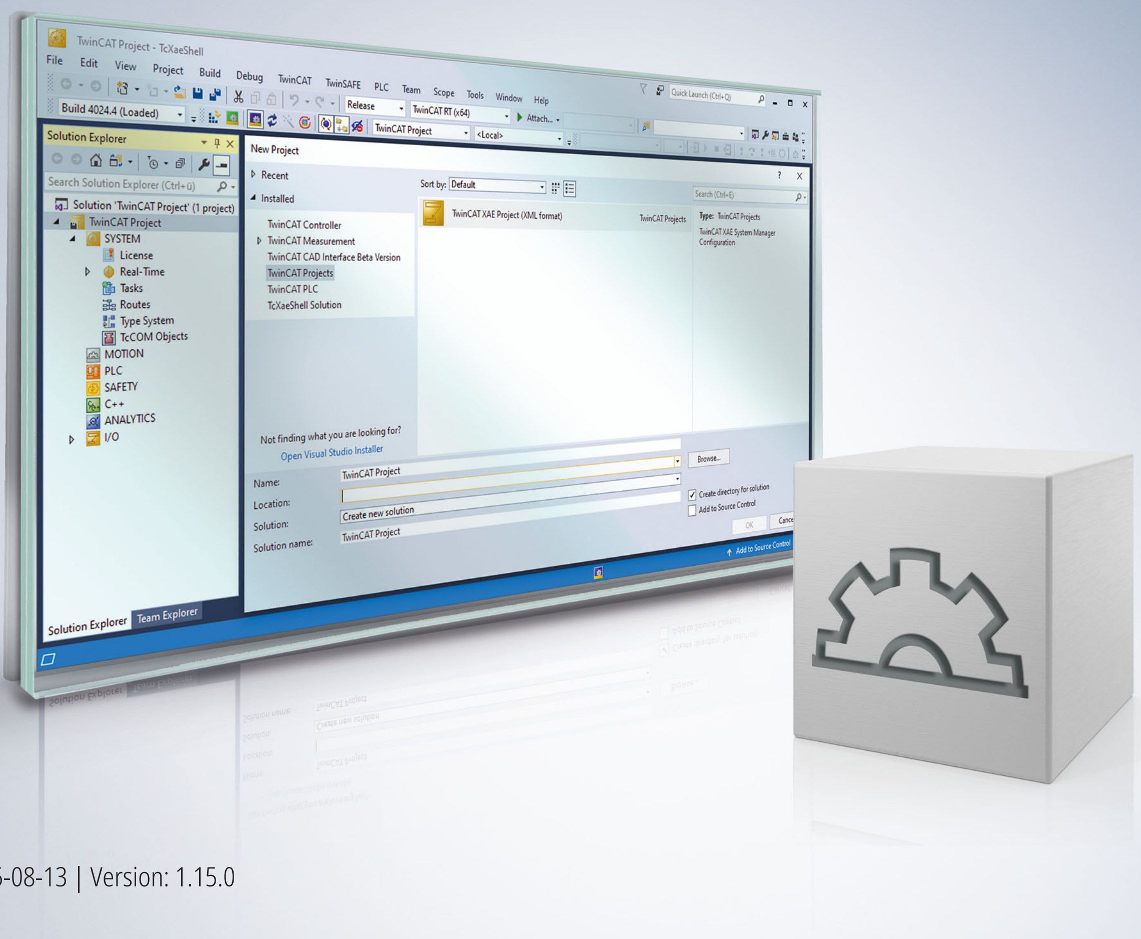
BECKHOFF New Automation Technology

Manual | EN

TF5420

TwinCAT 3 | Motion Pick-and-Place



2025-08-13 | Version: 1.15.0

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.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

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

Disclaimer

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

Trademarks

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

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

Copyright

The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

Third-party trademarks

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

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

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

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

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

Warning of damage to property or environment

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

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Introduction

The TF5420 TwinCAT 3 Motion Pick-and-Place software package is installed together with the TF5400 software package.

Target system

Windows 10

TwinCAT 3 Motion Pick-and-Place

TF5420 TwinCAT 3 Motion Pick-and-Place executes multi-dimensional motions. It was specially developed for the requirements of pick-and-place applications. It should be applied to motions where the precise path dynamics in segment transitions are not so important, but where the user wishes to get from one point to another as quickly as possible. The *Tc3_McCoordinatedMotion* library contains all the associated function blocks.

Additional licensing requirements

TF5420 TwinCAT 3 Motion Pick-and-Place requires the TC1260 license.

3 Overview of the new functions

As of TF5400 V3.3.72

- New: In the GroupDisabled state, PTP commands of the Tc2_MC2 library (e.g. MC_MoveAbsolute) are permitted on axes of the group.

As of TF5400 V3.3.19

- MC-Group Pick-and-Place no longer available.
- New: Function block MC_SetCoordinateTransformPreparation [▸ 54] for the MC-Group Coordinated Motion
- Requires TwinCAT V3.1.4024.40 or higher

As of TF5400 V3.2.27 for the MC Group Coordinated Motion:

- New: Introduction of additional dynamic constraints for path and auxiliary axes.
- New: Optionally, the override also affects the synchronization phase for the MC_TrackConveyorBelt [▸ 55].
- Optimizations to the MC_TrackConveyorBelt [▸ 55] that prevent SAF cycle misalignment between conveyor (master) and slave axis.
- Optimizations of the error reaction for the MC_TrackConveyorBelt [▸ 55]. In the event of a runtime error of the conveyor belt (master), an active MC_MovePath [▸ 70] is not aborted and an error reaction is to be triggered via the PLC.
- Requires an x64 platform

As of TF5400 V3.1.10.64:

- New: In an MC Group Coordinated Motion with Geo Blending, a blocker that is triggered early enough before it becomes active will be blended over and passed on without interruption.
- Requires TwinCAT V3.1.4024.24 or higher

As of TF5400 V3.1.10.1:

- New: Group type MC Group Coordinated Motion.
- New: Data for the MC Group Coordinated Motion in the cyclic group interface [▸ 96].
- New: Function blocks for MC Group Coordinated Motion:
  - MC_BlockerPreparation [▸ 71]
  - MC_ReleaseBlocker [▸ 73]
  - MC_GroupReadBlockerStatus [▸ 74]
  - MC_DwellTimePreparation [▸ 75]
- New: MC_GroupHalt [▸ 60] function block for MC Group Coordinated Motion.
- New: Enum MC_TRANSITION_MODE [▸ 83] extended by mcTransModeCornerDistance for MC Group Coordinated Motion.
- New: Enum MC_CIRC_PATHCHOICE [▸ 81] extended by mcCircPathchoiceShortSegment and mcCircPathchoiceLongSegment.
- Requires TwinCAT V3.1.4024.7 or higher

As of TF5400 V3.1.6.27:

- New: The remaining time and distance of the current segment can be read via ADS, see Index Offset Specification for MC Group Parameters [▸ 97].
- Requires TwinCAT V3.1.4022.0 or higher

As of TF5400 V3.1.6.3:

- New: Function block MC_SetCoordinateTransform [▸ 53].

- New: Function block MC_TrackConveyorBelt [▶ 55]

As of TF5400 V3.1.4.4:

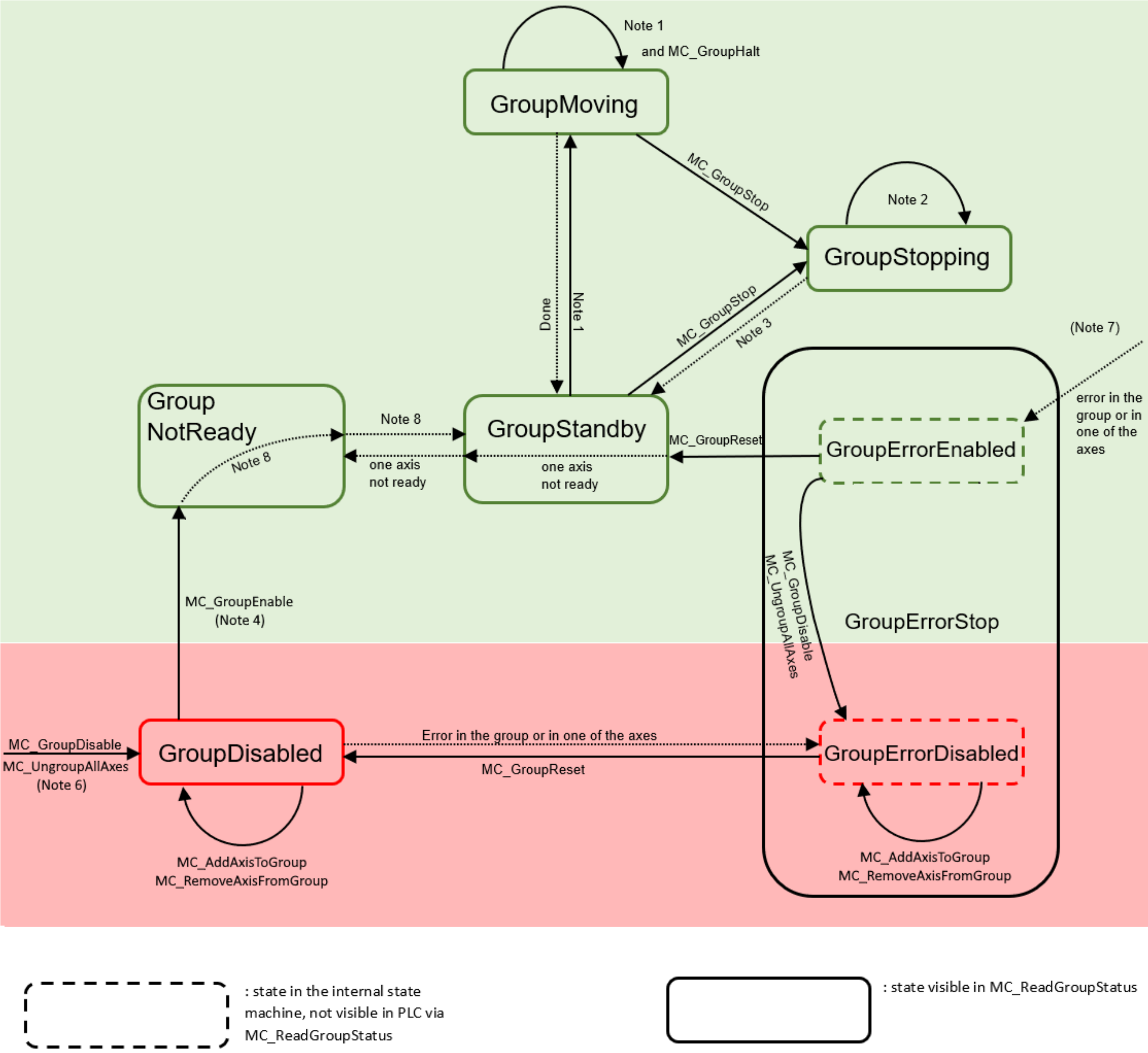
- New: Special input value [▶ 98] `MC_MAXIMUM`.

As of TF5400 V3.1.2.47:

- New: Function block MC_MoveCircularAbsolutePreparation.

4 Group State Diagram

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



Note	Description
1	Applicable for all non-administrative (movement) function blocks.
2	In the GroupStopping state, many function blocks can be called, but they are not executed. Exceptions are MC_GroupDisable [▶ 43] and MC_UngroupAllAxes [▶ 58], which cancel the stop and create the transition to the GroupDisabled state.
3	MC_GroupStop [▶ 62].DONE
4	The number of axes in the group (added via MC_AddAxisToGroup [▶ 40]) must be equal to the number of axes in the spatial axis convention plus the Additional Axes Count.
5	-
6	MC_GroupDisable can be called in all states and changes the state to GroupDisabled. When MC_GroupDisable is called in an error state, the state changes to GroupErrorDisabled.

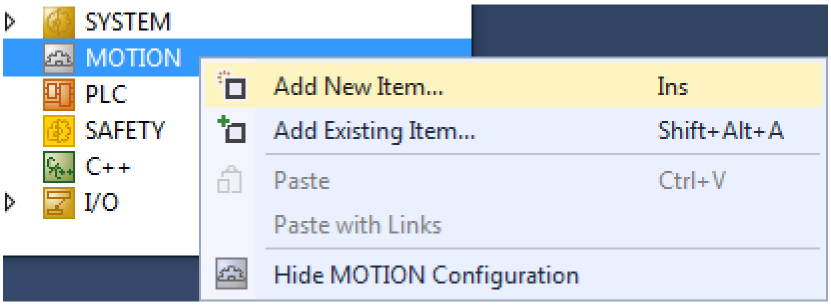
7	The state change to GroupErrorEnabled occurs in the axis/group error case from any state in which the group is enabled.
8	The state change occurs when "bIsControlLoopClosed" is TRUE for all axes. bPositiveDirection"/"bNegativeDirection" do not have to be enabled.
9	-
10	MC_GroupReset [▶ 48] has no effect if the state is different from GroupErrorStop. MC_GroupReset must be called to exit the GroupErrorStop state.

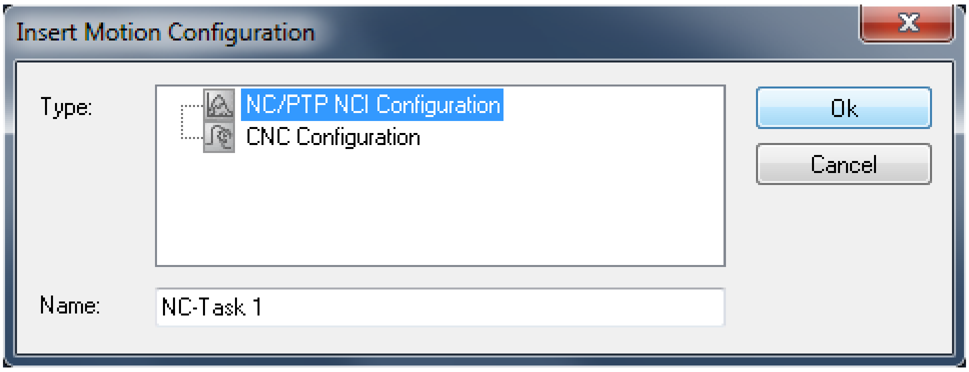
5 MC Group (TF5420 TwinCAT 3 Motion Pick-and-Place)

5.1 Configure an MC Group

Basically, the configuration described here is valid for all Motion Objects in the Advanced Motion Pack.

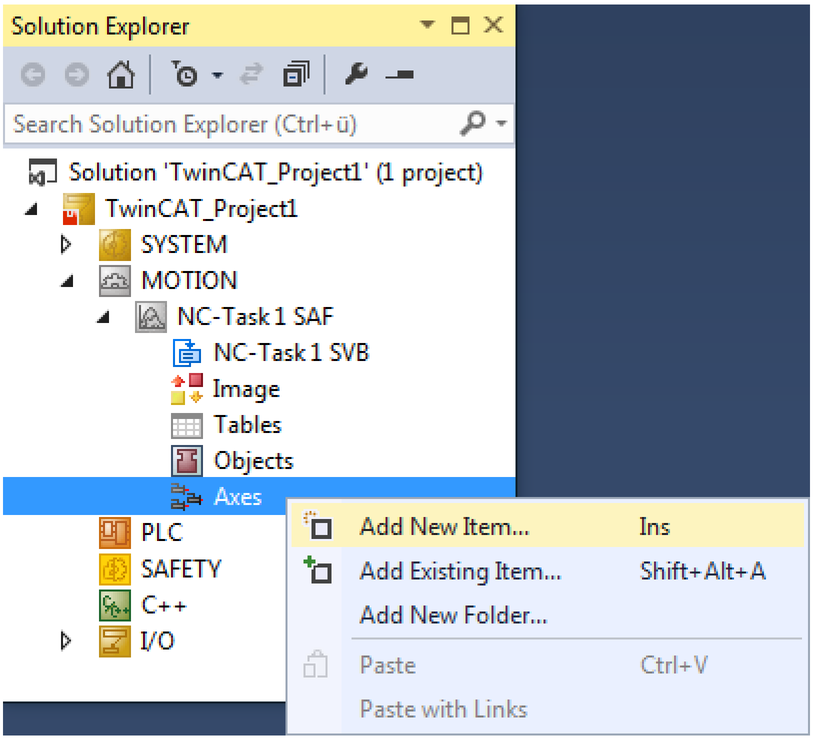
1. Add new “NC/PTP NCI Configuration” in the Motion section.

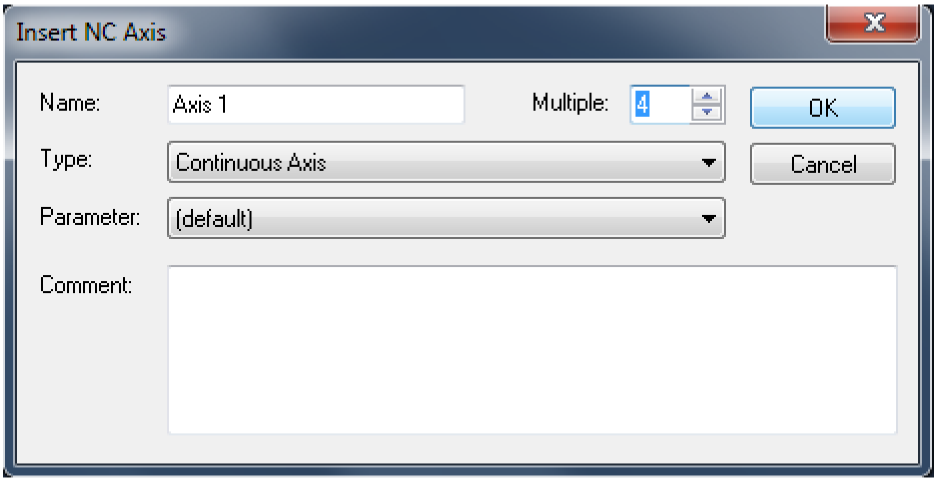




Name: NC-Task 1

2. Add all axes to the NC-Configuration.





Insert NC Axis

Name: Axis 1

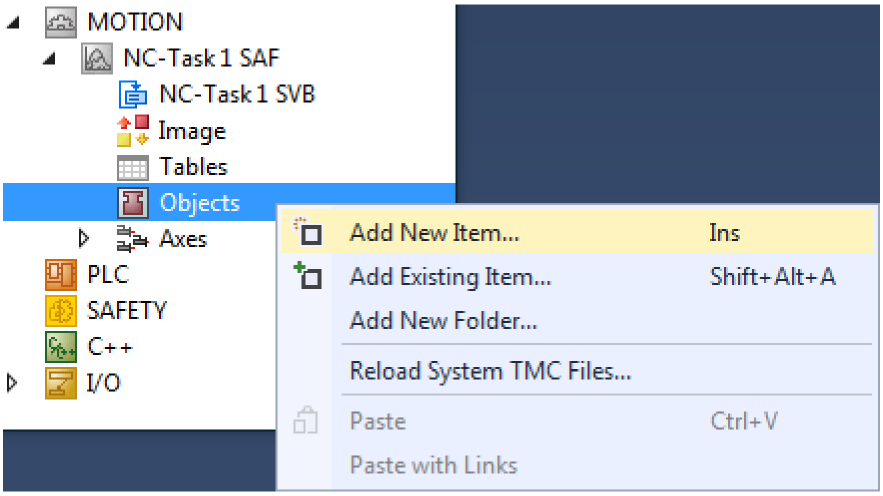
Multiple: 4

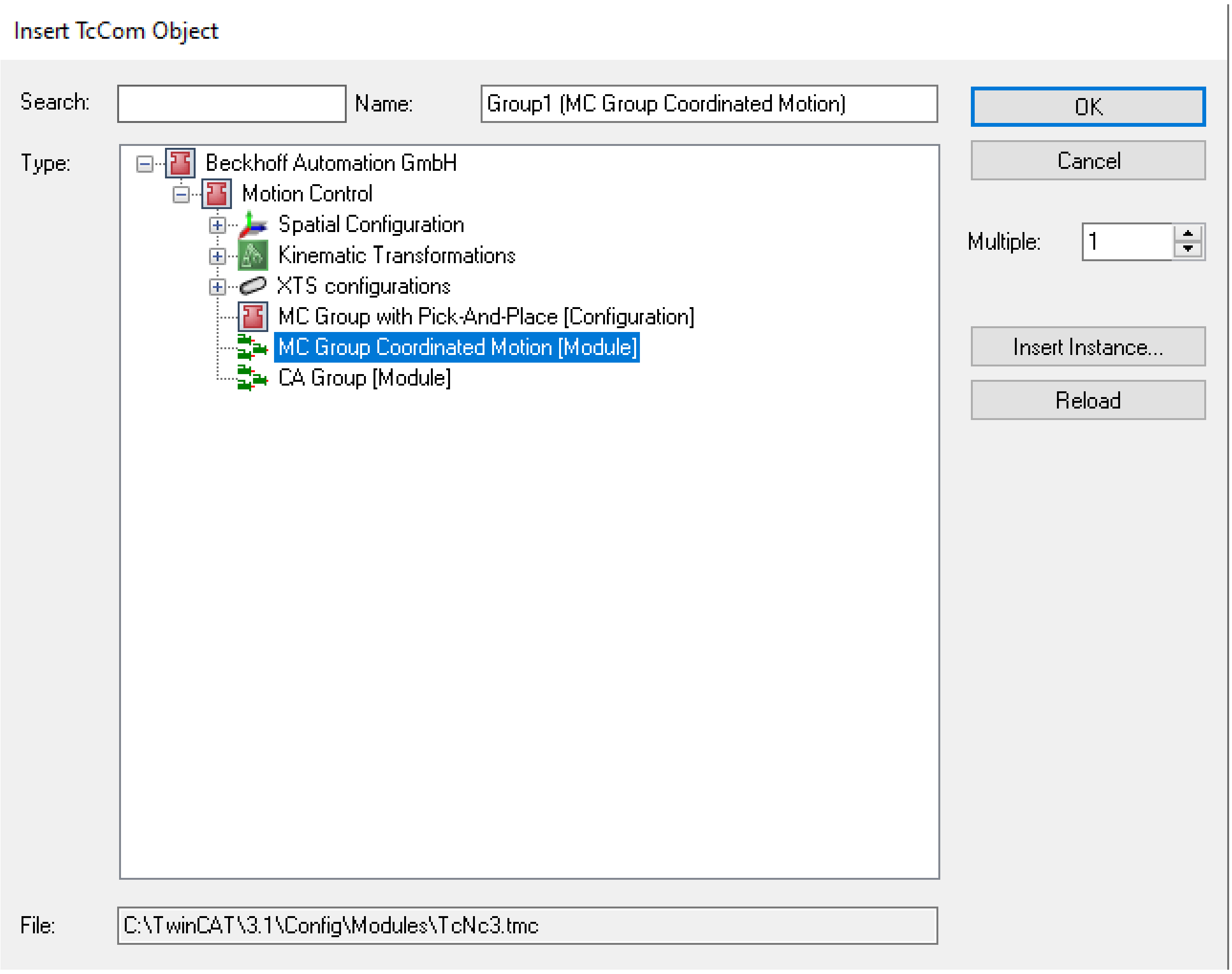
Type: Continuous Axis

Parameter: (default)

Comment:

3. Add the appropriate Group to the entry “Objects” in the NC-Configuration:
   For Coordinated Motion, multi-dimensional movements: MC Group Coordinated Motion [▶ 18] or MC Group with Pick-and-Place [▶ 20].





