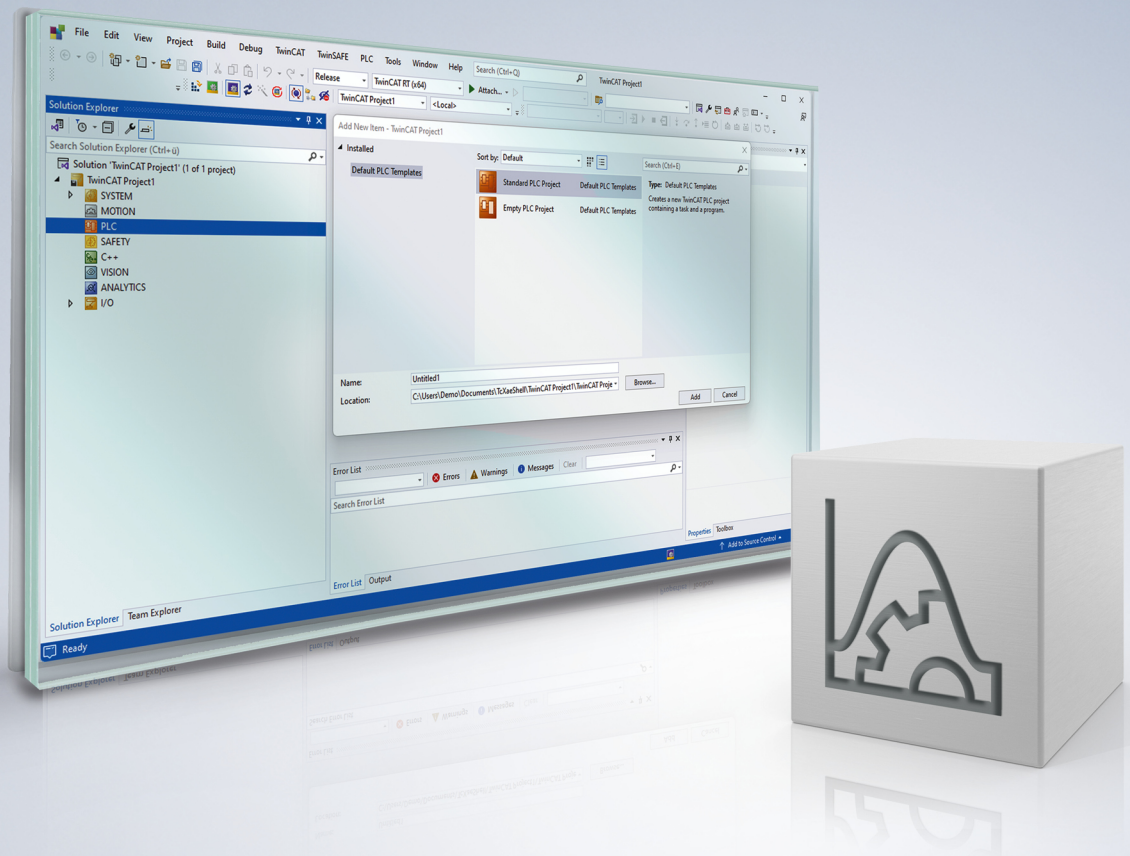
BECKHOFF New Automation Technology

Manual | EN

TF55xx

TwinCAT 3 | MC3



2026-08-14 | Version: 1.0.2

Table of contents

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

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

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

Copyright

Third-party trademarks

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

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

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

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

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

Warning of damage to property or environment

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

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Introduction to TwinCAT MC3

TF55xx | TwinCAT 3 MC3 is the new generation of motion control that integrates seamlessly with TwinCAT, providing a seamless and efficient automation solution.

Complete system integration

TF55xx | TwinCAT 3 MC3 is seamlessly integrated into the TwinCAT system, which means that this motion control solution can be operated in parallel with the other TwinCAT functions in one system and can also interact with the other TwinCAT functions.

Multi-core support

TF55xx | TwinCAT 3 MC3 can be distributed to several CPU cores as required. Movements can be synchronized across all CPU cores in use.

Multitasking support

Axes with different cycle times can be operated on a single CPU core, precisely matching the velocities and tasks of the respective axes. This facilitates optimum utilization of a CPU core, as the fastest axis does not necessarily dictate the clock rate for all axes. The axes of a conveyor belt with a delta picker can thus be operated at a cycle time of 1 ms to ensure that the workpieces are picked up quickly, while the axes for width adjustment of the conveyor belt are operated at a cycle time of 4 ms on the same CPU core.

Modular configuration

Functional extensions using TcCOM objects and flexible interfaces enable a future-proof architecture and a modular project configuration. The web-based commissioning interface can be customized for specific applications.

Hardware independence

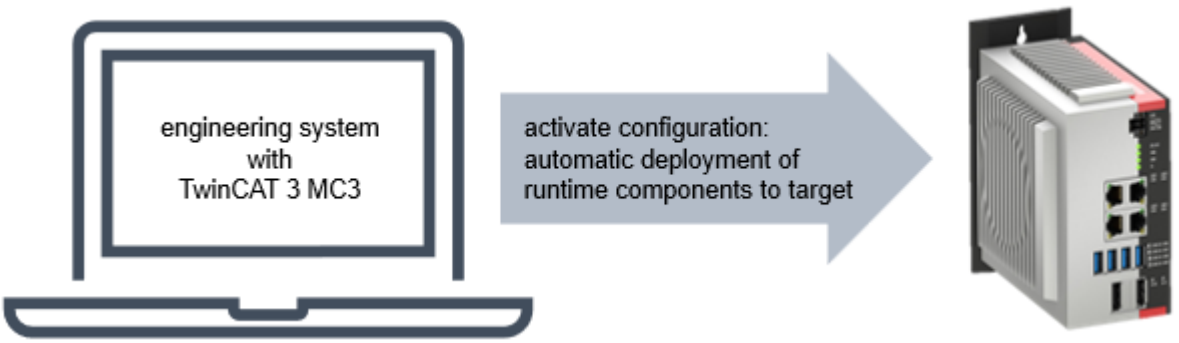
As with the previous Motion Control solution, axis objects separate the hardware from the programming. This creates hardware independence.

