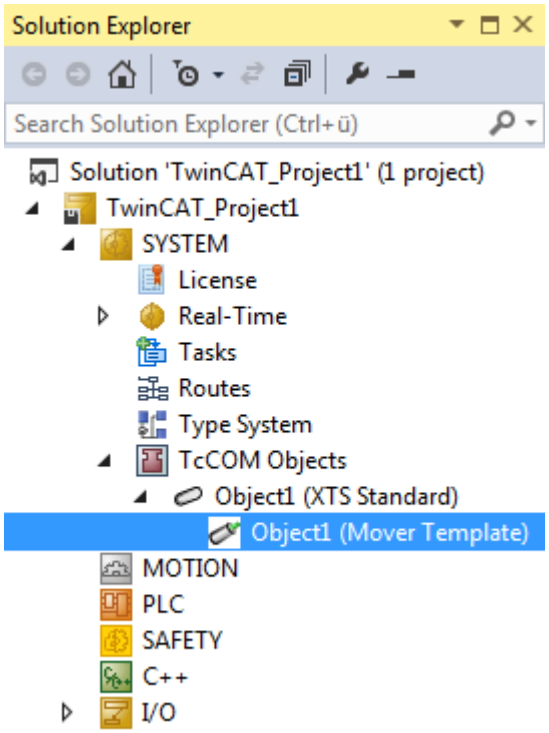
```
VAR
    stGroupRef : AXES_GROUP_REF;
END_VAR
```


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'TwinCAT_Project1' (1 project)
TwinCAT_Project1
SYSTEM
MOTION
NC-Task 1 SAF
NC-Task 1 SVB
Image
Tables
Objects
Object1 (CA Group)
Group Outputs
ToPLC
Group Inputs
FromPLC
Axes
PLC
Untitled1
Untitled1 Project
External Types
References
Tc2_Standard
Tc2_System
Tc3_Interfaces
Tc3_McCollisionAvoidance
Tc3_McCoordinatedMotion
Tc3_Module
DUTs
GVLs
POUs
MAIN (PRG)
VISUs
PlcTask (PlcTask)
Untitled1.tmc
Untitled1 Instance
SAFETY
C++
I/O



4.1 Geo Compensation

Geometrical Information



Geometrical information is required for geo-compensation. This geometrical information is configured in the *TwinCAT SYSTEM\TcCOM Objects* subtree.

Table of an XTS standard object

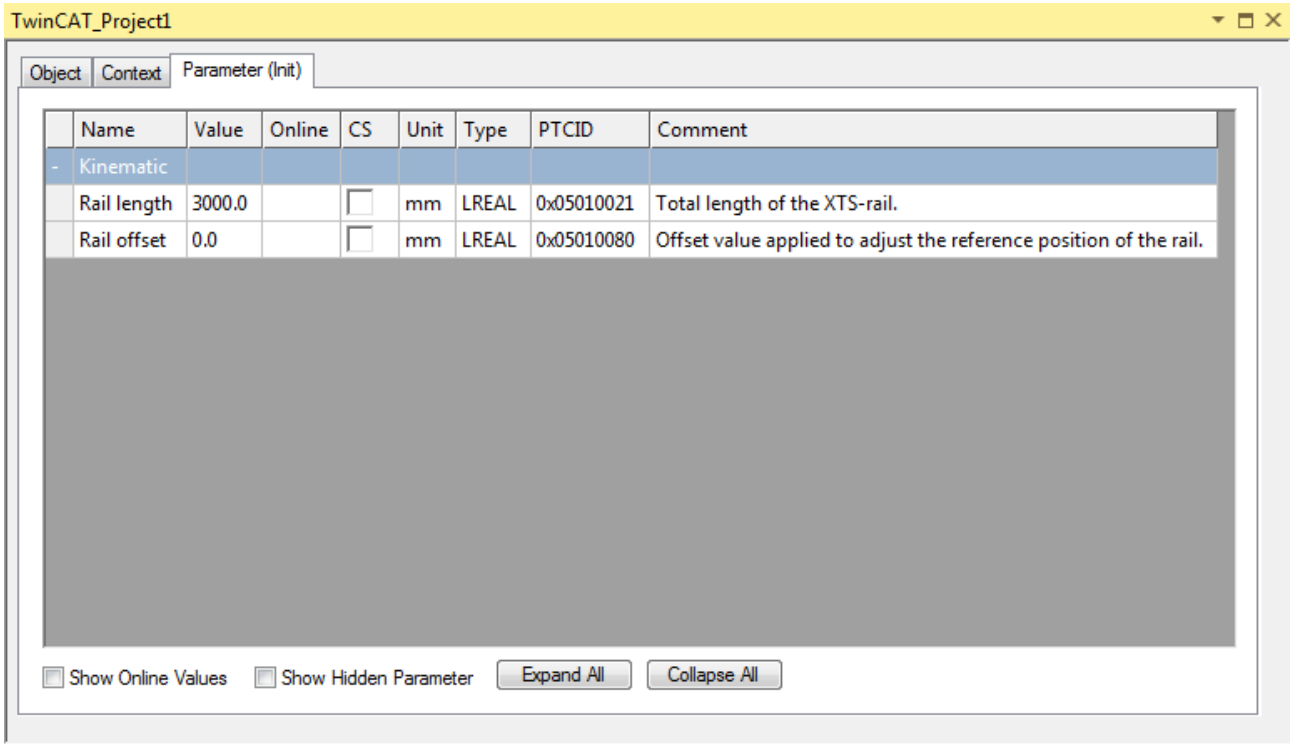
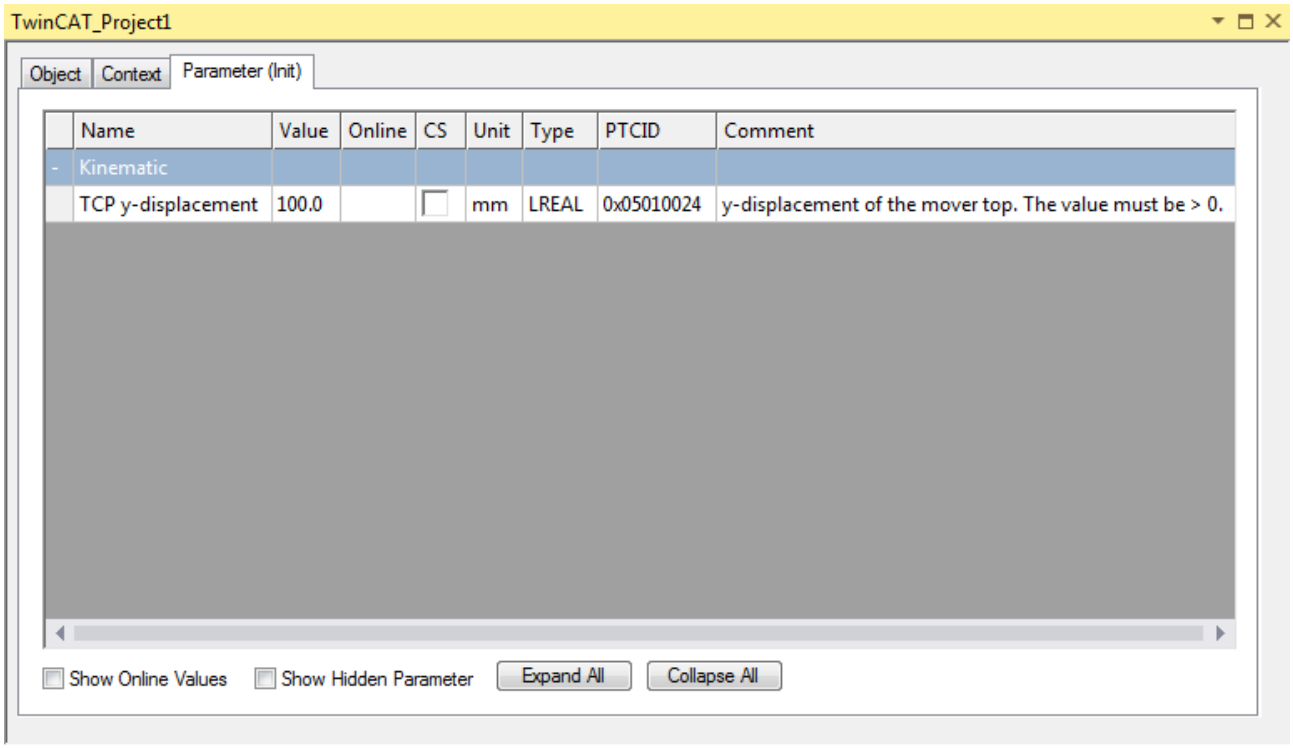
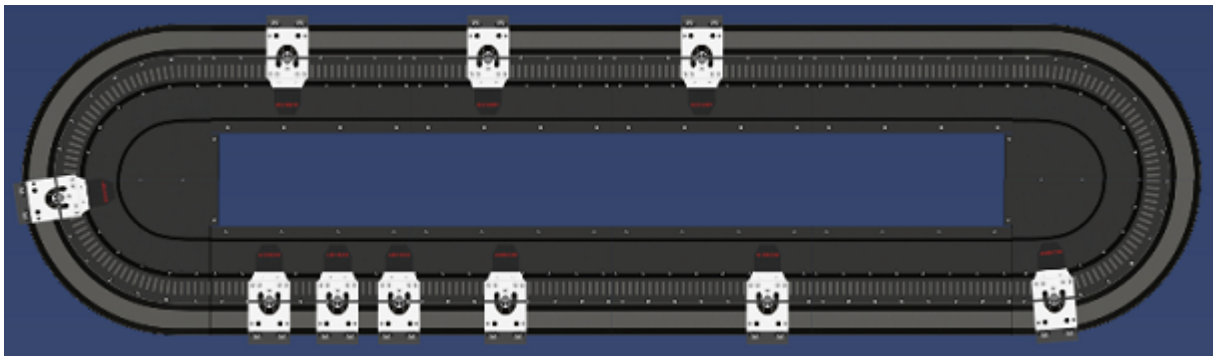


Table of a mover template object



The XTS Standard Object describes the geometry of a standard XTS motor path. Objects designated as mover templates each define the geometry of a single mover type, including a shift along the y-shift component. A Mover Template is added to the XTS Standard Object to extend the standard geometrical information with the geometrical information of the mover. A Mover Template can be referred to by all axes that use its configuration.

XTS Standard Object



The XTS Standard Object defines a motor path that has the starter kit geometry including two curves of 180 degrees. These curves are connected by two straight sections of equal length. The length of these straight sections can be changed during configuration. Thus, the **Rail Length** parameter of the XTS Standard Object configures the total length: both curves plus both straights. A zero shift (offset) can be configured in the XTS Standard Object for the position information on the XTS motor path x.

Rail length: Total length of the XTS rail.

Rail offset: Offset value for adjusting the reference position of the rail. See below and the figure Starter kit geometry.

Rail Offset: A zero shift

Each XTS system includes a segment that sets the zero position in the x-direction. The geo-compensation uses the starter kit geometry. In the geo-compensation, the segment for determining the zero position has a fixed place. It is the curved element in the upper left corner before the first straight element.

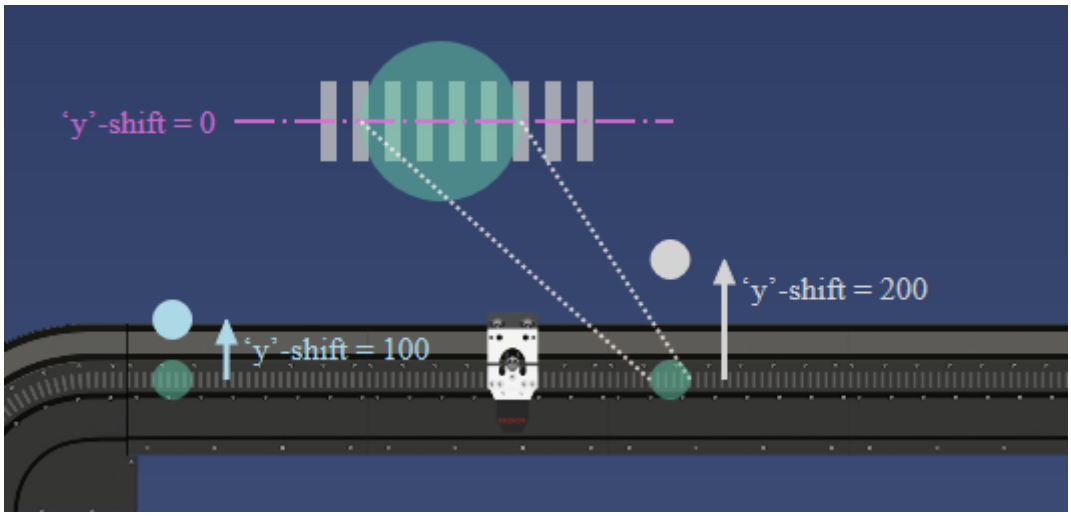
To set the zero position elsewhere and to start counting the x-coordinate from another position, a zero shift – the **Rail Offset –** can be defined.

The figure shows the **Rail Offset** between the two dashed lines. The left line shows where the segment for setting the zero position ends. The dashed line on the right and the mover shown illustrate how a position value is interpreted by a mover. The dashed line divides the mover into two halves. The mover is at the zero position. However, the determination of a **Rail Offset** itself does not require a mover.

Note on the availability of the zero shift

Currently, the starter kit geometry is the only geometry available for geo-compensation: Two curves of 180 degrees and two straight sections of equal length that connect these curves.

Mover Template Object



A Mover Template Object initially adopts the geometrical information of the XTS Standard Object. In addition, it describes a mover path geometry, i.e. the y-shift of a particular mover type. A Mover Template can be reused for different movers that have the same path geometry, i.e. the same tool center path. A Mover Template can be activated and deactivated in run mode. The template for a mover can thus be changed in run mode.

y-shift of the TCP: Configurable y-shift for controlling the path dynamics. The y-shift must be positive or equal to zero. For each point of an XTS motor path, it describes a point of a mover path that lies perpendicular to the tangent of the motor path that runs through this point of the motor path. In this direction, with orientation of the geometry of the XTS Standard Object to the outside, this point of the mover path is shifted away from the XTS motor path by the value of the y-shift. This shifted point is also known as the tool center point (TCP). Together, the y-points describe a path that is termed the tool center path.

On a straight section, the motor coils form a pattern similar to a zebra crossing. If this straight section lies in the middle of this pattern and divides each motor coil into an upper and a lower half, the y-shift has a value of zero on this straight section (see figure). If the y-shift is zero, the path dynamics are controlled collectively in the vertical center of the motor coils.

Note on the availability of the zero shift

Currently, the starter kit geometry is the only geometry available for geo-compensation: Two curves of 180 degrees and two straight sections of equal length that connect these curves.

5 Differences between MC2 and MC3

This chapter lists differences between MC2 and MC3 (as introduced in TF5400 Advanced Motion Pack).

Axes

	MC2	MC3
Maximum dynamics	The velocity defined in axis parameterization is interpreted as physical maximum value. Acceleration, deceleration and Jerk specified in the axis are default values that only have an effect if no dynamics is specified in FBs.	There are maximum values for velocity, acceleration, deceleration and jerk which limit the values that can be set in FBs. Moreover default dynamics can be selected by user at respective FB input.

PLC Library

	MC2	MC3
Default values	For dynamics parameters of type LREAL "0" is default value. If "0" is set the default parameters from the axes are used.	The constant MC_Default is introduced (see MC_LREAL/Special Input Values [▶ 115]). "0" is not interpreted as default value but as a normal value which in case of dynamics can be invalid.
Timing of FB outputs	FB returns values that were valid at the start of PLC cycle.	FB returns values that are valid at the moment PLC code is executed. This may lead to timing difference between cyclic interface and FB output.
Decoupling	A special function block can be used (e.g. MC_GearOut/ MC_CamOut)	The slave axis is decoupled by sending another motion command with Buffermode mcAborting.

6 CA Group (TF5410 TwinCAT 3 Collision Avoidance)

The CA group links axes to add Collision Avoidance to the PTP functionalities.

Dynamic values

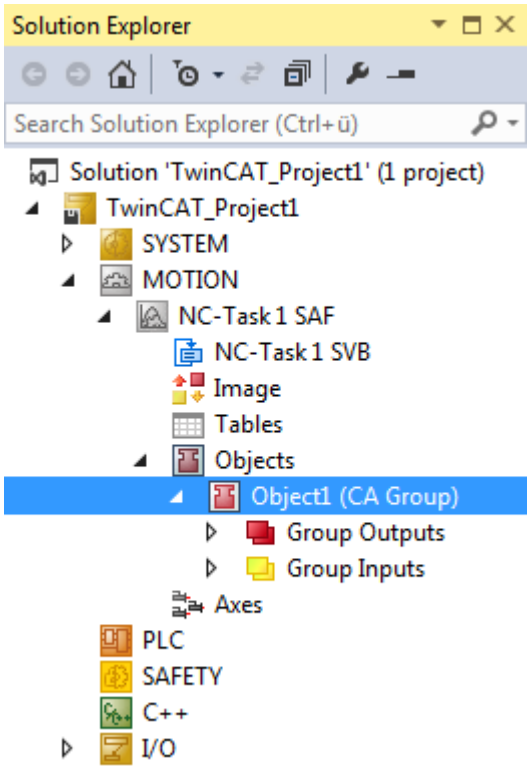
- Velocity **Vel**: velocity,
- Acceleration **Acc**: positive acceleration,
- Deceleration **Dec**: braking acceleration, negative acceleration,
- Jerk: jerk.

Setpoints and limits	• An axis traverses set dynamic values. During this motion, the maximum dynamic values set the limits for the dynamic profile.
CA Group	• A CA group provides parameters to set default values for dynamic values. These default values are used for Standby Gap Control. They are not used as default parameters for Motion commands if no parameter has been specified.
Axis	• The maximum values for the dynamic limits can be set in the axis parameters. • These maximum values can be determined by the physical properties (inertia, mass, maximum current, motor size, ..) of the axis or a workpiece, for example.

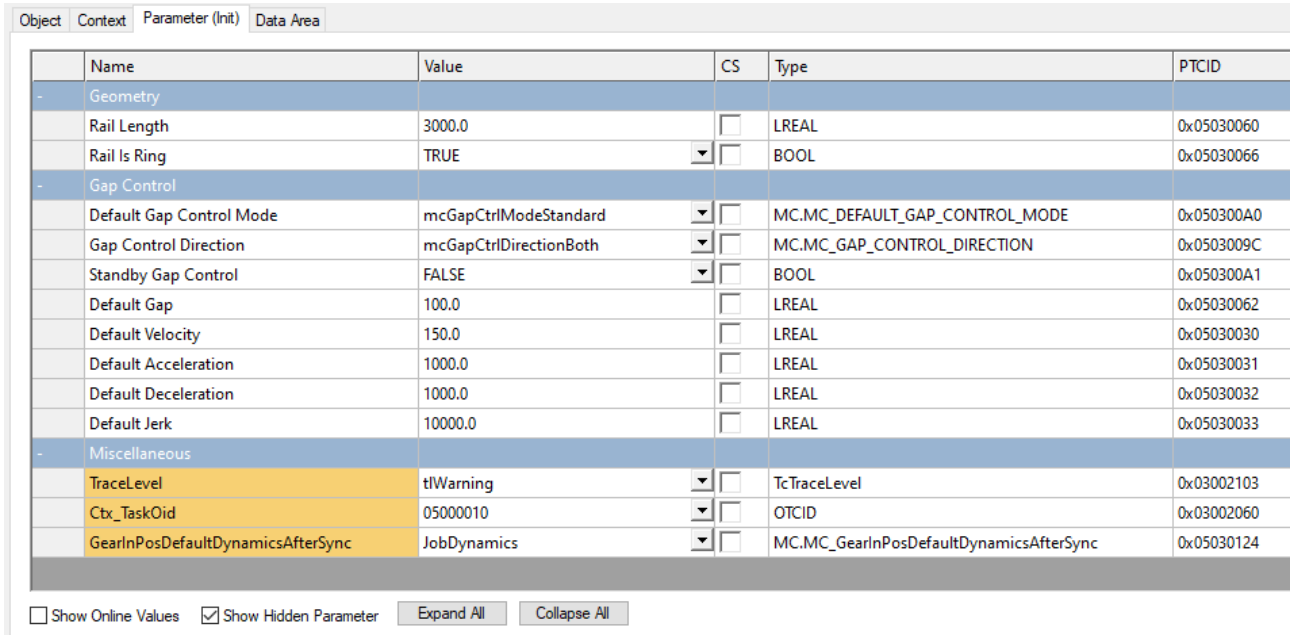
Gap

Neighborhood	• A Gap requires two or more adjacent axes (movers). • A Gap is always located between two directly adjacent movers.
Counting direction	• The Gap is defined in positive counting direction from the current mover to the mover directly ahead. • This positive counting direction corresponds to the counting direction of the setpoint generation.
Successor; predecessor	• Current mover: directly following mover, successor. • Mover directly ahead: predecessor.
Size	• Size of a respective Gaps = (set position predecessor) - (set position successor).
Lower limit	• The Gap size is set to a lower limit that must not be undershot. The parameter "Default Gap" sets a value for this lower limit for the entire CA group as long as no other gap size is set for a current mover. An individual value for this lower limit of the Gap size can be applied as an input value to each of the motion function blocks: "MC_MoveAbsoluteCA", "MC_MoveRelativeCA", "MC_HaltCA" or "MC_GearInPosCA".
Gap Control Mode	• "Gap Control Mode" mcGapCtrlModeFast generally controls closer to this lower Gap size limit than "Gap Control Mode" mcGapCtrlModeStandard.
Gap Control Direction	• "Gap Control Direction" mcGapCtrlDirectionPositive:: The successor is the only mover that controls the size of the Gaps enclosed by the predecessor and successor. • "Gap Control Direction" mcGapCtrlDirectionBoth: Both adjacent movers control the size of the Gaps they enclose.

Opening the dialog “Parameter (Init)”



Root node of a CA group.



Parameters for a CA group.
The table column "Value" shows the preset parameter value. The table column "Comment" contains brief parameter descriptions.

Parameter	Description
Geometry	
Rail Length	Length of the rail on which the axes (movers) are mounted.
Rail Is Ring	Indicates whether the rails form a closed circle. In this case (TRUE), Collision Avoidance is enabled between the first mover in the row and the last mover.

Parameter	Description
Gap Control	
Default Gap Control Mode	Different modes are available for gap control (see "MC_DEFAULT_GAP_CONTROL_MODE [▶ 27]").
Gap Control Direction	Various settings are available for the control direction of the Gaps (see section "MC_GAP_CONTROL_DIRECTION [▶ 29]“).
Active Gap Establishing	From version 3.3, 'Collision Avoidance' does not trigger any active movement and only intervenes in the motion profile with a delay. The behavior of version 3.2, in which the 'Collision Avoidance' triggered an active movement to establish the parameterized gap, can be restored by setting the 'Active Gap Establishing' parameter in the CA group to 'True'. From version 3.3, this is set to 'False' by default.
Standby Gap Control	If TRUE, Collision Avoidance is always active, even if no motion command was issued to the axis. ***Notice*** **The axes move directly after MC_GroupEnable when Standby Gap Control is TRUE. If the gap between two axes (mover) is smaller than the Default Gap (see next parameter), the axes will move to reach the demanded gap. This motion is independent of any motion command. This behavior also applies if the axes are too close to each other after a group reset.**
Default Gap	This gap is used for the Standby Gap Control and if no gap was specified at any CA motion command.
Default Velocity	This velocity is used for Standby Gap Control, i.e. when no motion command is active (e.g. directly after MC_GroupEnable [▶ 62]). It is not used as a default velocity for any motion command if no velocity was specified.
Default Acceleration	This acceleration is used for Standby Gap Control, i.e. when no motion command is active (e.g. directly after MC_GroupEnable [▶ 62]). It is not used as a default acceleration for any motion command if no acceleration was specified.
Default Deceleration	This deceleration is used for Standby Gap Control, i.e. when no motion command is active (e.g. directly after MC_GroupEnable [▶ 62]). It is not used as a default deceleration for any motion command if no deceleration was specified.
Default Jerk	This jerk is used for Standby Gap Control, i.e. when no motion command is active (e.g. directly after MC_GroupEnable [▶ 62]). It is not used as a default jerk for any motion command if no jerk was specified.
GearInPosDefaultDynamicsAfterSync (hidden!)	Specifies the default dynamics used in MC_GearInPosCA AfterSyncDynamics. In the default state, the value "JobDynamics" is set. The parameter is not available for older projects (created with versions < 3.1.10), but is set internally to 'MaximumSlaveDynamics'.

After reloading the TMC file, "JobDynamics" is set as the default value (see MC_GearInPosDefaultDynamicsAfterSync [▶ 30]).

✓ If an NC configuration has already been added, the MOTION subtree contains an SAF task subtree.
✓ The SAF task subtree contains the "Objects" subtree.
✓ The "Objects" subtree can contain a CA group.

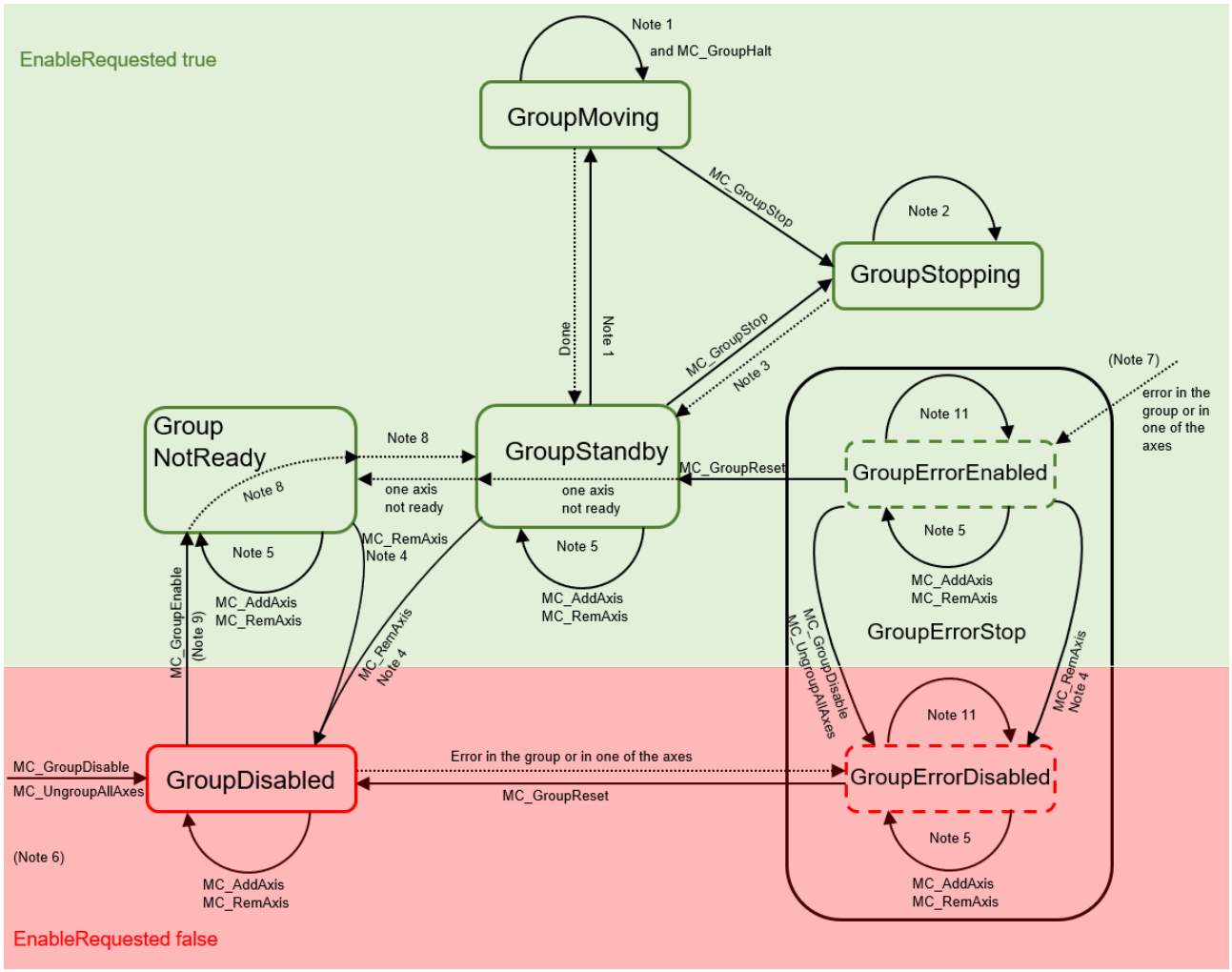
1. Double-click the root node of the CA group whose parameters you want to view or set.
2. Select the "Parameter (Init)" tab.