Insert TcCom Object

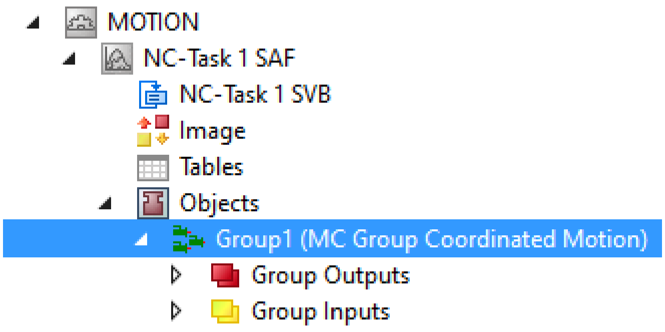
Search:

Name: Group1 (MC Group Coordinated Motion)

Type:
- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - MC Group Coordinated Motion [Module]
    - CA Group [Module]

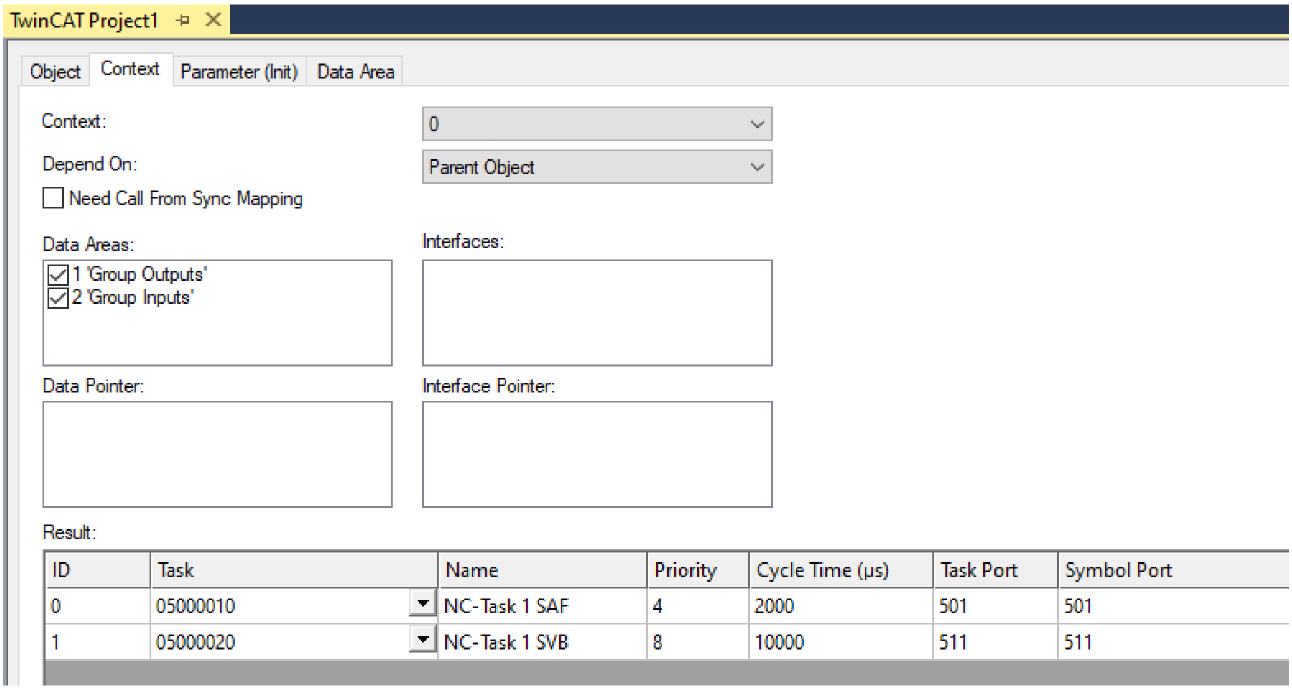
File: C:\TwinCAT\3.1\Config\Modules\TcNc3.tmc

Multiple: 1



- MOTION
  - NC-Task 1 SAF
    - NC-Task 1 SVB
    - Image
    - Tables
    - Objects
      - Group1 (MC Group Coordinated Motion)
        - Group Outputs
        - Group Inputs

4. Check the Tasks in the Group.
   Context ID 0 has to be set to **“NC-Task 1 SAF”**.
   Context ID 1 has to be set to **“NC-Task 1 SVB”**.



Context: 0

Depend On: Parent Object

□ Need Call From Sync Mapping

Data Areas:
☑ 1 'Group Outputs'
☑ 2 'Group Inputs'

Interfaces:

Data Pointer:

Interface Pointer:

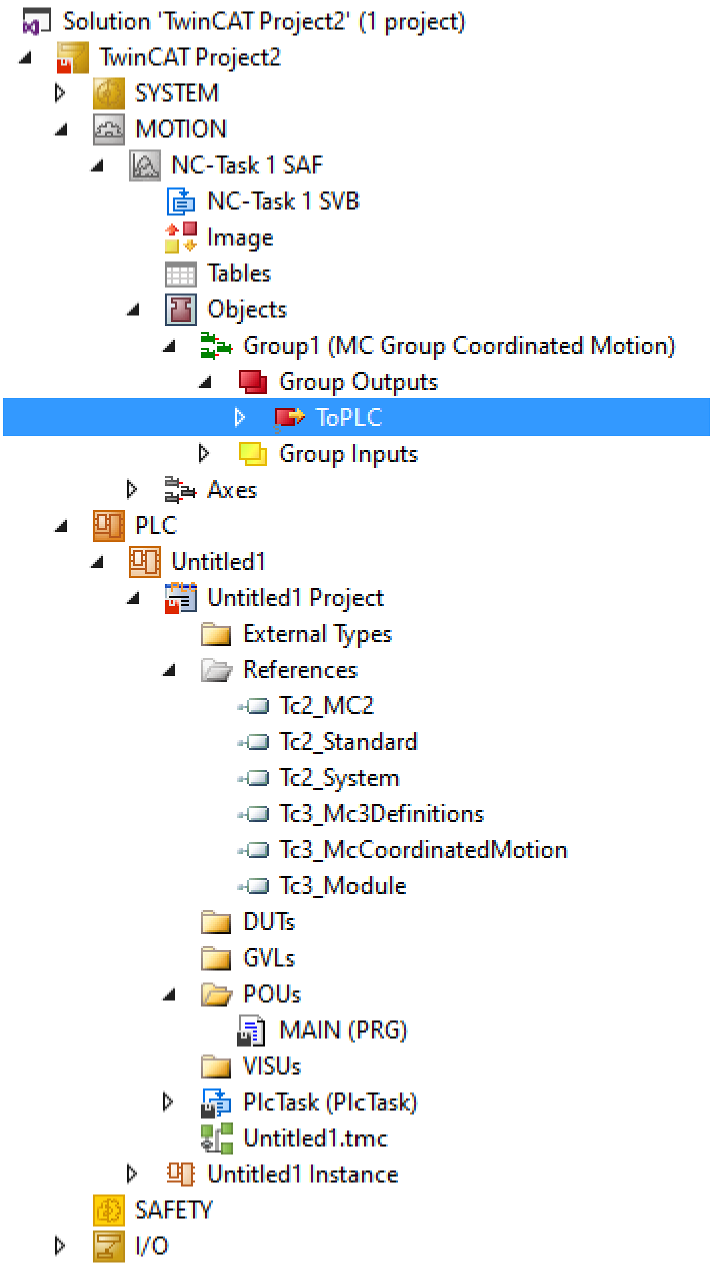
Result:

ID	Task	Name	Priority	Cycle Time (µs)	Task Port	Symbol Port
0	05000010	NC-Task 1 SAF	4	2000	501	501
1	05000020	NC-Task 1 SVB	8	10000	511	511

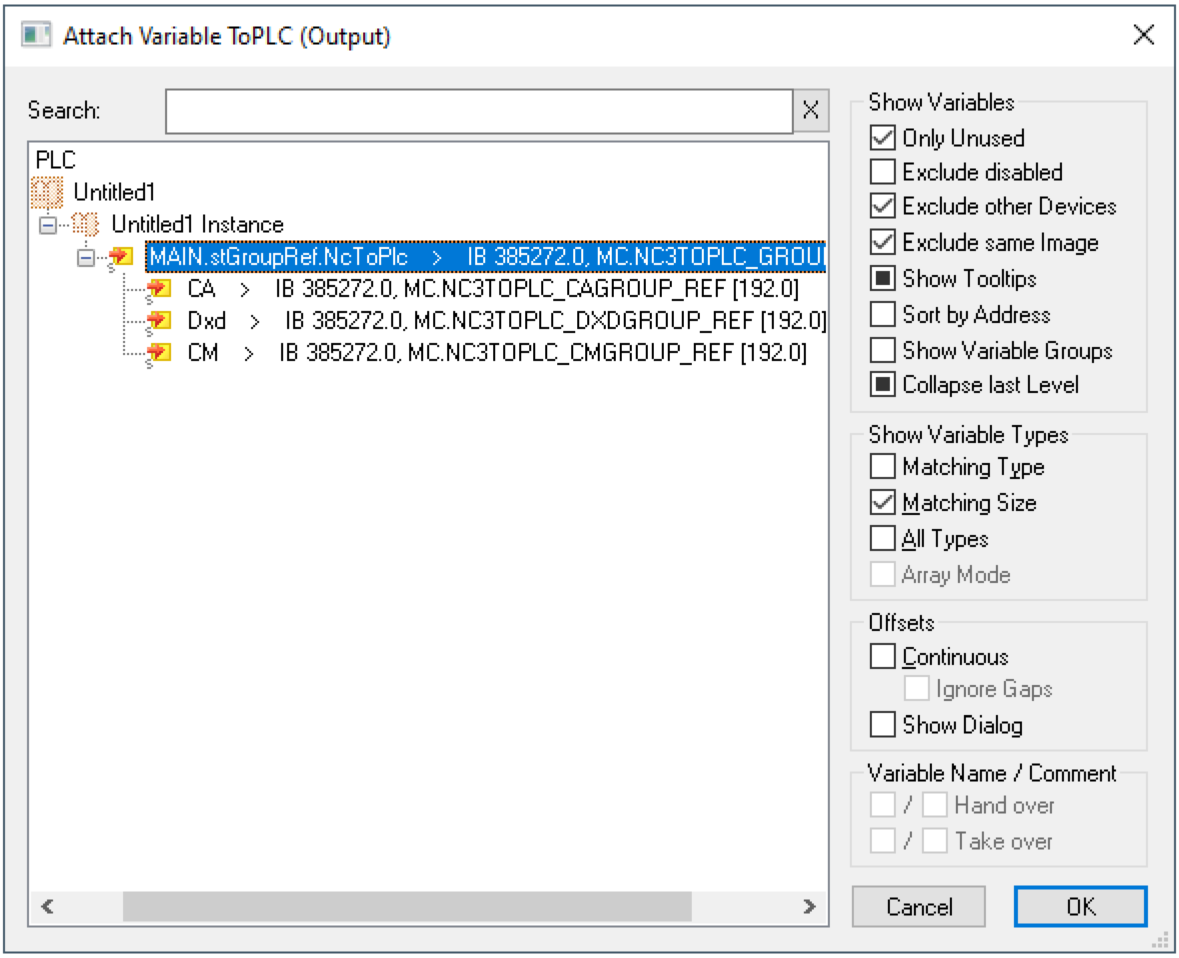
5. Configure the group parameters according to the desired application.
   For more explanations referring to the group parameters see the following sections.
6. To address the group from the PLC a cyclic interface has to be declared and linked to the IO of the group (see PLC Library Tc3_McCoordinatedMotion [▶ 18]). To address and enable the axes the library “Tc2_Mc2” has to be added to the project.

⇨ A new “NC/PTP NCI Configuration” has been established.

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



- Solution 'TwinCAT Project2' (1 project)
  - TwinCAT Project2
    - SYSTEM
    - MOTION
      - NC-Task 1 SAF
        - NC-Task 1 SVB
        - Image
        - Tables
        - Objects
          - Group1 (MC Group Coordinated Motion)
            - Group Outputs
              - ToPLC
            - Group Inputs
        - Axes
    - PLC
      - Untitled1
        - Untitled1 Project
          - External Types
          - References
            - Tc2_MC2
            - Tc2_Standard
            - Tc2_System
            - Tc3_Mc3Definitions
            - Tc3_McCoordinatedMotion
            - Tc3_Module
          - DUTs
          - GVLs
          - POUs
            - MAIN (PRG)
          - VISUs
          - PlcTask (PlcTask)
          - Untitled1.tmc
        - Untitled1 Instance
    - SAFETY
    - I/O


Attach Variable ToPLC (Output)

Search:

PLC
- Untitled1
  - Untitled1 Instance
    - MAIN.stGroupRef.NcToPlc > IB 385272.0, MC.NC3TOPLC_GROU
      - CA > IB 385272.0, MC.NC3TOPLC_CAGROUP_REF [192.0]
      - Dxd > IB 385272.0, MC.NC3TOPLC_DXDGROUP_REF [192.0]
      - CM > IB 385272.0, MC.NC3TOPLC_CMGROUP_REF [192.0]

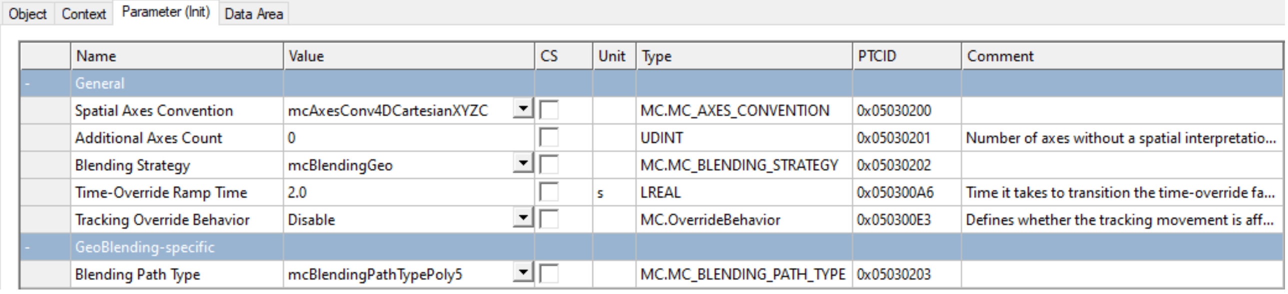
Show Variables
- ☑ Only Unused
- □ Exclude disabled
- ☑ Exclude other Devices
- ☑ Exclude same Image
- ■ Show Tooltips
- □ Sort by Address
- □ Show Variable Groups
- ■ Collapse last Level

Show Variable Types
- □ Matching Type
- ☑ Matching Size
- □ All Types
- □ Array Mode

Offsets
- □ Continuous
- □ Ignore Gaps
- □ Show Dialog

Variable Name / Comment
- □ / □ Hand over
- □ / □ Take over

5.2 MC Group Coordinated Motion



Name	Value	CS	Unit	Type	PTCID	Comment
General						
Spatial Axes Convention	mcAxesConv4DCartesianXYZC	□		MC.MC_AXES_CONVENTION	0x05030200	
Additional Axes Count	0	□		UDINT	0x05030201	Number of axes without a spatial interpretatio...
Blending Strategy	mcBlendingGeo	□		MC.MC_BLENDING_STRATEGY	0x05030202	
Time-Override Ramp Time	2.0	□	s	LREAL	0x050300A6	Time it takes to transition the time-override fa...
Tracking Override Behavior	Disable	□		MC.OverrideBehavior	0x050300E3	Defines whether the tracking movement is aff...
GeoBlending-specific						
Blending Path Type	mcBlendingPathTypePoly5	□		MC.MC_BLENDING_PATH_TYPE	0x05030203	

Parameter (Init)

Spatial Axes Conventions

Three axes conventions can be set.

The axes conventions define how the axes are interpreted in the axis group. In combination with "Additional Axes Count", they define the dimension of the axis group and thus the number of axes that need to be added, as well as the way in which each of the added axes is interpreted.

Parameter	Value	Type	Description
Spatial Axes Convention	mcAxesConv2DCartesianXY	`MC.MC_AXES_CONVENTION`	A 2D group consisting of `X`, `Y`. The order of the translatory axes in the configuration determines the order of translation.

Parameter	Value	Type	Description
Spatial Axes Convention	mcAxesConv3DCartesianXYZ	MC.MC_AXES_CONVENTION	A 3D group consisting of X, Y, Z. The order of the translatory axes in the configuration determines the order of translation.
Spatial Axes Convention	mcAxesConv4DCartesianXYZC	MC.MC_AXES_CONVENTION	A 4D group consisting of X, Y, Z and a rotary axis around Z (C). The order of the translatory and rotary axes in the configuration determines the order of translation and rotation.

Additional Axes Count

Number of axes in the axis group that have no geometric interpretation. Between 0 and 8 axes of this type can be inserted.

Blending Strategy

Sets the blending strategy.

Parameter	Value	Type	Description
Blending Strategy	mcBlendingGeo	MC.MC_BLENDING_STRATEGY	The blending path is defined geometrically and then executed with the dynamics allowed for the path.
Blending Strategy	mcBlendingSuperpos	MC.MC_BLENDING_STRATEGY	The blending path results dynamically from the superposition of two segments in the blending area.

Time Override Ramp Time

Ramp time for override modification from 0 % to 100 %. The time override is superimposed on the actual profile. This can result in higher dynamics in total during the override changes than were parameterized at the group.

Tracking Override Behavior

Defines whether the portion of the movement for accurate synchronization is subject to override or not.

Available from TF5400 3.2.27.

Parameter	Value	Type	Description
Tracking Override Behavior	Disable	MC.OverrideBehavior	The group override has no effect on the portion of movement for accurate synchronization.
Tracking Override Behavior	Enable	MC.OverrideBehavior	The group override also affects the portion of movement for accurate synchronization.

GeoBlending-specific parameters

Blending Path Type

Defines the geometry used for the blending path.

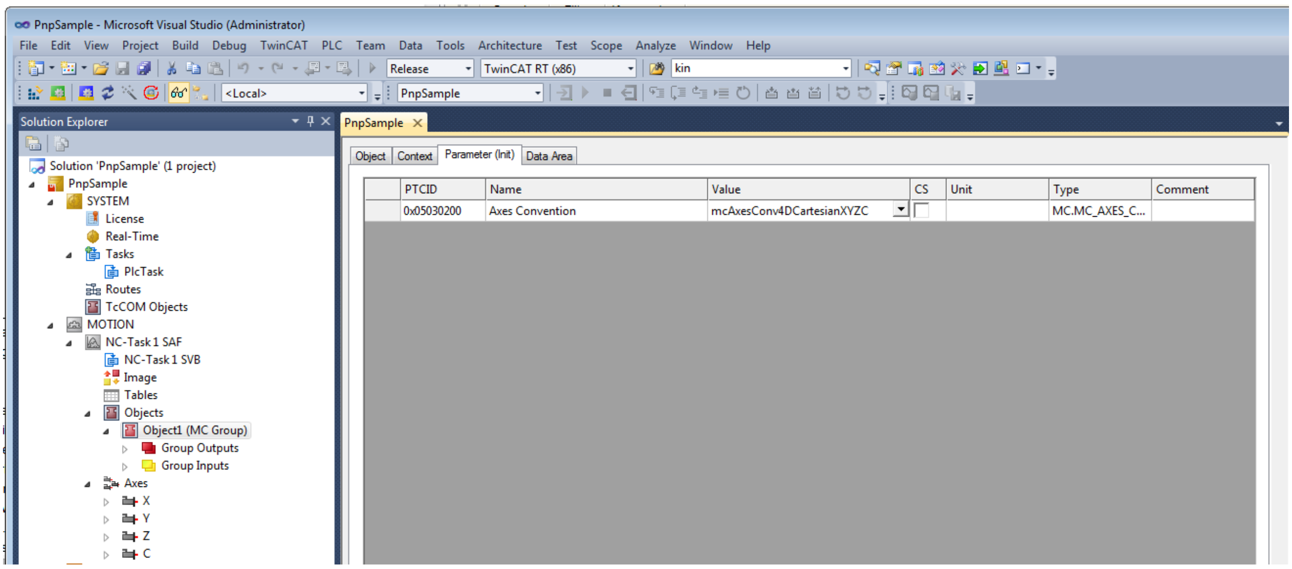
Parameter	Value	Type	Description
Blending Path Type	mcBlendingPathTypeIgnore	MC.MC_BLENDING_PATH_TYPE	No blending.
Blending Path Type	mcBlendingPathTypePoly5	MC.MC_BLENDING_PATH_TYPE	The blending path uses a fifth degree polynomial.

5.3 MC Group with Pick-and-Place

MC Group with Pick-and-Place no longer available

Instead of the MC Group with Pick-and-Place, use MC Group Coordinated Motion [▶ 18]. Up to TF5400 TwinCAT Advanced Motion Pack version 3.2.27, existing projects can be opened and executed with the MC Group with Pick-and-Place.

The MC Group connects axes in order to execute a multi-dimensional motion.



PTCID	Name	Value	CS	Unit	Type	Comment
0x05030200	Axes Convention	mcAxesConv4DCartesianXYZC	☐		MC.MC_AXES_C...	

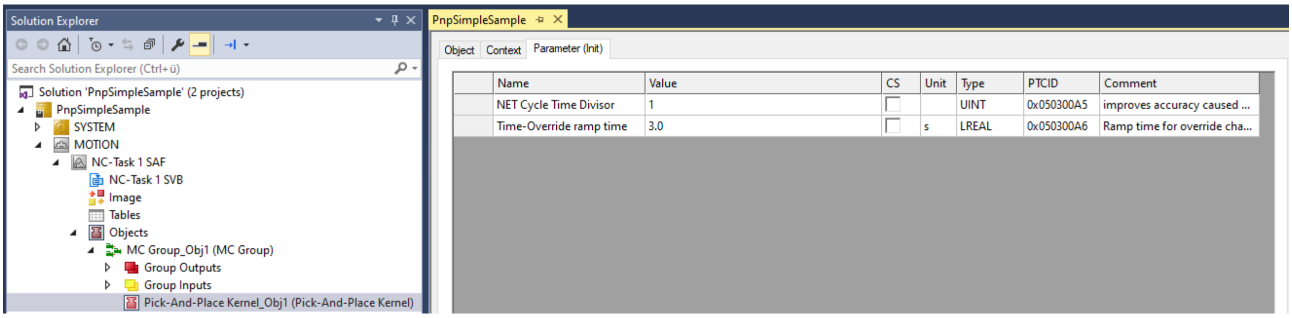
Axes conventions

Tab: Parameter (Init). Three axes conventions can be set.

The axes conventions define how the axes are interpreted in the axis group. They define the dimension of the axis group and thus the number of axes that need to be added, as well as the way in which each of the added axes is interpreted.

Parameter	Value	Type	Description
Axes Convention	mcAxesConv2DCartesianXY	MC.MC_AXES_CONVENTION	A 2D group consisting of X, Y. The order of the translatory axes in the configuration determines the order of translation.
Axes Convention	mcAxesConv3DCartesianXYZ	MC.MC_AXES_CONVENTION	A 3D group consisting of X, Y, Z. The order of the translatory axes in the configuration determines the order of translation.
Axes Convention	mcAxesConv4DCartesianXYZC	MC.MC_AXES_CONVENTION	A 4D group consisting of X, Y, Z and a rotary axis around Z (C). The order of the translatory and rotary axes in the configuration determines the order of translation and rotation.

Axis group parameters of the pick-and-place kernel



Name	Value	CS	Unit	Type	PTCID	Comment
NET Cycle Time Divisor	1	□		UINT	0x050300A5	improves accuracy caused ...
Time-Override ramp time	3.0	□	s	LREAL	0x050300A6	Ramp time for override cha...

Tab: Parameter (Init).

Parameter	Unit	Type	Description
NET Cycle Time Divisor		`UINT`	Improves accuracy on the basis of temporal discretization.
Time Override Ramp Time	`S`	`LREAL`	Ramp time for override modification from 0 % to 100 %.

The pick-and-place setpoint generator was specially developed for the requirements of pick-and-place applications. It is intended for motions where the precise path dynamics are not so important, but where the user wishes to get from one point to another as quickly as possible. It is therefore permissible for the algorithm to violate restrictions in the path dynamics within the tolerance sphere. Axis restrictions are never violated.

6 Spatial Configuration

The Spatial Configuration describes geometrical relationships between reference frames. Those relationships are of translation and rotation type.

6.1 Coordinate Frame Object

Coordinate Frame Objects can be used to hierarchically build up geometrical translation and rotation relationships. For straight interpretation the `x`-direction of the final element within the hierarchy should point into the conveying direction.

Parameter	Description	Unit
Rotation Convention	Convention used for the calculation of rotations. Default is `DIN9300 Z″Y′X` where Rotation `3` is the parameter for `Z″`, Rotation `2` is the parameter for `Y′` and Rotation `1` is the parameter for `X`.	
Definition Direction	Indicates the direction in which the displacement is programmed (from the point of view of the reference system or the MCS).	
Translation `X`	Translation in the `x`-axis direction.	mm
Translation `Y`	Translation in the `y`-axis direction.	mm
Translation `Z`	Translation in the `z`-axis direction.	mm
Rotation `1`	Rotation axis is defined by the Rotation Convention.	°
Rotation `2`	Rotation axis is defined by the Rotation Convention.	°
Rotation `3`	Rotation axis is defined by the Rotation Convention.	°

6.2 Conveyor Tracking Object

A Conveyor Tracking object can be used to synchronize an axes group with a conveyor belt. It is added as a child object to a Coordinate Frame object. While the Coordinate Frame describes the static transformation (translation and/or rotation) to the conveyor belt system, the Conveyor Tracking object handles the dynamic part of the tracking.

The Conveyor Tracking parameters are listed in the following table. The dynamics parameters are default values that are used when MC_DEFAULT is chosen for the corresponding parameter in the MC_TrackConveyorBelt function block instance.

Parameter	Description	Unit
Velocity	Default velocity for synchronization.	mm s^{-1}
Acceleration	Default acceleration for synchronization.	mm s^{-2}
Deceleration	Default deceleration for synchronization.	mm s^{-2}
Jerk	Default jerk for synchronization.	mm s^{-3}
Default Tracking Behavior	Conveyor tracking behavior after `InSync` has been reached.	
Synchronization Tolerance Distance	Distance to tracking target in which the tracking is considered synchronized (InSync = TRUE). Usage of this parameter might be useful if the master signal is noisy. This parameter only has influence if MC Group Coordinated Motion is in use.	mm

6.3 Conveyor Tracking Behavior

The Default Tracking Behavior defines the kind of default disturbance rejection during tracking. A disturbance may be an unexpected impulse or a conveyor indexing movement.

mcTrackingBehaviorDynLimited

Velocity synchronization to the `ConveyorBelt` is maintained using the given `Acceleration`, `Deceleration` and `Jerk`.
Relevant when disturbances are not known precisely or disturbance dynamics are significant.
Dynamic limits are input to the `MC_TrackConveyorBelt` function block. The values from the Conveyor Tracking Object will be used when `MC_Default` is input to the function block. When the conveyor indexes, the response will be limited by the dynamic parameters.
When the `DynLimited` setting is used, the response is compensated with the jerk limit. The function block output `MC_TrackConveyorBelt.InSync` indicates when there is synchronization.