Fluid power (hydraulics) support

TF55xx | TwinCAT 3 MC3 ensures equally deep integration of electrical and hydraulic axes and enables seamless interaction, precise synchronization, and flexible integration across different hardware platforms. With the scalable platform, TF5560 | TwinCAT 3 MC3 Fluid Power can be used for a wide range of applications, from isolated hydraulic solutions to complete machine control systems.

2.1 Installation

TF55xx | TwinCAT 3 MC3 is installed only on the engineering side. On the runtime side, a TwinCAT Standard installation version 3.1 Build 4026.23.1 or later is required. When the configuration is activated, specific MC3 components are transferred from the engineering system to the runtime system. The operating system on which the runtime system is based is automatically detected and taken into account.



Engineering installation (Windows)

From TwinCAT 3.1 Build 4026, TwinCAT products are installed via the TwinCAT Package Manager. Detailed instructions on installing products can be found in the chapter Managing TwinCAT software in the TwinCAT 3.1 Build 4026 setup instructions.

You can obtain a basic TwinCAT XAE installation using the following workload:

- TwinCAT.Standard.XAE

Basic MC3 components are installed with the following workload:

- TF5500.MC3.Base.XAE

Additional MC3 components can also be installed using the following workloads:

- TF5550.MC3.Camming.XAE
- TF5560.MC3.FluidPower.XAE

In addition, we recommend installing Drive Manager 2 for using and parameterizing Beckhoff Motion hardware, as well as TwinCAT Scope:

- TE5950.DriveManager2.XAE (Engineering) (v1.1.114.0 or later recommended)
- TE1300.ScopeViewProfessional.XAE (Engineering)
- TF3300.ScopeServer.XAR (Runtime)

Runtime installation

On the runtime side, only a TwinCAT Standard installation version 3.1 Build 4026.23.1 or later is required. All necessary TwinCAT MC3 components are transferred from the engineering system to the runtime system when the configuration is activated.

Windows XAR installation via the TwinCAT Package Manager using the following workload:

- TwinCAT.Standard.XAR

Beckhoff RT Linux® Runtime installation via the Beckhoff Linux package server using the following package:

- tc31-xar-um

TwinCAT/BSD Runtime installation:

- The TwinCAT Runtime installation is included in the TwinCAT/BSD installation.

2.2 License model

The TwinCAT 3 MC3 licensing model consists of a base license, axis extension licenses, and function extension licenses.

TF5500 | TwinCAT 3 MC3 Base

The TF5500 | TwinCAT 3 MC3 Base license is a prerequisite for any MC3 axis application.

TF5500 | TwinCAT 3 MC3 Base includes 10 axis licenses. An axis license is initially only valid for Beckhoff or simulation axes. The number of licensed axes can be expanded flexibly using an MC3 Axes license (TF551x).

Axes from other manufacturers can also be controlled. However, they require a multi-vendor license in addition to an axis license. Five multi-vendor licenses are included in the MC3 Base license, and more can be added through an additional TwinCAT 3 MC3 Multi Vendor license (TF553x).

This results in possible configurations consisting of five Beckhoff or simulation axes and five axes from other manufacturers (for a total of 10 axis licenses). The key factor is a total of no more than 10 axes, with a maximum of five axes from other manufacturers that can be used. It is not possible to combine four Beckhoff or simulation axes with six axes from other manufacturers without an additional TwinCAT 3 MC3 Multi Vendor license and a TwinCAT 3 MC3 Axes license. The Axes license is required because a Multi Vendor license always requires at least one Axes license of the same size.

In addition to axis licenses, the MC3 Base license includes the functionality needed for the implementation of point-to-point (PTP) applications and couplings, such as the flying saw. Further functionalities are available through supplementary products (TF5550, TF5560).

TF551x | TwinCAT 3 MC3 Axes

TwinCAT 3 MC3 Axes expand TF5500 TwinCAT 3 MC3 Base with additional axis licenses.

Depending on the TF551x license and platform, the number of axes can be increased to 25, 50, 75, 100, 150, 250, 500, 1000, or 1500. The number of axes from the TwinCAT 3 MC3 Axes license is considered the new maximum number of axes. The 10 axis licenses included in TwinCAT 3 MC3 Base (or up to five of those axes from other manufacturers) are already included in the total and are not added to it. It is not possible to combine TwinCAT 3 MC3 Axes licenses.

An axis license is initially only valid for Beckhoff or simulation axes. Axes from other manufacturers can also be controlled. However, they require a multi-vendor license in addition to an axis license. Five multi-vendor licenses are included in the MC3 Base license, and more can be added through an additional MC3 Multi Vendor license (TF553x).

TF number	Name
TF5510	TwinCAT 3 MC3 Axes 25
TF5511	TwinCAT 3 MC3 Axes 50
TF5512	TwinCAT 3 MC3 Axes 75
TF5513	TwinCAT 3 MC3 Axes 100
TF5514	TwinCAT 3 MC3 Axes 150
TF5515	TwinCAT 3 MC3 Axes 250
TF5516	TwinCAT 3 MC3 Axes 500
TF5517	TwinCAT 3 MC3 Axes 1000
TF5518	TwinCAT 3 MC3 Axes 1500

More than 1500 axes

The TF5518 | TwinCAT 3 MC3 Axes 1500 license does not represent the actual maximum possible number of axes. If you need to control more than 1500 axes, please contact your Beckhoff representative.

TF553x | TwinCAT 3 MC3 Multi-Vendor

In addition to the control of Beckhoff axes, TwinCAT 3 MC3 also supports the control of axes from other manufacturers.

However, these require a multi-vendor license in addition to an axis license. Five multi-vendor licenses are included in the MC3 Base license, and more can be added through an additional TwinCAT 3 MC3 Multi Vendor license.

A variant of TF551x TwinCAT 3 MC3 Axes is also required to obtain the necessary axis licenses.

TF number	Name
TF5530	TwinCAT 3 MC3 Multi-Vendor 25
TF5531	TwinCAT 3 MC3 Multi-Vendor 50
TF5532	TwinCAT 3 MC3 Multi-Vendor 75
TF5533	TwinCAT 3 MC3 Multi-Vendor 100
TF5534	TwinCAT 3 MC3 Multi-Vendor 150

TF5550 and TF5560 function extension licenses

The TF5500 enables the implementation of PTP applications with simple couplings. The following licenses can be used to expand the range of functionality.