⇨ The dialog “Parameter (Init)” is opened.
⇨ It contains a table with parameters for the selected CA group.
⇨ These parameters are divided into the groups "Geometry", "Gap Control" and "Misc.", if applicable.

7 State diagrams

7.1 State diagram valid for V3.1.6

The state diagram describes the state of an axis group. The states described here can be read from the PLC using the function block MC_GroupReadStatus.



: state in the internal state machine, not visible in PLC via MC_ReadGroupStatus

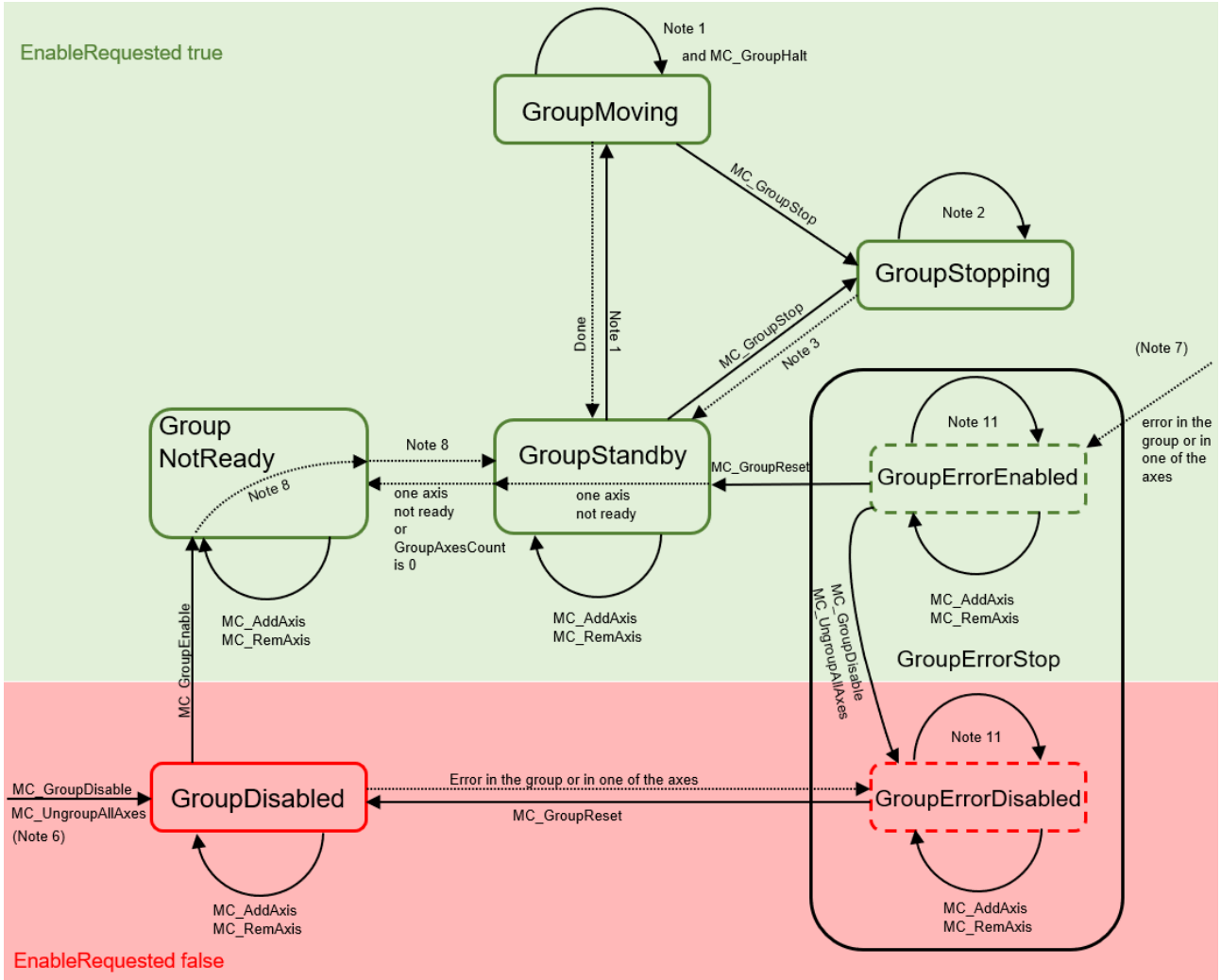
: state visible in MC_ReadGroupStatus

Note	Description
1	Applicable for all non-administrative (movement) function blocks.
2	In the GroupStopping state all function blocks can be called but they are not executed, with the exception of MC_GroupDisable and MC_UngroupAllAxes, which cancel the stop and create the transition to GroupDisabled.
3	MC_GroupStop.DONE AND NOT MC_GroupStop.EXECUTE
4	Transition is applicable when the last axis is removed from the group.
5	Transition is applicable while the group is not empty.
6	MC_GroupDisable and MC_UngroupAllAxes can be output in all states. They change the state to GroupDisabled. If they are output in an error state, the state changes to GroupErrorDisabled.
7	From any state with EnableRequested TRUE.

8	If "bIsControlLoopClosed" is TRUE for all axes and the group is not empty. "bPositiveDirection"/"bNegativeDirection" do not have be enabled.
9	MC_GroupEnable returns an error if the group is empty.
10	MC_GroupReset has no effect if the state is different from GroupErrorStop.
11	In the error states all administrative function blocks are permitted with the exception of MC_GroupEnable. However, in the error states you can only create state transitions, e.g. to GroupErrorDisabled for MC_GroupDisable or MC_UngroupAllAxes and MC_RemoveAxisFromGroup, when the last axis is removed.
12	MC_GroupReset must be called to exit the GroupErrorStop state.

7.2 State diagram valid for V3.1.10

The state diagram describes the state of an axis group. The states described here can be read from the PLC using the function block MC_GroupReadStatus.



: state in the internal state machine, not visible in PLC via MC_ReadGroupStatus

: state visible in MC_ReadGroupStatus

Note	Description
1	Applicable for all non-administrative (movement) function blocks.

2	In the GroupStopping state all function blocks can be called but they are not executed, with the exception of MC_GroupDisable, which cancels the stop and creates the transition to GroupDisabled.
3	MC_GroupStop.DONE AND NOT MC_GroupStop.EXECUTE
4	-
5	-
6	MC_GroupDisable can be output in all states and changes the state to GroupDisabled. If they are output in an error state, the state changes to GroupErrorDisabled.
7	From any state with EnableRequested TRUE.
8	If "bIsControlLoopClosed" is TRUE for all axes and the group is not empty. "bPositiveDirection"/"bNegativeDirection" do not have be enabled.
9	"bIsControlLoopClosed" and the two flags "bPositiveDirection"/ "bNegativeDirection" must be set to TRUE.
10	-
11	In the error states all administrative function blocks are permitted with the exception of MC_GroupEnable. However, in the error states you can only create state transitions, e.g. to GroupErrorDisabled for MC_GroupDisable or MC_UngroupAllAxes and MC_RemoveAxisFromGroup, when the last axis is removed.

In the GroupMoving state stationary axes may be added to and removed from a **CA group**. If an attempt is made to add a moving axis to a group or remove it from the group, the command is rejected with an error (the group change with a moving axis is also rejected).

MC_GroupReset has no effect if the state is different from GroupErrorStop.

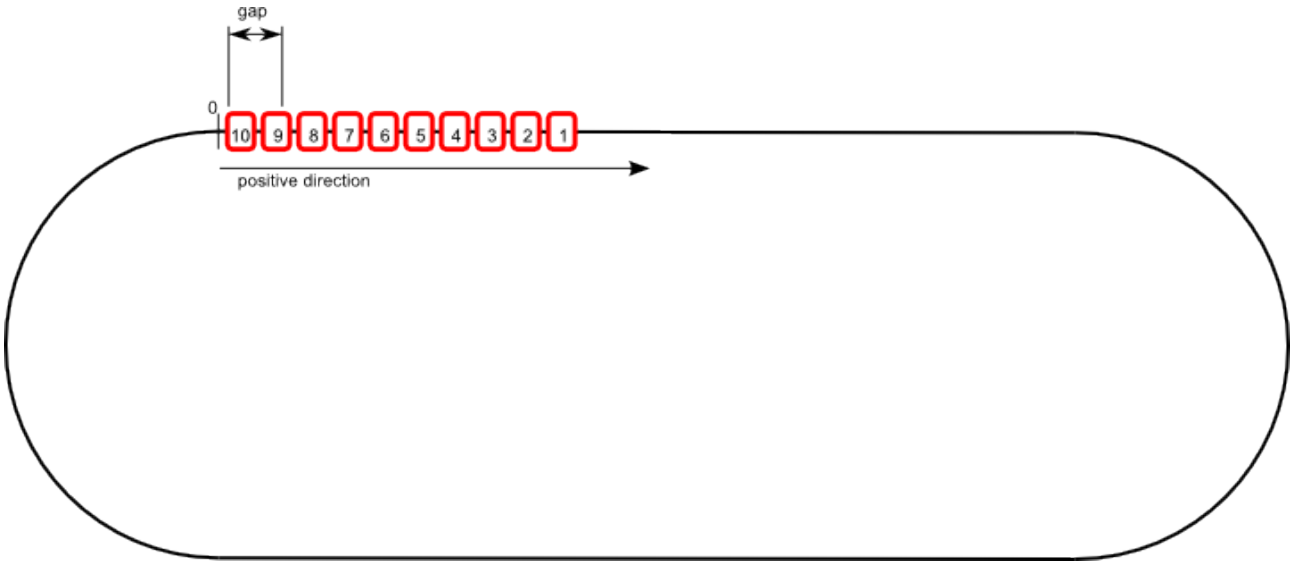
8 Background Information

8.1 Collision Avoidance

8.1.1 Basics of Collision Avoidance

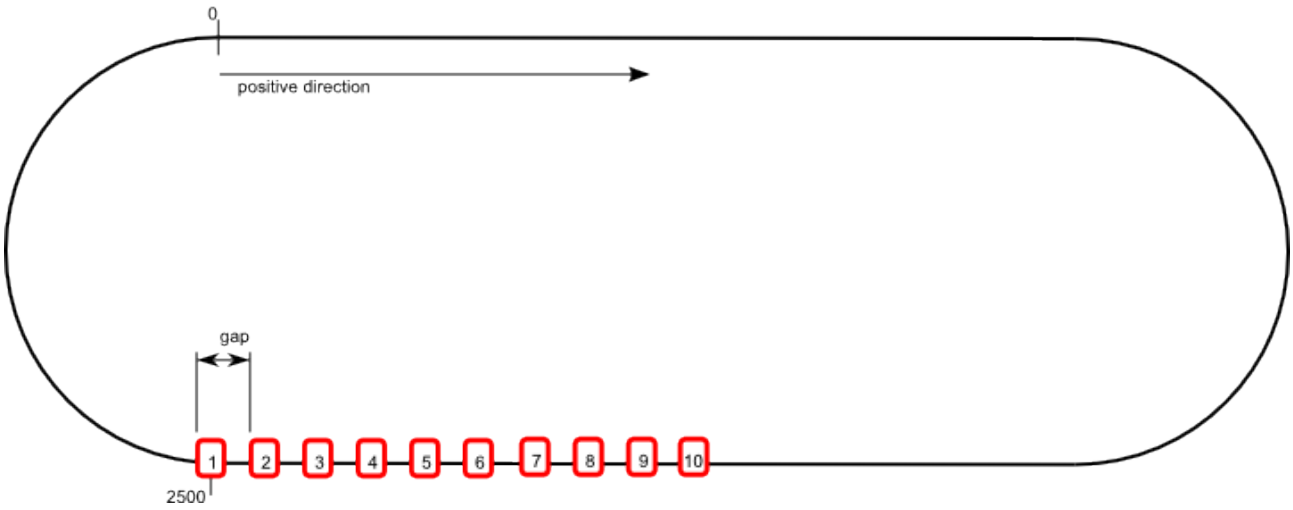
✓ All objects (CA Group and all axes) must be created, parameterized and linked (see "Configuration", "CA Group Parameterization [▶ 20]").

✓ This example uses the default values for all gap control parameters and 10 axes in the group. All axes are mounted on a closed rail (XTS) with a length of 3000 mm. The position of the axes (movers) is arbitrary, the default gap which is parameterized in the group is not observed:



1. All axes must be added to the group (see examples in MC_AddAxisToGroup [▶ 58]).
   ⇨ The order of the axes for the collision avoidance is determined by their actual position on the rail.
   ⇨ If the positions of the axes are equal (e.g. for simulation axes), the order in that the axes are added to the group is essential. In this case, the axis that was added last is the first axis in the group.
   ⇨ The "IdentInGroup" has no relevance for the order used for collision avoidance.
2. Enable Group (see MC_GroupEnable [▶ 62]).
   ⇨ The GroupState is now mcGroupStateStandby (see MC_GroupReadStatus [▶ 65] or Cyclic Group Interface [▶ 114]), the GroupAxesCount is 10 (see Cyclic Group Interface [▶ 114]).
   ⇨ The position of the axes (movers) has not changed, the gap is still not observed.
3. Issue "MC_MoveAbsoluteCA [▶ 33]" for all axes (movers) to the same position (2500 mm).

⇨ The first mover that is the mover with the largest absolute position, here mover 1, reaches the target position at 2500 mm. The other movers line up, each keeping the gap to its forerunner. The forerunner of the first mover is the last one (since the group parameter Rail Is Ring is set to TRUE).



8.1.2 MC_DEFAULT_GAP_CONTROL_MODE

The Gap Control Mode [▶ 20] specifies the behavior of the Collision Avoidance. Following modes are available:

```
TYPE MC_DEFAULT_GAP_CONTROL_MODE :
(
mcGapCtrlModeStandard               := 16#1,
mcGapCtrlModeFast                   := 16#2
)
END_TYPE
```

Examples

Example mcGapCtrlModeStandard:

- ✓ Configuration with four axes (mover) in the CA Group. The Rail Length [▶ 20] is 3000mm and the rail is closed (e.g. XTS-system).
- ✓ The first axis in line (blue) is standing at position 0.0mm, the remaining three axes are lined up behind with a respective gap of 100mm.
- ✓ The Gap Control Mode is set to mcGapCtrlModeStandard.

1. MC_MoveAbsoluteCA is issued to all axes to the Position 3000mm, the Gap is 100mm. All Axes have the same dynamics (Velocity, Acceleration, Deceleration, Jerk).

⇨ The axes fan out characteristically during the acceleration phase, such that a collision during the motion command is prevented. The first axis (blue) reaches the target position, the remaining axes line up successively with the configured Default Gap [▶ 20].

Example mcGapCtrlModeFast:

✓ Configuration with four axes (mover) in the CA Group. The RailLength is 3000mm and the rail is closed (XTS-system).

✓ The first axis in line (blue) is standing at position 0.0mm. The remaining three axes are lined up behind with a respective gap of 100mm.

✓ The Gap Control Mode is set to mcGapCtrlModeFast

1. MC_MoveAbsoluteCA is issued to all axes to the Position 3000mm, the Gap is 100mm. All Axes have the same dynamics (Velocity, Acceleration, Deceleration, Jerk).

⇨ All Axes move at the same time and with the full dynamics. The gap between the axes is kept nearly constant. The first axis reached the target position, the rest lines up behind.

8.1.3 MC_GAP_CONTROL_DIRECTION

Gap Control Direction "mcGapCtrlDirectionPositive"

CA Group

- The set Gap Control Direction applies to the entire CA group. [▶ 20]

Successor

- The size of the Gaps is regulated in each case.
- The successor is the only mover that controls the size of the Gaps enclosed by both adjacent movers.

Gap Control Mode

- Gap Control Mode "mcGapCtrlModeStandard" or Gap Control Mode "mcGapCtrlModeFast" can be used to calculate the dynamic values of a directly following mover.
- The initialization parameter Default Gap Control Mode sets the same Gap Control Mode as the default algorithm for each successor within a CA group.

Individual

- You can change Gap Control Mode individually for each mover using any of the motion function blocks - MC_MoveAbsoluteCA, MC_MoveRelativeCA, MC_HaltCA or MC_GearInPosCA.

Computing power

- Gap Control Mode "mcGapCtrlModeStandard" generally requires less computing power than Gap Control Mode "mcGapCtrlModeFast".

Gap Control Direction "mcGapCtrlDirectionBoth"

Motion profiles

- Allows more general motion profiles, for example reverse motion.

CA Group
- The set Gap Control Direction applies to the entire CA group [▶ 20].

Successor and predecessor
- The size of the Gaps is regulated in each case.
- Both adjacent movers, predecessor and successor, control the size of the Gaps they enclose.

Gap Control Mode
- Gap Control Mode [▶ 27] “mcGapCtrlModeStandard” or from v3.3.19 Gap Control Mode “mcGapCtrlModeFast” can be used to calculate the dynamic values of a directly following mover.
- The initialization parameter Default Gap Control Mode sets the same Gap Control Mode as the default algorithm for each successor within a CA group.

Correlating control behavior

Neighborhood
- The size of a Gaps is regulated between two neighboring movers.
- Two Gaps are (directly) adjacent if the mover separating them is both successor and predecessor.

Chain
- (Directly) adjacent Gaps form a (non-trivial) chain.
- Within a chain the control of the respective Gaps is correlated.

Gap Control Mode
- Gap Control Mode [▶ 27] influences the correlating control type.
- The Gap Control Mode [▶ 27] "mcGapCtrlModeStandard" allows deviations from the target gap value for a single Gap and for the Gaps of a chain for softer control.

8.1.4 MC_GearInPosDefaultDynamicsAfterSync

```
TYPE MC_GearInPosDefaultDynamicsAfterSync  :
(    MaximumSlaveDynamics  := 16#0,
     JobDynamics  := 16#1
);
END_TYPE
```

Defines the default dynamics used for the MC_GearInPosCA command after the slave axis has become synchronous for the first time (see ST_GearInPosCAOptions [▶ 46]).

MaximumSlaveDynamics:	The maximum slave axis dynamics (velocity, acceleration, deceleration) is used as the default value for the AfterSyncDynamics. The jerk is not limited.
JobDynamics:	Job Dynamics (GearInPosCAs velocity, acceleration, deceleration and jerk) is used as the default value for AfterSyncDynamics.

8.2 Geo Compensation

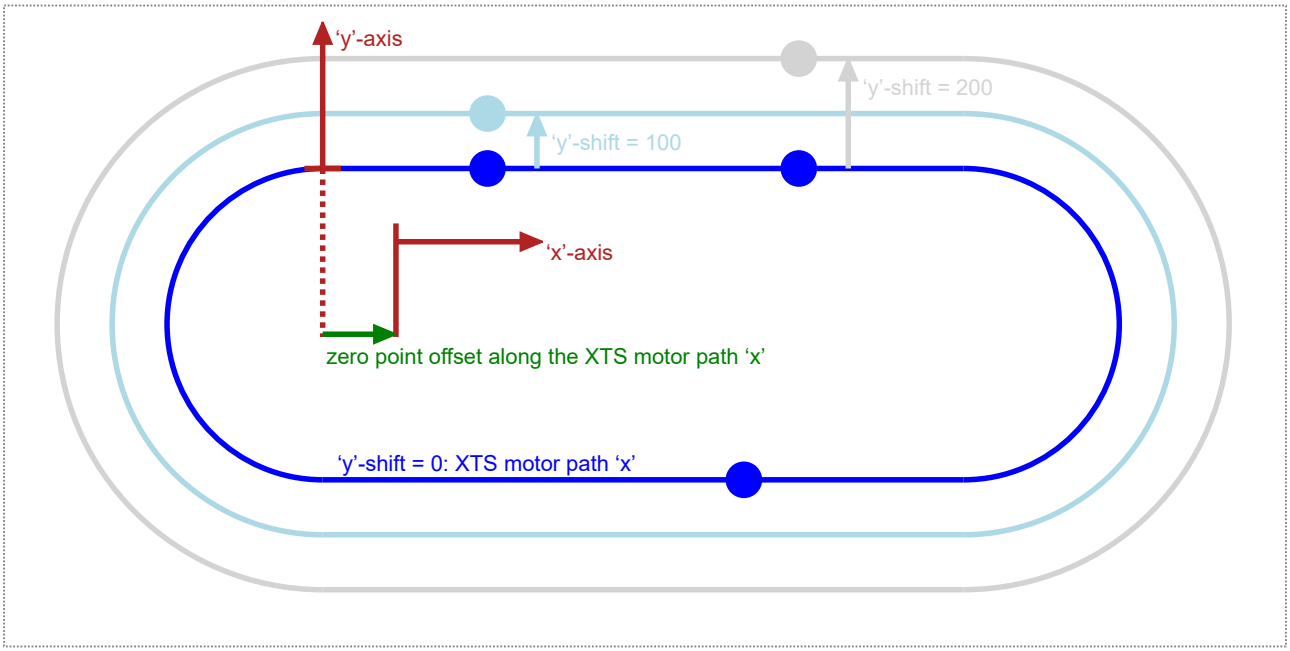


Fig. 1: Starter Kit Geometry.

Geo Compensation: Motivation

Geo Compensation defines an additional degree-of-freedom:

- A one-dimensional spatial transformation of motion dynamics control.
- Positional motion control always refers to the XTS motor path.

A y-axis perpendicular to the XTS motor path coordinate is introduced as an additional dimension. Motion dynamics can be controlled for a predefined path located on this y-component of displacement. This path may enable opportunities for enhanced mover motion dynamics.

- Motion dynamics refer to velocity, acceleration, deceleration and jerk behavior along a path.

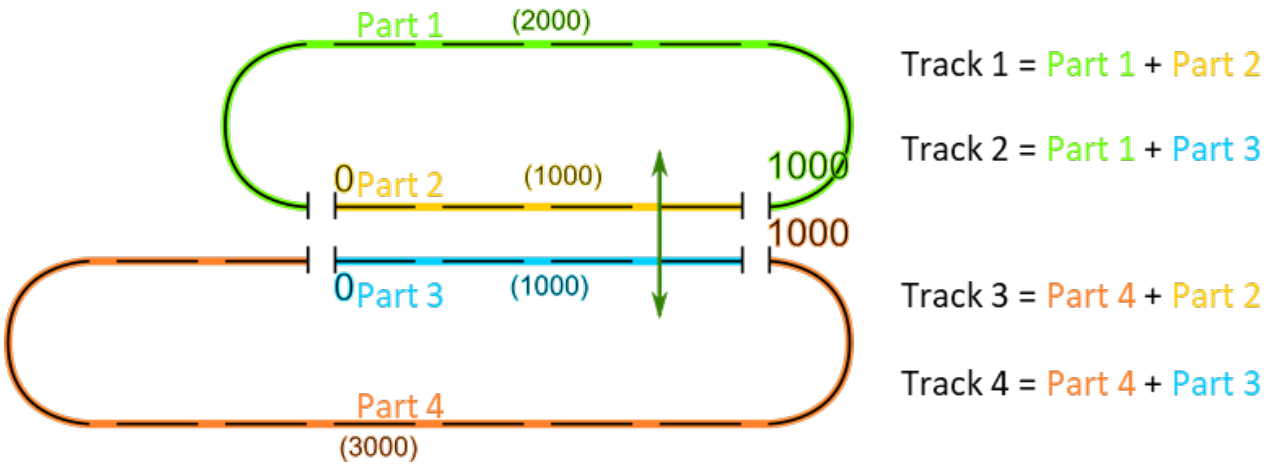
Generally, dynamical reference can be kept to the XTS motor path, thus leaving the usage of Geo Compensation as an optional opportunity.

Motivation Example: Center of Gravity

Within many XTS applications heavy tools or products are mounted on the movers. Altogether, a mover and its load form a vehicle. Generally, the vehicle center of gravity does not travel on the XTS motor path. On straight XTS segments XTS motor path velocity and center of gravity path velocity are identical. On curved XTS segments these path velocities differ. This difference leads to an acceleration or deceleration on the center of gravity path while XTS motor path velocity is kept constant. Thus, unintentional forces are at work on the XTS track, especially when curves are entered or left. To avoid some of these forces or to keep their magnitude low the center of gravity could be driven with a nearly constant velocity. This behavior is an example for what Geo Compensation may intend to achieve: As long as a mover and its load are not changed, the center of gravity path can be described and controlled dynamically by adding a radial shift to the XTS motor path. Because this shift points away perpendicularly from the XTS motor path, this shift is called y-shift.

8.3 Track management

With the aid of track management an XTS configuration can be divided into individual, spatially separated XTS parts. These can comprise just one or any number of consecutive motor modules. Individual, adjacent XTS parts can be combined to XTS tracks. The XTS parts and XTS tracks can be configured via the XTS Configurator. In the system manager, the XTS parts and XTS tracks are inserted as TcCOM modules, with a unique ObjectID, as child nodes below the XTS Processing Unit (see XTS documentation).



For each individual axis a track can be activated via the function block MC_ActivateTrack using the ObjectID of the XTS track. When activating an XTS track, the mover must be on an XTS part that is assigned to the track. ObjectID 0 can be used to reactivate the absolute reference system for the individual axis. The current target positions on the tracks and parts can be read out using the function block MC_ReadTrackPositions.

9 PLC Libraries

9.1 Tc3_McCollisionAvoidance

Overview

Function block	Description
Motion	
MC_MoveAbsoluteCA [▶ 33]	Moves a single axis to an absolute position with collision avoidance.
MC_MoveRelativeCA [▶ 36]	Moves a single axis over a relative distance with collision avoidance.
MC_HaltCA [▶ 39]	Stops a single axis with collision avoidance without locking it for further motion commands.
MC_GearInPosCA [▶ 41]	Couples a slave axis with a gearing factor and collision avoidance to a master axis.
MC_ReadTrackPositions [▶ 44]	Returns the current XTS track and XTS part target positions with the corresponding object IDs.
MC_ActivateTrack [▶ 45]	Activates a track as a reference system, which can then be used in various motion function blocks for positioning.

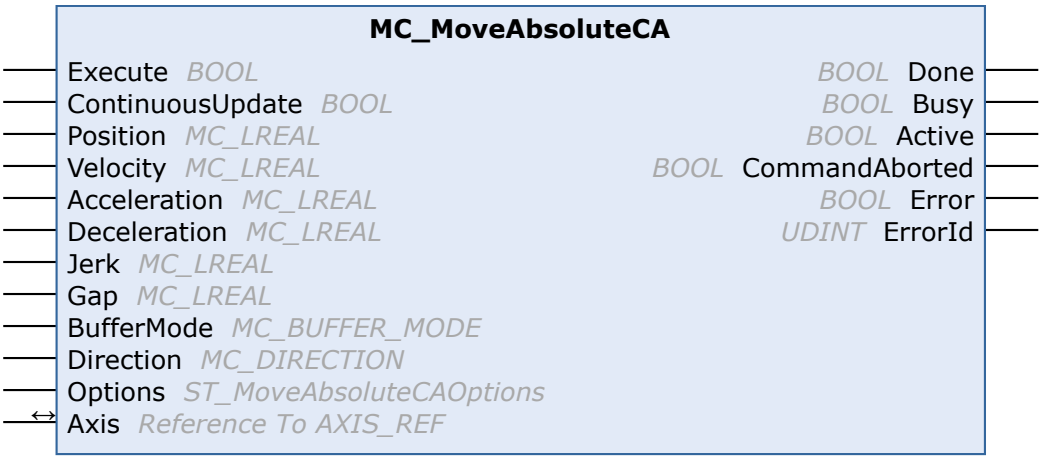
Structures and enumerations

Name	Description
ST_GearInPosCAOptions [▶ 46]	Options for MC_GearInPosCA [▶ 41].
ST_MoveAbsoluteCAOptions [▶ 49]	Options for MC_MoveAbsoluteCA [▶ 33].
ST_MoveRelativeCAOptions [▶ 50]	Options for MC_MoveRelativeCA [▶ 36].
ST_HaltCAOptions [▶ 51]	Options for MC_HaltCA [▶ 39].
MC_GAP_CONTROL_MODE [▶ 51]	Defines the gap control mode at function block level.

