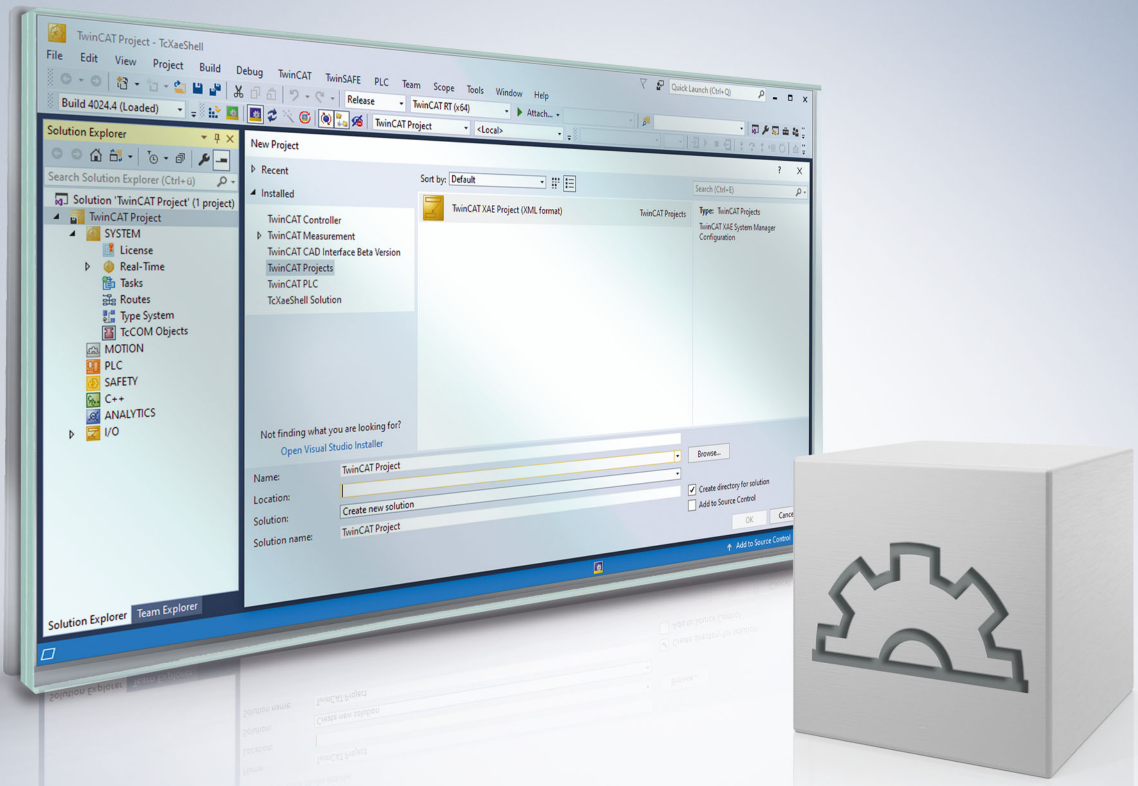
BECKHOFF New Automation Technology

Manual | EN

TF5110 - TF5113

TwinCAT 3 | Kinematic Transformation



2022-10-27 | Version: 2.2.1

Table of contents

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®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
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:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
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!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

ℹ **Tip or pointer**

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

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

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

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

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

2 Introduction

The TF5110 - TF5113 TwinCAT Kinematic Transformation software package is installed together with the TF5400 software package.

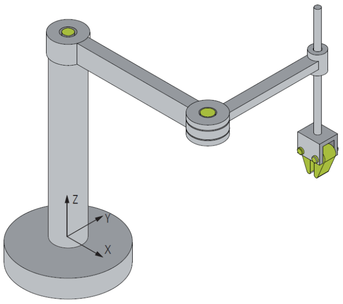
TwinCAT Kinematic Transformation

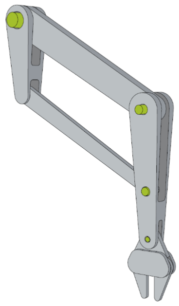
The TF5110 - TF5113 TwinCAT Kinematic Transformation is a software solution that combines robot control and conventional PLC in one system (see also https://www.beckhoff.com/tf5113/). The implementation of the entire control in one system eliminates interface losses between different CPUs for PLC, motion control and robot control. In practice, this implementation leads to a reduction of engineering costs and to reduced cycle times in the production process. In addition to the elimination of interfaces and components, the merging of PLC, robotics and motion control into one application makes the system homogeneous. Therefore, for the user there is no apparent difference in the treatment of the individual functions. Conveniently, a part on a conveyor belt operated with standard motion control can be taken and set aside by the robot quickly and handily.

Since the working area of the robot is determined by the configuration and the number of axes, it depends on a number of parameters: arm lengths, angular range, center of mass, maximum load, etc. The configuration of the arms and joints determines the kinematic structure, which is divided in two main classes: serial kinematics and parallel kinematics.

Serial kinematics

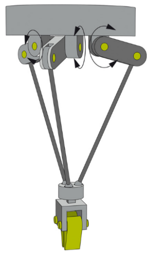
The current position of any axis always depends on the position of the preceding axis, i.e. all axes are arranged sequentially.
Examples: SCARA and crane kinematics

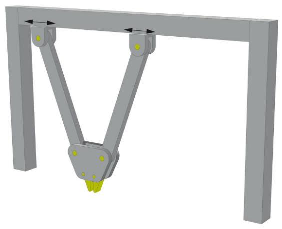




Parallel kinematics

All axes directly engage with the working platform via the kinematics.
Examples: delta kinematics, shear kinematics



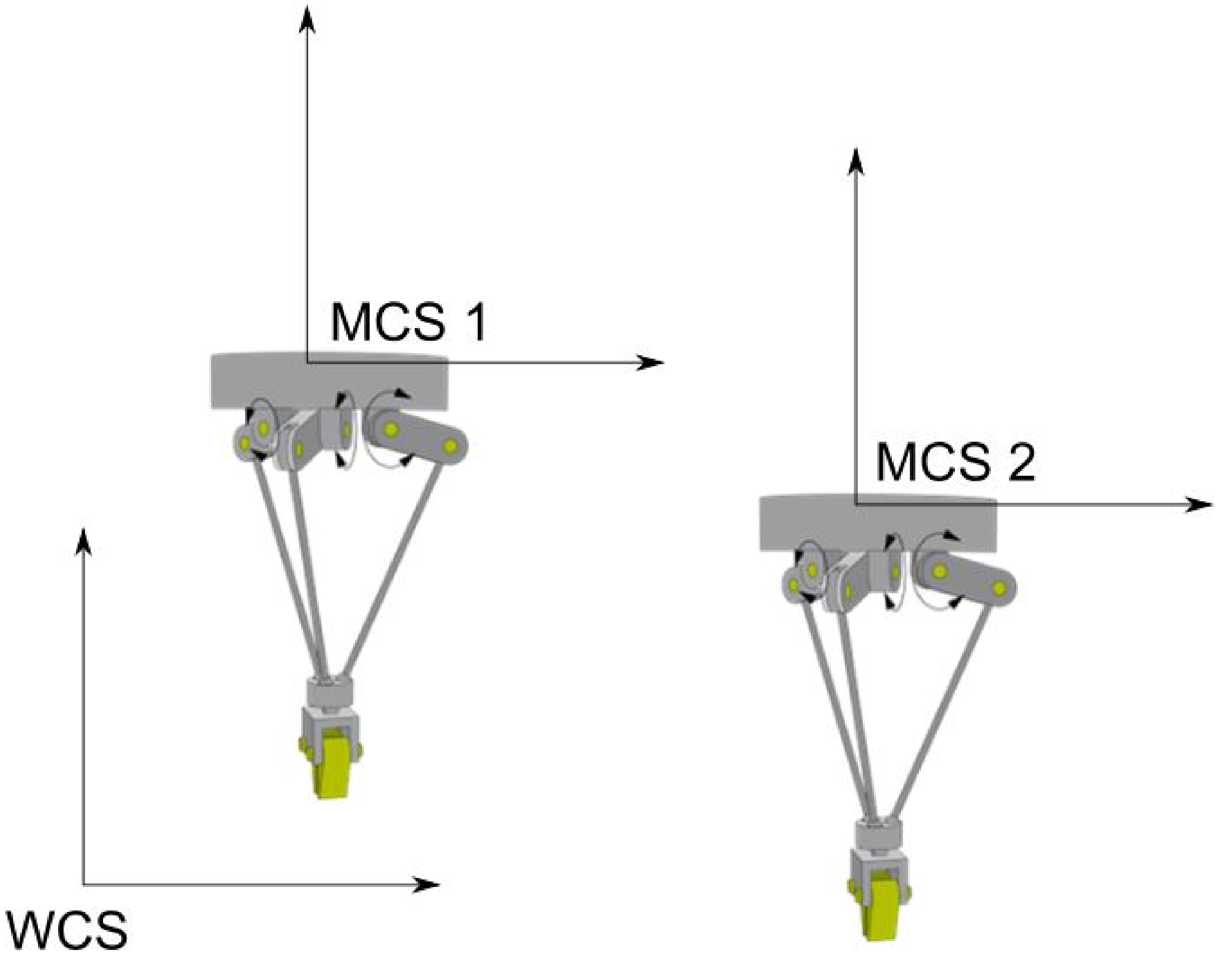


Coordinate systems

Coordinate systems are required in order to describe the positional behavior of a system. Different coordinate systems can be used as a basis for programming:

- The machine coordinate system (MCS) is a robot-based cartesian coordinate system, which usually has its origin in the robot base.

- The world coordinate system (WCS) is a cartesian coordinate system, which describes the whole modelled 'world'. It therefore does not refer to a specific robot, but to the whole system. The origin of a robot-based machine coordinate system (MCS) is at a particular point of the WCS. In other words, the user can specify, at which point of his “world” an industrial robot is located and how it is oriented. A WCS can contain several robots. When using a robot, world coordinates can coincide with the machine coordinates to improve transparency.
- The user can position the user coordinate system (UCS) at any position and with any orientation within the world coordinate system.
- The axis coordinate system (ACS) describes the position of the physical axes. It is generally not a cartesian coordinate system. Many robot joint axes perform rotary movements. Using the ACS makes it easier to take into account the limit values for angle, velocity and acceleration. If a robot axis performs rotary movements, it is often difficult for the user to predict and control the path. The axis coordinate system is usually used for referencing/homing.



Kinematic transformation

In many cases, robots are programmed in the MCS. Due to the way humans think, movements are usually programmed in Cartesian coordinate systems. To execute such movements, it is therefore necessary to convert between the axis coordinate system and the Cartesian space.

Transformation describes, in the context of the kinematics, the calculation necessary in order to change from one coordinate system to another. There are basically two problems in considering the kinematics of robots:

- The conversion from the axis coordinate system (ACS) to a Cartesian coordinate system is referred to as forward transformation. The Cartesian position of the tool center point (TCP) is calculated from the axis-specific joint coordinates of the robot.
- The conversion from Cartesian coordinates of the TCP to axis coordinates, which is required in order to move the actual robot axes, is referred to as backward transformation.

Realization in TwinCAT

TwinCAT Kinematic Transformation can be used to realize robotics applications. All PLC and NC features can be combined on a common hardware and software platform. TwinCAT Kinematic Transformation realizes several robot kinematics (e.g. H-Bot, delta robot, 6-axis robot) on the PC. The axes are controlled directly from the TwinCAT Motion Control system.

The user can program robot movements directly in the Cartesian coordinate system. The software calculates the transformation to the axis coordinate system of the robot in each cycle. To minimize vibrations and to increase the positioning accuracy, for many kinematics a current pre-control can be activated, if the drive amplifier and the fieldbus are fast enough and interfaces for an additional current pre-control are available. EtherCAT and the Beckhoff servo drives of type AX5000 meet these requirements.

The TwinCAT function seamlessly integrates in the motion control world of Beckhoff. TwinCAT NC I enables programming both via G-Code (DIN 66025) and directly from the PLC (PlcInterpolation library). The functions TF5055 TwinCAT 3 NC Flying Saw and TF5050 TwinCAT 3 NC Camming enable synchronization with conveyor belts for picking and placing of workpieces, for example. In addition, standard PTP functions from the familiar Beckhoff PTP motion libraries can be used.

The configuration of the robot takes place entirely in the TwinCAT 3 Engineering environment (XAE).

3 Overview of the new functions

From V3.1.10.66:

- New kinematics:
  - 3D-Tripod Type 1 (P_3Z)
  - 3D-Tripod Type 2 (P_3L)

From V3.1.10.30:

- New kinematics:
  - 3D-Kinematics Type 7 (PXX_SZ)
  - 3D-Delta T-Type 3 (P_3C3)
  - 3D-Cable Kinematic Type 2 (P_3L)
  - 4D-Kinematics Type 6 (S_XCZC)
  - 4D-Cable Kinematics (P_4L)

From V3.1.10.1:

- New function blocks for Extended Rotation Range are implemented:
  - FB_KinPresetRotation
  - FB_KinExtendedRotationRange
- New function F_KinAxesInTolerance
- Requires TwinCAT V3.1.4024.7 or higher

From V3.1.6.3:

- The TF511x TwinCAT 3 Kinematic Transformation become a part of the TF5400 installation package.

4 Installation

The TF5400 TwinCAT 3 Advanced Motion Pack installation package contains the components required for the TwinCAT functions TF5110 - TF5113 TwinCAT 3 Kinematic Transformation, TF5410 TwinCAT 3 Motion Collision Avoidance, TF5420 TwinCAT 3 Motion Pick-and-Place and TF5430 TwinCAT 3 Planar Motion.

Take care to install the Advanced Motion package as well on the engineering system as on the target system.

Installation requirements

The installation of TF5400 TwinCAT 3 Advanced Motion Pack requires TwinCAT 3.

Target system

Windows 7, Windows 10

Function level for TF5110-TF5113

The function TF5110 - TF5113 TwinCAT 3 Kinematic Transformation is subdivided into four different levels, depending on the number of transformation axes. A higher level includes all sublevels.

A separate installation file is required for Level 4, which you can obtain from Support on request.

Level 1: Supports the static transformation. This includes a translation and rotation of the coordinate system.

Level 2: Supports Level 1 and simple (mainly 2D) kinematic transformations such as H-Bot and 2D parallel kinematics.

Level 3: Supports Level 2 and more complex (3D, 4D) kinematic transformations such as delta robots.

Level 4: Supports Level 3 and complex kinematic transformations (up to 6D).

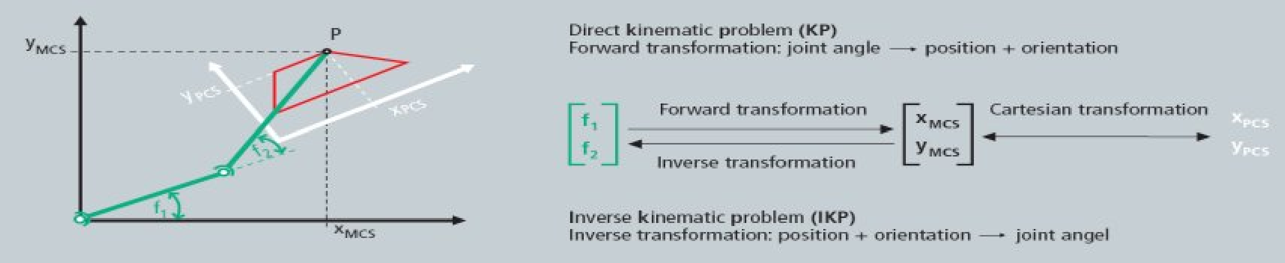
Additional licensing requirements

TF5110 - TF5113 TwinCAT 3 Kinematic Transformation requires the TC1260 license.

5 Configuration

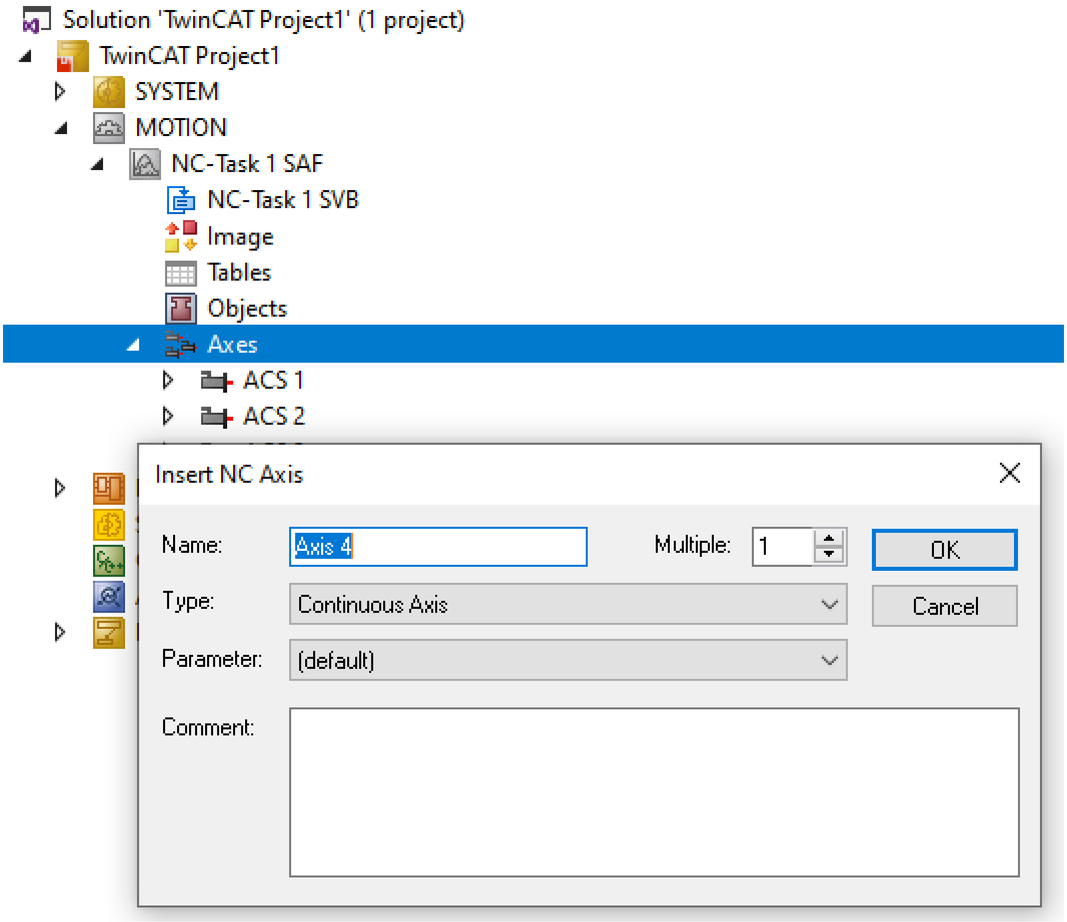
Based on PLCopen, we distinguish between two main coordinate systems (Introduction [▶ 8]):

- Axis coordinate system (ACS)
- Machine coordinate system (MCS)


yMCS
P
yPCS
xPCS
f1
f2
xMCS
Direct kinematic problem (KP)
Forward transformation: joint angle → position + orientation
f1
f2
Forward transformation
Inverse transformation
xMCS
yMCS
Cartesian transformation
xPCS
yPCS
Inverse kinematic problem (IKP)
Inverse transformation: position + orientation → joint angel

Configuring the kinematic transformation channel

1. Add all axes (ACS and MCS) to the NC configuration in the XAE, just like PTP axes. The axes of the ACS are hardware axes and are linked with drives. The axes of the MCS are pure software axes of the simulation encoder type. All ACS and MCS axes that are used in a kinematic transformation channel must be created in the XAE. A delta robot [▶ 30], for example, has three ACS axes (M1...M3) and three MCS axes (X, Y, Z).
2. Right-click on “Axes” and select “Add new item”.
3. Then create the axes in the “Insert NC Axis” window, according to the kinematics.



Name: Axis 4

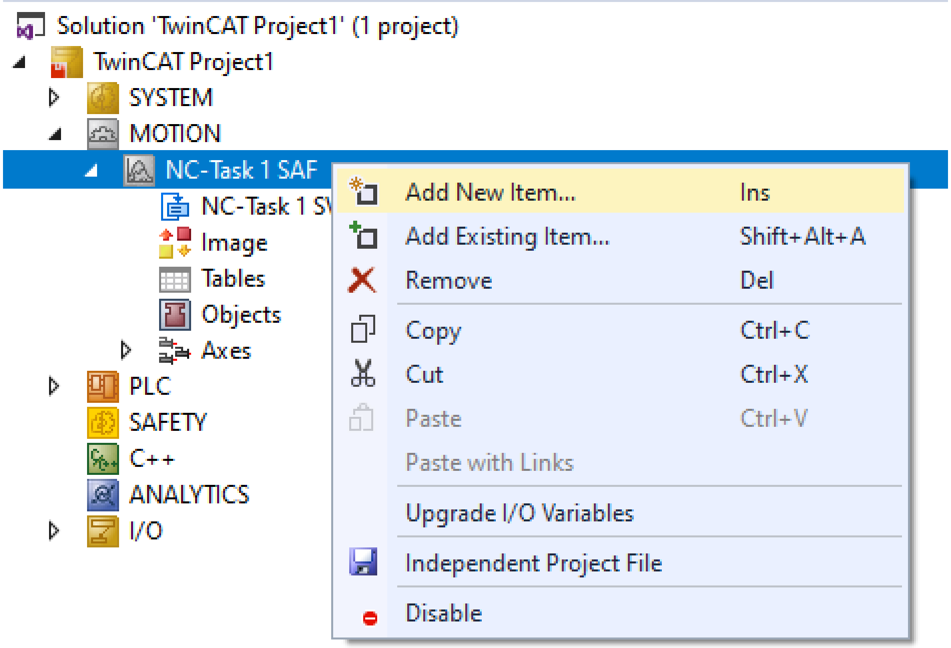
Multiple: 1

Type: Continuous Axis

Parameter: (default)

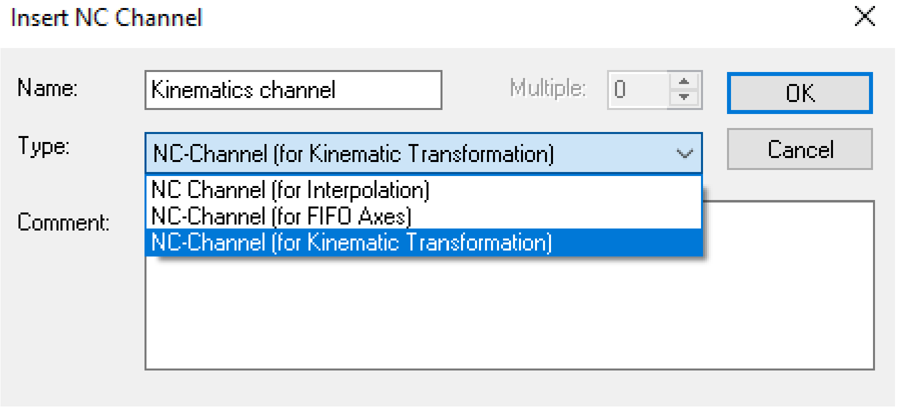
Comment:

4. Add a kinematic channel to the XAE configuration.



⇨ The addition of a channel creates an instance of a kinematic group.

5. Select the channel type: **NC channel (for kinematic transformation)** for performing a kinematic transformation.


Insert NC Channel

Name: Kinematics channel

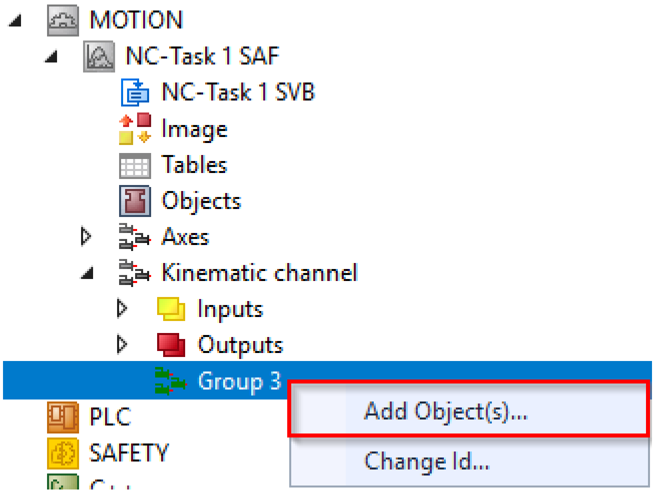
Multiple: 0

Type: NC-Channel (for Kinematic Transformation)

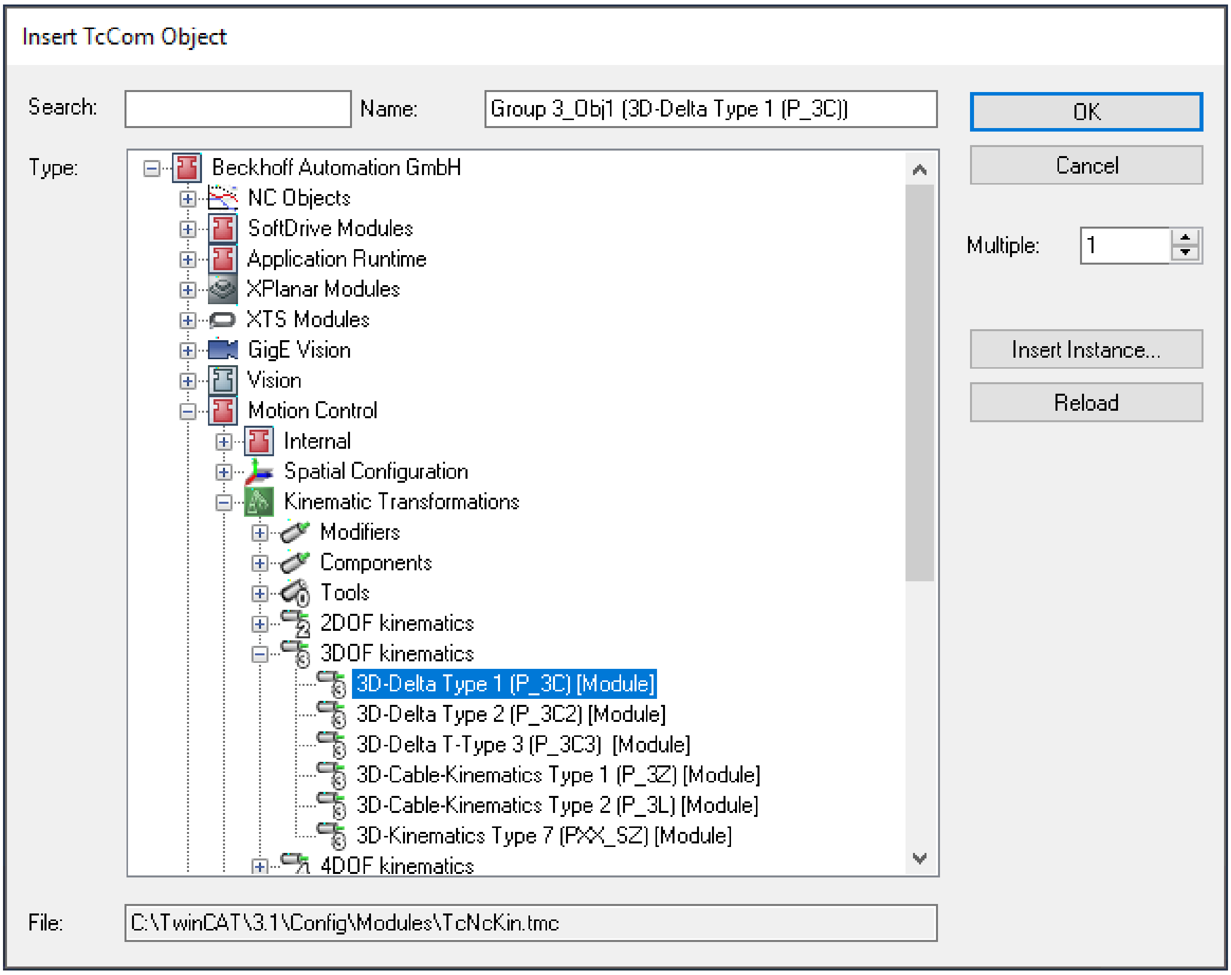
Comment:

NC Channel (for Interpolation)
NC-Channel (for FIFO Axes)
NC-Channel (for Kinematic Transformation)
OK
Cancel

6. Add the objects under the group representing the kinematic configuration of the user.



7. To start the transformation for a delta robot, select
   - Delta Type 1
   In addition, optional tools [▶ 51] and coordinate systems [▶ 53] (UCS) can be created.



Insert TcCom Object

Search:

Name: Group 3_Obj1 (3D-Delta Type 1 (P_3C))

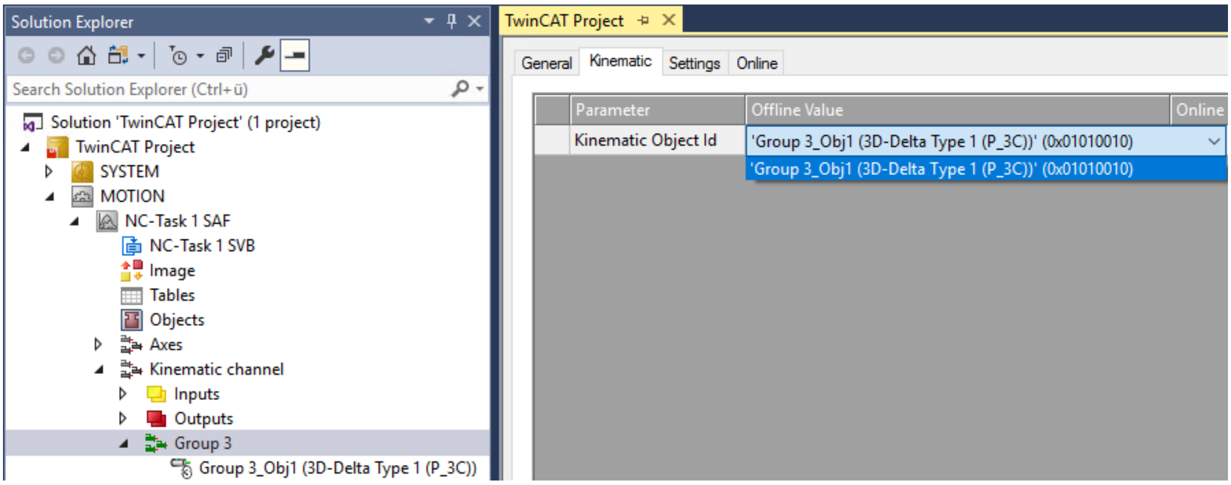
Type:

- Beckhoff Automation GmbH
  - NC Objects
  - SoftDrive Modules
  - Application Runtime
  - XPlanar Modules
  - XTS Modules
  - GigE Vision
  - Vision
  - Motion Control
    - Internal
    - Spatial Configuration
    - Kinematic Transformations
      - Modifiers
      - Components
      - Tools
      - 2DOF kinematics
      - 3DOF kinematics
        - 3D-Delta Type 1 (P_3C) [Module]
        - 3D-Delta Type 2 (P_3C2) [Module]
        - 3D-Delta T-Type 3 (P_3C3) [Module]
        - 3D-Cable-Kinematics Type 1 (P_3Z) [Module]
        - 3D-Cable-Kinematics Type 2 (P_3L) [Module]
        - 3D-Kinematics Type 7 (PXX_SZ) [Module]
      - 4DOF kinematics

Multiple: 1

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

8. The transformation group must know which root module is to be called. This is why the object ID of the kinematics (in this case Delta Type 1) must be selected. The kinematic object defines the number of ACS and MCS axes to be used in the PLC (see ST_KinAxes [▶ 81]).