TF number	Requirement	Description
TF5550	TF5500	TwinCAT 3 MC3 Camming Extension with non-linear couplings (cam plates).
TF5560	TF5500	TwinCAT 3 MC3 Fluid Power Extension with Fluid Power functionality.

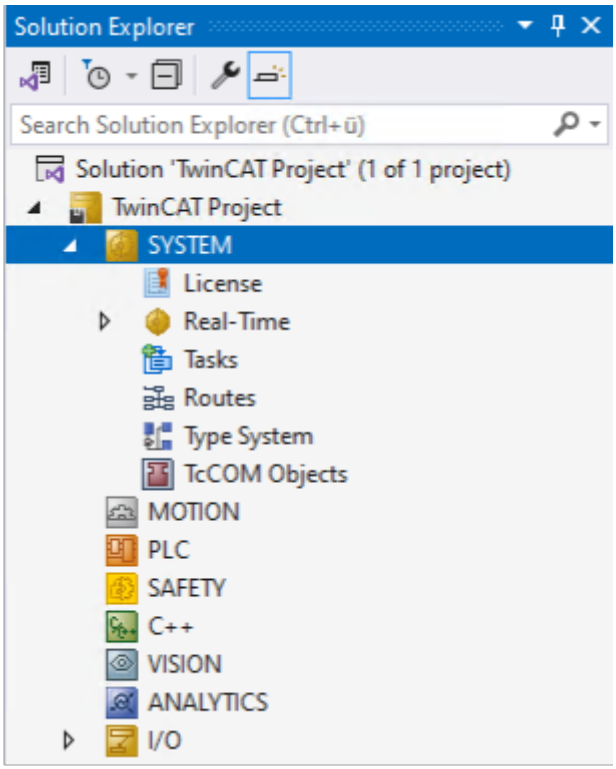
3 Development environment

TwinCAT MC3 is seamlessly integrated into the TwinCAT development environment (XAE) based on Microsoft Visual Studio. Within a TwinCAT project, the MOTION [▶ 15] node is the most important project node for TwinCAT MC3. In addition, the SYSTEM [▶ 12] node is relevant for general settings. Both nodes are described below in the context of TwinCAT MC3.

3.1 SYSTEM

The SYSTEM node in the tree view of the TwinCAT project in Solution Explorer contains the system configuration. These include, among other things, real-time settings, task configuration, and licenses. This documentation discusses the SYSTEM node only in conjunction with TwinCAT 3 MC3.

For general information about the SYSTEM node, see the TE1000 XAE Documentation.



3.1.1 Real-Time

The axis calculations of TwinCAT MC3 can be distributed across multiple CPU cores as required. Each axis is assigned to a cyclic task. A task can be assigned to multiple axes. Cycle times and priorities can be set based on the assignment of axes to a task. Movements can be synchronized across all CPU cores in use.

Object	RT-Core	Base Time (ms)	Cycle Time (ms)	Cycle Ticks	Priority
MC-Task (MET)	Core 0	1 ms	2 ms	2	3
MC-Task (MET)_1	Core 1	1 ms	1 ms	1	4
MC-Task (MPT)	Core 0	1 ms	10 ms	10	8
MC-Task (MPT)_1	Core 1	1 ms	5 ms	5	9
I/O Idle Task	Core 2	1 ms	1 ms	1	11
Job Task	Core 2	1 ms	(none)	0	32

The assignment of tasks to CPU cores takes place in the TwinCAT project under **SYSTEM > Real-Time > Settings.** To display the current CPU core configuration of the target system, click **Read from Target**. The tasks can then be distributed across the appropriate cores.

To change the CPU core configuration of the target system, click **Set on Target** and select the desired distribution of divided and isolated cores.

Even on the same CPU core, axes with different cycle times can be operated by assigning them to different contexts [▶ 19]. The “fastest” axis therefore does not determine the cycle time for all axes. This allows for optimized utilization of a CPU core - even with only a few axes.

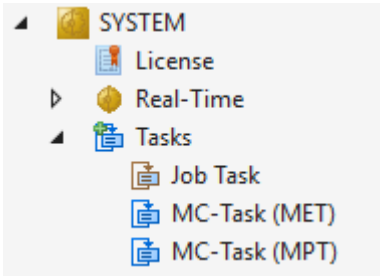
3.1.2 Tasks

The configuration of tasks is not specific to TwinCAT Motion, but is the same for all TwinCAT components. The following describes the relevant settings related to TwinCAT Motion.

Basically, all cyclic tasks required for TwinCAT Motion are created automatically. These tasks together form a context:

- Motion Execution Task (MET)
- Motion Preparation Task (MPT)
- Job Task

The MET processes commands and performs cyclic setpoint generation. This task also handles I/O communication. Preliminary calculations are performed in the MPT. The Job Task is currently not used and is intended for future non-cyclic calculations.



By default, the MET is created with a cycle time of 2 ms and the MPT with a cycle time of 10 ms. You can change the task settings under **SYSTEM > Tasks > [TaskName]**. MET and MPT must run on the same core.

Specifically, axes that require shorter cycle times can be assigned to a context with a MET that has, for example, a 1-millisecond cycle time. Axes that require a lower clock frequency can then be assigned to another context with a second MET, for example, with a 4-ms cycle time, which can run on the same or a different core. Creating the context and assigning the axes is done in the Context dialog of the Motion project [▶ 19].

3.1.3 TcCOM Objects

All TcCOM objects in the project are listed in the **SYSTEM > TcCOM Objects** node under the **Project Objects** tab. Among other things, an MC3 axis or a cam plate object in MC3 corresponds to a TcCOM object.

In the **Version** column, you can see the version of each object. For new projects, the object versions are automatically set to the latest version; no user action is required. If a project created with an older software version is to be migrated to a newer version, the setting can be configured here. Alternatively, the version can be set using the Motion Versioning extension [▶ 190].

Different versions of hierarchical objects

Different versions between child objects and their parent object can cause errors when activating the project.