InSync

Using the `mcTrackingBehaviorDynLimited` operation mode the `InSync = TRUE` output may disappear when the synchronized position has been lost. Staying within the parameterized dynamics the algorithm tries to return to the synchronized position on its own. When the synchronized position has been reached the `InSync = TRUE` output appears, again.

mcTrackingBehaviorStayInSync

Velocity synchronization to the `ConveyorBelt` is maintained with non-limited Acceleration, Deceleration and Jerk.
When the conveyor indexes, the tracking response will not be limited. Rather, the tracking response intends to remain synchronized and follow the conveyor unconditionally. The function block output `MC_TrackConveyorBelt.InSync` indicates when there is synchronization.

InSync

Using the `mcTrackingBehaviorStayInSync` operation mode when the `InSync` signal has once become `TRUE`, it stays `TRUE` as long as the command is active.

6.4 Node Connector Object

A Node Connector is an administrative Object, that establishes a transformation from one reference frame (node) to another.

Parameter	Description
Start node	Object ID of the starting point for the coordinate transformation. The default ID value is `0` and stands for the `WCS`, the World Coordinate System.
End node	Object ID of the end point of the coordinate transformation, e.g. a point on the Conveyor Belt.

6.5 Configuring a Node Connector

Configuring for `MC_SetCoordinateTransform` is illustrated at the example of a pallet located relative to the `WCS` or `MCS` coordinate system.

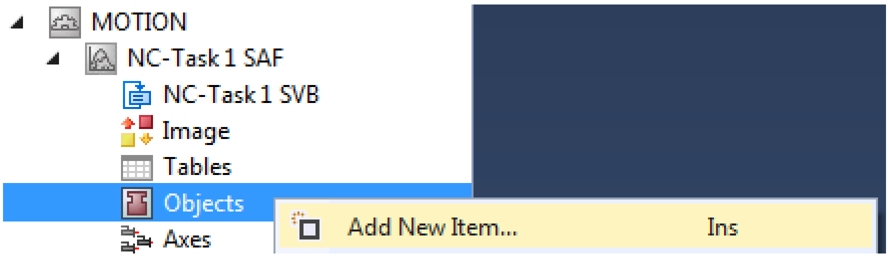
Node connector objects

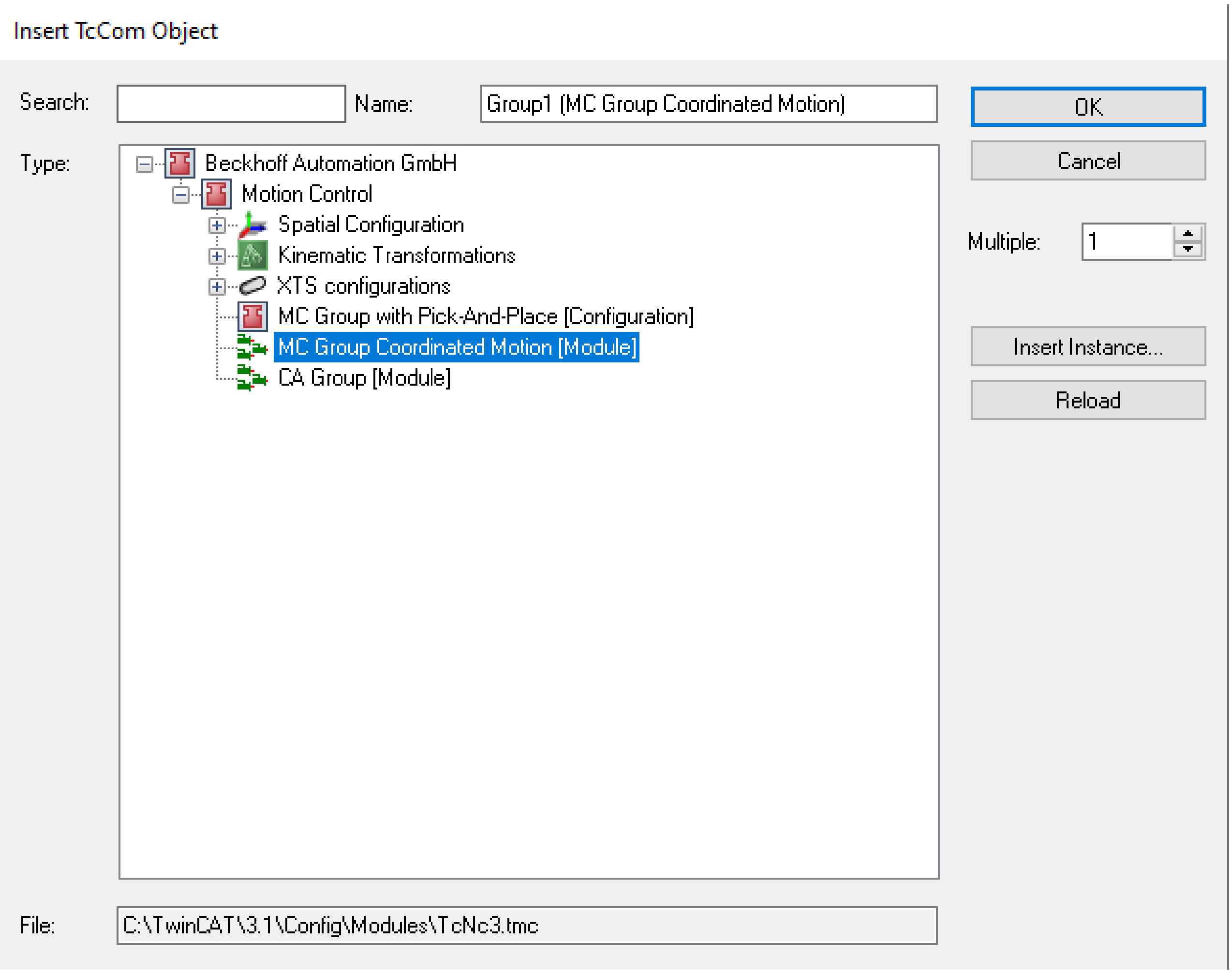
Node connector objects are used by MC_SetCoordinateTransform and MC_TrackConveyorBelt. Instead of coordinate frames, node connector objects are addressed by the PLC as representatives.

Example

To introduce a coordinate transform using `MC_SetCoordinateTransform`:

1. Insert an MC Group.




Insert TcCom Object

Search:

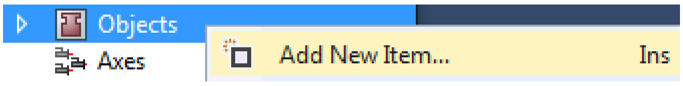
Name: Group1 (MC Group Coordinated Motion)

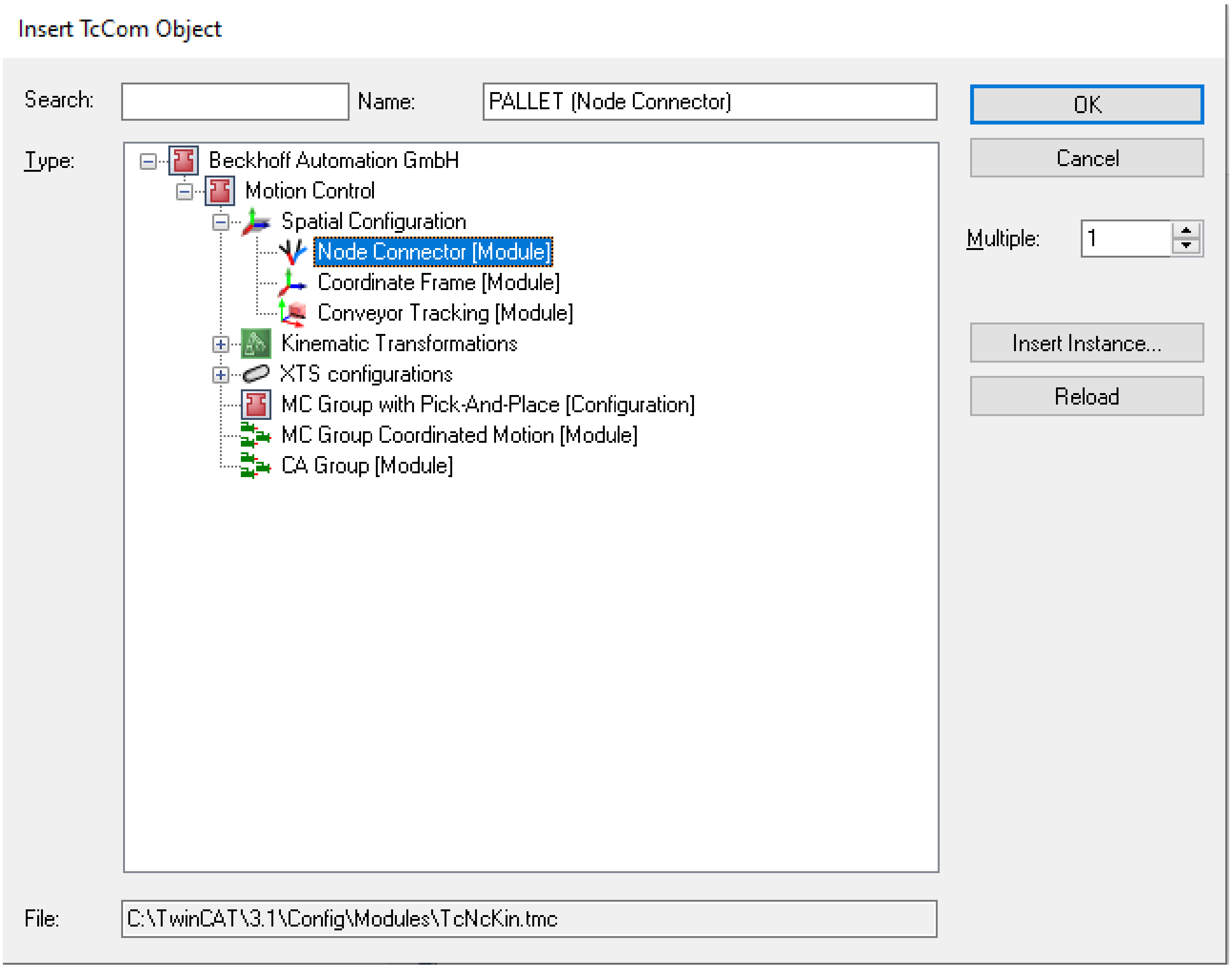
Type:
- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - MC Group Coordinated Motion [Module]
    - CA Group [Module]

Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcNc3.tmc

2. Insert a Node Connector.





Insert TcCom Object

Search:

Name: PALLET (Node Connector)

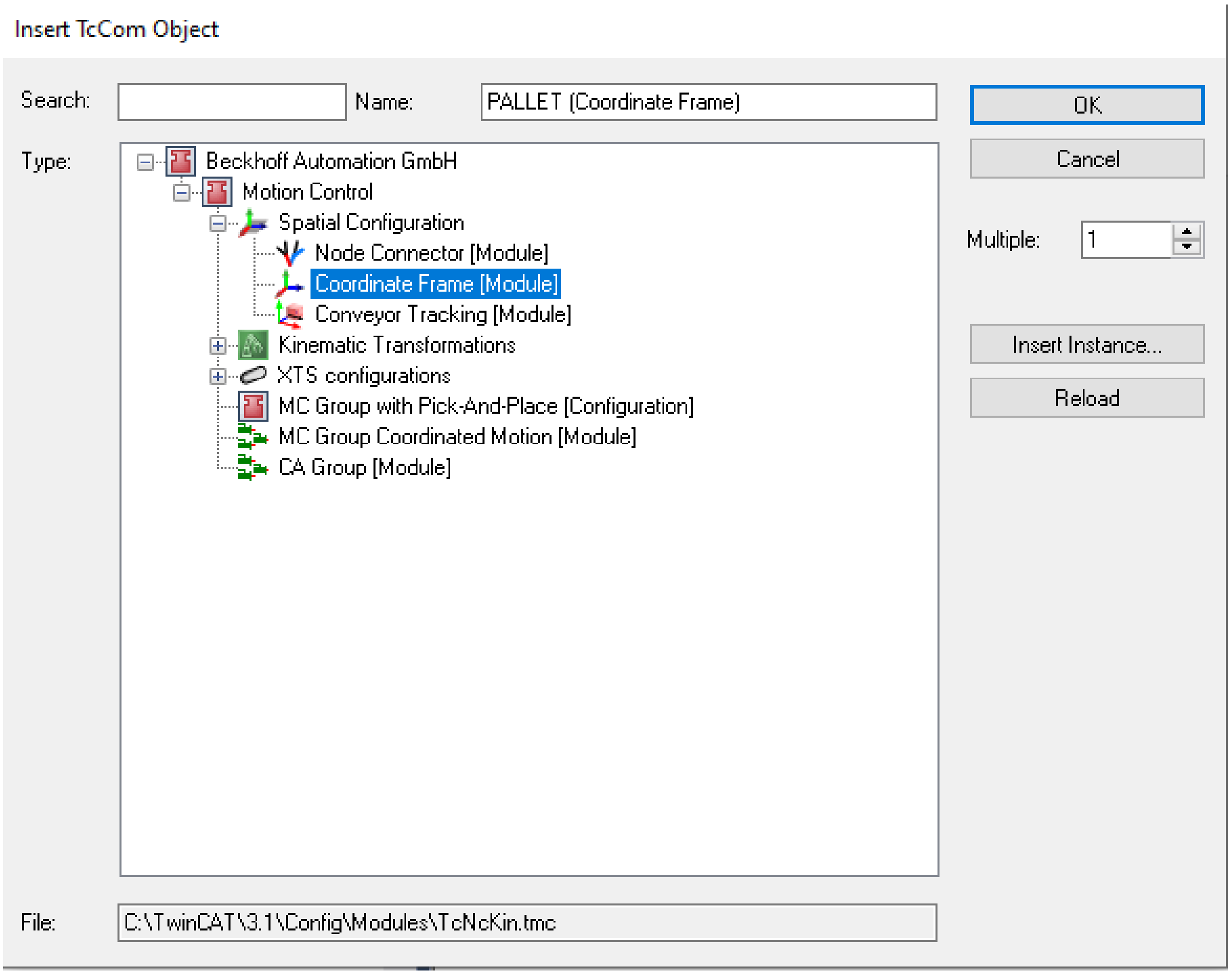
Type:

- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
      - Node Connector [Module]
      - Coordinate Frame [Module]
      - Conveyor Tracking [Module]
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - MC Group Coordinated Motion [Module]
    - CA Group [Module]

Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcNcKin.tmc

3. Insert a Coordinate Frame.



Insert TcCom Object

Search:

Name: PALLET (Coordinate Frame)

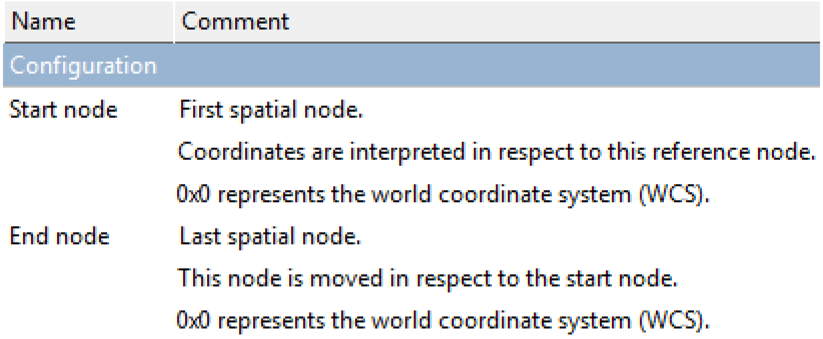
Type: Coordinate Frame [Module]

Multiple: 1

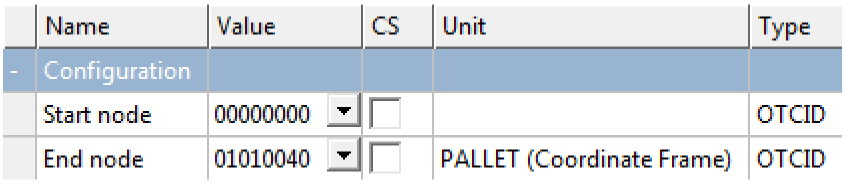
File: C:\TwinCAT\3.1\Config\Modules\TcNcKin.tmc

OK
Cancel
Insert Instance...
Reload
Beckhoff Automation GmbH
Motion Control
Spatial Configuration
Node Connector [Module]
Coordinate Frame [Module]
Conveyor Tracking [Module]
Kinematic Transformations
XTS configurations
MC Group with Pick-And-Place [Configuration]
MC Group Coordinated Motion [Module]
CA Group [Module]

4. Enter relevant Node Connector Parameters - in this example the end node refers to the pallet object identification.

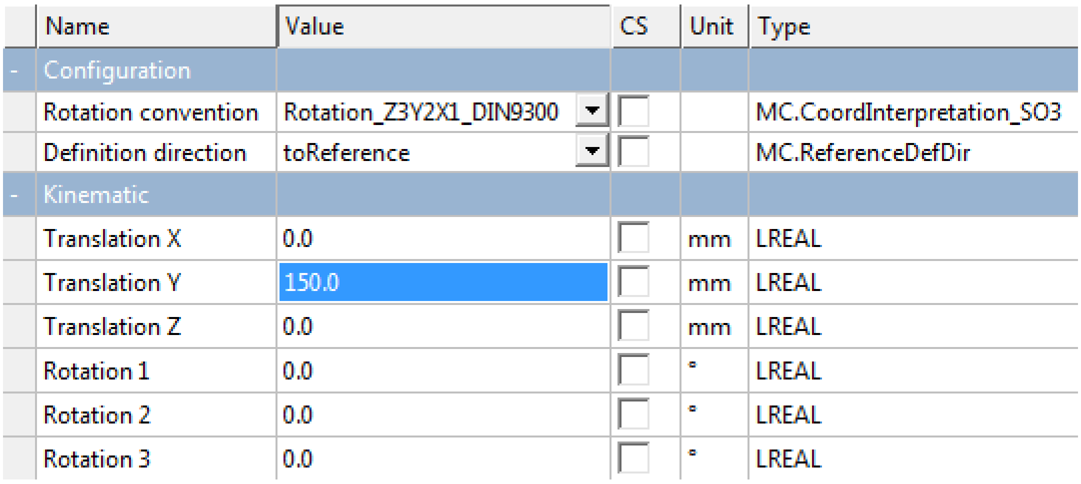


Name	Comment
Configuration	
Start node	First spatial node. Coordinates are interpreted in respect to this reference node. 0x0 represents the world coordinate system (WCS).
End node	Last spatial node. This node is moved in respect to the start node. 0x0 represents the world coordinate system (WCS).



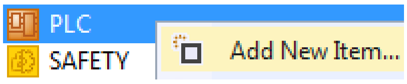
	Name	Value	CS	Unit	Type
-	Configuration				
	Start node	00000000	□		OTCID
	End node	01010040	□	PALLET (Coordinate Frame)	OTCID

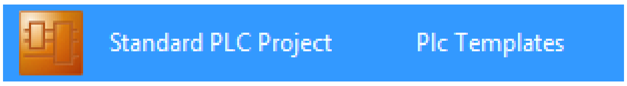
5. Enter relevant Coordinate Frame Parameters.

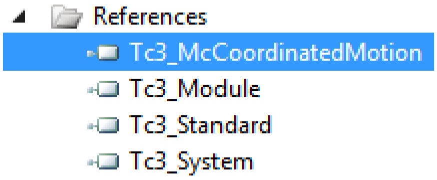


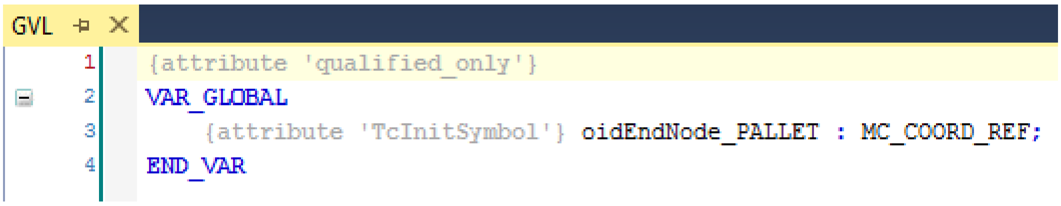
Name	Value	CS	Unit	Type
- Configuration				
Rotation convention	Rotation_Z3Y2X1_DIN9300	□		MC.CoordInterpretation_SO3
Definition direction	toReference	□		MC.ReferenceDefDir
- Kinematic				
Translation X	0.0	□	mm	LREAL
Translation Y	150.0	□	mm	LREAL
Translation Z	0.0	□	mm	LREAL
Rotation 1	0.0	□	°	LREAL
Rotation 2	0.0	□	°	LREAL
Rotation 3	0.0	□	°	LREAL

6. Link the inserted Node Connector to the `PLC`.

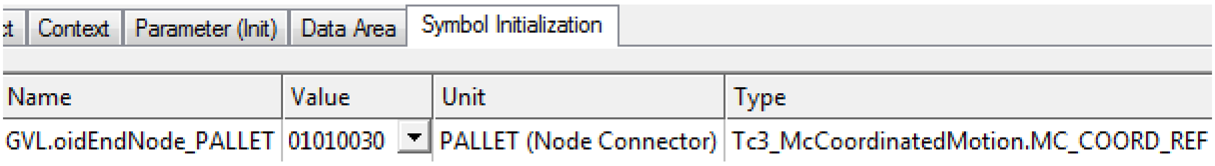






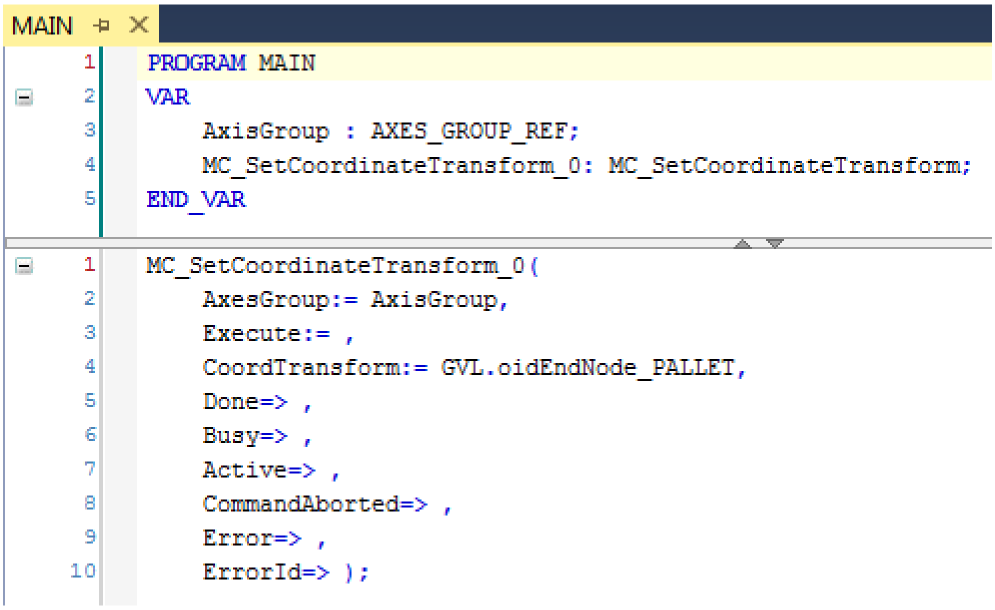


```
{attribute 'qualified_only'}
VAR_GLOBAL
    {attribute 'TcInitSymbol'} oidEndNode_PALLET : MC_COORD_REF;
END_VAR
```



Name	Value	Unit	Type
GVL.oidEndNode_PALLET	01010030	PALLET (Node Connector)	Tc3_McCoordinatedMotion.MC_COORD_REF

⇨ Finally, you can insert the `MC_SetCoordinateTransform` function block.



```
PROGRAM MAIN
VAR
    AxisGroup : AXES_GROUP_REF;
    MC_SetCoordinateTransform_0: MC_SetCoordinateTransform;
END_VAR
```

```
MC_SetCoordinateTransform_0(
    AxesGroup:= AxisGroup,
    Execute:= ,
    CoordTransform:= GVL.oidEndNode_PALLET,
    Done=> ,
    Busy=> ,
    Active=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorId=> );
```

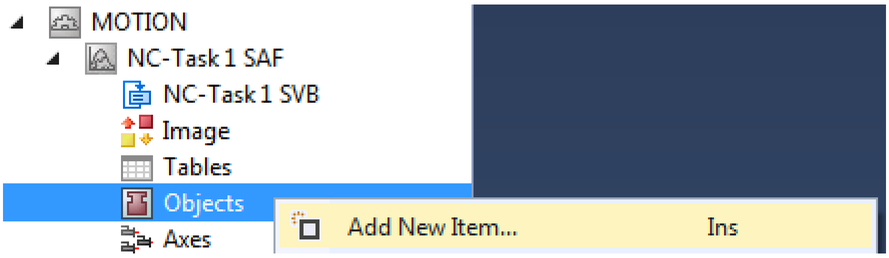
The axis group AxisGroup is linked with the Pick-and-Place function blocks.