Solution Explorer

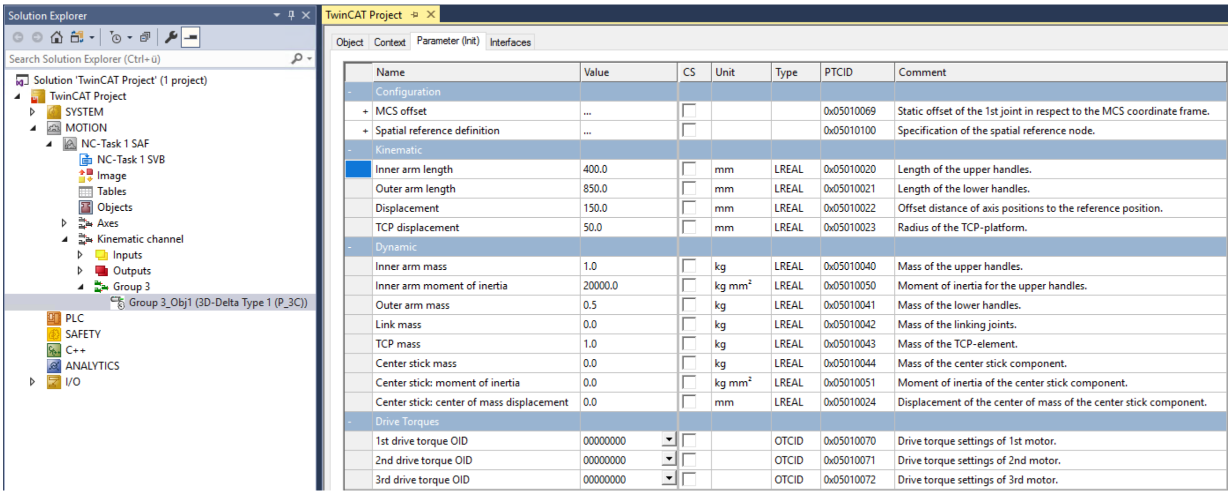
Search Solution Explorer (Ctrl+ü)

- Solution 'TwinCAT Project' (1 project)
  - TwinCAT Project
    - SYSTEM
    - MOTION
      - NC-Task 1 SAF
        - NC-Task 1 SVB
        - Image
        - Tables
        - Objects
        - Axes
        - Kinematic channel
          - Inputs
          - Outputs
          - Group 3
            - Group 3_Obj1 (3D-Delta Type 1 (P_3C))

TwinCAT Project

Parameter	Offline Value	Online
Kinematic Object Id	'Group 3_Obj1 (3D-Delta Type 1 (P_3C))' (0x01010010)	
	'Group 3_Obj1 (3D-Delta Type 1 (P_3C))' (0x01010010)	

9. Parameterize the object parameters according to the kinematics used.



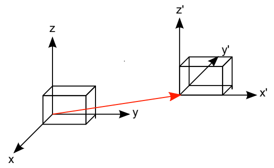
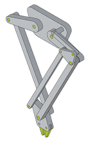
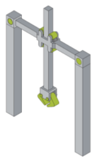
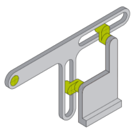
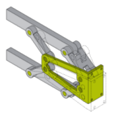
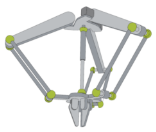
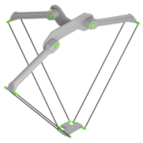
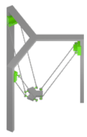
Name	Value	CS	Unit	Type	PTCID	Comment
- Configuration						
+ MCS offset	...	□			0x05010069	Static offset of the 1st joint in respect to the MCS coordinate frame.
+ Spatial reference definition	...	□			0x05010100	Specification of the spatial reference node.
- Kinematic						
Inner arm length	400.0	□	mm	LREAL	0x05010020	Length of the upper handles.
Outer arm length	850.0	□	mm	LREAL	0x05010021	Length of the lower handles.
Displacement	150.0	□	mm	LREAL	0x05010022	Offset distance of axis positions to the reference position.
TCP displacement	50.0	□	mm	LREAL	0x05010023	Radius of the TCP-platform.
- Dynamic						
Inner arm mass	1.0	□	kg	LREAL	0x05010040	Mass of the upper handles.
Inner arm moment of inertia	20000.0	□	kg mm²	LREAL	0x05010050	Moment of inertia for the upper handles.
Outer arm mass	0.5	□	kg	LREAL	0x05010041	Mass of the lower handles.
Link mass	0.0	□	kg	LREAL	0x05010042	Mass of the linking joints.
TCP mass	1.0	□	kg	LREAL	0x05010043	Mass of the TCP-element.
Center stick mass	0.0	□	kg	LREAL	0x05010044	Mass of the center stick component.
Center stick: moment of inertia	0.0	□	kg mm²	LREAL	0x05010051	Moment of inertia of the center stick component.
Center stick: center of mass displacement	0.0	□	mm	LREAL	0x05010024	Displacement of the center of mass of the center stick component.
- Drive Torques						
1st drive torque OID	00000000	□		OTCID	0x05010070	Drive torque settings of 1st motor.
2nd drive torque OID	00000000	□		OTCID	0x05010071	Drive torque settings of 2nd motor.
3rd drive torque OID	00000000	□		OTCID	0x05010072	Drive torque settings of 3rd motor.

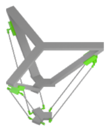
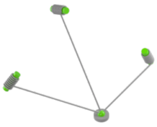
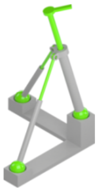
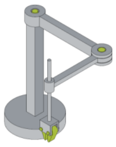
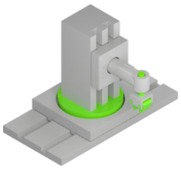
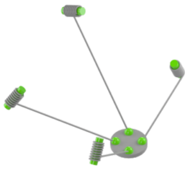
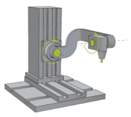
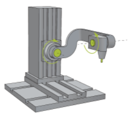
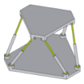
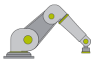
10. The transformation can now be activated via the PLC (see Plc Library [▶ 65]). To actuate the transformation, define a cyclic channel interface in the PLC and link it with the I/O of the kinematic channel.

```
in_stKinToPlc      AT %I*      : NCTOPLC_NCICHANNEL_REF;
out_stPlcToKin     AT %Q*      : PLCTONC_NCICHANNEL_REF;
```

6 Supported Transformation

Overview

Transformation type	Schema	Required TwinCAT function (level)
Static Transformation [▸ 21]		TF5110 TwinCAT 3 Kinematic Transformation (Level 1)
2D-Kinematics Type 1 (P_2C) [▸ 22]		TF5111 TwinCAT 3 Kinematic Transformation (Level 2)
2D-Kinematics Type 2 (P_2C2) [▸ 24]		TF5111 TwinCAT 3 Kinematic Transformation (Level 2)
2D-Kinematics Type 3 (S_CC) [▸ 25]		TF5111 TwinCAT 3 Kinematic Transformation (Level 2)
2D-Kinematics H-Bot (P_2Y) [▸ 26]		TF5111 TwinCAT 3 Kinematic Transformation (Level 2)
2D-Kinematics Type 5 (S_CC) [▸ 27]		TF5111 TwinCAT 3 Kinematic Transformation (Level 2)
2D-Kinematics Type 6 (P_2X) [▸ 29]		TF5111 TwinCAT 3 Kinematic Transformation (Level 2)
3D-Delta Type 1 (P_3C) [▸ 30]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
Delta Type 2 (P_3C2)		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
3D-Delta T Type 3 (P_3C3) [▸ 32]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
3D-Tripod Type 1 (P_3Z) [▸ 34]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

Transformation type	Schema	Required TwinCAT function (level)
3D-Tripod Type 2 (P_3L) [▶ 35]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
3D-Cable Type 1 (P_3Z)		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
3D-Cable Type 2 (P_3L) [▶ 36]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
3D-Kinematics Type 7 (PXX_SZ) [▶ 38]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
4D-SCARA (S_CCZC) [▶ 40]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
4D-Kinematics Type 6 (S_XCZC) [▶ 41]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
4D-Cable (4D_P_4L) [▶ 42]		TF5112 TwinCAT 3 Kinematic Transformation (Level 3)
5D-Kinematics Type 2 (XYZab) [▶ 43]		TF5113 TwinCAT 3 Kinematic Transformation (Level 4)
5D-Kinematics Type 3 (XYZAB) [▶ 45]		TF5113 TwinCAT 3 Kinematic Transformation (Level 4)
6D-Stewart Platform (P_6L) [▶ 47]		TF5113 TwinCAT 3 Kinematic Transformation (Level 4)
Six Axis Articulated (S_CBBCBC) [▶ 49]		TF5113 TwinCAT 3 Kinematic Transformation (Level 4)

Additional objects

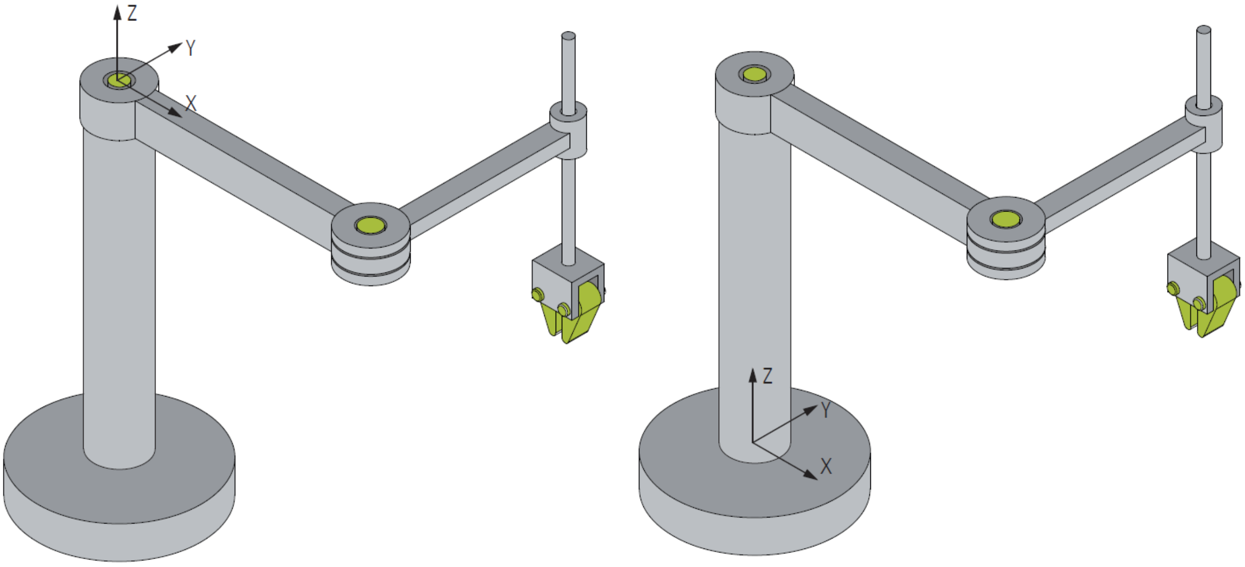
The following objects can be created and selected in the corresponding kinematics. A dropdown parameter list in the kinematics is used for the selection. Select the corresponding object ID (OTCID).

object type	Description	Required level and version
Tool Offset [▶ 51]	Tool offset - describes a tool at the level of the kinematics flange.	TF5110 TwinCAT 3 Kinematic Transformation (Level 1)
Tool Linear [▶ 52]	Linear tool - describes a 1D tool mounted on the kinematics flange, which offers the option to move the TCP towards the tool.	TF5110 TwinCAT 3 Kinematic Transformation (Level 1)
Drive Torque [▶ 50]	Drive torque - represents the inertia and the efficiency of the motor and drive, to enable more precise calculation of the dynamic model.	TF5110 TwinCAT 3 Kinematic Transformation (Level 1)
Coordinate system (Coordinate Frame) [▶ 53]	Coordinate system - describes a user-defined coordinate system.	TF5110 TwinCAT 3 Kinematic Transformation (Level 1)

6.1 General Parameters for the Kinematics

MCS Offset

The MCS offset can be used to parameterize additional offset parameters before the first axis (or before the basis) of the kinematics. For example, in the SCARA kinematics the origin of the MCS is located in the first joint (M1). The parameter Z-shift of the MCS offset can be used to parameterize the additional bar length so that the origin of the MCS resides at the robot base.



Parameter	Description	Type	Unit
X-shift	Static X-offset in the MCS.	LREAL	mm
Y-shift	Static Y-offset in the MCS.	LREAL	mm
Z-shift	Static Z-offset in the MCS.	LREAL	mm

MCS to Spatial reference

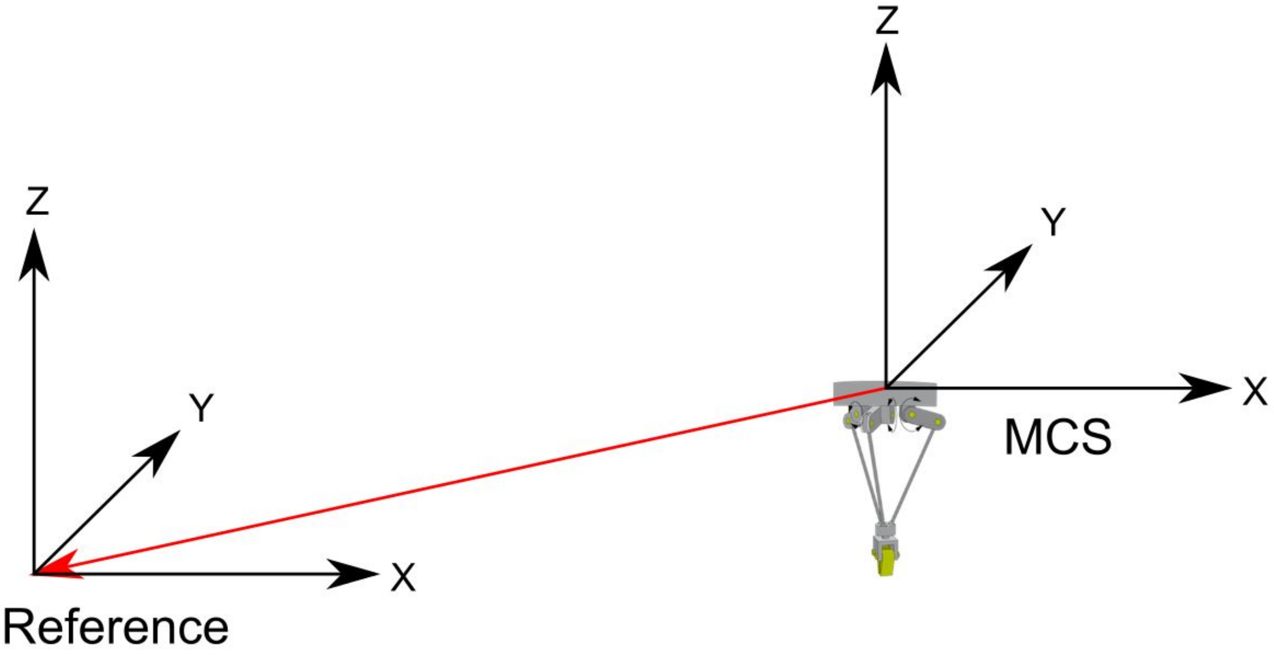
The parameter MCS to Spatial reference can be used to move the MCS in a reference coordinate system. All coordinate systems are right-handed (counterclockwise).

Parameter	Description	Type	Unit
Translation X	Shift in X-direction.	LREAL	mm
Translation Y	Shift in Y-direction.	LREAL	mm
Translation Z	Shift in Z-direction.	LREAL	mm

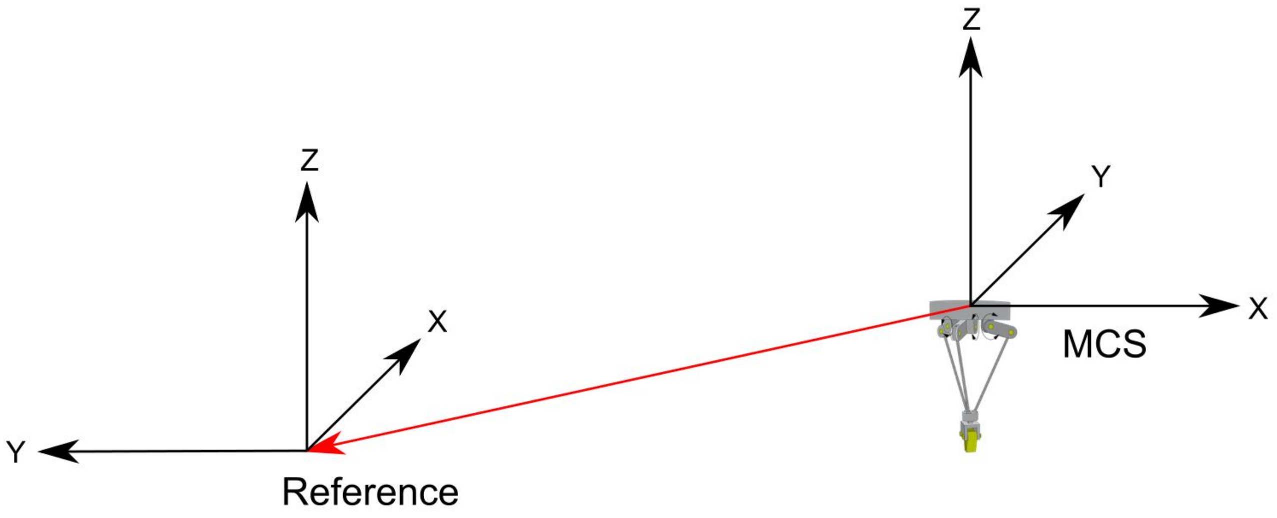
Parameter	Description	Type	Unit
Rotation 1	First rotation angle. The interpretation is defined by the parameter Rotation convention.	LREAL	°
Rotation 2	Second rotation angle. The interpretation is defined by the parameter Rotation convention.	LREAL	°
Rotation 3	Third rotation angle. The interpretation is defined by the parameter Rotation convention.	LREAL	°
Rotation convention	The rotation convention indicates the order of the axis rotations (parameter Rotation 1-3). Doing so, the letters (X, Y, Z) from left to right indicate the order of rotation around the corresponding axes. The succeeding number indicates the parameter (Rotation 1-3) for the value parameterization. The translational shift is always performed before the rotation.	MC.CoordInterpretation_SO3	
Spatial reference	The parameter Spatial reference indicates which coordinate system is used as a basis for the MCS. If the value has been set to 0 here, the WCS is used as a basis. To use another coordinate system as starting point for the shift, a Coordinate system (Coordinate Frame) [▶ 53] object can be created. The object ID of this coordinate system can be selected via the drop-down menu.	OTCID	
Definition direction	Indicates the direction in which the shift is programmed (from the perspective of the reference system or from the perspective of the MCS). The translation is always performed first, followed by the rotation. See example below.	MC.ReferenceDefDir	

Example: Definition Direction

If the definition direction MCS -> Reference is used, the shift displayed below between the original coordinate system (MCS) towards the target coordinate system (reference) is specified with negative vectors.



If a positive rotation around the Z-axis (here 90°) is specified in addition to the translation, the translation is performed first, followed by the rotation of the target coordinate system (here +90° around the Z-axis).



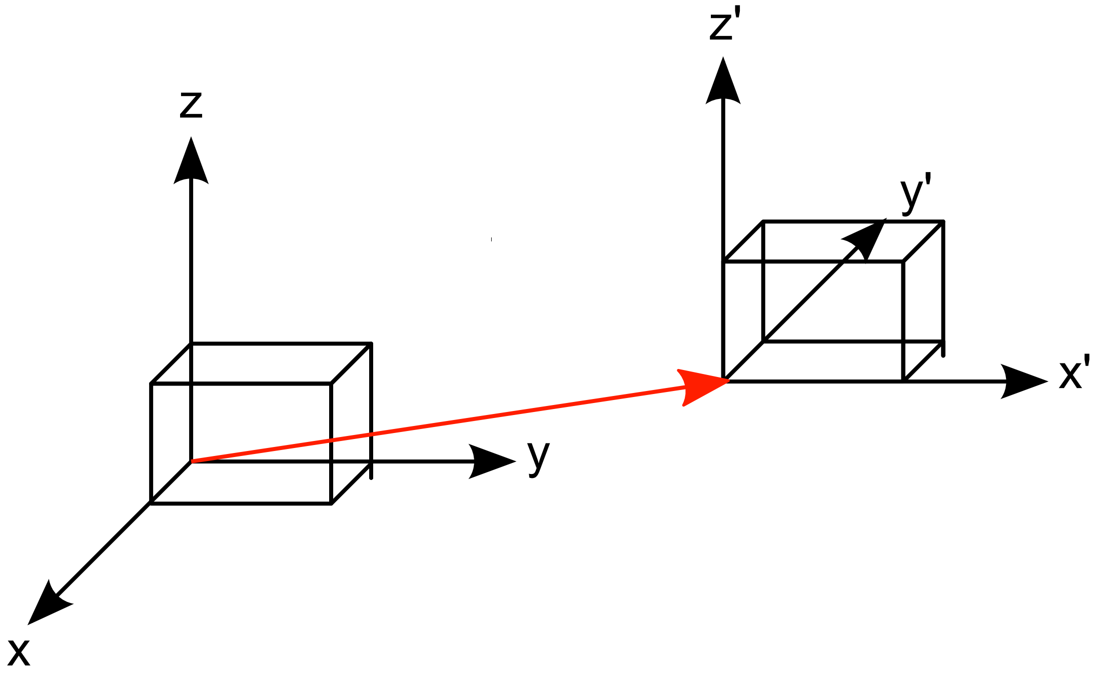
Tool offset OID

Parameter	Description	Type
Tool offset OID	To define a tool for the kinematics a Tool Offset [▶ 51] object or a Tool Linear [▶ 52] object has to be created at first. The object ID of this tool can be selected via the drop-down menu.	OTCID

6.2 Static Transformation

The static transformation enables creation of a cartesian gantry. It supports translation and rotation between two cartesian coordinate systems.

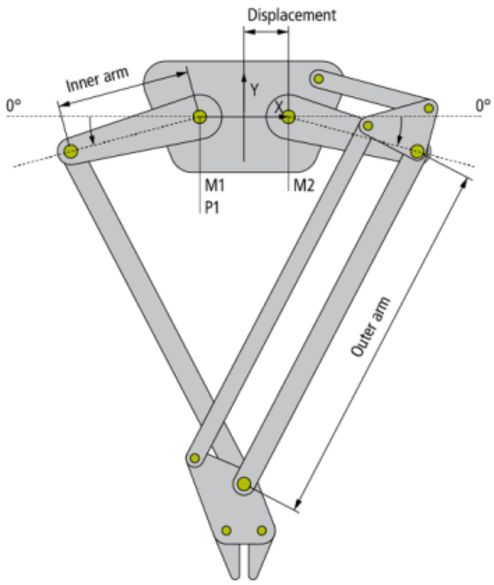
The static transformation forms the kinematic, so that the kinematic group is formed with the static transformation. In contrast, a Coordinate system (Coordinate Frame) [▶ 53] adds a translation and a rotation to an existing kinematic.



First the translation is calculated, then the rotation. The order of the rotations affects the orientation of the coordinate system. The roll/pitch/yaw rule described in DIN 9300 is used as default for the rotation sequence. The calculation sequence for the forward transformation is Z, Y', X''.

Parameter	Description	Type	Unit
Translation X	Shift in X-direction	LREAL	mm
Translation Y	Shift in Y-direction	LREAL	mm
Translation Z	Shift in Z-direction	LREAL	mm
Rotation 1	First rotation angle. The interpretation is defined by the parameter Rotation convention.	LREAL	°
Rotation 2	Second rotation angle. The interpretation is defined by the parameter Rotation convention.	LREAL	°
Rotation 3	Third rotation angle. The interpretation is defined by the parameter Rotation convention.	LREAL	°
Rotation convention	The rotation convention indicates the order of the axis rotations (parameter Rotation 1-3). The letters (X, Y, Z) from left to right indicate the order of the rotation around the corresponding axes. The number indicates the parameter (Rotation 1-3) for the value parameterization. The translatory shift is always performed before the rotation.	MC.CoordInterpretation_SO3	
Spatial reference	The parameter Spatial reference indicates which coordinate system is used as basis for this coordinate system. If the value is set to 0, the WCS is used as basis. To use another coordinate system as starting point for the shift, a further Coordinate system (Coordinate Frame) [▶ 53] object can be created. The object ID of this coordinate system can be selected via the dropdown menu.	OTCID	
Definition direction	Indicates the direction in which the shift is programmed (from the perspective of the reference system or this coordinate system).	MC.ReferenceDefDir	

6.3 2D-Kinematics Type 1 (P_2C)



The 2D-Kinematics Type 1 (P_2C) is configured as shown in the diagram above.

All motor axes are scaled in degrees; 0° is defined as shown in the diagram. The arrow indicates the positive direction.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Inner arm length	Length between pivots of the inner arm	LREAL	mm
Outer arm length	Length between pivots of the outer arm	LREAL	mm
Displacement	Length between the center of the base plate and the virtual rotation axes of the inner arm	LREAL	mm

Parameters for the Dynamic Model

Parameter	Description	Type	Unit
Inner arm mass	Total mass of the inner arm	LREAL	kg
Inner arm moment of inertia	Moment of inertia of the inner arm in relation to pivot point P1, which is linked with the motor	LREAL	kg mm^2
Outer arm mass	The mass of the external arms minus the mass of the joint can optionally be described as a separate parameter.	LREAL	kg
First link mass	Mass of the joint linking the inner and outer arm; can be used if the mass of the joint is not already included in the outer and inner arms. The mass of the joint linking the gripper plate with the outer arm is not specified here. It can be added to TcpMass. The mass of the first joint refers to the inner arm that is linked to motor 1.	LREAL	kg
Second link mass	See FirstLinkMass The mass of the second joint refers to the inner arm that is linked to motor 2.	LREAL	kg
TCP mass	Mass of the tool center point, including gripper plate and gripper. The payload is usually described with a separate parameter.	LREAL	kg
First drive torque OID	Object ID of the first drive torque (see here [▸ 50]) If the motors and gear units of all motors behave in a similar way, all drive torques can be represented via an OID. Both parameters therefore refer to the same object ID.	OTCID	
Second drive torque OID	Object ID of the second drive torque	OTCID	

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

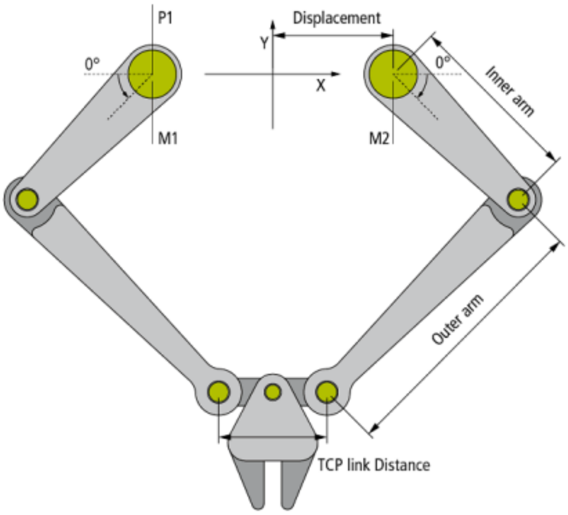
For all kinematics with tool also applies:

- Tool Offset OID [▸ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5111 TwinCAT 3 Kinematic Transformation (Level 2)

6.4 2D-Kinematics Type 2 (P_2C2)



The 2D-Kinematics Type 2 (P_2C2) is configured as shown in the diagram above.

All motor axes are scaled in degrees; 0° is defined as shown in the diagram. The arrow indicates the positive direction.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Inner arm length	Length between pivots of the inner arm	LREAL	mm
Outer arm length	Length between pivots of the outer arm	LREAL	mm
Displacement	Length between the center of the base plate and the virtual rotation axes of the inner arm	LREAL	mm
TCP link distance	Distance between pivots of the outer arm	LREAL	mm

Parameters for the Dynamic Model

Parameter	Description	Type	Unit
Inner arm mass	Total mass of the inner arm	LREAL	kg
Inner arm moment of inertia	Moment of inertia of the inner arm in relation to pivot point P1, which is linked with the motor.	LREAL	kg mm^2
Outer arm mass	The mass of the external arms minus the mass of the joint can optionally be described as a separate parameter.	LREAL	kg
First link mass	Mass of the joint linking the inner and outer arm. Can be used if the mass of the joint is not already included in the outer and inner arms. The mass of the joint linking the gripper plate with the outer arm is not specified here. It can be added to the TCP mass. The mass of the first joint refers to the inner arm that is linked to M1.	LREAL	kg
Second link mass	see First link mass The mass of the second joint refers to the inner arm that is linked to M2.	LREAL	kg
TCP mass	Mass of the TCP, including gripper plate and gripper. The payload is usually described with a separate parameter.	LREAL	kg

Parameter	Description	Type	Unit
First drive torque OID	Object ID of the first drive torque (see here [▸ 50]) If the motors and gear units of all motors behave in a similar way, all drive torques can be represented via an OID. Both parameters therefore refer to the same object ID.	OTCID	
Second drive torque OID	Object ID of the second drive torque	OTCID	

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

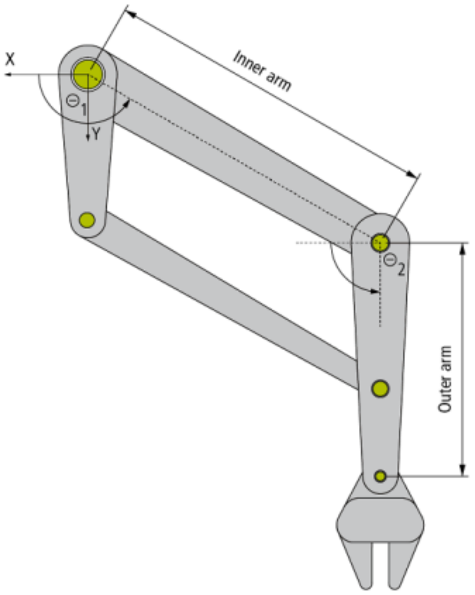
For all kinematics with tool also applies:

- Tool Offset OID [▸ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5111 TwinCAT 3 Kinematic Transformation (Level 2)

6.5 2D-Kinematics Type 3 (S_CC)



2D-Kinematics Type 3 (S_CC) is configured as shown in the diagram above.

All motor axes are scaled in degrees; 0° is defined as shown in the diagram. The arrow indicates the positive direction.

This kinematics type is implemented as a left-handed system. The shafts of motor M1 and M2 are located at the origin of the coordinate system.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Inner arm length	Length between the motor shaft and the pivot point of the external arm	LREAL	mm
Outer arm length	Length between the pivot point and the tool center point of the outer arm	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

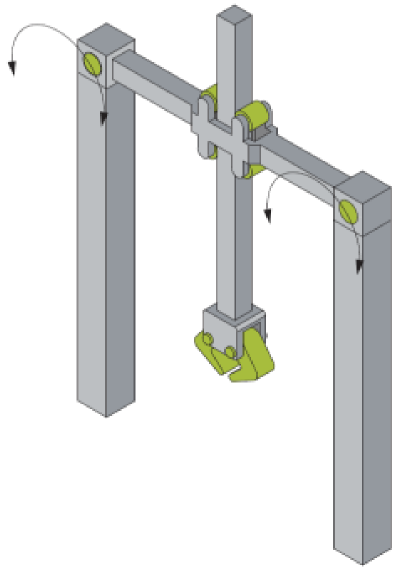
For all kinematics with tool also applies:

- Tool Offset OID [▸ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5111 TwinCAT 3 Kinematic Transformation (Level 2)

6.6 2D-Kinematics H-Bot (P_2Y)



The 2D-Kinematics H-Bot (P_2Y) is configured as shown in the diagram above.