9.1.1 Function Blocks

9.1.1.1 Motion

9.1.1.1.1 MC_MoveAbsoluteCA



The function block MC_MoveAbsoluteCA instructs a single axis to move to the absolute position defined in the function block, based on Collision Avoidance. Collision Avoidance has higher priority than the motion command. Therefore, the axis may slow down or wait while the motion command is executed to avoid a collision. The function block does not output the signal `Done` until the axis has reached its target position.

Inputs

```
VAR_INPUT
    Execute               : BOOL;
    ContinuousUpdate      : BOOL;
    Position              : MC_LREAL := MC_INVALID;
    Velocity              : MC_LREAL := MC_INVALID;
    Acceleration          : MC_LREAL := MC_DEFAULT;
    Deceleration          : MC_LREAL := MC_DEFAULT;
    Jerk                  : MC_LREAL := MC_DEFAULT;
    Gap                   : MC_LREAL := MC_DEFAULT;
    BufferMode            : MC_BUFFER_MODE := mcAborting;
    Direction             : Tc3_Mc3Definitions.MC_DIRECTION;
    Options               : ST_MoveAbsoluteCAOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
ContinuousUpdate	BOOL	In this version, continuous updating is only available for the `Gap`.
Position	MC_LREAL	Specified absolute target position for the command. **From TF5400 V3.1.10.30:** If positioning is performed using modulo (`Direction != mcDirectionNonModulo`), the target position must be in the `Interval[0, ModuloFaktor]`. Additional turns are commanded via the parameter `ST_MoveAbsoluteCAOptions.AdditionalTurns`. If the target position is within the `Tolerance Window`, then the `Direction = mcDirectionPositive` and `Direction = mcDirectionNegative` for the position will be ignored without additional turns. **Up to TF5400 V3.1.10.14** additional turns are commanded by commanding larger target positions than the `ModuloFaktor`. Further details in the notes to Modulo positioning [▶ 116].
Velocity	MC_LREAL	The velocity value must be greater than 0. It is automatically limited by the axis parameter 'Maximum Velocity'. A velocity must always be specified. Special input values: `MC_DEFAULT` = invalid, as there is no default velocity `MC_MAXIMUM` = corresponds to the value of the axis parameter 'Maximum Velocity'
Acceleration	MC_LREAL	The acceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Acceleration'. The input is preset with `MC_DEFAULT`. Special input values:

Name	Type	Description
		MC_DEFAULT = corresponds to the value of the axis parameter 'Default Acceleration' MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Acceleration'
Deceleration	MC_LREAL	The deceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Deceleration'. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Deceleration' MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Deceleration'
Jerk	MC_LREAL	The jerk must be ≥100. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Jerk' MC_MAXIMUM = invalid
Gap	MC_LREAL	This value determines the minimum gap to the predecessor for Collision Avoidance. If no value is entered, the default value of the group is used. **Notice When using geo-compensation, special attention must be paid to the gap. The mover gap for Collision Avoidance always relates positionally and dynamically to the offset path geometry. Since the gap refers to the offset path when using geo-compensation, adjacent movers in the curves can collide if it is set too low. Ensure the gap is large enough.**
BufferMode	MC_BUFFER_MODE	In this version only mcAborting and mcBuffered are available (see MC_BUFFER_MODE [▶ 104]).
Direction (available from V3.1.10.1)	Tc3_Mc3Definitions.MC_DIRECTION	Defines the direction of the movement (default mcDirectionNonModulo), see MC_DIRECTION [▶ 108].
Options	ST_MoveAbsoluteCAOptions	For more information about the available options (from V3.1.2.47) see the documentation for ST_MoveAbsoluteCAOptions [▶ 49].

The axis does not reach the target velocity, acceleration or deceleration

The values for velocity, acceleration or deceleration may be automatically limited to the maximum axis velocity, acceleration and deceleration. Check the parameters Maximum Dynamics and Default Dynamics of the axis. It is also possible that the values of Maximum Dynamics are smaller than the Default Dynamics.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

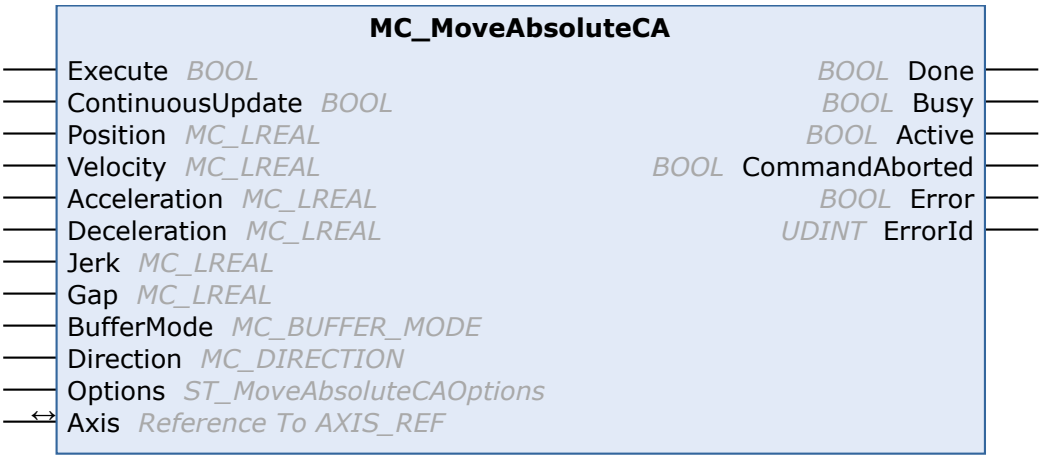
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Active              : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes `TRUE` when the command was successfully executed.
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `Done`, `CommandAborted` or `Error` is set.
Active	BOOL	If `Active` is `TRUE`, the function block controls the axis.
CommandAborted	BOOL	This output becomes `TRUE` if the command was interrupted by another command.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.1.1.2 MC_MoveRelativeCA



The function block MC_MoveRelativeCA instructs a single axis to move over the relative distance defined in the function block, based on Collision Avoidance. Collision Avoidance has higher priority than the motion command. The axis may slow down or wait while the motion command is executed to avoid a collision. The function block does not output the signal `Done` until the axis has traveled the specified distance.

Inputs

```
VAR_INPUT
    Execute               : BOOL;
    ContinuousUpdate      : BOOL;
    Distance              : MC_LREAL := MC_INVALID;
    Velocity              : MC_LREAL := MC_INVALID;
    Acceleration          : MC_LREAL := MC_DEFAULT;
    Deceleration          : MC_LREAL := MC_DEFAULT;
    Jerk                  : MC_LREAL := MC_DEFAULT;
    Gap                   : MC_LREAL := MC_DEFAULT;
    BufferMode            : MC_BUFFER_MODE := mcAborting;
    Options               : ST_MoveRelativeCAOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
ContinuousUpdate	BOOL	In this version, continuous updating is only available for the Gap.
Distance	MC_LREAL	Specified relative distance for the command.
Velocity	MC_LREAL	The velocity value must be greater than 0. It is automatically limited by the axis parameter 'Maximum Velocity'. A velocity must always be specified. Special input values: MC_DEFAULT = invalid, as there is no default velocity MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Velocity'
Acceleration	MC_LREAL	The acceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Acceleration'. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Acceleration' MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Acceleration'
Deceleration	MC_LREAL	The deceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Deceleration'. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Deceleration' MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Deceleration'
Jerk	MC_LREAL	The jerk must be ≥100. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Jerk' MC_MAXIMUM = invalid
Gap	MC_LREAL	This value determines the minimum gap to the predecessor for Collision Avoidance. If no value is entered, the default value of the group is used.

Name	Type	Description
		Notice When using geo-compensation, special attention must be paid to the gap. The mover gap for Collision Avoidance always relates positionally and dynamically to the offset path geometry. Since the gap refers to the offset path when using geo-compensation, adjacent movers in the curves can collide if it is set too low. Ensure the gap is large enough.
BufferMode	MC_BUFFER_MODE	In this version only `mcAborting` and `mcBuffered` are available (see MC_BUFFER_MODE [▶ 104]).
Options	ST_MoveRelativeCAOptions	For more information about the available options (from V3.1.2.47) see the documentation for ST_MoveRelativeCAOptions [▶ 50].

The axis does not reach the target velocity, acceleration or deceleration

The values for velocity, acceleration or deceleration may be automatically limited to the maximum axis velocity, acceleration and deceleration. Check the parameters Maximum Dynamics and Default Dynamics of the axis. It is also possible that the values of Maximum Dynamics are smaller than the Default Dynamics.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

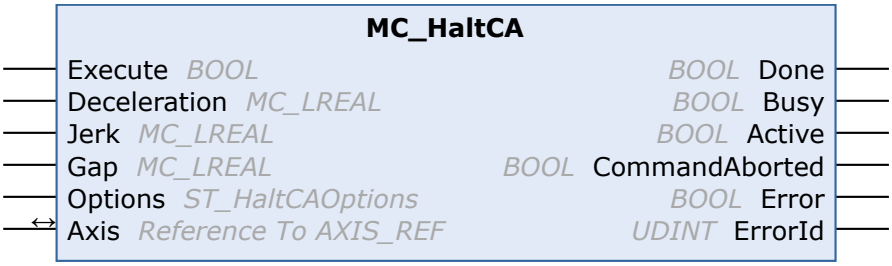
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Active              : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes `TRUE` when the command was successfully executed.
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `Done`, `CommandAborted` or `Error` is set.
Active	BOOL	If `Active` is `TRUE`, the function block controls the axis.
CommandAborted	BOOL	This output becomes `TRUE` if the command was interrupted by another command.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.1.1.3 MC_HaltCA



The function block MC_HaltCA commands a single axis to stop with Collision Avoidance.

Inputs

```
VAR_INPUT
    Execute             : BOOL;
    Deceleration        : MC_LREAL := MC_DEFAULT;
    Jerk                : MC_LREAL := MC_DEFAULT;
    Gap                 : MC_LREAL := MC_DEFAULT;
    Options             : ST_HaltCAOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
Deceleration	MC_LREAL	The deceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Deceleration'. The input is preset with `MC_DEFAULT`. Special input values: `MC_DEFAULT` = corresponds to the value of the axis parameter 'Default Deceleration' `MC_MAXIMUM` = corresponds to the value of the axis parameter 'Maximum Deceleration'
Jerk	MC_LREAL	The jerk must be ≥100. The input is preset with `MC_DEFAULT`. Special input values: `MC_DEFAULT` = corresponds to the value of the axis parameter 'Default Jerk' `MC_MAXIMUM` = invalid
Gap	MC_LREAL	This value determines the minimum gap to the predecessor for Collision Avoidance. If no value is entered, the default value of the group is used. When using geo-compensation, special attention must be paid to the gap. Since the gap refers to the offset path when using geo-compensation, adjacent movers in the curves can collide if it is set too low.
Options	ST_HaltCA Options	For more information about the available options (from V3.1.2.47) see the documentation for ST_HaltCAOptions [▶ 51].

The axis does not stop fast enough

The given deceleration could be automatically limited to the maximum axis deceleration. Check the parameters Maximum Dynamics and Default Dynamics of the axis. It is also possible that the values of Maximum Dynamics are below the Default Dynamics.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

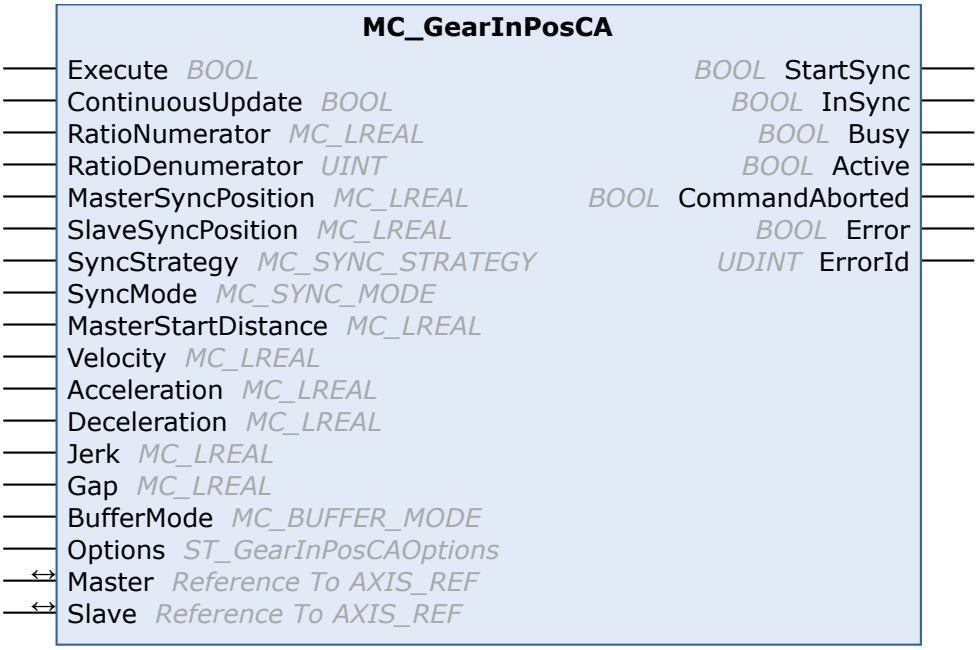
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Active              : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, Done, CommandAborted or Error is set.
Active	BOOL	If Active is TRUE, the function block controls the axis.
CommandAborted	BOOL	This output becomes TRUE if the command was interrupted by another command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.1.1.4 MC_GearInPosCA



The function block MC_GearInPosCA couples a slave axis to a master axis. The set values always form the source for the master values. Collision Avoidance has higher priority than axis coupling. The slave axis can be decoupled by sending a motion command using the buffer mode BufferMode `mcAborting`.

Optimizations regarding MC_GearInPosCA as of TF5400 v3.2.27

- Optimizations to MC_GearInPosCA that prevent SAF cycle offset between master and slave axis.
- Optimizations to the gap controller when the axis is already in the target position and only the gap changes. If the adjacent mover is commanded, the new gap takes effect.

Inputs

```
VAR_INPUT
    Execute             : BOOL;
    ContinuousUpdate    : BOOL;
    RatioNumerator      : MC_LREAL := 1;
    RatioDenumerator    : UINT := 1;
    MasterSyncPosition  : MC_LREAL := MC_INVALID;
    SlaveSyncPosition   : MC_LREAL := MC_INVALID;
    SyncStrategy        : MC_SYNC_STRATEGY := mcSyncStrategyLate;
    SyncMode            : MC_SYNC_MODE;
    MasterStartDistance : MC_LREAL := MC_IGNORE;
    Velocity            : MC_LREAL := MC_INVALID;
    Acceleration        : MC_LREAL := MC_DEFAULT;
    Deceleration        : MC_LREAL := MC_DEFAULT;
    Jerk                : MC_LREAL := MC_DEFAULT;
    Gap                 : MC_LREAL := MC_DEFAULT;
    BufferMode          : MC_BUFFER_MODE := mcAborting;
    Options             : ST_GearInPosCAOptions;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
ContinuousUpdate	BOOL	In this version, continuous updating is only available for the Gap.
RatioNumerator	MC_LREAL	A gear ratio can be set by entering integer values at the `RatioNumerator` and `RatioDenumerator` inputs or by entering a decimal value for the `RatioNumerator` and leaving the `RatioDenumerator` unchanged (the default value is `1`). The gear ratio is entered as a factor, e.g. a value of `0.8` means that the slave velocity is `0.8 * master axis velocity` (or `80 %` of the master axis velocity). The value for the factor is not limited, it could be greater than `1.0` or negative.

Name	Type	Description
RatioDenumerator	UINT	Denominator for the gear ratio.
MasterSyncPosition	MC_LREAL	This input is of type LREAL. Position of the master at which the slave is InSync and has the correct gear ratio.
SlaveSyncPosition	MC_LREAL	This input is of type LREAL. Position of the slave at which it is InSync with the correct gear ratio.
SyncStrategy	MC_SYNC_STRATEGY	Defines the strategy that the slave uses for synchronization (see MC_SYNC_STRATEGY [▶ 109]). The default strategy is mcSyncStrategyLate.
SyncMode (available from V3.1.10.1)	MC_SYNC_MODE	Defines the direction in which the SlaveSync position is to be interpreted, see MC_SYNC_MODE [▶ 109].
MasterStartDistance	MC_LREAL	If a positive value is set, the slave axis will not start synchronization until the master position is greater than or equal to the master position (MasterSyncPosition – MasterStartDistance). If a negative value is set, the synchronization will not start until the master position is less than or equal to (MasterSyncPosition – MasterStartDistance). If MasterStartDistance is not set, the slave starts synchronization as soon as the function block gives the Active signal. The exact behavior of the slave axis during the synchronization phase depends on the SyncStrategy.
Velocity	MC_LREAL	Maximum velocity of the slave axis during the synchronization phase. The velocity value must be greater than 0. It is automatically limited by the axis parameter 'Maximum Velocity' of the slave axis. A velocity must always be specified. Special input values: MC_DEFAULT = invalid, as there is no default velocity MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Velocity' of the slave axis
Acceleration	MC_LREAL	Maximum acceleration of the slave axis during the synchronization phase. The acceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Acceleration' of the slave axis. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Acceleration' of the slave axis MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Acceleration' of the slave axis
Deceleration	MC_LREAL	Maximum deceleration of the slave axis during the synchronization phase. The deceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum deceleration' of the slave axis. The input is preset with MC_DEFAULT. Special input values: MC_DEFAULT = corresponds to the value of the axis parameter 'Default Deceleration' of the slave axis MC_MAXIMUM = corresponds to the value of the axis parameter 'Maximum Deceleration' of the slave axis
Jerk	MC_LREAL	Maximum jerk of the axis during the synchronization phase. The jerk must be ≥100. The input is preset with MC_DEFAULT.

Name	Type	Description
		Special input values: `MC_DEFAULT` = corresponds to the value of the axis parameter 'Default Jerk' of the slave axis `MC_MAXIMUM` = invalid
Gap	MC_LREAL	This value determines the minimum gap to the predecessor for Collision Avoidance. If no value is entered, the default value of the group is used. **Notice When using geo-compensation, special attention must be paid to the gap. The mover gap for Collision Avoidance always relates positionally and dynamically to the offset path geometry. Since the gap refers to the offset path when using geo-compensation, adjacent movers in the curves can collide if it is set too low. Ensure the gap is large enough.**
BufferMode	MC_BUFFER_MODE	In this version only `mcAborting` and `mcBuffered` are available (see MC_BUFFER_MODE [▶ 104]).
Options	ST_GearInPosCAOptions	The `Options` can be used to influence the synchronization profile of the slave, in addition to the `SyncStrategy` (from V3.1.2.47) (see ST_GearInPosCAOptions [▶ 46]).

The axis does not reach the target velocity, acceleration or deceleration

The values for velocity, acceleration or deceleration may be automatically limited to the maximum axis velocity, acceleration and deceleration. Check the parameters Maximum Dynamics and Default Dynamics of the axis. It is also possible that the values of Maximum Dynamics are smaller than the Default Dynamics.

Inputs/outputs

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

Name	Type	Description
Master	AXIS_REF	Reference to an axis (see AXIS_REF).
Slave	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

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

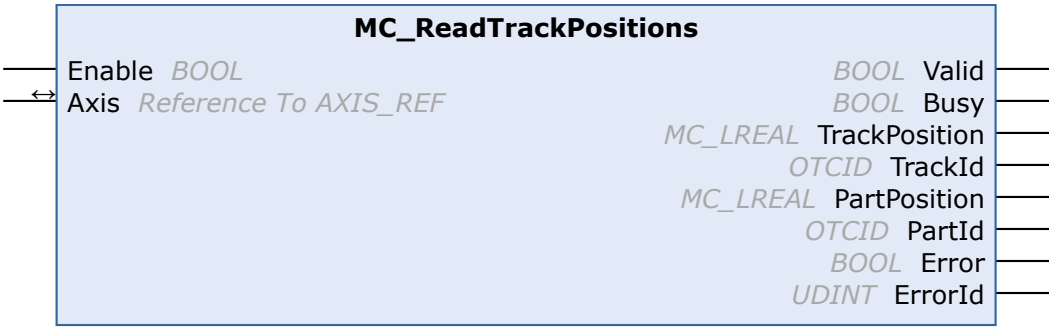
Name	Type	Description
StartSync	BOOL	This output is set when the slave actively starts synchronization and reset when the slave is InSync.
InSync	BOOL	This output becomes `TRUE` when the slave is synchronized. If the dynamics of the slave axis is too low to follow the movement of the master axis, the output `InSync` could be reset to `FALSE`, after which the slave axis starts synchronizing again.

Name	Type	Description
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `CommandAborted` or `Error` is set.
Active	BOOL	If `Active` is `TRUE`, the function block controls the axis.
CommandAborted	BOOL	This output becomes `TRUE` if the command was interrupted by another command.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.1.1.5 MC_ReadTrackPositions



The function block MC_ReadTrackPositions returns the current XTS track and XTS part target positions with the corresponding object IDs. The axis must be in a CA group for the function block to supply valid values. If no track is activated for the axis, the current absolute setpoints are returned with TrackId/PartId = 0.

Inputs

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

Name	Type	Description
Enable	BOOL	The command is executed as long as Enable is active.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

```
VAR_OUTPUT
    Valid                : BOOL;
    Busy                 : BOOL;
    TrackPosition        : LREAL;
    TrackId              : OTCID;
    PartPosition         : LREAL;
    PartId               : OTCID;
    Error                : BOOL;
    ErrorId              : UDINT;
END_VAR
```

Name	Type	Description
Valid	BOOL	This output indicates that other output values are valid for this function block.
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `Done`, `CommandAborted` or `Error` is set.
TrackPosition	LREAL	Position in the active track reference system.
TrackId	OTCID	Object ID of the active track reference system.
PartPosition	LREAL	Position on the current XTS part.
PartId	OTCID	Object ID of the current XTS part.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.1.1.6 MC_ActivateTrack

The function block MC_ActivateTrack activates a track as a reference system, which can then be used in various motion function blocks for positioning. The XTS track object must be created under the XTS Processing Unit and is then selected via the object Id. The XTS tracks are configured via the XTS Configurator (see XTS documentation for more information). ObjectID 0 can be used to reactivate the absolute reference system.

Inputs

```
VAR_INPUT
    Execute    : BOOL;
    TrackId    : OTCID;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
TrackId	OTCID	Object ID of the active track reference system.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

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

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time, one of the outputs Done, CommandAborted or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.2 Datatypes

9.1.2.1 ST_GearInPosCAOptions

The options can be set to specialize the synchronization profile of the slave.

```
TYPE ST_GearInPosCAOptions :
STRUCT
    AfterSyncVelocity               : MC_LREAL := MC_DEFAULT;
    AfterSyncAcceleration           : MC_LREAL := MC_DEFAULT;
    AfterSyncDeceleration           : MC_LREAL := MC_DEFAULT;
    AfterSyncJerk                   : MC_LREAL := MC_DEFAULT;
    MasterVelocityUndershootAllowed : BOOL := TRUE;
    MasterVelocityOvershootAllowed  : BOOL := TRUE;
    MinimalSlavePosition            : MC_LREAL := MC_IGNORE;
    DirectionReversalAllowed        : BOOL := TRUE;
    OverrideSlaveDynamicRestrictions : BOOL := FALSE;
    GapControlMode                  : MC_GAP_CONTROL_MODE := mcGapControlModeGroupDefault;
    SlaveSyncPositionReferenceSystem : OTCID := 0;
    DynamicsReferenceSystem         : OTCID := 0;
    MasterSignalCorrection          : MC_MASTER_SIGNAL_CORRECTION := mcMasterSignalCorrectionAuto;
END_STRUCT
END_TYPE
```

Name	Type	Description
AfterSyncVelocity (From TF5400 V3.1.10.1)	MC_LREAL	Maximum velocity of the slave axis after it has synchronized for the first time. The velocity value must be greater than 0. It is automatically limited by the axis parameter 'Maximum Velocity' of the slave axis. The input is preset with MC_DEFAULT. The velocity is set according to the CA-Group parameter 'GearInPosAfterSyncDynamics'.
AfterSyncAcceleration (From TF5400 V3.1.10.1)	MC_LREAL	Maximum acceleration of the slave axis after it has synchronized for the first time. The acceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum Acceleration' of the slave axis. The input is preset with MC_DEFAULT. The acceleration is set according to the CA-Group parameter 'GearInPosAfterSyncDynamics'.
AfterSyncDeceleration (From TF5400 V3.1.10.1)	MC_LREAL	Maximum deceleration of the slave axis after it has synchronized for the first time. The deceleration value must be greater than or equal to 1 and is limited by the axis parameter 'Maximum deceleration' of the slave axis. The input is preset with MC_DEFAULT. The deceleration is set according to the CA-Group parameter 'GearInPosAfterSyncDynamics'.
AfterSyncJerk (From TF5400 V3.1.10.1)	MC_LREAL	Maximum jerk of the slave axis after it has synchronized for the first time. The jerk must be ≥100. The input is preset with MC_DEFAULT. The deceleration is set according to the CA-Group parameter 'GearInPosAfterSyncDynamics'.
MasterVelocityUndershootAllowed	BOOL	This option only affects the synchronization profile and has no effect once the slave is `InSync`. `TRUE`: No restrictions for the profile `FALSE`: The slave velocity during the synchronization phase is always greater than or equal to the master velocity. If the slave velocity is lower than the master velocity at the time the command is issued, the slave accelerates with its synchronization dynamics to reach the master velocity as quickly as possible. MasterVelocityUndershootAllowed and MasterVelocityOvershootAllowed cannot both be set to FALSE.
MasterVelocityOvershootAllowed	BOOL	This option only affects the synchronization profile and has no effect once the slave is `InSync`. `TRUE`: No restrictions for the profile. `FALSE`: The slave velocity during the synchronization phase is always less than or equal to the master velocity. If the slave velocity is greater than the master velocity at the time the command is issued, the slave decelerates with its synchronization dynamics in order to reach the master velocity. MasterVelocityUndershootAllowed and MasterVelocityOvershootAllowed cannot both be set to FALSE.
MinimalSlavePosition	MC_LREAL	Absolute minimum position of the slave during the synchronization phase. This option only affects the synchronization profile and has no effect once the slave is `InSync`.
DirectionReversalAllowed	BOOL	This option only affects the synchronization profile and has no effect once the slave is `InSync`. `TRUE`: No restrictions for the profile.

Name	Type	Description
		FALSE: The direction is determined by the sign of the slave velocity in the SlaveSyncPosition (gear ratio * master velocity). The slave is not allowed to move in the opposite direction.
OverrideSlaveDynamicRestrictions	BOOL	This option only affects the synchronization profile and has no effect once the slave is InSync. It only has an effect if the strategies mcSyncStrategyLate or mcSyncStrategySlow are used. FALSE: The synchronization profile is recalculated each time the master velocity changes. An error may occur if no valid synchronization profile can be generated within the dynamic limits specified in the function block MC_GearInPosCA [▶ 41]. In particular, a noisy master signal can lead to such an error (e.g. encoder axis). Furthermore, a high load may result if the master velocity changes frequently, e.g. if the master accelerates or decelerates, or if the master signal is noisy. TRUE: The synchronization profile is not necessarily recalculated when the velocity of the master changes. Instead, the originally calculated profile is stretched or compressed. This avoids the errors described above (see FALSE). However, this could lead to violation of the dynamic limits specified in the function block MC_GearInPosCA [▶ 41] (the Maximum Axis Dynamic Limits are not violated). This option can be used for synchronization to a noisy master axis (e.g. encoder axis) and can also reduce the computing time.
GapControlMode	MC_GAP_CONTROL_MODE	See the description of the data type MC_GAP_CONTROL_MODE [▶ 51] for further information.
SlaveSyncPositionReferenceSystem (From TF5400 V3.1.6.03)	OTCID	• This input is of type OTCID and can therefore refer to a mover template. • For the position reference of a synchronized slave axis to the XTS motor path, the input SlaveSyncPositionReferenceSystem can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For the position reference of a synchronized slave axis to the path defined by a Mover Template Object, set SlaveSyncPositionReferenceSystem to its object ID. Then the position input for the synchronized slave axis is interpreted according to the offset path.
DynamicsReferenceSystem (From TF5400 V3.1.6.03)	OTCID	• This input is of type OTCID and can therefore refer to a mover template. • For the dynamic reference to the XTS motor path, the input DynamicsReferenceSystem can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For dynamic reference to the path defined by a Mover Template Object, set DynamicsReferenceSystem to its object ID. Then the dynamics of the motion profile will be constrained to the given path.
MasterSignalCorrection (From TF5400 V3.2.60)	MC_MASTER_SIGNAL_CORRECTION	• See the description of the data type MC_MASTER_SIGNAL_CORRECTION [▶ 52] for further information.

Restricting the synchronization profile could make synchronization impossible for the slave.

If synchronization is impossible, MC_GearInPosCA [▶ 41] issues an error.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.2.2 ST_MoveAbsoluteCAOptions

```
TYPE ST_MoveAbsoluteCAOptions :
STRUCT
    GapControlMode          : MC_GAP_CONTROL_MODE := mcGapControlModeGroupDefault;
    PositionReferenceSystem : OTCID := 0;
    DynamicsReferenceSystem : OTCID := 0;
    AdditionalTurns         : UDINT := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
GapControlMode	MC_GAP_CONTROL_MODE	See the description of the data type MC_GAP_CONTROL_MODE [▶ 51] for further information.
PositionReferenceSystem (as of TF5400 V3.1.6.03)	OTCID	• This input is of type `OTCID` and can therefore refer to a mover template. • For the position reference to the XTS motor path, the input `PositionReferenceSystem` can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For the position reference to the path defined by a Mover Template Object, set `PositionReferenceSystem` to its object ID. Then the position input will be interpreted according to the offset path.
DynamicsReferenceSystem (as of TF5400 V3.1.6.03)	OTCID	• This input is of type `OTCID` and can therefore refer to a mover template. • For the dynamic reference to the XTS motor path, the input `DynamicsReferenceSystem` can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For dynamic reference to the path defined by a Mover Template Object, set `DynamicsReferenceSystem` to its object ID. Then the dynamics of the motion profile will be constrained to the given path.
AdditionalTurns (as of TF5400 V3.1.10.30)	UDINT	• This input is used to command additional full turns. • `AdditionalTurns` may only be used (take a value > 0) if: ◦ `Direction = mcDirectionPositive` or ◦ `Direction = mcDirectionNegative` . • If positioning is performed using modulo, i.e. if `Direction` assumes one of the three following values {`mcDirectionPositive, mcDirectionNegative, ShortestWay`}, the target position must be located in `Interval [0, ModuloFaktor]` as of V3.1.10.30. This is a departure from previous behavior. Before the introduction of the parameter `AdditionalTurns`,

Name	Type	Description
		additional turns were commanded by commanding larger target positions than the modulo factor. **Example:** ModuloFactor = 360, StartPosition = 5; 2 full turns are to be commanded and moved to position 10: ◦ Up to V3.1.10.14: TargetPosition = 730 ◦ From V3.1.10.30: TargetPosition = 10, AdditionalTurns = 2 • Further details in the notes to Modulo positioning [▶ 116].