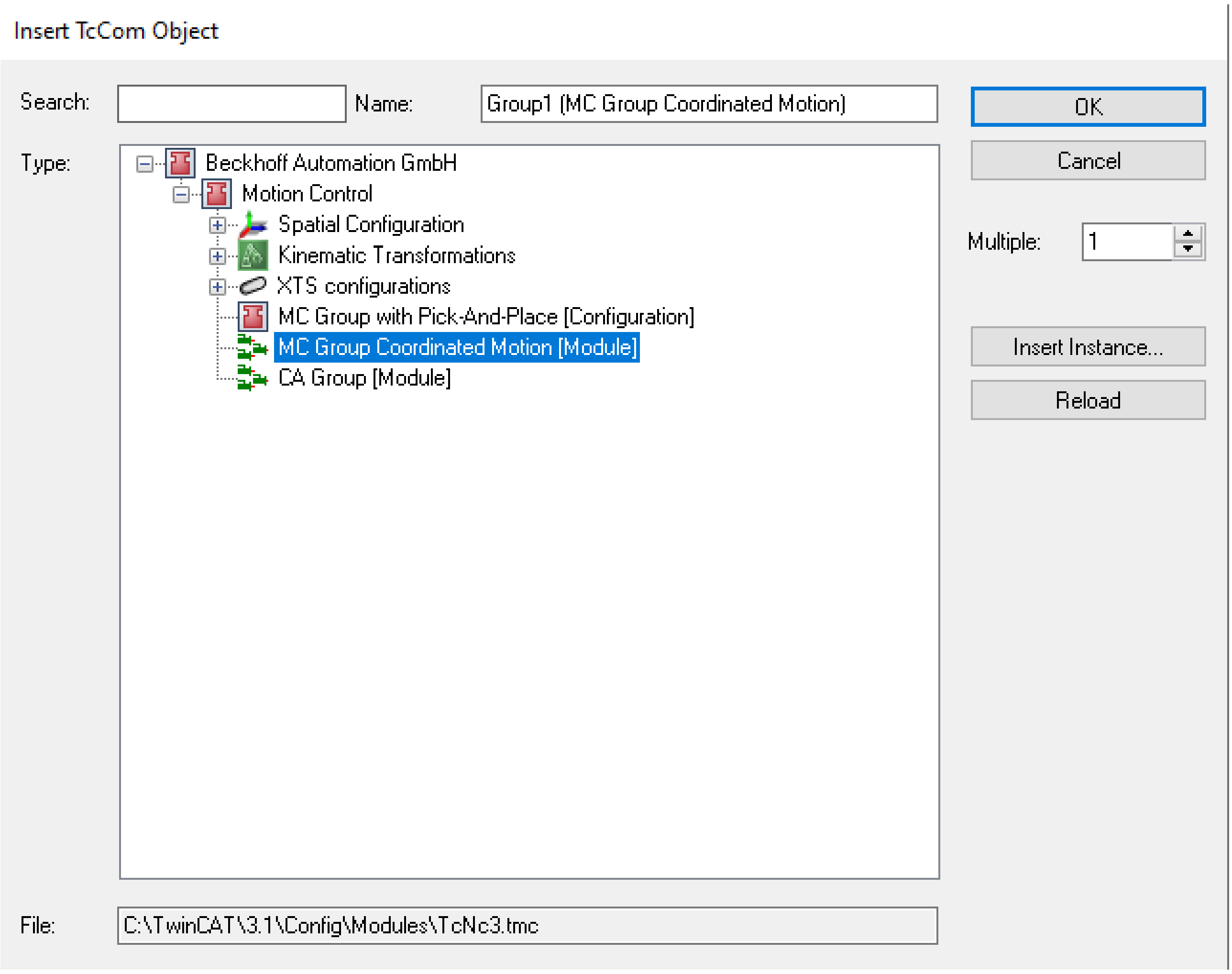
For axis movements a move command has to be programmed, e.g. MovePath.

6.6 Configure for MC_TrackConveyorBelt

To track a conveyor belt using `MC_TrackConveyorBelt`:

1. Insert an `MC` Group.





Insert TcCom Object

Search:

Name: Group1 (MC Group Coordinated Motion)

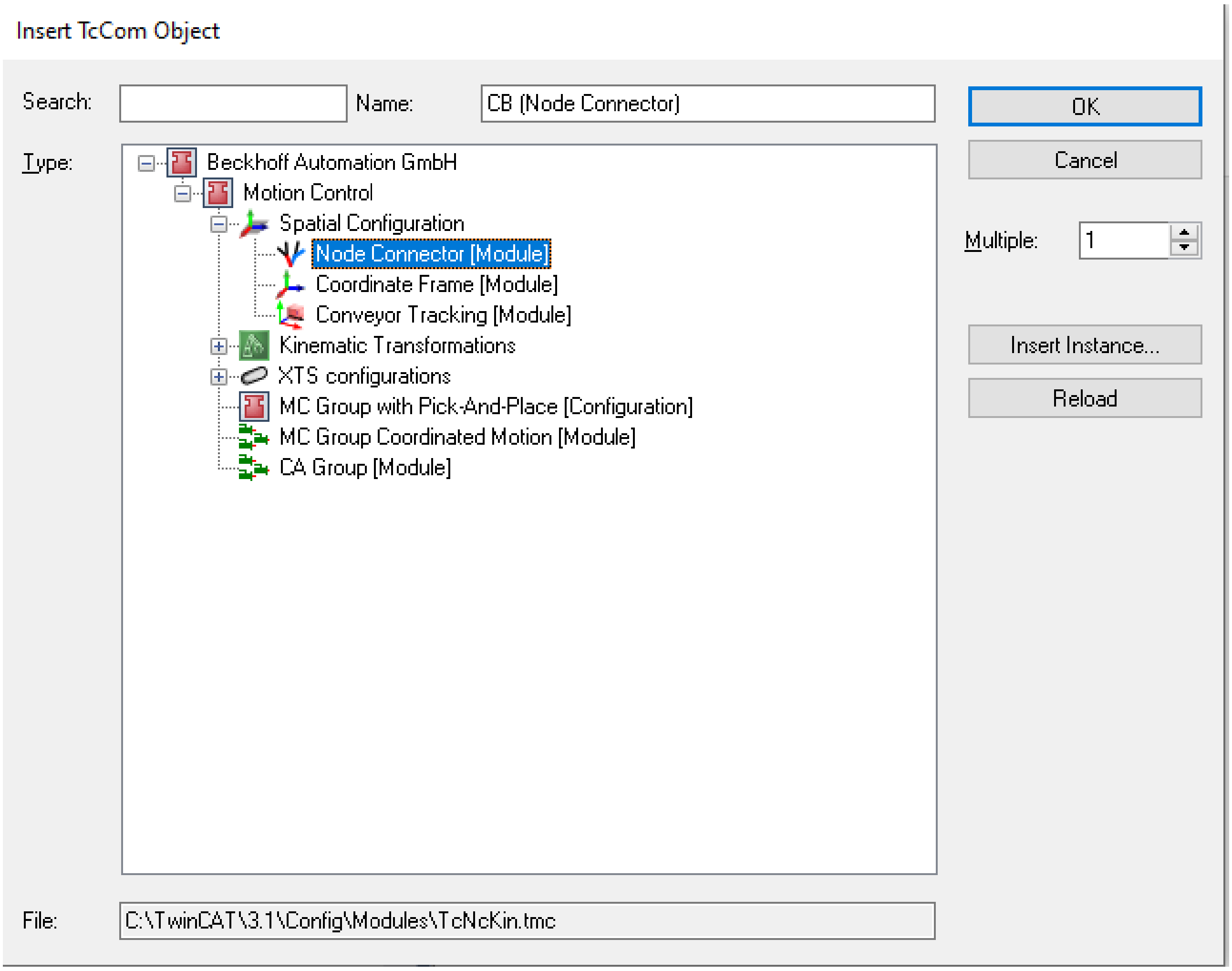
Type:

- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - MC Group Coordinated Motion [Module]
    - CA Group [Module]

Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcNc3.tmc

2. Insert a Node Connector.



Insert TcCom Object

Search:

Name: CB (Node Connector)

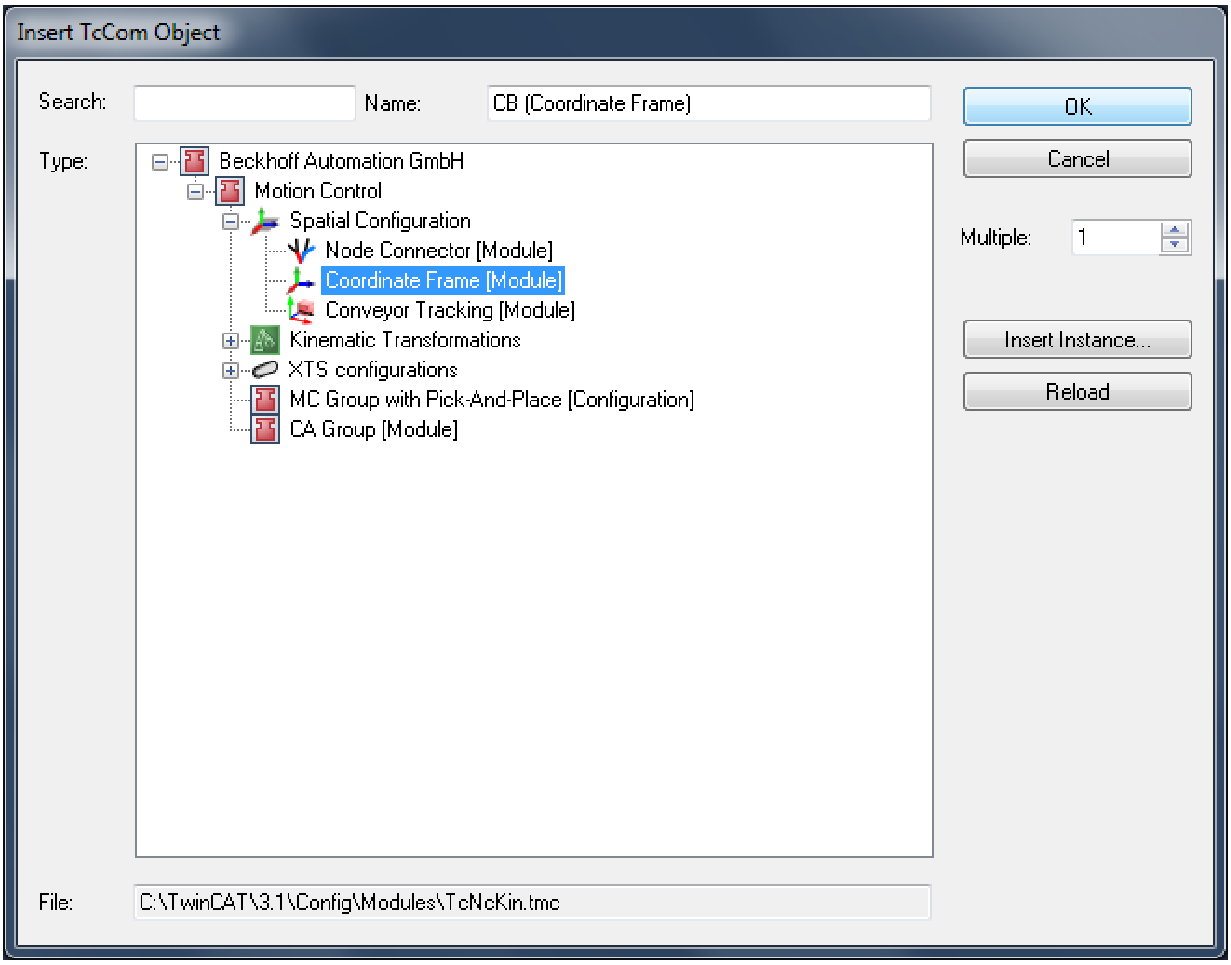
Type:

- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
      - Node Connector [Module]
      - Coordinate Frame [Module]
      - Conveyor Tracking [Module]
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - MC Group Coordinated Motion [Module]
    - CA Group [Module]

Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcNcKin.tmc

3. Insert a Coordinate Frame.



Insert TcCom Object

Search:

Name: CB (Coordinate Frame)

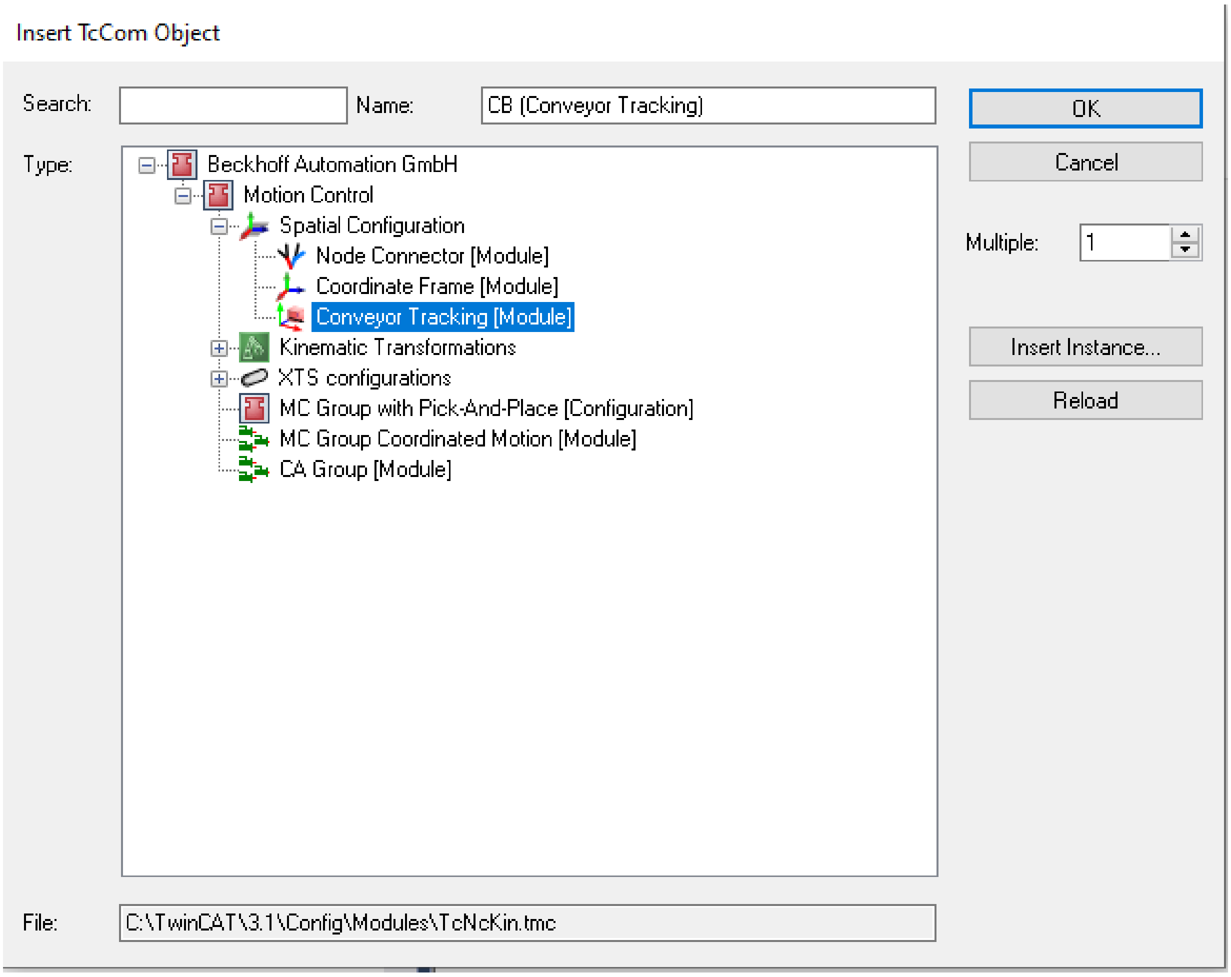
Type:

- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
      - Node Connector [Module]
      - Coordinate Frame [Module]
      - Conveyor Tracking [Module]
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - CA Group [Module]

Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcNcKin.tmc

4. Insert Conveyor Tracking. Firstly, a Coordinate Frame has been created. Secondly, the Conveyor Tracking Object has to be added as a child element to the Coordinate Frame created previously.



Insert TcCom Object

Search:

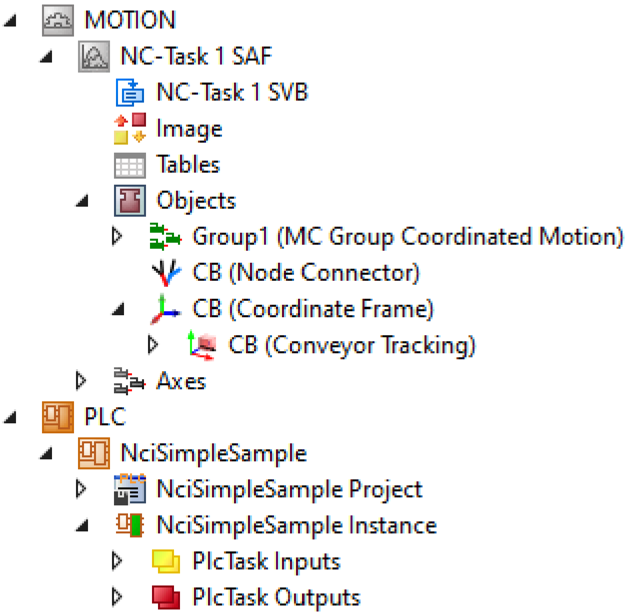
Name: CB (Conveyor Tracking)

Type:

- Beckhoff Automation GmbH
  - Motion Control
    - Spatial Configuration
      - Node Connector [Module]
      - Coordinate Frame [Module]
      - Conveyor Tracking [Module]
    - Kinematic Transformations
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]
    - MC Group Coordinated Motion [Module]
    - CA Group [Module]

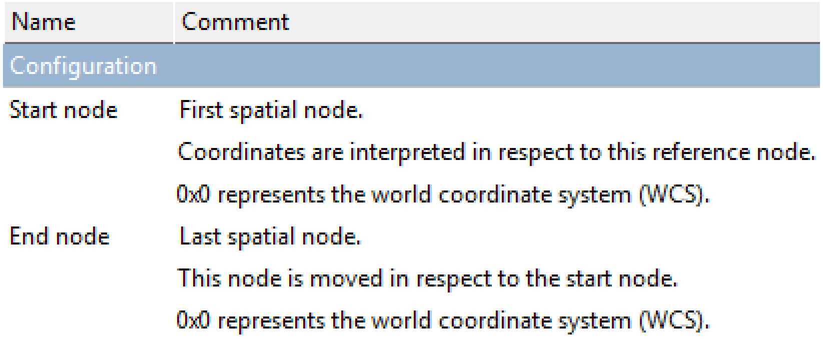
Multiple: 1

File: C:\TwinCAT\3.1\Config\Modules\TcNcKin.tmc

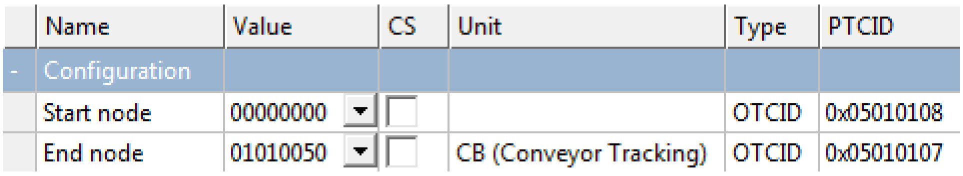


- MOTION
  - NC-Task 1 SAF
    - NC-Task 1 SVB
    - Image
    - Tables
    - Objects
      - Group1 (MC Group Coordinated Motion)
      - CB (Node Connector)
      - CB (Coordinate Frame)
        - CB (Conveyor Tracking)
    - Axes
- PLC
  - NciSimpleSample
    - NciSimpleSample Project
    - NciSimpleSample Instance
      - PlcTask Inputs
      - PlcTask Outputs

5. Enter relevant Node Connector Parameters - the end node refers to the conveyor tracking object identification.

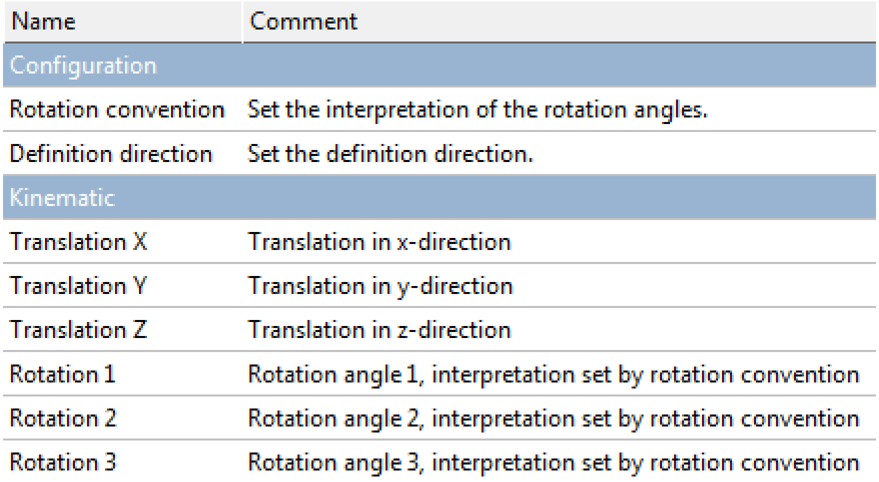


Name	Comment
Configuration	
Start node	First spatial node. Coordinates are interpreted in respect to this reference node. 0x0 represents the world coordinate system (WCS).
End node	Last spatial node. This node is moved in respect to the start node. 0x0 represents the world coordinate system (WCS).

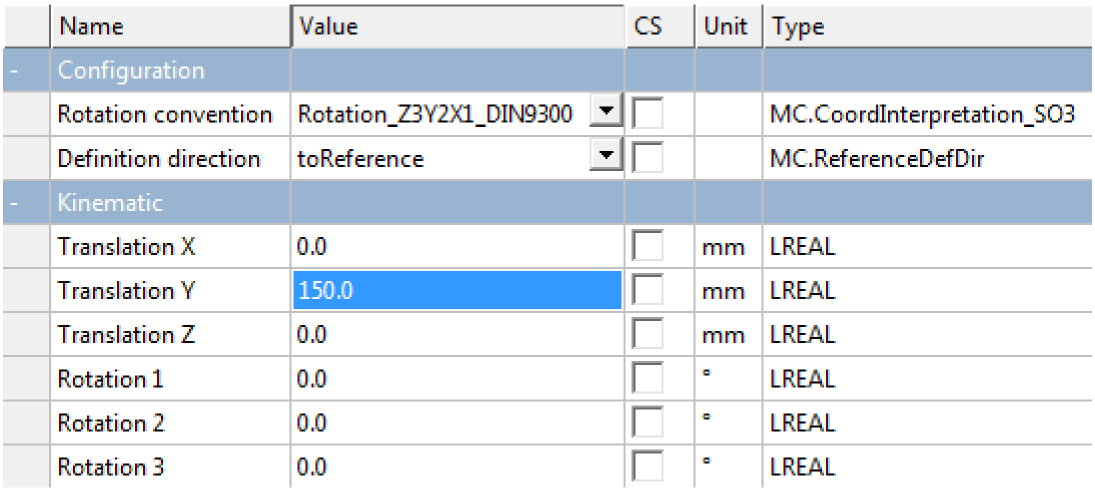


	Name	Value	CS	Unit	Type	PTCID
-	Configuration					
	Start node	00000000	□		OTCID	0x05010108
	End node	01010050	□	CB (Conveyor Tracking)	OTCID	0x05010107

6. Enter relevant Coordinate Frame Parameters.

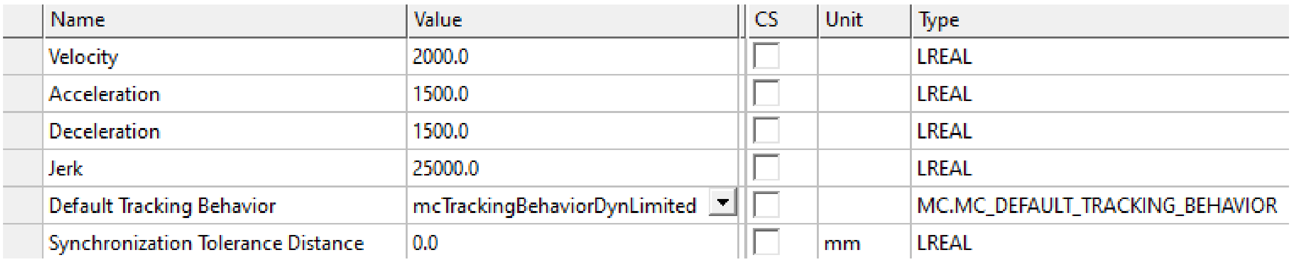


Name	Comment
Configuration	
Rotation convention	Set the interpretation of the rotation angles.
Definition direction	Set the definition direction.
Kinematic	
Translation X	Translation in x-direction
Translation Y	Translation in y-direction
Translation Z	Translation in z-direction
Rotation 1	Rotation angle 1, interpretation set by rotation convention
Rotation 2	Rotation angle 2, interpretation set by rotation convention
Rotation 3	Rotation angle 3, interpretation set by rotation convention



	Name	Value	CS	Unit	Type
-	Configuration				
	Rotation convention	Rotation_Z3Y2X1_DIN9300	□		MC.CoordInterpretation_SO3
	Definition direction	toReference	□		MC.ReferenceDefDir
-	Kinematic				
	Translation X	0.0	□	mm	LREAL
	Translation Y	150.0	□	mm	LREAL
	Translation Z	0.0	□	mm	LREAL
	Rotation 1	0.0	□	°	LREAL
	Rotation 2	0.0	□	°	LREAL
	Rotation 3	0.0	□	°	LREAL

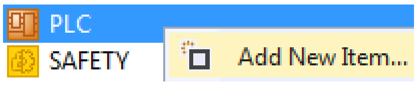
7. Enter relevant Conveyor Tracking Parameters.