The motor axes have to be scaled in millimeters. All the other position parameters result from the kinematic constraints.

The point of origin of the machine coordinate system MCS is defined by the point for that the positions of the two motors are zero.

Parameters for the Dynamic Model

Parameter	Description	Type
FirstDriveTorqueOID	Object ID of the first drive torque (see here [▸ 50]). If the motors and gear units of all motors behave in a similar way, all drive torques can be represented via an OID. Both parameters therefore refer to the same object ID.	OTCID
SecondDriveTorqueOID	Object ID of the second drive torque	OTCID

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

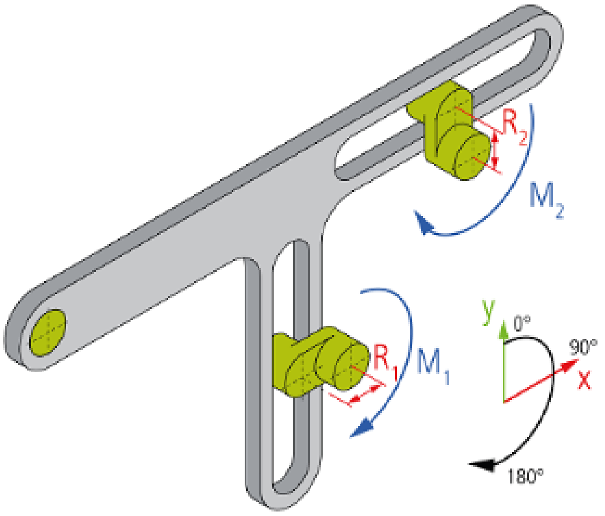
For all kinematics with tool also applies:

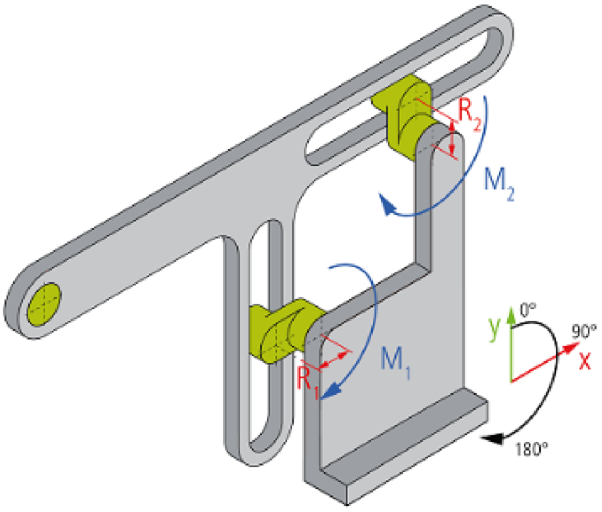
- Tool Offset OID [▸ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5111 TwinCAT 3 Kinematic Transformation (Level 2)

6.7 2D-Kinematics Type 5 (S_CC)





A crank consists of a wheel with an eccentrically located pin. Two cranks whose ends lead to bearings facilitate two dimensional movements of the TCP. The cranks are moved by motors that are obstructed in a stationary machine.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Radius R_1	• The quantity R_1 describes the lever arm of crank 1. This lever arm is measured from the crank center to the center of the pin in the bearing. • The rotational center of crank 1 is fixed. • Crank 1 is moved by motor M1. • Motor M1 evokes movements in X-direction of the TCP.	`LREAL`	mm
Radius R_2	• The quantity R_2 describes the lever arm of crank 2. This lever arm is measured from the crank center to the center of the pin in the bearing. • The rotational center of crank 2 is fixed. • Crank 2 is moved by motor M2. • Motor M2 evokes movements in Y-direction of the TCP.	`LREAL`	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

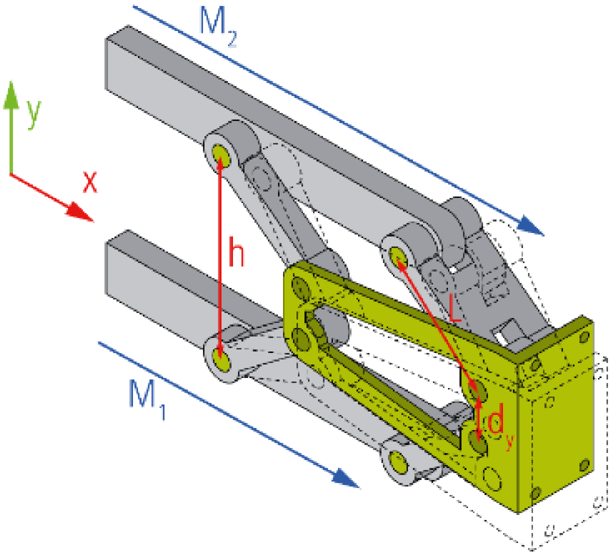
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5111 TwinCAT 3 Kinematic Transformation (Level 2)

6.8 2D-Kinematics Type 6 (P_2X)



With the Kinematics the two linear axes M1 and M2 enable movements within the XY-plane.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Flange translation X	Spatial shift in X-direction.	LREAL	mm
Arm length (L)	Length of the joint segments each measured from the motor to the joint of the flange.	LREAL	mm
Motor distance (h)	The motor distance h is the Y-distance of the joints at the motors. (The motors M1 and M2 both move in X-direction. The motor distance h is measured from joint center point to joint center point.)	LREAL	mm
Link distance (d_y)	The link distance d_y is the distance between the joints in the flange.	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

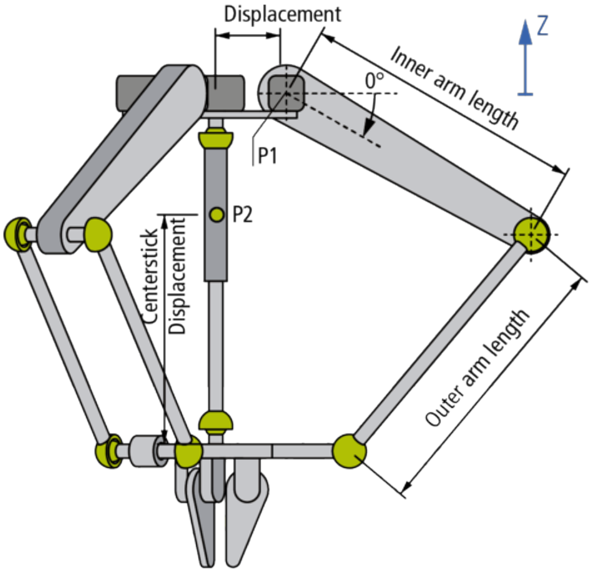
For all kinematics with tool also applies:

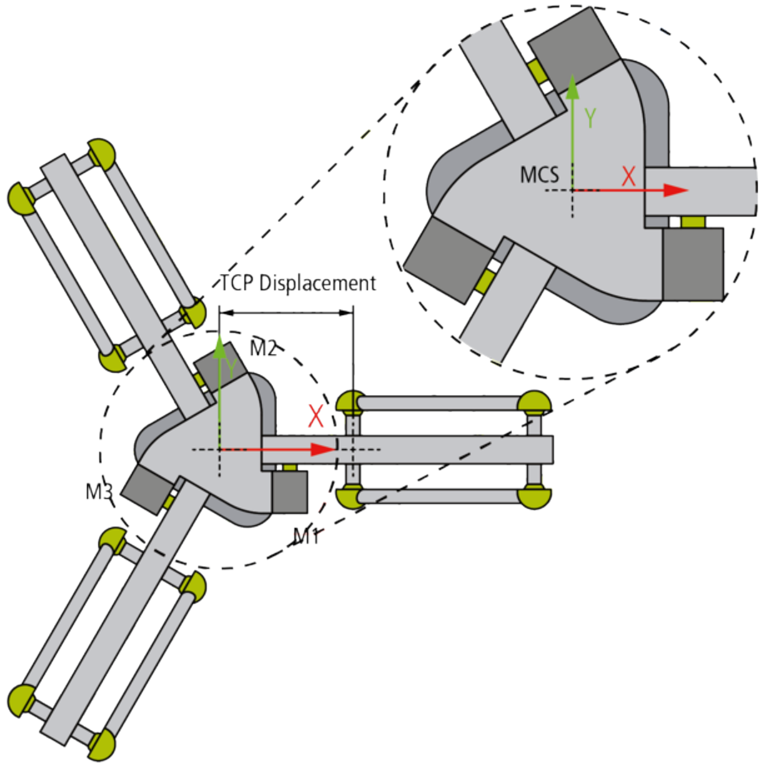
- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5111 TwinCAT 3 Kinematic Transformation (Level 2)

6.9 3D-Delta Type 1 (P_3C)





The 3D-Delta Kinematics Type 1 (P_3C) is configured as shown in the diagram above. The Kinematic Transformation expects ball joints (or elements with the same behavior) in the link between the arms and the lower plate.

Parameterization of the center stick for aligning the gripper is optional.

All motor axes are scaled in degrees; 0° is defined as shown in the diagram. The arrow indicates the positive direction. This applies for all three motors.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Inner arm length	Length between the pivot points of the inner arm; this is the arm that is directly linked with the motor.	`LREAL`	mm
Outer arm length	Length between pivots of the outer arm	`LREAL`	mm
Displacement	Length between the center of the base plate and the virtual rotation axes of the inner arm	`LREAL`	mm
TCP displacement	Length between the center of the gripper plate and the virtual rotation axes of the outer arm	`LREAL`	mm

Parameters for the Dynamic Model

Parameter	Description	Type	Unit
Inner arm mass	Total mass of the inner arm	`LREAL`	kg
Inner arm moment of inertia	Moment of inertia of the inner arm in relation to pivot point P1, which is linked with the motor	`LREAL`	kg mm^2
Outer arm mass	Mass of the outer arm. If two bars are used, the total mass is required. The mass of the joint can optionally be described as a separate parameter.	`LREAL`	kg
Link mass	Mass of the joint linking the inner and outer arm. Can be used if the mass of the joint is not already included in the outer and inner arms. The mass of the joint linking the gripper plate with the outer arm is not specified here. It can be added to TcpMass.	`LREAL`	kg
TCP mass	Mass of the TCP, including gripper plate and gripper. The payload is usually described with a separate parameter.	`LREAL`	kg
Center stick mass	Total mass of the center stick	`LREAL`	kg
Center stick: moment of inertia	Moment of inertia of the center stick in relation to the center of gravity (P2)	`LREAL`	kg mm^2
Center stick: center of mass displacement	Length between the gripper plate and the center of gravity of the bar	`LREAL`	mm
First drive torque OID	Object ID of the first drive torque (see here) If the motors and gear units of all motors behave in a similar way, all drive torques can be represented via an OID. All three parameters refer to the same object ID.	`OTCID`	
Second drive torque OID	Object ID of the second drive torque	`OTCID`	
Third drive torqueOID	Object ID of the third drive torque	`OTCID`	

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

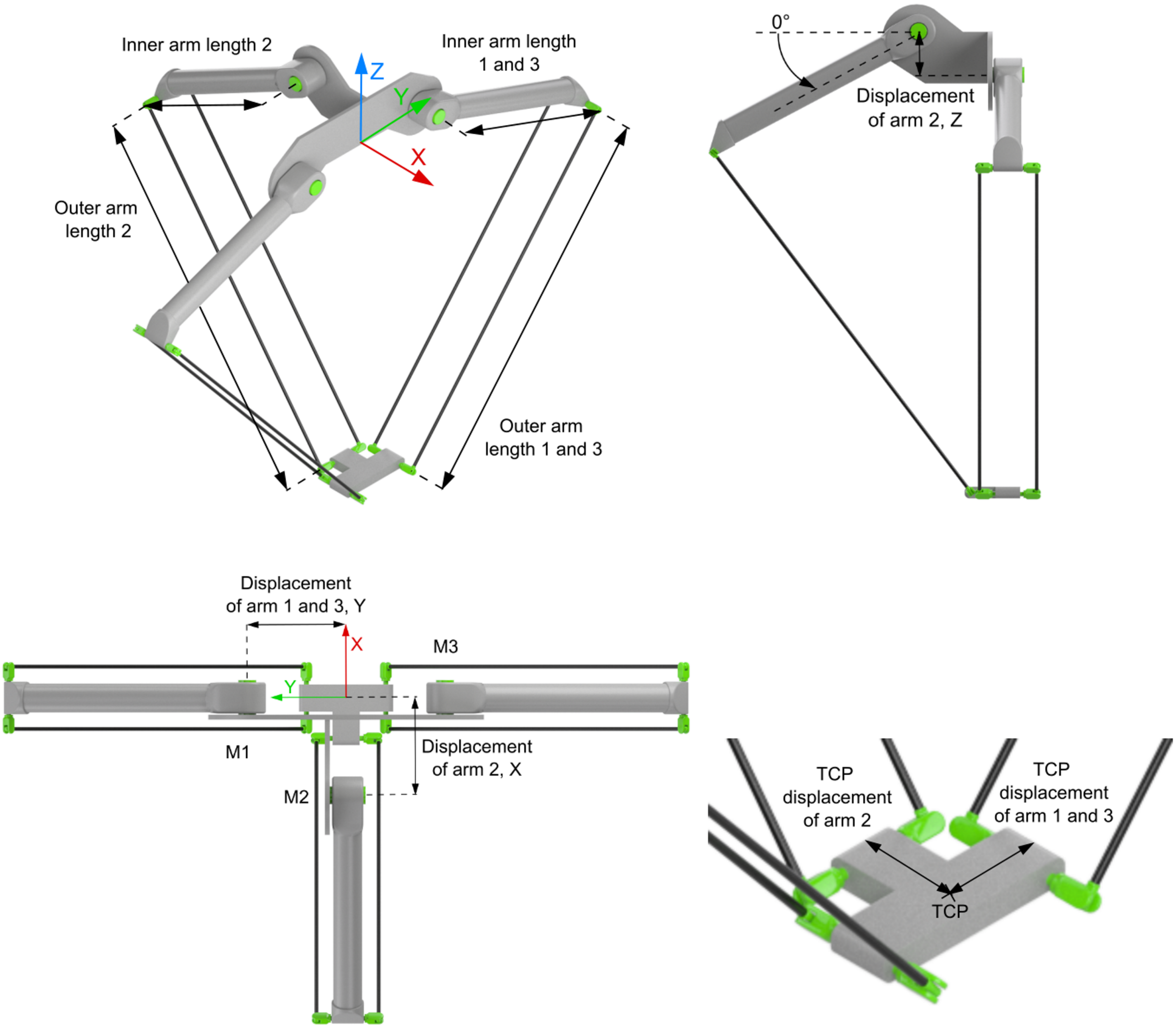
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.10 3D-Delta T Type 3 (P_3C3)



The 3D-Delta T Type 3 (P_3C3) is structured as shown above. Two arms are directly opposite each other, while the third arm is at an angle of 90 degrees to them. This arm configuration allows two robots of this type to be placed very close together.

The machine coordinate system (MCS) is located centrally between the two opposite arms at the height of motors M1 and M3.

All motor axes are scaled in degrees and 0° is defined, as shown in the schematic, with the arrow indicating the positive direction of rotation. This applies to all three motors.

Kinematics parameters

Parameter	Description	Type	Unit
Inner arm length 1 and 3	Arm 1, Arm 3: Length from center of rotation to center of rotation of the inner arm (connected directly to the motor)	LREAL	mm
Inner arm length 2	Arm 2: Length from center of rotation to center of rotation of the inner arm (connected directly to the motor)	LREAL	mm
Outer arm length 1 and 3	Arm 1, Arm 3: Length between pivots of the outer arm	LREAL	mm
Outer arm length 2	Arm 2: Length between pivots of the outer arm	LREAL	mm
Displacement of arm 1 and 3, Y	Arm 1, Arm 3: The distance between the MCS origin and each motor axis	LREAL	mm
Displacement of arm 2, X	Arm 2: The distance between the MCS origin and the motor axis	LREAL	mm
Displacement of arm 2, Z	Arm 2: The distance between the MCS origin and the motor axis	LREAL	mm
TCP displacement of arm 1 and 3	Arm 1, Arm 3: Length between the center of the gripper plate and the virtual rotation axes of the outer arm	LREAL	mm
TCP displacement of arm 2	Arm 2: Length between the center of the gripper plate and the virtual rotation axis of the outer arm	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

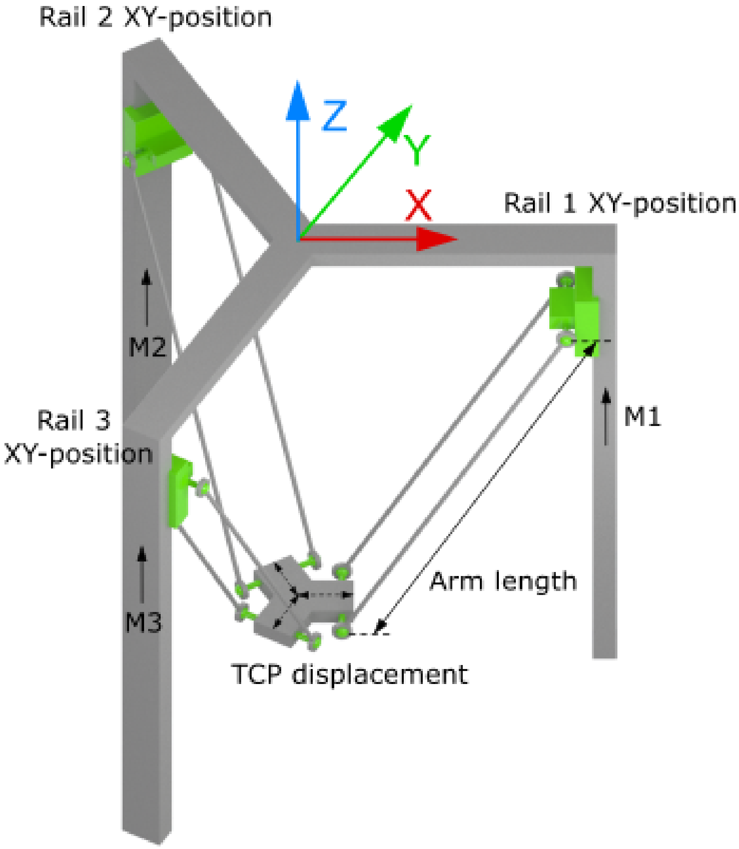
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4024.7 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.30	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.11 3D-Tripod Type 1 (P_3Z)



The 3D-Tripod Type 1 (P_3Z) is structured as shown in the figure above.

All linear axes (ACS) are scaled in mm.

Kinematics parameters

Parameter	Description	Type	Unit
Arm length	Arm length from pivot point to pivot point	LREAL	mm
Rail 1 X-position	X-position of the 1st rail in relation to the MCS	LREAL	mm
Rail 1 Y-position	Y-position of the 1st rail in relation to the MCS	LREAL	mm
Rail 2 X-position	X-position of the 2nd rail in relation to the MCS	LREAL	mm
Rail 2 Y-position	Y-position of the 2nd rail in relation to the MCS	LREAL	mm
Rail 3 X-position	X-position of the 3rd rail in relation to the MCS	LREAL	mm
Rail 3 Y-position	Y-position of the 3rd rail in relation to the MCS	LREAL	mm
TCP displacement	Length between the center of the gripper plate and the virtual rotation axis of the arm	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

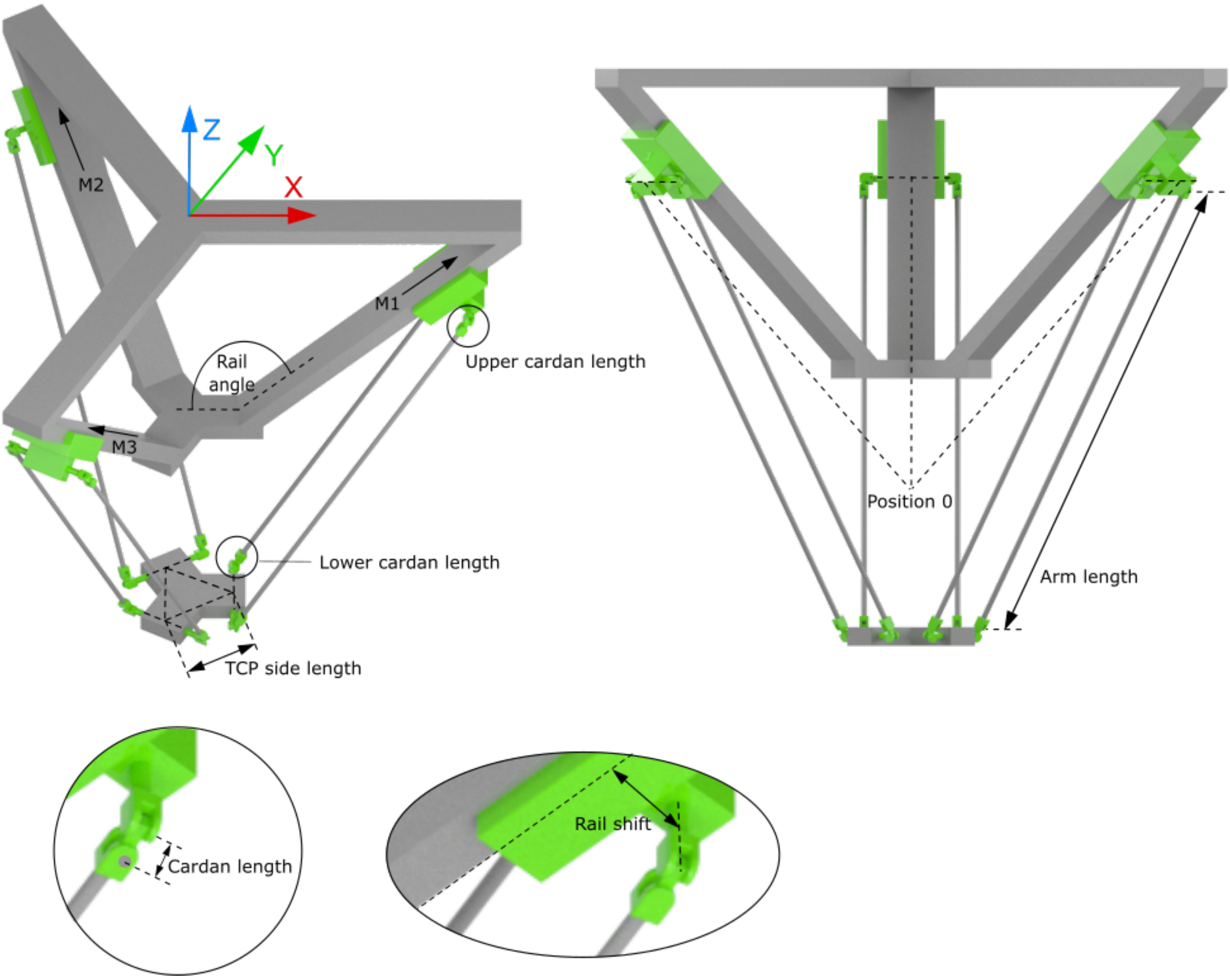
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development environment Installation package	Target platform	TwinCAT function
TwinCAT V3.1.4024.24 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.66	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.12 3D-Tripod Type 2 (P_3L)



The 3D-Tripod Type 2 (P_3L) is structured as shown in the figure above.

All linear axes (ACS) are scaled in millimeters (mm). The 0-position of the axes is only a "virtual" point, which cannot be approached. A positive velocity of the motors moves the tool upwards so that the linear axes cannot reach a negative position.

Kinematics parameters

Parameter	Description	Type	Unit
Arm length	Arm length from pivot point to pivot point	LREAL	mm
Rail angle	Angle in which the guide rails of the linear motors are mounted.	LREAL	°
Rail shift	Shift of the arm suspension points to the guide rails of the linear motors.	LREAL	mm

Parameter	Description	Type	Unit
Upper cardan length	If a cardan joint is used at the upper arm suspension points, this parameter can be used to specify the shift of the two joints within the cardan joint. When using a ball joint, enter length 0.	LREAL	mm
Lower cardan length	If a cardan joint is used at the lower arm suspension points, this parameter can be used to specify the shift of the two joints within the cardan joint. When using a ball joint, enter length 0.	LREAL	mm
TCP side length	Side length of the virtual triangle in the TCP.	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

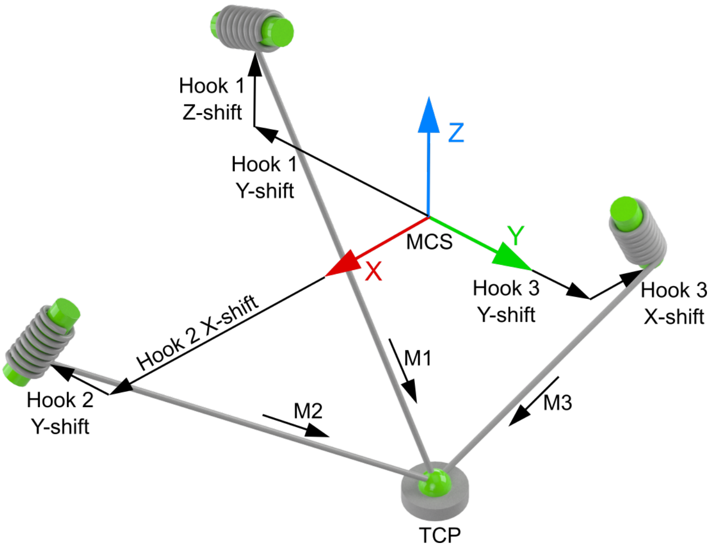
For all kinematics with tool also applies:

- Tool Offset OID [▸ 21].

Requirements

Development environment Installation package	Target platform	TwinCAT function
TwinCAT V3.1.4024.24 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.66	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.13 3D-Cable Kinematics Type 2 (P_3L)



The 3D-Cable Kinematics (P_3L) is structured as shown in the above schematic.

The zero point of the machine coordinate system (MCS) can be anywhere in space. The hooks of the cable/ rope are defined starting from the MCS origin.

All motor axes are scaled in millimeters, the arrow indicating the positive direction.

Parameters for joint hooks

Parameter	Description	Type	Unit
Hook 1	Hook of the first cable		
X-shift	X-position of Hook 1 in relation to the MCS	`LREAL`	mm
Y-shift	Y-position of Hook 1 in relation to the MCS	`LREAL`	mm
Z-shift	Z-position of Hook 1 in relation to the MCS	`LREAL`	mm
Hook 2	Hook of the second cable		
X-shift	X-position of Hook 2 in relation to the MCS	`LREAL`	mm
Y-shift	Y-position of Hook 2 in relation to the MCS	`LREAL`	mm
Z-shift	Z-position of Hook 2 in relation to the MCS	`LREAL`	mm
Hook 3	Hook of the central cable		
X-shift	X-position of Hook 3 in relation to the MCS	`LREAL`	mm
Y-shift	Y-position of Hook 3 in relation to the MCS	`LREAL`	mm
Z-shift	Z-position of Hook 3 in relation to the MCS	`LREAL`	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

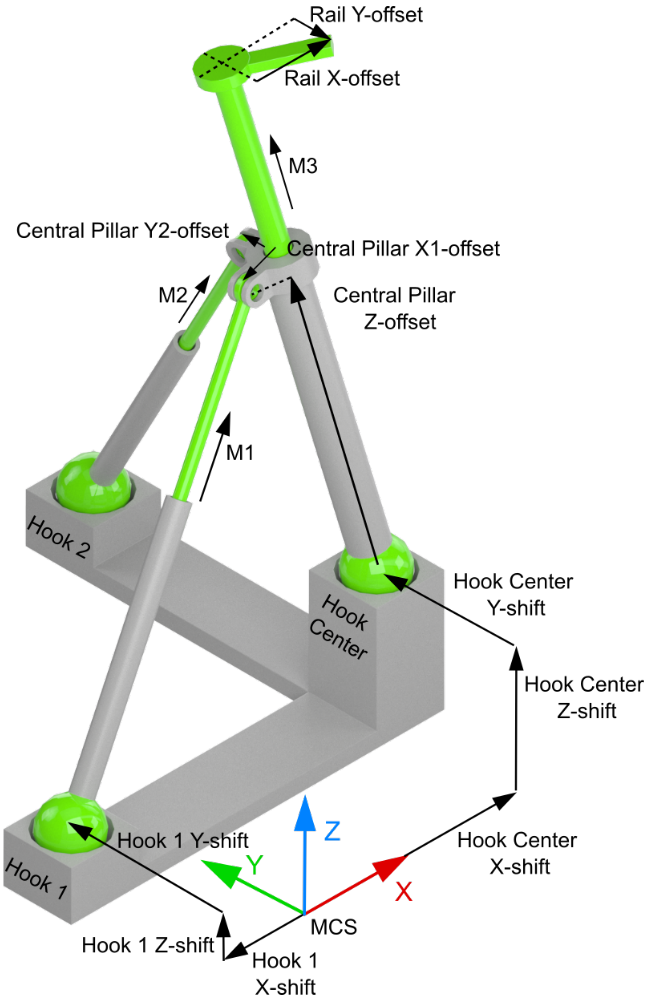
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4024.7 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.30	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.14 3D-Kinematics Type 7 (PXX SZ)



The 3D-Kinematics Type 7 (PXX SZ) is structured as shown in the schematic above.

The zero point of the machine coordinate system (MCS) can be anywhere in space. The hooks of the three arms are defined starting from the MCS origin. For the definition of the offset, the starting point for the central pillar is the position parallel to the Z-axis of the MCS, so that the fixed length from the central pillar to the suspension of the other two arms is the "central pillar Z-offset". At the end of the central pillar there may also be a rail, whose tip is specified by the "rail offset".

All motor axes are scaled in millimeters, the arrow indicating the positive direction.

Parameters Joint Hooks

Parameter	Description	Type	Unit
Hook 1	Hook of the first arm		
X-shift	X-position of Hook 1 in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook 1 in relation to the MCS	LREAL	mm
Z-shift	Z-position of Hook 1 in relation to the MCS	LREAL	mm
Hook 2	Hook of the second arm		
X-shift	X-position of Hook 2 in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook 2 in relation to the MCS	LREAL	mm
Z-shift	Z-position of Hook 2 in relation to the MCS	LREAL	mm
Hook Center	Hook of the central pillar		
X-shift	X-position of Hook Center in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook Center in relation to the MCS	LREAL	mm

Parameter	Description	Type	Unit
Z-shift	Z-position of Hook Center in relation to the MCS	LREAL	mm

Parameters Central Pillar

The parameters for the central pillar including the rail are specified in relation to the arm position parallel to the Z-axis.

Parameter	Description	Type	Unit
X1 offset	Distance between the hook of the first arm and the central pillar in the X-direction	LREAL	mm
X2 offset	Distance between the hook of the second arm and the central pillar in the X-direction	LREAL	mm
Y1 offset	Distance between the hook of the first arm and the central pillar in the Y-direction	LREAL	mm
Y2 offset	Distance between the hook of the second arm and the central pillar in the Y-direction	LREAL	mm
Z offset	Distance between the hook of the central pillar and the connection of the arms to the central pillar	LREAL	mm
Rail offset			
X rail offset	Distance between the central point of the tip of the central pillar and the tip of the rail in the X-direction	LREAL	mm
Y rail offset	Distance between the central point of the tip of the central pillar and the tip of the rail in the Y-direction	LREAL	mm
Z rail offset	Distance between the central point of the tip of the central pillar and the tip of the rail in the Z-direction	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

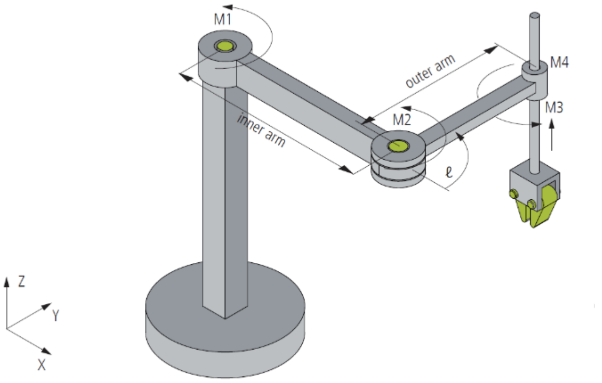
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4024.7 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.30	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.15 4D-SCARA (S_CCZC)



The 4D-SCARA (**S**elective **C**ompliance **A**ssembly **R**obot **A**rm) Kinematics (S_CCZC) are configured as shown in the diagram above.