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.2.3 ST_MoveRelativeCAOptions

```
TYPE ST_MoveRelativeCAOptions :
STRUCT
    GapControlMode          : MC_GAP_CONTROL_MODE := mcGapControlModeGroupDefault;
    PositionReferenceSystem : OTCID := 0;
    DynamicsReferenceSystem : OTCID := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
GapControlMode	MC_GAP_CONTROL_MODE	See the description of the data type MC_GAP_CONTROL_MODE [▶ 51] for further information.
PositionReferenceSystem (From TF5400 V3.1.6.03)	OTCID	• This input is of type `OTCID` and can therefore refer to a mover template. • For the position reference to the XTS motor path, the input `PositionReferenceSystem` can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For the position reference to the path defined by a Mover Template Object, set `PositionReferenceSystem` to its object ID. Then the position input will be interpreted according to the offset path.
DynamicsReferenceSystem (From TF5400 V3.1.6.03)	OTCID	• This input is of type `OTCID` and can therefore refer to a mover template. • For the dynamic reference to the XTS motor path, the input `DynamicsReferenceSystem` can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For dynamic reference to the path defined by a Mover Template Object, set `DynamicsReferenceSystem` to its object ID. Then the dynamics of the motion profile will be constrained to the given path.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.2.4 ST_HaltCAOptions

```
TYPE ST_HaltCAOptions :
STRUCT
    GapControlMode          : MC_GAP_CONTROL_MODE := mcGapControlModeGroupDefault;
    DynamicsReferenceSystem : OTCID := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
GapControlMode	MC_GAP_CONTROL_MODE	See the description of the data type MC_GAP_CONTROL_MODE [▶ 51] for further information.
DynamicsReferenceSystem (From TF5400 V3.1.6.03)	OTCID	• This input is of type `OTCID` and can therefore refer to a mover template. • For the dynamic reference to the XTS motor path, the input `DynamicsReferenceSystem` can be left open or set to the value zero so that compatibility with earlier versions of this function block is maintained. • For dynamic reference to the path defined by a Mover Template Object, set `DynamicsReferenceSystem` to its object ID. Then the dynamics of the motion profile will be constrained to the given path.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.2.5 MC_GAP_CONTROL_MODE

The MC_GAP_CONTROL_MODE data type can be used to specify the Gap Control Mode at the function block level.

```
TYPE MC_GAP_CONTROL_MODE :
(
    mcGapControlModeGroupDefault := 16#0,
    mcGapControlModeStandard := 16#1,
    mcGapControlModeFast := 16#2
    mcGapControlModeNone := 16#3
) UDINT;
END_TYPE
```

This data type can only be used at the FB input “GapControlMode”, which exists at all Motion function blocks in Tc3_McCollisionAvoidance [▶ 33].

Name	Type	Description
mcGapControlModeGroupDefault	UDINT	This value indicates that the GapControlMode which was specified in the group parameters should be used for this motion command.
mcGapControlModeStandard	UDINT	See the description for MC_DEFAULT_GAP_CONTROL_MODE [▶ 27].

Name	Type	Description
mcGapControlModeFast	UDINT	See the description for MC_DEFAULT_GAP_CONTROL_MODE [▶ 27].
mcGapControlModeNone	UDINT	This value indicates that the Gap Control is not active in the command. After the command, the Standby Gap Control takes effect again with the mode, which is set in the group and the gap size of the last valid command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.1.2.6 MC_MASTER_SIGNAL_CORRECTION

The data type MC_MASTER_SIGNAL_CORRECTION is used in the function block MC_GearInPosCA [▶ 41] via the structure ST_GearInPosCAOptions [▶ 46] to define whether and how the input signal of the master is to be corrected.

```
TYPE MC_MASTER_SIGNAL_CORRECTION :
(
    mcMasterSignalCorrectionAuto := 0,
    mcMasterSignalCorrectionNone := 1
)INT;
END_TYPE
```

Name	Type	Description
mcMasterSignalCorrectionAuto	INT	Automatic correction of the master signal.
mcMasterSignalCorrectionNone	INT	No correction of the master signal.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.40 TF5400 Advanced Motion Pack V3.2.60	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.2 Tc3_McCompensations

What do setpoints refer to?

Setpoints always refer to the XTS motor path, because it is the motor that physically has to be moved. Consequently, a motor movement leads to a target position on the XTS motor path and thus on the path of the tool center point.

Even if the setpoints for the path dynamics are applied to the XTS motor path, they can be calculated for the dynamic control of the path of the tool center point. Accordingly, the y offset depends on the desired application and may be different for different applications. For example, the control of the center of gravity dynamics or the improvement of the performance of a tool mounted on a mover could be intended. In particular, a different tool size may require a different mover template. When the path for the tool center point is selected, setpoints are calculated to control it dynamically.

Coordinate system of the XTS motor path

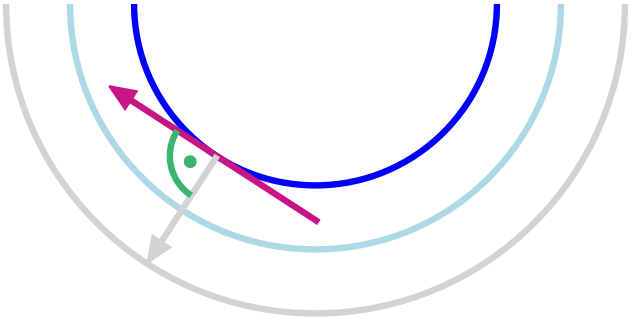
The origin of the coordinate system of the mover's motor path is located on the mover's motor path at the offset x value defined in the XTS standard object.

Coordinate system of the Tool Center Point

The origin of the Tool Center Point coordinate system is at the Tool Center Point.

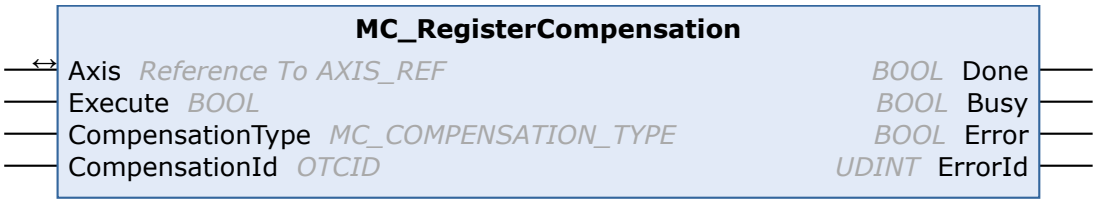
XTS motor path to Tool Center Point: Understanding the coordinate transformation

The coordinate transformation from the motor path of the mover to the Tool Center Point is always perpendicular to the motor path of the mover. Ideally and theoretically, the scalar product of the vector describing the translation of the motor path of the mover to the Tool Center Point and the corresponding vector of the tangent of the motor path of the mover has the value zero.



9.2.1 Function Blocks

9.2.1.1 MC_RegisterCompensation



Reference to a mover template: An axis refers to a mover template via the function block `MC_RegisterCompensation`.

- This function block selects the compensation type.
- This function block influences axis behavior.

Inputs

```
VAR_INPUT
    Execute          : BOOL;
    CompensationType : MC_COMPENSATION_TYPE;
    CompensationId   : OTCID;
END_VAR
```

Name	Type	Description
Execute	BOOL	This function block activates the selected compensation type when a rising edge is triggered at its `Execute` input. When activated, the geometry information for geo-compensation is taken into account by the Motion function blocks that refer to `Axis`.
CompensationType	MC_COMPENSATION_TYPE	Select `mcTypeGeoCompensation` for geo-compensation (see MC_COMPENSATION_TYPE [▶ 107]).
CompensationId	OTCID	This input `CompensationId` is of type `OTCID` and can therefore refer to a mover template. The reference to the geometry information required for geo-compensation is made via the object ID `CompensationId`, which refers to a mover template.

Inputs/outputs

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

Name	Type	Description
Axis	AXIS_REF	The input Axis is of type AXIS_REF and refers to an axis, e.g. to a mover.

Outputs

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

Name	Type	Description
Done	BOOL	This output becomes `TRUE` when the command was successfully executed.
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `Done` or `Error` is set.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4022.25 TF5400 Advanced Motion Pack V3.1.6.03	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.2.1.2 MC_TransformPosition

- This function block calculates a coordinate transformation.
- A position specified in the origin coordinate system is returned in the target coordinate system.
- An origin mover template object refers to the origin coordinate system.
- A destination mover template object refers to the target coordinate system.
- Object ID 0, `Oid = 0`, refers to the absolute coordinate system.
- The origin mover template object can refer to the absolute coordinate system, and the destination mover template object can refer to the coordinate system of the Tool Center Point: thus, a calculation of the coordinate transformation from the coordinate system of the Tool Center Point to the absolute coordinate system is to be performed.
- The origin mover template object can refer to the coordinate system of the Tool Center Point, and the destination mover template object can refer to the absolute coordinate system: thus, a calculation of the coordinate transformation from the coordinate system of the Tool Center Point to the absolute coordinate system is to be performed.
- For information purposes only: without effect on the setpoints.

Inputs

```
VAR_INPUT
    Execute             : BOOL;
    OriginOid           : OTCID;
    DestinationOid      : OTCID;
    OriginPosition      : MC_LREAL;
END_VAR
```

Name	Type	Description
Execute	BOOL	This function block outputs the target position when a rising edge is triggered at input `Execute`.
OriginOid	OTCID	This input refers to the origin mover template object as a coordinate system reference.
DestinationOid	OTCID	This input refers to the target mover template object as a coordinate system reference.
OriginPosition	MC_LREAL	Position value in the frame of the coordinate system to which the origin mover template object refers.

Outputs

```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
    DestinationPosition : MC_LREAL;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes `TRUE` if the command was executed and the execution was successful.
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `Done` or `Error` is set.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command.
DestinationPosition	MC_LREAL	Position value within the coordinate system to which the destination mover template object refers.

Sample

```
VAR
    fbAbsoluteToTcp         : MC_TransformPosition;
    fbTcpToAbsolute         : MC_TransformPosition;
    inputPositionAbsolute   : LREAL;
    inputPositionTcp        : LREAL;
    outputPositionTcp       : LREAL;
    outputPositionAbsolute  : LREAL;
    oidMoverTemplate        : OTCID;
END_VAR
```

```
fbAbsoluteToTcp(
    Execute                 := TRUE,
    OriginOid               := 0, //absolute
    DestinationOid          := oidMoverTemplate,
    OriginPosition          := inputPositionAbsolute,
    DestinationPosition     => outputPositionTcp
);

fbTcpToAbsolute(
    Execute                 := TRUE,
    OriginOid               := oidMoverTemplate,
    DestinationOid          := 0, //absolute
    OriginPosition          := inputPositionTcp,
    DestinationPosition     => outputPositionAbsolute
);
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4022.25 TF5400 Advanced Motion Pack V3.1.6.07	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.3 Tc3_McCoordinatedMotion

The Tc3_McCoordinatedMotion library is used for TF5410 TwinCAT 3 Motion Collision Avoidance and also for TF5420 TwinCAT 3 Motion Pick-and-Place.

Overview

Function block	Description	TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
			MC Group with Pick-and-Place	MC Group Coordinated Motion
Administrative				
MC_AddAxisToGroup [▶ 58]	Adds an axis group.	✔	✘ (✔ up to v3.2)	✔
MC_GroupDisable [▶ 61]	Disables an axis group.	✔	✘ (✔ up to v3.2)	✔
MC_GroupEnable [▶ 62]	Enables an axis group.	✔	✘ (✔ up to v3.2)	✔
MC_GroupReadError [▶ 63]	Reads the error ID of a group.	✔	✘ (✔ up to v3.2)	✔
MC_GroupReadStatus [▶ 65]	Reads the group status.	✔	✘ (✔ up to v3.2)	✔
MC_GroupReset [▶ 66]	Resets a group.	✔	✘ (✔ up to v3.2)	✔
MC_GroupSetOverride [▶ 68]	Sets the override of a group and returns the actual override value.	✘	✘ (✔ up to v3.2)	✔
MC_RemoveAxisFromGroup [▶ 69]	Removes an axis from a group.	✔	✘ (✔ up to v3.2)	✔
MC_SetCoordinateTransform [▶ 71]	Activates a reference system.	✘	✘ (✔ up to v3.2)	✔

Function block	Description	TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
			MC Group with Pick-and-Place	MC Group Coordinated Motion
MC_SetCoordinateTransformPreparation [▶ 72]	Adds a change of reference system to the list of movement segments.	✗	✗	✓
MC_TrackConveyorBelt [▶ 73]	Assists in synchronizing velocity to an object moving along a straight line through space.	✗	✗ (✓ up to v3.2)	✓
MC_UngroupAllAxes [▶ 76]	Disables a group and removes all axes.	✓	✗ (✓ up to v3.2)	✓
UDINT_TO_IDENTINGROUP [▶ 77]	Converts an integer value to IDENT_IN_GROUP_REF, so axes without special interpretation can be added to a group.	✓	✗	✓
Motion				
MC_GroupHalt [▶ 78]	Stops a group without locking it for further motion commands.	✓	✗	✓
MC_GroupStop [▶ 80]	Stops a group and locks it for further motion commands.	✓	✗ (✓ up to v3.2)	✓
MC_MoveLinearAbsolutePreparation [▶ 82]	Adds an absolute linear movement to a list of motion segments.	✗	✗ (✓ up to v3.2)	✓
MC_MoveCircularAbsolutePreparation [▶ 84]	Adds an absolute circular movement to a list of motion segments.	✗	✗ (✓ up to v3.2)	✓
MC_MovePath [▶ 88]	Executes a list of motion segments.	✗	✗ (✓ up to v3.2)	✓
MC_BlockerPreparation [▶ 89]	Appends a blocking job to the list of segments in the structure PathData.	✗	✗	✓
MC_ReleaseBlocker [▶ 91]	Resolves a blocking job that is blocking further execution of the path.	✗	✗	✓
MC_GroupReadBlockerStatus [▶ 92]	Reads the current blocker status.	✗	✗	✓
MC_DwellTimePreparation [▶ 93]	Appends a standstill job with a defined time to the list of segments in the structure PathData.	✗	✗	✓

Structures and enumerations

Function block	Description	TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
			MC Group with Pick-and-Place	MC Group Coordinated Motion
IDENT_IN_GROUP_REF [▶ 94]	Defines how an axis is interpreted in a group.	✗	✗ (✓ up to v3.2)	✓
MC_CIRC_MODE [▶ 95]	The circle mode defines which definition is used to program a circle.	✗	✗ (✓ up to v3.2)	✓
MC_CIRC_PATHCHOICE [▶ 99]	The data type defines the rotation direction of a circle.	✗	✗ (✓ up to v3.2)	✓
MC_PATH_DATA_REF [▶ 100]	Represents the path to be executed at MC_MovePath [▶ 88].	✗	✗ (✓ up to v3.2)	✓
ClearPath [▶ 101]	Resets the path represented by MC_PATH_DATA_REF [▶ 100].	✗	✗ (✓ up to v3.2)	✓
MC_TRANSITION_MODE [▶ 101]	Characterizes the way a segment transition is executed.	✗	✗ (✓ up to v3.2)	✓
MC_COORD_REF [▶ 103]	Object ID of a coordinate system.	✗	✗ (✓ up to v3.2)	✓

9.3.1 Function Blocks

9.3.1.1 Administrative

9.3.1.1.1 MC_AddAxisToGroup

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

The function block MC_AddAxisToGroup adds an axis to a group.

From V3.1.10.1, stationary axes can be added to and removed from a **CA group** in the GroupMoving group state. If a moving axis is added to a group, the command is rejected with an error message (a change of the group state with a moving axis is also rejected).

Only axes in GroupDisabled or GroupErrorDisabled state can be added to a **MC group**.

Inputs

```
VAR_INPUT
    Execute             : BOOL;
    IdentInGroup        : IDENT_IN_GROUP_REF;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
IdentInGroup	IDENT_IN_GROUP_REF	Defines the interpretation of the axis to be added to the group. For multi-dimensional motions, this can be the Cartesian interpretation. The global variables [▶ 94] (e.g. MCS_X) must be used. For Collision Avoidance the function UDINT_TO_IDENTINGROUP [▶ 77] must be used. **Notice The use of integer values for the input IdentInGroup is NOT supported and may lead to incompatibility with future releases. If integer values are used, it may no longer be possible to build the project. We recommend using global variables (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP.**

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup           : AXES_GROUP_REF;
    Axis                : AXIS_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface [▶ 114]).
Axis	AXIS_REF	Reference to an axis (see AXIS_REF).

Outputs

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

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.

Name	Type	Description
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Sample for TwinCAT 3 Motion Pick-and-Place

Multi-dimensional movements

Multi-dimensional movements are only applicable when TF5420 is used.

```
VAR_GLOBAL CONSTANT
cAxesCount           : UINT := 4;
END_VAR
VAR
    stGroupRef            : AXES_GROUP_REF; // link to MC Group
    stAxis                : ARRAY[1..cAxesCount] OF AXIS_REF;
    fbAddAxis             : ARRAY[1..cAxesCount] OF MC_AddAxisToGroup;
    i                     : UINT;
END_VAR
```

```
fbAddAxis[1].IdentInGroup := MCS_X; //X-Axis
fbAddAxis[2].IdentInGroup := MCS_Y; //Y-Axis
fbAddAxis[3].IdentInGroup := MCS_Z; //Z-Axis
fbAddAxis[4].IdentInGroup := MCS_C1;//1st rotation is C-rotation (around Z-Axis)

FOR i:=1 TO cAxesCount DO
    fbAddAxis[i](
    AxesGroup:=stGroupRef,
    Axis := stAxis[i],
    Execute := TRUE);
END_FOR
```

Sample for TF5410 TwinCAT 3 Motion Collision Avoidance

PTP with Collision Avoidance

PTP with Collision Avoidance is only applicable when TF5410 is used.

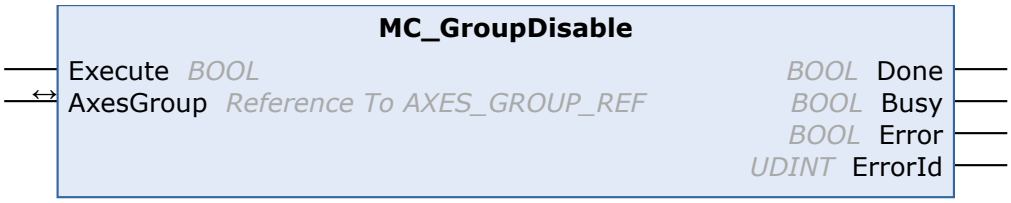
```
VAR_GLOBAL CONSTANT
    cAxesCount            : UDINT:=10;
END_VAR
VAR
    stGroupRef            : AXES_GROUP_REF; // link to CA Group
    stAxis                : ARRAY[1..cAxesCount] OF AXIS_REF;
    fbAddAxis             : ARRAY[1..cAxesCount] OF MC_AddAxisToGroup;
    i                     : UDINT;
END_VAR
```

```
FOR i:=1 TO cAxesCount DO
    fbAddAxis[i](
        AxesGroup:=stGroupRef,
        Axis := stAxis[i],
        IdentInGroup := UDINT_TO_IDENTINGROUP(i),
        Execute := TRUE);
END_FOR
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.2 MC_GroupDisable



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

The function block MC_GroupDisable disables the group. After successful execution the group changes into group state GroupDisabled (see State diagrams [▶ 23]).

NOTICE

Disabling a Moving Group results in an instant stop.

The sudden stopping of axes is likely to exceed its allowed deceleration limits. Depending on the drive hardware, this could lead to current peaks and runtime errors.
Before executing MC_GroupDisable, use MC_GroupHalt [▶ 78] or MC_GroupStop [▶ 80] to avoid this situation.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

```
VAR_OUTPUT
    Done                 : BOOL;
    Busy                 : BOOL;
```

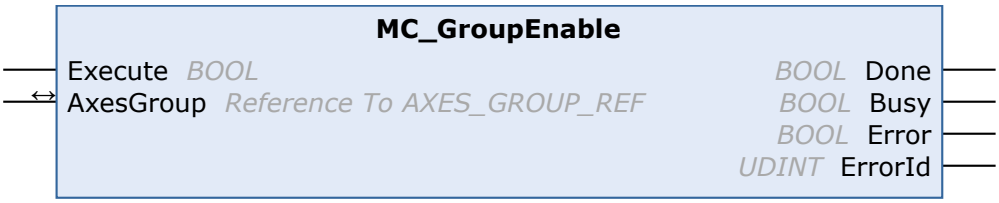
```
    Error                : BOOL;
    ErrorId              : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.3 MC_GroupEnable



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✓	✗ (✓ up to and including v3.2)	✓

The function block MC_GroupEnable enables the group. If it succeeds and all axes are ready, the group is then in group state GroupStandby (see State diagrams [▶ 23]).

An **MC Group** can only be enabled, if all axes were added to the group before.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup  : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic group interface [▶ 114])

Outputs

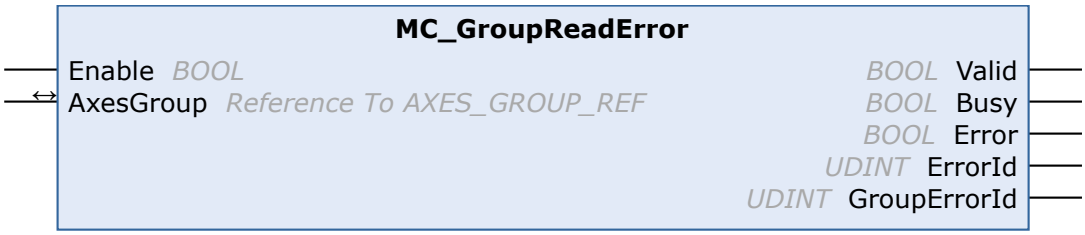
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.4 MC_GroupReadError



The function block MC_GroupReadError returns the error code for the group. It does not return any errors for function blocks (e.g. invalid parameterization).

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

Inputs

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

Name	Type	Description
Enable	BOOL	The command is executed as long as Enable is active.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

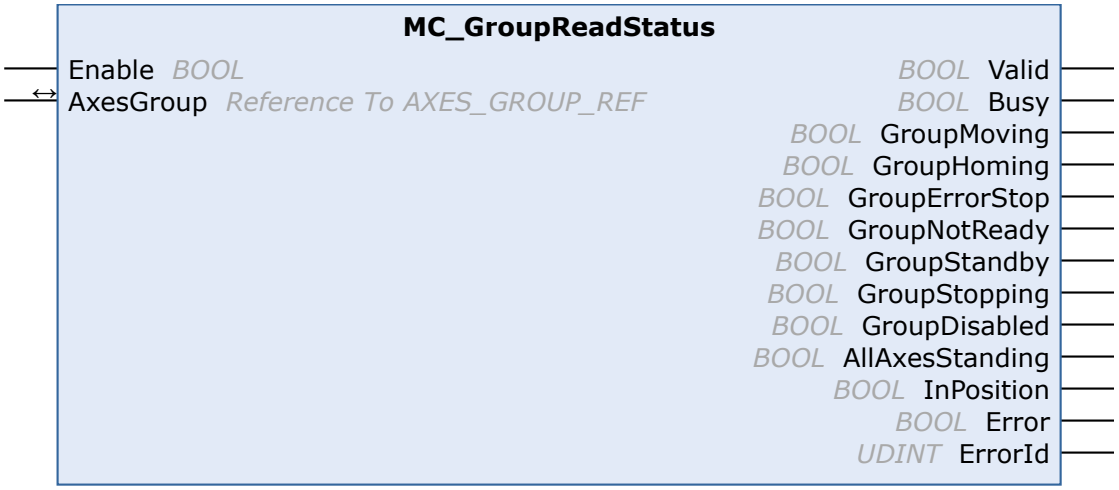
```
VAR_OUTPUT
    Valid        : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorId      : UDINT;
    GroupErrorId : UDINT;
END_VAR
```

Name	Type	Description
Valid	BOOL	This output indicates that other output values are valid for this function block.
Busy	BOOL	This output becomes TRUE when the command is started with Enable and remains so as long as the function block executes the command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).
GroupErrorId	UDINT	Returns the error ID of the group (see NC error documentation).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.5 MC_GroupReadStatus



The function block MC_GroupReadStatus reads the status of an axis group (see State diagrams [▶ 23]).

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

Inputs

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

Name	Type	Description
Enable	BOOL	The command is executed as long as Enable is active.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup            : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

```
VAR_OUTPUT
    Valid                : BOOL;
    Busy                 : BOOL;
    GroupMoving          : BOOL;
    GroupHoming          : BOOL;
    GroupErrorStop       : BOOL;
    GroupNotReady        : BOOL;
    GroupStandby         : BOOL;
    GroupStopping        : BOOL;
    GroupDisabled        : BOOL;
    AllAxesStanding      : BOOL;
    ConstantVelocity     : BOOL; // hidden
    Accelerating         : BOOL; // hidden
    Decelerating         : BOOL; // hidden
    InPosition           : BOOL;
```

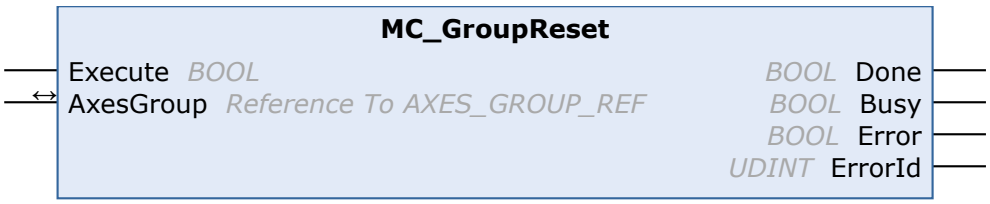
```
    Error                : BOOL;
    ErrorId              : UDINT;
END_VAR
```

Name	Type	Description
Valid	BOOL	This output indicates that other output values are valid for this function block.
Busy	BOOL	This output becomes TRUE when the command is started with Enable and remains so as long as the function block executes the command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).
GroupMoving	BOOL	The group is in the GroupMoving state (see State diagrams [▶ 23]).
GroupHoming	BOOL	The group is in the GroupHoming state (see State diagrams [▶ 23]).
GroupErrorStop	BOOL	The group is in the GroupErrorStop state (see State diagrams [▶ 23]).
GroupNotReady	BOOL	The group is in the GroupNotReady state (see State diagrams [▶ 23]).
GroupStandby	BOOL	The group is in the GroupStandby state (see State diagrams [▶ 23]).
GroupStopping	BOOL	The group is in the GroupStopping state (see State diagrams [▶ 23]).
GroupDisabled	BOOL	The group is in the GroupDisabled state (see State diagrams [▶ 23]).
AllAxesStanding	BOOL	None of the axes in the group move physically (velocity = 0 and acceleration = 0), regardless of whether a Motion Command exists or not.
ConstantVelocity	BOOL	Not supported. Not visible as of TF5400 3.2.27.
Accelerating	BOOL	Not supported. Not visible as of TF5400 3.2.27.
Decelerating	BOOL	Not supported. Not visible as of TF5400 3.2.27.
InPosition	BOOL	Not supported.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.6 MC_GroupReset



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

The function block MC_GroupReset resets all internal errors of a group and all axes which are part of the group. If the group was enabled when the error occurred, the group goes into state GroupStandby. If the group was disabled, it goes to state GroupDisabled (see State diagrams [▶ 23]).

If this function block is called while there is no error, it has no effect.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

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

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.7 MC_GroupSetOverride

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The function block MC_GroupSetOverride changes the override of a group. A change is made with a certain delay. An override input value is valid between 0 [0%] and 1 [100%]. If the value is set outside this range, it is automatically set to the respective limit value.

The behavior for override modifications in relation to the **MC group** can be defined as an axis group parameter, see Time Override Ramp Time.

Inputs

```
VAR_INPUT
    Enable              : BOOL;
    VelFactor           : MC_LREAL := 1.0;
END_VAR
```

Name	Type	Description
Enable	BOOL	The command is executed as long as Enable is active.
VelFactor	MC_LREAL	The override is set to this value (value range between 0 [0 %] and 1 [100 %]).

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup            : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

```
VAR_OUTPUT
    Enabled             : BOOL;
    Busy                : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
    ActualVelFactor     : UDINT;
END_VAR
```

Name	Type	Description
Enabled	BOOL	This output signals that the `VelFactor` has been set successfully. The `VelFactor` shows the type of an override factor.

Name	Type	Description
Busy	BOOL	This output becomes `TRUE` when the command is started with `Enable` and remains so as long as the function block executes the command.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).
ActualVelFactor	UDINT	Override that is currently active in the group (value range between 0 [0 %] and 1 [100 %]).

Sample

```
VAR
    stGroupRef            : AXES_GROUP_REF;
    fbSetOverride         : MC_GroupSetOverride;
END_VAR
```

```
fbSetOverride(
    AxesGroup:=stGroupRef ,
    Enable:= TRUE ,
    VelFactor:=1.0 , (* 1.0 = 100% *)
);
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.8 MC_RemoveAxisFromGroup

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

The function block MC_RemoveAxisFromGroup removes an axis from the axis group.