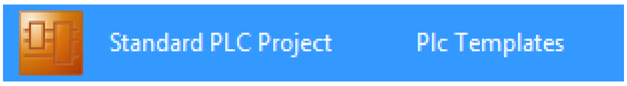


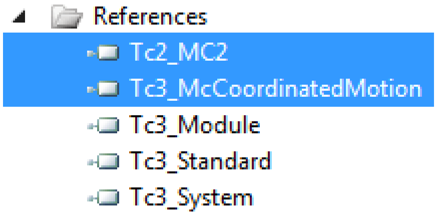
Name	Value	CS	Unit	Type
Velocity	2000.0	☐		LREAL
Acceleration	1500.0	☐		LREAL
Deceleration	1500.0	☐		LREAL
Jerk	25000.0	☐		LREAL
Default Tracking Behavior	mcTrackingBehaviorDynLimited	☐		MC.MC_DEFAULT_TRACKING_BEHAVIOR
Synchronization Tolerance Distance	0.0	☐	mm	LREAL

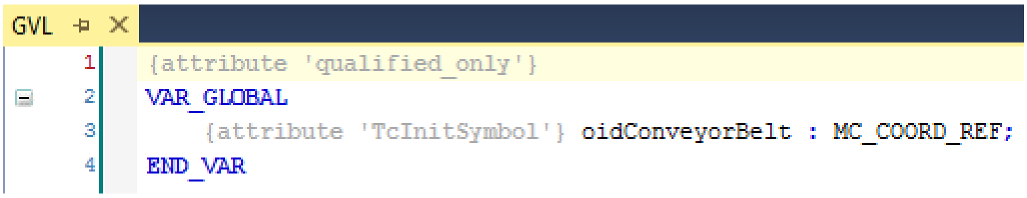
The Default Tracking Behavior specifies whether, after `InSync` has been reached for the first time, the tracking movement is still limited by the specified dynamic limits (`InSync` may be lost again) or synchronization is forced (even if the dynamic limits need to be violated in order to do so).

8. Link the Node Connector to the `PLC`.

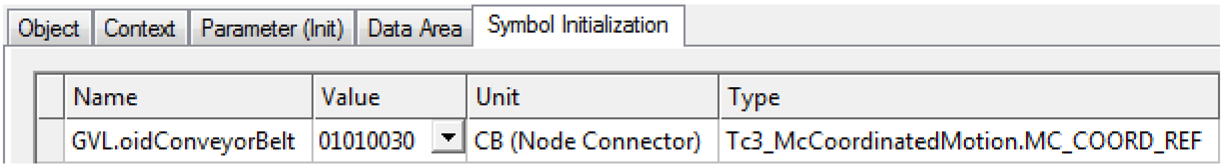






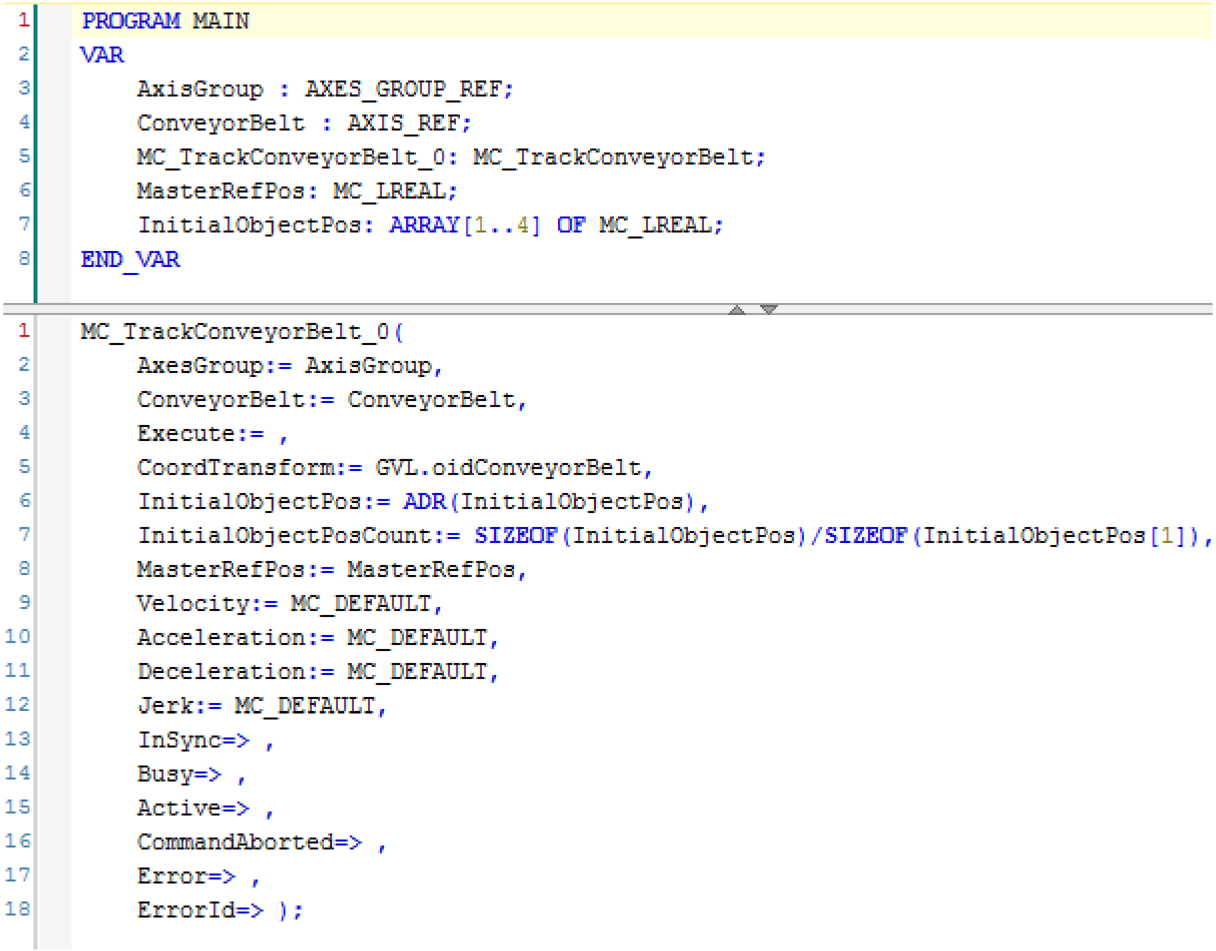


```
{attribute 'qualified_only'}
VAR_GLOBAL
    {attribute 'TcInitSymbol'} oidConveyorBelt : MC_COORD_REF;
END_VAR
```



Name	Value	Unit	Type
GVL.oidConveyorBelt	01010030	CB (Node Connector)	Tc3_McCoordinatedMotion.MC_COORD_REF

⇨ Finally, you can insert the `MC_TrackConveyorBelt` function block.



```
PROGRAM MAIN
VAR
    AxisGroup : AXES_GROUP_REF;
    ConveyorBelt : AXIS_REF;
    MC_TrackConveyorBelt_0: MC_TrackConveyorBelt;
    MasterRefPos: MC_LREAL;
    InitialObjectPos: ARRAY[1..4] OF MC_LREAL;
END_VAR
```

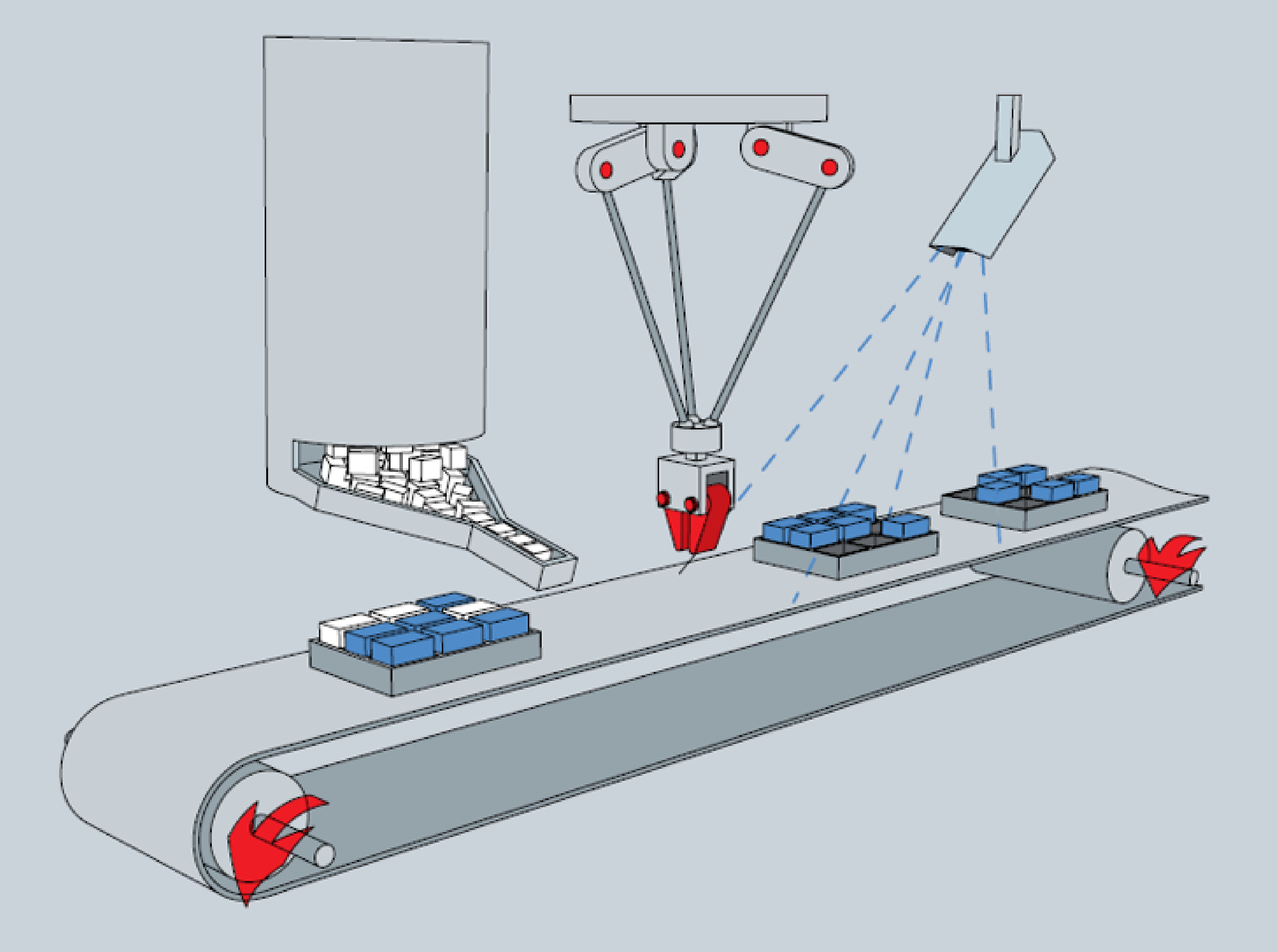
```
MC_TrackConveyorBelt_0(
    AxesGroup:= AxisGroup,
    ConveyorBelt:= ConveyorBelt,
    Execute:= ,
    CoordTransform:= GVL.oidConveyorBelt,
    InitialObjectPos:= ADR(InitialObjectPos),
    InitialObjectPosCount:= SIZEOF(InitialObjectPos)/SIZEOF(InitialObjectPos[1]),
    MasterRefPos:= MasterRefPos,
    Velocity:= MC_DEFAULT,
    Acceleration:= MC_DEFAULT,
    Deceleration:= MC_DEFAULT,
    Jerk:= MC_DEFAULT,
    InSync=> ,
    Busy=> ,
    Active=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorId=> );
```

6.7 Background Information

Coordinate systems – relationships

- **WCS**
  World Coordinate System.
- **MCS**
  Machine Coordinate System.
- **UCS**
  User Coordinate System.
- **PCS**
  Programmed Coordinate System. Workpiece.

Pick-and-Place: From warehouse to carrier

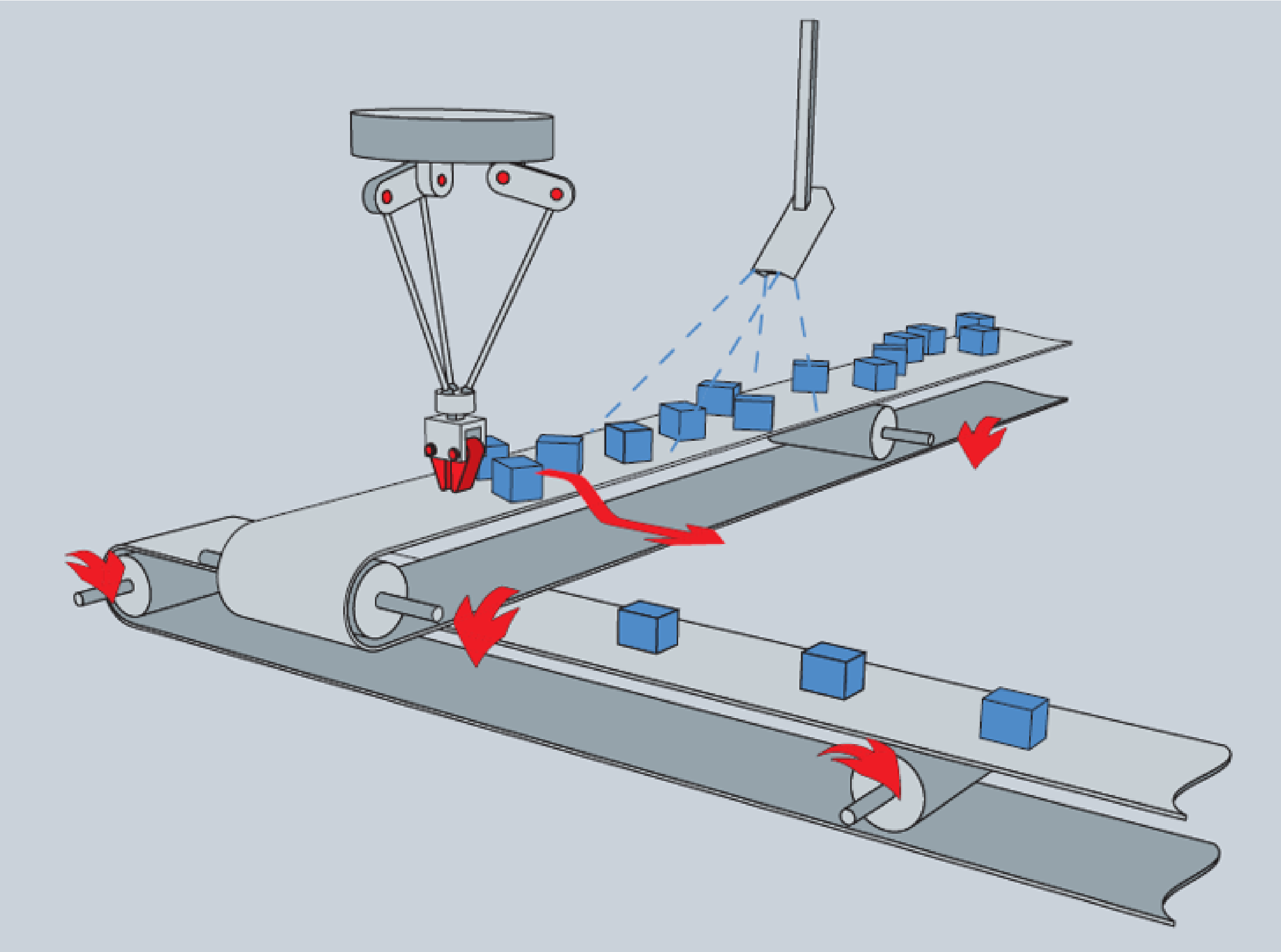


The workpieces must be taken from the warehouse and placed in the free carrier positions running on the conveyor belt.

Thereby,

- the storage place is defined within the `WCS`,
- the robot is located somewhere within the `WCS`,
- the robot can be controlled within its `MCS`,
- the conveyor belt is located somewhere in the `WCS`,
- on the conveyor belt a carrier can be located within the `UCS`,
- a workpiece can be located within its carrier within the `PCS`.

Pick-and-Place: From conveyor belt towards conveyor belt



The workpieces have to be taken from the upper conveyor belt and placed on the lower conveyor belt.

Thereby,

- the robot is located somewhere within the `WCS`,
- the robot can be controlled within its `MCS`,
- each conveyor belt is located somewhere in the `WCS`,
- workpieces on the conveyor belt can be located within a `UCS`.

7 PLC Libraries

7.1 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 [▸ 40]	Adds an axis group.	✓	✗ (✓ up to v3.2)	✓
MC_GroupDisable [▸ 43]	Disables an axis group.	✓	✗ (✓ up to v3.2)	✓
MC_GroupEnable [▸ 44]	Enables an axis group.	✓	✗ (✓ up to v3.2)	✓
MC_GroupReadError [▸ 45]	Reads the error ID of a group.	✓	✗ (✓ up to v3.2)	✓
MC_GroupReadStatus [▸ 47]	Reads the group status.	✓	✗ (✓ up to v3.2)	✓
MC_GroupReset [▸ 48]	Resets a group.	✓	✗ (✓ up to v3.2)	✓
MC_GroupSetOverride [▸ 50]	Sets the override of a group and returns the actual override value.	✗	✗ (✓ up to v3.2)	✓
MC_RemoveAxisFromGroup [▸ 51]	Removes an axis from a group.	✓	✗ (✓ up to v3.2)	✓
MC_SetCoordinateTransform [▸ 53]	Activates a reference system.	✗	✗ (✓ up to v3.2)	✓
MC_SetCoordinateTransformPreparation [▸ 54]	Adds a change of reference system to the list of movement segments.	✗	✗	✓

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_TrackConveyorBelt [▶ 55]	Assists in synchronizing velocity to an object moving along a straight line through space.	✗	✗ (✓ up to v3.2)	✓
MC_UngroupAllAxes [▶ 58]	Disables a group and removes all axes.	✓	✗ (✓ up to v3.2)	✓
UDINT_TO_IDENTINGROUP [▶ 59]	Converts an integer value to IDENT_IN_GROUP_REF, so axes without special interpretation can be added to a group.	✓	✗	✓
Motion				
MC_GroupHalt [▶ 60]	Stops a group without locking it for further motion commands.	✓	✗	✓
MC_GroupStop [▶ 62]	Stops a group and locks it for further motion commands.	✓	✗ (✓ up to v3.2)	✓
MC_MoveLinearAbsolutePreparation [▶ 64]	Adds an absolute linear movement to a list of motion segments.	✗	✗ (✓ up to v3.2)	✓
MC_MoveCircularAbsolutePreparation [▶ 66]	Adds an absolute circular movement to a list of motion segments.	✗	✗ (✓ up to v3.2)	✓
MC_MovePath [▶ 70]	Executes a list of motion segments.	✗	✗ (✓ up to v3.2)	✓
MC_BlockerPreparation [▶ 71]	Appends a blocking job to the list of segments in the structure PathData.	✗	✗	✓
MC_ReleaseBlocker [▶ 73]	Resolves a blocking job that is blocking further execution of the path.	✗	✗	✓
MC_GroupReadBlockerStatus [▶ 74]	Reads the current blocker status.	✗	✗	✓
MC_DwellTimePreparation [▶ 75]	Appends a standstill job with a defined time to the list of segments in the structure PathData.	✗	✗	✓

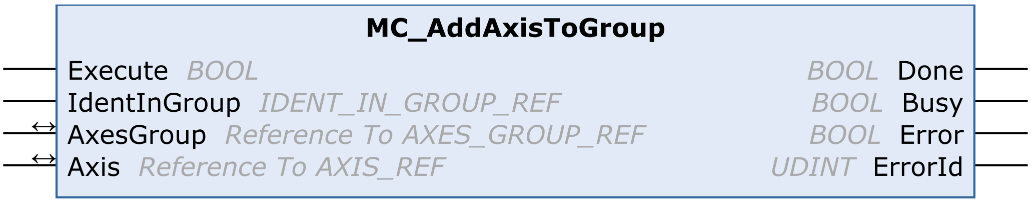
Structures and enumerations

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 [▶ 76]	Defines how an axis is interpreted in a group.	✗	✗ (✓ up to v3.2)	✓
MC_CIRC_MODE [▶ 77]	The circle mode defines which definition is used to program a circle.	✗	✗ (✓ up to v3.2)	✓
MC_CIRC_PATHCHOICE [▶ 81]	The data type defines the rotation direction of a circle.	✗	✗ (✓ up to v3.2)	✓
MC_PATH_DATA_REF [▶ 82]	Represents the path to be executed at MC_MovePath [▶ 70].	✗	✗ (✓ up to v3.2)	✓
ClearPath [▶ 83]	Resets the path represented by MC_PATH_DATA_REF [▶ 82].	✗	✗ (✓ up to v3.2)	✓
MC_TRANSITION_MODE [▶ 83]	Characterizes the way a segment transition is executed.	✗	✗ (✓ up to v3.2)	✓
MC_COORD_REF [▶ 85]	Object ID of a coordinate system.	✗	✗ (✓ up to v3.2)	✓

7.1.1 Function Blocks

7.1.1.1 Administrative

7.1.1.1.1 MC_AddAxisToGroup


MC_AddAxisToGroup
Execute BOOL
IdentInGroup IDENT_IN_GROUP_REF
AxesGroup Reference To AXES_GROUP_REF
Axis Reference To AXIS_REF
BOOL Done
BOOL Busy
BOOL Error
UDINT ErrorId

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 [▸ 76] (e.g. MCS_X) must be used. For Collision Avoidance the function UDINT_TO_IDENTINGROUP [▸ 59] 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 [▸ 96]).
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.