The motor axes M1, M2 and M4 are scaled in degrees whereat the arrow indicates the positive direction of rotation. The third motor axis M3 is scaled in millimeters.

The origin of the MCS is located in the first joint (M1). The X-axis is determined by the SCARA arm, when all rotational motor axes reside at 0°.

The extension position of the SCARA arm (all rotary motor axes at position 0°) cannot be reached in cartesian mode because the robot is in a singular position there. It is only possible to drive to these positions in axis mode (Direct Mode).

Parameters for the Kinematics

Parameter	Description	Type	Unit
Inner arm length	Length from pivot point to pivot point of the inner arm; this is the arm on the origin side.	LREAL	mm
Outer arm length	Length from pivot point to pivot point of the outer arm; this is the arm on the TCP side.	LREAL	mm
Gear coupling	Coupling factor between the axes M4 and M3.	LREAL	mm/°
Flange rotation A	Rotation angle around the local X-axis.	LREAL	°
Tool offset OID	Object ID of a tool mounted on the kinematics flange. The flange coordinate system is rotated 180° around the X-axis so that its Z-axis points downwards.	OTCID	

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

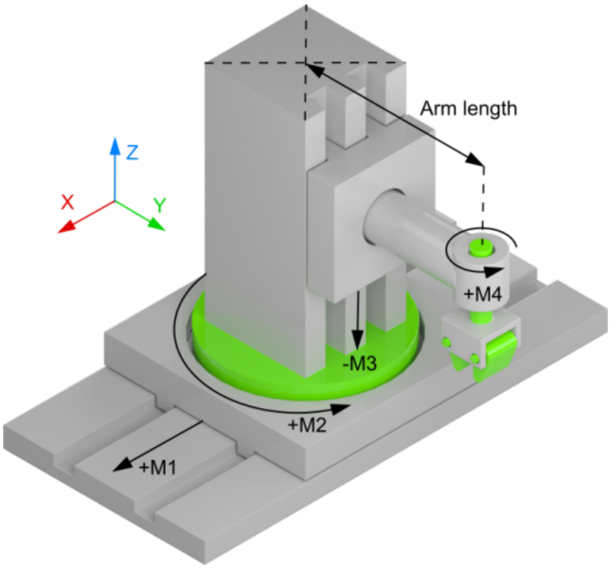
For all kinematics with tool also applies:

- Tool Offset OID [▸ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.16 4D-Kinematics Type 6 (S_XCZC)



The 4D-Kinematics Type 6 (S_XCZC) describes a serial kinematic transformation that is structured as shown in schematic above.

The motor axes M2 and M4 are scaled in degrees, the positive direction of rotation being in the direction of the arrow.

The motor axes M1 and M3 are scaled in millimeters. In relation to the MCS, M1 specifies a movement on the X-axis and M3 a movement on the Z-axis. The origin of the MCS coordinate system is located on the linear axis M1 in joint M2.

Moving to singular positions

Singular positions, as with this robot type, e.g. M2 = +-90°, cannot be moved to in cartesian mode. It is only possible to drive to these positions in axis mode (Direct Mode).

Kinematics parameters

Parameter	Description	Type	Unit
Arm length	Distance between rotary axis M2 and rotary axis M4 Arm length > 0	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

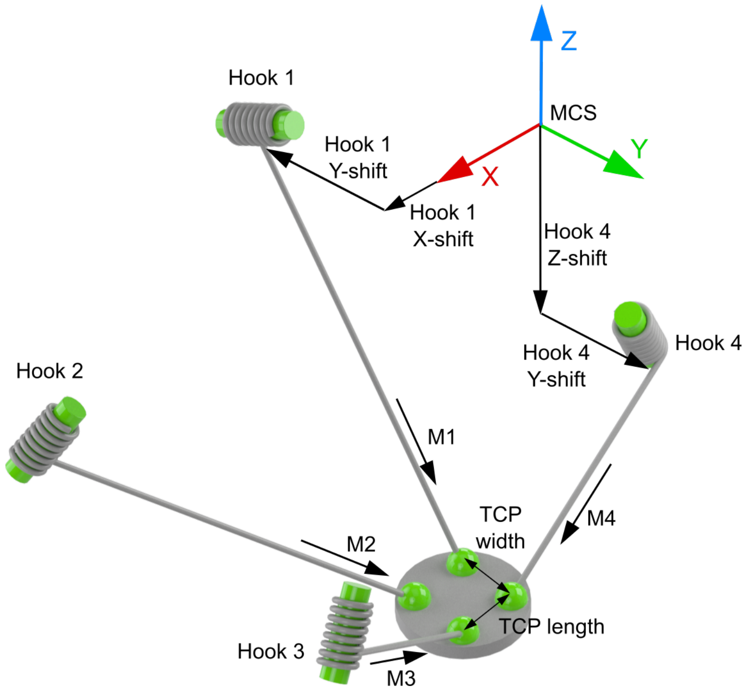
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4024.7 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.30	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.17 4D-Cable-Kinematics (P_4L)



The 4D-Cable Kinematics (P_4L) is structured as shown in the above schematic.

The zero point of the machine coordinate system (MCS) can be anywhere in space. The hooks of the cable/ rope are defined starting from the MCS origin.

All motor axes are scaled in millimeters, the arrow indicating the positive direction.

Parameters for joint hooks

Parameter	Description	Type	Unit
Hook 1	Hook of the first cable		
X-shift	X-position of Hook 1 in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook 1 in relation to the MCS	LREAL	mm
Z-shift	Z-position of Hook 1 in relation to the MCS	LREAL	mm
Hook 2	Hook of the second cable		
X-shift	X-position of Hook 2 in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook 2 in relation to the MCS	LREAL	mm
Z-shift	Z-position of Hook 2 in relation to the MCS	LREAL	mm
Hook 3	Hook of the third cable		
X-shift	X-position of Hook 3 in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook 3 in relation to the MCS	LREAL	mm
Z-shift	Z-position of Hook 3 in relation to the MCS	LREAL	mm
Hook 4	Hook of the fourth cable		
X-shift	X-position of Hook 4 in relation to the MCS	LREAL	mm
Y-shift	Y-position of Hook 4 in relation to the MCS	LREAL	mm
Z-shift	Z-position of Hook 4 in relation to the MCS	LREAL	mm
TCP	Tool Center Point		
TCP length	Distance between the hooks at the TCP along the X-axis	LREAL	mm
TCP width	Distance between the hooks at the TCP along the Y-axis	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

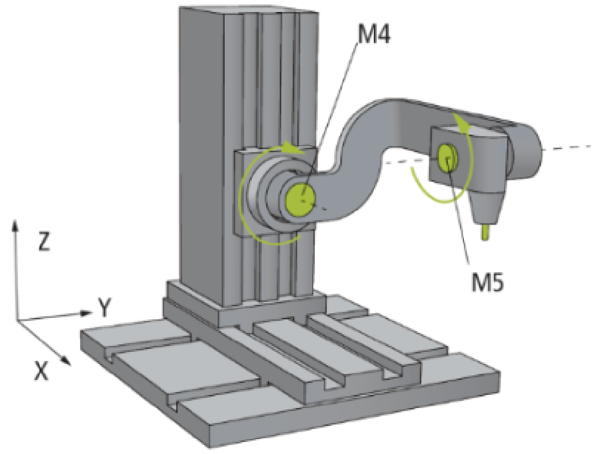
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4024.7 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.10.30	PC or CX (x86 or x64)	TF5112 TwinCAT 3 Kinematic Transformation (Level 3)

6.18 5D-Kinematics Type 2 (XYZab)



The 5D-Kinematics Type 2 (XYZab) are configured as shown in the drawing above.

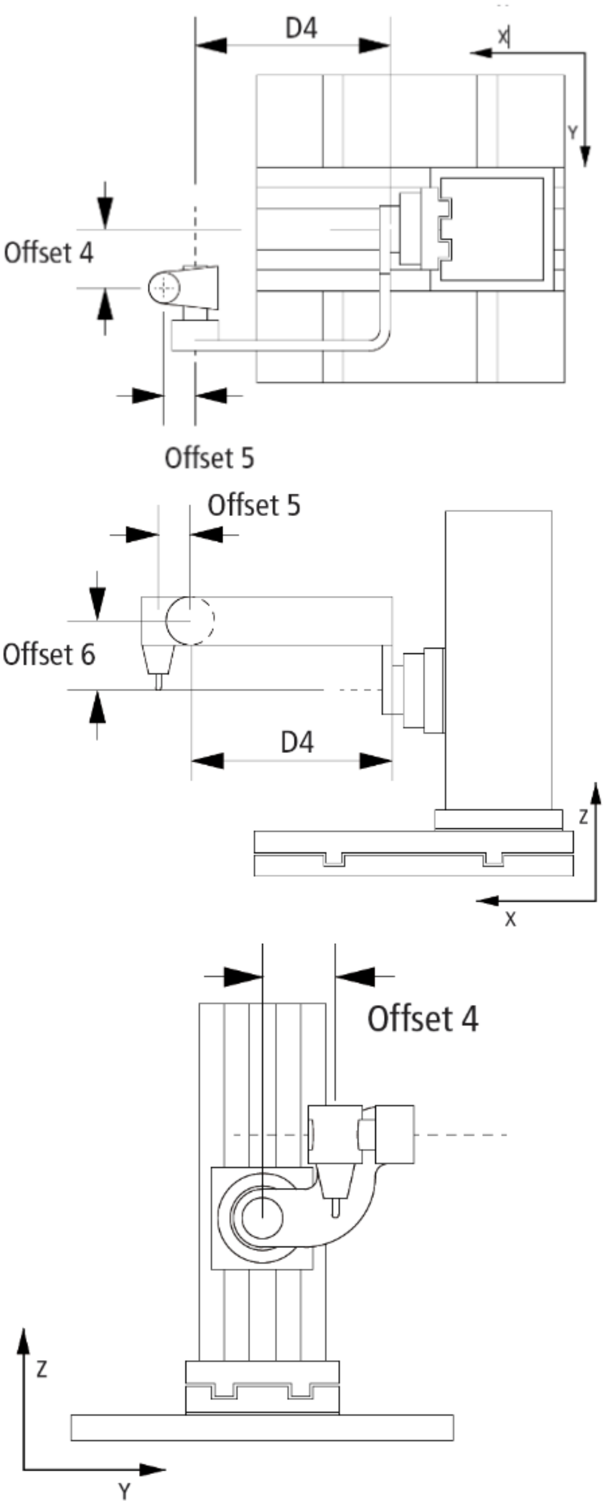
The motors M1 to M3 (X, Y, Z) are scaled in millimeters. The motors M4 and M5 are scaled in degrees. The 0° position is the axis position shown in the drawing; the arrows indicate the positive direction of rotation.

Difference of Type 2

The 5D-Kinematics Type 2 differ from the 5D-Kinematics Type 3 in the orientation of the positive direction of axis rotation around the motor axes M4 and M5.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Handle D4	Arm length in X-direction between motor axis 4 and motor axis 5 as shown in the drawing.	LREAL	mm
Offset 4	Offset in y-direction between motor axis 4 and TCP.	LREAL	mm
Offset 5	Offset in X-direction between motor axis 5 and TCP.	LREAL	mm
Offset 6	Offset in Z-direction between motor axis 4 and motor axis 5.	LREAL	mm
Tool offset OID	Object ID of a tool mounted on the kinematics flange. The flange coordinate system is rotated 180 ° around the X-axis so that its Z-axis points downwards.	OTCID	



General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

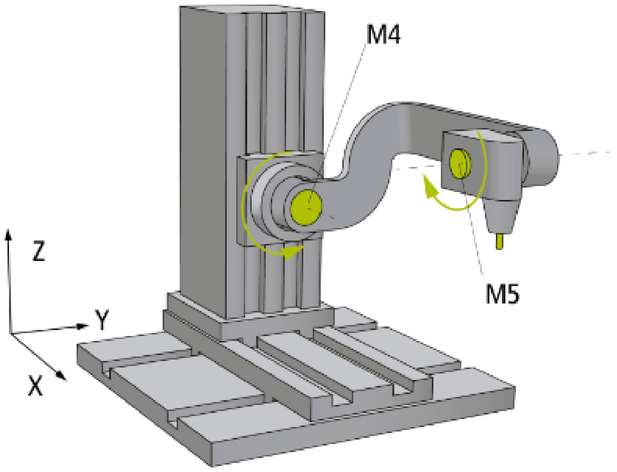
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5113 TwinCAT 3 Kinematic Transformation (Level 4)

6.19 5D-Kinematics Type 3 (XYZAB)



The 5D-Kinematics Type 3 (XYZAB) are configured as shown in the drawing above.

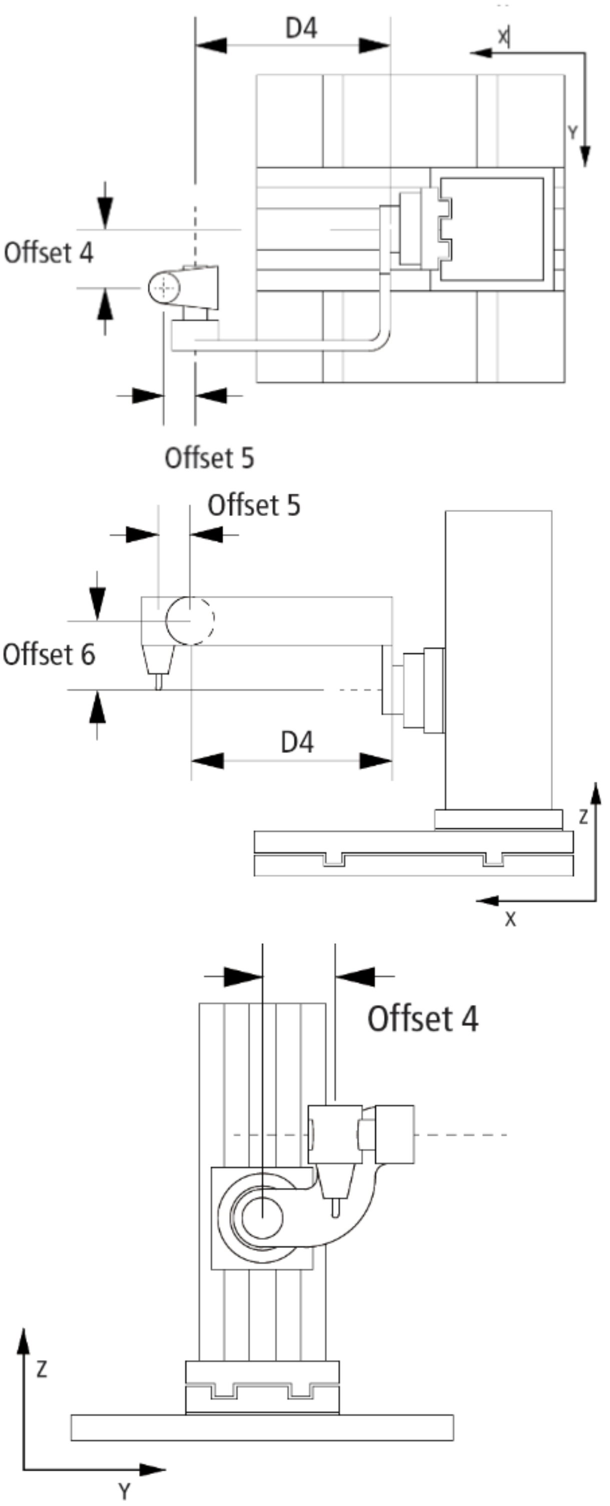
The motors M1 to M3 (X, Y, Z) are scaled in millimeters. The motors M4 and M5 are scaled in degrees. The 0° position is the axis position shown in the drawing; the arrows indicate the positive direction of rotation.

Difference of Type 3

The 5D-Kinematics Type 3 differ from the 5D-Kinematics Type 2 in the orientation of the positive direction of axis rotation around the motor axes M4 and M5.

Parameters for the Kinematics

Parameter	Description	Type	Unit
Handle D4	Arm length in X-direction between motor axis 4 and motor axis 5 as shown in the drawing.	LREAL	mm
Offset 4	Offset in y-direction between motor axis 4 and TCP.	LREAL	mm
Offset 5	Offset in X-direction between motor axis 5 and TCP.	LREAL	mm
Offset 6	Offset in Z-direction between motor axis 4 and motor axis 5.	LREAL	mm
Tool offset OID	Object ID of a tool mounted on the kinematics flange. The flange coordinate system is rotated 180° around the X-axis so that its Z-axis points downwards.	OTCID	



General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

For all kinematics with tool also applies:

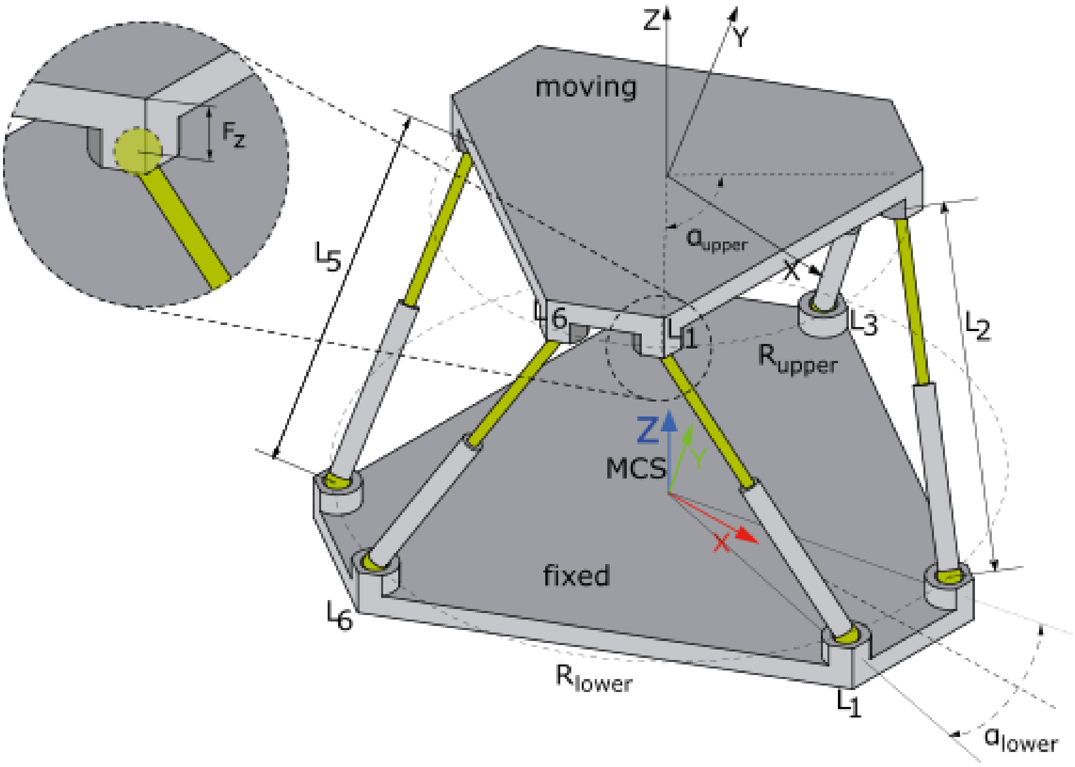
- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4018.26 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.14	PC or CX (x86 or x64)	TF5113 TwinCAT 3 Kinematic Transformation (Level 4)

6.20 6D-Stewart Platform (P_6L)

For the kinematic transformation 6D-Stewart Platform (P_6L), a moving platform is supported by six cylinders. The Stewart Platform is a parallel kinematics with six degrees of freedom.



The X axis of the machine coordinate system (MCS) points toward the point that is in the center between joints 1 and 2. The zero point in the Z direction is usually slightly above the lower (base) platform and in the plane that is spanned by the joint centers of the anchor points.

All joints on a platform are on a circular path and thus have the same distance to the center of the platform. This distance must be specified with R_{lower} (lower platform) and R_{upper} (upper platform).

The angle between joints 1 and 2, which is equally between joints 3 and 4 and between joints 5 and 6, is specified with as α_{lower} (lower platform) and α_{upper} (upper platform).

The ACS axis positions always refer to the entire cylinder length L. Startup with an ACS axis position equal to 0 is therefore not possible.

Kinematics parameters

The following parameters are available for Stewart kinematics:

Parameter	Description	Type	Unit
Flange translation Z	Moves the upper level to the surface of the upper platform so that the thickness of the platform is taken into account.	LREAL	mm
Tool offset OID	Set the tool elongation.	OTCID	

Parameter	Description	Type	Unit
Lower radius R_{lower}	Radius of the lower platform. Describes the distance from the origin in the center of the lower platform to the arm joints of the lower platform.	`LREAL`	mm
Upper radius R_{upper}	Radius of the upper platform. Describes the distance from the origin in the center of the upper platform to the arm joints of the upper platform.	`LREAL`	mm
Lower angle α_{lower}	The angle α_{lower} describes the half angle between the anchor points of L_1 and L_2, L_3 and L_4, and L_5 and L_6 on the lower platform.	`LREAL`	°
Upper angle α_{upper}	The angle α_{upper} describes the half angle between the anchor points of L_1 and L_2, L_3 and L_4, and L_5 and L_6 on the upper platform.	`LREAL`	°

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▶ 19],
- Spatial reference definition [▶ 19].

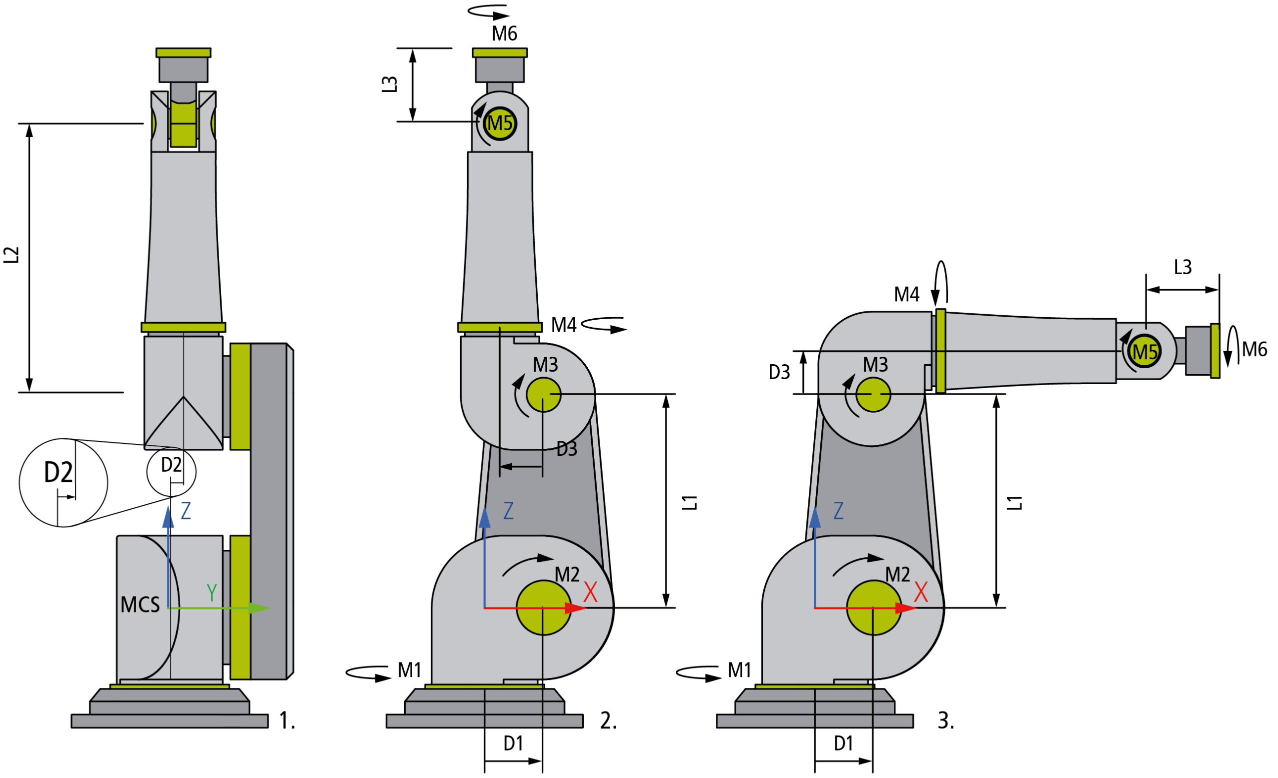
For all kinematics with tool also applies:

- Tool Offset OID [▶ 21].

Requirements

Development Environment Installation Package	Target System	TwinCAT Function
TwinCAT V3.1.4024.11 TF5400 TwinCAT 3 Advanced Motion Pack V3.1.6.67	PC or CX (x86 or x64)	TF5113 TwinCAT 3 Kinematic Transformation (Level 4)

6.21 Six Axis Articulated (S_CBBCBC)



The motor axes of the Serial Six Axis Articulated (S_CBBCBC) Kinematics each are referred to in units of degree. The drawings 1. and 2. above show the kinematics with all axes in zero position. The zero positions of the axes M4 and M6 are defined in a way that the machine coordinate system and the flange coordinate system exhibit the same orientation. The drawing 3. shows the axis M3 in 90° position.

The MCS origin is located within the intersection of the first kinematic joint M1 with the second kinematic joint M2. It is oriented in a way that joint M2 prescribes a rotation around the Y-axis. The center of M1 delivers the X zero coordinate. The intersection of M1 and M2 delivers the Y zero coordinate. The center of M2 delivers the Z zero coordinate.

Singular Positions

The positions displayed in figures 1., 2. and 3. cannot be approached in cartesian Mode because the robot resides in a singular position. Approaching these positions is only possible in axis mode (Direct Mode).

Configuration Parameters

Parameter	Description	Type	Unit
MCS offset	Static offset to the MCS coordinate frame.		
X-shift	Static X-offset in MCS coordinate frame.	LREAL	mm
Y-shift	Static Y-offset in MCS coordinate frame.	LREAL	mm
Z-shift	Static Z-offset in MCS coordinate frame.	LREAL	mm
Spatial reference definition	Specification of the spatial reference node.		
.Translation X	Translation in X-direction.	LREAL	mm
.Translation Y	Translation in Y-direction.	LREAL	mm
.Translation Z	Translation in Z-direction.	LREAL	mm
.Rotation 1	Rotation angle 1, its interpretation is set by the rotation convention.	LREAL	°

Parameter	Description	Type	Unit
.Rotation 2	Rotation angle 2, its interpretation is set by the rotation convention.	LREAL	°
.Rotation 3	Rotation angle 3, its interpretation is set by the rotation convention.	LREAL	°
.Rotation convention	Set the interpretation of the rotation angles.	MC.CoordInterpretation_SO3	
.Spatial reference	Set the spatial frame of reference, 0x0 refers to WCS.	OTCID	
.Definition direction	Set the definition direction.	MC.ReferenceDefDir	

Parameters for the Kinematics

For the Six Axis Articulated Kinematics, a serial six axis kinematic, there are the following joint parameters.

Parameter	Description	Type	Unit
Arm length L1	Distance between the motor axes M2 and M3.	LREAL	mm
Arm length L2	Distance between the motor axes M3 and M5.	LREAL	mm
Arm length L3	Distance between the motor axis M5 and the flange.	LREAL	mm
Arm offset D1	Distance between the motor axes M1 and M2 in X-direction.	LREAL	mm
Arm offset D2	Distance in Y-direction between the motor axes M1 and M4.	LREAL	mm
Arm offset D3	Distance in X-direction between the motor axes M3 and M5. The sign within the example sketch is positive.	LREAL	mm

General Parameters for the Kinematics

General parameters that apply to any kinematics are described in the following sections:

- MCS Offset [▸ 19],
- Spatial reference definition [▸ 19].

For all kinematics with tool also applies:

- Tool Offset OID [▸ 21].

6.22 Drive Torque

The drive torque represents the inertia and the efficiency of the motor and gear unit. It is used for the precise computation of the dynamic model.

A parameter in the kinematics can be used to assign an object drive torque to a kinematics.

Parameter for drive

Parameter	Description	Unit
Drive moment of inertia	Rotor Moment of inertia of the motor	kg mm^2

Gear unit parameters

Parameter	Description	Unit
Ratio	Gear ratio	

Parameter	Description	Unit
Gearbox moment of inertia	Moment of inertia of the gear unit in relation to the drive	kg mm^2
Coulomb friction	Represents the kinetic friction coefficient	Nm
Stokes friction	Represents the friction ratio that increases proportionally to the speed	Nms

Required product level:
Level 1

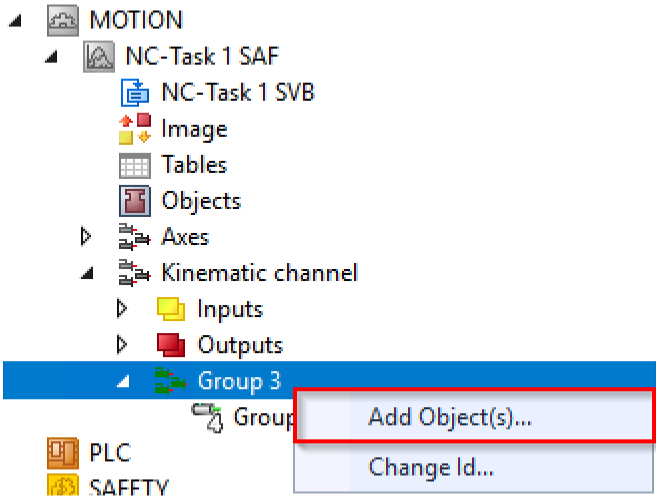
6.23 Tool Offset

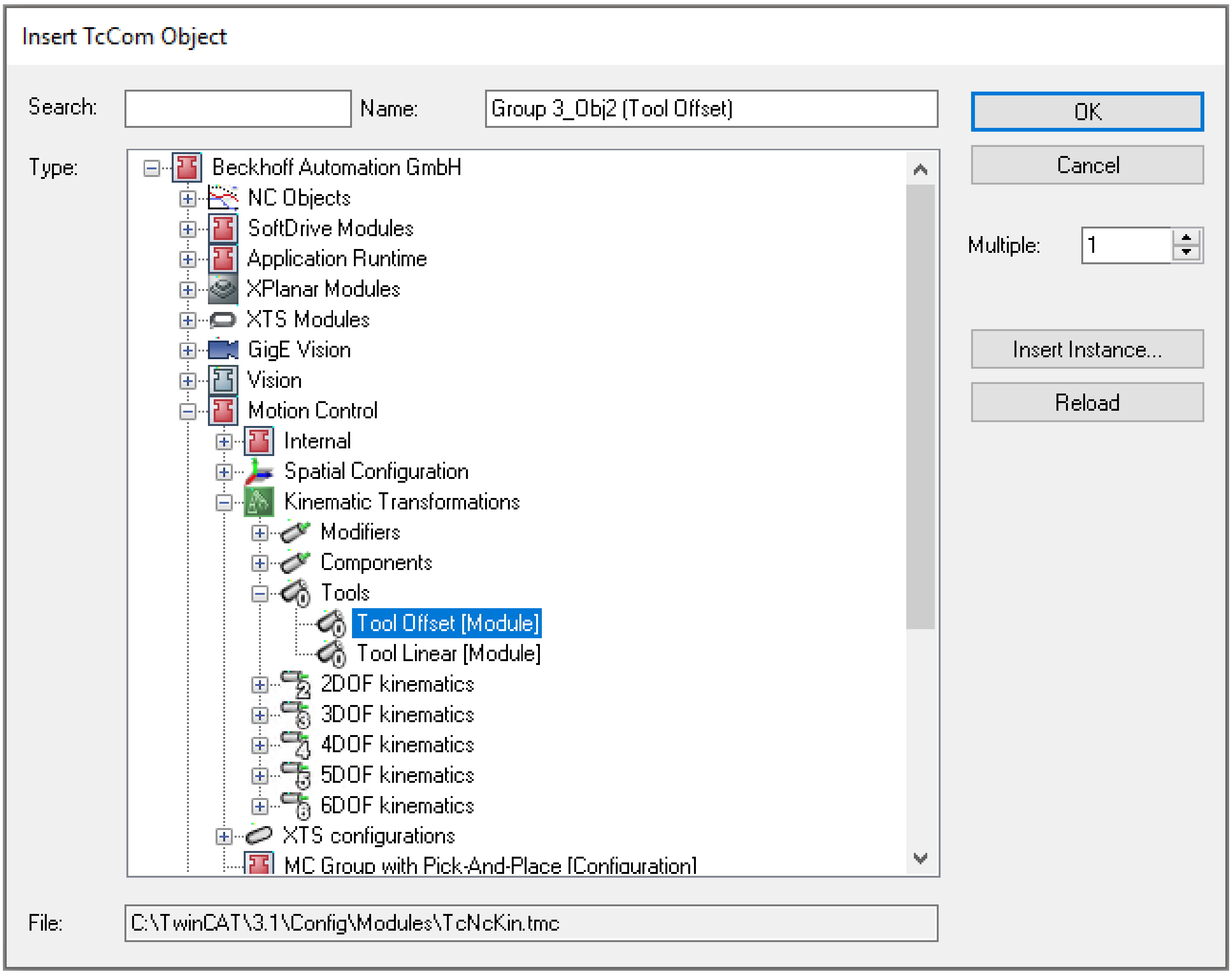
The user can use the tool offset to link a tool with the flange of the kinematics. Unless specified otherwise in the kinematics, the flange coordinate system is defined such that the orientation of the flange coordinate system matches that of the machine coordinate system MCS, if all axes are at 0.

