# Table of Contents

1 Foreword ......................................................................................................................................................... 5
  1.1 Notes on the documentation .................................................................................................................. 5
  1.2 Safety instructions .................................................................................................................................. 6

2 Introduction ...................................................................................................................................................... 7

3 TF5420 TC3 Motion Pick-and-Place - Overview New Features ........................................................................... 8

4 Group State Diagram ..................................................................................................................................... 9

5 MC Group (TF5420 TC3 Motion Pick-and-Place) ....................................................................................... 11
  5.1 Configure an MC Group ......................................................................................................................... 11
  5.2 MC Group Coordinated Motion ............................................................................................................. 16
  5.3 MC Group with Pick-and-Place ............................................................................................................. 18

6 Spatial Configuration ...................................................................................................................................... 20
  6.1 Coordinate Frame Object ....................................................................................................................... 20
  6.2 Conveyor Tracking Object ...................................................................................................................... 20
  6.3 Conveyor Tracking Behavior .................................................................................................................. 20
  6.4 Node Connector Object .......................................................................................................................... 21
  6.5 Configuring a Node Connector ............................................................................................................... 21
  6.6 Configure for MC_TrackConveyorBelt .................................................................................................... 26
  6.7 Background Information ........................................................................................................................ 33

7 PLC Libraries .................................................................................................................................................. 36
  7.1 Tc3_McCoordinatedMotion .................................................................................................................... 36
    7.1.1 Function Blocks ............................................................................................................................... 38
    7.1.2 Datatypes ........................................................................................................................................ 66
  7.2 Tc3_Mc3Definitions .................................................................................................................................. 76
    7.2.1 Datatypes ........................................................................................................................................ 76

8 Samples .......................................................................................................................................................... 85

9 Appendix ......................................................................................................................................................... 86
  9.1 Cyclic Group Interface ............................................................................................................................ 86
    9.1.1 NcToPlc .......................................................................................................................................... 86
    9.1.2 PlcToNc .......................................................................................................................................... 87
  9.2 Index Offset Specification for MC Group Parameters .............................................................................. 87
  9.3 Differences between MC2 and MC3 ........................................................................................................ 87
  9.4 Glossary .................................................................................................................................................... 88
  9.5 MC_LREAL/Special Input Values ............................................................................................................ 88
1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

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

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

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

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

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

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DANGER</td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td>WARNING</td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td>CAUTION</td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td>NOTE</td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer
This symbol indicates information that contributes to better understanding.
2 Introduction

The TF5420 TC3 Motion Pick-and-Place software package is installed together with the TF5400 software package.

Target System

Windows XP or Windows 7/8/10.

TC3 Motion Pick-and-Place

TF5420 TC3 Motion Pick-and-Place executes multi-dimensional movements. It was designed especially for the requirements of Pick-and-Place applications. It is intended to be applied to movements for that the exact path dynamics in segment transitions are not that important, but for that the user desires to move from one point to another as quickly as possible. All associated function blocks are included in the Tc3_McCoordinatedMotion library.
3 TF5420 TC3 Motion Pick-and-Place - Overview New Features

Since V3.1.10.1:
- New Group type “MC Group Coordinated Motion” is available.
- Cyclic Interface is extended for “MC Group Coordinated Motion”.
- New Functionblocks for “MC Group Coordinated Motion”:
  - MC_BlockerPreparation
  - MC_ReleaseBlocker
  - MC_GroupReadBlockerStatus
  - MC_DwellTimePreparation
- MC_GroupHalt is implemented for “MC Group Coordinated Motion”.
- mcTransModeCornerDistance, mcCircPathchoiceShortSegment, mcCircPathchoiceLongSegment are implemented for “MC Group Coordinated Motion”.
- Requires TwinCAT V3.1.4024.7 or higher

Since V3.1.6.27:
- Remaining time and distance of current segment can be read via ADSREAD now.
- Requires TwinCAT V3.1.4022.0 or higher

Since V3.1.6.3:
- New function blocks for spatial transformations, i.e. for changing the reference system (MC_SetCoordinateTransform) and for Conveyor Tracking (MC_TrackConveyorBelt).

Since V3.1.4.4:
- New: Since software version 3.1.4.4 there is a new support of MC_MAXIMUM as an input value. For further details have a look at the particular function block documentations.

Since V3.1.2.47:
- New function block MC_MoveCircularAbsolutePreparation.
4 Group State Diagram

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

**Note**

**Description**

1. Applicable for all non-administrative (moving) function blocks.

2. In the state GroupStopping, all Function Blocks can be called, although they will not be executed, except MC_GroupDisable and MC_UngroupAllAxes, which will abort the stop and generate the transition to GroupDisabled.

3. MC_GroupStop.DONE AND NOT MC_GroupStop.EXECUTE

4. -

5. -

6. MC_GroupDisable and MC_UngroupAllAxes can be issued in all states and will change the state to GroupDisabled. If issued in any error state, the state changes to GroupErrorDisabled.

7. From any state with EnableRequested TRUE.

8. If "bIsControlLoopClosed" is true for all axes and the group is not empty, "bPositiveDirection"/"bNegativeDirection" need not be enabled.
"bIsControlLoopClosed" and both flags "bPositiveDirection"/"bNegativeDirection" have to be set to TRUE.

MC_GroupReset has no effect if issued in any other state than GroupErrorStop.

In the error states, all administrative Function Blocks except MC_GroupEnable are allowed. They can, however, only generate state transitions inside the error states, e.g. to GroupErrorDisabled for MC_GroupDisable or MC_UngroupAllAxes and MC_RemoveAxisFromGroup if the last axis is removed.

To leave the state GroupErrorStop, MC_GroupReset must be called.
5 MC Group (TF5420 TC3 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.

2. Add all axes to the NC-Configuration.
3. Add the appropriate Group to the entry “Objects” in the NC-Configuration:
For Coordinated Motion, multi-dimensional movements: MC Group Coordinated Motion [16] or MC Group with Pick-and-Place [18].
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".
Configure the group parameters according to the desired application.
For more explanations referring to the group parameters see the following sections.