- Make sure that the same version is set and active for the parent context element and all child Motion Control objects.

3.1.3.1 Version control

Multiple versions of the TwinCAT MC3 packages can be installed on your engineering system at the same time. This is especially the case if your machines are deployed in the field with different software versions. The following sections explain how to work with different versions.

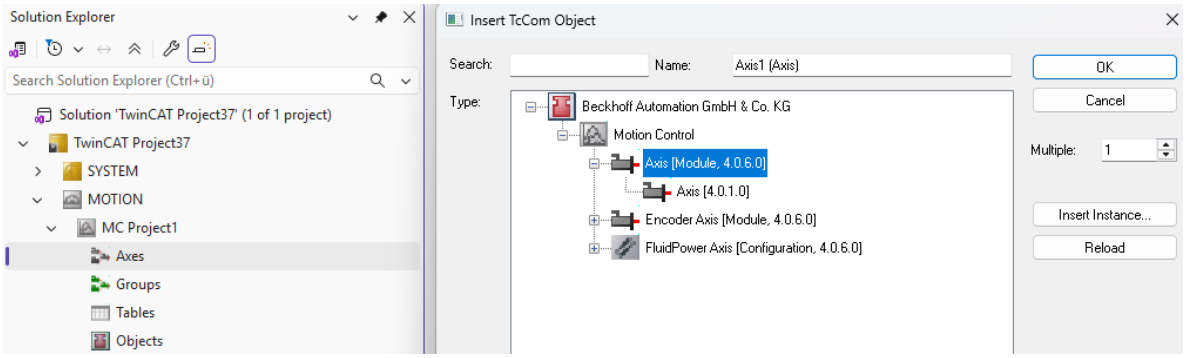
Alternatively, the version can be set using the Motion Versioning extension [▶ 190].

Requirements

- A TwinCAT project with an MC project has been created;
  see Creating an MC Project [▶ 17].
- Optionally, an axis may already have been created within the MC project. If you are not yet familiar with creating an axis within the MC project,
  please refer to the chapter Creating an Axis Object [▶ 25].

Add an axis to the MC project in a specific version

1. Right-click **Axes** > **Add New Item...**
2. In the **Insert TcCOM Object** dialog box, under **Type**, select the axis type and the version of the desired object.
3. To do this, expand the desired object by clicking the **+**.
   ⇨ All installed versions are displayed.
   ⇨ The default setting uses the latest (highest) installed version of the object. In this case, the view does not need to be expanded.



4. Finally, under **Multiple**, enter the desired number. Then click **OK** to confirm your entries.
   ⇨ The object has been created and is located under the Axes node.

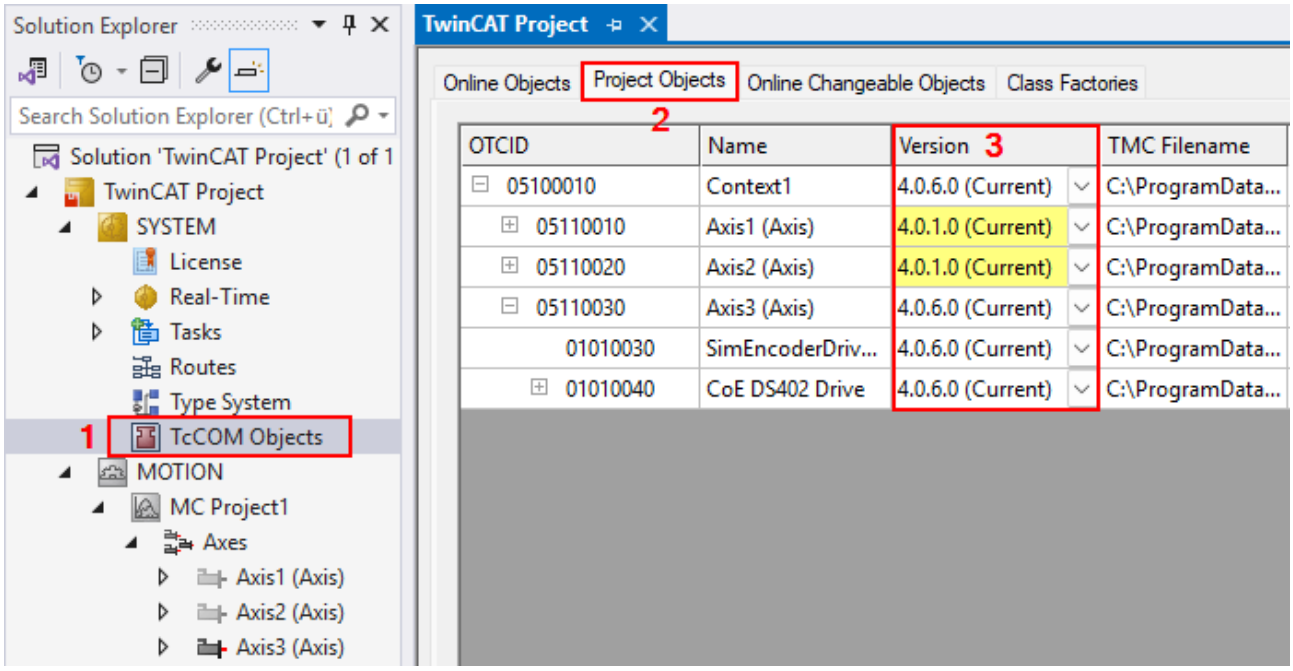
Set the version for motion objects in existing projects