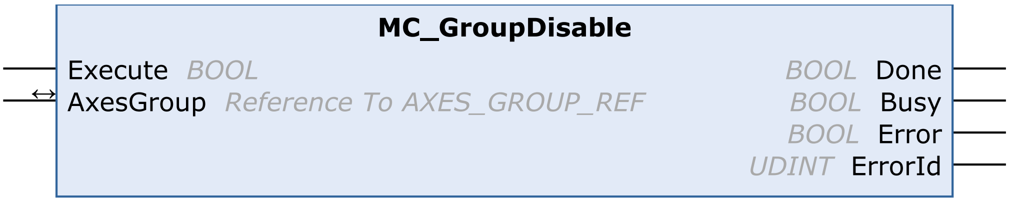
```
VAR_GLOBAL CONSTANT
    cAxesCount            : UDINT:=10;
END_VAR
VAR
    stGroupRef            : AXES_GROUP_REF; // link to CA Group
    stAxis                : ARRAY[1..cAxesCount] OF AXIS_REF;
    fbAddAxis             : ARRAY[1..cAxesCount] OF MC_AddAxisToGroup;
    i                     : UDINT;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

7.1.1.1.2 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).

As of TF5400 V3.3.72, PTP commands of the Tc2_MC2 library (e.g. MC_MoveAbsolute) are permitted on axes of the MC Group Coordinated Motion in the GroupDisabled state.

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 [▸ 60] or MC_GroupStop [▸ 62] 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

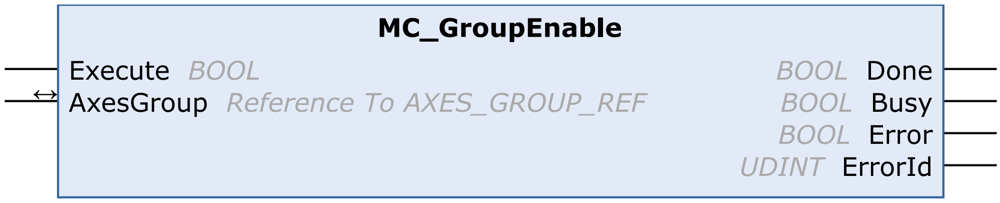
```
VAR_OUTPUT
    Done                    : BOOL;
    Busy                    : BOOL;
    Error                   : BOOL;
    ErrorId                 : UDINT;
END_VAR
```

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

7.1.1.1.3 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).

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 [▶ 96])

Outputs

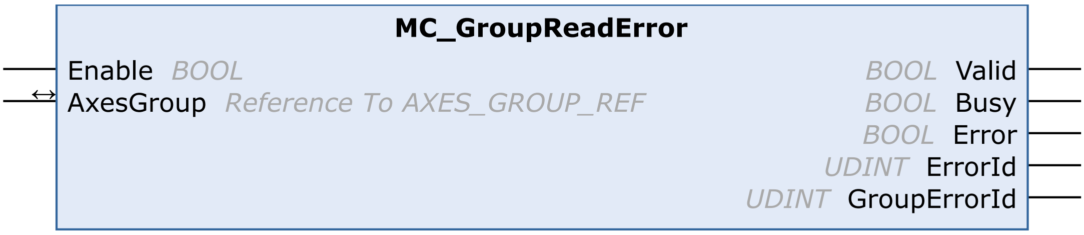
```
VAR_OUTPUT
    Done                  : BOOL;
    Busy                  : BOOL;
    Error                 : BOOL;
    ErrorId               : UDINT;
END_VAR
```

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

7.1.1.1.4 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

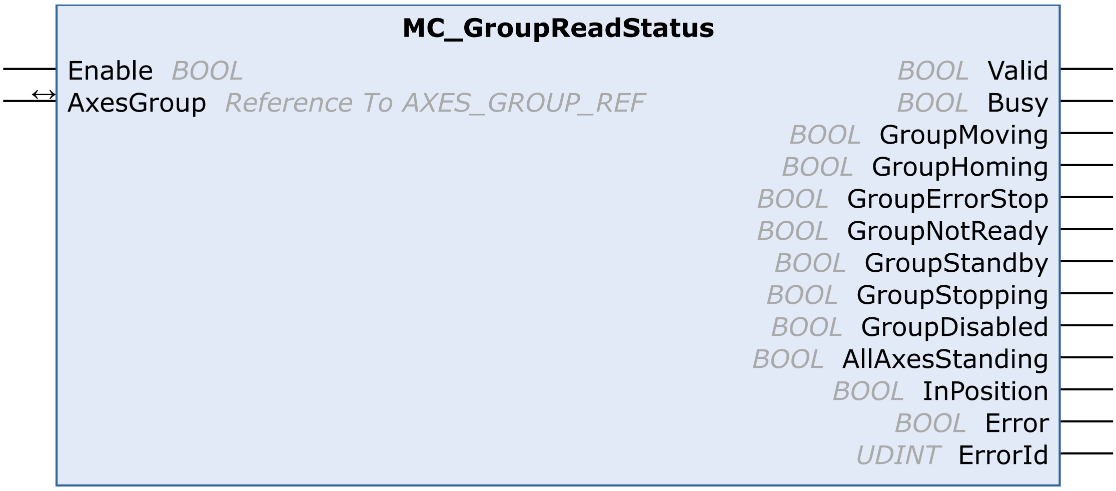
```
VAR_OUTPUT
    Valid        : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorId      : UDINT;
    GroupErrorId : UDINT;
END_VAR
```

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

7.1.1.1.5 MC_GroupReadStatus



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

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

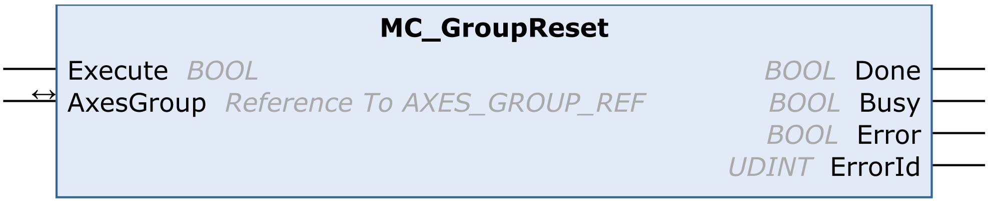
```
VAR_OUTPUT
    Valid                     : BOOL;
    Busy                      : BOOL;
    GroupMoving               : BOOL;
    GroupHoming               : BOOL;
    GroupErrorStop            : BOOL;
    GroupNotReady             : BOOL;
    GroupStandby              : BOOL;
    GroupStopping             : 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).
GroupHoming	BOOL	The group is in the GroupHoming state (see State diagrams).
GroupErrorStop	BOOL	The group is in the GroupErrorStop state (see State diagrams).
GroupNotReady	BOOL	The group is in the GroupNotReady state (see State diagrams).
GroupStandby	BOOL	The group is in the GroupStandby state (see State diagrams).
GroupStopping	BOOL	The group is in the GroupStopping state (see State diagrams).
GroupDisabled	BOOL	The group is in the GroupDisabled state (see State diagrams).
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

7.1.1.1.6 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).

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

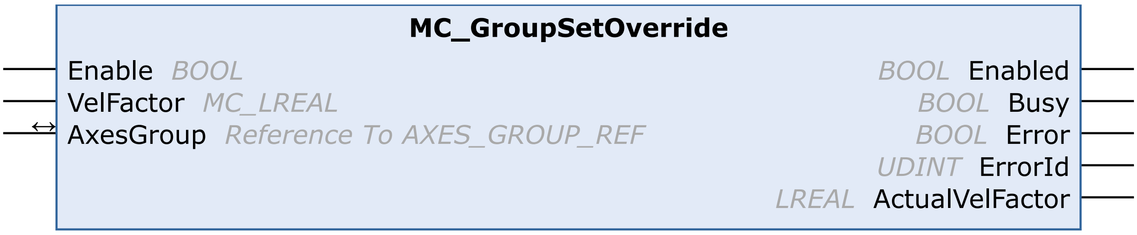
```
VAR_OUTPUT
    Done                     : BOOL;
    Busy                     : BOOL;
    Error                    : BOOL;
    ErrorId                  : UDINT;
END_VAR
```

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

7.1.1.1.7 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

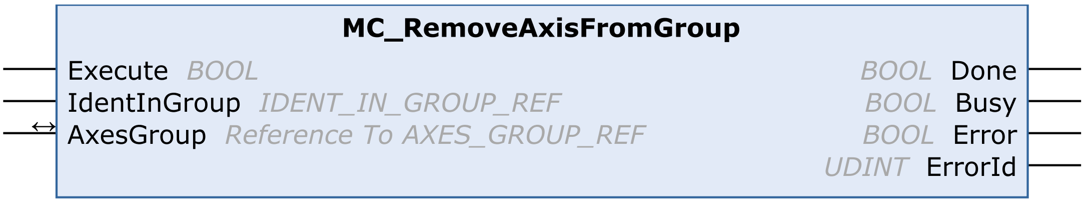
```
VAR
    stGroupRef              : AXES_GROUP_REF;
    fbSetOverride           : MC_GroupSetOverride;
END_VAR
```

```
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

7.1.1.1.8 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 [▶ 76] (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP [▶ 59].

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

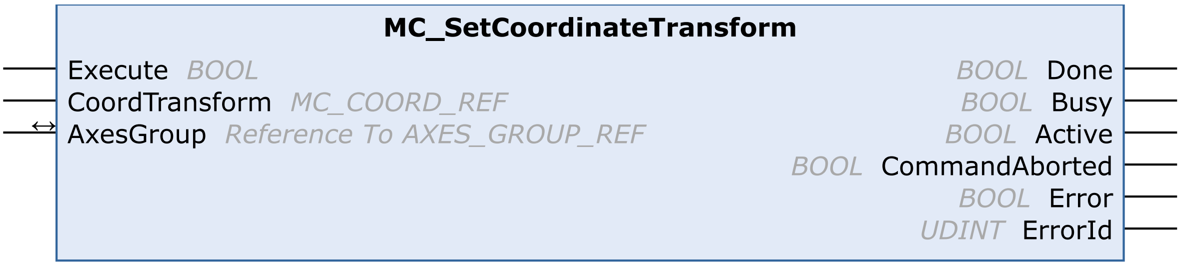
```
VAR_OUTPUT
    Done                    : BOOL;
    Busy                    : BOOL;
    Error                   : BOOL;
    ErrorId                 : UDINT;
END_VAR
```

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

7.1.1.1.9 MC_SetCoordinateTransform


MC_SetCoordinateTransform
Execute BOOL
CoordTransform MC_COORD_REF
AxesGroup Reference To AXES_GROUP_REF
BOOL Done
BOOL Busy
BOOL Active
BOOL CommandAborted
BOOL Error
UDINT ErrorId

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 [▸ 55]).

Successor movements (e.g. MC_MovePath [▸ 70]) 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 [▸ 85]).

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 [▸ 96]).

Outputs

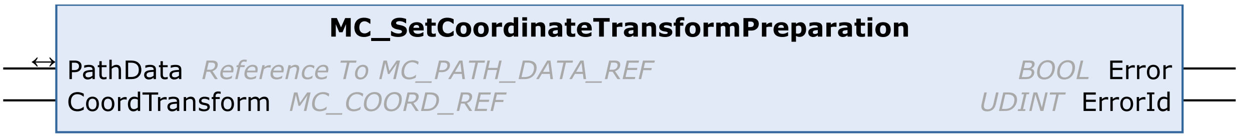
```
VAR_OUTPUT
    Done           : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorId        : UDINT;
END_VAR
```

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

7.1.1.1.10 MC_SetCoordinateTransformPreparation


MC_SetCoordinateTransformPreparation
↔
PathData Reference To MC_PATH_DATA_REF
CoordTransform MC_COORD_REF
BOOL Error
UDINT ErrorId

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 [▶ 82]	Table containing the segments of a path. The table is written by the function block MC_…Preparation and executed by MC_MovePath [▶ 70].
CoordTransform	MC_COORD_REF [▶ 85]	Reference to a coordinate system.

Outputs

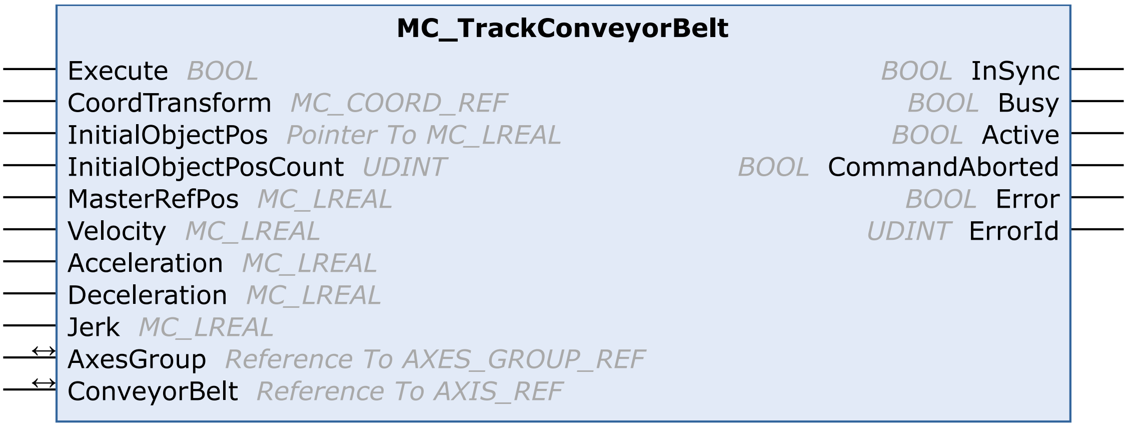
```
VAR_OUTPUT
    Error   : BOOL;
    ErrorId : UDINT;
END_VAR
```

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

7.1.1.1.11 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 [▶ 18].
- 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.

Name	Type	Description
CoordTransform	MC_COORD_REF	Reference to a coordinate system (see MC_COORD_REF [▸ 85]).
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 [▸ 96]).
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
```

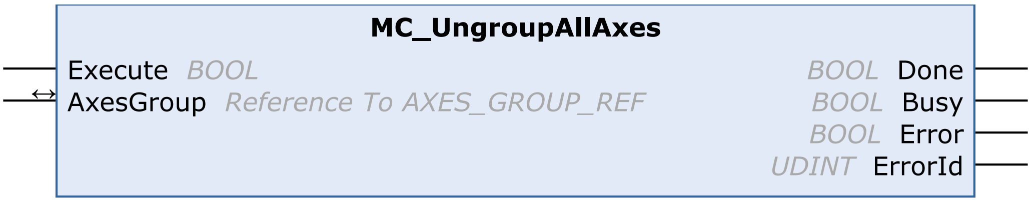
Name	Type	Description
InSync	BOOL	The output `InSync` becomes `TRUE` for the first time when the slave is synchronized with the velocity. If the slave dynamics is too low to follow the master movement, the output `InSync` could be reset to `FALSE`, after which the slave axis starts synchronizing again. ***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**

Name	Type	Description
		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. ***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

7.1.1.1.12 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).

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

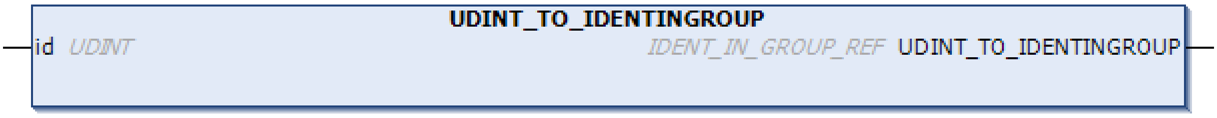
```
VAR_OUTPUT
    Done                       : BOOL;
    Busy                       : BOOL;
    Error                      : BOOL;
    ErrorId                    : UDINT;
END_VAR
```

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

7.1.1.1.13 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. This conversion function returns a valid input for MC_AddAxisToGroup [▸ 40] and MC_RemoveAxisFromGroup [▸ 51]. For axes intended for multi-dimensional movement (TF5420) see IDENT_IN_GROUP_REF [▸ 76].

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 [▸ 76] (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP [▸ 59].

Return value

Name	Type	Description
UDINT_TO_IDENTINGROUP	IDENT_IN_GROUP_REF [▸ 76]	Converts an integer value, so a PTP axis can be added to a motion group.

Inputs

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

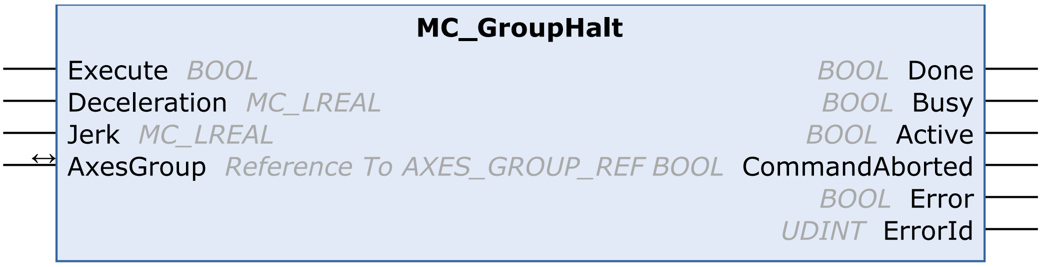
Name	Type	Description
id	UDINT	The unique identifier an axis shall have in the group. This does not have to be the axis ID from the cyclic axis interface.

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

7.1.1.2 Motion

7.1.1.2.1 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 [▶ 62]", 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 [▶ 98]" 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 [▶ 98]" 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

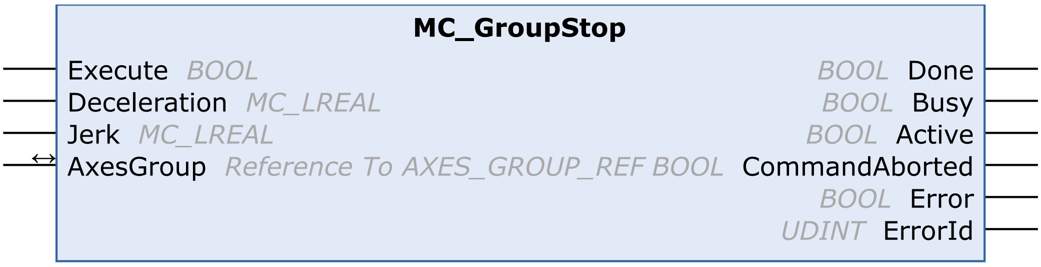
```
VAR_OUTPUT
    Done                    : BOOL;
    Busy                    : BOOL;
    Active                  : BOOL;
    CommandAborted          : BOOL;
    Error                   : BOOL;
    ErrorId                 : UDINT;
END_VAR
```

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).
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

7.1.1.2.2 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).

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 [▸ 98]" 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 [▸ 98]" 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
```

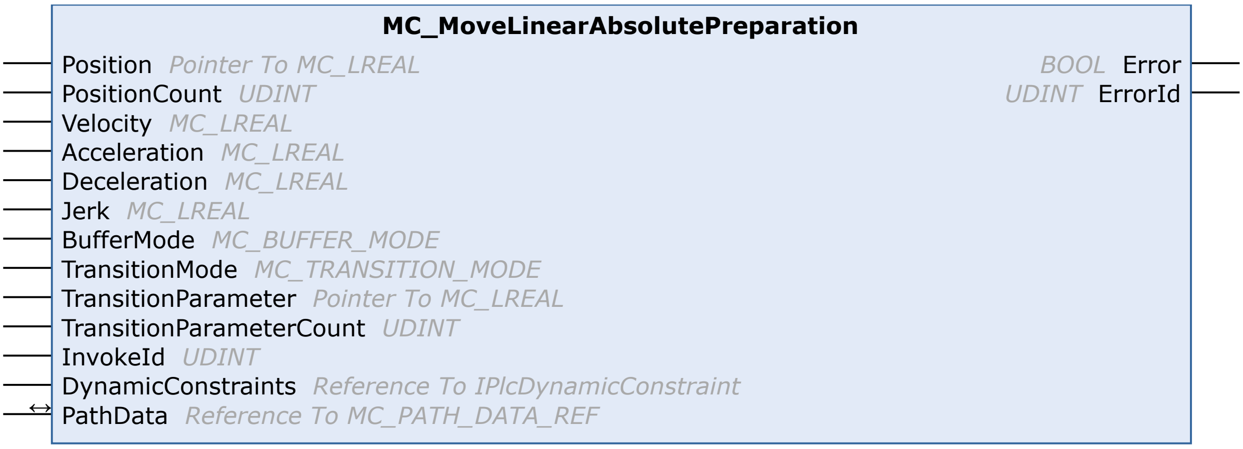
Name	Type	Description
Done	BOOL	Becomes TRUE when the group has been stopped and has come to a standstill. The group remains in the GroupStopping state while *Execute* is TRUE, at least until the axes have come to a stop. The group is then in the GroupStandby state (see State diagrams).
Busy	BOOL	Becomes 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.

Name	Type	Description
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.
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

7.1.1.2.3 MC_MoveLinearAbsolutePreparation



MC_MoveLinearAbsolutePreparation

```
Position  Pointer To MC_LREAL                              BOOL  Error
PositionCount  UDINT                                      UDINT  ErrorId
Velocity  MC_LREAL
Acceleration  MC_LREAL
Deceleration  MC_LREAL
Jerk  MC_LREAL
BufferMode  MC_BUFFER_MODE
TransitionMode  MC_TRANSITION_MODE
TransitionParameter  Pointer To MC_LREAL
TransitionParameterCount  UDINT
InvokeId  UDINT
DynamicConstraints  Reference To IPlcDynamicConstraint
↔ PathData  Reference To 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)	✓

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 [▶ 70]. 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 [▶ 98] 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 [▶ 98] 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 [▶ 98] 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 [▶ 86]).
Transition mode	MC_TRANSITION_MODE	Defines the blending mode (see MC_TRANSITION_MODE [▶ 83]).
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 [▶ 83]).
TransitionParameterCount	UDINT	Number of blending parameters (see MC_TRANSITION_MODE [▶ 83]).
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 [▶ 70] (see MC_PATH_DATA_REF [▶ 82]).

Resetting a table

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

Outputs

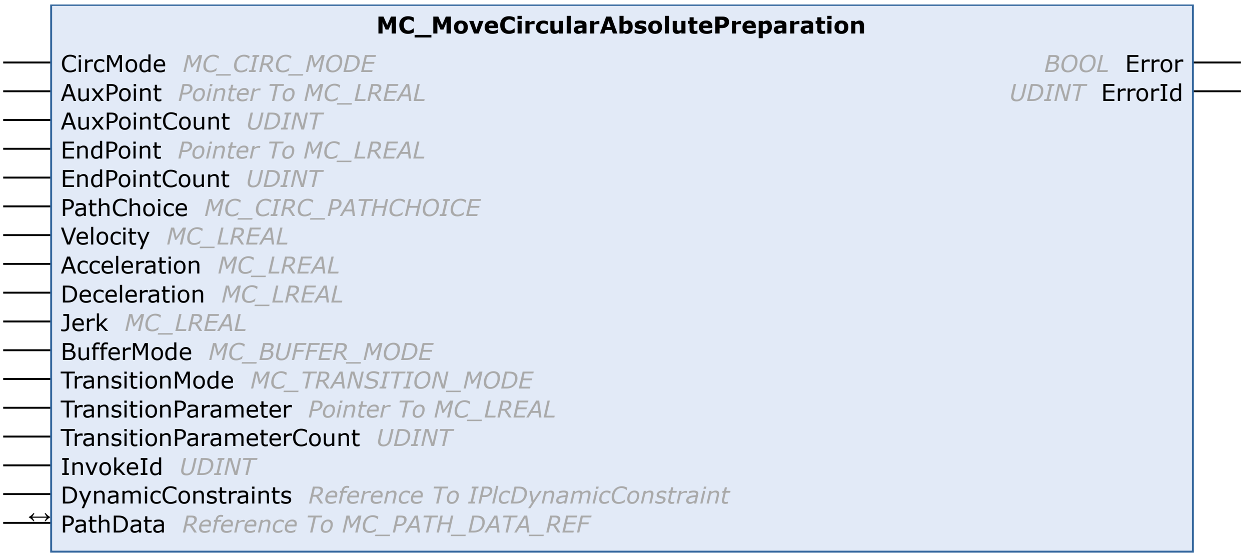
```
VAR_OUTPUT
    Error                       : BOOL;
    ErrorId                     : UDINT;
END_VAR
```

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

7.1.1.2.4 MC_MoveCircularAbsolutePreparation



MC_MoveCircularAbsolutePreparation

CircMode *MC_CIRC_MODE*
AuxPoint *Pointer To MC_LREAL*
AuxPointCount *UDINT*
EndPoint *Pointer To MC_LREAL*
EndPointCount *UDINT*
PathChoice *MC_CIRC_PATHCHOICE*
Velocity *MC_LREAL*
Acceleration *MC_LREAL*
Deceleration *MC_LREAL*
Jerk *MC_LREAL*
BufferMode *MC_BUFFER_MODE*
TransitionMode *MC_TRANSITION_MODE*
TransitionParameter *Pointer To MC_LREAL*
TransitionParameterCount *UDINT*
InvokeId *UDINT*
DynamicConstraints *Reference To IPlcDynamicConstraint*
PathData *Reference To MC_PATH_DATA_REF*

BOOL Error
UDINT ErrorId

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_MoveCircularAbsolutePreparation function block adds an absolute circular motion to the table of segments in the MC_PATH_DATA_REF [▶ 82] structure. After creating a table, it can be executed via MC_MovePath [▶ 70]. The function block MC_MoveCircularAbsolutePreparation can be called several times per cycle. A maximum of 30 entries are permitted per MC_Path_DATA_REF table.

Resetting a table

A table is not reset during execution. To reset, the ClearPath [▶ 83] method must be called from MC_PATH_DATA_REF [▶ 82].

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 [▶ 77]).
AuxPoint	POINTER TO MC_LREAL	Pointer to an array [1..AuxPointCount] of the AuxPoint vector. The interpretation of the AuxPoint vector depends on the spatial axes convention (see MC Group Coordinated Motion).
AuxPointCount	UDINT	Dimension of the AuxPoint vector.
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 [▶ 81]).
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 [▶ 98] 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 [▶ 98] 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 [▶ 98] 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 [▶ 86]).
Transition mode	MC_TRANSITION_MODE	Defines the blending mode (see MC_TRANSITION_MODE [▶ 83]).
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 [▶ 83]).
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 [▶ 70] (see MC_PATH_DATA_REF [▶ 82]).