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). 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.

```plaintext
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
        - OUTs
        - GVLs
    - POUs
      - MAIN (PRG)
        - VISUs
      - PlcTask (PlcTask)
        - Untitled1.tmc
      - Untitled1 Instance
    - SAFETY
    - I/O
Also see about this

Tc3_McCoordinatedMotion [36]

5.2 MC Group Coordinated Motion

Parameter (Init)

Spatial Axis Conventions

There are three axis conventions that can be set.
The axis conventions define how the axes in the axis group are interpreted. In combination with “Additional Axes Count” they define the axis group dimension and therefore the number of axes that have to be added and the way each one of the added axes is interpreted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial Axes Convention</td>
<td>mcAxesConv2DCartesianXY</td>
<td>mc.MC_AXES_CONVENTION</td>
<td>A 2D group consisting of X, Y. The order of the translational axes within the configuration defines the translation order.</td>
</tr>
<tr>
<td>Spatial Axes Convention</td>
<td>mcAxesConv3DCartesianXYZ</td>
<td>mc.MC_AXES_CONVENTION</td>
<td>A 3D group consisting of X, Y, Z. The order of the translational axes within the configuration defines the translation order.</td>
</tr>
<tr>
<td>Spatial Axes Convention</td>
<td>mcAxesConv4DCartesianXYZC</td>
<td>mc.MC_AXES_CONVENTION</td>
<td>A 4D group consisting of X, Y, Z and a rotational axis around Z (C). The order of the translational and rotational axes within the configuration defines the translation and rotation order.</td>
</tr>
</tbody>
</table>

**Additional Axes Count**

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

**Blending Strategy**

Specifies the blending strategy.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blending Strategy</td>
<td>mcBlendingGeo</td>
<td>mc.MC_BLENDING_STRATEGY</td>
<td>Blending path is defined geometrically and then run with the dynamics allowed for the path.</td>
</tr>
<tr>
<td>Blending Strategy</td>
<td>mcBlendingSuperpos</td>
<td>mc.MC_BLENDING_STRATEGY</td>
<td>Blending path results dynamically from the superposition of two segments within the blending area.</td>
</tr>
</tbody>
</table>

**Time-Override Ramp Time**

Ramp time for override change from 0 % to 100 %.

**GeoBlending-specific parameters**

**Blending Path Type**

Defines the geometry that is used for the blending path.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blending Path Type</td>
<td>mcBlendingPathTypeIgnore</td>
<td>mc.MC_BLENDING_PATH_TYPE</td>
<td>No blending.</td>
</tr>
<tr>
<td>Blending Path Type</td>
<td>mcBlendingPathTypePoly5</td>
<td>mc.MC_BLENDING_PATH_TYPE</td>
<td>Blending path uses a polynomial of order five.</td>
</tr>
</tbody>
</table>
5.3 MC Group with Pick-and-Place

The MC Group connects axes to perform a multi-dimensional movement. For new projects the usage of "MC Group Coordinated Motion" is recommended.

Axis Conventions

Tab: Parameter (Init). There are three axis conventions that can be set.

The axis conventions define how the axes in the axis group are interpreted. They define the axis group dimension and therefore the number of axes that have to be added and the way each one of the added axes is interpreted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axes Convention</td>
<td>mcAxesConv2DCartesianXY</td>
<td>MC.MC_AXES_CONVENTION</td>
<td>A 2D group consisting of X, Y. The order of the translational axes within the configuration defines the translation order.</td>
</tr>
<tr>
<td>Axes Convention</td>
<td>mcAxesConv3DCartesianXYZ</td>
<td>MC.MC_AXES_CONVENTION</td>
<td>A 3D group consisting of X, Y, Z. The order of the translational axes within the configuration defines the translation order.</td>
</tr>
<tr>
<td>Axes Convention</td>
<td>mcAxesConv4DCartesianXYZC</td>
<td>MC.MC_AXES_CONVENTION</td>
<td>A 4D group consisting of X, Y, Z and a rotational axis around Z (C). The order of the translational and rotational axes within the configuration defines the translation and rotation order.</td>
</tr>
</tbody>
</table>
Axis Group Parameters of the Pick-and-Place Kernel

Tab: Parameter (Init).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NET Cycle Time Divisor</td>
<td></td>
<td>UINT</td>
<td>Improves accuracy caused by time discretization.</td>
</tr>
<tr>
<td>Time-Override Ramp Time</td>
<td>S</td>
<td>LREAL</td>
<td>Ramp time for override change from 0 % to 100 %.</td>
</tr>
</tbody>
</table>

The Pick and Place setpoint generator was especially designed for the requirements of Pick and Place applications. It is intended for movements for which the exact path dynamics are not that important, but the user wants to move as fast as possible from one point to another. Therefore it is allowed that the algorithms violates path dynamics restrictions while in tolerance ball. Axes 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation Convention</td>
<td>Convention used for the calculation of rotations. Default is DIN9300 Z&quot;Y'X where Rotation 3 is the parameter for Z&quot;, Rotation 2 is the parameter for Y' and Rotation 1 is the parameter for X.</td>
<td></td>
</tr>
<tr>
<td>Definition Direction</td>
<td>Indicates the direction in which the displacement is programmed (from the point of view of the reference system or the MCS).</td>
<td></td>
</tr>
<tr>
<td>Translation $x$</td>
<td>Translation in the $x$-axis direction.</td>
<td>mm</td>
</tr>
<tr>
<td>Translation $y$</td>
<td>Translation in the $y$-axis direction.</td>
<td>mm</td>
</tr>
<tr>
<td>Translation $z$</td>
<td>Translation in the $z$-axis direction.</td>
<td>mm</td>
</tr>
<tr>
<td>Rotation 1</td>
<td>Rotation axis is defined by the Rotation Convention.</td>
<td>°</td>
</tr>
<tr>
<td>Rotation 2</td>
<td>Rotation axis is defined by the Rotation Convention.</td>
<td>°</td>
</tr>
<tr>
<td>Rotation 3</td>
<td>Rotation axis is defined by the Rotation Convention.</td>
<td>°</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Default velocity for synchronization.</td>
<td>mm s$^{-1}$</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Default acceleration for synchronization.</td>
<td>mm s$^{-2}$</td>
</tr>
<tr>
<td>Deceleration</td>
<td>Default deceleration for synchronization.</td>
<td>mm s$^{-2}$</td>
</tr>
<tr>
<td>Jerk</td>
<td>Default jerk for synchronization.</td>
<td>mm s$^{-3}$</td>
</tr>
<tr>
<td>Default Tracking Behavior</td>
<td>Conveyor tracking behavior after InSync has been reached.</td>
<td></td>
</tr>
<tr>
<td>Synchronization Tolerance Distance</td>
<td>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.</td>
<td>mm</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start node</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>End node</td>
<td>Object ID of the end point of the coordinate transformation. E.g. a point on the Conveyor Belt.</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

Insert Instance...
Reload

File: C:\TwincAT\3.1\Config\Modules\TcNcKino.mc
3. Insert a Coordinate Frame.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start node</strong></td>
<td>First spatial node. Coordinates are interpreted in respect to this reference node. 0x0 represents the world coordinate system (WCS).</td>
</tr>
<tr>
<td><strong>End node</strong></td>
<td>Last spatial node. This node is moved in respect to the start node. 0x0 represents the world coordinate system (WCS).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>CS</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start node</strong></td>
<td>00000000</td>
<td></td>
<td></td>
<td>OTCID</td>
</tr>
<tr>
<td><strong>End node</strong></td>
<td>01010040</td>
<td></td>
<td>Pallet (Coordinate Frame)</td>
<td>OTCID</td>
</tr>
</tbody>
</table>
5. Enter relevant Coordinate Frame Parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>CS</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation convention</td>
<td>Rotation.Z3Y2X1.DIN9300</td>
<td></td>
<td></td>
<td>MC.CoordInterpretation.SO3</td>
</tr>
<tr>
<td>Definition direction</td>
<td>toReference</td>
<td></td>
<td></td>
<td>MC.ReferenceDefDir</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kinematic</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation X</td>
<td>0.0</td>
<td></td>
<td>mm</td>
<td>LREAL</td>
</tr>
<tr>
<td>Translation Y</td>
<td>150.0</td>
<td></td>
<td>mm</td>
<td>LREAL</td>
</tr>
<tr>
<td>Translation Z</td>
<td>0.0</td>
<td></td>
<td>mm</td>
<td>LREAL</td>
</tr>
<tr>
<td>Rotation 1</td>
<td>0.0</td>
<td></td>
<td>°</td>
<td>LREAL</td>
</tr>
<tr>
<td>Rotation 2</td>
<td>0.0</td>
<td></td>
<td>°</td>
<td>LREAL</td>
</tr>
<tr>
<td>Rotation 3</td>
<td>0.0</td>
<td></td>
<td>°</td>
<td>LREAL</td>
</tr>
</tbody>
</table>

6. Link the inserted Node Connector to the PLC.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GVL.oidEndNode_PALLET</td>
<td>0101030</td>
<td>Pallet (Node Connector)</td>
<td>Tc3_McCoordinatedMotion.MC_COORD_REF</td>
</tr>
</tbody>
</table>
Finally, you can insert the `MC_SetCoordinateTransform` function block.

```plaintext
VAR
   AxisGroup : AXES_GROUP_REF;
MC_SetCoordinateTransform_0: MC_SetCoordinateTransform;
END_VAR

MC_SetCoordinateTransform_0(
   AxesGroup:= AxisGroup,
   Execute:= ,
   CoordTransform:= GVL.qldEndNode_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.
2. Insert a Node Connector.
Insert TcCom Object

Search: Name: CB (Node Connector)  OK

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]

Files: \C:\TwinCAT\3.1\Config\Modules\TcNoK.in.mo
3. Insert a Coordinate Frame.
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.
Spatial Configuration

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td></td>
</tr>
<tr>
<td>Start node</td>
<td>First spatial node. Coordinates are interpreted in respect to this reference node. 0x0 represents the world coordinate system (WCS).</td>
</tr>
<tr>
<td>End node</td>
<td>Last spatial node. This node is moved in respect to the start node. 0x0 represents the world coordinate system (WCS).</td>
</tr>
</tbody>
</table>

6. Enter relevant Coordinate Frame Parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>CS</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start node</td>
<td>00000000</td>
<td></td>
<td>OTCID 0x05010108</td>
<td></td>
</tr>
<tr>
<td>End node</td>
<td>01010050</td>
<td></td>
<td>CB (Conveyor Tracking)</td>
<td>OTCID 0x05010107</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td></td>
</tr>
<tr>
<td>Rotation convention</td>
<td>Set the interpretation of the rotation angles.</td>
</tr>
<tr>
<td>Definition direction</td>
<td>Set the definition direction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>CS</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation X</td>
<td>Translation in x-direction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation Y</td>
<td>Translation in y-direction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation Z</td>
<td>Translation in z-direction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation 1</td>
<td>Rotation angle 1, interpretation set by rotation convention</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation 2</td>
<td>Rotation angle 2, interpretation set by rotation convention</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation 3</td>
<td>Rotation angle 3, interpretation set by rotation convention</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7. Enter relevant Conveyor Tracking Parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>CS</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>2000.0</td>
<td></td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Acceleration</td>
<td>1500.0</td>
<td></td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Deceleration</td>
<td>1500.0</td>
<td></td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Jerk</td>
<td>25000.0</td>
<td></td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>Default Tracking Behavior</td>
<td>mcTrackingBehaviorDynLimited</td>
<td></td>
<td>MC_MC_DEFAULT_TRACKING_BEHAVIOR</td>
<td></td>
</tr>
<tr>
<td>Synchronization Tolerance Distance</td>
<td>0.0</td>
<td></td>
<td>mm</td>
<td>LREAL</td>
</tr>
</tbody>
</table>