Parameter	Description	Unit
Extension X	X-offset of the static tool, which is mounted at the coordinate system of the flange of the higher-level transformation	mm
Extension Y	Y-offset of the static tool, which is mounted at the coordinate system of the flange of the higher-level transformation	mm
Extension Z	Z-offset of the static tool, which is mounted at the coordinate system of the flange of the higher-level transformation	mm

Creating a tool

1. First, a tool has to be created under the group of the kinematics.





Insert TcCom Object

Search:

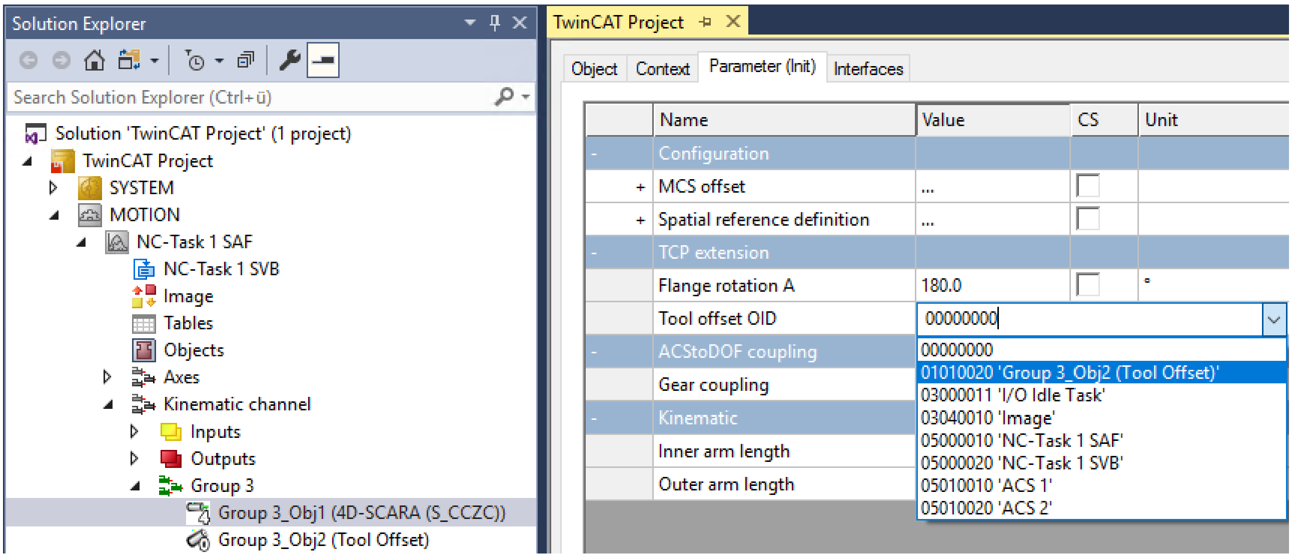
Name: Group 3_Obj2 (Tool Offset)

Type:

- Beckhoff Automation GmbH
  - NC Objects
  - SoftDrive Modules
  - Application Runtime
  - XPlanar Modules
  - XTS Modules
  - GigE Vision
  - Vision
  - Motion Control
    - Internal
    - Spatial Configuration
    - Kinematic Transformations
      - Modifiers
      - Components
      - Tools
        - Tool Offset [Module]
        - Tool Linear [Module]
      - 2DOF kinematics
      - 3DOF kinematics
      - 4DOF kinematics
      - 5DOF kinematics
      - 6DOF kinematics
    - XTS configurations
    - MC Group with Pick-And-Place [Configuration]

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

2. The created tool object can be allocated to the kinematics in the parameters via its tool OID.



Solution Explorer

Search Solution Explorer (Ctrl+ü)

- Solution 'TwinCAT Project' (1 project)
  - TwinCAT Project
    - SYSTEM
    - MOTION
      - NC-Task 1 SAF
        - NC-Task 1 SVB
        - Image
        - Tables
        - Objects
        - Axes
        - Kinematic channel
          - Inputs
          - Outputs
          - Group 3
            - Group 3_Obj1 (4D-SCARA (S_CCZC))
            - Group 3_Obj2 (Tool Offset)

TwinCAT Project

	Name	Value	CS	Unit
-	Configuration			
+	MCS offset	...	□	
+	Spatial reference definition	...	□	
-	TCP extension			
	Flange rotation A	180.0	□	°
	Tool offset OID	00000000		
-	ACStoDOF coupling			
	Gear coupling			
-	Kinematic			
	Inner arm length			
	Outer arm length			

00000000
01010020 'Group 3_Obj2 (Tool Offset)'
03000011 'I/O Idle Task'
03040010 'Image'
05000010 'NC-Task 1 SAF'
05000020 'NC-Task 1 SVB'
05010010 'ACS 1'
05010020 'ACS 2'

3. The tool can now be configured via its object parameters.

6.24 Tool Linear

The Tool Linear describes a 1D tool, which is mounted at the flange of the kinematics. An additional simulation axis can be used for movement in tool direction. The 1D tool can be used to move the TCP at a certain distance from a workpiece.

Unless specified otherwise in the kinematics, the flange coordinate system is defined such that the orientation of the flange coordinate system matches that of the machine coordinate system, if all axes are at 0.

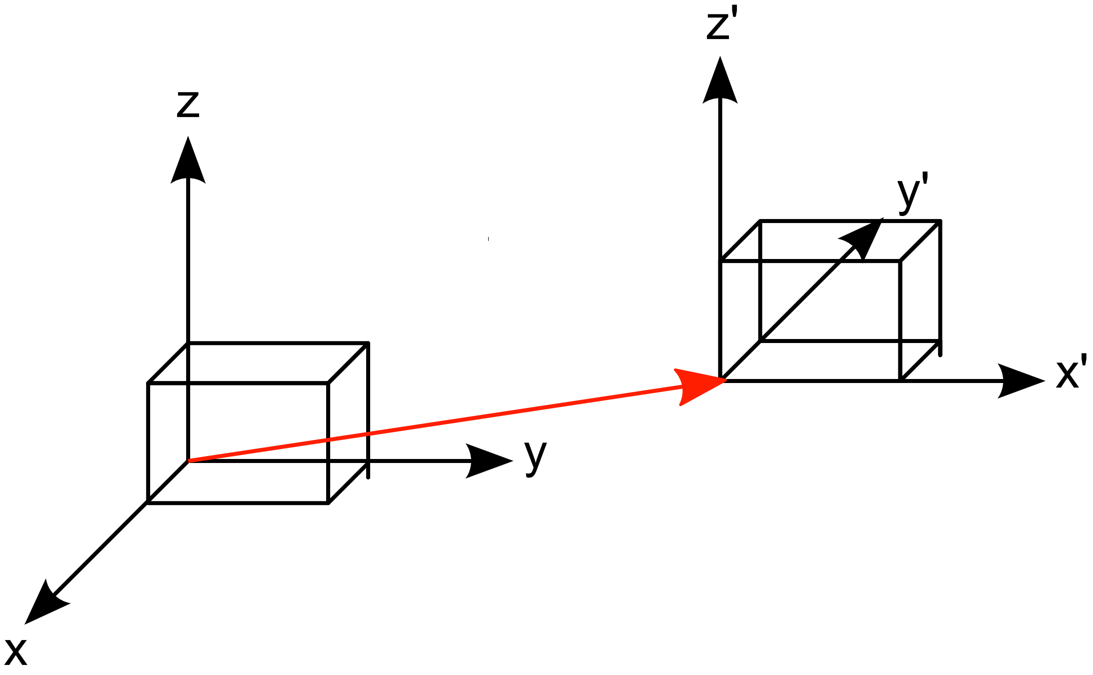
If the axis position of the additional simulation axes is 0, the TCP is at the position of the tool offset (parameter L_init).

Parameter	Description	Type	Unit
Length offset	Tool length	LREAL	mm
Length axis ID	Axis ID of the simulation axis; when this axis is moved, the TCP moves in the direction of the linear tool.	LREAL	

To create a tool see Tool Offset [▸ 51].

6.25 Coordinate system (Coordinate Frame)

The coordinate system supports a translation and a rotation. This transformation can be used to define a user coordinate system (UCS). A general introduction to coordinate systems can be found Introduction [▸ 8].



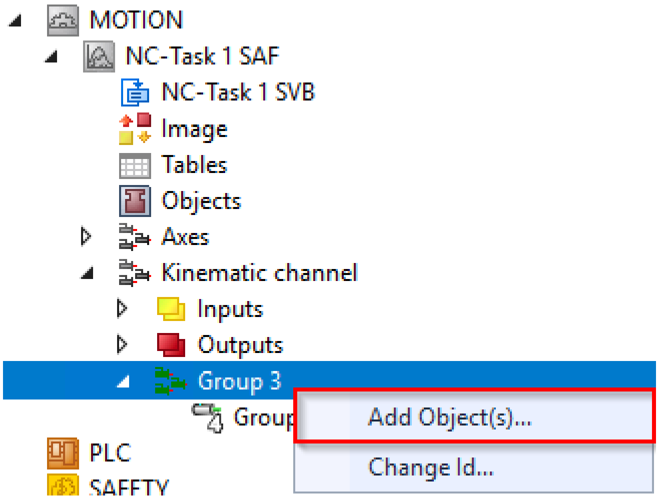
First the translation is calculated, then the rotation. The order of the rotations affects the orientation of the coordinate system. The roll/pitch/yaw rule described in DIN 9300 is used as default for the rotation sequence. The calculation sequence for the forward transformation is Z, Y', X".

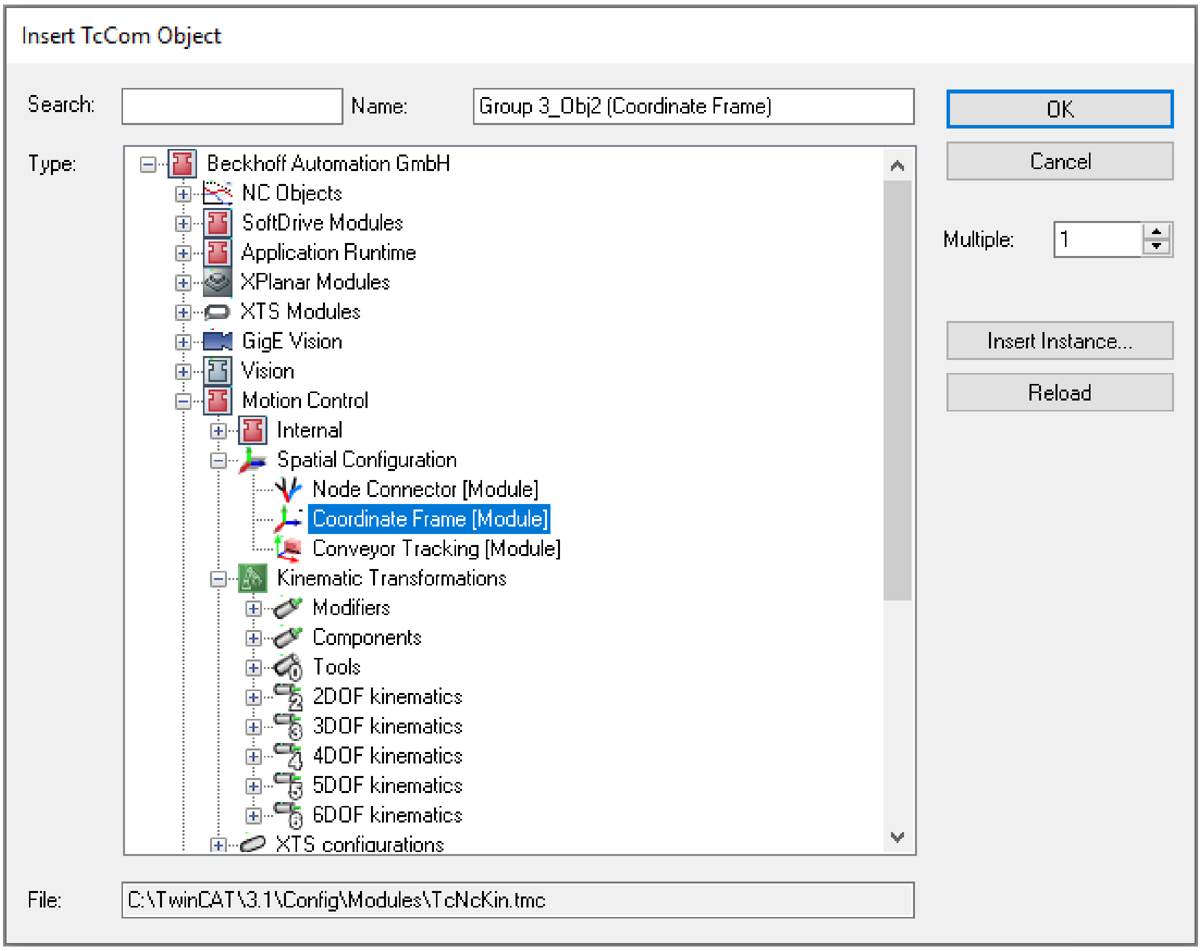
Parameter	Description	Unit
Translation X	Shift in x-direction	mm
Translation Y	Shift in y-direction	mm
Translation Z	Shift in z-direction	mm
Rotation 1	First rotation angle. The interpretation is defined by the parameter Rotation Convention.	°
Rotation 2	Second rotation angle. The interpretation is defined by the parameter Rotation Convention.	°
Rotation 3	Third rotation angle. The interpretation is defined by the parameter Rotation Convention.	°

Parameter	Description	Unit
Rotation convention	The rotation convention indicates the order of the axis rotations (parameter Rotation 1-3). The letters (X, Y, Z) from left to right indicate the order of the rotation around the corresponding axes. The number indicates the parameter (Rotation 1-3) for the value parameterization. The translatory shift is always performed before the rotation.	
Spatial reference	The parameter Spatial reference indicates which coordinate system is used as basis for this coordinate system. If the value is set to 0, the WCS is used as basis. To use another coordinate system as starting point for the shift, a further coordinate system object can be created. The object ID of this coordinate system can be selected via the dropdown menu.	
Definition direction	Indicates the direction in which the shift is programmed (from the perspective of the reference system or this coordinate system).	

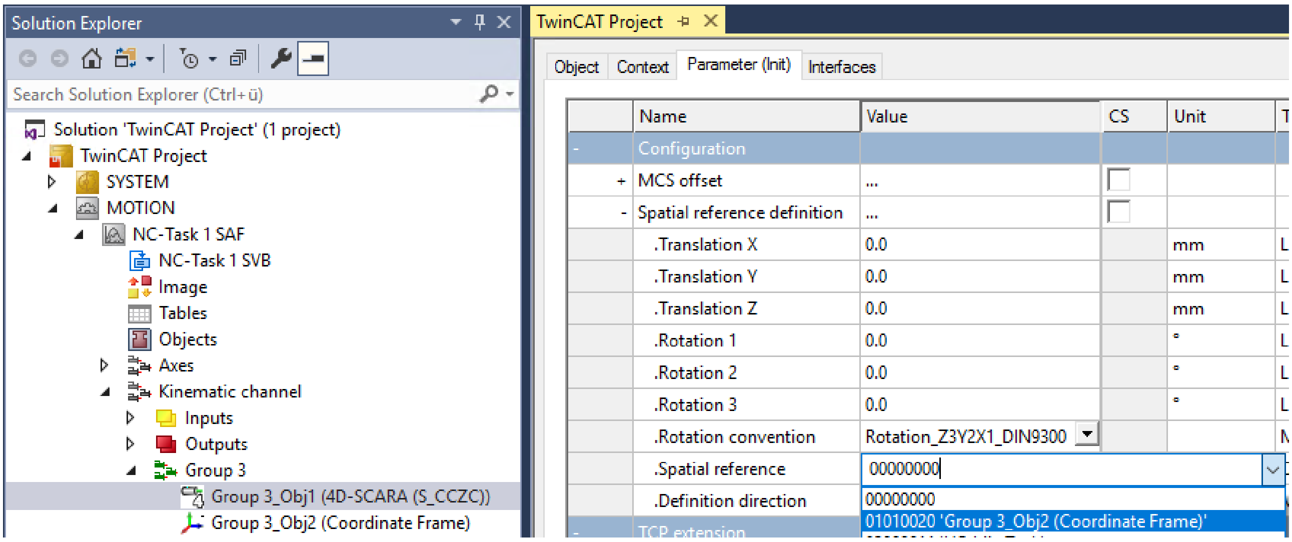
Creating a coordinate system

1. First create the coordinate system under the kinematic group.





2. The created coordinate system object can be defined in the kinematics via the parameter Spatial reference as origin of the MCS of the kinematics.



Solution Explorer

Search Solution Explorer (Ctrl+ü)

- Solution 'TwinCAT Project' (1 project)
  - TwinCAT Project
    - SYSTEM
    - MOTION
      - NC-Task 1 SAF
        - NC-Task 1 SVB
        - Image
        - Tables
        - Objects
        - Axes
        - Kinematic channel
          - Inputs
          - Outputs
          - Group 3
            - Group 3_Obj1 (4D-SCARA (S_CCZC))
            - Group 3_Obj2 (Coordinate Frame)

TwinCAT Project

Object | Context | Parameter (Init) | Interfaces

	Name	Value	CS	Unit	T
-	Configuration				
+	MCS offset	...	□		
-	Spatial reference definition	...	□		
	.Translation X	0.0		mm	L
	.Translation Y	0.0		mm	L
	.Translation Z	0.0		mm	L
	.Rotation 1	0.0		°	L
	.Rotation 2	0.0		°	L
	.Rotation 3	0.0		°	L
	.Rotation convention	Rotation_Z3Y2X1_DIN9300			N
	.Spatial reference	00000000			
	.Definition direction	00000000 01010020 'Group 3_Obj2 (Coordinate Frame)'			
-	TCP extension				

3. The coordinate system can now be configured via its object parameters.

7 User Specific Transformation - How to...

⚠ WARNING

Avoid Online Change

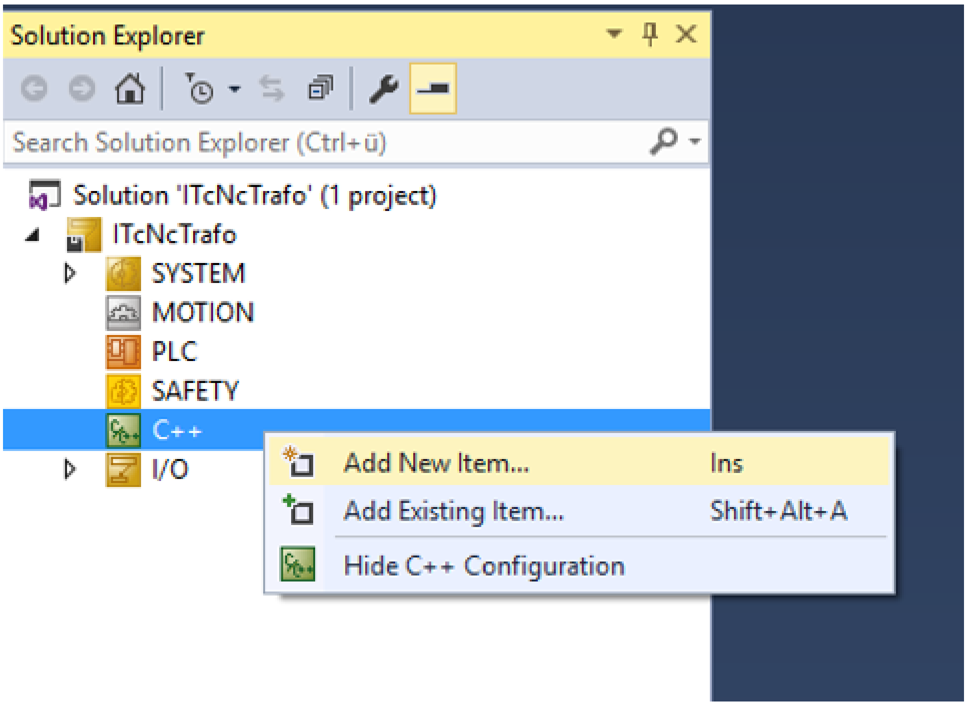
When doing an Online Change in combination with user specific kinematics there might occur discontinuities on C++ side. Therefore, an online change should generally be avoided. Note the chapter “Quickstart with Online Change”.

ITcNcTrafo

Step by Step Manual for how to Integrate own Kinematics with TF511x

✓ Start with an empty TwinCAT Project.

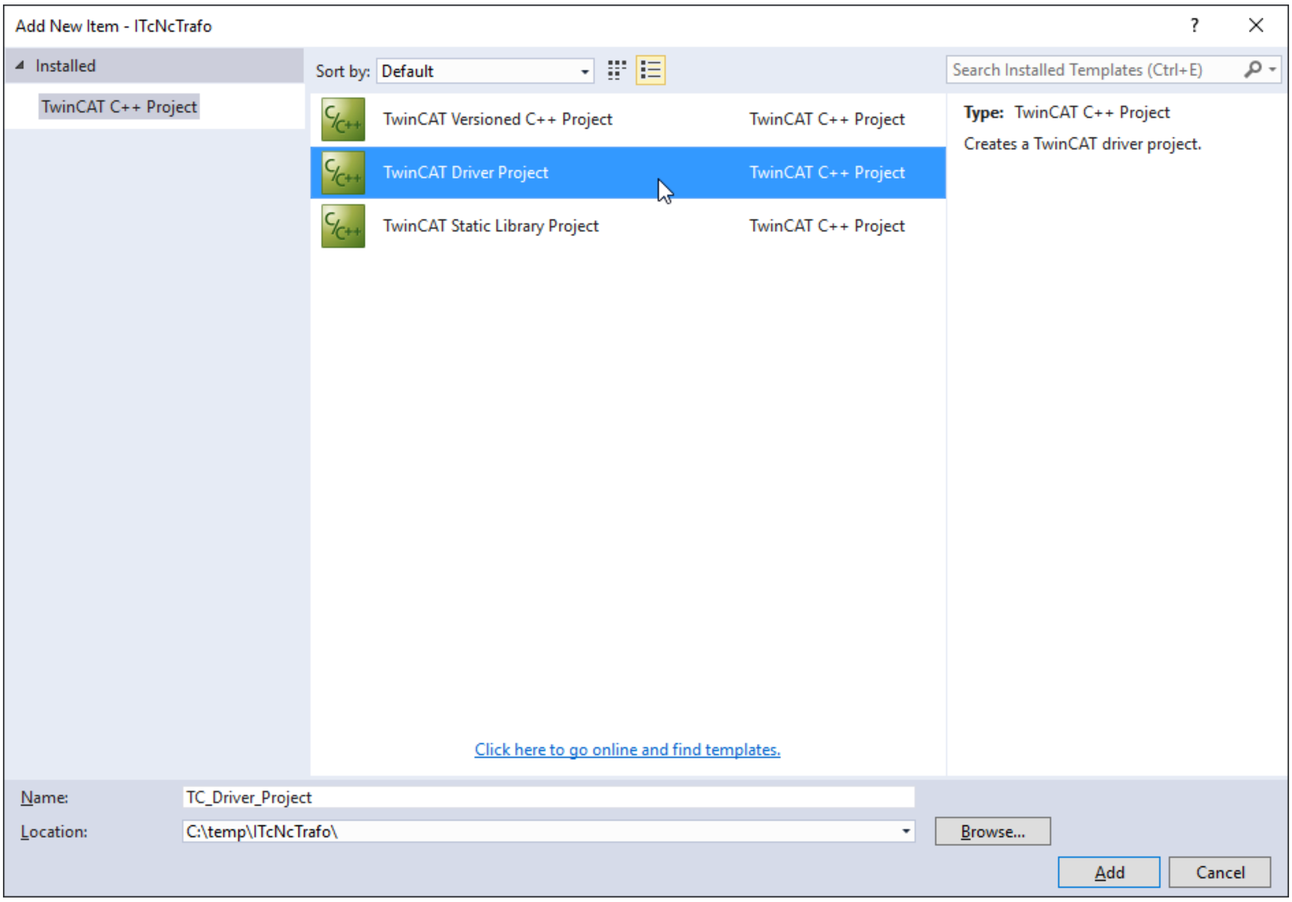
1. Add a C++ Object.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'ITcNcTrafo' (1 project)
ITcNcTrafo
SYSTEM
MOTION
PLC
SAFETY
C++
I/O
Add New Item... Ins
Add Existing Item... Shift+Alt+A
Hide C++ Configuration

2. Select a TwinCAT Driver Project or a TwinCAT Versioned C++ project.

Versioned C++ Project

A Versioned C++ Project can be used from TwinCAT V3.1.4024.10 for user specific kinematics.



Add New Item - ITcNcTrafo

Installed

TwinCAT C++ Project

Sort by: Default

TwinCAT Versioned C++ Project	TwinCAT C++ Project
TwinCAT Driver Project	TwinCAT C++ Project
TwinCAT Static Library Project	TwinCAT C++ Project

Type: TwinCAT C++ Project

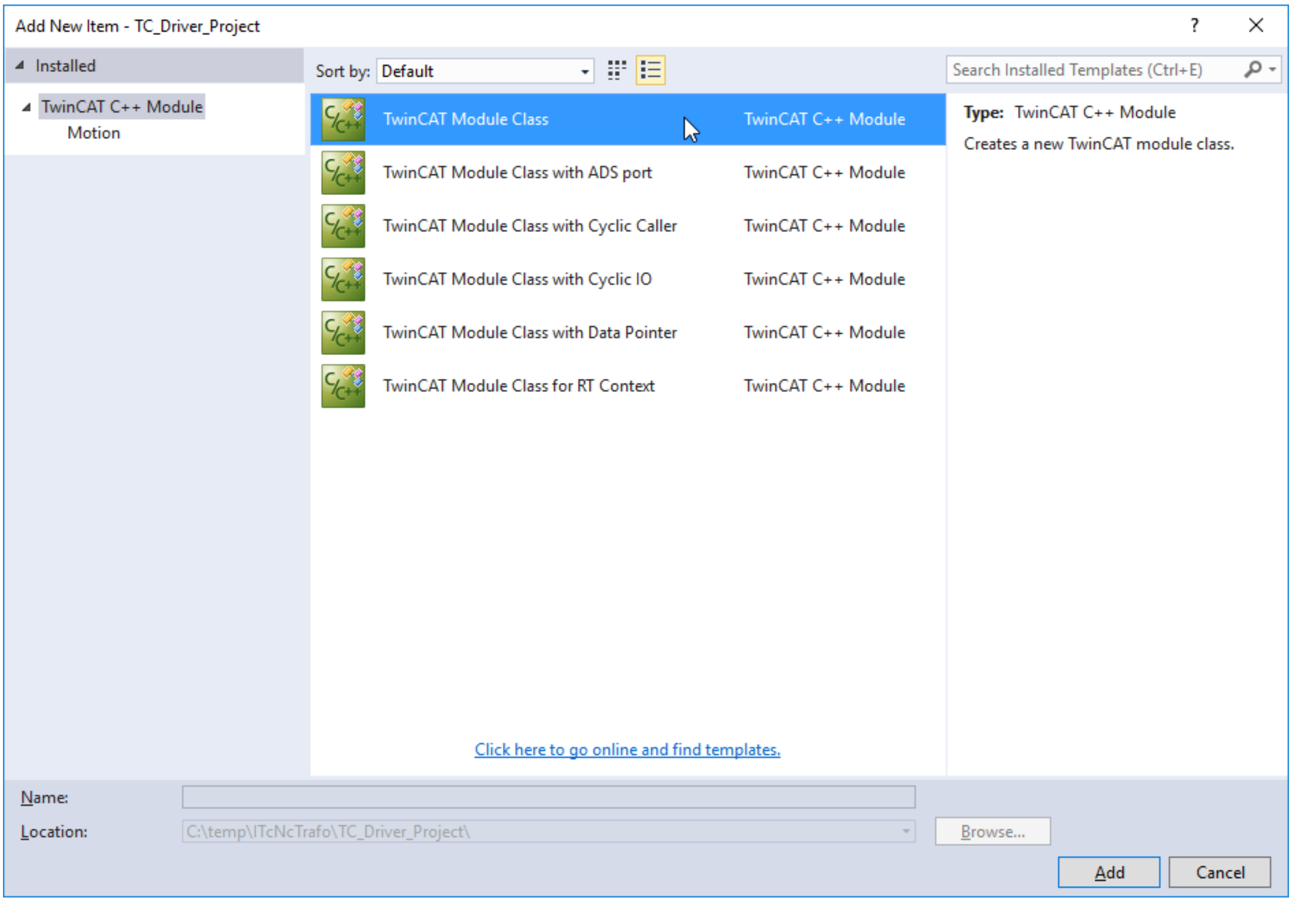
Creates a TwinCAT driver project.

Click here to go online and find templates.