From TF5400 V3.1.10.1, stationary axes can be added to and removed from a **CA group** in the GroupMoving group state. If a moving axis is added to a group, the command is rejected with an error message (a change of the group state with a moving axis is also rejected).

Axes can only be added to an **MC group** if EnableRequested is FALSE, e.g. in the GroupDisabled state.

Success of the function block

The function block always returns DONE if the axis no longer belongs to the group. This means that DONE is returned even if the axis was not in the group before the function block was called.

Inputs

```
VAR_INPUT
    Execute               : BOOL;
    IdentInGroup          : IDENT_IN_GROUP_REF;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
IdentInGroup	IDENT_IN_GROUP_REF	Defines the interpretation of the axis to be added to the group. For multidimensional motions, this can be the Cartesian interpretation. The global variables (e.g. MCS_X) must be used. For Collision Avoidance the function UDINT_TO_IDENTINGROUP must be used.

Use of integer values for the input IdentInGroup

The use of integer values for the input IdentInGroup is NOT supported and may lead to incompatibility with future releases. If integer values are used, it may no longer be possible to build the project. We recommend using global variables [▶ 94] (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP [▶ 77].

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

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

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.9 MC_SetCoordinateTransform

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The function block MC_SetCoordinateTransform activates a coordinate transformation for successor movements. The successful activation is indicated by `Active` or `Done`.

Decouples the successor movements from a transport system (see MC_TrackConveyorBelt [▶ 73]).

Successor movements (e.g. MC_MovePath [▶ 88]) take place relative to the coordinate transformation.

Use case for changing the reference system

The `MC` group can be decoupled by using `MC_SetCoordinateTransform` and changing the reference system.

Inputs

```
VAR_INPUT
    Execute        : BOOL;
    CoordTransform : MC_COORD_REF;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
CoordTransform	MC_COORD_REF	Reference to a coordinate system (see MC_COORD_REF [▶ 103]).

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup      : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface [▶ 114]).

Outputs

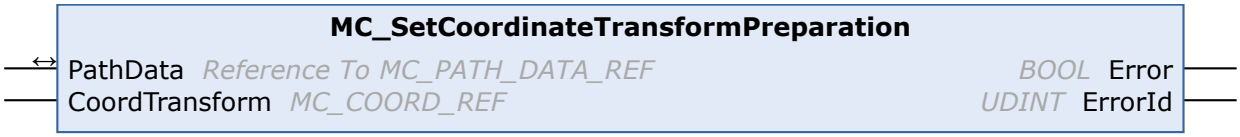
```
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes `TRUE` when the command was successfully executed.
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `Busy` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `Done`, `CommandAborted` or `Error` is set.
Active	BOOL	`Active` indicates that the command is being executed. `Active` indicates that the reference system has been successfully set (MC Coordinated Motion Group only). `Active` indicates a delay in conveyor tracking (MC Coordinated Motion Group only). `Active` becomes `FALSE` if one of the outputs `Done`, `CommandAborted` or `Error` is set to `TRUE`. Note: According to the PLCopen definition, `Active` is reset if `Done` is set to `TRUE`. In the event of an insignificant or non-existent delay, `Active` can only be set to `TRUE` for a short period of time. If the PLC program checks `Active`, it is therefore advisable to also check `Done`.
CommandAborted	BOOL	This output becomes `TRUE` if the command was interrupted by another command.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4022.25 TF5400 Advanced Motion Pack V3.1.6.03	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.10 MC_SetCoordinateTransformPreparation



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗	✓

The function block MC_SetCoordinateTransformPreparation inserts a change of reference system into the table containing the segments of a path.

Inputs

```
VAR_INPUT
    PathData       : MC_PATH_DATA_REF;
    CoordTransform : MC_COORD_REF;
END_VAR
```

Name	Type	Description
PathData	MC_PATH_DATA_REF [▶ 100]	Table containing the segments of a path. The table is written by the function block MC_…Preparation and executed by MC_MovePath [▶ 88].
CoordTransform	MC_COORD_REF [▶ 103]	Reference to a coordinate system.

Outputs

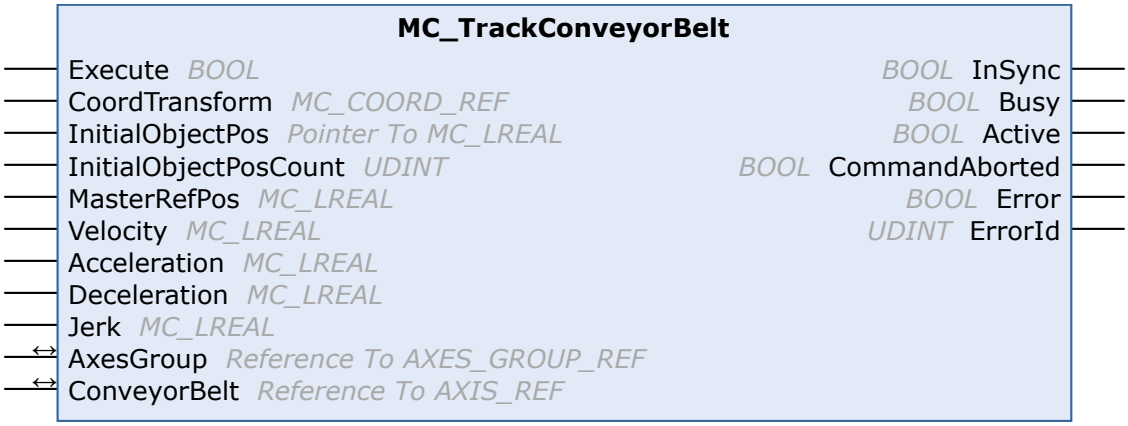
```
VAR_OUTPUT
    Error   : BOOL;
    ErrorId : UDINT;
END_VAR
```

Name	Type	Description
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC library to include
TwinCAT V3.1.4024.40 TF5400 Advanced Motion Pack V3.3.19	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.11 MC_TrackConveyorBelt



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The function block Mc_TrackConveyorBelt enables a reference system that is in motion. It synchronizes the `AxesGroup` with the `ConveyorBelt` in terms of velocity.

Synchronization with a position requires a motion command.

The function block thus helps to synchronize with an object that moves in a straight line through space. Example: products moving on a conveyor belt or other transport system.

The origin of the conveyor belt is parameterized with a coordinate system (`CoordTransform`). `X` is the conveying direction. The detected object position (`InitialObjectPos`) and the corresponding touch probe position (`MasterRefPos`) are entered in the function block.

Synchronization dynamics can be entered in the function block.

Movements executed after `Active = TRUE` are synchronized with the conveyor belt.

Execution of `MC_TrackConveyorBelt` with another instance causes direct synchronization with a second conveyor belt.

When changing the reference system, a conveyor belt can be decoupled.

Use case for changing the reference system

The `MC` group can be decoupled by using `MC_TrackConveyorBelt` and changing the reference system. The reference system can be changed with `MC_SetCoordinateTransform`.

News and optimizations regarding MC_TrackConveryorBelt with TF5400 V3.2.27 for MC Group Coordinated Motion

- New: Optionally, the override also affects the synchronization phase for the MC_TrackConveyorBelt. The setting is made in the parameter "Tracking Override Behavior" in the MC Group Coordinated Motion.
- Optimizations to the MC_TrackConveyorBelt that prevent SAF cycle misalignment between conveyor (master) and slave axis.
- Optimizations of the error reaction for the MC_TrackConveyorBelt. In the event of a runtime error of the conveyor belt (master), an active MC_MovePath is not aborted and an error reaction is to be triggered via the PLC.

Inputs

```
VAR_INPUT
    Execute               : BOOL;
    CoordTransform        : MC_COORD_REF;
    InitialObjectPos      : POINTER TO MC_LREAL;
    InitialObjectPosCount : UDINT;
    MasterRefPos          : MC_LREAL;
    Velocity              : MC_LREAL := MC_DEFAULT;
    Acceleration          : MC_LREAL := MC_DEFAULT;
    Deceleration          : MC_LREAL := MC_DEFAULT;
    Jerk                  : MC_LREAL := MC_DEFAULT;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
CoordTransform	MC_COORD_REF	Reference to a coordinate system (see MC_COORD_REF [▶ 103]).

Name	Type	Description
InitialObjectPos	POINTER TO MC_LREAL	Pointer to array `[1..InitialObjectPosCount]`.
InitialObjectPosCount	UDINT	Dimension of the `InitialObjectPos` vector.
MasterRefPos	MC_LREAL	Touch probe position.
Velocity	MC_LREAL	Velocity for synchronization. The velocity must exceed the conveyor belt velocity. The velocity is not limited by the maximum axis velocity.
Acceleration	MC_LREAL	Used in the Conveyor Tracking object. The acceleration for synchronization. The acceleration is not limited by the maximum axis acceleration. If no value is entered, then the default acceleration of the Conveyor Tracking object is used.
Deceleration	MC_LREAL	Used in the Conveyor Tracking object. The deceleration for synchronization. The deceleration is not limited by the maximum axis deceleration. If no value is entered, then the default deceleration of the Conveyor Tracking object is used.
Jerk	MC_LREAL	The jerk for synchronization. If no value is entered, then the default jerk of the Conveyor Tracking object is used. The maximum jerk is not limited.

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup           : AXES_GROUP_REF;
    ConveyorBelt        : AXIS_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface [▶ 114]).
ConveyorBelt	AXIS_REF	Reference to an axis. Reference to the conveyor axis.

Outputs

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

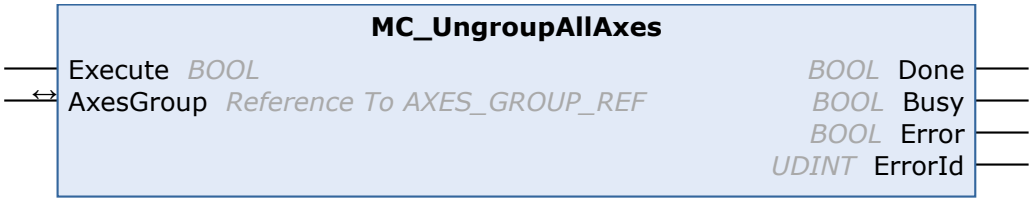
Name	Type	Description
InSync	BOOL	The output `InSync` becomes `TRUE` for the first time when the slave is synchronized with the velocity. If the slave dynamics is too low to follow the master movement, the output `InSync` could be reset to `FALSE`, after which the slave axis starts synchronizing again. **Notice Velocity synchronization: Active and InSync - the function block MC_TrackConveyorBelt synchronizes the AxesGroup with the velocity of the ConveyorBelt axis. The function block uses the given parameters for Acceleration, Deceleration and Jerk. When this synchronization movement begins, Active is set to TRUE. When the velocity of the ConveyorBelt is reached, InSync is set to TRUE. The synchronization status is continuously monitored and indicated with InSync.**

Name	Type	Description
		Notice Conveyor movement, default tracking behavior and InSync - once the output signal InSync has been set, there are two options to maintain synchronization. mcTrackingBehaviorDynLimited - this behavior is the default (MC_Default) tracking behavior. The AxesGroup maintains velocity synchronization with the ConveyorBelt using the given parameters for Acceleration, Deceleration and Jerk. – mcTrackingBehaviorStayInSync - the AxesGroup maintains the velocity synchronization with the ConveyorBelt with unlimited parameters for Acceleration, Deceleration and Jerk. **Notice Position synchronization: MasterRefPos and InitialObjectPos - the function blocks MC_TrackConveyorBelt and MC_MovePath should be used together for flexible synchronization with a moving target position. After MC_TrackConveyorBelt.Active is set to TRUE, InitialObjectPos and the distance to MasterRefPos are appended to the next call to MC_MovePath. MC_TrackConveyorBelt.InSync = TRUE and MC_MovePath.Done = TRUE indicate that the synchronized position has been reached.**
Busy	BOOL	This output becomes `TRUE` when the command is started with `Execute` and remains so as long as the function block executes the command. If `BUSY` becomes `FALSE` again, the function block is ready for a new command. At the same time, one of the outputs `CommandAborted` or `Error` is set.
Active	BOOL	If `Active` is `TRUE`, the function block controls the group.
CommandAborted	BOOL	This output becomes `TRUE` if the command was interrupted by another command.
Error	BOOL	This output becomes `TRUE` if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4022.25 TF5400 Advanced Motion Pack V3.1.6.03	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.12 MC_UngroupAllAxes



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

The function block MC_UngroupAllAxes removes all axes and disables the group. If the function block succeeds, the group is then in group state GroupDisabled (see State diagrams [▶ 23]).

Inputs

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

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

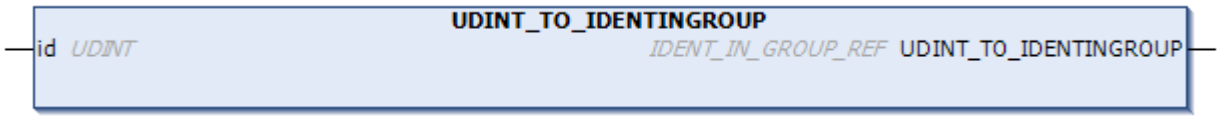
```
VAR_OUTPUT
    Done                 : BOOL;
    Busy                 : BOOL;
    Error                : BOOL;
    ErrorId              : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.1.13 UDINT_TO_IDENTINGROUP



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✘	✔

The function UDINT_TO_IDENTINGROUP is a conversion function, which converts an integer value to IDENT_IN_GROUP_REF. It is required to add a PTP axis without spatial interpretation to a CA-Group [▶ 20]. This conversion function returns a valid input for MC_AddAxisToGroup [▶ 58] and MC_RemoveAxisFromGroup [▶ 69]. For axes intended for multi-dimensional movement (TF5420) see IDENT_IN_GROUP_REF [▶ 94].

Use of integer values for the input IdentInGroup

The use of integer values for the input IdentInGroup is NOT supported and may lead to incompatibility with future releases. If integer values are used, it may no longer be possible to build the project. We recommend using global variables [▶ 94] (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP [▶ 77].

Return value

Name	Type	Description
UDINT_TO_IDENTINGROUP	IDENT_IN_GROUP_REF [▶ 94]	Converts an integer value, so a PTP axis can be added to a motion group.

Inputs

```
VAR_INPUT
    id              : UDINT;
END_VAR
```

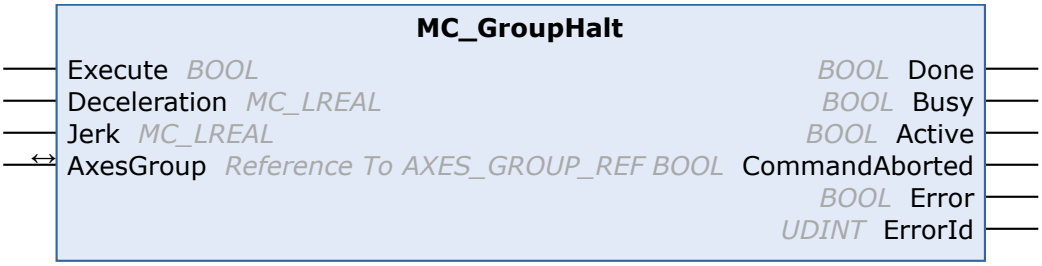
Name	Type	Description
id	UDINT	The unique identifier an axis shall have in the group. This does not have to be the axis ID from the cyclic axis interface.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2 Motion

9.3.1.2.1 MC_GroupHalt



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✘	✔

The MC_GroupHalt function block stops a group with a defined deceleration ramp. Unlike "MC_GroupStop [▶ 80]", the group is not locked for further motion commands. Therefore, the group can be restarted by another command during the deceleration ramp or after stopping.

⚠ WARNING

Possible delayed axis stop

If Standby `Gap Control` is active with a CA group and the gap is also less than the minimum, the gap is first extended before the axes can be stopped with an `MC_GroupHalt`.

- Make sure that you actually need the behavior of Standby `Gap Control`; if not, consider disabling it (default setting).
- Use an MC_GroupStop instead of an MC_GroupHalt if the axes need to be stopped without a delay.

NOTICE

MC_GroupHalt not implemented for MC group with pick-and-place

The MC_GroupHalt function block is only implemented for the MC Group Coordinated Motion and for PTP movements with Collision Avoidance (CA group). When used with another group type, the command is rejected.

Applies to the MC_Group: MC_GroupHalt cancels the active coordinate transformation and deletes all jobs in the queue.

Inputs

```
VAR_INPUT
    Execute              : BOOL;
    Deceleration         : MC_LREAL := MC_DEFAULT;
    Jerk                 : MC_LREAL := MC_DEFAULT;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
Deceleration	MC_LREAL	[mm/s²]. The deceleration can be programmed as a scalar value (≥1), or "Special input values [▶ 115]" can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values.
Jerk	MC_LREAL	[mm/s³]. The jerk can be programmed as a scalar value (≥100), or "Special input values [▶ 115]" can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values. MC_IGNORE executes the command with unlimited jerk.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

```
VAR_OUTPUT
    Done                 : BOOL;
    Busy                 : BOOL;
    Active               : BOOL;
    CommandAborted       : BOOL;
```

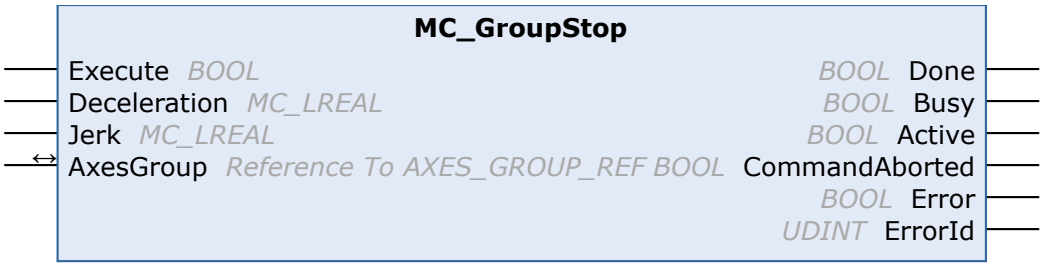
```
    Error                : BOOL;
    ErrorId              : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	Becomes TRUE when the group has been stopped and has come to a standstill. Once the group has come to a standstill, the group state becomes GroupStandby (see State diagrams [▶ 23]).
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Active	BOOL	Active indicates that the command is being executed. If the command was in the queue, it becomes active as soon as an executed command is completed.
CommandAborted	BOOL	This output becomes TRUE if the command was interrupted by another command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.2 MC_GroupStop



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖ (✔ up to and including v3.2)	✔

The function block MC_GroupStop stops the group and all associated axes with a defined deceleration ramp and locks the axis for motion commands. While the group is in the GroupStopping state, no other function block can move an axis of the group (see State diagrams [▶ 23]).

The group can only be moved again once the signal *Execute* has been set to FALSE after the velocity is 0.

MC_GroupStop cancels the active coordinate transformation and deletes all jobs in the queue.

Inputs

```
VAR_INPUT
    Execute               : BOOL;
    Deceleration          : MC_LREAL := MC_DEFAULT;
    Jerk                  : MC_LREAL := MC_DEFAULT;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
Deceleration	MC_LREAL	[mm/s²]. The deceleration can be programmed as a scalar value (≥1), or "Special input values [▶ 115]" can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values.
Jerk	MC_LREAL	[mm/s³]. The jerk can be programmed as a scalar value (≥100), or "Special input values [▶ 115]" can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values. MC_IGNORE executes the command with unlimited jerk.

Inputs/outputs

```
VAR_IN_OUT
   AxesGroup             : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic Group Interface).

Outputs

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

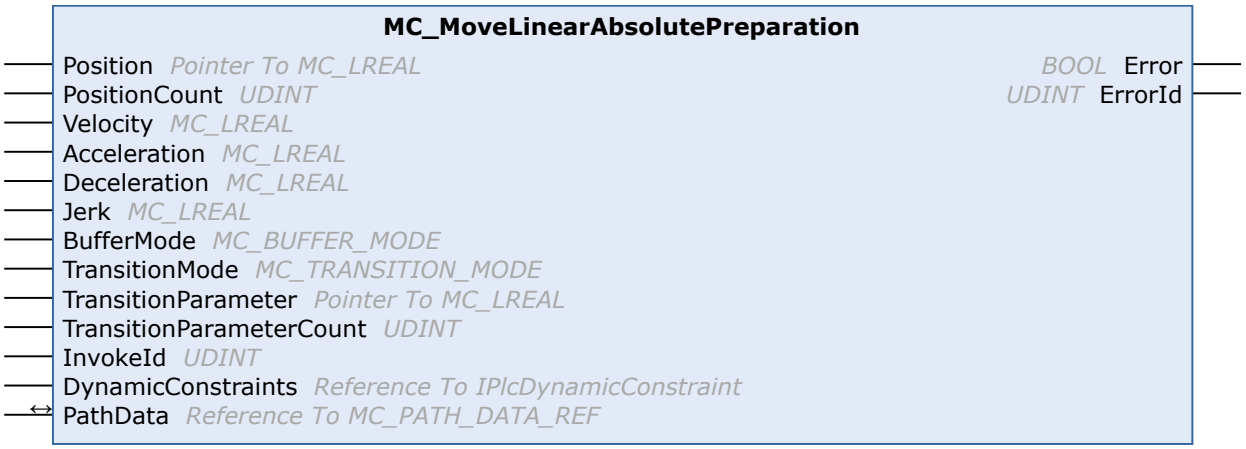
Name	Type	Description
Done	BOOL	Becomes TRUE when the group has been stopped and has come to a standstill. The group remains in the GroupStopping state while *Execute* is TRUE, at least until the axes have come to a stop. The group is then in the GroupStandby state (see State diagrams [▶ 23]).
Busy	BOOL	Becomes TRUE when the command is started with *Execute* and remains so as long as the command is executed. If *Busy* becomes FALSE again, the group is ready for a new command. After the group is stopped, *Busy* remains TRUE until the group is released with *Execute*=FALSE.
Active	BOOL	Indicates that the function block controls the group. After the group is stopped, *Active* remains TRUE until the group is released with Execute=FALSE.
CommandAborted	BOOL	The command is aborted by disabling MC_Power of at least one axis of the group or if the group is disabled during the command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.

Name	Type	Description
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.3 MC_MoveLinearAbsolutePreparation



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The function block MC_MoveLinearAbsolutePreparation adds an absolute linear movement to the table of segments in the PathData structure. After creating a table, it can be executed via MC_MovePath [▶ 88]. The function block MC_MoveLinearAbsolutePreparation can be called several times per cycle. A maximum of 30 entries are allowed per PathData table.

Inputs

```
VAR_INPUT
    Position                : POINTER TO LREAL;
    PositionCount           : UDINT;
    Velocity                : MC_LREAL := MC_INVALID;
    Acceleration            : MC_LREAL := MC_DEFAULT;
    Deceleration            : MC_LREAL := MC_DEFAULT;
    Jerk                    : MC_LREAL := MC_DEFAULT;
    BufferMode              : MC_BUFFER_MODE := mcAborting;
    TransitionMode          : MC_TRANSITION_MODE := mcTransModeNone;
    TransitionParameter     : POINTER TO LREAL;
    TransitionParameterCount : UDINT;
    InvokeId                : UDINT;
    DynamicConstraints      : REFERENCE TO IPlcDynamicConstraint := 0;
END_VAR
```

Name	Type	Description
Position	POINTER TO LREAL	Pointer to an array [1..PositionCount] of the target position vector.
PositionCount	UDINT	Dimension of the position vector. Must match the number of axes in the axis convention (see MC Group Coordinated Motion or MC Group with Pick-and-Place).
Velocity	MC_LREAL	The maximum velocity for the programmed segment. The velocity does not always have to be reached. The velocity must be set >0.
Acceleration	MC_LREAL	Maximum path acceleration for the programmed segment. Special input values [▶ 115] can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values. The acceleration must be set ≥1.
Deceleration	MC_LREAL	Maximum path deceleration for the programmed segment. Special input values [▶ 115] can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values. The deceleration must be set ≥1.
Jerk	MC_LREAL	Path jerk for the programmed segment. Special input values [▶ 115] can be used. MC_DEFAULT executes the command with default axis values. The jerk must be set ≥100. **From TF5400 V3.2.27:** MC_MAXIMUM is supported for MC Group Coordinated Motion. Here MC_MAXIMUM = 100 * MC_DEFAULT.
BufferMode	MC_BUFFER_MODE	Defines how successive motion commands are to be processed (see MC_BUFFER_MODE [▶ 104]).
Transition mode	MC_TRANSITION_MODE	Defines the blending mode (see MC_TRANSITION_MODE [▶ 101]).
TransitionParameter	POINTER TO LREAL	Pointer to array [1..TransitionParameterCount] of blending parameters. Transition parameters define the blending from the last programmed position (see MC_TRANSITION_MODE [▶ 101]).
TransitionParameterCount	UDINT	Number of blending parameters (see MC_TRANSITION_MODE [▶ 101]).
InvokeId	UDINT	Segment ID for analysis purposes.
DynamicConstraints	REFERENCE TO IPlcDynamicConstraint	**From TF5400 V3.2.27, MC Group Coordinated Motion:** Optional input to further limit the allowed values for velocity, acceleration, deceleration or jerk during motion. ***Notice*** **Reference assignment:** `<InstanceOfMC_MoveLinearAbsolutePreparation>.DynamicConstraints` **REF=** `<InstanceOfIPlcDynamicConstraint>;`

Inputs/outputs