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
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>GVL.oidConveyorBelt</td>
<td>01010030</td>
<td>CB (Node Connector)</td>
<td>Tc3_McCoordinatedMotion.MC_COORD_REF</td>
</tr>
</tbody>
</table>
Finally, you can insert the `MC_TrackConveyorBelt` function block.

```plaintext
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.crdConveyorBelt,
 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.
Workpieces have to be put from the storage into free carrier places moving 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 $PCS$. 
Workpieces have to be taken from the upper conveyor belt and put onto 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 a conveyor belt can be located within a UCS.
7 PLC Libraries

7.1 Tc3_McCoordinatedMotion

The Tc3_McCoordinatedMotion library is used for TF5410 TC3 Motion Collision Avoidance and also for TF5420 TC3 Motion Pick-and-Place.

### Overview

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Description</th>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
<th>MC Group with Pick-and-Place</th>
<th>MC Group Coordinated Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Administrative</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_AddAxisToGroup [p.38]</td>
<td>Adds an axis to a motion group.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MC_GroupDisable [p.40]</td>
<td>Disables a motion group.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MC_GroupEnable [p.41]</td>
<td>Enables a motion group.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MC_GroupReadError [p.42]</td>
<td>Reads the error id of a group.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MC_GroupReadStatus [p.43]</td>
<td>Reads the group state.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MC_GroupSetOverride [p.45]</td>
<td>Sets the override of a group and returns the actual override value.</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>MC_RemoveAxisFromGroup [p.47]</td>
<td>Removes an axis from a group.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MC_SetCoordinateTransformation [p.48]</td>
<td>Activates a reference system.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_TrackConveyorBelt [p.49]</td>
<td>Assists in synchronizing velocity to an object moving along a straight line through space.</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>MC_UngroupAllAxes [p.51]</td>
<td>Disables a group and removes all axes.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>UDINT_TO_IDENTINGROUP [p.52]</td>
<td>Converts an integer value to IDENT_IN_GROUP_REF, so axes without special interpretation can be added to a group.</td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Motion</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_GroupHalt [p.53]</td>
<td>Stops a group without locking it for further motion commands.</td>
<td></td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>MC_GroupStop [p.55]</td>
<td>Stops a group and locks it for further motion commands.</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## PLC Libraries

### TF5420

#### TC3 Motion Collision Avoidance

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Description</th>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC_MoveLinearAbsolutePreparation</strong> [6]</td>
<td>Adds an absolute linear movement to a table of motion segments.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_MoveCircularAbsolutePreparation</strong> [58]</td>
<td>Adds an absolute circular movement to a table of motion segments.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_MovePath</strong> [61]</td>
<td>Executes a table of motion segments.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_BlockerPreparation</strong> [62]</td>
<td>Appends a blocking job to the table of segments in the structure PathData.</td>
<td>[X]</td>
<td>[X]</td>
</tr>
<tr>
<td><strong>MC_ReleaseBlocker</strong> [63]</td>
<td>Resolves a blocking job that is blocking further execution of the path.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_GroupReadBlockerStatus</strong> [64]</td>
<td>Reads the current blocker status.</td>
<td>[X]</td>
<td>[X]</td>
</tr>
<tr>
<td><strong>MC_DwellTimePreparation</strong> [65]</td>
<td>Appends a standstill job with a defined time to the table of segments in the structure PathData.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
</tbody>
</table>

#### MC Group Motion

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Description</th>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IDENT_IN_GROUP_REF</strong> [66]</td>
<td>Defines how an axis is interpreted in a group.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_CIRC_MODE</strong> [67]</td>
<td>The circle mode defines which definition is used to program a circle.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_CIRC_PATHCHOICE</strong> [71]</td>
<td>The datatype defines the rotation direction of a circle.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_PATH_DATA_REF</strong> [72]</td>
<td>Represents the path to be executed at <strong>MC_MovePath</strong> [61].</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>ClearPath</strong> [72]</td>
<td>Resets the path represented by <strong>MC_PATH_DATA_REF</strong> [72].</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_TRANSITION_MODE</strong> [73]</td>
<td>Characterizes the way a segment transition is executed.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_COORD_REF</strong> [75]</td>
<td>Object Id of a Coordinate System.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
</tbody>
</table>

### Structures and Enums

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Description</th>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IDENT_IN_GROUP_REF</strong> [66]</td>
<td>Defines how an axis is interpreted in a group.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_CIRC_MODE</strong> [67]</td>
<td>The circle mode defines which definition is used to program a circle.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_CIRC_PATHCHOICE</strong> [71]</td>
<td>The datatype defines the rotation direction of a circle.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_PATH_DATA_REF</strong> [72]</td>
<td>Represents the path to be executed at <strong>MC_MovePath</strong> [61].</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>ClearPath</strong> [72]</td>
<td>Resets the path represented by <strong>MC_PATH_DATA_REF</strong> [72].</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_TRANSITION_MODE</strong> [73]</td>
<td>Characterizes the way a segment transition is executed.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
<tr>
<td><strong>MC_COORD_REF</strong> [75]</td>
<td>Object Id of a Coordinate System.</td>
<td>[X]</td>
<td>[✓]</td>
</tr>
</tbody>
</table>
7.1.1 Function Blocks

7.1.1.1 Administrative

7.1.1.1.1 MC_AddAxisToGroup

This function block adds one axis to a group.

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

- Axes can only be added to an MC Group, if EnableRequested is FALSE, e.g. in state GroupDisabled.

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxesGroup</td>
<td>Execute</td>
</tr>
<tr>
<td>Axis</td>
<td>IdentInGroup</td>
</tr>
</tbody>
</table>

AxesGroup: Reference to a group of axes (see Cyclic Group Interface [86]).

Axis: Reference to an axis (see AXIS_REF).

Execute: The command is triggered by a rising edge at this input.

IdentInGroup: Defines the interpretation of the axis to be added to the group. For multi-dimensional movements this can be the cartesian interpretation. The global variables [66] (e.g. MCS_X) have to be used. For Collision Avoidance the function UDINT_TO_IDENTINGROUP [52] has to be used.

Using integer values for input IdentInGroup

Using integer values for input IdentInGroup is NOT supported and may lead to incompatibility to future releases. Using integer values the project may not build anymore. It is recommended to use global variables [66] (e.g. MCS_X) or conversion function UDINT_TO_IDENTINGROUP [52].
VAR_OUTPUT
VAR_OUTPUT
- Done : BOOL;
- Busy : BOOL;
- Error : BOOL;
- ErrorId : UDINT;
END_VAR

Done: This output becomes TRUE if the command has succeeded.

Busy: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

Error: This output becomes TRUE if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Example for TC3 Motion Pick-and-Place

Multi-dimensional movements

Multi-dimensional movements are merely employed using TF5420.

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

Example for TF5410 TC3 Motion Collision Avoidance

PTP with Collision Avoidance

PTP with Collision Avoidance is merely employed using TF5410.

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],
  Execute := TRUE);
END_FOR
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.1.2 MC_GroupDisable

This function block disables the group. After successful execution the group changes into group state GroupDisabled (see Group State Diagram).

**NOTE**
Disabling a Moving Group results in an instant stop.
The sudden stopping of axes is likely to exceed its allowed deceleration limits. Depending of the drive hardware this might result in power peaks and runtime errors.
To avoid this situation use MC_GroupHalt or MC_GroupStop before executing MC_GroupDisable.

VAR_IN_OUT

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

VAR_INPUT

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

Execute: The command is triggered by a rising edge at this input.

VAR_OUTPUT

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

Done: This output becomes TRUE if the command has succeeded.
**Busy**: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.1.1.3 MC_GroupEnable

![MC_GroupEnable](image)

This function block enables the group. If it succeeds and all axes are ready the group is in group state GroupStandby afterwards (see Group State Diagram [9]).

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

### VAR_IN_OUT

VAR_IN_OUT

- AxesGroup : AXES_GROUP_REF;

**AxesGroup**: Reference to a group of axes (see Cyclic Group Interface [86])

### VAR_INPUT

VAR_INPUT

- Execute : BOOL;

**Execute**: The command is triggered by a rising edge at this input.

### VAR_OUTPUT

VAR_OUTPUT

- Done : BOOL;
- Busy : BOOL;
- Error : BOOL;
- ErrorId : UDINT;

**Done**: This output becomes TRUE if the command has succeeded.
Busy: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

Error: This output becomes TRUE if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.1.4 MC_GroupReadError

This function block returns the error code of the group. It does not return errors at function blocks (e.g. invalid parameterization).

VAR_IN_OUT

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxesGroup</td>
</tr>
</tbody>
</table>

VAR_INPUT

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
</tr>
</tbody>
</table>

Enable: The command is executed as long as Enable is active.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
</tr>
<tr>
<td>Busy</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>ErrorId</td>
</tr>
<tr>
<td>GroupErrorId</td>
</tr>
</tbody>
</table>

Valid: This output indicates that other output values at this function block are valid.

Busy: This output becomes TRUE when the command is started with Enable as long as the function block is executing the command.

Error: This output becomes TRUE if an error occurs as the command is executed.
ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

GroupErrorId: Returns the Error ID of the group (see NC error documentation).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.1.5 MC_GroupReadStatus

This function block reads the state of an axes group (see Group State Diagram [9]).

VAR_IN_OUT

VAR_IN_OUT
AxesGroup : AXES_GROUP_REF;
END_VAR

VAR_INPUT

VAR_INPUT
Enable : BOOL;
END_VAR

Enable: The command is executed as long as Enable is active.

VAR_OUTPUT

VAR_OUTPUT
Valid : BOOL;
Busy : BOOL;
GroupMoving : BOOL;
GroupHoming : BOOL;
GroupErrorStop : BOOL;
GroupNotReady : BOOL;
GroupStandby : BOOL;
GroupStopping : BOOL;
GroupDisabled : BOOL;
ConstantVelocity : BOOL;
END_VAR

TF5410 TC3 Motion Collision Avoidance
TF5420 TC3 Motion Pick-and-Place

TF5420 TC3 Motion Coordinated Motion