Name: TC_Driver_Project

Location: C:\temp\ITcNcTrafo\

3. Select a TwinCAT Module Class.



Add New Item - TC_Driver_Project

Installed

TwinCAT C++ Module

Motion

Sort by: Default

TwinCAT Module Class	TwinCAT C++ Module
TwinCAT Module Class with ADS port	TwinCAT C++ Module
TwinCAT Module Class with Cyclic Caller	TwinCAT C++ Module
TwinCAT Module Class with Cyclic IO	TwinCAT C++ Module
TwinCAT Module Class with Data Pointer	TwinCAT C++ Module
TwinCAT Module Class for RT Context	TwinCAT C++ Module

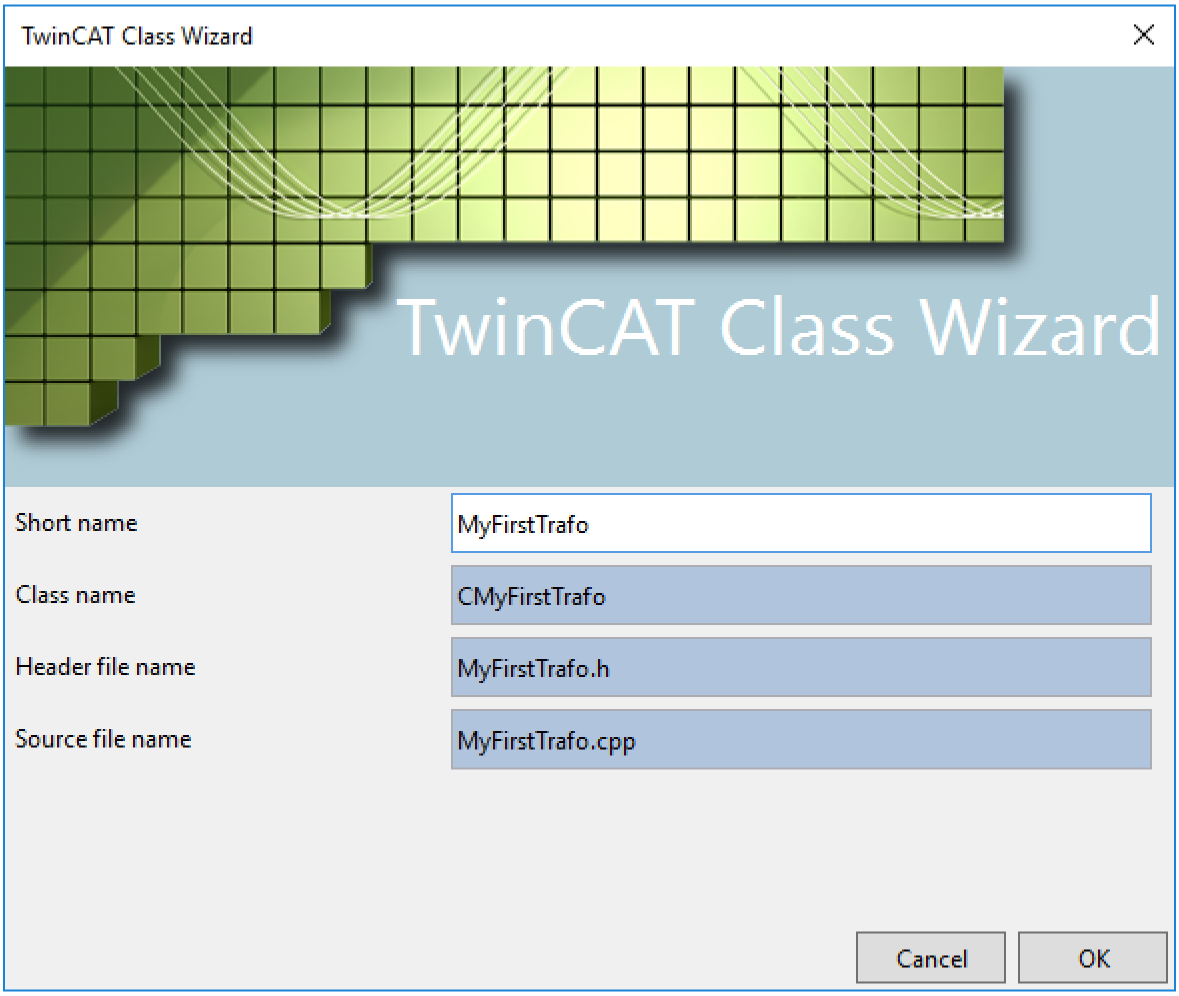
Type: TwinCAT C++ Module

Creates a new TwinCAT module class.

Click here to go online and find templates.

Name:

Location: C:\temp\ITcNcTrafo\TC_Driver_Project\



TwinCAT Class Wizard

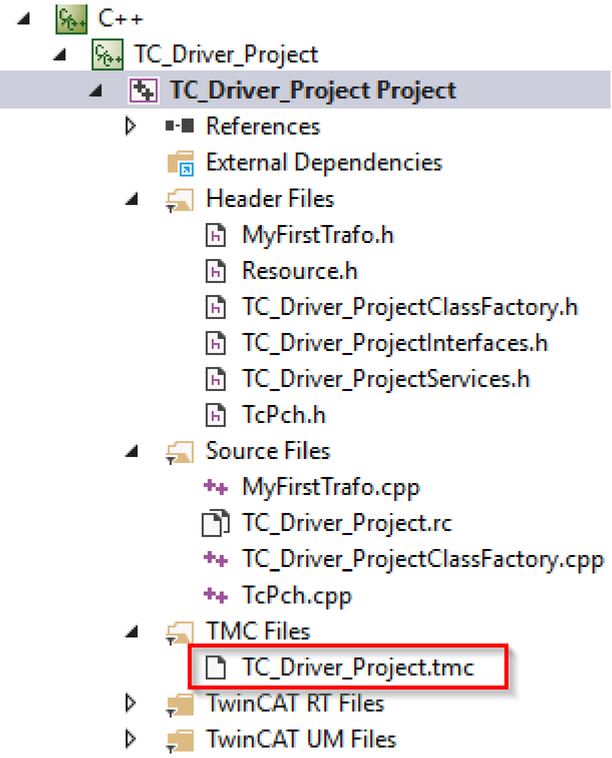
Short name: MyFirstTrafo

Class name: CMyFirstTrafo

Header file name: MyFirstTrafo.h

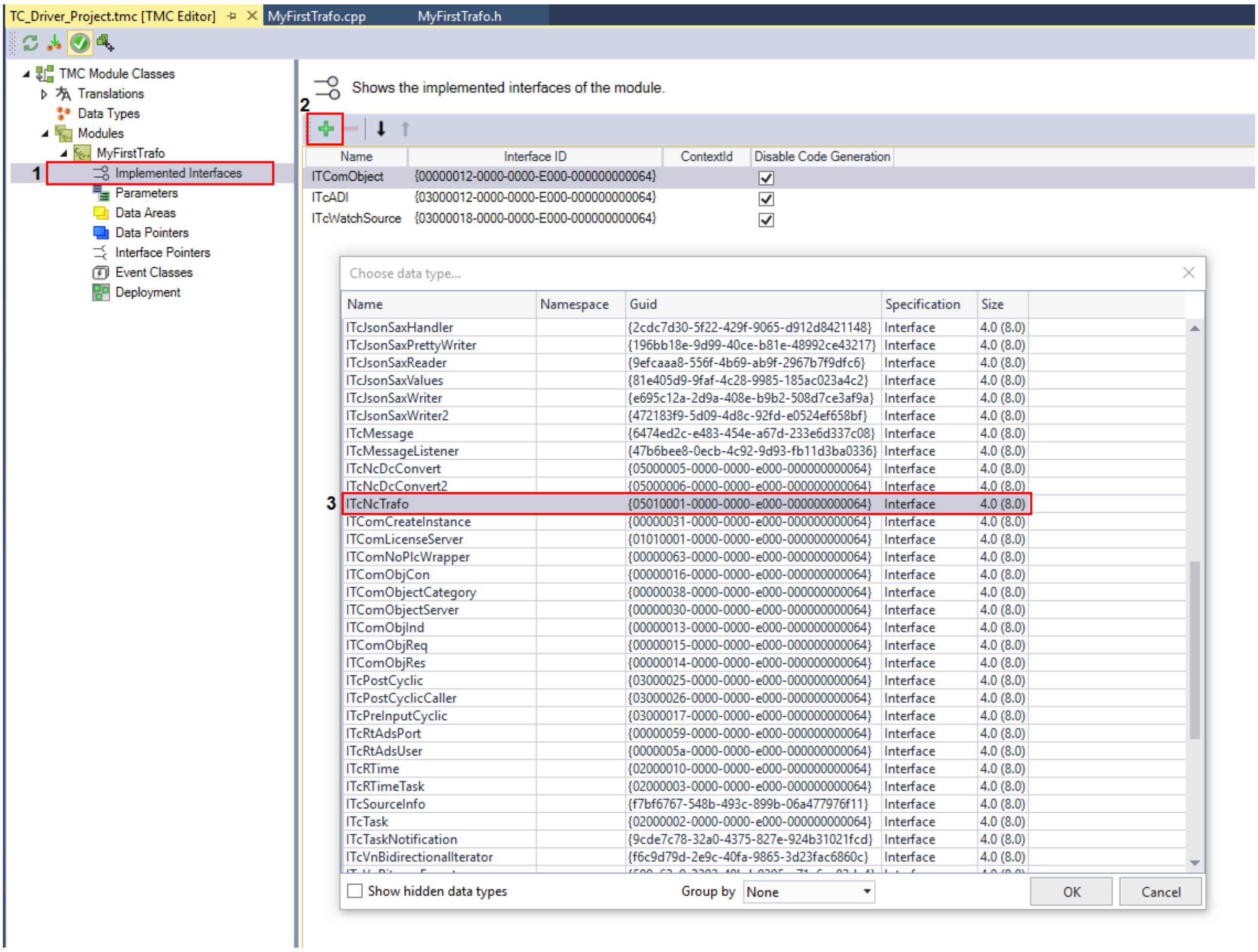
Source file name: MyFirstTrafo.cpp

4. Select the `TMC` Editor.



```
C++
  TC_Driver_Project
    TC_Driver_Project Project
      References
      External Dependencies
      Header Files
        MyFirstTrafo.h
        Resource.h
        TC_Driver_ProjectClassFactory.h
        TC_Driver_ProjectInterfaces.h
        TC_Driver_ProjectServices.h
        TcPch.h
      Source Files
        MyFirstTrafo.cpp
        TC_Driver_Project.rc
        TC_Driver_ProjectClassFactory.cpp
        TcPch.cpp
      TMC Files
        TC_Driver_Project.tmc
      TwinCAT RT Files
      TwinCAT UM Files
```

5. Add the `ITcNcTrafo` interface.



Shows the implemented interfaces of the module.

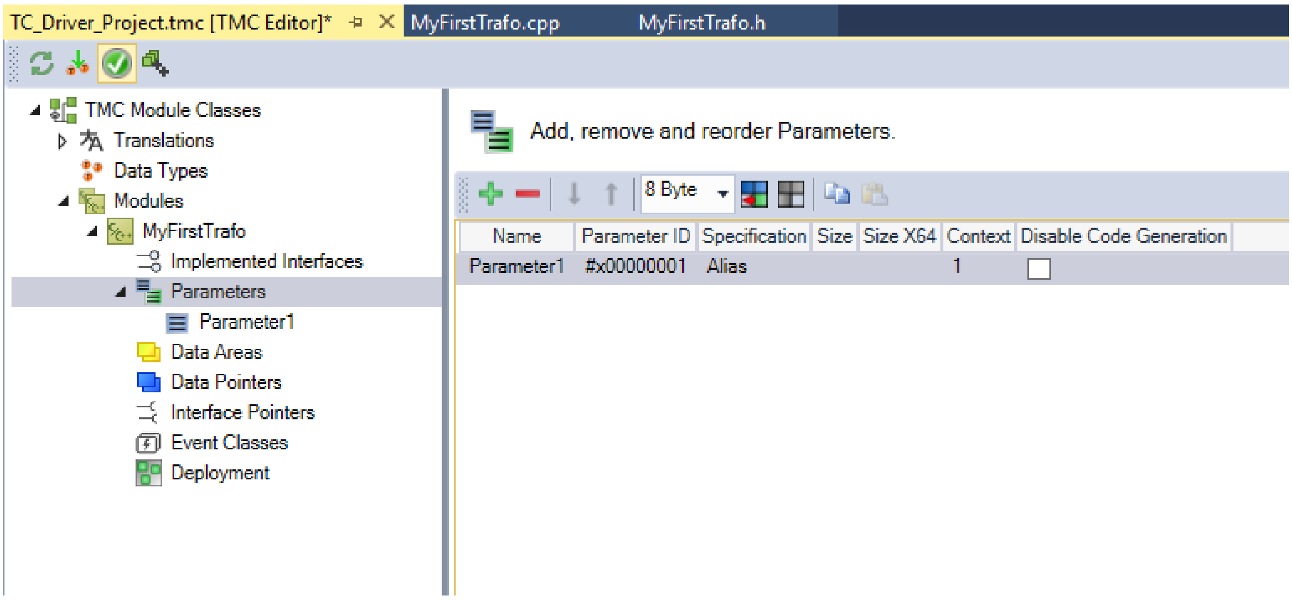
Name	Interface ID	ContextId	Disable Code Generation
ITComObject	{00000012-0000-0000-E000-000000000064}		☑
ITcADI	{03000012-0000-0000-E000-000000000064}		☑
ITcWatchSource	{03000018-0000-0000-E000-000000000064}		☑

Name	Namespace	Guid	Specification	Size
ITcJsonSaxHandler		{2cdc7d30-5f22-429f-9065-d912d8421148}	Interface	4.0 (8.0)
ITcJsonSaxPrettyWriter		{196bb18e-9d99-40ce-b81e-48992ce43217}	Interface	4.0 (8.0)
ITcJsonSaxReader		{9efcaaa8-556f-4b69-ab9f-2967b7f9dfc6}	Interface	4.0 (8.0)
ITcJsonSaxValues		{81e405d9-9faf-4c28-9985-185ac023a4c2}	Interface	4.0 (8.0)
ITcJsonSaxWriter		{e695c12a-2d9a-408e-b9b2-508d7ce3af9a}	Interface	4.0 (8.0)
ITcJsonSaxWriter2		{472183f9-5d09-4d8c-92fd-e0524ef658bf}	Interface	4.0 (8.0)
ITcMessage		{6474ed2c-e483-454e-a67d-233e6d337c08}	Interface	4.0 (8.0)
ITcMessageListener		{47b6bee8-0ecb-4c92-9d93-fb11d3ba0336}	Interface	4.0 (8.0)
ITcNcDcConvert		{05000005-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcNcDcConvert2		{05000006-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcNcTrafo		{05010001-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComCreateInstance		{00000031-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComLicenseServer		{01010001-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComNoPlcWrapper		{00000063-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComObjCon		{00000016-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComObjectCategory		{00000038-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComObjectServer		{00000030-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComObjInd		{00000013-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComObjReq		{00000015-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITComObjRes		{00000014-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcPostCyclic		{03000025-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcPostCyclicCaller		{03000026-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcPreInputCyclic		{03000017-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcRtAdsPort		{00000059-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcRtAdsUser		{0000005a-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcRTime		{02000010-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcRTimeTask		{02000003-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcSourceInfo		{f7bf6767-548b-493c-899b-06a477976f11}	Interface	4.0 (8.0)
ITcTask		{02000002-0000-0000-e000-000000000064}	Interface	4.0 (8.0)
ITcTaskNotification		{9cde7c78-32a0-4375-827e-924b31021fcd}	Interface	4.0 (8.0)
ITcVnBidirectionalIterator		{f6c9d79d-2e9c-40fa-9865-3d23fac6860c}	Interface	4.0 (8.0)

□ Show hidden data types

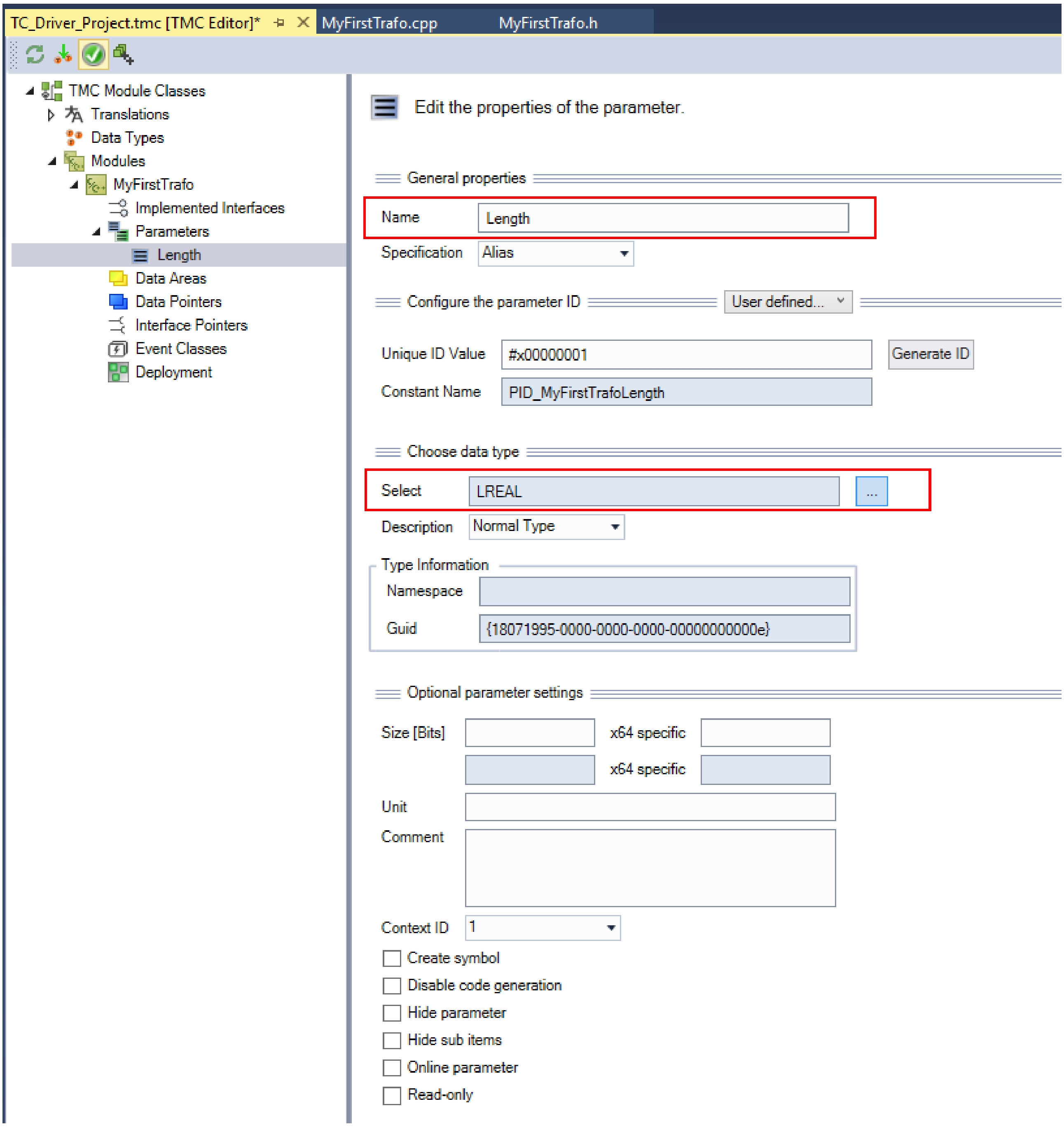
Group by: None

6. Add kinematic specific parameters (arm length, displacements, etc.).



Add, remove and reorder Parameters.

Name	Parameter ID	Specification	Size	Size X64	Context	Disable Code Generation
Parameter1	#x00000001	Alias			1	□



Edit the properties of the parameter.

General properties

Name: Length

Specification: Alias

Configure the parameter ID: User defined...

Unique ID Value: #x00000001 [Generate ID]

Constant Name: PID_MyFirstTrafoLength

Choose data type

Select: LREAL [...]

Description: Normal Type

Type Information

Namespace:

Guid: {18071995-0000-0000-0000-00000000000e}

Optional parameter settings

Size [Bits] | x64 specific

x64 specific

Unit

Comment

Context ID: 1

□ Create symbol

□ Disable code generation

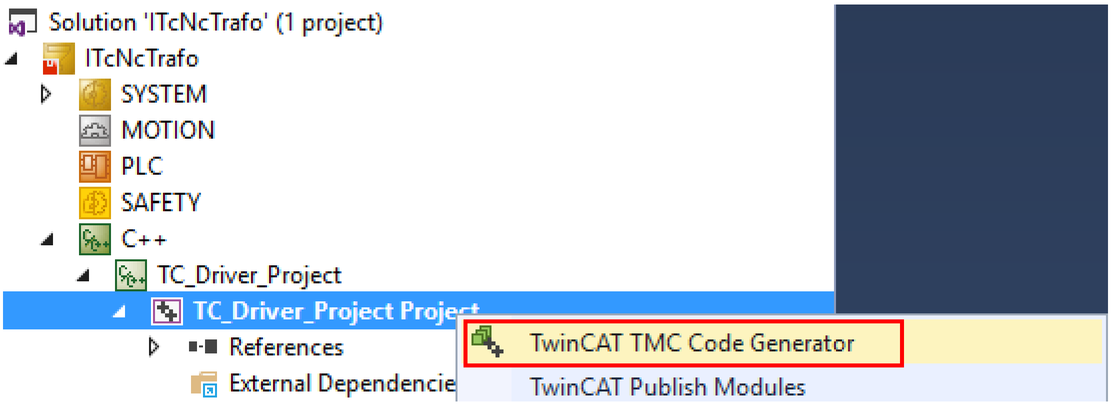
□ Hide parameter

□ Hide sub items

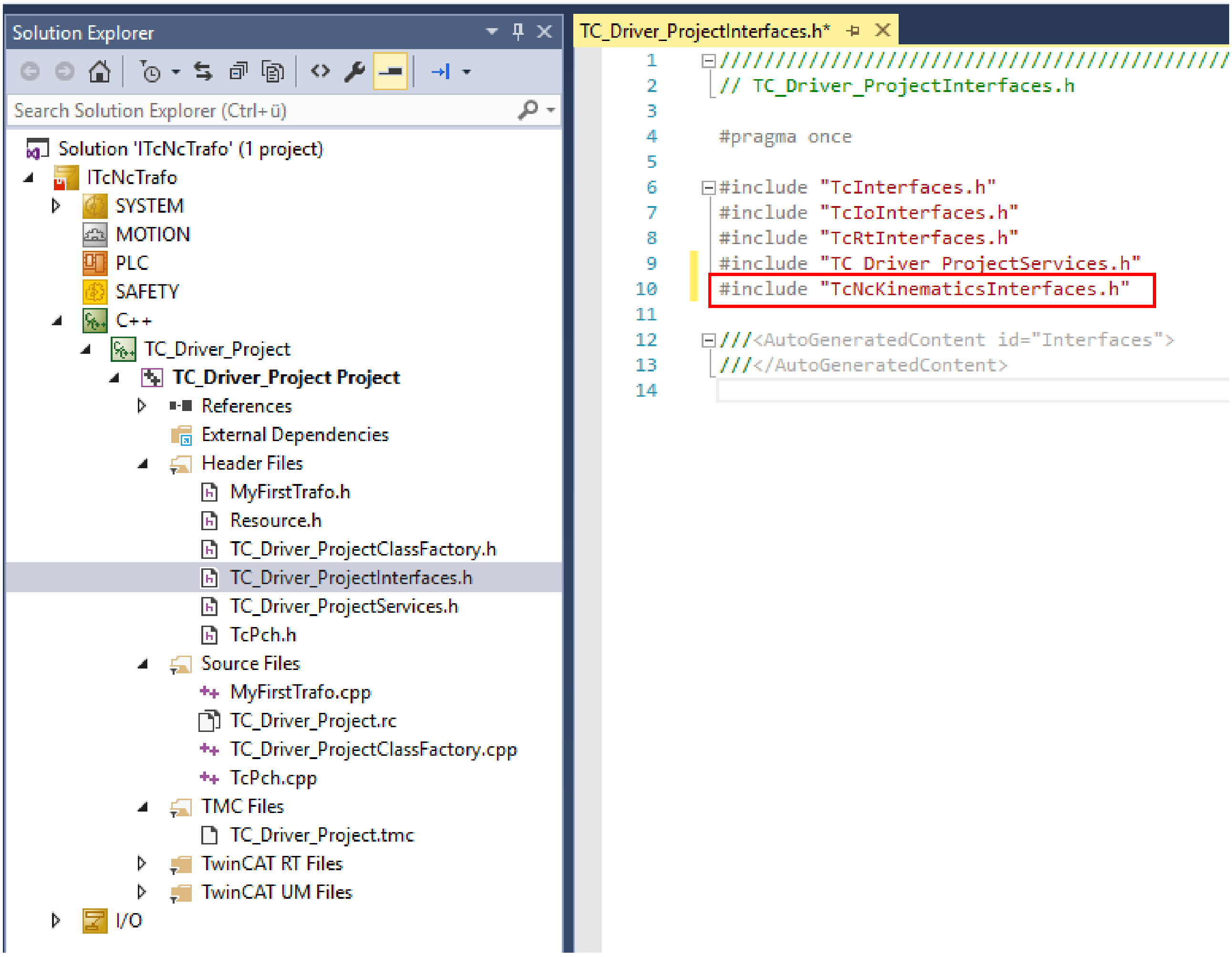
□ Online parameter

□ Read-only

7. Run the TMC Code Generator.



8. Add a header. `<ProjectName>Interfaces.h` requires an include statement, additionally. Also add `TcNcKinematicsInterfaces.h`.



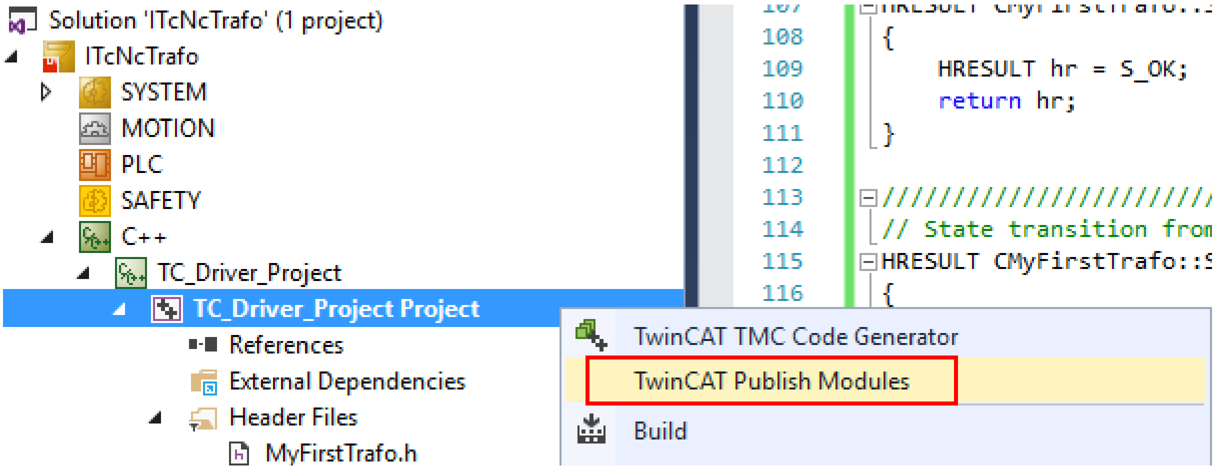
```
//////////////////////////////////////////////
// TC_Driver_ProjectInterfaces.h

#pragma once

#include "TcInterfaces.h"
#include "TcIoInterfaces.h"
#include "TcRtInterfaces.h"
#include "TC_Driver_ProjectServices.h"
#include "TcNcKinematicsInterfaces.h"

///<AutoGeneratedContent id="Interfaces">
///</AutoGeneratedContent>
```

9. Implement the methods `Forward`, `Backward`, `TrafoSupported` and `GetDimensions`. These methods have to have a valid implementation and are automatically placed at `Source Files\<TrafoName>.cpp`.

E_NOTIMPL

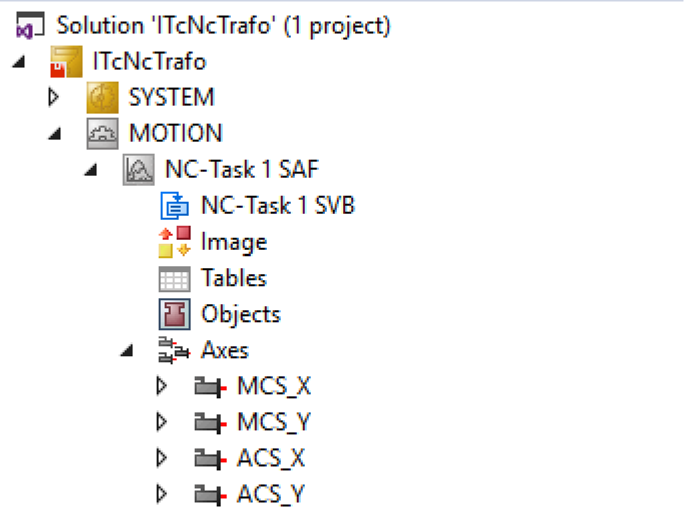
The default return value `E_NOTIMPL` of the methods `Forward`, `Backward`, `TrafoSupported` and `GetDimensions` using the kinematics leads to error messages and really has to be overwritten.

10. Build and publish the modules.

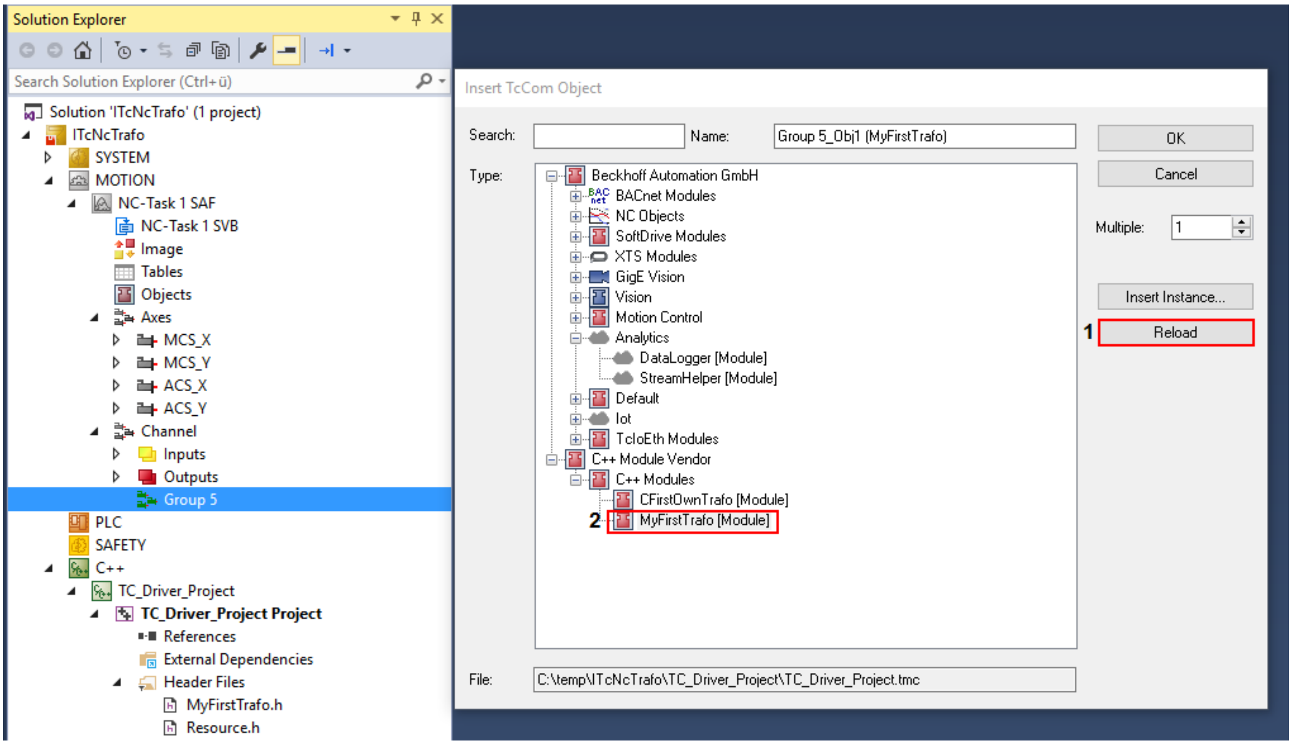


```
HRESULT CMyFirstTrafo::
{
    HRESULT hr = S_OK;
    return hr;
}

///////////////////////
// State transition from
HRESULT CMyFirstTrafo::S
{
```

11. Configure axes in the `MOTION` subtree and add PTP axes.



12. Add user specific modules (Channel).
13. Reload the `TcCom` objects by pressing the `Reload` button.
14. Select your transformation object and confirm your choice by pressing the `OK` button.