```
VAR_IN_OUT
    PathData                : MC_PATH_DATA_REF;
END_VAR
```

Name	Type	Description
PathData	MC_PATH_DATA_REF	Table containing the segments of a path. The table is written by MC_Move…Preparation and executed by MC_MovePath [▶ 88] (see MC_PATH_DATA_REF [▶ 100]).

Resetting a table

A table is not reset during execution. To reset, the method `ClearPath()` must be called from MC_PATH_DATA_REF.

Outputs

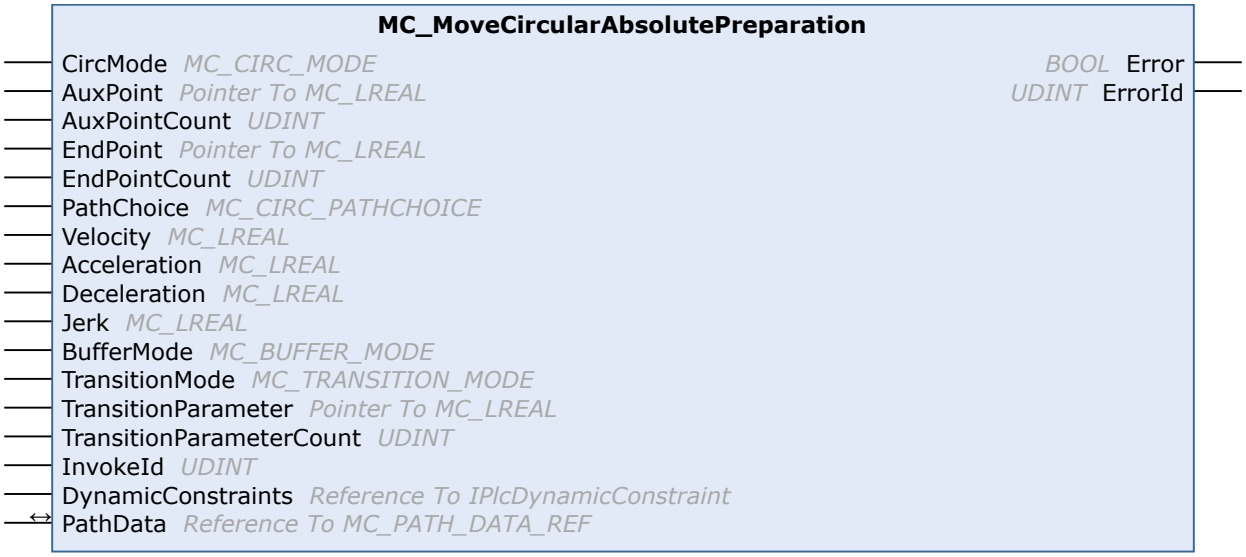
```
VAR_OUTPUT
    Error                   : BOOL;
    ErrorId                 : UDINT;
END_VAR
```

Name	Type	Description
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.4 MC_MoveCircularAbsolutePreparation



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The function block MC_MoveCircularAbsolutePreparation adds an absolute circular motion to the table of segments in the PathData structure. After creating a table, it can be executed via MC_MovePath. The function block MC_MoveCircularAbsolutePreparation can be called several times per cycle. A maximum of 30 entries are allowed per PathData table.

Resetting a table

A table is not reset during execution. To reset, the method `ClearPath()` must be called from MC_PATH_DATA_REF [▶ 100].

Inputs

```
VAR_INPUT
    CircMode                : MC_CIRC_MODE := mcCircModeInvalid;
    AuxPoint                : POINTER TO MC_LREAL;
    AuxPointCount           : UDINT;
    EndPoint                : POINTER TO MC_LREAL;
    EndPointCount           : UDINT;
    PathChoice              : MC_CIRC_PATHCHOICE := mcCircPathchoiceCounterClockwise;
    Velocity                : MC_LREAL := MC_INVALID;
    Acceleration            : MC_LREAL := MC_DEFAULT;
    Deceleration            : MC_LREAL := MC_DEFAULT;
    Jerk                    : MC_LREAL := MC_DEFAULT;
    BufferMode              : MC_BUFFER_MODE := mcAborting;
    TransitionMode          : MC_TRANSITION_MODE := mcTransModeNone;
    TransitionParameter     : POINTER TO MC_LREAL;
    TransitionParameterCount : UDINT;
    InvokeId                : UDINT;
    DynamicConstraints      : REFERENCE TO IPlcDynamicConstraint := 0;
END_VAR
```

Name	Type	Description
CircMode	MC_CIRC_MODE	Specifies which circle definition is used to program the circle. Specifies the meaning of the "AuxPoint" input signal (see MC_CIRC_MODE [▶ 95]).
AuxPoint	POINTER TO MC_LREAL	Pointer to an array [1..AuxPointCount] of the AuxPoint vector. The interpretation of the AuxPoint vector depends on the rotation convention (see MC Group Coordinated Motion or MC Group with Pick-and-Place) and is always (x, y, z).
AuxPointCount	UDINT	Dimension of the AuxPoint vector. Must be 3. If a 2D rotation convention (see MC Group Coordinated Motion or MC Group with Pick-and-Place) is used, the input value must also be 3. With a 2D rotation convention and CircMode of *mcCircModeBorder* or *mcCircModeCenter*, the component that is independent of the working plane must be set to MC_Ignore (see MC_LREAL/Special Input Values [▶ 115]).
EndPoint	POINTER TO MC_LREAL	Pointer to an array [1..EndPointCount] of the target position vector.
EndPointCount	UDINT	Dimension of the EndPoint vector. Must match the number of axes in the axis convention (see MC Group Coordinated Motion or MC Group with Pick-and-Place).
PathChoice	MC_CIRC_PATHCHOICE	Defines the direction of rotation with respect to the normal vector. The input is ignored if the input *CircMode* is set to *mcCircModeBorder* (see MC_CIRC_PATHCHOICE [▶ 99]).
Velocity	MC_LREAL	The maximum velocity for the programmed segment. The velocity does not always have to be reached. The velocity must be set >0.
Acceleration	MC_LREAL	Maximum path acceleration for the programmed segment. Special input values [▶ 115] can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values. The acceleration must be set ≥1.
Deceleration	MC_LREAL	Maximum path deceleration for the programmed segment. Special input values [▶ 115] can be used. MC_DEFAULT executes the command with default axis values. MC_MAXIMUM executes the command with the maximum axis values. The deceleration must be set ≥1.
Jerk	MC_LREAL	Path jerk for the programmed segment. Special input values [▶ 115] can be used. MC_DEFAULT executes the command with default axis values. The jerk must be set ≥100.

Name	Type	Description
		From TF5400 V3.2.27: MC_MAXIMUM is supported for MC Group Coordinated Motion. Here MC_MAXIMUM = 100 * MC_DEFAULT.
BufferMode	MC_BUFFER_MODE	Defines how successive motion commands are to be processed (see MC_BUFFER_MODE [▶ 104]).
Transition mode	MC_TRANSITION_MODE	Defines the blending mode (see MC_TRANSITION_MODE [▶ 101]).
TransitionParameter	POINTER TO MC_LREAL	Pointer to array [1..TransitionParameterCount] of blending parameters. Transition parameters define the blending from the last programmed position (see MC_TRANSITION_MODE [▶ 101]).
TransitionParameterCount	UDINT	Number of blending parameters.
InvokeId	UDINT	Segment ID for analysis purposes.
DynamicContraints	REFERENCE TO IPlcDynamicConstraint	**From TF5400 V3.2.27, MC Group Coordinated Motion:** Optional input to further limit the allowed values for velocity, acceleration, deceleration or jerk during motion. **Notice** **Reference assignment:** `<InstanceOfMC_MoveCircularAbsolutePreparation>.DynamicConstraints REF= <InstanceOfIPlcDynamicConstraint>;`

Inputs/outputs

```
VAR_IN_OUT
    PathData                : MC_PATH_DATA_REF;
END_VAR
```

Name	Type	Description
PathData	MC_PATH_DATA_REF	Table containing the segments of a path. The table is written by MC_Move…Preparation and executed by MC_MovePath [▶ 88] (see MC_PATH_DATA_REF [▶ 100]).

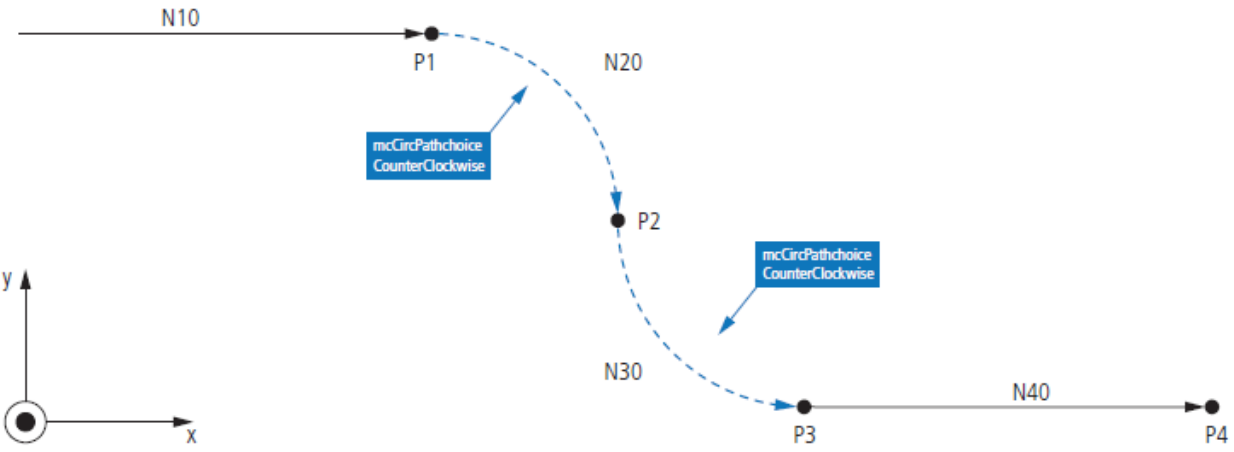
Outputs

```
VAR_OUTPUT
    Error                   : BOOL;
    ErrorId                 : UDINT;
END_VAR
```

Name	Type	Description
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Sample of center point programming

Assuming a path consisting of 4 segments as shown in the figure is to be programmed in mcCircModeCenter mode: the user defines the center of the circle as an auxiliary point ("AuxPoint"). When using mcCircModeCenter, the input MC_CIRC_PATHCHOICE [▶ 99] determines the direction of rotation. Since the plane is defined by the cross product, mcCircPathchoiceCounterClockwise must be selected for both circle segments N20 and N30.




```
VAR
    Buffer                                  : ARRAY[1..4096] OF BYTE;
    Path                                    : MC_PATH_DATA_REF (ADR(buffer), SIZEOF(buffer));
    fbMoveLinPrep                           : MC_MoveLinearAbsolutePreparation;
    fbMoveCircPrep                          : MC_MoveCircularAbsolutePreparation;

    aTargetPos                              : ARRAY[1..cAxesCount] OF MC_LREAL;
    aCircPos                                : ARRAY[1..cAxesCount] OF MC_LREAL;
    aAuxPoint                               : ARRAY[1..3] OF MC_LREAL;
    aTransitionParam                        : ARRAY[1..2] OF MC_LREAL;
END_VAR
VAR CONSTANT
    cAxesCount                              : UINT:=3;
END_VAR
```

```
    fbMoveLinPrep.Position                  := ADR(aTargetPos);
    fbMoveLinPrep.PositionCount             := cAxesCount;
    fbMoveLinPrep.TransitionParameter       := ADR(aTransitionParam);
    fbMoveLinPrep.TransitionParameterCount  := 2;
    fbMoveLinPrep.BufferMode                := mcBuffered;
    fbMoveLinPrep.TransitionMode            := mcTransModeNone;

    fbMoveCircPrep.EndPoint                 := ADR(aTargetPos);
    fbMoveCircPrep.EndPointCount            := cAxesCount;
    fbMoveCircPrep.AuxPoint                 := ADR(aAuxPoint);
    fbMoveCircPrep.AuxPointCount            := 3;
    fbMoveCircPrep.CircMode                 := mcCircModeCenter;
    fbMoveCircPrep.TransitionParameter      := ADR(aTransitionParam);
    fbMoveCircPrep.TransitionParameterCount := 2;
    fbMoveCircPrep.BufferMode               := mcBuffered;
    fbMoveCircPrep.TransitionMode           := mcTransModeNone;

    aTargetPos[1]                           := 200;
    aTargetPos[2]                           := 0;
    aTargetPos[3]                           := 0;
    aTransitionParam[1]                     := 0;
    aTransitionParam[2]                     := 0;
    fbMoveLinPrep(PathData:= path, Velocity:= 3000, InvokeId:= 10);

    aTargetPos[1]                           := 300;
    aTargetPos[2]                           := -100;
    aTargetPos[3]                           := 0;
    aAuxPoint[1]                            := 200;
    aAuxPoint[2]                            := -100;
    aAuxPoint[3]                            := 0;
    aTransitionParam[1]                     := 0;
    aTransitionParam[2]                     := 0;
    fbMoveCircPrep(PathData:= path, PathChoice:= mcCircPathchoiceCounterClockwise, Velocity:= 1000,
InvokeId:= 20);

    aTargetPos[1]                           := 400;
    aTargetPos[2]                           := -200;
    aTargetPos[3]                           := 0;
    aAuxPoint[1]                            := 400;
    aAuxPoint[2]                            := -100;
    aAuxPoint[3]                            := 0;
    aTransitionParam[1]                     := 0;
    aTransitionParam[2]                     := 0;
    fbMoveCircPrep(PathData:= path, PathChoice:= mcCircPathchoiceCounterClockwise, Velocity:= 1000,
```

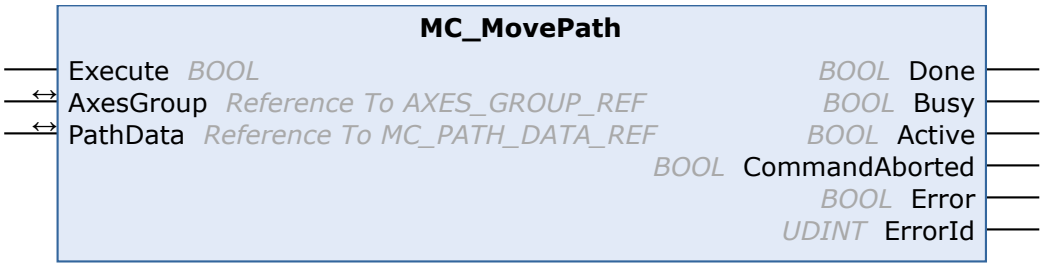
```
InvokeId:= 30);

    aTargetPos[1]                           := 600;
    aTargetPos[2]                           := -200;
    aTargetPos[3]                           := 100;
    aTransitionParam[1]                     := 0;
    aTransitionParam[2]                     := 0;
    fbMoveLinPrep(PathData:= path, Velocity:= 3000, InvokeId:= 40);
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.2.47	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.5 MC_MovePath



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The function block MC_MovePath executes a movement defined in the PathData table by MC_MoveLinearAbsolutePreparation [▶ 82], MC_MoveCircularAbsolutePreparation [▶ 84], MC_BlockerPreparation [▶ 89] and MC_SetCoordinateTransformPreparation [▶ 72].

Re-triggering of an FB instance during motion

It is possible to execute different motion commands with one instance of this function block. However, the outputs of the function block only indicate the last command executed. The user loses the ability to diagnose for the previously sent motion commands. Re-triggering of a function block is therefore not recommended.

Inputs

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

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup           : AXES_GROUP_REF;
    PathData            : MC_PATH_DATA_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to a group of axes (see Cyclic group interface [▶ 114]).
PathData	MC_PATH_DATA_REF [▶ 100]	Table containing the segments of a path. The table is written by MC_MoveLinearAbsolutePreparation [▶ 82], MC_MoveCircularAbsolutePreparation [▶ 84], MC_BlockerPreparation [▶ 89] and MC_SetCoordinateTransformPreparation [▶ 72] and executed by MC_MovePath [▶ 88].

Outputs

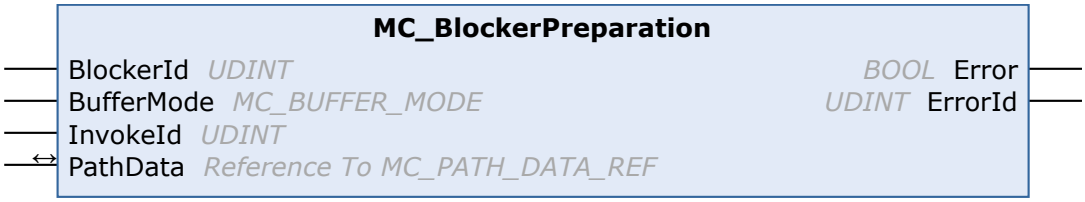
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Active              : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
END_VAR
```

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed. This means that the last command defined by the reference variable PathData was executed successfully.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Active	BOOL	If Active is TRUE, the FB controls the axis.
CommandAborted	BOOL	This output becomes TRUE if the command was interrupted by another command.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.6 MC_BlockerPreparation



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗	✓

The function block MC_BlockerPreparation appends a blocking job to the list of segments in the PathData structure. The PathData table can be executed via MC_MovePath. The function block MC_BlockerPreparation can be called several times per cycle. A maximum of 30 entries are allowed per PathData table.

A blocking job is an entry that suspends execution of the path until it is resolved with MC_ReleaseBlocker [▶ 91]. As long as the blocker is not resolved, the execution of the path is stopped at this segment. Each blocker has an Id so that the individual blockers can be distinguished in the PLC.

When a blocking job is active, the group status is still "moving".
If the override is changed while the blocking job is active, it will take effect for the next moving job.
If a new job with BufferMode mcAborting is executed while the blocking job is active, the blocking job is aborted.
If MC_GroupHalt [▶ 78] or MC_GroupStop [▶ 80] are executed while the blocking job is active, the path is terminated and the blocking job is automatically released.

Inputs

```
VAR_INPUT
    BlockerId       : UDINT;
    BufferMode      : MC_BUFFER_MODE  := mcBuffered;
    InvokeId        : UDINT;
END_VAR
```

Name	Type	Description
BlockerId	UDINT	Id of the blocker. Can be any UDINT >0.
BufferMode	MC_BUFFER_MODE	Defines how successive motion commands are to be processed (see MC_BUFFER_MODE [▶ 104]). Only mcBuffered and mcAborting are allowed here.
InvokeId	UDINT	Segment ID for analysis purposes.

Inputs/outputs

```
VAR_IN_OUT
    PathData        : MC_PATH_DATA_REF;
END_VAR
```

Name	Type	Description
PathData	MC_PATH_DATA_REF	Table containing the segments of a path. The table is written by the Preparation function blocks, like this one, and executed by MC_MovePath (see MC_PATH_DATA_REF).

Outputs

```
VAR_OUTPUT
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Name	Type	Description
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.

Name	Type	Description
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.7 MC_ReleaseBlocker

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗	✓

The function block MC_ReleaseBlocker releases a blocking job that blocks further execution of the path. A blocking job is inserted into the path with MC_BlockerPreparation [▶ 89].

With the Superpos blending strategy or, from TF5400 3.1.10.63, also with the GeoBlending strategy, the blocker can be resolved before the blocker position is reached. Blending between motion segments surrounding this blocker can be executed if those segments allow it and are still executable at the time the blocking job is released.

Inputs

```
VAR_INPUT
    Execute         : BOOL;
    BlockerId       : UDINT;
END_VAR
```

Name	Type	Description
Execute	BOOL	The command is triggered by a rising edge at this input.
BlockerId	UDINT	Id of the blocker. Can be any UDINT >0.

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup       : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to an axis group (see Cyclic Group Interface [▶ 114]).

Outputs

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

Name	Type	Description
Done	BOOL	This output becomes TRUE when the command was successfully executed.
Busy	BOOL	This output becomes TRUE when the command is started with Execute and remains so as long as the function block executes the command. If Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if available) or Error is set.
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.8 MC_GroupReadBlockerStatus

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗	✓

The function block MC_GroupReadBlockerStatus reads the current blocker status.

Inputs

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

Name	Type	Description
Enable	BOOL	Enables reading of the current blocker status.

Inputs/outputs

```
VAR_IN_OUT
    AxesGroup       : AXES_GROUP_REF;
END_VAR
```

Name	Type	Description
AxesGroup	AXES_GROUP_REF	Reference to an axis group (see Cyclic Group Interface [▶ 114]).

Outputs

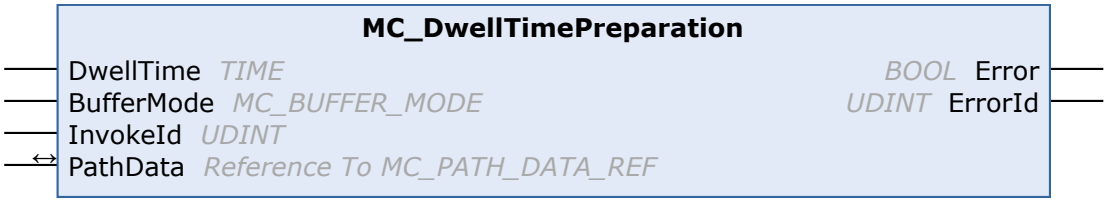
```
VAR_OUTPUT
    Valid           : BOOL;
    Blocked         : BOOL;
    BlockerId       : UDINT;
END_VAR
```

Name	Type	Description
Valid	BOOL	Returns TRUE if a valid group type is used. Only group type MC Group Coordinated Motion is allowed.
Blocked	BOOL	Returns TRUE if a blocking job is active, i.e. execution of the path is stopped. Returns FALSE if no blocking job is active.
BlockerId	UDINT	Id of the blocker. Can be any UDINT >0.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.3.1.2.9 MC_DwellTimePreparation



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗	✓

The function block MC_DwellTimePreparation appends a standstill job with a defined time to the table of segments in the PathData structure. The PathData table can be executed via MC_MovePath. The function block MC_DwellTimePreparation can be called several times per cycle.

Inputs

```
VAR_INPUT
    DwellTime       : Time;
    BufferMode      : MC_BUFFER_MODE  := mcBuffered;
    InvokeId        : UDINT;
END_VAR
```

Name	Type	Description
DwellTime	Time	Time during which the path is stationary at velocity 0. Any timespan >= 0 is allowed. A DwellTime of zero leads to an exact stop, even if the surrounding segments would allow a transition with a velocity > 0.

Name	Type	Description
BufferMode	MC_BUFFER_MODE	Defines how successive motion commands are to be processed (see MC_BUFFER_MODE [▶ 104]). Only mcBuffered and mcAborting are allowed here.
InvokeId	UDINT	Segment ID for analysis purposes.

Inputs/outputs

```
VAR_IN_OUT
    PathData        : MC_PATH_DATA_REF;
END_VAR
```

Name	Type	Description
PathData	MC_PATH_DATA_REF	Table containing the segments of a path. The table is written by the Preparation function blocks, like this one, and executed by MC_MovePath (see MC_PATH_DATA_REF).