MC Group with Pick-and-Place ✔ ✔ ✔
Valid: This output indicates that other output values at this function block are valid.

Busy: This output becomes TRUE when the command is started with Enable as long as the function block is executing the command.

Error: This output becomes TRUE if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

GroupMoving: Group is in state GroupMoving (see Group State Diagram [9]).

GroupHoming: Group is in state GroupHoming (see Group State Diagram [9]).

GroupErrorStop: Group is in state GroupErrorStop (see Group State Diagram [9]).

GroupNotReady: Group is in state GroupNotReady (see Group State Diagram [9]).

GroupStandby: Group is in state GroupStandby (see Group State Diagram [9]).

GroupStopping: Group is in state GroupStopping (see Group State Diagram [9]).

GroupDisabled: Group is in state GroupDisabled (see Group State Diagram [9]).

ConstantVelocity: Group is moving with constant velocity on commanded path.

Accelerating: Group is accelerating on commanded path.

Decelerating: Group is decelerating on commanded path.

InPosition: Group has reached target position.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.1.6 MC_GroupReset

This function block 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 Group State Diagram [9]).
If this function block is called while there is no error, it has no effect.

**VAR_IN_OUT**

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

**AxesGroup**: Reference to a group of axes (see Cyclic Group Interface [86]).

**VAR_INPUT**

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

**Execute**: The command is triggered by a rising edge at this input.

**VAR_OUTPUT**

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

**Done**: This output becomes TRUE if the command has succeeded.

**Busy**: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**7.1.1.1.7 MC_GroupSetOverride**

This function block MC_GroupSetOverride changes the group override. A change works with a certain delay. Override input is valid between 0 [0 %] and 1 [100 %]. If the value is set outside this range, it is set automatically to the respective limit.
The behavior for override changes referring to the **MC Group** can be set as Axis Group Parameter.

**MC_GroupSetOverride is not implemented yet for CA-Group**

The function block `MC_GroupSetOverride` is in this release only implemented for the MC Group and will be rejected if used with the CA Group.

**VAR_IN_OUT**

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

**VAR_INPUT**

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

**Enable**: The command is executed as long as Enable is active.

**VelFactor**: The override is set to this value (value range between 0 [0%] and 1 [100%]).

**VAR_OUTPUT**

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

**Enabled**: This output signals that the `VelFactor` has been set successfully. The `VelFactor` exhibits the type of an override factor.

**Busy**: This output becomes `TRUE` when the command is started with `Enable` as long as the function block is executing the command.

**Error**: This output becomes `TRUE` if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes `0x4nnn` and `0x8nnn`).

**ActualVelFactor**: Override that is currently active in the group (value range between 0 [0%] and 1 [100%]).

**Example**

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

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

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
</tbody>
</table>
7.1.1.8 MC_RemoveAxisFromGroup

This function block removes one axis from the axis group.

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

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

Success of Function block
The function block always returns DONE if the axis is not part of the group anymore. This means it also returns DONE if the axis has not been in the group before calling the function block.

VAR_IN_OUT

VAR_INPUT

VAR_OUTPUT

Done : BOOL;
Busy : BOOL;
Error : BOOL;
ErrorId : UDINT;

Execute: The command is triggered by a rising edge at this input.

IdentInGroup: Defines the interpretation of the axis to be added to the group. For multi-dimensional movements this can be the cartesian interpretation. The global variables (e.g. MCS_X) have to be used. For Collision Avoidance the function UDINT_TO_IDENTINGROUP [52] has to be used.

Using integer values for input IdentInGroup
Using integer values for input IdentInGroup is NOT supported and may lead to incompatibility to future releases. Using integer values the project may not build anymore. It is recommended to use global variables (e.g. MCS_X) or conversion function UDINT_TO_IDENTINGROUP [52].
Busy: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

Error: This output becomes TRUE if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.1.9 MC_SetCoordinateTransform

Activates a reference system.

Movements executed will use the Coordinate System that has been set after the command has succeeded indicated by Done = TRUE.

In changing the reference system a conveyor belt can be decoupled.

Use Case for Changing the Reference System

Using MC_SetCoordinateTransform and changing the reference system the MC group can be decoupled.

VAR_IN_OUT

AxesGroup: Reference to a group of axes (see Cyclic Group Interface [86]).

VAR_INPUT

Execute: The command is triggered by a rising edge at this input.

CoordTransform: Reference to a Coordinate System (see MC_COORD_REF [75]).
VAR_OUTPUT

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

Done: This output becomes TRUE if the command has succeeded.

Busy: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When 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: Active indicates that the command is executed. If the command was queued, it becomes active once a running command is completed.

CommandAborted: This output becomes TRUE if the command has been interrupted by another command.

Error: This output becomes TRUE if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation.

Requirements

<table>
<thead>
<tr>
<th>Development Environment</th>
<th>Target System Type</th>
<th>PLC Libraries to be Linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4022.25</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.6.03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.10  MC_TrackConveyorBelt

The Mc_TrackConveyorBelt function block activates a moving reference system. Thereby, it synchronizes the AxesGroup to the ConveyorBelt on velocity.

Synchronization to a position requires a motion command.

Thus, the function block assists in synchronizing to an object moving along a straight line through space. E.g. products moving on a Conveyor Belt or another Transport System.

The Conveyor Belt Origin 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 input to the function block.
Synchronization dynamics may be input to the function block.

Movements performed after \texttt{Active = TRUE} are synchronized with the Conveyor Belt.

Executing \texttt{MC\_TrackConveyorBelt} with a further instance synchronizes directly to a second conveyor belt.

In changing the reference system a conveyor belt can be decoupled.

### Use Case for Changing the Reference System

Using \texttt{MC\_TrackConveyorBelt} and changing the reference system the MC group can be decoupled. The reference system can be changed with \texttt{MC\_SetCoordinateTransform}.

#### VAR-IN-OUT

<table>
<thead>
<tr>
<th>VAR-IN-OUT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxesGroup</td>
<td>Reference to a group of axes (see \texttt{Cyclic Group Interface [86]}).</td>
</tr>
<tr>
<td>ConveyorBelt</td>
<td>Reference to an axis. Reference to the conveyor axis.</td>
</tr>
</tbody>
</table>

#### VAR-INPUT

<table>
<thead>
<tr>
<th>VAR-INPUT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>The command is triggered by a rising edge at this input.</td>
</tr>
<tr>
<td>CoordTransform</td>
<td>Reference to a Coordinate System (see \texttt{MC_COORD_REF [75]}).</td>
</tr>
<tr>
<td>InitialObjectPos</td>
<td>Pointer to array [1..InitialObjectPosCount].</td>
</tr>
<tr>
<td>InitialObjectPosCount</td>
<td>Dimension of the \texttt{InitialObjectPos} vector.</td>
</tr>
<tr>
<td>MasterRefPos</td>
<td>Touch probe position.</td>
</tr>
<tr>
<td>Velocity</td>
<td>The velocity for synchronization. The velocity has to exceed the conveyor belt velocity. The velocity is restricted by the maximal axis velocity.</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Used in the conveyor tracking object. The acceleration for synchronization. The acceleration has to exceed the value 1.0. The acceleration is restricted by the maximal axis acceleration. If no value is entered, the default axis acceleration is used.</td>
</tr>
<tr>
<td>Deceleration</td>
<td>Used in the conveyor tracking object. The deceleration for synchronization. The deceleration has to exceed the value 1.0. The deceleration is restricted by the maximal axis deceleration. If no value is entered, the default axis deceleration is used.</td>
</tr>
<tr>
<td>Jerk</td>
<td>The jerk for synchronization. The jerk has to exceed the value 100.0. If no value is entered, the default axis jerk is used. The maximal jerk is not restricted.</td>
</tr>
</tbody>
</table>

#### VAR-OUTPUT

<table>
<thead>
<tr>
<th>VAR-OUTPUT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InSync</td>
<td></td>
</tr>
<tr>
<td>Busy</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>CommandAborted</td>
<td></td>
</tr>
</tbody>
</table>
InSync: The `InSync` output becomes `TRUE` first time as soon as the slave is synchronized to the velocity. If the slave dynamic is too low to follow the master movement, the `InSync` output might be reset to `FALSE` and the slave axis starts to synchronize again.

**Velocity Synchronization: Active and InSync**

The `MC_TrackConveyorBelt` function block synchronizes the `AxesGroup` to the velocity of the `ConveyorBelt` axis. Thereby, the function block uses the given `Acceleration`, `Deceleration` and `Jerk`. When this synchronization movement starts, `Active` will be set to `TRUE`. When the velocity of the `ConveyorBelt` is achieved, `InSync` will be set to `TRUE`. The synchronization status is continuously monitored and indicated with `InSync`.

**Conveyor Motion, Default Tracking Behavior and InSync**

After the `InSync` output signal has been set, there are two options for maintaining synchronization.

- `mcTrackingBehaviorDynLimited`
  This behavior is the default (MC_Default) tracking behavior. The `AxesGroup` maintains velocity synchronization to the `ConveyorBelt` using the given `Acceleration`, `Deceleration` and `Jerk`.

- `mcTrackingBehaviorStayInSync`
  The `AxesGroup` maintains velocity synchronization to the `ConveyorBelt` with non-limited `Acceleration`, `Deceleration` and `Jerk`.

**Position Synchronization: MasterRefPos and InitialObjectPos**

The function blocks `MC_TrackConveyorBelt` and `MC_MovePath` are designed to be used together to flexibly synchronize to a moving target position.

After `MC_TrackConveyorBelt`.`Active` has been set to `TRUE`, `InitialObjectPos` and the distance to `MasterRefPos` will be appended to the next call of `MC_MovePath`. `MC_TrackConveyorBelt`.`InSync` = `TRUE` and `MC_MovePath`.`Done` = `TRUE` indicate the synchronized position is achieved.

**Busy:** This output becomes `TRUE` when the command is started with `Execute` as long as the function block is executing the command. When `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:** If `Active` is `TRUE`, the function block controls the group.

**CommandAborted:** This output becomes `TRUE` if the command has been interrupted by another command.

**Error:** This output becomes `TRUE` if an error occurs as the command is executed.

**ErrorId:** Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation.

### Requirements

<table>
<thead>
<tr>
<th>Development Environment</th>
<th>Target System Type</th>
<th>PLC Libraries to be Linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4022.25</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.6.03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 7.1.1.1.11 MC_UngroupAllAxes