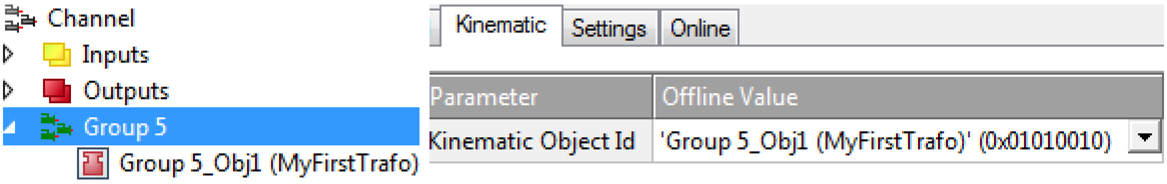


Name: Group 5_Obj1 (MyFirstTrafo)

Multiple: 1

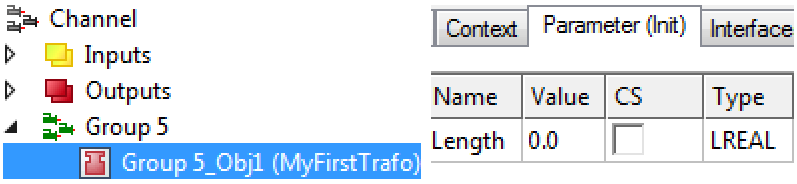
File: C:\temp\ITcNcTrafo\TC_Driver_Project\TC_Driver_Project.tmc

15. The transformation group has to know which root module is to be called. This is why the object ID of the kinematics (in this case `MyFirstTrafo`) has to be selected. The kinematic object defines the number of ACS axes and MCS axes to be used in the PLC (see ST_KinAxes [▶ 81]).



Parameter	Offline Value
Kinematic Object Id	'Group 5_Obj1 (MyFirstTrafo)' (0x01010010)

16. Parameterize the object parameters according to the kinematics used. Once this is done, the XAE configuration is complete.



Name	Value	CS	Type
Length	0.0	□	LREAL

⇨ The transformation can now be activated via the PLC (see PLC Library). To actuate the transformation define a cyclic channel interface in the PLC and link it with the IO of the kinematic channel.

```
in_stKinToPlc       AT %I*      : NCTOPLC_NCICHANNEL_REF;
out_stPlcToKin      AT %Q*      : PLCTONC_NCICHANNEL_REF;
```

Customer Specific Kinematic

The function block `FB_KinCalcTrafo` cannot be employed for transformation that have been set up on one’s own.

8 Plc Library

Function block	Description
Kinematic Transformation	
FB_KinConfigGroup [▶ 66]	Configures ACS and MCS axes according to the kinematic transformation group and enables cartesian mode or joint mode (ACS).
FB_KinResetGroup [▶ 68]	Resets the kinematic transformation group.
F_KinGetChnOperationState [▶ 79]	Reads the status of the kinematic transformation group cyclically.
F_KinGetAcsMcsAxisIds [▶ 79]	Reads the active ACS and MCS axes of the kinematic group.
Transformation calculation	
FB_KinCalcTrafo [▶ 69]	Calculates the Kinematic Transformation without link to the axes.
FB_KinCalcMultiTrafo [▶ 71]	Calculates the Kinematic Transformation for several positions.
Edit parameters and coordinate systems online	
FB_KinLockTrafoParam [▶ 75]	Locks the parameters of the kinematic transformation group, denies write access.
FB_KinUnlockTrafoParam [▶ 73]	Unlocks the parameters of the kinematic transformation group, enables write access.
Extended rotation range	
FB_KinExtendedRotationRange [▶ 76]	Saves and restores the rotational state of the kinematic group.
FB_KinPresetRotation [▶ 78]	Sets the rotational state.

Structures and enumerations

Name	Description
ST_KinAxes [▶ 81]	Structure of the ACS and MCS axes, which form the kinematics
E_KinStatus [▶ 81]	Status of the kinematic group (enum)

Development environment	Target system	PLC libraries to include
TwinCAT 3	PC or CX (x86, x64)	Tc2_NcKinematicTransformation

Function blocks for compatibility with existing programs

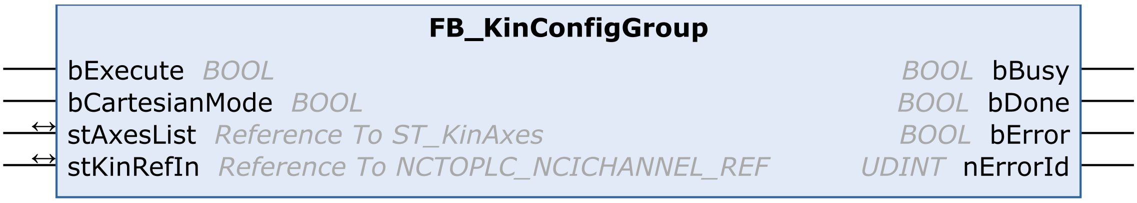
Function blocks for compatibility

The purpose of the function blocks listed is to ensure compatibility with existing projects. It is not advisable to use these function blocks for new projects. Instead, the equivalent function blocks shown in the table above should be used.

Function block	Description
FB_KinCheckActualStatus [▶ 83]	Reads the status of the kinematic transformation group acyclically

8.1 Function Blocks

8.1.1 FB_KinConfigGroup



The function block FB_KinConfigGroup configures axes according to the kinematic transformation. These are axes for the ACS (joint) and the MCS (cartesian). The function block takes the ACS and MCS axes defined in the **stAxesList** and configures them in the kinematic group of **stKinRefIn**.

VAR_INPUT

```
VAR_INPUT
    bExecute            : BOOL;
    bCartesianMode      : BOOL;
END_VAR
```

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

bCartesianMode: If FALSE, the ACS axes (joint) can be moved directly. If TRUE, the movement described in the MCS axes (cartesian) is transformed into a movement of the ACS axes (joint). The ACS axes cannot be moved directly.

VAR_IN_OUT

```
VAR_IN_OUT
    stAxesList          : ST_KinAxes;
    stKinRefIn          : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

stAxesList: Determines the ACS and MCS axes included in the configuration. See ST_KinAxes.

stKinRefIn: Determines the kinematic group of the configuration.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy               : BOOL;
    bDone               : BOOL;
    bError              : BOOL;
    nErrorId            : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

bDone: The output becomes TRUE when the command was executed successfully.

bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

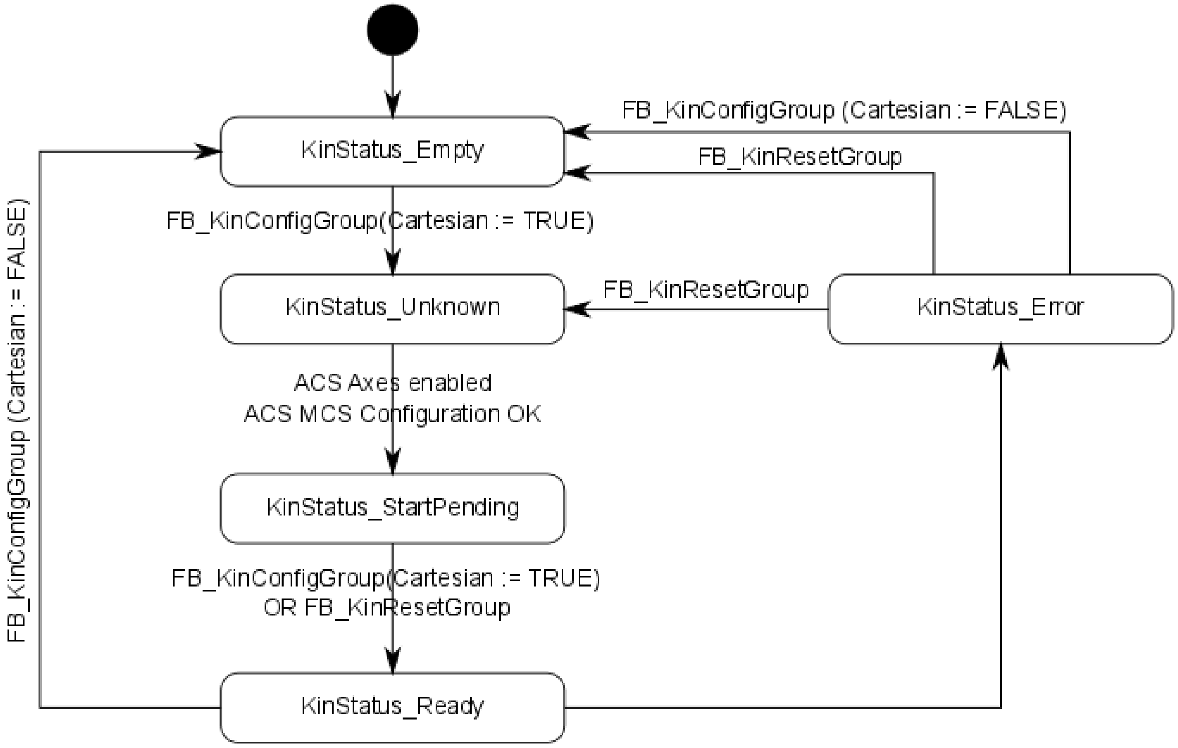
nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

```
VAR
    io_X                : AXIS_REF;
    io_Y                : AXIS_REF;
    io_Z                : AXIS_REF;
    io_M1               : AXIS_REF;
    io_M2                   : AXIS_REF;
    io_M3                   : AXIS_REF;
    in_stKinToPlc AT %I*    : NCTOPLC_NCICHANNEL_REF;
    fbConfigKinGroup        : FB_KinConfigGroup;
    stAxesConfig            : ST_KinAxes;
    bAllAxesReady           : BOOL;
    bExecuteConfigKinGroup: BOOL;
    bUserConfigKinGroup     : BOOL;
    bUserCartesianMode      : BOOL := TRUE;
    (*true: cartesian mode - false: direct mode (without transformation) *)
END_VAR
```

```
(* read the IDs from the cyclic axis interface so the axes can mapped later to the kinematic group
*)
stAxesConfig.nAxisIdsAcs[1] := io_M1.NcToPlc.AxisId;
stAxesConfig.nAxisIdsAcs[2] := io_M2.NcToPlc.AxisId;
stAxesConfig.nAxisIdsAcs[3] := io_M3.NcToPlc.AxisId;
stAxesConfig.nAxisIdsMcs[1] := io_X.NcToPlc.AxisId;
stAxesConfig.nAxisIdsMcs[2] := io_Y.NcToPlc.AxisId;
stAxesConfig.nAxisIdsMcs[3] := io_Z.NcToPlc.AxisId;

IF bAllAxesReady AND bUserConfigKinGroup THEN
    bExecuteConfigKinGroup := TRUE;
ELSE
    bExecuteConfigKinGroup := FALSE;
END_IF

fbConfigKinGroup(
    bExecute        := bExecuteConfigKinGroup ,
    bCartesianMode := bUserCartesianMode ,
    stAxesList      := stAxesConfig,
    stKinRefIn      := in_stKinToPlc );
```

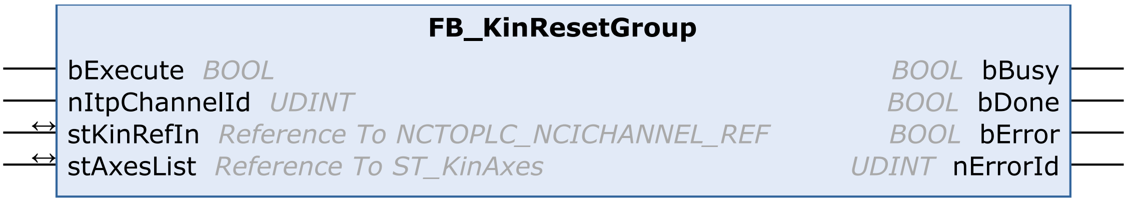
State of the kinematic group



Enable configuration

The ACS axes must be enabled through MC_Power, to ensure that the state can reach the value **KinStatus_Ready**. If the ACS axes are not enabled, enable the axes and then call up FB_KinConfigGroup or FB_KinResetGroup.

8.1.2 FB_KinResetGroup



The function block FB_KinResetGroup resets the kinematic group. All ACS and MCS axes are reset. In addition, the input *nItpChannelId* can be used for specifying the corresponding interpolation channel. The channel is reset, if the nItpChannelId is not 0.

When all axes are enabled and the group was in cartesian mode, the group returns to state KinStatus_Ready. If the group was not in cartesian mode, the group returns to state KinStatus_Empty. If the axes are not enabled, the group remains in state KinStatus_Empty.

VAR_INPUT

```
VAR_INPUT
    bExecute      : BOOL;
    nItpChannelId : UDINT;
END_VAR
```

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

nItpChannelId: ID of the corresponding interpolation channel. If the input us not 0, the corresponding interpolation channel is reset.

VAR_IN_OUT

```
VAR_IN_OUT
    stAxesList              : ST_KinAxes;
    stKinRefIn              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

stAxesList: Determines the ACS and MCS axes included in the configuration. See ST_KinAxes.

stKinRefIn: Determines the kinematic group of the configuration.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                   : BOOL;
    bDone                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

bDone: The output becomes TRUE when the command was executed successfully.

bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

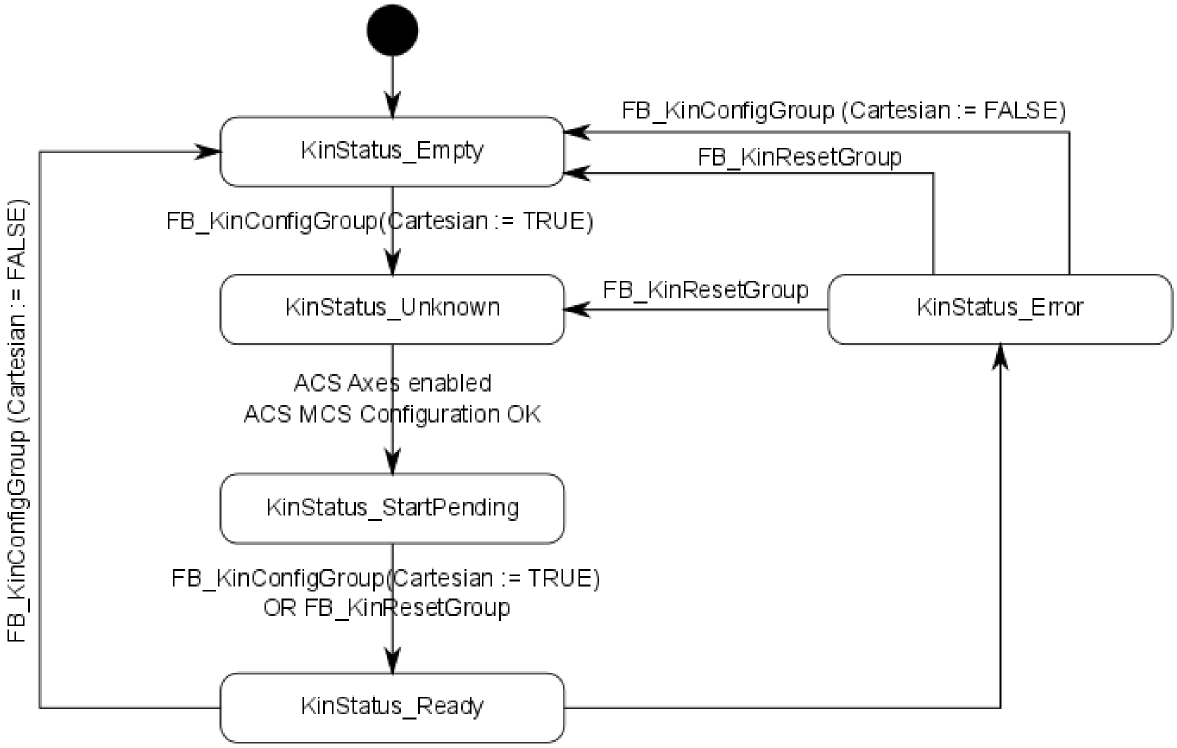
nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

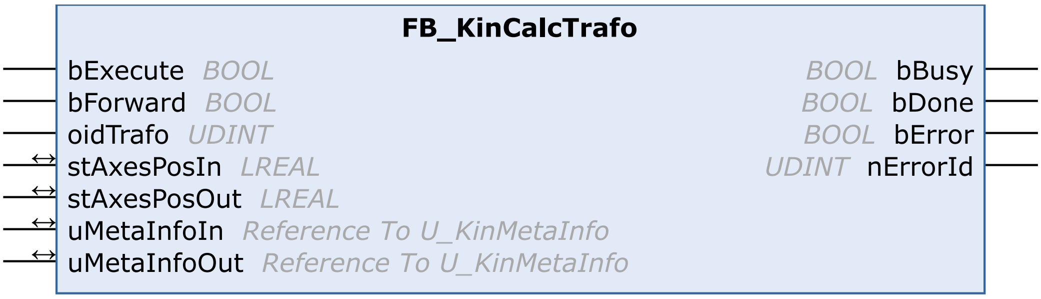
```
VAR
    fbFB_ResetKinGroup   : FB_KinResetGroup;
    stAxesConfig         : stAxesConfig;
    in_stKinToPlc AT %I* : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
fbFB_ResetKinGroup(
    bExecute := TRUE,
    nItpChannelId := 3,
    stKinRefIn := in_stKinToPlc,
    stAxesList := stAxesConfig,
    bBusy=> ,
    bDone=> ,
    bError=> ,
    nErrorId=> );
```

State of the kinematic group



8.1.3 FB_KinCalcTrafo



The function block FB_KinCalcTrafo calculates the forward or backward transformation, even if no kinematic group was created with FB_KinConfigGroup [▸ 66].

VAR_INPUT

```
VAR_INPUT
    bExecute              : BOOL;
    bForward              : BOOL;
    oidTrafo              : UDINT;
END_VAR
```

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

bForward: Determines whether the forward or backward transformation is calculated.

oidTrafo: Object-ID of the kinematic transformation object to be calculated. See example [▸ 70] below.

VAR_IN_OUT

```
VAR_IN_OUT
    stAxesPosIn             : ARRAY[1..8] OF LREAL;
    stAxesPosOut            : ARRAY[1..8] OF LREAL;
    uMetaInfoIn             : U_KinMetaInfo;
    uMetaInfoOut            : U_KinMetaInfo;
END_VAR
```

stAxesPosIn: Array containing the input positions of the transformation. For the calculation of a forward transformation they represent the joint positions. For the calculation of a backward transformation they represent the cartesian axis positions.

stAxesPosOut: Array containing the result positions of the transformation. For the calculation of a forward transformation they represent the cartesian axis positions. For the calculation of a backward transformation they represent the joint positions.

uMetaInfoIn: In cases where different robot configurations lead to a solution, the preferred solution can be selected (see sample [▸ 70]). For kinematics in which this parameter is not required, a dummy variable can be assigned to this input.

uMetaInfoOut: If different solutions are possible for a transformation, the solution that was found is specified. For kinematics in which this parameter is not required, a dummy variable can be assigned to this input.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                   : BOOL;
    bDone                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

bDone: The output becomes TRUE when the command was executed successfully.

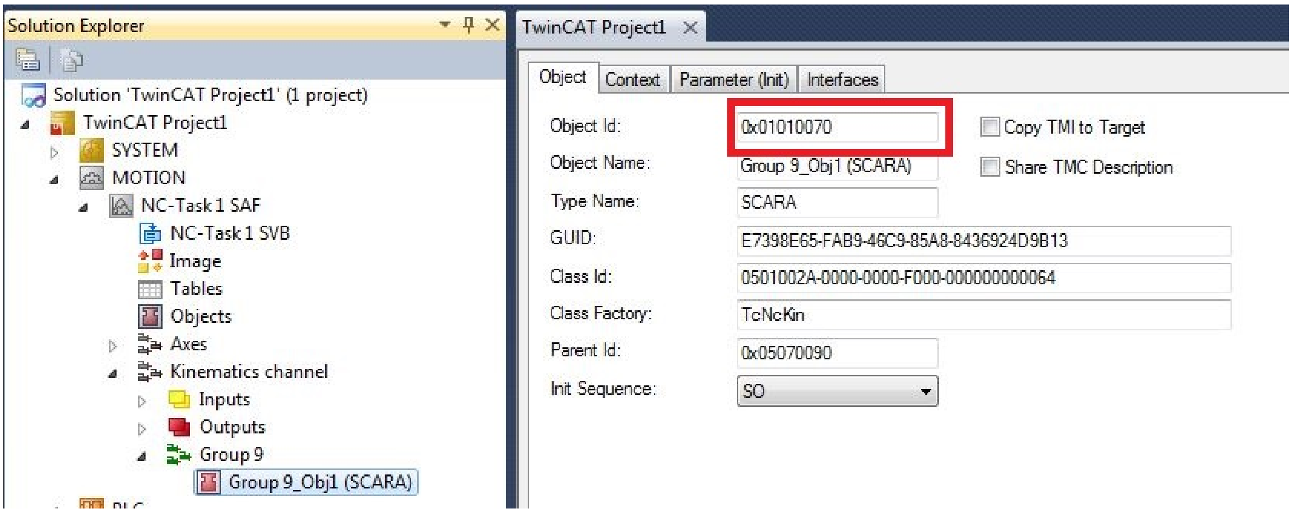
bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

The object ID of the transformation is shown in the transformation object under the kinematic channel.

SCARA transformation [▸ 40] - sample object ID



Object Id: 0x01010070

□ Copy TMI to Target

Object Name: Group 9_Obj1 (SCARA)

□ Share TMC Description

Type Name: SCARA

GUID: E7398E65-FAB9-46C9-85A8-8436924D9B13

Class Id: 0501002A-0000-0000-F000-000000000064

Class Factory: TcNcKin

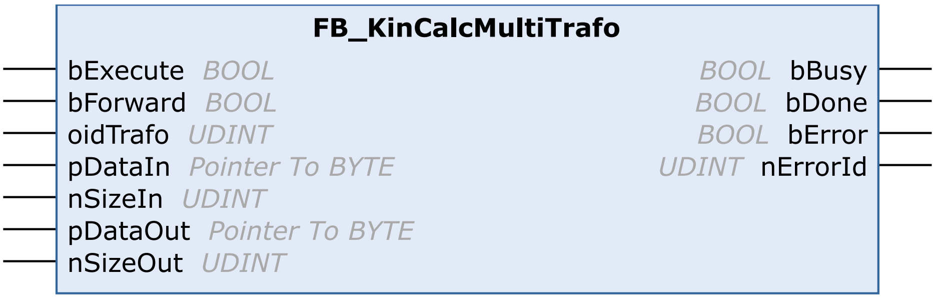
Parent Id: 0x05070090

Init Sequence: SO

```
VAR
    fbKinCalcTrafo      : FB_KinCalcTrafo;
    stAxesPosIn         : ARRAY[1..8] OF LREAL;
    stAxesPosOut        : ARRAY[1..8] OF LREAL;
    bUserExecute        : BOOL;
    bUserCalcFwdTrafo   : BOOL;
    uScaraMetaInfoIn : U_KinMetaInfo;
    uScaraMetaInfoOut : U_KinMetaInfo;
END_VAR
```

```
uScaraMetaInfoIn.eScara := E_KinMetaInfoScara.scaraLeftArm;

fbKinCalcTrafo(
    bExecute := bUserExecute,
    bForward := bUserCalcFwdTrafo,
    oidTrafo := 16#01010070,
    stAxesPosIn := stAxesPosIn,
    stAxesPosOut := stAxesPosOut,
    uMetaInfoIn:= uScaraMetaInfoIn ,
    uMetaInfoOut:= uScaraMetaInfoOut,
    bBusy=> ,
    bDone=> ,
    bError=> ,
    nErrorId=> );
```

8.1.4 FB_KinCalcMultiTrafo



The function block FB_KinCalcMultiTrafo calculates the forward or backward transformation for several positions, even if no kinematic group was created with FB_KinConfigGroup [▶ 66].

Alternatively, the function block FB_KinCalcTrafo [▶ 69] can be used to calculate the kinematic transformations individually.

Inputs

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

Name	Type	Description
bForward	BOOL	Determines whether the forward or backward transformation is calculated.
oidTrafo	UDINT	Object ID of the kinematic transformation object to be calculated.
pDataIn	POINTER TO BYTE	Pointer to the input data, consisting of an instance of ST_KinMultiTrafoHeader [▶ 83] and an array of input positions. For the calculation of a forward transformation they represent the joint positions. For the calculation of a backward transformation they represent the Cartesian axis positions.
nSizeIn	UDINT	Size of the input data to which pDataIn points
pDataOut	POINTER TO BYTE	Pointer to the output data.
nSizeOut	UDINT	Size of the output data to which pDataOut points.

Outputs

Name	Type	Description
bBusy	BOOL	The output becomes TRUE when the command is started with bExecute and remains TRUE as long as the function block executes the command. While bBusy is TRUE, no new command is accepted at the inputs. If bBusy becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs bDone or bError is set.
bDone	BOOL	The output becomes TRUE when the command was executed successfully.
bError	BOOL	The output bError is set to TRUE, if an error occurred during the command execution.
nErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

ST_KinCalcMultiTrafoIn

```
TYPE ST_KinCalcMultiTrafoIn :
STRUCT
hdr   : ST_KinMultiTrafoHeader;
fPos : ARRAY[1..2] OF ARRAY[1..4] OF LREAL;
END_STRUCT
END_TYPE
```

ST_KinCalcMultiTrafoOut

```
TYPE ST_KinCalcMultiTrafoOut :
STRUCT
    fPos      : ARRAY[1..2] OF ARRAY[1..4] OF LREAL;
    fMetaInfo : ARRAY[1..2] OF U_KinMetaInfo;
END_STRUCT
END_TYPE
```

MAIN

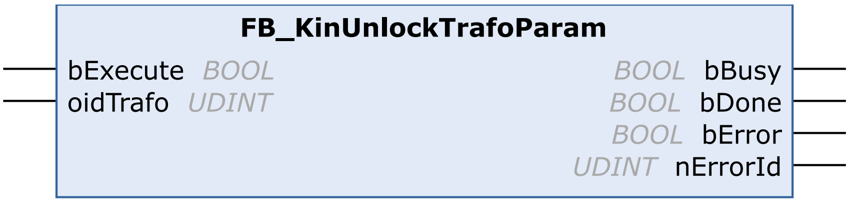
```
PROGRAM MAIN
VAR
    {attribute 'TcInitSymbol'} oidKinematic: OTCID;
    nState: UDINT := 0;

    fbKinCalcMultiTrafo   : FB_KinCalcMultiTrafo;
    stKinCalcMultiIn      : ST_KinCalcMultiTrafoIn;
    stKinCalcMultiOut     : ST_KinCalcMultiTrafoOut;
END_VAR
```

```
CASE nState OF
0:
    // Header for Multi Trafo
    stKinCalcMultiIn.hdr.nColumnsIn := 4;
    stKinCalcMultiIn.hdr.nColumnsOut := 4;
    stKinCalcMultiIn.hdr.nLines := 2;
    stKinCalcMultiIn.hdr.uMetaInfo.eScara := E_KinMetaInfoScara.scaraLeftArm;
    stKinCalcMultiIn.hdr.bGetMetaInfo := TRUE;

    // Positions
    stKinCalcMultiIn[1][1]:=0;
    stKinCalcMultiIn[1][2]:=90;
    stKinCalcMultiIn[1][3]:=0;
    stKinCalcMultiIn[1][4]:=0;

    stKinCalcMultiIn[2][1]:=0;
    stKinCalcMultiIn[2][2]:=-90;
    stKinCalcMultiIn[2][3]:=0;
    stKinCalcMultiIn[2][4]:=0;

    nState := nState + 10;

10:
    fbKinCalcMultiTrafo( bExecute := TRUE,
                         bForward := TRUE,
                         oidTrafo := oidKinematic,
                         pDataIn  := ADR(stKinCalcMultiIn),
                         nSizeIn  := SIZEOF(stKinCalcMultiIn),
                         pDataOut := ADR(stKinCalcMultiOut),
                         nSizeOut := SIZEOF(stKinCalcMultiOut) );

    IF NOT fbKinCalcMultiTrafo.bBusy THEN
        fbKinCalcMultiTrafo(bExecute:= FALSE, bForward:= TRUE, oidTrafo:= oidKinematic,
                            pDataIn:=ADR(stKinCalcMultiIn), nSizeIn:= SIZEOF(stKinCalcMultiIn),
                            pDataOut:=ADR(stKinCalcMultiOut), nSizeOut:=
SIZEOF(stKinCalcMultiOut) );

        nState := nState + 10;
    END_IF
END_CASE
```

Requirements

Development environment	Target system type	PLC libraries to include
Advanced Motion Pack V3.1.10.51	PC or CX (x64)	Tc2_NcKinematicTransformation

8.1.5 FB_KinUnlockTrafoParam



The function block FB_KinUnlockTrafoParam unlocks transformation parameters that have an influence on the position so that they can be written.

Once the kinematics parameters have been unlocked, the PLC has write access via ADSWRITE. The required index group is the object ID and the index offset is the parameter ID. The written parameters are not persistent. Parameters that have no influence on the position (e.g. torques and masses) can be written without calling FB_KinUnlockTrafoParam.

⚠ **CAUTION**

Changing the parameters can lead to discontinuities.

Please note that utmost caution is required. Redefinition of kinematic parameters can lead to position set-point step changes in the kinematic chain.

After kinematic parameters have been written, writing with FB_LockTrafoParam can be locked again.

VAR_INPUT

```
VAR_INPUT
    bExecute                : BOOL;
    oidTrafo                : UDINT;
END_VAR
```

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

oidTrafo: Object ID of the kinematic transformation object. See sample [▶ 74] below.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                   : BOOL;
    bDone                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

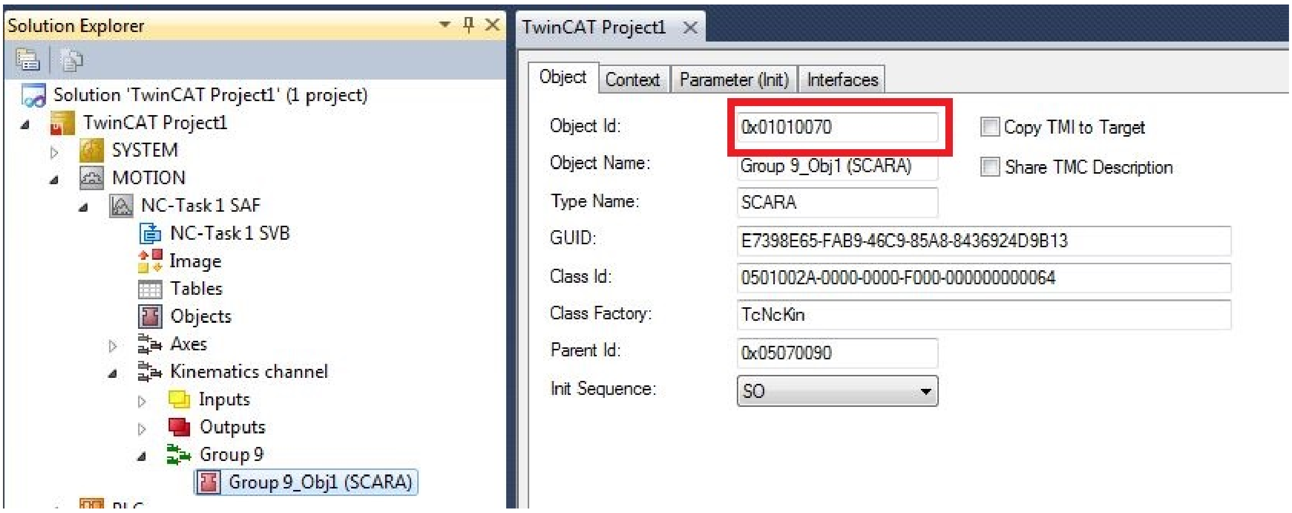
bDone: The output becomes TRUE when the command was executed successfully.

bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

The object ID and parameter ID required for enabling a transformation parameter and for writing a corresponding new value can be read from the transformation object in the XAE.