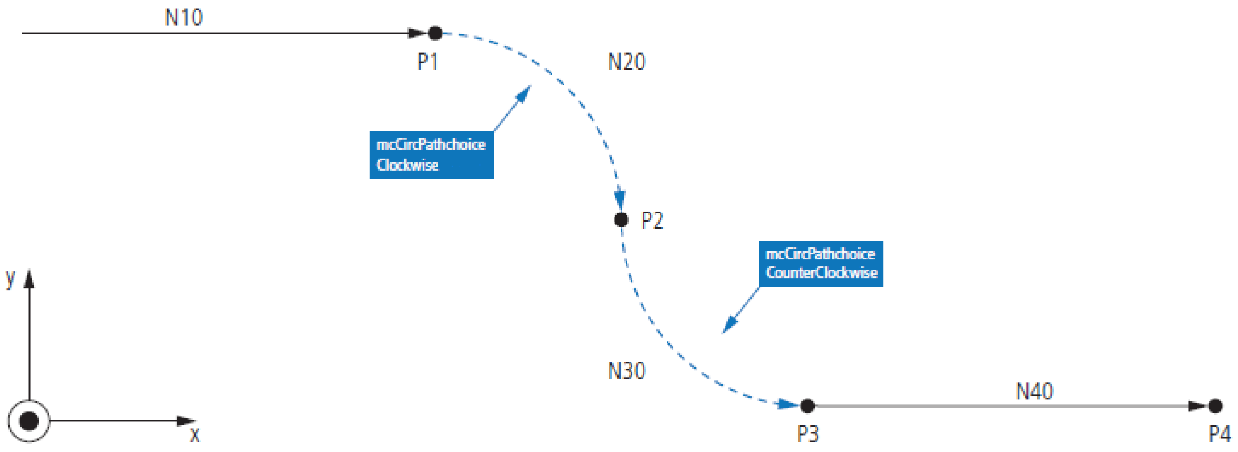
Outputs

```
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 [▶ 81] 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.



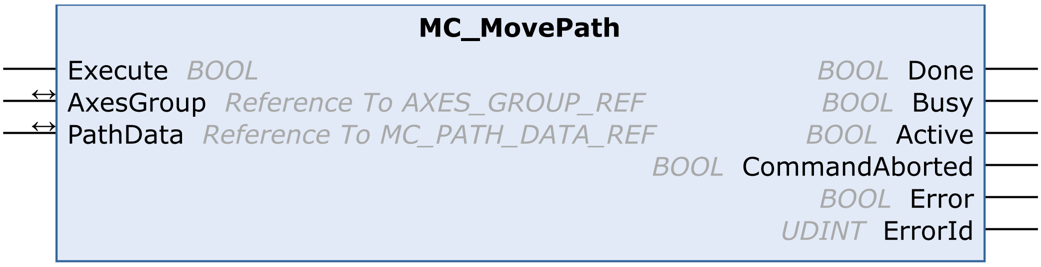
```
VAR
    Buffer                                   : ARRAY[1..4096] OF BYTE;
    Path                                     : MC_PATH_DATA_REF (ADR(buffer), SIZEOF(buffer));
    fbMoveLinPrep                            : MC_MoveLinearAbsolutePreparation;
    fbMoveCircPrep                           : 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:= mcCircPathchoiceClockwise, Velocity:= 1000, InvokeI
d:= 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

7.1.1.2.5 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 [▸ 64], MC_MoveCircularAbsolutePreparation [▸ 66], MC_BlockerPreparation [▸ 71] and MC_SetCoordinateTransformPreparation [▸ 54].

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 [▶ 96]).
PathData	MC_PATH_DATA_REF [▶ 82]	Table containing the segments of a path. The table is written by MC_MoveLinearAbsolutePreparation [▶ 64], MC_MoveCircularAbsolutePreparation [▶ 66], MC_BlockerPreparation [▶ 71] and MC_SetCoordinateTransformPreparation [▶ 54] and executed by MC_MovePath [▶ 70].

Outputs

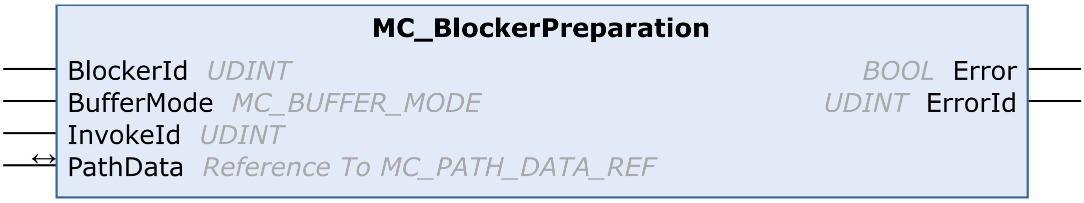
```
VAR_OUTPUT
    Done                        : BOOL;
    Busy                        : BOOL;
    Active                      : BOOL;
    CommandAborted              : BOOL;
    Error                       : BOOL;
    ErrorId                     : UDINT;
END_VAR
```

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

7.1.1.2.6 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 [▸ 73]. 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 [▸ 60] or MC_GroupStop [▸ 62] 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 [▸ 86]). 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
```

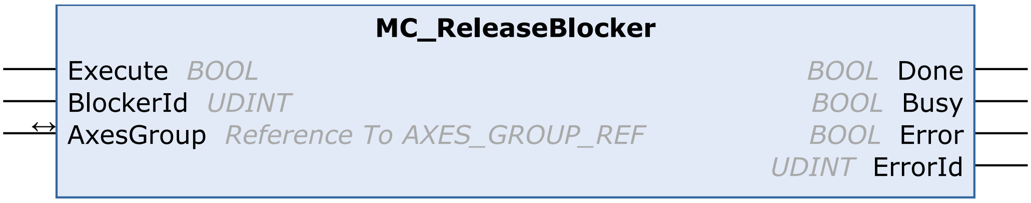
Name	Type	Description
Error	BOOL	This output becomes TRUE if an error has occurred during command execution.

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

7.1.1.2.7 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 [▶ 71].

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 [▶ 96]).

Outputs

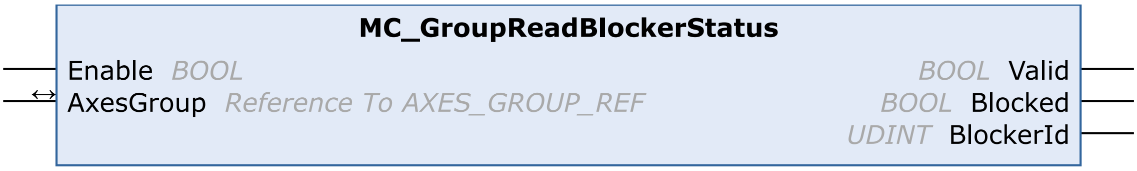
```
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

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

7.1.1.2.8 MC_GroupReadBlockerStatus


MC_GroupReadBlockerStatus
Enable BOOL
AxesGroup Reference To AXES_GROUP_REF
BOOL Valid
BOOL Blocked
UDINT BlockerId

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 [▶ 96]).

Outputs

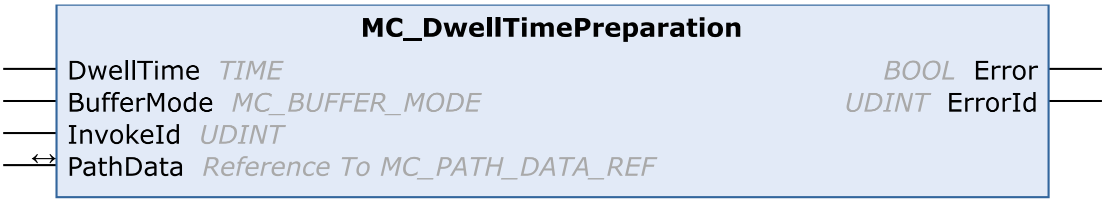
```
VAR_OUTPUT
    Valid           : BOOL;
    Blocked         : BOOL;
    BlockerId       : UDINT;
END_VAR
```

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

7.1.1.2.9 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 [▶ 86]). 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

7.1.2 Datatypes

7.1.2.1 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 [▶ 59] 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 [▶ 76] (e.g. MCS_X) or the conversion function UDINT_TO_IDENTINGROUP [▶ 59].

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

7.1.2.2 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)	✓

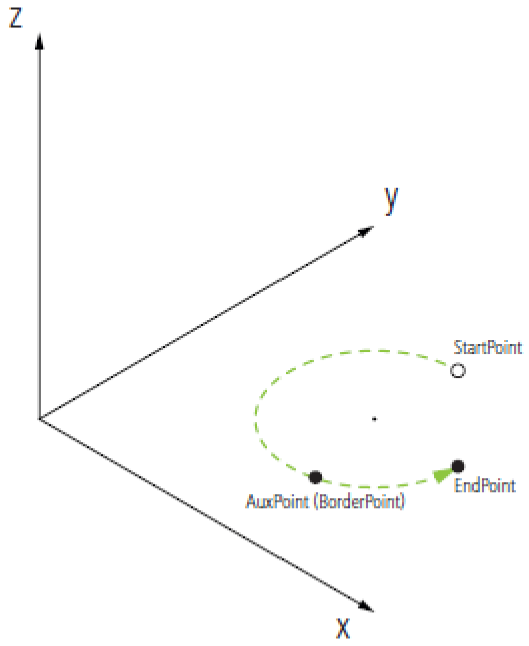
The circle mode determines which circle definition is used to program a circle.

```
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.

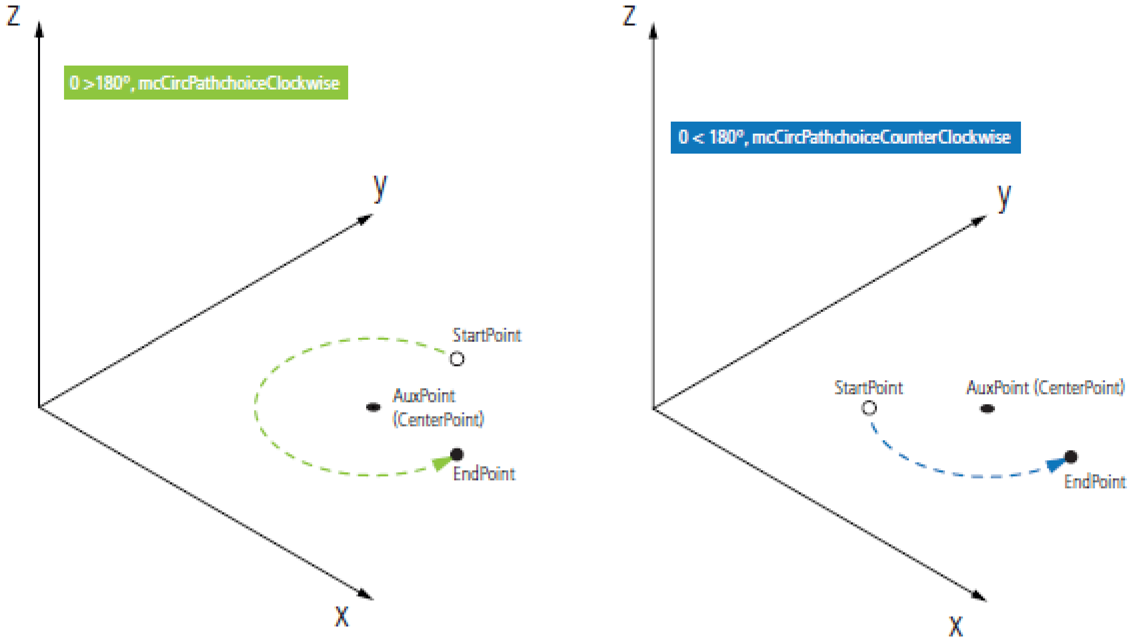
PathChoice

- 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.

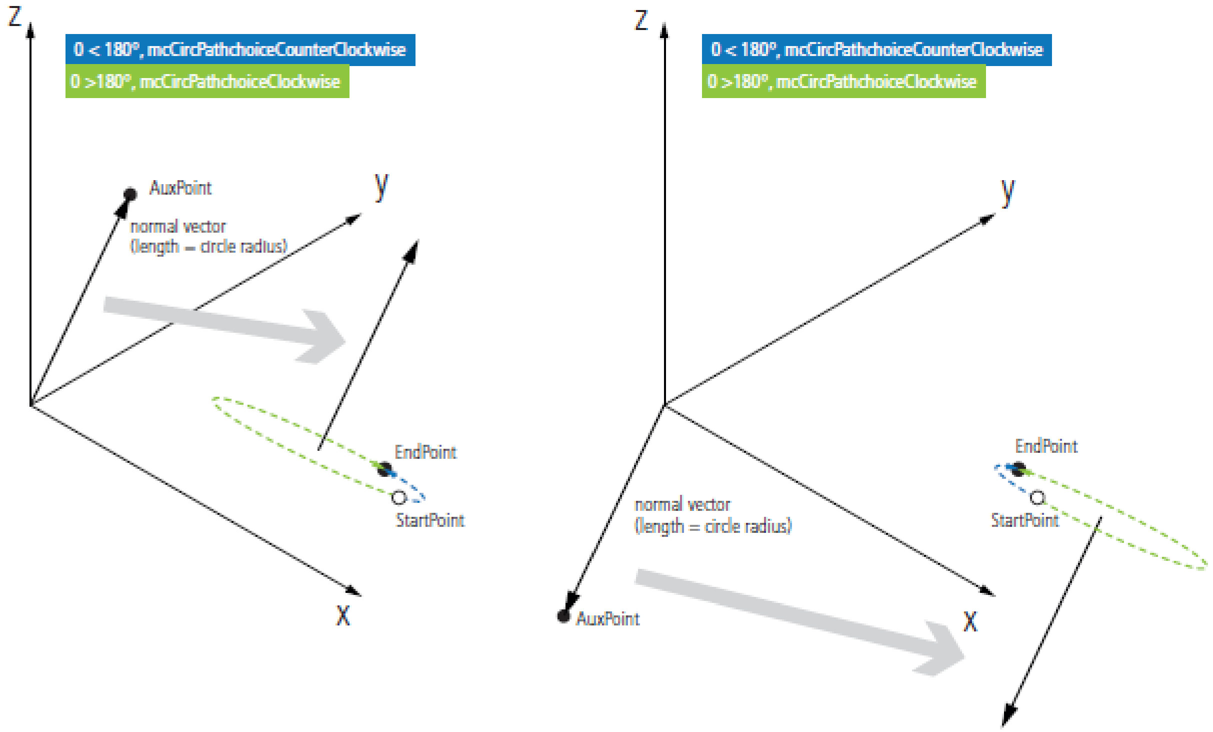
PathChoice

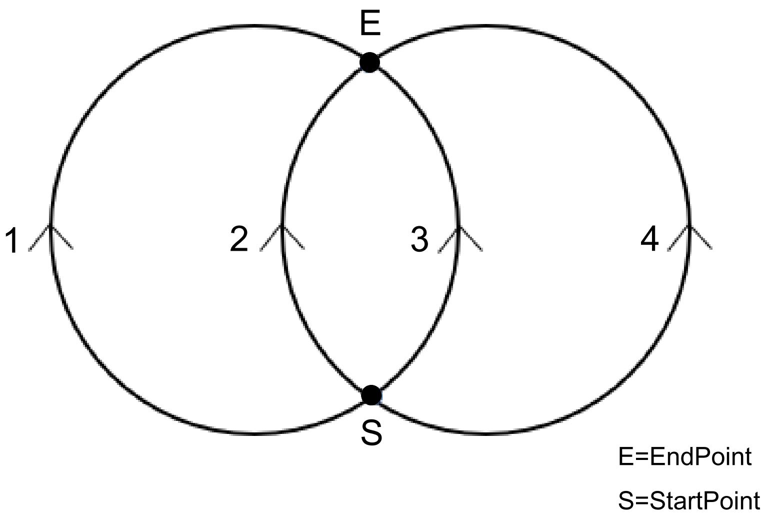
- There are usually two possible arcs of the circle that can be traversed from starting point "StartPoint" to the endpoint "EndPoint". The “PathChoice” parameter makes the two unique. See MC_CIRC_PATHCHOICE for more information.

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


Z
0 < 180°, mcCircPathchoiceCounterClockwise
0 >180°, mcCircPathchoiceClockwise
AuxPoint
y
normal vector
(length = circle radius)
EndPoint
StartPoint
X
Z
0 < 180°, mcCircPathchoiceCounterClockwise
0 >180°, mcCircPathchoiceClockwise
y
EndPoint
normal vector
(length = circle radius)
StartPoint
X
AuxPoint



MC_CIRC_PATHCHOICE	$\vec{n}$	
Clockwise	⊙	1
	⊗	4
Counterclockwise	⊙	3
	⊗	2
Short segment	⊙	3
	⊗	2
Long segment	⊙	4
	⊗	1

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

7.1.2.3 MC_CIRC_PATHCHOICE

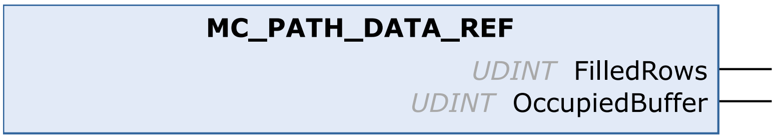
TF5410 TwinCAT 3 Motion Collision Avoidance	TF5420 TwinCAT 3 Motion Pick-and-Place	
	MC Group with Pick-and-Place	MC Group Coordinated Motion
✗	✗ (✓ up to and including v3.2)	✓

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 [▶ 77].

```
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.

7.1.2.4 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 [▶ 70], whereby the number of entries is limited to 30. The path to be executed is written by MC_MoveLinearAbsolutePreparation [▶ 64], MC_MoveCircularAbsolutePreparation [▶ 66] and MC_BlockerPreparation [▶ 71]. 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 [▶ 83] 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

7.1.2.4.1 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.

7.1.2.5 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

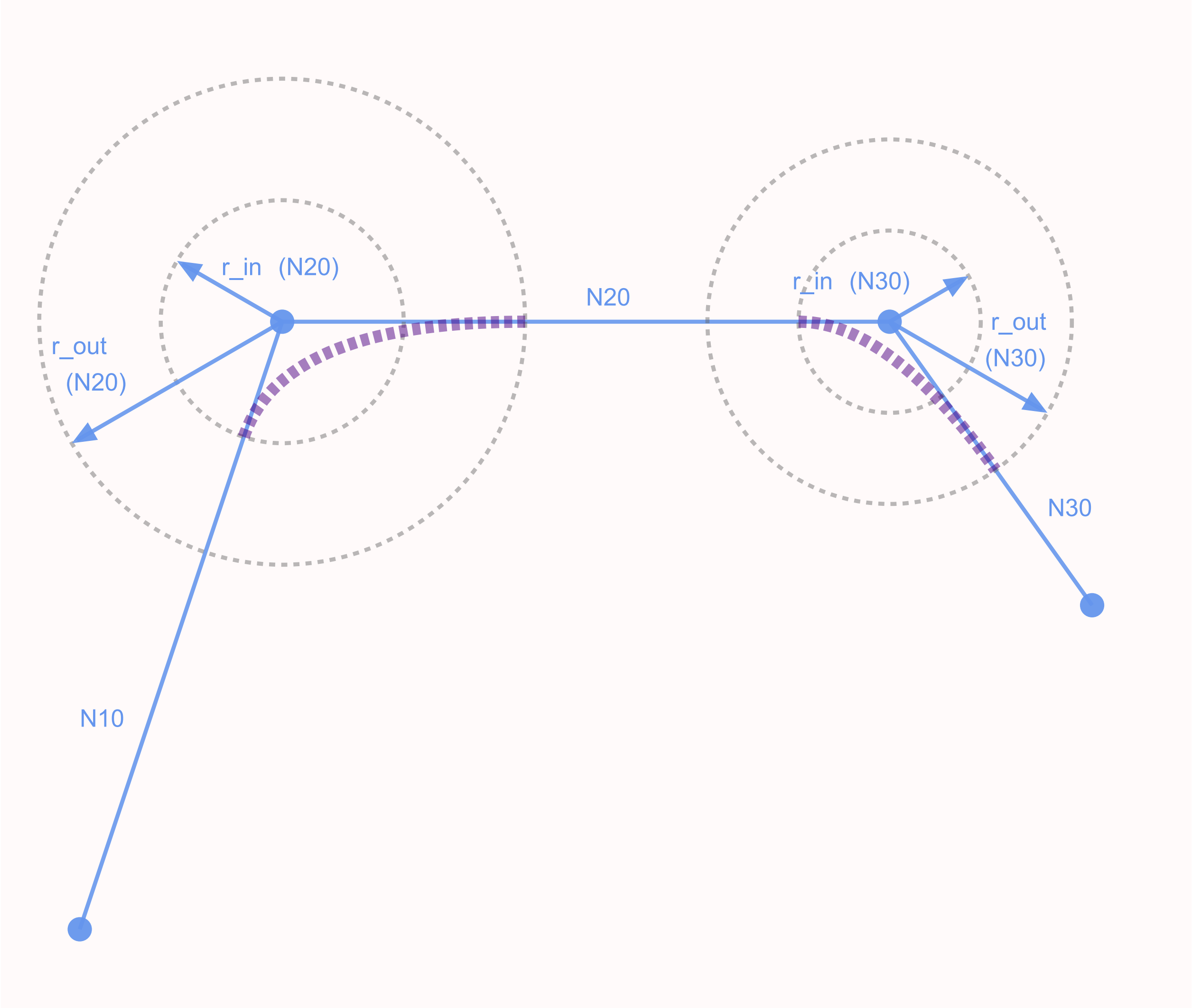
Blending is executed between the segments. The transition parameters act as tolerance ball in which the programmed path is not followed. The parameter describes the radius on the previous and second segment at which the blending starts and ends.

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

7.1.2.6 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.

7.2 Tc3_Mc3Definitions

Structures and enumerations

Name	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_BUFFER_MODE [▶ 86]	Defines how successive travel commands are to be processed.	✓	✓	✓
MC_COMPENSATION_TYPE [▶ 89]	The value defines the compensation type.	✓	✗	✗
MC_DIRECTION [▶ 90]	The value determines the direction of the movement.	✓	✗	✗
MC_SYNC_MODE [▶ 91]	The value defines the direction in which synchronization is to be performed.	✓	✗	✗
MC_SYNC_STRATEGY [▶ 91]	Defines the synchronization profile of the slave axis.	✓	✗	✗