This function block removes all axes and disables the group. If the function block succeeds group is in group state GroupDisabled afterwards (see Group State Diagram [9]).

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxesGroup : AXES_GROUP_REF;</td>
</tr>
</tbody>
</table>

**VAR_INPUT**

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute : BOOL;</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done : BOOL;</td>
</tr>
<tr>
<td>Busy : BOOL;</td>
</tr>
<tr>
<td>Error : BOOL;</td>
</tr>
<tr>
<td>ErrorId : UDINT;</td>
</tr>
</tbody>
</table>

**Execute**:
The command is triggered by a rising edge at this input.

**Done**: This output becomes TRUE if the command has succeeded.

**Busy**: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**7.1.1.1.12 UDINT_TO_IDENTINGROUP**

<table>
<thead>
<tr>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MC Group with Pick-and-Place</td>
</tr>
<tr>
<td></td>
<td>MC Group Coordinated Motion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MC Group with Pick-and-Place</td>
</tr>
<tr>
<td></td>
<td>MC Group Coordinated Motion</td>
</tr>
</tbody>
</table>
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 [38] and MC_RemoveAxisFromGroup [47]. For axes intended for multi-dimensional movement (TF5420) see IDENT_IN_GROUP_REF [66].

Using integer values for input IdentInGroup

Using integer values for input IdentInGroup is NOT supported and may lead to incompatibility to future releases. Using integer values the project may not build anymore. It is recommended to use global variables [66] (e.g. MCS_X) or conversion function UDINT_TO_IDENTINGROUP [52].

Function UDINT_TO_IDENTINGROUP : IDENT_IN_GROUP_REF

| VAR_INPUT | 1d : UDINT; |
| END_VAR |

id: the unique identifier an axis shall have in the group. This does not have to be the Axis-Id from the cyclic axis interface.

Return value

IDENT_IN_GROUP_REF [66]: Converts an integer value, so a PTP axis can be added to a motion group.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.2 Motion

7.1.1.2.1 MC_GroupHalt

The function block MC_GroupHalt stops a group with a defined braking ramp. In contrast to "MC_GroupStop [55]" the group is not locked against further motion commands. The group can therefore be restarted through a further command during the braking ramp or after it has come to a halt.

**NOTE**

MC_GroupHalt not implemented for MC-Group with Pick-and-Place

The function block MC_GroupHalt is only implemented for the MC Group Coordinated Motion and for PTP-Movements with Collision Avoidance (CA-Group). If it is used with another group type the command will be rejected.
MC_GroupHalt clears the active Coordinate Transformation and deletes all queued jobs.

### VAR_IN_OUT

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
<th>AXES_GROUP_REF;</th>
</tr>
</thead>
</table>

**VAR_INPUT**

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
<th>AXES_GROUP_REF;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>BOOL;</td>
</tr>
<tr>
<td>Deceleration</td>
<td>MC_LREAL := MC_DEFAULT;</td>
</tr>
<tr>
<td>Jerk</td>
<td>MC_LREAL := MC_DEFAULT;</td>
</tr>
</tbody>
</table>

**Execute**: The command is triggered by a rising edge at this input.

**Deceleration**: [mm/s²]. The deceleration can be programmed as a scalar value (>0) or "Special Input Values [88]" may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values.

**Jerk**: [mm/s³]. The jerk can be programmed as a scalar value (>0) or "Special Input Values [88]" may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values. MC_IGNORE executes the command with unlimited jerk.

### VAR_OUTPUT

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
<th>AXES_GROUP_REF;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done</td>
<td>BOOL;</td>
</tr>
<tr>
<td>Busy</td>
<td>BOOL;</td>
</tr>
<tr>
<td>Active</td>
<td>BOOL;</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>BOOL;</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL;</td>
</tr>
<tr>
<td>ErrorId</td>
<td>UDINT;</td>
</tr>
</tbody>
</table>

**Done**: Becomes TRUE, if the group was stopped and has come to a standstill. As soon as group has come to a standstill the group state becomes GroupStandby (see Group State Diagram[9]).

**Busy**: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

**Active**: Active indicates that the command is executed. If the command was queued, it becomes active once a running command is completed.

**CommandAborted**: This output becomes TRUE if the command has been interrupted by another command.

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the "ADS error documentation" or in the "NC error documentation" (error codes 0x4nnn and 0x8nnn).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.1.1.2.2 MC_GroupStop

The function block stops the group and all attached axes with a defined braking ramp and blocks the axis for any motion commands. While the group is in state GroupStopping, no other function block can move any group axis (see Group State Diagram [9]). The group can only be moved again once the Execute signal has been set to FALSE after the velocity is 0.

MC_GroupStop clears the active Coordinate Transformation and deletes all queued jobs.

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
<th>VAR_INPUT</th>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxesGroup  : AXES_GROUP_REF;</td>
<td>Execute : BOOL; Deceleration : MC_LREAL := MC_DEFAULT; Jerk : MC_LREAL := MC_DEFAULT;</td>
<td>Done : BOOL; Busy : BOOL; Active : BOOL; CommandAborted : BOOL; Error : BOOL; ErrorId : UDINT;</td>
</tr>
</tbody>
</table>

**Execute**: The command is triggered by a rising edge at this input. The group is locked during the stop. The group can only be moved once the Execute signal has been set to FALSE after the axis has stopped.

**Deceleration**: [mm/s²]. The deceleration can be programmed as a scalar value (>0) or Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values.

**Jerk**: [mm/s³]. The jerk can be programmed as a scalar value (>0) or Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values. MC_IGNORE executes the command with unlimited jerk.

**Done**: Becomes TRUE, if the group was stopped and has come to a standstill. While Execute is TRUE but at least until the axes have come to standstill the group stays in state GroupStopping. Afterwards the group is in state GroupStandby (see Group State Diagram [9]).
**Busy**: Becomes TRUE when the command is started with Execute and remains TRUE as long as the command is processed. When Busy becomes FALSE again, the group is ready for a new command. After group was stopped Busy remains TRUE until the group is released with Execute=FALSE.

**Active**: Indicates that the function block has control on the group. After group was stopped Active remains TRUE until the group is released with Execute=FALSE.

**CommandAborted**: Command is aborted by disabling MC-Power of at least one axis of the group or if the group is disabled during command.

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 7.1.1.2.3 MC_MoveLinearAbsolutePreparation

The function block adds an absolute linear movement to the table of segments in the structure PathData. After creating a table it can be executed via *MC_MovePath*

<table>
<thead>
<tr>
<th><strong>TF5410</strong></th>
<th><strong>TF5420</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>TC3 Motion Collision Avoidance</td>
<td>TC3 Motion Pick-and-Place</td>
</tr>
<tr>
<td>❌</td>
<td>☑️</td>
</tr>
<tr>
<td>☑️</td>
<td>☑️</td>
</tr>
</tbody>
</table>

The function block *MC_MoveLinearAbsolutePreparation* may be called various times per cycle.

### VAR_IN_OUT

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

**PathData**: Table that contains the segments of a path. The table is written by *MC_Move…Preparation* and executed by *MC_MovePath* (see *MC_PATH_DATA_REF*).

- **Resetting a table**

  A table is not reset at execution. To reset call method `ClearPath()` of *MC_PATH_DATA_REF*.
VAR_INPUT

Position : POINTER TO LREAL;
PositionCount : UDINT;
Velocity : MC_LREAL := MC_INVALID;
Acceleration : MC_LREAL := MC_DEFAULT;
Jerk : MC_LREAL := MC_DEFAULT;
BufferMode : MC_BUFFERMODE := mcAborting;
TransitionMode : MC_TRANSITION_MODE := mcTransModeNone;
TransitionParameter : POINTER TO LREAL;
TransitionParameterCount : UDINT;
InvokeId : UDINT;
END_VAR

Position: Pointer to an array [1..PositionCount] of target position vector.

PositionCount: Dimension of position vector. Needs to be equal to the number of axes in axes convention (see MC Group Coordinated Motion [16] or MC Group with Pick-and-Place [18]).

Velocity: The maximum velocity for the programmed segment. The velocity does not always need to be reached. The velocity must be set >0.

Acceleration: Maximum path acceleration for the programmed segment. Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values.

Deceleration: Maximum path deceleration for the programmed segment. Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values.

Jerk: Path jerk for the programmed segment. Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values.

BufferMode: Specifies how successive travel commands are to be processed (see MC BUFFER_MODE [76]).

Transition Mode: Defines the blending mode (see MC_TRANSITION_MODE [73]).

TransitionParameter: Pointer to array [1..TransitionParameterCount] of the blending parameters. Transition parameters define the blending from the last programmed position (see MC_TRANSITION_MODE [73]).

TransitionParameterCount: Number of blending parameters.

InvokeId: Segment Id for analysis purpose.

VAR_OUTPUT

Error : BOOL;
ErrorId : UDINT;
END_VAR

Error: This output becomes TRUE if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_MC2CoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.1.1.2.4 MC_MoveCircularAbsolutePreparation

The function block adds an absolute circular movement to the table of segments in the structure PathData. After creating a table it can be executed via MC_MovePath. The function block MC_MoveCircularAbsolutePreparation may be called various times per cycle.

### VAR_IN_OUT

<table>
<thead>
<tr>
<th>PathData</th>
<th>MC_PATH_DATA_REF</th>
</tr>
</thead>
</table>

**PathData**: Table that contains the segments of a path. The table is written by MC_Move...Preparation and executed by MC_MovePath [61] (see MC_PATH_DATA_REF [72]).

**Resetting a table**

A table is not reset at execution. To reset call method `ClearPath()` of MC_PATH_DATA_REF [72].

### 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; |

**CircMode**: Specifies by which circle definition the circle is programmed. Specifies the meaning of the input signal 'AuxPoint' (See MC_CIRC_MODE [67]).
**AuxPoint**: Pointer to an array [1..AuxPointCount] of aux point vector. The interpretation of the AuxPoint-Vector is independent from rotation convention (see MC Group Coordinated Motion [16] or MC Group with Pick-and-Place [18]) and is always (x, y, z).

**AuxPointCount**: Dimension of aux point vector. Must be 3. If a 2D-rotation convention (see MC Group Coordinated Motion [16] or MC Group with Pick-and-Place [18]) is used, the input value also has to be 3. In case of a 2D-rotation convention and CircMode of mcCircModeBorder or mcCircModeCenter the component which is independent of working plane has to be set to MC_Ignore (see MC_LREAL/Special Input Values [88]).

**EndPoint**: Pointer to an array [1..EndPointCount] of target position vector.

**EndPointCount**: Dimension of EndPoint vector. Needs to be equal to the number of axes in axes convention (see MC Group Coordinated Motion [16] or MC Group with Pick-and-Place [18]).

**PathChoice**: defines the rotation direction in respect to the normal vector. The input is ignored if input CircMode is set to mcCircModeBorder (see MC_CIRC_PATHCHOICE [71]).

**Velocity**: The maximum velocity for the programmed segment. The velocity does not always need to be reached. The velocity must be set >0.

**Acceleration**: Maximum path acceleration for the programmed segment. Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values.

**Deceleration**: Maximum path deceleration for the programmed segment. Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values. MC_MAXIMUM executes the command with the axes maximum values.

**Jerk**: Path jerk for the programmed segment. Special Input Values [88] may be used. MC_DEFAULT executes the command with default axes values.

**BufferMode**: Specifies how successive travel commands are to be processed (see MC_BUFFER_MODE [76]).

**Transition Mode**: Defines the blending mode (see MC_TRANSITION_MODE [73]).

**TransitionParameter**: Pointer to array [1..TransitionParameterCount] of the blending parameters. Transition parameters define the blending from the last programmed position (see MC_TRANSITION_MODE [73]).

**TransitionParameterCount**: Number of blending parameters.

**InvokeId**: Segment Id for analysis purpose.

**VAR_OUTPUT**

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