Different versions of hierarchical objects

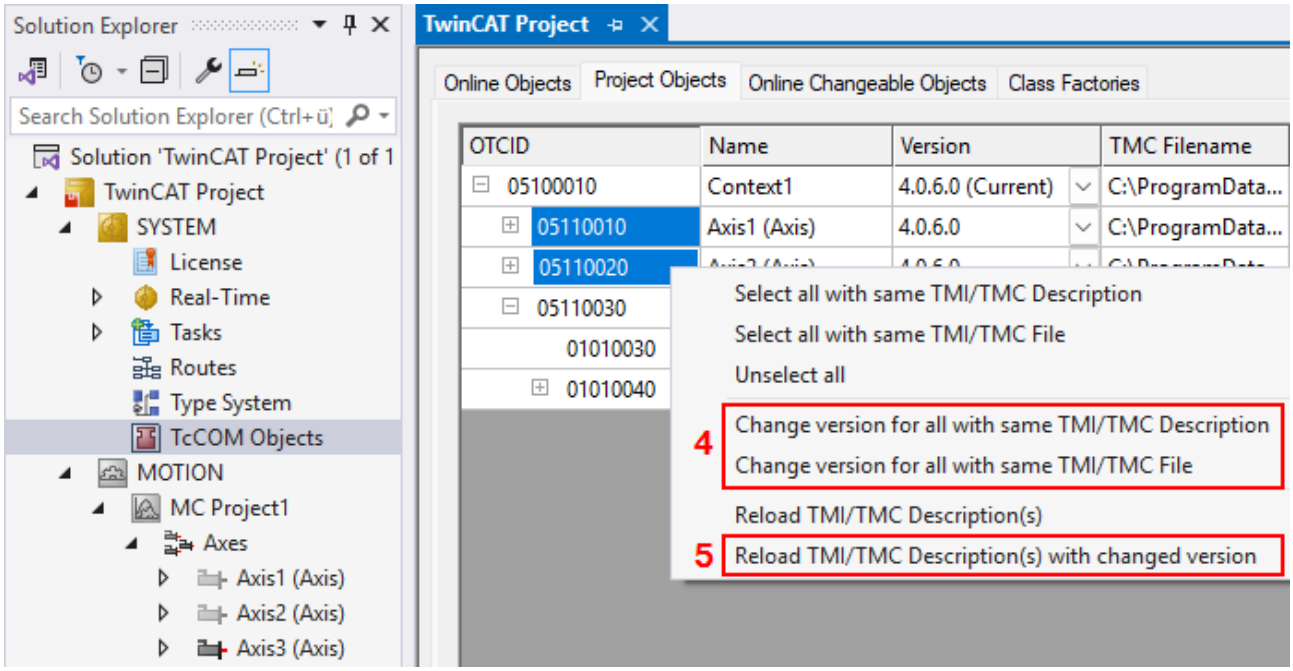
Different versions between child objects and their parent object can cause errors when activating the project.

- Make sure that the same version is set and active for the parent context element and all child Motion Control objects.

1. Select the **SYSTEM > TcCOM Objects** node (1).
2. Go to the **Project Objects** tab (2).
   ⇨ All of the project's TcCOM objects are listed there. Among other things, an MC3 axis or a cam plate object in MC3 corresponds to a TcCOM object.
3. In the **Version** column, select the desired version of an object.
   ⇨ The currently loaded and active version of an object is identified as **x.x.x.x (Current)**, as opposed to the selected version **x.x.x.x** (3).
   ⇨ Versions of an object for which a newer version is installed on the development system are highlighted in yellow.



4. You can apply a modified version to multiple objects. Select **Change version for all with same TMI/TMC Description** to apply the version of one axis to all other axes (objects of the same type). With the **Change version for all with same TMI/TMC File** option, you apply the version to all objects from the same TMC file, for example, to axes and their subordinate drive/encoder objects (4).
5. Finally, a modified version of an object must be activated. To do this, reload the object with the updated version and the associated TMC object description. Select **Reload TMI/TMC Description(s) with changed version** (5).



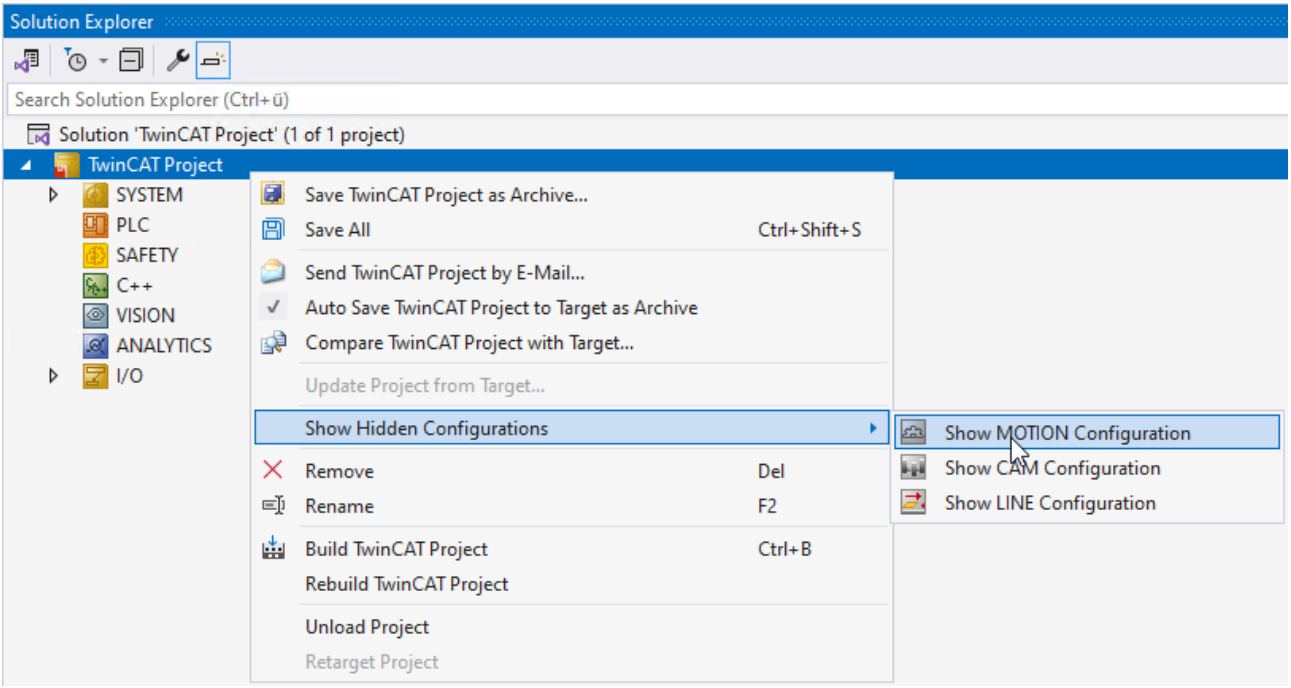
3.2 MOTION

The NC2, MC3, and CNC configurations are managed via the MOTION node in the TwinCAT project's tree view in Solution Explorer.