7.2.1 Datatypes

7.2.1.1 MC_BUFFER_MODE

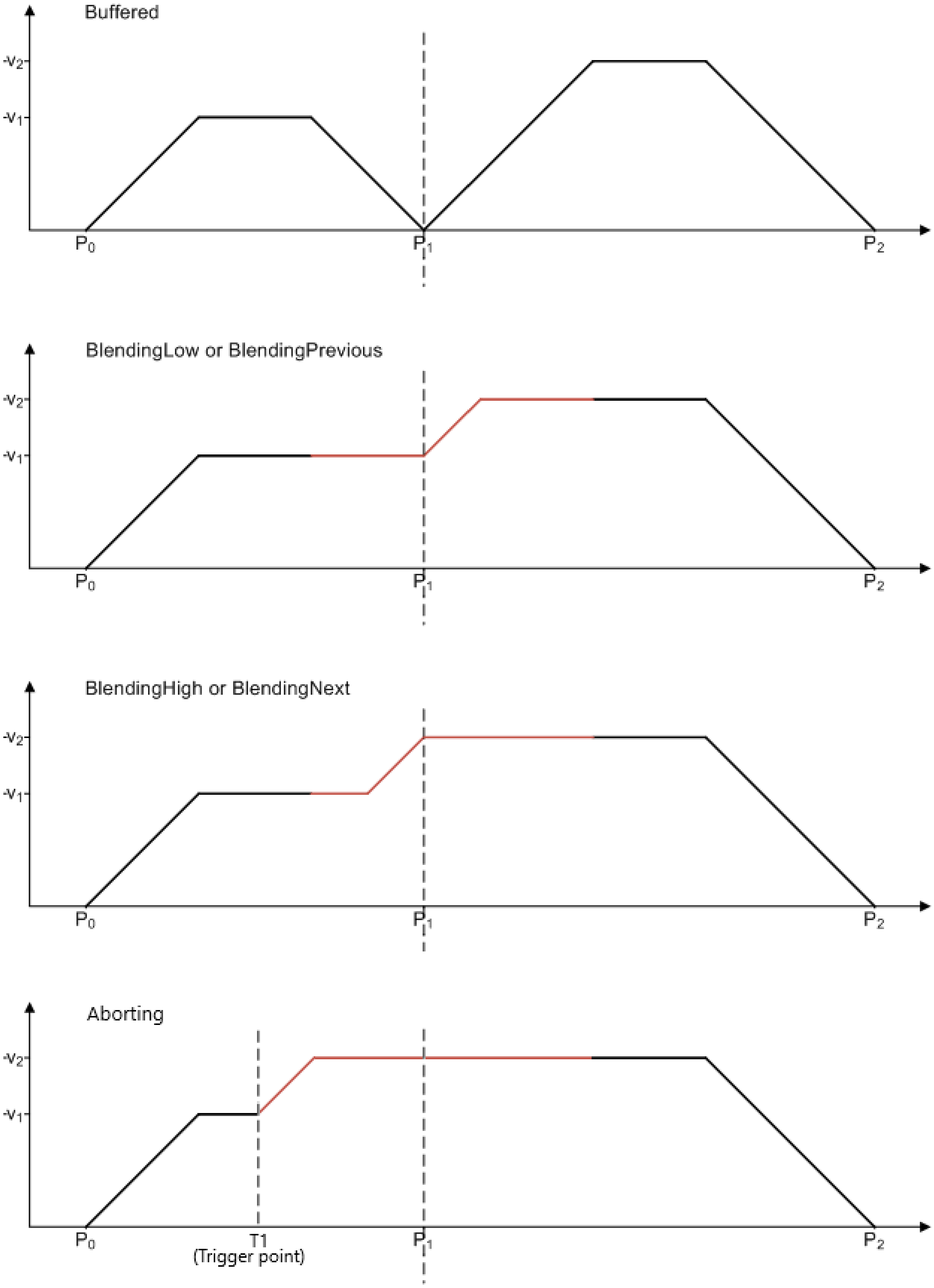
The data type MC_BUFFER_MODE is used to specify how successive travel commands are to be processed. At least two function blocks are required for buffer mode to have an effect.

```
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
✓	✓	✓

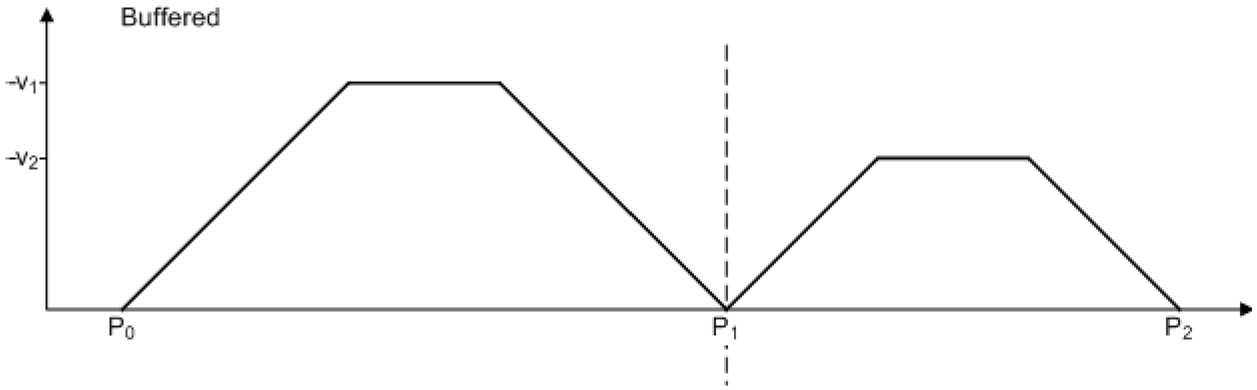
Example:

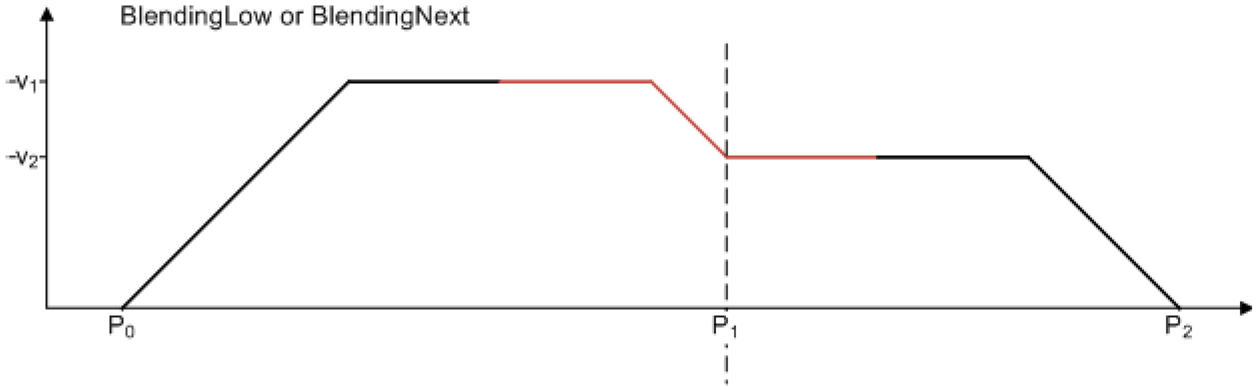
In the following example, a move command is used to move a group from position P_0 to P_1 and then to P_2. The reference point for the different velocity profiles is always P_1. The mode specifies the velocity v_1 or v_2 at this point.

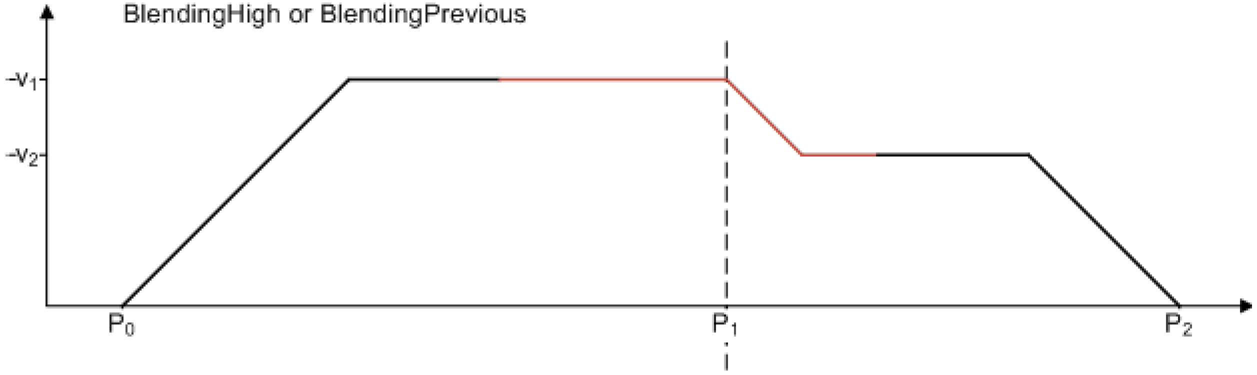


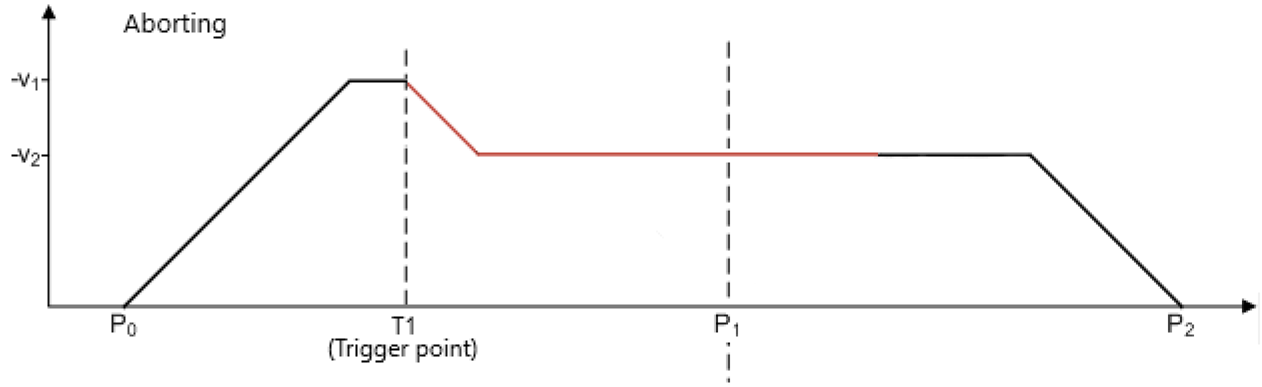
Since the speed of the first command is lower than the second, the modes BlendingLow/BlendingPrevious and BlendingHigh/BlendingNext have the same result.

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

7.2.1.2 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

7.2.1.3 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

7.2.1.4 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

7.2.1.5 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.

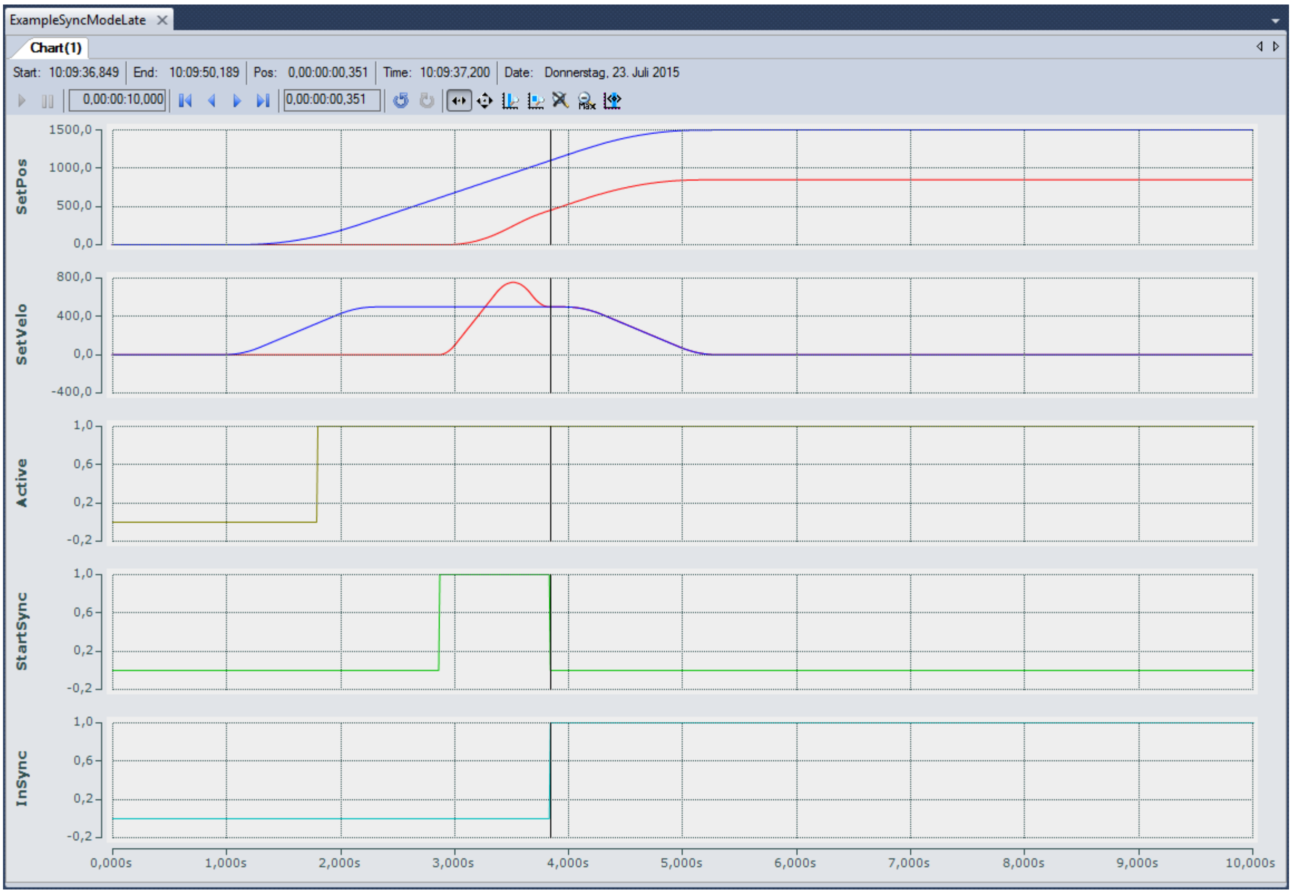
- A motion command is issued to the master.

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).



ExampleSyncModeLate

Chart(1)

Start: 10:09:36,849 | End: 10:09:50,189 | Pos: 0,00:00:00,351 | Time: 10:09:37,200 | Date: Donnerstag, 23. Juli 2015

0,00:00:10,000 0,00:00:00,351

Example 2: mcSyncStrateySlow

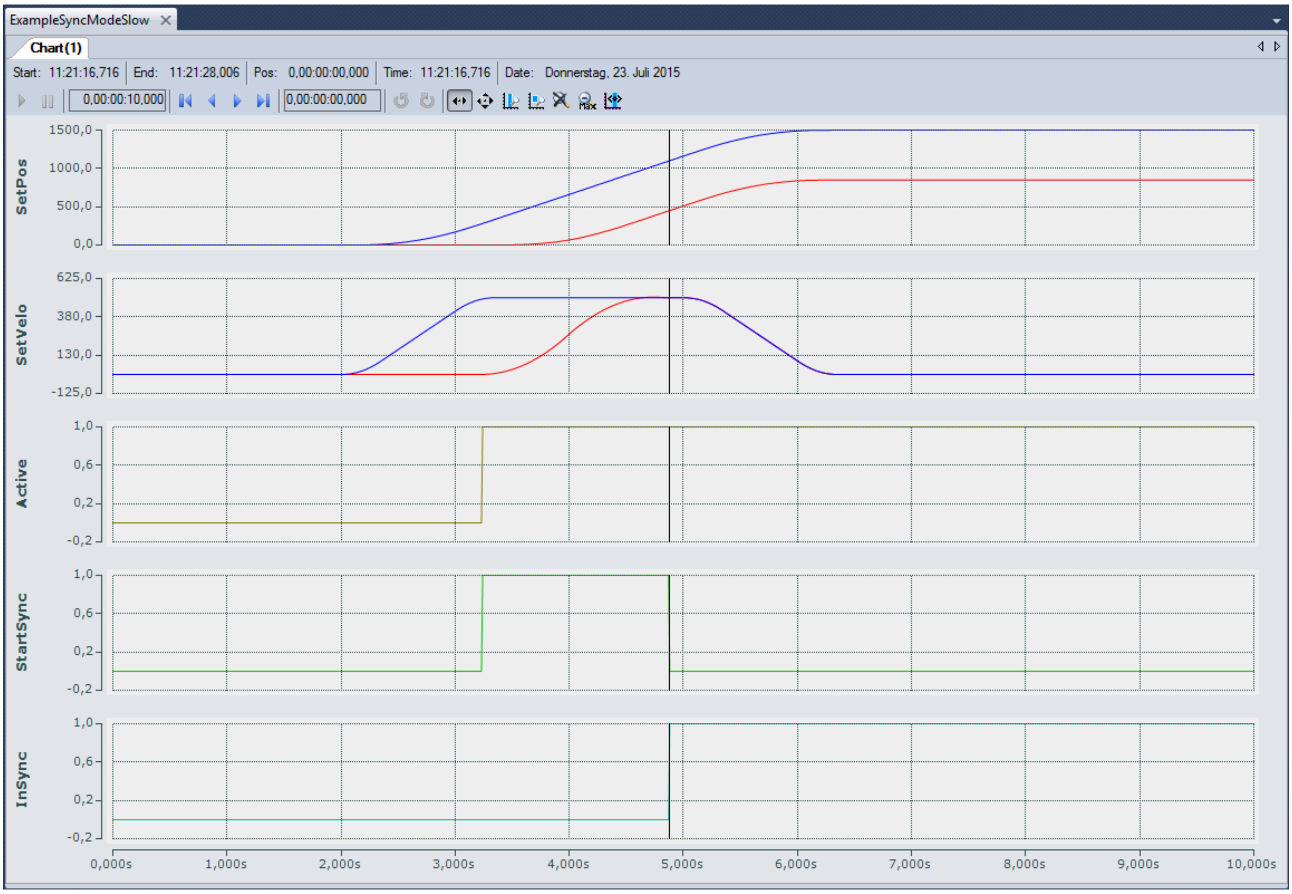
The slave starts its synchronization in motion as soon as the master passes (MasterSyncPosition - MasterStartDistance) in the right direction if a MasterStartDist was set, otherwise as soon as the function block is Active. The dynamics of the slave are reduced so that the slave reaches the SlaveSyncPos 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.

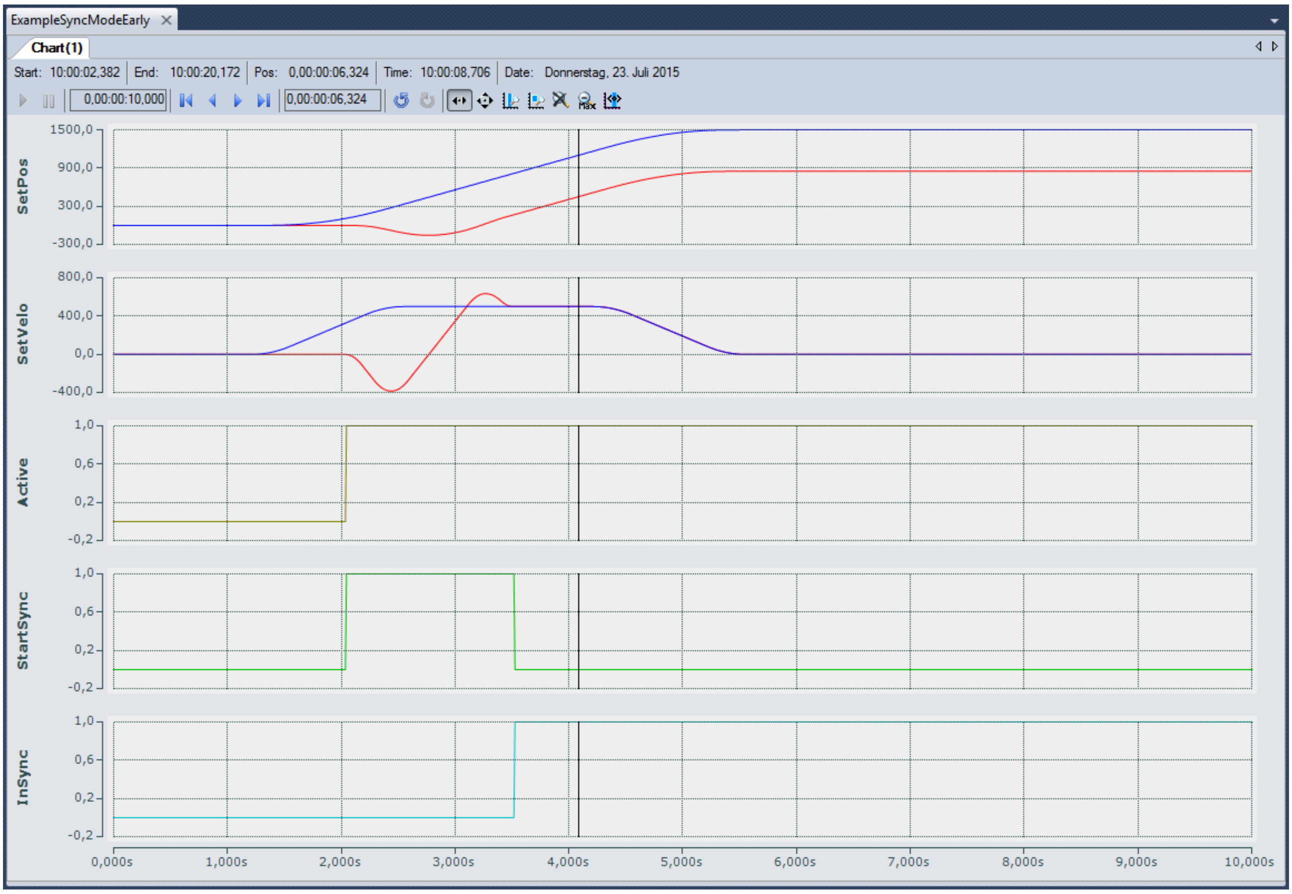


Example 3: mcSyncStrategyEarly

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).



ExampleSyncModeEarly

Chart(1)

Start: 10:00:02,382 End: 10:00:20,172 Pos: 0,00:00:06,324 Time: 10:00:08,706 Date: Donnerstag, 23. Juli 2015

0,00:00:10,000 0,00:00:06,324

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

8 Samples

Multi-dimensional motion

PnpSimpleSample

Download: https://infosys.beckhoff.com/content/1033/TF5420_TC3_Advanced_Pick_And_Place/Resources/9725595531.zip

Description:
Project that executes a simple pick and place-movement.

PnPSimpleSample with an additional axis and blocking

Download: https://infosys.beckhoff.com/content/1033/TF5420_TC3_Advanced_Pick_And_Place/Resources/9725597195.zip

Description:
Project that contains an additional axis and a blocking job.

9 Appendix

9.1 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 [▶ 96] and PlcToNc [▶ 97]. 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 [▶ 97] 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 [▶ 96] 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.

9.1.1 NcToPlc

Data structure which is cyclically exchanged between NC and PLC. Some data is group-specific.

Name				Type	Description
Common					
	GroupOID			OTCID	TcCOM object ID (OID) of this group
	GroupType			MC_GROUP_TYPE [▶ 99]	Type of group
	GroupStatus			ST_GroupStatus [▶ 100]	Status information about the group
		State		MC_GROUP_STATE [▶ 99]	State of the group, see Group State Diagram.
		Flags		ST_GroupStatusFlags [▶ 100]	Additional status information
			IsEnableRequested	BIT	Defines whether enabling or disabling of a group is requested.
			AllAxesStanding	BIT	Indicates whether all axes are physically stationary, i.e. the velocity and acceleration for all axes is zero, regardless of whether a motion command is active.
			InPosition	BIT	Indicates whether the group is in position.
	GroupErrorId			UDINT	Error code in the event of a group error
	GroupAxesCount			UDINT	Number of axes that are currently part of this group (e.g. added via MC_AddAxisToGroup).

Name	Type	Description
ActOverrideFactor	LREAL	Currently used override factor
PathVelo	LREAL	Only MC Group Coordinated Motion: Absolute value of cartesian velocity on the path
InvokeId	UDINT	Only MC Group Coordinated Motion: Segment ID for analysis purposes
IsInBlendingSegment	BIT	Only MC Group Coordinated Motion: Indicates if a blending segment is active.
RemainingTimeActiveJob	LREAL	Only MC Group Coordinated Motion: Remaining time of the current segment
RemainingCartesianDistanceActiveJob	LREAL	Only MC Group Coordinated Motion: Remaining distance for the current segment
ActiveBlockerId	UDINT	Only MC Group Coordinated Motion: Id of the active blocker.
RemainingTimeToSync	LREAL	Only MC Group Coordinated Motion: Remaining time until the axis group is synchronized with the conveyor belt during conveyor tracking.
RemainingCartesianTimeToSync	LREAL	Only MC Group Coordinated Motion: Remaining distance until the axis group is synchronized with the conveyor belt during conveyor tracking.

9.1.2 PlcToNc

Data structure which is cyclically exchanged between PLC and NC.

Name	Type	Description
Common		
OverrideFactor	LREAL	Desired Override Factor (1.0 = 100%, Default Value is 1.0)

9.2 Index Offset Specification for MC Group Parameters

Port 501: `AMSPORT_R0_NCSAF: UINT := 501;`.
The Object ID (OID) of the MC group must be designated as the index group.

Index offset (Hex)	Access	Group type	Data Type	Phys. unit	Definition range	Description	Comments
0x5030084	Read	MC Group	LREAL	mm	>=0	Remaining distance of the current segment.	Only available for MC Group with Pick-and-Place from V3.1.6.27
0x5030085	Read	MC Group	LREAL	s	>=0	Remaining time for the current segment.	Only available for MC Group with Pick-and-Place from V3.1.6.27

9.3 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 [▶ 98]). "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.

9.4 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

Value	Signification	Example
		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

9.5 Global data types

9.5.1 MC_GROUP_TYPE

The enum `MC.MC_GROUP_TYPE` defines the type of a group.

Syntax

Definition:

```
TYPE MC_GROUP_TYPE
(
    mcGroupTypeInvalid := 0,
    mcGroupTypeCA      := 1,
    mcGroupTypeCM      := 3
)
END_TYPE
```

Values

Name	Description
mcGroupTypeInvalid	No valid group
mcGroupTypeCA	Collision Avoidance group
mcGroupTypeCM	Coordinated Motion group [▶ 18]

9.5.2 MC_GROUP_STATE

The enum `MC.MC_GROUP_STATE` defines the state of a group. State transitions, see state diagram [▶ 11].

Syntax

Definition

```
TYPE MC_GROUP_STATE
(
    mcGroupStateInvalid   := 0,
    mcGroupStateDisabled  := 1,
    mcGroupStateStandby   := 2
    mcGroupStateMoving    := 3,
    mcGroupStateStopping  := 4,
    mcGroupStateErrorStop := 5,
    mcGroupStateHoming    := 6,
    mcGroupStateNotReady  := 7,
    mcGroupStateSuspended := 8
)
END_TYPE
```

Values

Name	Description
mcGroupStateInvalid	No valid group state
mcGroupStateDisabled	The group is disabled.
mcGroupStateStandby	The group is in standby state.
mcGroupStateMoving	The group is moving.
mcGroupStateStopping	The group is stopping.
mcGroupStateErrorStop	The group stops due to an error.
mcGroupStateHoming	
mcGroupStateNotReady	The group is not ready.
mcGroupStateSuspended	

9.5.3 ST_GroupStatus

The structure ST_GroupStatus defines the status of a group.

Syntax

Definition:

```
TYPE ST_GroupStatus
STRUCT
    State : MC_GROUP_STATE;
    Flags : ST_GroupStatusFlags;
END_STRUCT
END_TYPE
```

Values

Name	Type	Description
State	MC_GROUP_STATE [▶ 99]	State of the group
Flags	ST_GroupStatusFlags [▶ 100]	Additional status information

9.5.4 ST_GroupStatusFlags

The structure ST_GroupStatusFlags defines additional status information.

Syntax

Definition:

```
TYPE
STRUCT
    IsEnableRequested : BIT;
    AllAxesStanding   : BIT;
    InPosition        : BIT;
END_STRUCT
END_TYPE
```

Values

Name	Type	Description
IsEnableRequested	BIT	Defines whether enabling or disabling of a group is requested.
AllAxesStanding	BIT	Indicates whether all axes are physically stationary, i.e. the velocity and acceleration for all axes is zero, regardless of whether a motion command is active.
InPosition	BIT	Indicates whether the group is in position.

Trademark statements

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

More Information:
www.beckhoff.com/TF5420

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