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

**Example Center point programming**

Assuming that a path of 4 segments as shown in the picture is to be programmed in Mode mcCircModeCenter: The user defines the center point of the circle as auxiliary point (‘AuxPoint’). When using mcCircModeCenter the input MC_CIRC_PATHCHOICE [71] defines the rotation direction. As the plane is defined by the cross product mcCircPathchoiceCounterClockwise has to 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]                                    := 0;
aTargetPos[3]                                    := -100;
aAuxPoint[1]                                     := 200;
aAuxPoint[2]                                     := 0;
aAuxPoint[3]                                     := -100;
aTransitionParam[1]                             := 0;
aTransitionParam[2]                             := 0;
fbMoveCircPrep(PathData:= path, PathChoice:= mcCircPathchoiceCounterClockwise, Velocity:= 1000, InvokeId:= 20);

aTargetPos[1]                                    := 400;
aTargetPos[2]                                    := 200;
aTargetPos[3]                                    := 0;
aAuxPoint[1]                                     := 400;
aAuxPoint[2]                                     := -100;
aAuxPoint[3]                                     := 0;
aTransitionParam[1]                             := 0;
aTransitionParam[2]                             := 0;
fbMoveCircPrep(PathData:= path, PathChoice:= mcCircPathchoiceCounterClockwise, Velocity:= 1000,
InvokeId:= 30);

aTargetPos[1] := 600;
aTargetPos[2] := -200;
aTargetPos[3] := 100;
aTransitionParam[1] := 0;
aTransitionParam[2] := 0;

fbMoveLinPrep(PathData:= path, Velocity:= 3000, InvokeId:= 40);

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.2.47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.1.2.5 MC_MovePath

The function block MC_MovePath executes a movement defined in table PathData by MC_MoveLinearAbsolutePreparation. The function block MC_MovePath is used to move a group of axes to a specified position. The movement is executed by MC_MovePath and requires a reference to an axes group and a path data table.

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
</tr>
</thead>
</table>
| AxesGroup  : AXES_GROUP_REF;
| PathData   : MC_PATH_DATA_REF; |

**AxesGroup**: Reference to a group of axes (see **Cyclic Group Interface**)

**PathData**: Table that contains the segments of a path. The table is written by MC_MoveLinearAbsolutePreparation and executed by MC_MovePath (see MC_PATH_DATA_REF).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute    : BOOL;</td>
</tr>
</tbody>
</table>

**Execute**: The command is triggered by a rising edge at this input.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done       : BOOL;</td>
</tr>
<tr>
<td>Busy       : BOOL;</td>
</tr>
<tr>
<td>Active     : BOOL;</td>
</tr>
<tr>
<td>CommandAborted : BOOL;</td>
</tr>
<tr>
<td>Error       : BOOL;</td>
</tr>
<tr>
<td>ErrorId     : UDINT;</td>
</tr>
</tbody>
</table>

TF5410  
TC3 Motion Collision Avoidance

TF5420  
TC3 Motion Pick-and-Place

TF5420  
TC3 Motion Coordinated Motion

---

The function block MC_MovePath executes a movement defined in table PathData by MC_MoveLinearAbsolutePreparation. The function block MC_MovePath is used to move a group of axes to a specified position. The movement is executed by MC_MovePath and requires a reference to an axes group and a path data table.

**VAR_IN_OUT**

<table>
<thead>
<tr>
<th>VAR_IN_OUT</th>
</tr>
</thead>
</table>
| AxesGroup  : AXES_GROUP_REF;
| PathData   : MC_PATH_DATA_REF; |

**AxesGroup**: Reference to a group of axes (see **Cyclic Group Interface**)

**PathData**: Table that contains the segments of a path. The table is written by MC_MoveLinearAbsolutePreparation and executed by MC_MovePath (see MC_PATH_DATA_REF).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute    : BOOL;</td>
</tr>
</tbody>
</table>

**Execute**: The command is triggered by a rising edge at this input.

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done       : BOOL;</td>
</tr>
<tr>
<td>Busy       : BOOL;</td>
</tr>
<tr>
<td>Active     : BOOL;</td>
</tr>
<tr>
<td>CommandAborted : BOOL;</td>
</tr>
<tr>
<td>Error       : BOOL;</td>
</tr>
<tr>
<td>ErrorId     : UDINT;</td>
</tr>
</tbody>
</table>
**Done**: This output becomes TRUE if the command has succeeded. This means that the last command defined by reference variable *PathData* was executed successfully.

**Busy**: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

**Active**: If Active is TRUE, the FB controls the axis.

**CommandAborted**: This output becomes TRUE if the command has been interrupted by another command.

**Error**: This output becomes TRUE if an error occurs as the command is executed.

**ErrorId**: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.1.2.6 MC_BlockerPreparation

This function block appends a blocking job to the table of segments in the structure *PathData*. The *PathData* table can be executed via *MC_MovePath*. The function block *MC_BlockerPreparation* may be called multiple times per cycle.

A blocking job is an entry that suspends the execution of the path until it is released with *MC_ReleaseBlocker*. As long as the blocker is not resolved, path execution is paused at this segment. Every blocker has an Id, so various blocking events in the PLC can be distinguished.

If a blocking job is active, the group state is still "moving". If the override is changed while the blocking job is active, it takes effect for the next move job. If a new job with BufferMode mcAborting is executed while the blocking job is active, the blocking job is aborted. If *MC_GroupHalt* or *MC_GroupStop* are executed while the blocking job is active, the path is terminated and the blocking job is released automatically.

### VAR_IN_OUT

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

**PathData**: Table that contains the segments of a path. The table is written by preparation functions blocks like this one and executed by *MC_MovePath* (see *MC_PATH_DATA_REF*).
VAR_INPUT

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

BlockerId: Id of the blocker. May be any UDINT >0.

BufferMode: Specifies how successive travel commands are to be processed (see MC_BUFFER_MODE [76]). Only mcBuffered and mcAborting are allowed here.

InvokeId: Segment Id for analysis purposes.

VAR_OUTPUT

VAR_OUTPUT
  Error : BOOL;
  ErrorId : UDINT;
END_VAR

Error: This output becomes TRUE, if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.7</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.2.7 MC_ReleaseBlocker

This function block releases a blocking job that is blocking further execution of the path. A blocking job is inserted into the path with MC_BlockerPreparation.

If MC_ReleaseBlocker is executed, the next blocker with the given Id is released. Blending between motion segments surrounding that blocker may be executed if permitted by those segments and still viable at the time the blocking job is released.

VAR_IN_OUT

VAR_IN_OUT
  AxesGroup : AXES_GROUP_REF;
END_VAR

AxesGroup: Reference to a group of axes (see Cyclic Group Interface [86]).
VAR_INPUT

VAR_INPUT
  Execute : BOOL;
  BlockerId : UDINT;
END_VAR

Execute: The command is triggered by a rising edge at this input.

BlockerId: Id of the blocker. May be any UDINT >0.

VAR_OUTPUT

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

Done: This output becomes TRUE if the command has succeeded.

Busy: This output becomes TRUE when the command is started with Execute as long as the function block is executing the command. When Busy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs Done, CommandAborted (if existing) or Error is set.

Error: This output becomes TRUE, if an error occurs as the command is executed.

ErrorId: Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.7</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.2.8  MC_GroupReadBlockerStatus

This function block reads the current blocker status.

VAR_IN_OUT

VAR_IN_OUT
  AxesGroup : AXES_GROUP_REF;
END_VAR

AxesGroup: Reference to a group of axes (see Cyclic Group Interface [86]).

VAR_INPUT

VAR_INPUT
  Enable : BOOL;
END_VAR
Enable: Enables reading of the actual blocker status.

VAR_OUTPUT

<table>
<thead>
<tr>
<th>Valid</th>
<th>Blocked</th>
<th>BlockerId</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>BOOL</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

Valid: Returns TRUE if a valid group type is in use. Only group type MC Group Coordinated Motion is allowed.

Blocked: Returns TRUE if a blocking job is active, i.e. currently halting path execution. Returns FALSE if no blocking job is active.

BlockerId: Id of the blocker. May be any UDINT >0.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.7</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.1.2.9 MC_DwellTimePreparation

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

VAR_IN_OUT

<table>
<thead>
<tr>
<th>PathData</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_PATH_DATA_REF</td>
</tr>
</tbody>
</table>

PathData: Table that contains the segments of a path. The table is written by preparation functions blocks like this one and executed by MC_MovePath (see MC_PATH_DATA_REF).

VAR_INPUT

<table>
<thead>
<tr>
<th>DwellTime</th>
<th>BufferMode</th>
<th>InvokeId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>MC_BUFFER_MODE := mcBuffered;</td>
<td>UDINT</td>
</tr>
</tbody>
</table>

DwellTime: Time for which the path stands still with velocity 0. Any time span >= 0 is allowed. A DwellTime equal to zero will lead to an exact stop, even if the surrounding segments would allow a transition with velocity > 0.
**BufferMode:** Specifies how successive travel commands are to be processed (see MC BUFFER MODE [76]). Only mcBuffered and mcAborting are allowed here.

**InvokedId:** Segment Id for analysis purposes.

### VAR OUTPUT

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

**Error:** This output becomes TRUE, if an error occurs as the command is executed.

**ErrorId:** Contains the command-specific error code of the most recently executed command. The error code can be found in the ADS error documentation or in the NC error documentation (error codes 0x4nnn and 0x8nnn).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.7</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.2 Datatypes

#### 7.1.2.1 IDENT_IN_GROUP_REF

<table>
<thead>
<tr>
<th>TF5410</th>
<th>TF5420</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC3 Motion Collision Avoidance</td>
<td>TC3 Motion Pick-and-Place</td>
</tr>
<tr>
<td>MC Group with Pick-and-Place</td>
<td>MC Group Coordinated Motion</td>
</tr>
</tbody>
</table>

IDENT_IN_GROUP_REF defines how an axis is interpreted in a group. For multi-dimensional movements global variables can be used. For PTP Collision Avoidance groups the function UDINT_TO_IDENTINGROUP [52] has to be called.

**Using integer values for input IdentInGroup**

Using integer values for input IdentInGroup is NOT supported and may lead to incompatibility to future releases. Using integer values the project may not build anymore. It is recommended to use global variables [66] (e.g. MCS_X) or conversion function UDINT_TO_IDENTINGROUP [52].

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 defines the rotation order. For example if one axis is defined MCS_C1 and another MCS_B2 the system rotates first around Z axis and second around Y axis.