This documentation focuses exclusively on the MC3 configuration.

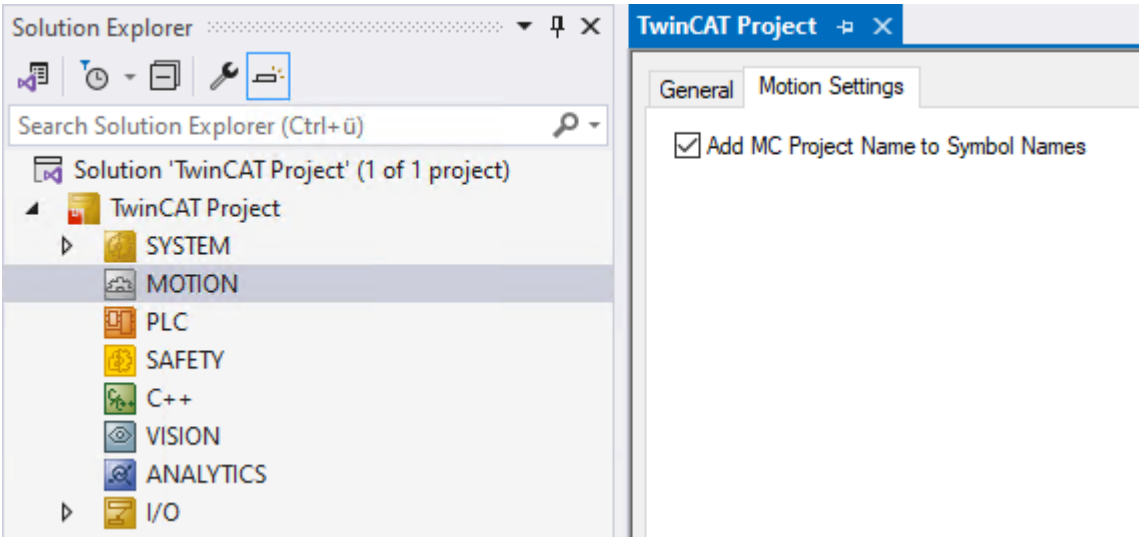
Display MOTION nodes

If the MOTION node is not displayed in the TwinCAT system, you can show it by right-clicking on **TwinCAT Project > Show Hidden Configurations > Show MOTION Configuration**.



Motion Settings tab

The **Motion Settings** tab allows you to configure motion settings that apply to all child projects (of a given type).

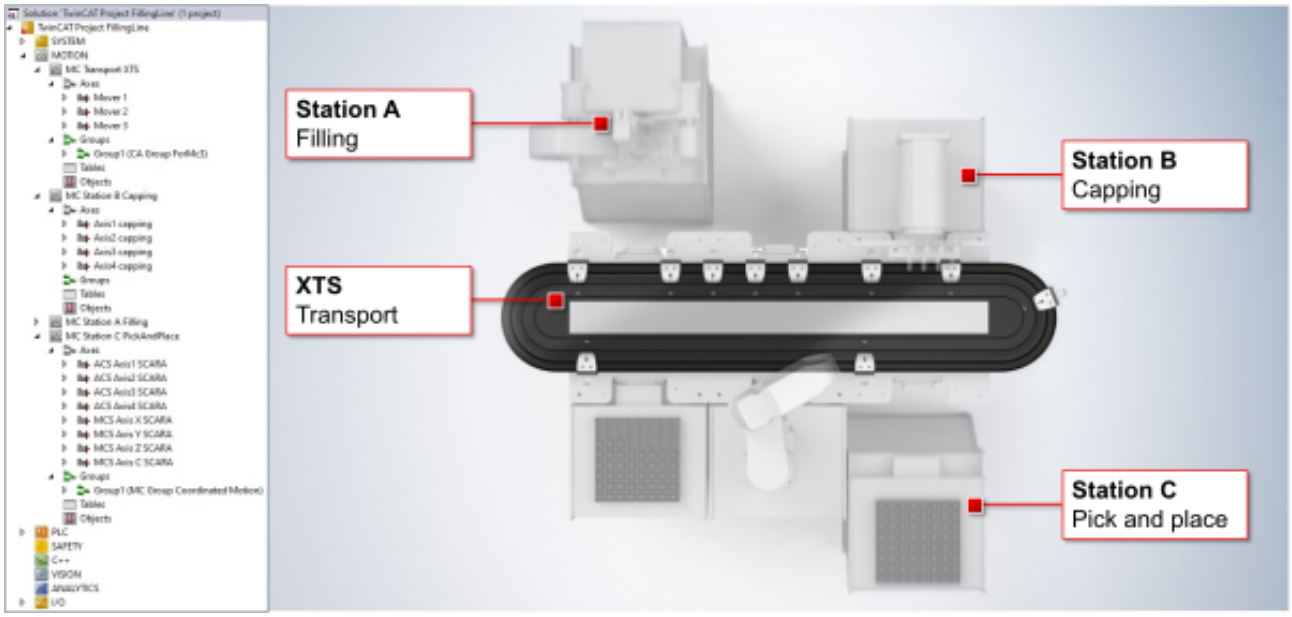


Add MC Project Name to Symbol Names	
▣	Default setting If there is only one MC project in a TwinCAT project, the name of the MC project is not included in ADS symbol names. If a TwinCAT project contains multiple MC projects, the ADS symbol names for an MC project begin with the name of the MC project to avoid duplicate ADS symbol names.
☑	The ADS symbol names of an MC project begin with the name of the MC project.
☐	The ADS symbol names of an MC project do not include the name of the MC project. If there are multiple MC projects within a TwinCAT project, it is the user's responsibility to prevent duplicate ADS symbol names.

3.2.1 MC Project

An MC project is an organizational unit consisting of MC3 components, such as axis objects, groups, tables, and other objects.