Outputs

```
VAR_OUTPUT
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Name	Type	Description
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.
ErrorId	UDINT	Contains the command-specific error code of the last executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.3.2 Datatypes

9.3.2.1 IDENT_IN_GROUP_REF

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✖	✔	✔

IDENT_IN_GROUP_REF defines how an axis is interpreted in a group. Global variables can be used for multi-dimensional movements. For PTP collision-avoidance groups, the UDINT_TO_IDENTINGROUP [▶ 77] function must be called.

Use of integer values for the input IdentInGroup

The use of integer values for the input IdentInGroup is NOT supported and may lead to incompatibility with future releases. If integer values are used, it may no longer be possible to build the project. We recommend using global variables [▶ 94] (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP [▶ 77].

The constants below define axes as Cartesian axes in the machine coordinate system (MCS). A to C define the rotation axis (C: rotation around Z; B: rotation around Y; A: rotation around X). The number determines the rotation order. For example, if one axis is defined as MCS_C1 and another as MCS_B2, the system will first rotate around the Z-axis and second around the Y-axis

```
VAR_GLOBAL
    MCS_X                           : IDENT_IN_GROUP_REF;
    MCS_Y                           : IDENT_IN_GROUP_REF;
    MCS_Z                           : IDENT_IN_GROUP_REF;

    MCS_A1                          : IDENT_IN_GROUP_REF;
    MCS_A2                          : IDENT_IN_GROUP_REF;
    MCS_A3                          : IDENT_IN_GROUP_REF;

    MCS_B1                          : IDENT_IN_GROUP_REF;
    MCS_B2                          : IDENT_IN_GROUP_REF;
    MCS_B3                          : IDENT_IN_GROUP_REF;

    MCS_C1                          : IDENT_IN_GROUP_REF;
    MCS_C2                          : IDENT_IN_GROUP_REF;
    MCS_C3                          : IDENT_IN_GROUP_REF;

//new from TF5400 V3.1.10.1, only compatible with MC Group Coordinated Motion
    ADDAX1                          : IDENT_IN_GROUP_REF;
    ADDAX2                          : IDENT_IN_GROUP_REF;
    ADDAX3                          : IDENT_IN_GROUP_REF;
    ADDAX4                          : IDENT_IN_GROUP_REF;

// new from TF5400 V3.2.27, only compatible with MC Group
    ADDAX5                          : IDENT_IN_GROUP_REF;
    ADDAX6                          : IDENT_IN_GROUP_REF;
    ADDAX7                          : IDENT_IN_GROUP_REF;
    ADDAX8                          : IDENT_IN_GROUP_REF;
END_VAR
```

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.2.2 MC_CIRC_MODE

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The circle mode determines which circle definition is used to program a circle.

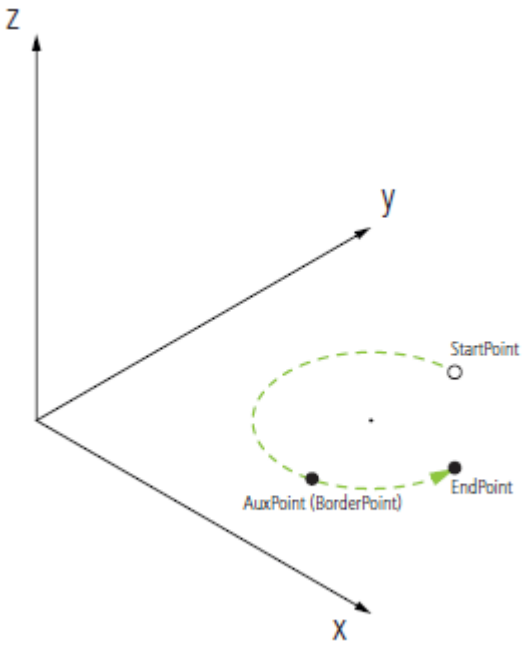
```
TYPE MC_CIRC_MODE :
(
mcCircModeInvalid        := 16#0000,
mcCircModeBorder         := 16#2000,
mcCircModeCenter         := 16#2001,
mcCircModeRadius         := 16#2002
)
END_TYPE
```

mcCircModeInvalid

Returns errors

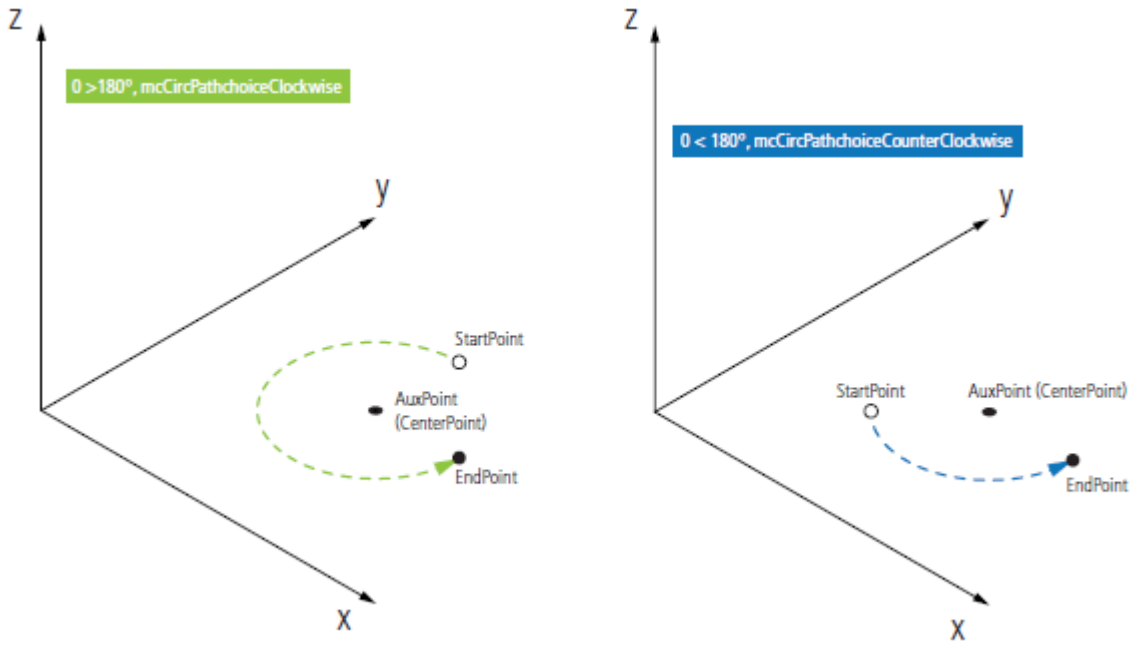
- This parameter is invalid and will lead to an error where a valid MC_CIRC_MODE argument is required.

mcCircModeBorder



StartPoint	• The movement starts at the starting point “StartPoint”. • This point is the endpoint of the preceding move command.
EndPoint	• The user configures the endpoint “EndPoint”. • The circular movement will end at this point.
AuxPoint	• The user configures the auxiliary point "AuxPoint". • The circular movement will pass through this point.
PathChoice	• The input parameter "PathChoice" and the data type "MC_CIRC_PATHCHOICE" are ignored.
Applicability	• The mode mcCircModeBorder cannot be used to describe a full circle (i.e. "StartPoint" equals "EndPoint"). This is due to the fact that in this the center point of the circle would be ambiguous. • The mode *mcCircModeBorder* cannot be used to describe paths with more than one full rotation of the circle.

mcCircModeCenter



StartPoint

- The movement starts at the starting point “StartPoint”.
- This point is the endpoint of the preceding move command.

EndPoint

- The user configures the endpoint “EndPoint”.
- The circular movement will end at this point.

AuxPoint

- The user configures the auxiliary point "AuxPoint".
- For the circular movement this auxiliary point will act as the circle center point.
- The center point is required to have the same distance from “StartPoint” and “EndPoint”. If the distances differ only slightly, the center point will be adjusted. If the distances differ significantly, the circle description will not be accepted.

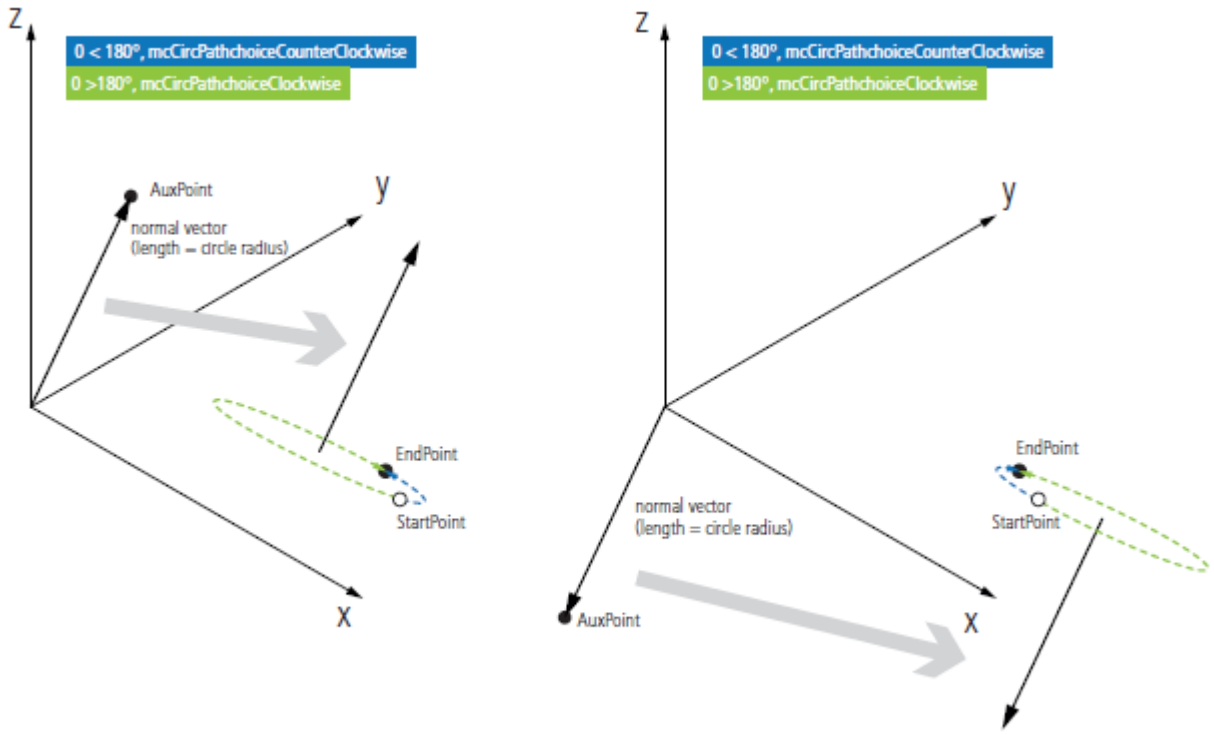
PathChoice

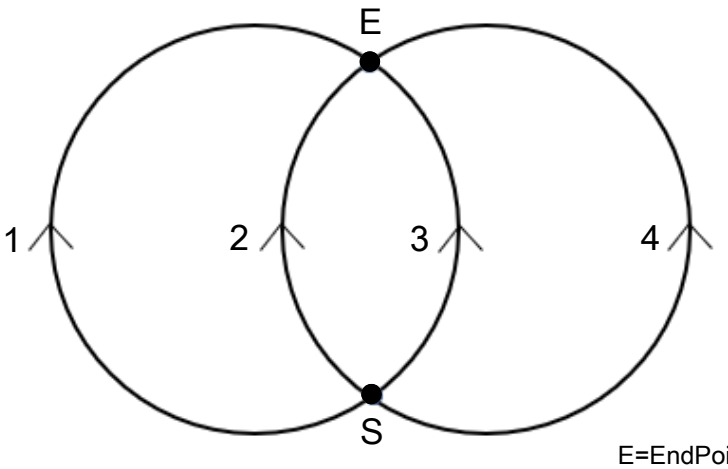
- There are usually two possible arcs of the circle that can be traversed from starting point "StartPoint" to the endpoint "EndPoint". The “PathChoice” parameter makes the two unique. See MC_CIRC_PATHCHOICE for more information.

Applicability

- The mode mcCircModeCenter cannot be used to describe a semicircle (i.e. an arc that traverses an angle of 180° or very close to that) or a full circle (i.e. "StartPoint" equals "EndPoint"). This is due to the fact that in these cases starting point, center point and endpoint would be collinear and thus the plane in which the circle lies would be ambiguous.
- The mode mcCircModeCenter cannot be used to describe paths with more than one full rotation of the circle.

mcCircModeRadius

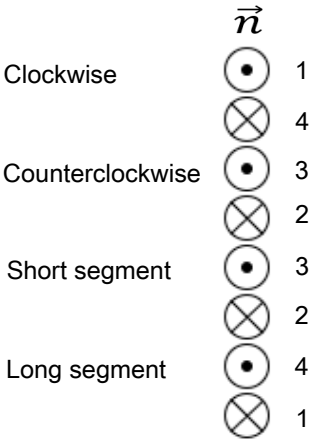




E=EndPoint

S=StartPoint

MC_CIRC_PATHCHOICE



Images

- Four different arcs are distinguished by the orientation of the normal vector and the parameter “PathChoice”.

StartPoint

- The movement starts at the starting point “StartPoint”.
- This point is the endpoint of the preceding move command.
- The circle to be constructed and its plane contain the starting point.

AuxPoint
Normal Vector

- The user configures the parameter "AuxPoint", which in this mode acts as the normal vector of the plane of the circle. Its length is taken as the radius of the circle.

EndPoint

- The user configures the endpoint “EndPoint”.
- The movement will end at this point.
- MC group with pick-and-place only: If this point is outside the plane defined by “StartPoint” and the normal vector, movement will follow a helix instead of a circle.

PathChoice and resulting arc

- The right-hand rule is applied for all “PathChoice” values except mcCircPathchoiceClockwise, which follows the left-hand rule.
- *mcCircPathchoiceCounterClockwise* and *mcCircPathchoiceShortSegment* describe an arc that covers an angle <= 180°, *mcCircPathchoiceClockwise* and *mcCircPathchoiceLongSegment* describe an arc that covers an angle >= 180°.
- Which of the 4 possible arcs with a given radius is chosen depends on the “PathChoice” argument and the orientation of the normal vector. See above table for more information.

Applicability

- The mode mcCircModeRadius can only be used to describe arcs that cover an angle < 360°.
- The length of the normal vector (i.e. the radius of the circle) must be at least half the distance between starting point and endpoint.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.2.47	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.2.3 MC_CIRC_PATHCHOICE

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

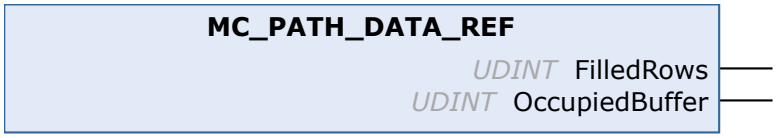
The MC_CIRC_PATHCHOICE data type defines the direction of rotation of a circle if mcCircModeCenter or mcCircModeRadius is selected from the enumeration MC_CIRC_MODE [▶ 95].

```
TYPE MC_CIRC_PATHCHOICE :
(
    mcCircPathchoiceClockwise          := 16#3000,
    mcCircPathchoiceCounterClockwise   := 16#3001

//new from TF5400 V3.1.10.1
    mcCircPathchoiceShortSegment       := 16#3002,
    mcCircPathchoiceLongSegment        := 16#3003
);
END_TYPE
```

Name	Type	Description
mcCircPathchoiceClockwise	INT	represents the circle segment with an angle >180°.
mcCircPathchoiceCounterClockwise	INT	represents the circle segment with an angle <180°.
mcCircPathchoiceShortSegment	INT	represents the circle segment with the smaller angle.
mcCircPathchoiceLongSegment	INT	represents the circle segment with the larger angle.

9.3.2.4 MC_PATH_DATA_REF



TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

MC_PATH_DATA_REF represents the path to be executed by MC_MovePath [▶ 88], whereby the number of entries is limited to 30. The path to be executed is written by MC_MoveLinearAbsolutePreparation [▶ 82], MC_MoveCircularAbsolutePreparation [▶ 84] and MC_BlockerPreparation [▶ 89]. It is initialized with a pointer to a user-defined buffer. Hereby the user can define the size of the path. The initialization has to be done during declaration. The path table is not reset at execution. To reset, the method ClearPath [▶ 101] must be called.

Outputs

```
VAR_OUTPUT
    FilledRows          : UDINT;
    OccupiedBuffer      : UDINT;
END_VAR
```

Name	Type	Description
FilledRows	UDINT	Number of path entries (e.g. path segments).
OccupiedBuffer	UDINT	Occupied buffer size in byte. By analyzing this output the user can analyze if the end of the defined buffer will be reached.

Sample

The sample below shows how to declare a path reference and how to reset an existing path.

```
VAR
    buffer          : ARRAY[1..4096] OF BYTE;
    Path            : MC_PATH_DATA_REF(ADR(buffer), SIZEOF(buffer));
END_VAR
```

```
//delete all segments of path table
Path.ClearPath();
```

The data type `MC_PATH_DATA_REF` is part of the Motion Control (MC) library. Use the method `ClearPath()` to clear path information of type `MC_PATH_DATA_REF` and thus to reset an existing path. Use only Motion Control functions or Motion Control function blocks on the data type `MC_PATH_DATA_REF`. In particular, do not use any storage functions such as `MEMCMP`, `MEMCPY`, `MEMSET` or `MEMMOVE` with data type `MC_PATH_DATA_REF`.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.2.4.1 ClearPath

ClearPath

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The method ClearPath resets the path represented by MC_PATH_DATA_REF. The path table is not reset automatically at execution.

9.3.2.5 MC_TRANSITION_MODE

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

The transition mode characterizes the way a segment transition is executed.

```
TYPE MC_TRANSITION_MODE :
(
    mcTransModeNone                         := 16#1000,
    mcTransModeStartVelocity                := 16#1001,
    mcTransModeConstantVelocity             := 16#1002,
    mcTransModeCornerDistance               := 16#1003,
    mcTransModeMaxCornerDeviation           := 16#1004,
    mcTransModeCornerDistanceAdvanced       := 16#100A
);
END_TYPE
```

The following table shows an overview of the implemented transition modes and the number of parameters that must be defined in TransitionParameterCount.

Name	TransitionParameterCount	Description
mcTransModeNone	No effect	No blending
mcTransModeCornerDistance **not compatible with MC Group with Pick-and-Place, available from TF5400 V3.1.10.1**	1	Transition parameters act as a tolerance sphere in which the path may be left.
mcTransModeCornerDistanceAdvanced	2	Transition parameters act as a tolerance sphere in which the path may be left.

mcTransModeNone

No blending is executed. Stop at segment transition.

mcTransModeCornerDistance

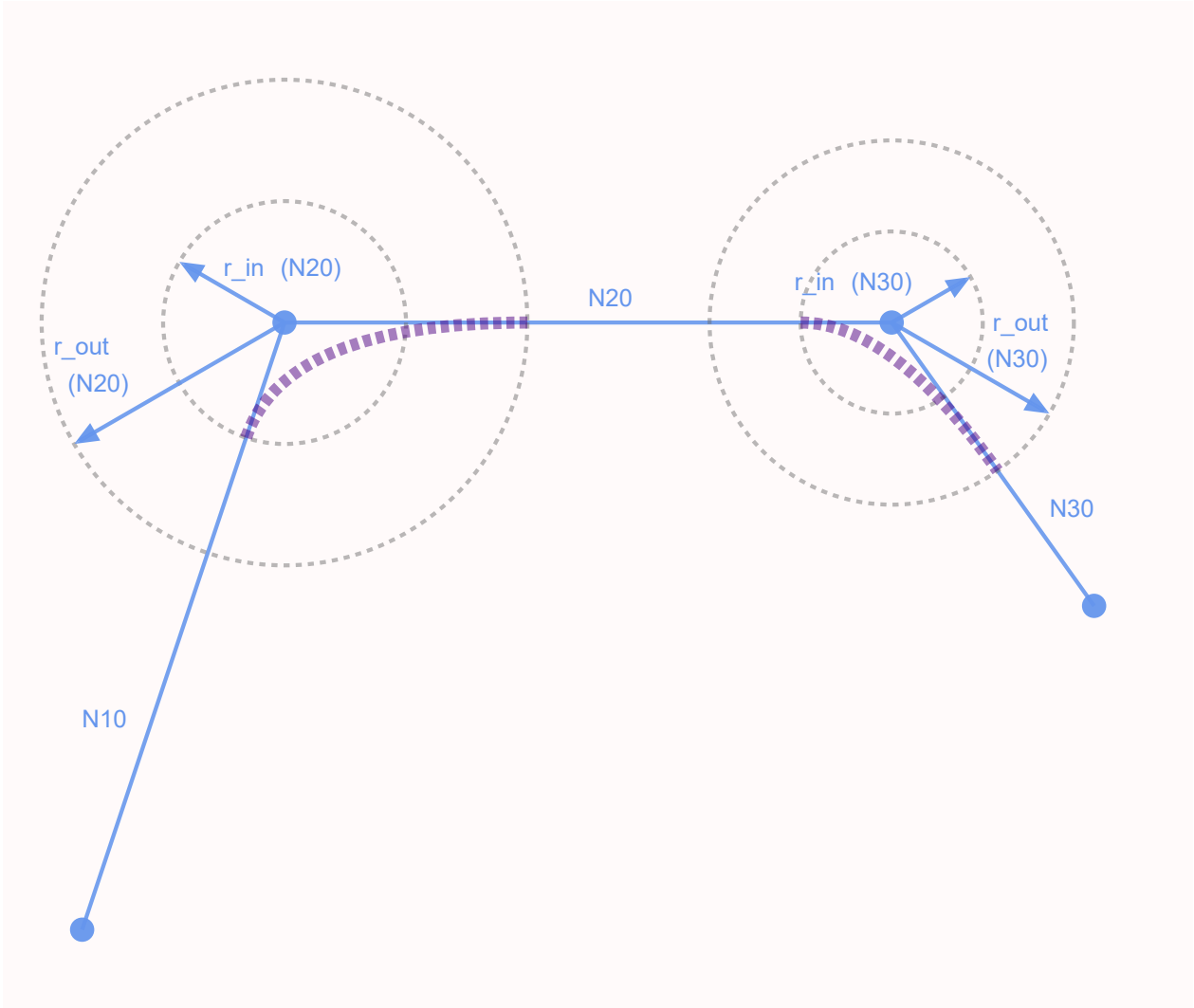
Blending is executed between the segments. The transition parameters act as tolerance ball in which the programmed path is not followed. The parameter describes the radius on the previous and second segment at which the blending starts and ends.

This mode is only compatible with MC Group Coordinated Motion.

mcTransModeCornerDistanceAdvanced

Blending is executed between the segments. The transition parameter act as tolerance ball in which the programmed path is not followed. The first parameter describes the radius on the previous segment at which the blending starts (r_in). The second parameter describes the radius on the following segment (r_out) which defines a position for which it is guaranteed that the blending is done. The parameter r_out is a maximum value. The blending can end before r_out is reached.

Blending (r_in) with MC Group with Pick-and-Place is limited to 90 % of previous segment. r_out is not limited.



Recommended Transition Parameter Relation for Blending with MC Group with Pick-and-Place

The graphics sketch a planar movement within two dimensional space. Let two axes be involved in this movement. Assuming that the involved axes exhibit similar dynamics `r_out` should measure at least `2 * r_in`.

Combinations of buffer mode and transition mode

Buffer mode and transition mode are combined only when TF5420 is used.

The following table shows the possible combinations of transition mode and buffer mode and their effect.

TM/PM	mcAborting	mcBuffered	mcBlendingPrevious	Others
mcTransModeNone	The previous command is canceled immediately. A new movement is started. The velocity in transition is 0. This combination is only permitted for the first segment of a path.	Stop at the end of the previous command. The next command is then executed.	Not permissible	Not permissible
mcTransModeCornerDistance **New from TF5400 V3.1.10.1, only compatible with MC Group Coordinated Motion**	Blending from the active segment to the first segment of the new command. The intersection of the segments is defined by the distance needed to stop on the active segment. This combination is only permitted for the first segment of a path.	Not permissible	Blending from the last programmed command to the new command	Not permissible
mcTransModeCornerDistanceAdvanced	Blending from the active segment to the first segment of the new command. The intersection of the segments is defined by the distance needed to stop on the active segment. This combination is only permitted for the first segment of a path.	Not permissible	Blending from the last programmed command to the new command	Not permissible
Others	Not permissible	Not permissible	Not permissible	Not permissible

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.3.2.6 MC_COORD_REF

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	**MC Group Coordinated Motion**
✗	✗ (✓ up to and including v3.2)	✓

Object Id that refers to a node connector.

9.4 Tc3_Mc3Definitions

Structures and enumerations

Name	Description	TF5410 Twin-CAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
			MC Group with Pick-and-Place	MC Group Coordi-nated Motion
MC_BUFFER_MODE [▶ 104]	Defines how successive travel commands are to be processed.	✔	✔	✔
MC_COMPENSATION_TYPE [▶ 107]	The value defines the compensation type.	✔	✘	✘
MC_DIRECTION [▶ 108]	The value determines the direction of the movement.	✔	✘	✘
MC_SYNC_MODE [▶ 109]	The value defines the direction in which synchronization is to be performed.	✔	✘	✘
MC_SYNC_STRATEGY [▶ 109]	Defines the synchronization profile of the slave axis.	✔	✘	✘

9.4.1 Datatypes

9.4.1.1 MC_BUFFER_MODE

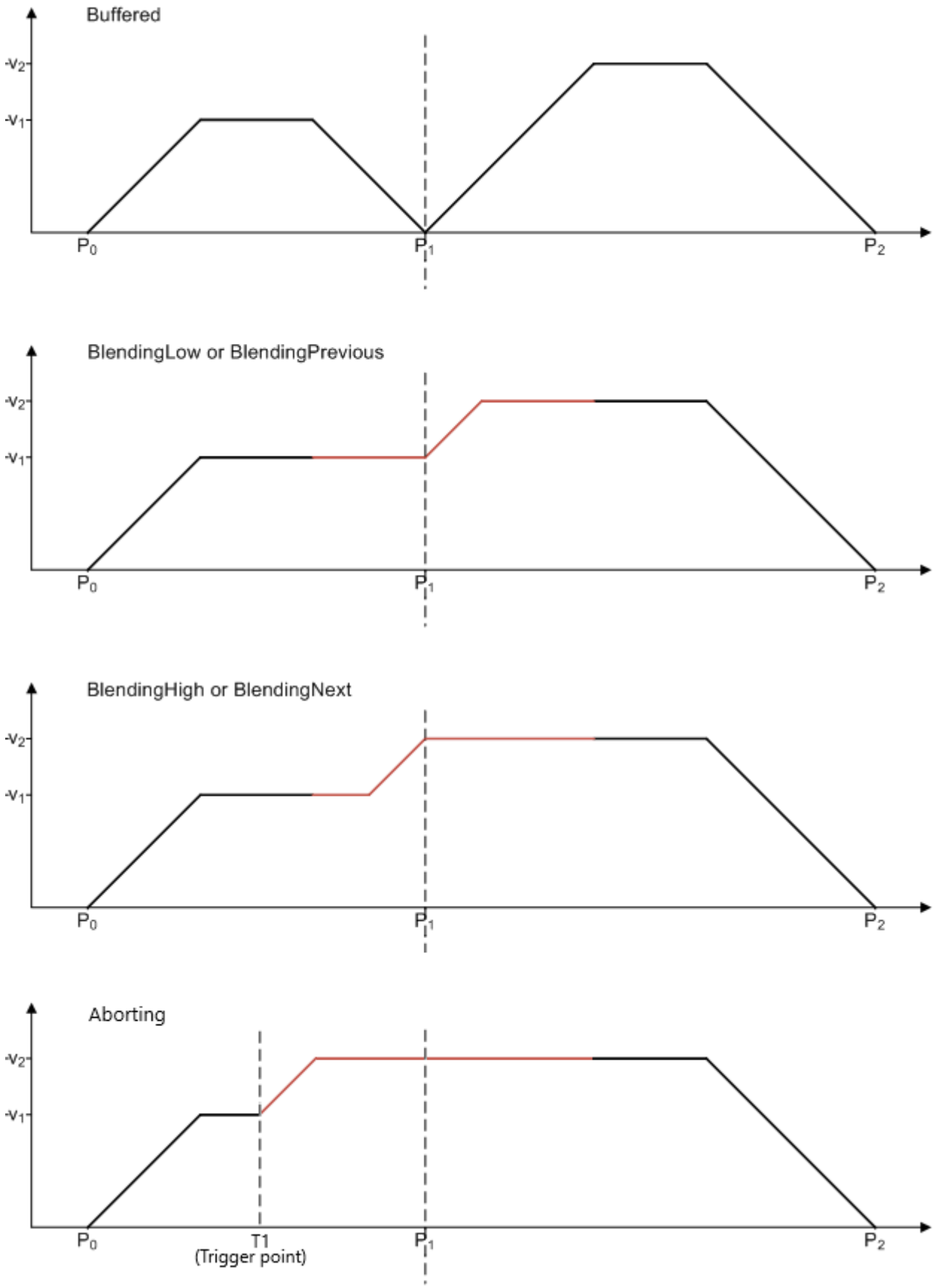
The data type MC_BUFFER_MODE is used to specify how successive travel commands are to be processed. At least two function blocks are required for buffer mode to have an effect.

```
TYPE MC_BUFFER_MODE :
(
    mcAborting                      := 16#0,
    mcBuffered                      := 16#1,
    mcBlendingLow                   := 16#12,
    mcBlendingPrevious              := 16#13,
    mcBlendingNext                  := 16#14,
    mcBlendingHigh                  := 16#15
) UINT;
END_TYPE
```

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✔	✔

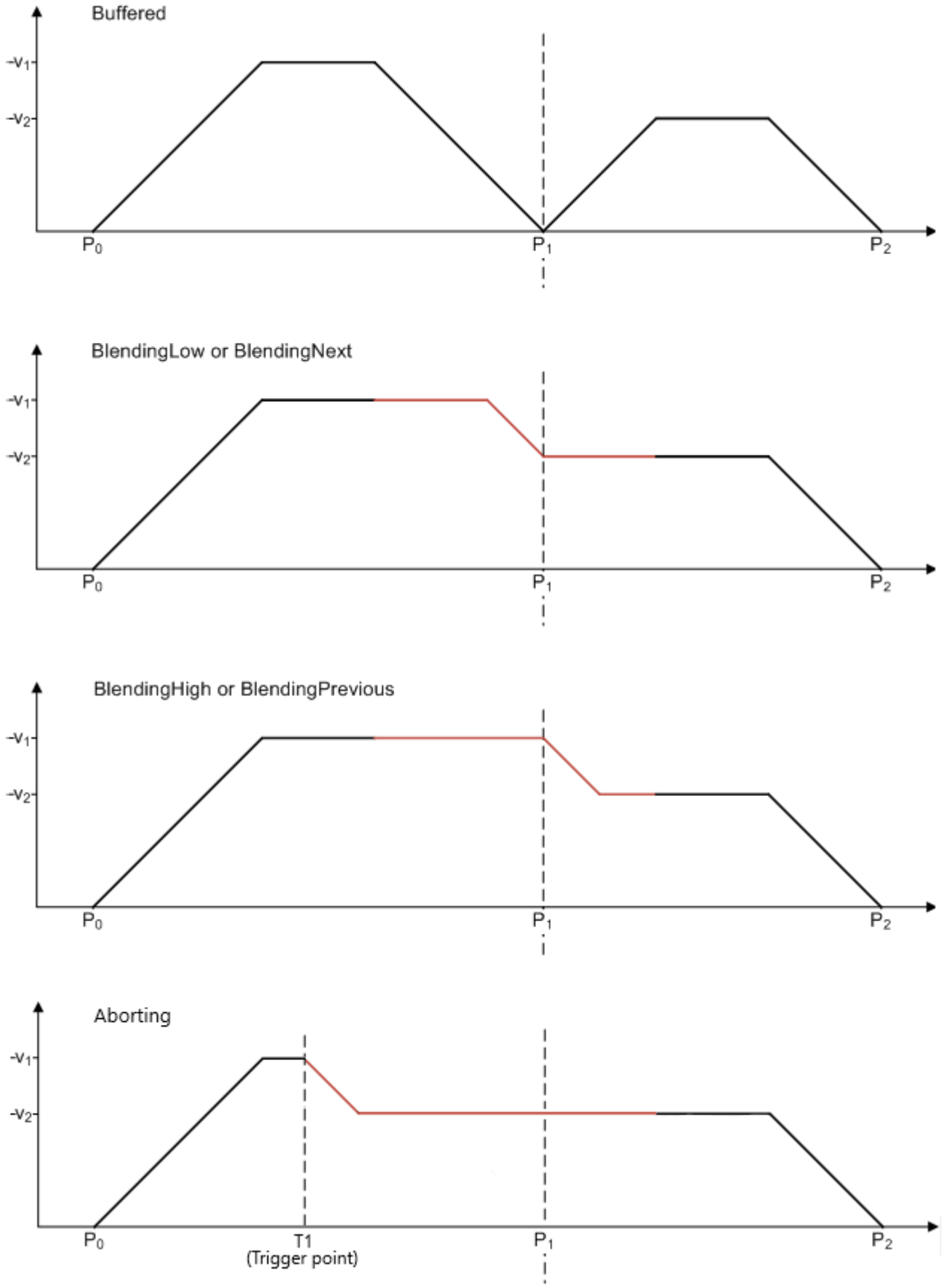
Example:

In the following example, a move command is used to move a group from position P_0 to P_1 and then to P_2. The reference point for the different velocity profiles is always P_1. The mode specifies the velocity v_1 or v_2 at this point.



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

If the speed of the second command is lower than the first the modes BlendingLow/BlendingNext and BlendingHigh/BlendingPrevious are equivalent.



Combinations of buffer mode and transition mode

Notice Buffer mode and transition mode are merely combined using TF5420.

The following table shows possible combinations of transition mode and buffer mode and its effect.

TM/BM	mcAborting	mcBuffered	mcBlendingPrevious	Other
mcTransModeNone	Previous command is aborted immediately. New movement is started. Velocity in transition is 0. This combination is only allowed for the 1st segment of a path.	Stop at the end of previous command. Subsequently next command is executed.	Not allowed	Not allowed
mcTransModeCornerDistance **new from V3.1.10.1, only compatible with MC Group Coordinated Motion**	Blending from active segment to first segment of new command. The intersection point of the segments is defined by the distance needed to stop on the active segment. This combination is only allowed for the 1st segment of a path.	Not allowed	Blending from last programmed command to new command	Not allowed
mcTransModeCornerDistanceAdvanced	Blending from active segment to first segment of new command. The intersection point of the segments is defined by the distance needed to stop on the active segment. This combination is only allowed for the 1st segment of a path.	Not allowed	Blending from last programmed command to new command	Not allowed
Other	Not allowed	Not allowed	Not allowed	Not allowed

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

9.4.1.2 MC_COMPENSATION_TYPE

The data type MC_COMPENSATION_TYPE is used to specify which compensation type is to be used.