```plaintext
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 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;
END_VAR

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.2.2 MC_CIRC_MODE

<table>
<thead>
<tr>
<th>TF5410</th>
<th>TF5420</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC3 Motion Collision Avoidance</td>
<td>TC3 Motion Pick-and-Place</td>
</tr>
<tr>
<td>![X]</td>
<td>![✓]</td>
</tr>
<tr>
<td>![✓]</td>
<td>![✓]</td>
</tr>
</tbody>
</table>

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

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

- Movement starts at the starting-point "StartPoint".
- This-point is the endpoint of the preceding move command.

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

- The user configures the auxiliary point "AuxPoint".
- The circular movement will pass through this point.

- The input parameter "PathChoice" and the datatype "MC_CIRC_PATHCHOICE" are ignored.

- 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 describe paths with more than one full rotation of the circle.
mcCircModeCenter

- **StartPoint**: 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 disambiguates between the two. See MC_CIRC_PATHCHOICE for details.

- **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 start-, center- and endpoint would be collinear and thus the plane in which the circle lies would be ambiguous. The mode mcCircModeCenter cannot be used describe paths with more than one full rotation of the circle.
**mcCircModeRadius**

**MC_CIRC_PATHCHOICE**

- Clockwise: 1 4
- Counterclockwise: 3 2
- Short segment: 3 2
- Long segment: 4 1

*E=* EndPoint
*S=* StartPoint

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

**StartPoint**
- 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 details.

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 start- and endpoint.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.2.47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.2.3 MC_CIRC_PATHCHOICE

The datatype MC_CIRC_PATHCHOICE defines the rotation direction of a circle in case mcCircModeCenter or mcCircModeRadius is selected from ENUM MC_CIRC_MODE [67].

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

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

mcCircPathchoiceClockwise: represents the circle segment with angle > 180°.

mcCircPathchoiceCounterClockwise: represents the circle segment with angle < 180°.

mcCircPathchoiceShortSegment: represents the circle segment with the smaller angle.

mcCircPathchoiceLongSegment: represents the circle segment with the larger angle.
7.1.2.4  MC_PATH_DATA_REF

MC_PATH_DATA_REF represents the path to be executed by MC_MovePath [61]. The path to be executed is written by MC_MoveLinearAbsolutePreparation [56]. 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 (see MC_PATH_DATA_REF [72]). The path table is not reset at execution. To reset call method ClearPath [72].

VAR_OUTPUT

VAR_OUTPUT
  FilledRows : UDINT;
  OccupiedBuffer : UDINT;
END_VAR

FilledRows: Number of path entries (e.g. path segments).

OccupiedBuffer: Occupied buffer size in byte. By analyzing this output the user can analyze if the end of the defined buffer will be reached.

Example

The example 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();

TIP:

The MC_PATH_DATA_REF data type 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 MC_PATH_DATA_REF data type. In particular, do not use memory functions like MEMCMP, MEMCPY, MEMSET or MEMMOVE on the MC_PATH_DATA_REF data type.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.1.2.4.1  ClearPath
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

<table>
<thead>
<tr>
<th>Name</th>
<th>TransitionParameterCount</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcTransModeNone</td>
<td>No impact</td>
<td>No blending</td>
</tr>
<tr>
<td>mcTransModeCornerDistance</td>
<td>1</td>
<td>TransitionParameter acts as tolerance ball in which the path may be left.</td>
</tr>
<tr>
<td>mcTransModeCornerDistanceAdvanced</td>
<td>2</td>
<td>TransitionParameter act as tolerance ball in which the path may be left.</td>
</tr>
</tbody>
</table>

**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 \((r_{\text{in}})\) is limited to 90\% of previous segment. \(R_{\text{out}}\) 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 \(r_{\text{out}}\) should measure at least \(2 \times r_{\text{in}}\).

**Combinations of buffer mode and transition mode**

*Note* 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.