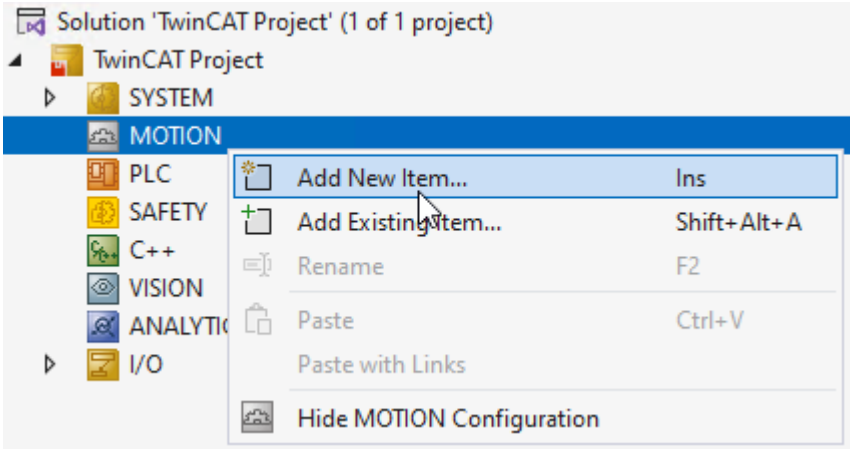
Multiple MC projects can be created within a TwinCAT project. For example, one MC project for each machine module, as shown in the following diagram. It is still possible to manage all axes through an MC project.



3.2.1.1 Creating an MC project

✓ A TwinCAT project has been created.

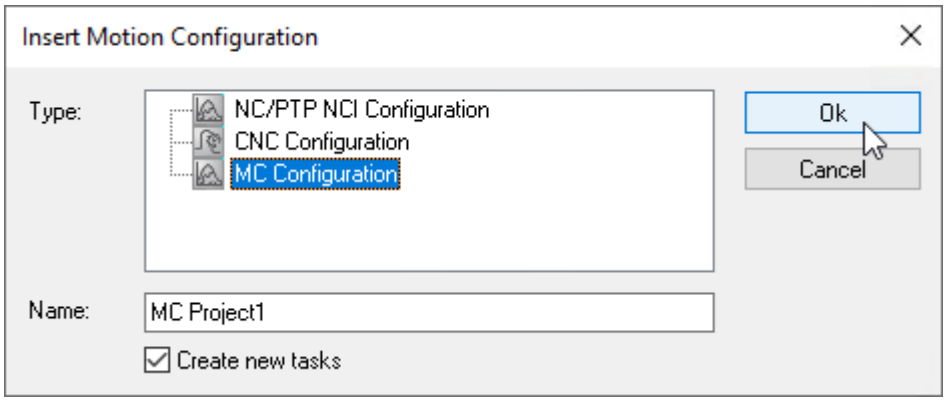
1. In the tree view, right-click **MOTION > Add New Item…**



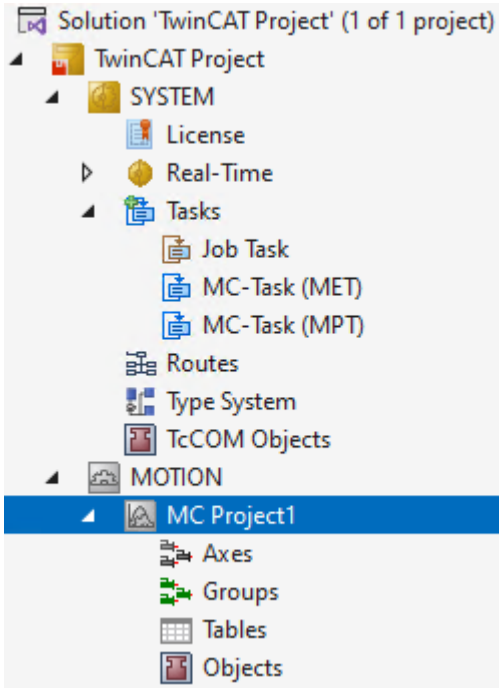
⇨ The **Insert Motion Configuration** dialog opens.

2. Select **MC Configuration**.
3. Assign a project name.

4. Confirm your entries by clicking **OK**.



⇨ A MC project with the selected name is created.

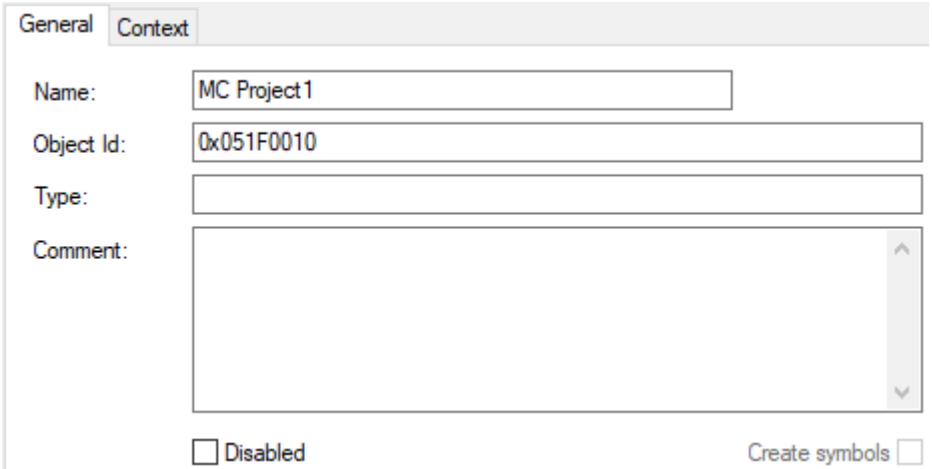


"Create new tasks" option selected

A new MET task and an MPT task will be created for the MC project.

- If you do not want a new set of tasks, clear this option. The first-created tasks will be used for the new MC project.

3.2.1.2 General tab



Property	Description
Name	The selected name for the MC project. **Do not use the same project name more than once.**
Object Id	32-bit identification number that is automatically assigned and unique within a TwinCAT project.
Type	
Comment	Freely editable field for user notes.
Disabled	Option to disable an MC project.