Object Id: 0x01010070

□ Copy TMI to Target

Object Name: Group 9_Obj1 (SCARA)

□ Share TMC Description

Type Name: SCARA

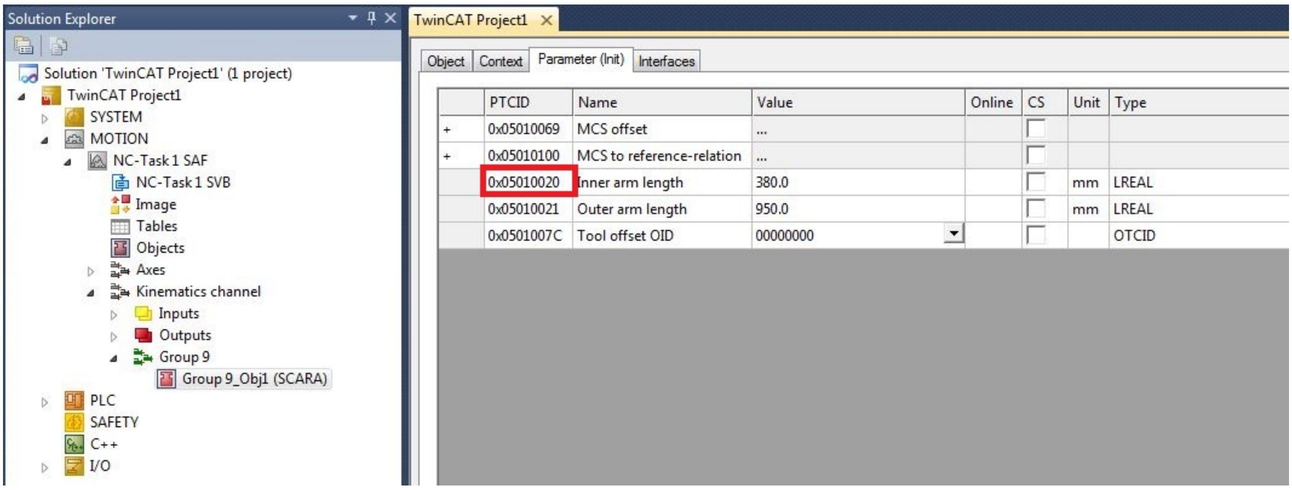
GUID: E7398E65-FAB9-46C9-85A8-8436924D9B13

Class Id: 0501002A-0000-0000-F000-000000000064

Class Factory: TcNcKin

Parent Id: 0x05070090

Init Sequence: SO

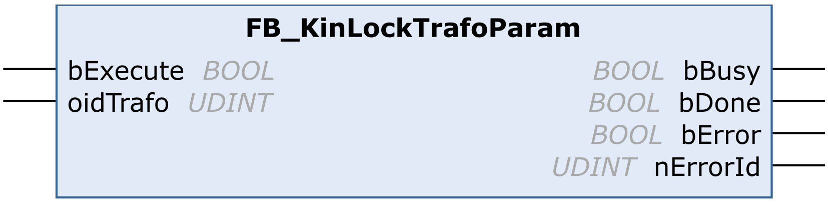


	PTCID	Name	Value	Online	CS	Unit	Type
+	0x05010069	MCS offset	...		□		
+	0x05010100	MCS to reference-relation	...		□		
	0x05010020	Inner arm length	380.0		□	mm	LREAL
	0x05010021	Outer arm length	950.0		□	mm	LREAL
	0x0501007C	Tool offset OID	00000000		□		OTCID

```
VAR
    bUserExecuteUnlock      : BOOL;
    fbFB_UnlockTrafoParam   : FB_KinUnlockTrafoParam;
    bUserExecuteWriteParam  : BOOL;
    fbADSWRITE              : ADSWRITE;
    oidTrafo                : UDINT := 16#01010170; (*Trafo object id*)
    pidTrafo                : UDINT := 16#05010020; (*parameter id*)
    fParamValue             : LREAL;
END_VAR
```

```
fbFB_UnlockTrafoParam(
    bExecute := bUserExecuteUnlock,
    oidTrafo := oidTrafo,
    bBusy=>,
    bDone=>,
    bError=>,
    nErrorId=> );

(*After unlocking new parameter value can be written*)
fbADSWRITE(
    NETID:='' ,
    PORT:= AMSPORT_R0_NCSAF,
    IDXGRP:=oidTrafo ,
    IDXOFFS:= pidTrafo,
    LEN:=SIZEOF(fParamValue) ,
    SRCADDR:= ADR(fParamValue),
    WRITE:=bUserExecuteWriteParam ,
    TMOUT:= ,
    BUSY=> ,
    ERR=> ,
    ERRID=> );
```

8.1.6 FB_KinLockTrafoParam



Once the transformation parameters have been modified with the aid of FB_KinUnlockTrafoParam [▶ 73], the function block FB_KinLockTrafoParam locks the transformation parameters again, so that write access is no longer possible.

VAR_INPUT

```
VAR_INPUT
    bExecute               : BOOL;
    oidTrafo               : UDINT;
END_VAR
```

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

oidTrafo: Object-ID of the kinematic transformation object. See example [▸ 76] below.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                  : BOOL;
    bDone                  : BOOL;
    bError                 : BOOL;
    nErrorId               : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

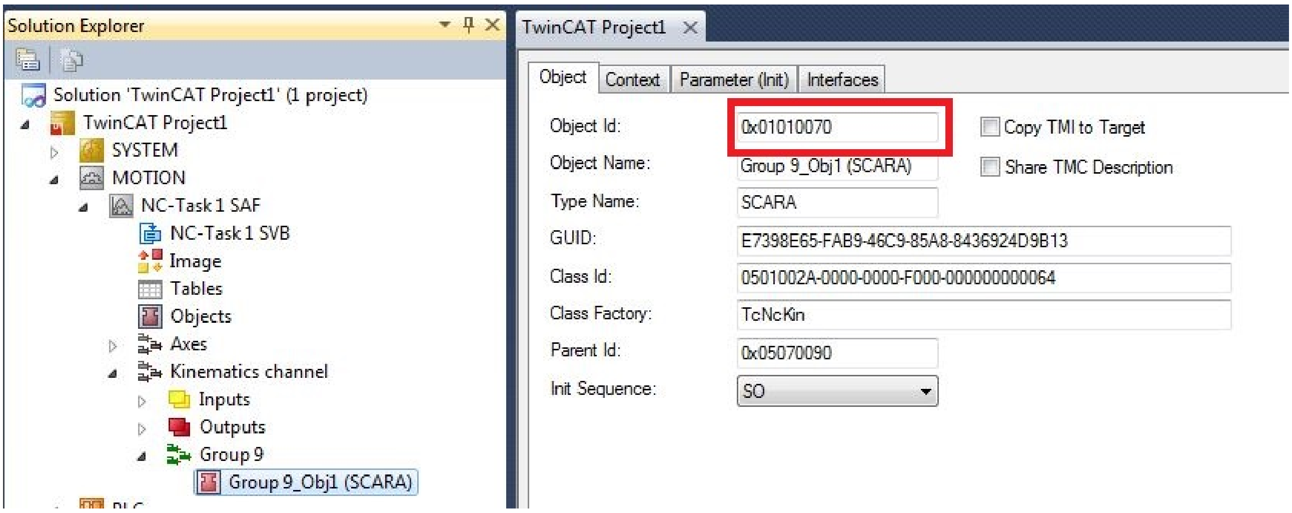
bDone: The output becomes TRUE when the command was executed successfully.

bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

SCARA transformation - sample object ID



Solution Explorer

- Solution 'TwinCAT Project1' (1 project)
  - TwinCAT Project1
    - SYSTEM
    - MOTION
      - NC-Task 1 SAF
        - NC-Task 1 SVB
        - Image
        - Tables
        - Objects
        - Axes
        - Kinematics channel
          - Inputs
          - Outputs
          - Group 9
            - Group 9_Obj1 (SCARA)

TwinCAT Project1

Object | Context | Parameter (Init) | Interfaces

Object Id: 0x01010070

Object Name: Group 9_Obj1 (SCARA)

Type Name: SCARA

GUID: E7398E65-FAB9-46C9-85A8-8436924D9B13

Class Id: 0501002A-0000-0000-F000-000000000064

Class Factory: TcNcKin

Parent Id: 0x05070090

Init Sequence: SO

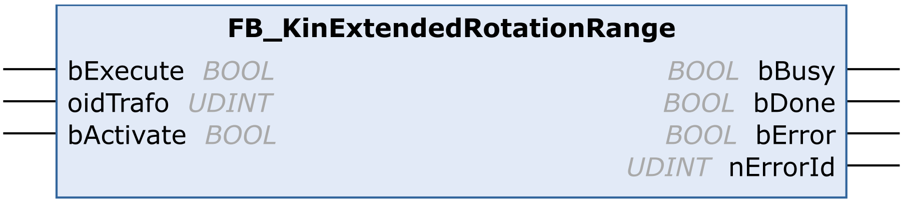
□ Copy TMI to Target

□ Share TMC Description

```
VAR
    bUserExecute          : BOOL;
    fbFB_LockTrafoParam   : FB_KinLockTrafoParam;
    oidTrafo              : UDINT := 16#01010070; (*Trafo object id*)
END_VAR
```

```
fbFB_LockTrafoParam(
    bExecute := bUserExecute,
    oidTrafo := oidTrafo,
    bBusy=>,
    bDone=>,
    bError=>,
    nErrorId=> );
```

8.1.7 FB_KinExtendedRotationRange



Extended rotation range

✓ For a unique solution the standard rotation range is limited to:

a) Rotation1: -180 to 180 degrees,

b) Rotation2: -90 to 90 degrees,

c) Rotation3: -180 to 180 degrees.

⇨ In some 6-axis applications it is desirable to be able to rotate beyond this rotation range. The function blocks FB_KinExtendedRotationRange and FB_KinPresetRotation enable the rotational state to be extended, saved and restored beyond the default values.

The function block FB_KinExtendedRotationRange saves and restores the rotational state of the kinematic group.

If the function block is executed with bActivate:=TRUE, the rotational state is saved until the kinematic group is resolved. If the kinematic group is subsequently built or reset, the saved rotational state is restored. If the rotation deviates significantly (>10.0 degrees per axis), the saved rotational state is not restored and FB_KinConfigGroup or FB_KinResetGroup fail with error 0x815D.

If the function block is executed with bActivate:=FALSE, the rotational state is not saved or restored (default behavior).

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    oidTrafo       : UDINT;
    bActivate      : Bool;
END_VAR
```

bExecute: The command is executed with a rising edge.

oidTrafo: Object ID (OTCID) of the kinematic transformation object.

bActivate: If set to TRUE, the extended rotation range is activated.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                 : BOOL;
    bDone                 : BOOL;
    bError                : BOOL;
    nErrorId              : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

bDone: The output becomes TRUE when the command was executed successfully.

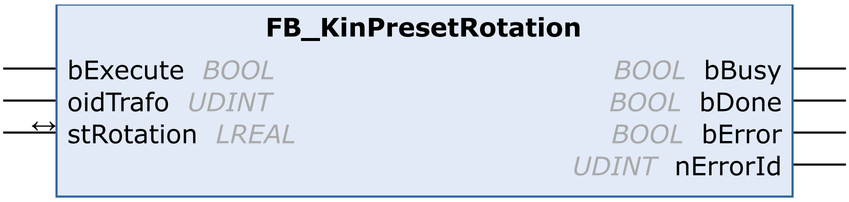
bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 Advanced Motion Pack V3.1.10.1	PC or CX (x86 or x64)	Tc2_KinematicTransformation (V3.2.7.3 or later)

8.1.8 FB_KinPresetRotation



The function block FB_KinPresetRotation sets the rotational state.

The rotational state is not persistent and must be reset after a TwinCAT restart or if a path is started after an ACS axis movement (direct mode).

Extended rotation range

✓ For a unique solution the standard rotation range is limited to:

a) Rotation1: -180 to 180 degrees,

b) Rotation2: -90 to 90 degrees,

c) Rotation3: -180 to 180 degrees.

⇨ In some 6-axis applications it is desirable to be able to rotate beyond this rotation range. The function blocks FB_KinExtendedRotationRange and FB_KinPresetRotation enable the rotational state to be extended, saved and restored beyond the default values.

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    oidTrafo       : UDINT;
    stRotation     : ARRAY[1..3] OF LREAL;
END_VAR
```

bExecute: The command is executed with a rising edge.

oidTrafo: Object ID (OTCID) of the kinematic transformation object.

stRotation: Presetting of MCS Rotation1, Rotation2 and Rotation3

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy                 : BOOL;
    bDone                 : BOOL;
    bError                : BOOL;
    nErrorId              : UDINT;
END_VAR
```

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

bDone: The output becomes TRUE when the command was executed successfully.

bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Example: Equivalent rotations (same tool orientation)

Rotation1:= -180	Rotation1:= -180
Rotation2:= 45	Rotation2:= 45

Rotation3:= 157.95 Rotation3:= -202.05

FB_KinPresetRotation must be used before *FB_KinConfigGroup* or *FB_KinCalcTrafo* perform the forward transformation.

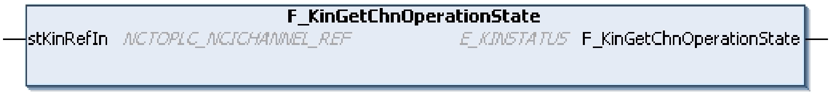
To use the extended rotation range with *FB_KinCalcTrafo(bForward:=TRUE)* without a kinematic group, the meta information uMetaInfo.aData[4] := 1 must be set.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 Advanced Motion Pack V3.1.10.1	PC or CX (x86 or x64)	Tc2_KinematicTransformation (V3.2.7.3 or later)

8.2 Functions

8.2.1 F_KinGetChnOperationState


F_KinGetChnOperationState
stKinRefIn NCTOPLC_NCICHANNEL_REF
E_KINSTATUS F_KinGetChnOperationState

This function returns the operating state of the kinematic channel.

Function F_KinGetChnOperationState : E_KINSTATUS

```
VAR_IN_OUT
    stKinRefIn : NCTOPLC_NCICHANNEL_REF
END_VAR
```

stKinRefIn: Determines the kinematic group of the configuration.

Return value

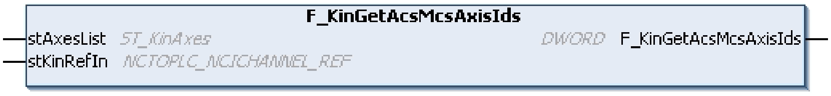
E_KINSTATUS [▸ 81]: State of the kinematic channel (see below). If an invalid version of the cyclic interface is used, *KinStatus_InvalidItfVersion* is returned.

Sample

```
VAR
    stKinRefIn AT %I*       : NCTOPLC_NCICHANNEL_REF;
    nErrId                  : UDINT;
    eKinOperationState      : E_KINSTATUS;
END_VAR
```

```
IF F_KinGetChnOperationState (stKinRefIn)<> KinStatus_InvalidItfVersion THEN
    eKinOperationState := F_KinGetChnOperationState (stKinRefIn);
ELSE
    nErrId := F_KinGetChnOperationState (stKinRefIn);
END_IF
```

8.2.2 F_KinGetAcsMcsAxisIds


F_KinGetAcsMcsAxisIds
stAxesList ST_KinAxes
stKinRefIn NCTOPLC_NCICHANNEL_REF
DWORD F_KinGetAcsMcsAxisIds

This function reads the configured ACS and MCS axes of the cyclic interface. The IDs are written to stAxesList.

FUNCTION F_KinGetAcsMcsAxisIds : UDINT

```
VAR_IN_OUT
    stAxesList : ST_KinAxes;
    stKinRefIn : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

stAxesList: List of axis IDs for the axis coordinate system (ACS) and the machine coordinate system (MCS).

stKinRefIn: The structure of the cyclic channel interface between the kinematic channel and the PLC. This structure is only accessed for reading.

Return value

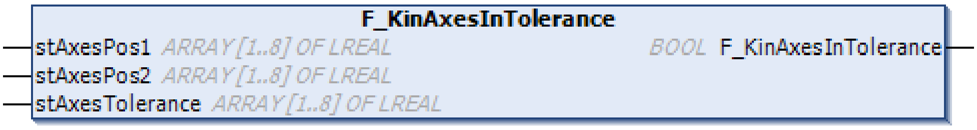
UDINT: Error code

Sample

```
VAR
    stAxesList          : ST_KinAxes;
    stKinRefIn AT %I*   : NCTOPLC_NCICHANNEL_REF;
    nErrId              : UDINT;
END_VAR
```

```
nErrId := F_KinGetAcsMcsAxisIds (stAxesList, stKinRefIn);
IF nErrId=0 THEN
    ;(*Axes List is valid*)
END_IF
```

8.2.3 F_KinAxesInTolerance


F_KinAxesInTolerance
stAxesPos1 ARRAY [1..8] OF LREAL
BOOL F_KinAxesInTolerance
stAxesPos2 ARRAY [1..8] OF LREAL
stAxesTolerance ARRAY [1..8] OF LREAL

The function block F_KinAxesInTolerance compares two arrays element by element.

The function returns TRUE if the difference between the respective array elements is within the expected tolerance.

VAR_INPUT

```
VAR_INPUT
    stAxesPosIn1     : ARRAY[1..8] OF LREAL;
    stAxesPosIn2     : ARRAY[1..8] OF LREAL;
    stAxesTolerance : ARRAY[1..8] OF LREAL;
END_VAR
```

stAxesPosIn1: First array. This is compared with the second array.

stAxesPosIn2: Second array. This is compared with the first array.

stAxesTolerance: Contains the tolerance for each array element to be compared.

Return value

BOOL: The function returns TRUE if the difference between the respective array elements is within the expected tolerance.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.7 Advanced Motion Pack V3.1.10.1	PC or CX (x86 or x64)	Tc2_KinematicTransformation (V3.2.7.3 or later)

8.3 Datatypes

8.3.1 ST_KinAxes

This structure defines the axes, which form a kinematic system.

```
TYPE ST_KinAxes :
STRUCT
    nAxisIdsMcs: ARRAY[1..8] OF DWORD;
    nAxisIdsAcs: ARRAY[1..8] OF DWORD;
END_STRUCT
END_TYPE
```

nAxisIdsMcs: List of axis IDs of the axes that form the MCS. Usually, the first three array elements specify the cartesian axes (X,Y,Z), the subsequent array elements specify the rotational axes.

nAxisIdsAcs: List of axis IDs of the axes that form the ACS.

Sample

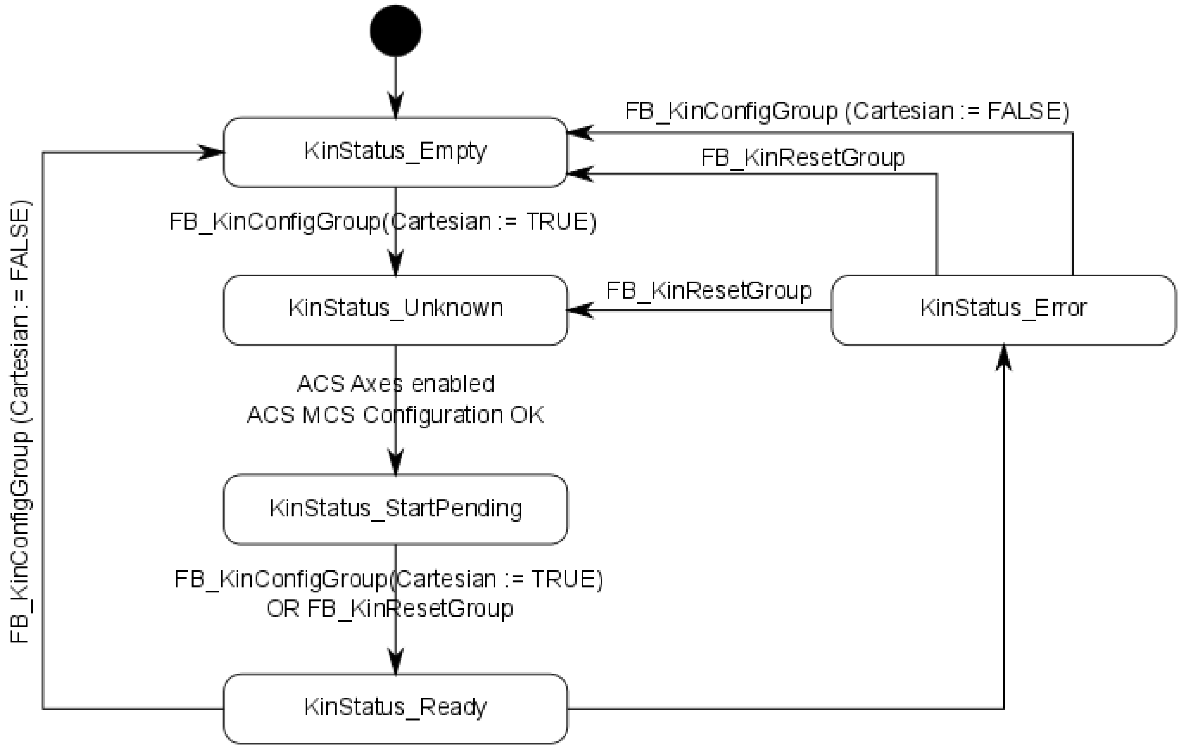
```
VAR
    stAxesConfig          : ST_KinAxes;
    io_X                  : AXIS_REF;
    io_Y                  : AXIS_REF;
    io_Z                  : AXIS_REF;
    io_M1                 : AXIS_REF;
    io_M2                 : AXIS_REF;
    io_M3                 : AXIS_REF;
END_VAR
```

```
(* read the IDs from the cyclic axis interface so the axes can mapped later to the kinematic group
*)
stAxesConfig.nAxisIdsAcs[1] := io_M1.NcToPlc.AxisId;
stAxesConfig.nAxisIdsAcs[2] := io_M2.NcToPlc.AxisId;
stAxesConfig.nAxisIdsAcs[3] := io_M3.NcToPlc.AxisId;
stAxesConfig.nAxisIdsMcs[1] := io_X.NcToPlc.AxisId;
stAxesConfig.nAxisIdsMcs[2] := io_Y.NcToPlc.AxisId;
stAxesConfig.nAxisIdsMcs[3] := io_Z.NcToPlc.AxisId;
```

8.3.2 E_KinStatus

This enumeration defines the state of the kinematic group.

```
TYPE E_KinStatus :
(
    KinStatus_Error,
    KinStatus_Empty,
    KinStatus_Unknown,
    KinStatus_StartPending,
    KinStatus_Ready,
    KinStatus_InvalidItfVersion := 16#4000
);
END_TYPE
```



KinStatus_Empty: ACS axes can be moved. No transformation enabled.

KinStatus_Ready: MCS axes can be moved. Transformation active.

KinStatus_InvalidItfVersion: A function or function block is not supported by this version of the cyclic channel interface. An update is required in order to be able to use the function.

Enable configuration

The ACS axes must be enabled through MC_Power, to ensure that the state can reach the value **KinStatus_Ready**.

8.3.3 CalcTrafo

8.3.3.1 E_KinMetaInfo5DType1

```
TYPE E_KinMetaInfo5DType1 :
(
    d5Type1Quad14  := 1,
    d5Type1Quad23  := 2,
    d5Type1ActualConfig := 3
);
END_TYPE
```

8.3.3.2 E_KinMetaInfoScara

Enum for defining the arm position for 4D-SCARA [▶ 40] kinematics.

```
TYPE E_KinMetaInfoScara :
(
    scaraLeftArm      := 1,
    scaraRightArm     := 2,
    scaraActualConfig := 3
);
END_TYPE
```

8.3.3.3 ST_KinMultiTrafoHeader

Header of the input data structure for the function block FB_KinCalcMultiTrafo [▶ 71].

```
Type ST_KinMultiTrafoHeader
STRUCT
    nColumnsIn   : UDINT;
    nColumnsOut  : UDINT;
    nLines       : UDINT;
    bGetMetaInfo : BOOL;
    uMetaInfo    : U_KinMetaInfo;
END_STRUCT
END_TYPE
```

Name	Type	Description
nColumnsIn	UDINT	Real number of columns in the input array.
nColumnsOut	UDINT	Real number of columns in the output array.
nLines	UDINT	Number of lines to be calculated, may be less than the actual number of lines declared.
bGetMetaInfo	BOOL	If TRUE, MetaInfos are also output. Accordingly, memory of the source data structure must be available for this purpose.
uMetaInfo	U_KinMetaInfo [▶ 83]	

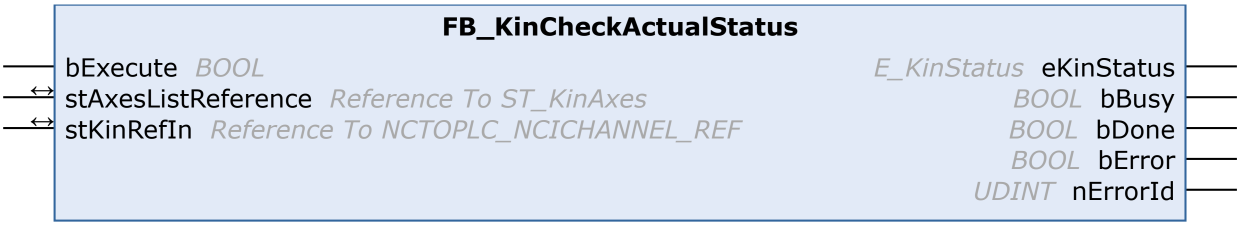
8.3.3.4 U_KinMetaInfo

```
Type U_KinMetaInfo
UNION
    aData     : ARRAY[1..4] OF UDINT;
    eScara    : E_KinMetaInfoScara;
    e5dType1  : E_KinMetaInfo5DType1;
END_UNION
END_TYPE
```

Name	Type	Description
aData	ARRAY[1..4] OF UDINT	
eScara	E_KinMetaInfoScara [▶ 82]	Definition of the arm position with 4D-SCARA [▶ 40] kinematics.
e5dType1	E_KinMetaInfo5DType1 [▶ 82]	

8.4 Legacy

8.4.1 FB_KinCheckActualStatus



Outdated version

The sole purpose of the function block is to ensure compatibility with existing projects. For new projects please use F_KinGetChnOperationState [▶ 79]. This function block needs more than one PLC cycle to read the status of the kinematic channel. To obtain the status for each cycle please use F_KinGetChnOperationState [▶ 79].

The function block FB_KinCheckActualStatus returns the status of the kinematic channel.

VAR_INPUT

```
VAR_INPUT
    bExecute                : BOOL;
END_VAR
```

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

VAR_IN_OUT

```
VAR_IN_OUT
    stAxesList              : ST_KinAxes;
    stKinRefIn              : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

stAxesList: Determines the ACS and MCS axes included in the configuration. See ST_KinAxes.

stKinRefIn: Determines the kinematic group of the configuration.

VAR_OUTPUT

```
VAR_OUTPUT
    eKinStatus              : E_KINSTATUS;
    bBusy                   : BOOL;
    bDone                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

eKinStatus: Returns the status of the kinematic channel. See E_KINSTATUS [▶ 81].

bBusy: The output becomes TRUE when the command is started with *bExecute* and remains TRUE as long as the function block executes the command. While *bBusy* is TRUE, no new command is accepted at the inputs. If *bBusy* becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs *bDone* or *bError* is set.

bDone: The output becomes TRUE when the command was executed successfully.

bError: The output *bError* is set to TRUE, if an error occurred during the command execution.

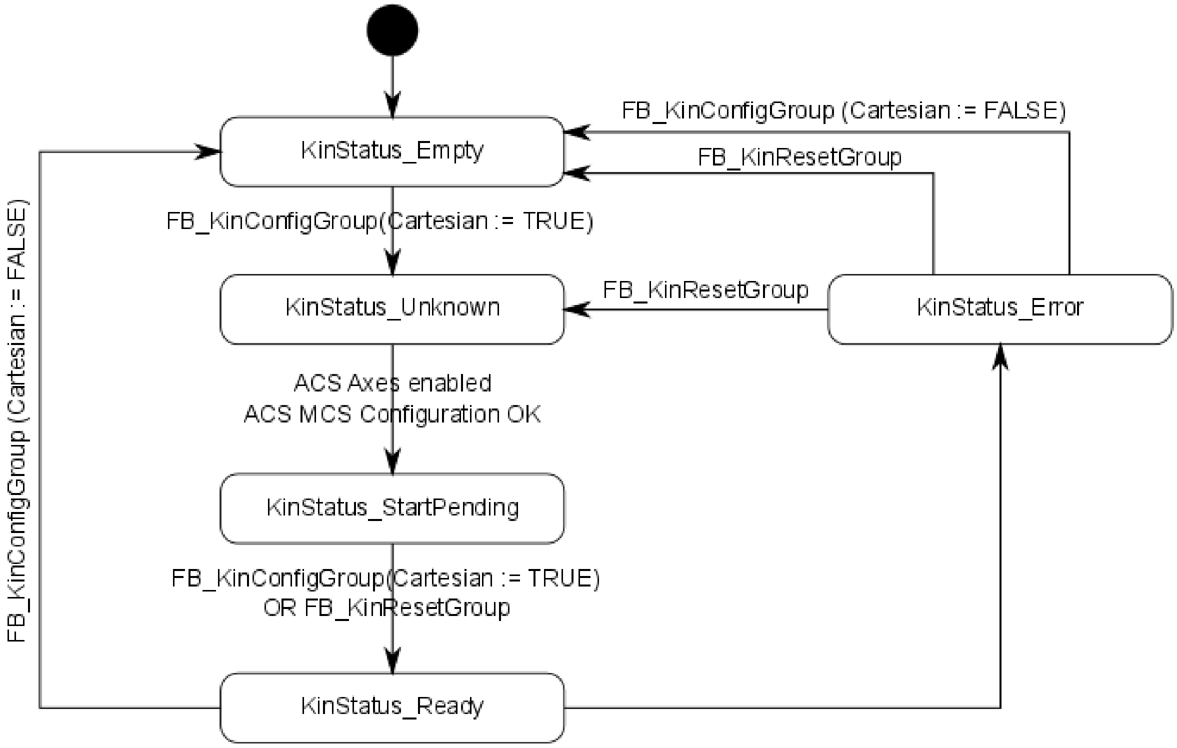
nErrorId: contains the command-specific error code of the most recently executed command. Details of the error code can be found in the ADS error documentation or in the NC error documentation (error codes from 0x4000).

Sample

```
VAR
    fbFB_KinCheckActualStatus  : FB_KinCheckActualStatus;
    in_stKinToPlc AT %I*       : NCTOPLC_NCICHANNEL_REF;
    stAxesConfig               : ST_KinAxes;
    eKinStatus                 : E_KINSTATUS;
END_VAR
```

```
fbFB_KinCheckActualStatus(
    bExecute              := TRUE,
    stAxesListReference   := stAxesConfig,
    stKinRefIn            := in_stKinToPlc,
    eKinStatus            => eKinStatus      );
```

State of the kinematic group



More Information:
www.beckhoff.com/tf5110

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