<table>
<thead>
<tr>
<th>TM/BM</th>
<th>mcAborting</th>
<th>mcBuffered</th>
<th>mcBlendingPrevious</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcTransModeNone</td>
<td>Previous command is aborted immediately. New movement is started. Velocity in transition is 0. This combination is only allowed for the 1\textsuperscript{st} segment of a path.</td>
<td>Stop at the end of previous command. Subsequently next command is executed.</td>
<td>Not allowed</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>
### TM/BM

<table>
<thead>
<tr>
<th>TM/BM</th>
<th>mcAborting</th>
<th>mcBuffered</th>
<th>mcBlendingPrevious</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcTransModeCornerDistance</td>
<td>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.</td>
<td>Not allowed</td>
<td>Blending from last programmed command to new command</td>
<td>Not allowed</td>
</tr>
<tr>
<td>mcTransModeCornerDistanceAdvanced</td>
<td>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.</td>
<td>Not allowed</td>
<td>Blending from last programmed command to new command</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Other</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 7.1.2.6 MC_COORD_REF

<table>
<thead>
<tr>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MC Group with Pick-and-Place</td>
</tr>
<tr>
<td></td>
<td>MC Group Coordinated Motion</td>
</tr>
</tbody>
</table>

- [x]

Object Id that refers to a node connector.
7.2  Tc3_Mc3Definitions

Structures and enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
<th>MC Group with Pick-and-Place</th>
<th>MC Group Coordinated Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC_BUFFER_MODE</td>
<td>Defines how successive travel commands are to be processed.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>[76]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_COMPENSATION_TYPE</td>
<td>The value defines the compensation type.</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>[79]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_DIRECTION</td>
<td>The value determines the direction of the movement.</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>[80]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_SYNC_MODE</td>
<td>The value defines the direction in which synchronization is to be performed.</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>[80]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC_SYNC_STRATEGY</td>
<td>Defines the synchronization profile of the slave axis.</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>[81]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

```plaintext
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
```

Example:

In the following example, a move command is used to move a group from position P₀ to P₁, and then to P₂. The reference point for the different velocity profiles is always P₁. The mode specifies the velocity v₁ or v₂ 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

**Note** 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.
## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 7.2.1.2 MC_COMPENSATION_TYPE

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

```plaintext
TYPE MC_COMPENSATION_TYPE:

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

### TF5410 TC3 Motion Collision Avoidance

<table>
<thead>
<tr>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
<th>MC Group with Pick-and-Place</th>
<th>MC Group Coordinated Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

TF5420 Version: 1.5
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26 Advanced Motion Pack V3.1.6.07</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCompensations</td>
</tr>
</tbody>
</table>

### 7.2.1.3 MC_DIRECTION

(* Defines the direction of the movement (e.g. for a modulo axis). *)

```delphi
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 TC3 Motion Collision Avoidance

<table>
<thead>
<tr>
<th>TF5410 TC3 Motion Collision Avoidance</th>
<th>TF5420 TC3 Motion Pick-and-Place</th>
<th>MC Group with Pick-and-Place</th>
<th>MC Group Coordinated Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

The value determines the direction of the movement. The direction 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 target position.

- **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 library it is possible that the data type cannot be resolved uniquely (ambiguous use of name 'MC_Direction'). In this case you have to specify the namespace when using the data type (Tc3_Mc3PlanarMotion.MC_DIRECTION, Tc3_Mc3Definitions.MC_DIRECTION or Tc2_MC2.MC_DIRECTION).

### 7.2.1.4 MC_SYNC_MODE

(* Defines the direction of the synchronization position of modulo axes. *)

```delphi
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
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.7 Advanced Motion Pack V3.1.10.1</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.7</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_MccCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**7.2.1.5 MCSYNC_STRATEGY**

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

```plaintext
TYPE MC_SYNC_STRATEGY :
{  
  mcSyncStrategyLate := 16#1,
  mcSyncStrategySlow := 16#2,
  mcSyncStrategyEarly := 16#3
}
END_TYPE
```

**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 dynamic. The slave cannot violate its
dynamic restrictions and therefore cannot compensate any master acceleration, this situation will result in an
error at the FB.

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

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

Example 2: mcSyncStrateySlow

The slave starts its sync in motion as soon as the master passes (MasterSyncPosition -
MasterStartDistance) in the right direction if a MasterStartDist was set, otherwise as soon as the FB is
Active. The dynamics of the slave is reduced such that the slave reaches the SlaveSyncPos with the correct
gear ratio just in time when the master reaches the MasterSyncPos. The slave can compensate an
acceleration of the master once StartSync is set, but only until the slave reaches its maximum dynamics.

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

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

Synchronizing 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 command MC_GearInPosCA to axis. The Command becomes active while the master is still accelerating.

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

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4018.26</td>
<td>PC or CX (x86 or x64)</td>
<td>Tc3_McCollisionAvoidance, Tc3_McCoordinatedMotion, Tc2_MC2</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.1.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Diagram:**

[Chart showing synchronized motion data with various axes and time ranges]
8  Samples

Multi-dimensional motion

PnpSimpleSample
Download: https://infosys.beckhoff.com/content/1033/tf5420_tc3_advanced_pick_and_place/Resources/zip/9007208980336523.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/zip/9007208980338187.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 \texttt{NcToPlc} \[86\] and \texttt{PlcToNc} \[87\]. Both direction are divided in common and group specific data.

\begin{verbatim}
AXES_GROUP_REF
TYPE AXES_GROUP_REF :
  STRUCT
  PlcToNc AT %Q* : PLCTOMC_GROUP_REF;
  NcToPlc AT %I* : MCTOPLC_GROUP_REF;
END_STRUCT
END_TYPE
\end{verbatim}

\textbf{PlcToNc}: \texttt{PlcToNc} \[87\] 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.

\textbf{NcToPlc}: \texttt{NcToPlc} \[86\] 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 \textbf{NcToPlc}

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

\textbf{Common}

\textbf{GroupOID}: TcCOM-Object-Id (OID) of this Group.

\textbf{GroupType}: Type of this Group: 0 = Invalid (mcGroupTypeInvalid), 1 = CollisionAvoidance (mcGroupTypeCA), 2 = DXD/CNC (mcGroupTypeDxd).

\textbf{GroupStatus}: Contains information about the group status (see \texttt{GroupStatus} \[86\]).

\textbf{GroupErrorId}: Current Error Identifier (0 = no error).

\textbf{GroupAxesCount}: Number of Axes that are currently part of this Group (e.g. added via MC\_AddAxisToGroup).

\textbf{GroupStatus}:

\textbf{State}: See Group State Diagram.

1 = Disabled (mcGroupStateDisabled)
2 = Standby (mcGroupStateStandby)
3 = Moving (mcGroupStateMoving)
4 = Stopping (mcGroupStateStopping)
5 = ErrorStop (mcGroupStateErrorStop)
6 = Homing (mcGroupStateHoming)
7 = NotReady (mcGroupStateNotReady)
8 = Suspended (mcGroupStateSuspended)

\textbf{Flags} : Additional status information.

\textit{IsEnableRequested}: Defines if group is requested to be enabled or disabled.
Dxd (multi-dimensional movement)

**PathVelo**: Velocity on path without direction.

**InvokeId**: Segment Id for analysis purpose.

**CM (MC Group Coordinated Motion)**

available from V3.1.10.1

**PathVelo**: Absolute value of cartesian speed on the path.

**InvokeId**: Segment Id for analysis purpose.

**IsInBlendingSegment**: Indicates if a blending segment is active.

**RemainingTimeActiveJob**: Remaining time of the current segment.

**RemainingCartesianDistanceActiveJob**: Remaining distance for the current segment.

**ActiveBlockerId**: Id of the active blocker.

### 9.1.2 PlcToNc

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

**Common**

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

### 9.2 Index Offset Specification for MC Group Parameters

**Port 501**: `AMSPORT_R0_NCSAF: UINT := 501;`

As index group the object id (OID) of the MC group has to be quoted.

<table>
<thead>
<tr>
<th>Index Offset (Hex)</th>
<th>Access</th>
<th>Group Type</th>
<th>Data Type</th>
<th>Phys. Unit</th>
<th>Definition Range</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5030084</td>
<td>Read</td>
<td>MC Group</td>
<td>LREAL</td>
<td>mm</td>
<td>&gt;=0</td>
<td>Remaining distance of the current segment.</td>
<td>Since V3.1.6.27, only available for MC Group with Pick-and-Place</td>
</tr>
<tr>
<td>0x5030085</td>
<td>Read</td>
<td>MC Group</td>
<td>LREAL</td>
<td>s</td>
<td>&gt;=0</td>
<td>Remaining time for the current segment.</td>
<td>Since V3.1.6.27, only available for MC Group with Pick-and-Place</td>
</tr>
</tbody>
</table>

### 9.3 Differences between MC2 and MC3

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

A axes

<table>
<thead>
<tr>
<th>Maximum dynamics</th>
<th>MC2</th>
<th>MC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

PLC Library

<table>
<thead>
<tr>
<th>Default values</th>
<th>MC2</th>
<th>MC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>For dynamics parameters of type LREAL &quot;0&quot; is default value. If &quot;0&quot; is set the default parameters from the axes are used.</td>
<td>The constant MC_Default is introduced (see MC_LREAL/Special Input Values [88]). &quot;0&quot; is not interpreted as default value but as a normal value which in case of dynamics can be invalid.</td>
<td></td>
</tr>
<tr>
<td>Timing of FB outputs</td>
<td>FB returns values that were valid at the start of PLC cycle.</td>
<td>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.</td>
</tr>
<tr>
<td>Decoupling</td>
<td>A special function block can be used (e.g. MC_GearOut/MC_CamOut)</td>
<td>The slave axis is decoupled by sending another motion command with Buffermode mcAborting.</td>
</tr>
</tbody>
</table>

9.4 Glossary

Coordinate Systems

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS</td>
<td>Axis Coordinate System</td>
</tr>
<tr>
<td>DOF</td>
<td>Degree-of-Freedom</td>
</tr>
<tr>
<td>MCS</td>
<td>Machine Coordinate System</td>
</tr>
<tr>
<td>PCS</td>
<td>Programmed Coordinate System, Workpiece</td>
</tr>
<tr>
<td>UCS</td>
<td>User Coordinate System</td>
</tr>
<tr>
<td>WCS</td>
<td>World Coordinate System</td>
</tr>
</tbody>
</table>

9.5 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.

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