```
TYPE MC_ COMPENSATION_TYPE:
(
    mcTypeInvalidCompensation           := 16#0,
    mcTypeGeoCompensation               := 16#1,
)UINT;
END_TYPE
```

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖	✖

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.6.07	PC or CX (x64)	Tc3_McCompensations

9.4.1.3 MC_DIRECTION

```
(* Defines the direction of the movement (e.g. for a modulo axis). *)
TYPE MC_DIRECTION :
(
    mcDirectionNonModulo        := 0, (* Position is interpreted as absolute position. *)
    mcDirectionPositive         := 1, (* Moves in positive direction. *)
    mcDirectionShortestWay      := 2, (* The direction of movement depends on whether the positive
direction of movement or the negative direction of movement is the shortest distance from the target
position. *)
    mcDirectionNegative         := 3 (* Moves in negative
                                        direction. *)
)
END_TYPE
```

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖	✖

MC_DIRECTION is used to specify the direction of movement during modulo positioning. Modulo positioning is only applicable to periodic systems. For open systems such as open tracks, only the value `mcDirectionNonModulo` is accepted.

mcDirectionNonModulo: The position is always interpreted as an absolute position.

mcDirectionPositive: Positive direction of movement

mcDirectionNegative: Negative direction of movement

mcDirectionShortestWay: The direction of movement depends on whether the positive direction or the negative direction has the shortest distance to the target position.

In combination with the Tc2_MC2 or Tc3_Mc3Definitions library it is possible that the data type cannot be resolved unambiguously (ambiguous use of name 'MC_Direction'). In this case the namespace must be specified when using the data type (Tc3_Mc3PlanarMotion.MC_DIRECTION or Tc3_Mc3Definitions.MC_DIRECTION or Tc2_MC2.MC_DIRECTION).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.4.1.4 MC_SYNC_MODE

```
(* Defines the direction of the synchronization position of modulo axes. *)
TYPE MC_SYNC_MODE :
(
    mcSyncModeNonModulo         := 0, (* SyncSlavePosition is interpreted as absolute position. *)
    mcSyncModePositive          := 1, (* Synchronizes in positive direction. *)
    mcSyncModeNegative          := 3 (* Synchronizes in negative direction. *)
)
END_TYPE
```

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖	✖

The value defines the direction in which synchronization is to be performed. The SyncMode specification is only effective if a modulo coordinate system has been defined for the axis. This can be a closed XTS track or a closed CA group, for example. The value is ignored if there is only one mathematical solution for reaching the synchronous position.

mcSyncModeNonModulo: The SlaveSyncPosition is always interpreted as an absolute position.

mcSyncModePositive: The slave axis synchronizes itself in positive direction of movement.

mcSyncModeNegative: The slave axis synchronizes itself in negative direction of movement.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 TF5400 Advanced Motion Pack V3.1.10.1	PC or CX (x64)	Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2

9.4.1.5 MC_SYNC_STRATEGY

The data type MC_SYNC_STRATEGY defines the synchronization profile of the slave for e.g. a MC_GearInPosCA command.

```
TYPE MC_SYNC_STRATEGY :
(
    mcSyncStrategyLate              := 16#1,
    mcSyncStrategySlow              := 16#2,
    mcSyncStrategyEarly             := 16#3
)
END_TYPE
```

TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✔	✖	✖

Examples:

The boundary conditions in the following examples are equal:

- The master motion is equal.
- The MasterStartDistance is equal.
- The distances (MasterSyncPosition – current master position) and (SlaveSyncPosition – current slave position) are in all three examples equal.
- The slave dynamics are equal.
- Configuration with one axis in the CA Group, one PTP axis as master.

- A motion command is issued to the master.

Example 1: mcSyncStrategyLate

The slave starts the synchronization as late as possible and with full dynamics (according to the input values velocity, acceleration, deceleration, jerk). It reaches the SlaveSyncPosition just in time with the correct gear ratio. The user has to take care that the master does not accelerate once the slave signals StartSync, since the synchronization profile is already planned with the maximal slave dynamics. The slave cannot violate its dynamic restrictions and therefore cannot compensate any master acceleration. This situation will result in an error at the function block.

1. Issue the command MC_GearInPosCA to the axis. The command becomes active while the master is still accelerating.

⇨ The slave starts synchronizing as late as possible and with full dynamics, and reached the SlaveSyncPosition when the master reached the MasterSyncPosition (black x-Cursor).

Example 2: mcSyncStrateySlow

The slave starts its synchronization in motion as soon as the master passes (MasterSyncPosition - MasterStartDistance) in the right direction if a MasterStartDist was set, otherwise as soon as the function block is Active. The dynamics of the slave are reduced so that the slave reaches the SlaveSyncPos with the right gear factor just in time when the master reaches the MasterSyncPos. The slave can compensate for an acceleration of the master if StartSync is also set, but only until the slave reaches its maximum dynamics.

1. Issue the command MC_GearInPosCA to the axis. The command becomes active while the master is still accelerating.

⇨ The slave starts synchronizing as soon as MC_GearInPosCA is Active. The dynamics is reduced such that the slave reaches the SlaveSyncPosition at the same time the master reaches the MasterSyncPosition (black x-Cursor).

A synchronization on a standing master can lead to a high load if mcSyncStrategySlow is used.

It is best to use mcSyncStrategyEarly in this case.

Example 3: mcSyncStrategyEarly

The slave starts synchronization immediately (if a MasterStartDistance is set: immediately after it was passed) and with full dynamics. The slave signals earlier InSync than demanded by the SlaveSyncPosition, but it is still guaranteed that demanded offset between master and slave (MasterSyncPosition – SlaveSyncPosition) is reached with the correct gear ratio. This strategy can synchronize on a standing master and is best suited if the master velocity is not constant. The slave will constantly try to synchronize. If the boundary conditions do not allow the slave to be InSync at the SlaveSyncPosition, this will not result in an error but the slave constantly tries to synchronize to the master.

1. Issue the command MC_GearInPosCA to the axis. The command becomes active while the master is still accelerating.

⇨ The slave starts the synchronization as soon as MC_GearInPosCA is Active and with full dynamics. The slave is InSync as soon as possible, but still reaches the SlaveSyncPosition at the same time the master reaches the MasterSyncPosition (black x-Cursor).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4018.26 TF5400 Advanced Motion Pack V3.1.1.17	PC or CX (x64)	Tc3_McCoordinatedMotion, Tc2_MC2

10 Samples

PTP Collision Avoidance

XTS Demo 1

Download: https://infosys.beckhoff.com/content/1033/tf5410_tc3_collision_avoidance/Resources/1546301963.zip

Description:
Project for XTS Starterkit (closed rail (3000 mm) with 10 movers) that executes MC_MoveAbsoluteCA [▶ 33] movements.

XTS Demo 2

Download: https://infosys.beckhoff.com/content/1033/tf5410_tc3_collision_avoidance/Resources/1546304267.zip

Description:
Project for XTS Starterkit (closed rail (3000 mm) with 10 movers) that executes MC_GearInPosCA [▶ 41] movements.

11 Appendix

11.1 Cyclic Group Interface

The cyclic group interface provides the cyclical data exchange between PLC and a NC group object. The group interface contains the directions NcToPlc [▶ 114] and PlcToNc [▶ 115]. Both direction are divided in common and group specific data.

AXES_GROUP_REF

```
TYPE AXES_GROUP_REF :
STRUCT
    PlcToNc AT %Q*       : CDT_PLCTOMC_GROUP;
    NcToPlc AT %I*       : CDT_MCTOPLC_GROUP;
END_STRUCT
END_TYPE
```

PlcToNc: PlcToNc [▶ 115] is a data structure that is cyclically exchanged between PLC and NC. Via this data structure the MC function blocks communicate with the motion group and send control information from the PLC to the NC. This data structure is automatically placed in the output process image of the PLC and must be linked with the input process image of a motion group.

NcToPlc: NcToPlc [▶ 114] is a data structure that is cyclically exchanged between PLC and NC. Via this data structure the MC function blocks communicate with the NC and receive status information from the NC. This data structure is automatically placed in the input process image of the PLC and must be linked in TwinCAT System Manager with the output process image of an NC axis.

11.1.1 NcToPlc

The structure is divided into general data and group-specific data.

General

GroupOID: TcCOM object ID (OID) of this group.

GroupType: Type of this group: 0 = Invalid (mcGroupTypeInvalid), 1 = Collision avoidance (mcGroupTypeCA), 2 = DXD/CNC (mcGroupTypeDxd).

GroupStatus: Contains information about the group status (see GroupStatus [▶ 114]).

GroupErrorId: Identification of current error (0 = no error).

GroupAxesCount: Number of axes currently belonging to this group (e.g. added via MC_AddAxisToGroup).

GroupStatus:

State: See Group State Diagram.

1 = Disabled (mcGroupStateDisabled)
2 = Standby (mcGroupStateStandby)
3 = Moving (mcGroupStateMoving)
4 = Stopping (mcGroupStateStopping)
5 = Error Stop (mcGroupStateErrorStop)
6 = Homing (mcGroupStateHoming)
7 = Not Ready (mcGroupStateNotReady)
8 = Suspended (mcGroupStateSuspended)

Flags: Additional optional status information.

IsEnableRequested: Defines whether an activation or deactivation of a group is requested.

Dxd (multi-dimensional movement)

PathVelo: Velocity on the path without direction.

InvokeId: Segment ID for analysis purposes.

CM (MC Group Coordinated Motion)

available from V3.1.10.1

PathVelo: Absolute value of the Cartesian velocity on the path.

InvokeId: Segment ID for analysis purposes.

IsInBlendingSegment: Indicates whether a blendig segment is active.

RemainingTimeActiveJob: Remaining time of the current segment.

RemainingCartesianDistanceActiveJob: Remaining distance for the current segment.

ActiveBlockerId: Id of the active blocker.

available from V3.1.10.30

RemainingTimeToSync: Remaining time until the axis group is synchronized with the conveyor belt during conveyor tracking.

RemainingCartesianDistanceToSync: Remaining distance until the axis group is synchronized with the conveyor belt during conveyor tracking.

11.1.2 PlcToNc

The structure is divided in a common data and a group specific data.

Common

OverrideFactor: Desired Override Factor (1.0 = 100%, Default Value is 1.0)

11.2 MC_LREAL/Special Input Values

Data type MC_LREAL, is equivalent to data type LREAL. However, there exist a few additional values that have a special signification.

Value	Signification	Example
MC_DEFAULT	The input is executed with default value for this input.	Acceleration, Deceleration, Jerk for all motion commands
MC_MAXIMUM	The command is executed with maximum value for this input.	Generally, from software version 3.1.4.4 on for specific motion commands the value `MC_MAXIMUM` can be assigned to the inputs `Velocity`, `Acceleration`, `Deceleration` and `Jerk`. For more detailed information refer to the particular documentation of the function block the input intended to be supplied with the `MC_MAXIMUM` value belongs to.
MC_IGNORE	The input is ignored.	MC_GearInPosCA.MasterStartDistance
MC_INVALID	The input must be set by the user, there exists no default or maximum value, nor can the input be ignored.	MC_MoveAbsoluteCA.Position

11.3 Modulo positioning

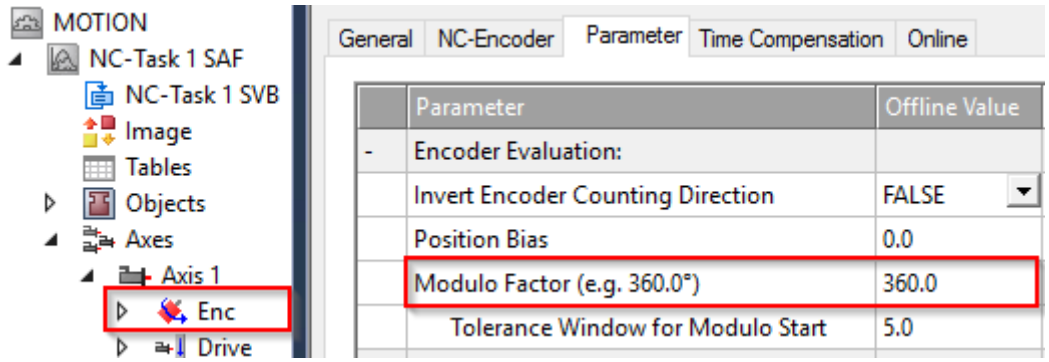
The modulo positioning can be applied to closed linear axes as well as to rotary axes. TwinCAT does not distinguish between these types. A modulo axis has a consecutive absolute position in the range ±∞. The modulo position of the axis is simply a piece of additional information about the absolute axis position. Modulo positioning represents the required target position in a different way. As opposed to absolute positioning, in which the user clearly specifies the target, the absolute target position is formed from the following parameters in modulo positioning:

- Modulo target position
- Modulo Factor
- Tolerance Window
- Direction, see MC_DIRECTION [▶ 108]
- (Additional Turns, see ST_MoveAbsoluteCAOptions [▶ 49])

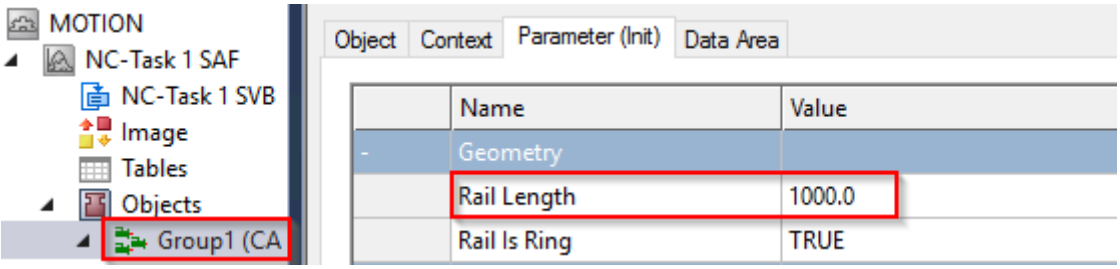
Modulo Factor

The modulo positioning basically refers to an adjustable Modulo Factor, which is set in the TwinCAT Engineering. The axis and its use must be observed here, for example:

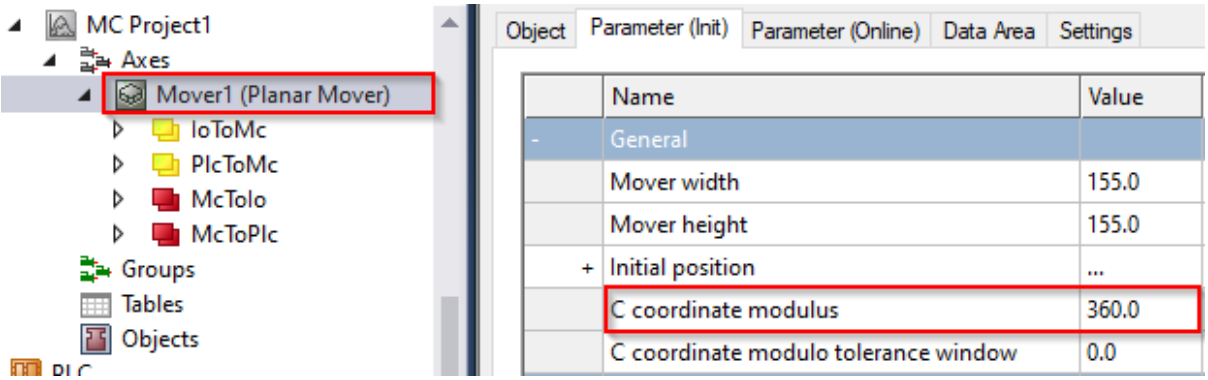
- If a PTP axis is used, the Modulo Factor of the axis encoder applies; details in the Notes on the modulo positioning of a PTP axis.



- If, for example, a mover is used on an XTS system in a CA group, the Rail Length set in the CA group applies.



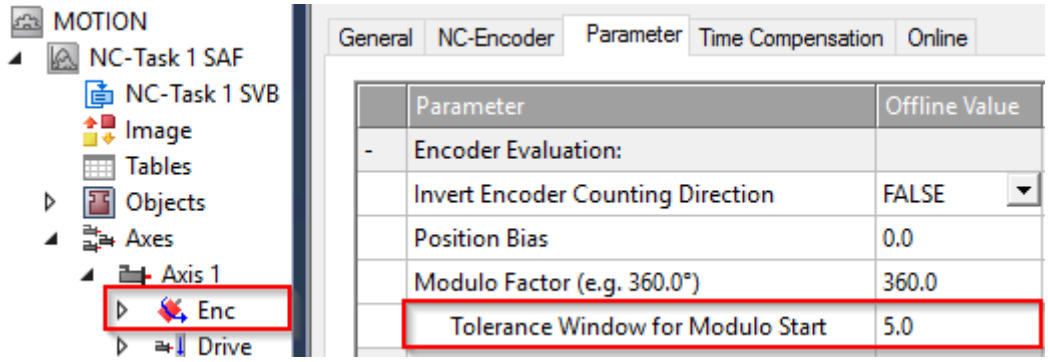
- If an XPlanar mover is used, its "C-axis" modulo can be positioned. Here, the Modulo Factor is set as "C coordinated modulus" in the Init parameters of the XPlanar mover.



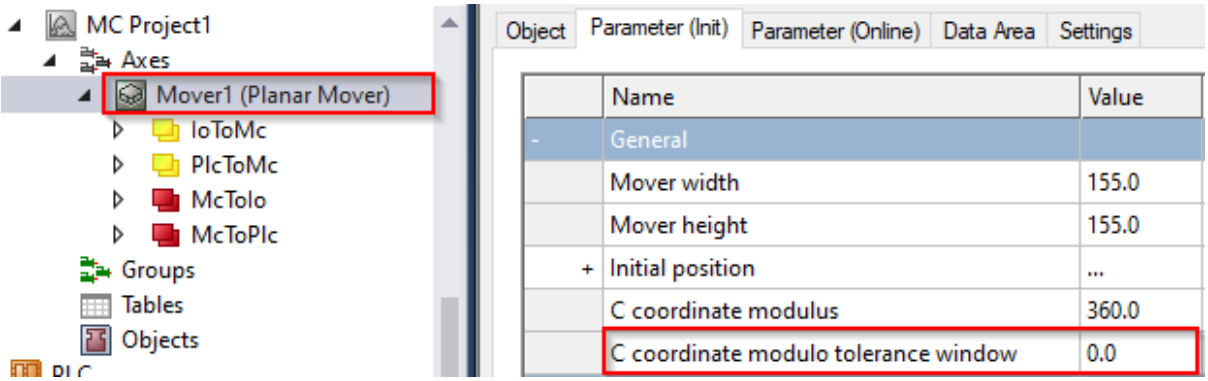
Modulo Tolerance Window

The Modulo Tolerance Window defines a position window around the current modulo set position of the axis. The window width corresponds to twice the specified value (set position ± tolerance value) and is specified in the TwinCAT Engineering:

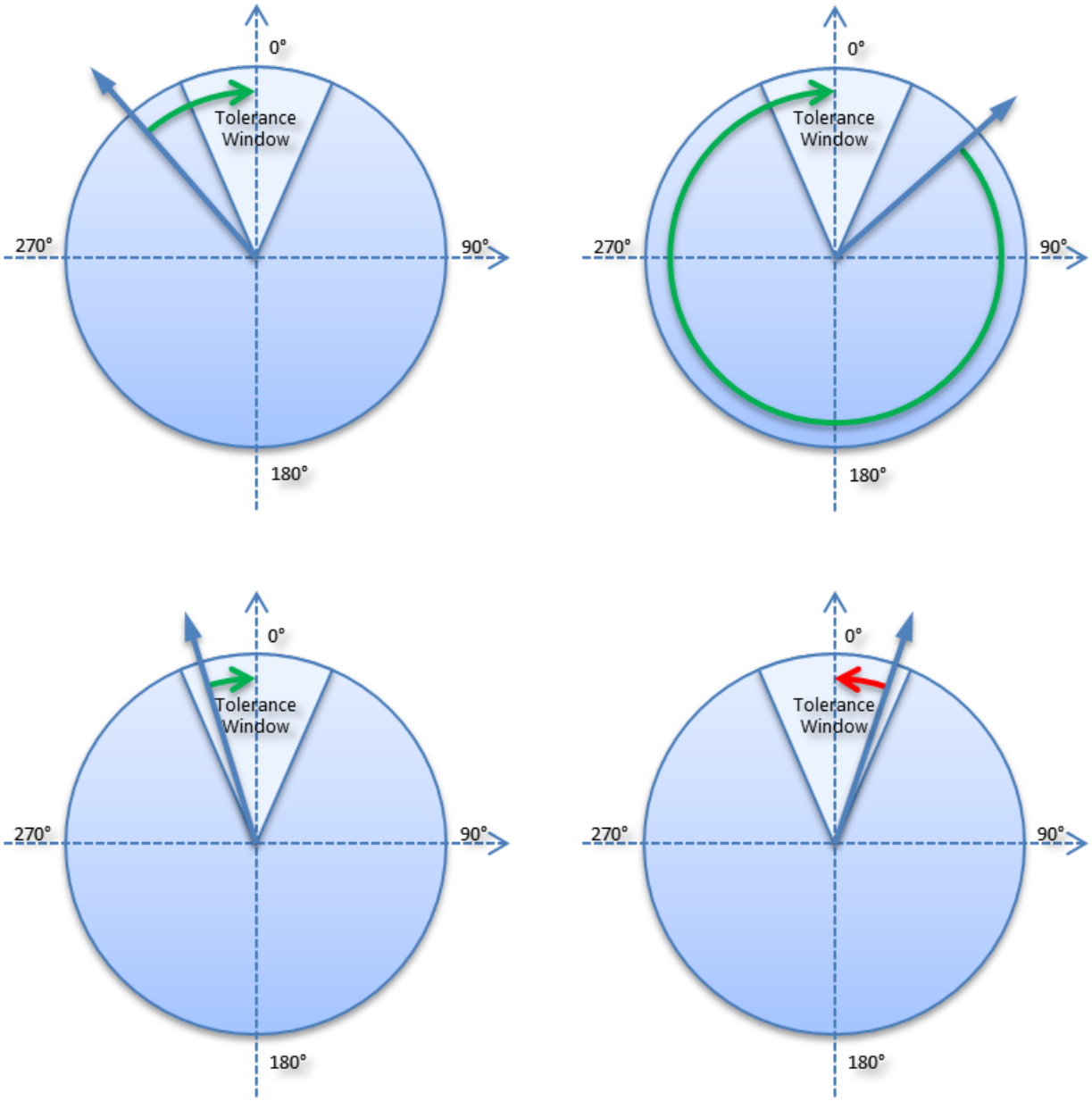
- In the case of a PTP axis or an axis in a CA group, the Tolerance Window is defined in the axis encoder



- In the case of the C-axis of an XPlanar mover, the Tolerance Window is defined in the Init parameters of the XPlanar mover.



The positioning of an axis is always referenced to its current actual position. Unintentional revolutions may be performed if the actual position and the target position are very close to each other, for example, if the actual position is minimally greater than the target position and `Direction = mcDirectionPositive` was selected. This can occur in particular if the actual position is determined inaccurately (e.g. on account of incorrect positioning due to the axis stalling, or due to the finite resolution of the encoder). In order to avoid this, a tolerance window for modulo positioning can be set. If the distance between the start and target positions is less than or equal to the Tolerance Window, then the target position is driven to by the shortest route (as with `Direction = mcDirectionShortestWay`), i.e. also contrary to the `direction` specified.



Examples

- Modulo Factor = 100
- Tolerance Window = 1

Parameter Direction	Absolute Startposition	Target position	Parameter Additional Turns	Relative path	Absolute end position	Modulo end position
mcDirectionPositive	110	10	0	0	110	10
mcDirectionPositive	110.9	10	0	-0.9	110	10
mcDirectionPositive	112	10	0	98	110	10
mcDirectionPositive	95	10	0	15	110	10
mcDirectionPositive	110	110	0	ERROR: INVALID TARGET POSITION		
mcDirectionPositive	110	10	3	300	410	10
mcDirectionPositive	110.9	10	3	299.1	410	10
mcDirectionPositive	112	10	3	398	410	10

Parameter Direction	Absolute Startposition	Target position	Parameter Additional Turns	Relative path	Absolute end position	Modulo end position
mcDirectionPositive	95	10	3	315	410	10
mcDirectionPositive	110	110	3	ERROR: INVALID TARGET POSITION		
mcDirectionNegative	110	10	0	0	110	10
mcDirectionNegative	109.9	10	0	0.1	110	10
mcDirectionNegative	108	10	0	-98	10	10
mcDirectionNegative	95	10	0	-85	10	10
mcDirectionNegative	110	110	0	ERROR: INVALID TARGET POSITION		
mcDirectionNegative	410	10	3	-300	110	10
mcDirectionNegative	409.9	10	3	-299.9	110	10
mcDirectionNegative	408	10	3	-398	10	10
mcDirectionNegative	495	10	3	-385	10	10
mcDirectionNegative	410	110	3	ERROR: INVALID TARGET POSITION		
mcDirectionShortestWay	440	50	0	10	450	50
mcDirectionShortestWay	440	10	0	-30	410	10
mcDirectionShortestWay	440	50	1	ERROR: INVALID ADDITIONAL TURN COUNT		

Further samples

Further examples without the Additional Turns parameter can be found in the Notes on the modulo positioning of a PTP axis.

Trademark statements

Third-party trademark statements

More Information:
www.beckhoff.com/TF5410

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