3.2.1.3 Context tab

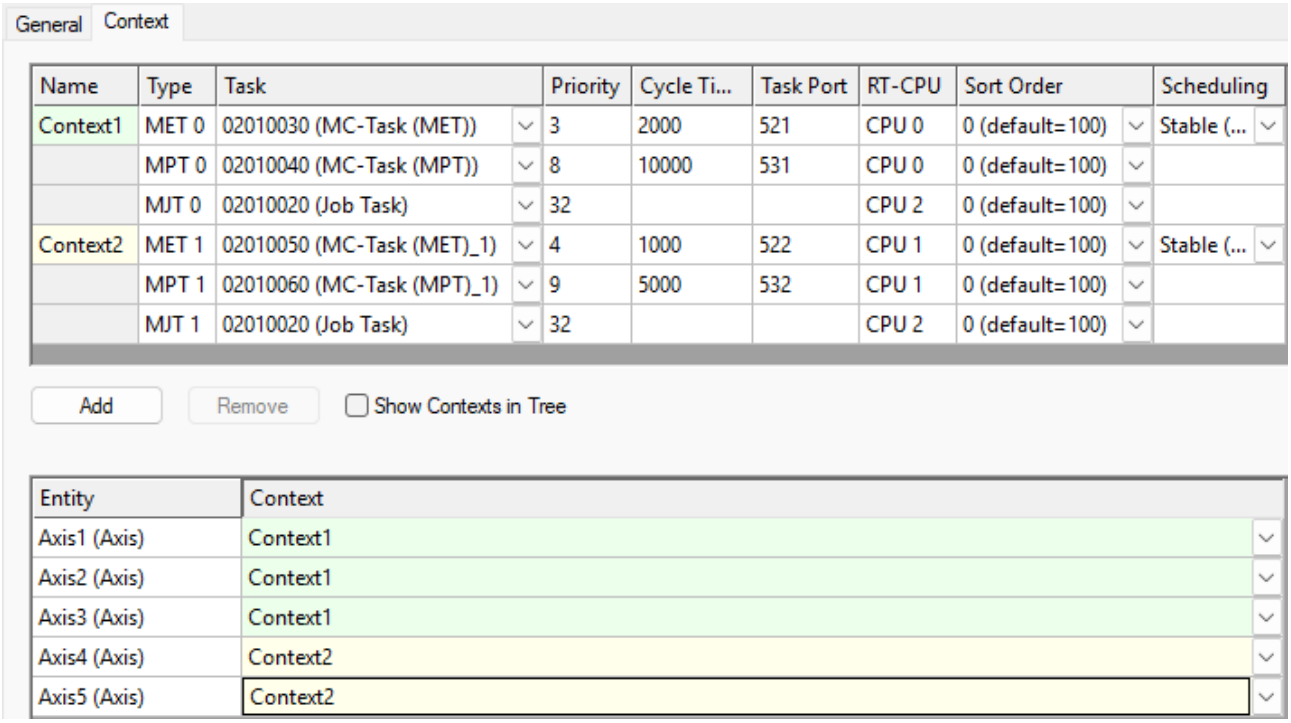
In TwinCAT MC3, the execution of the axes can be assigned to different task contexts. Contexts make it possible to distribute axis execution across multiple CPU cores and to specify different cycle times for different axes, even on a single core. In the MC project, you can configure the settings on the **Context** tab.

A task context always consists of three tasks:

- MET – Motion Execution Task
- MPT – Motion Preparation Task
- Job Task

The MET and MPT tasks of a context must always be assigned to the same core, and both tasks may only be used in one context. Conversely, a job task may be assigned to multiple contexts.

Task cycle times are configured via the respective task object under SYSTEM > Tasks [▶ 13]. CPU core allocation is configured under SYSTEM > Real-Time [▶ 12].



Alternatively, the task context for an axis can be configured on the axis's **Settings** tab.

Scheduling

An MC3 axis is a TcCOM object. The basics of TcCOM real-time module execution are explained in the “Basics” documentation at the end of the Real-Time > Differences in execution between PLC and TcCOM runtime modules chapter. Based on this, the following configuration options can be used to influence the internal execution sequence of command processing and setpoint generation.

Stable (Output – Commands – Exec)

This setting corresponds to “IO at the start of the task”. Command processing and setpoint generation are performed during the “post-cycle update” and therefore after the “output Update”. Select this setting to reduce jitter in the “output update”. Calculated setpoints are written to the output process image in the next cycle.

Fast Reaction (Commands - Exec - Output)

With this setting, the commands and setpoints are processed or calculated during the “cycle update” between the “input update” and the “output update”. Please note that the “Output Update” option has higher jitter than the “Stable” option. Calculated setpoints are written to the output process image immediately after their generation (in the same cycle).

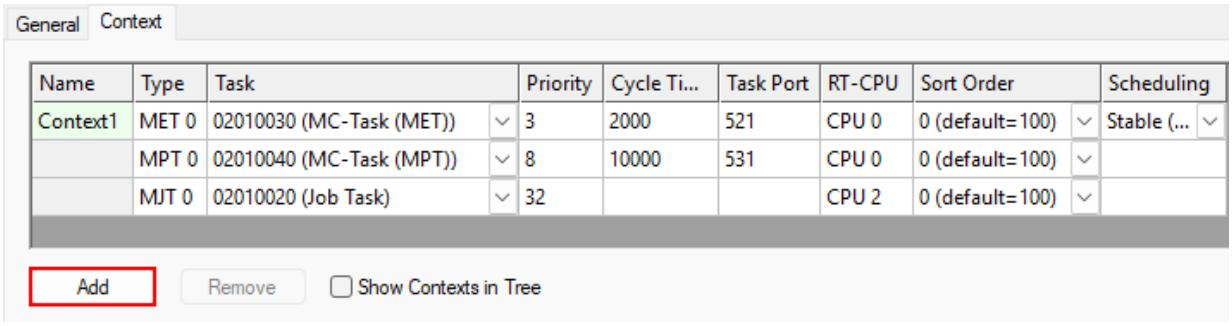
Scheduling when using the SAF NC task

Please note that this setting has no effect on instances that use the NC SAF task. In this case, Fast Reaction (Commands – Exec – Output) is used regardless of the parameter settings.

3.2.1.3.1 Create a new context

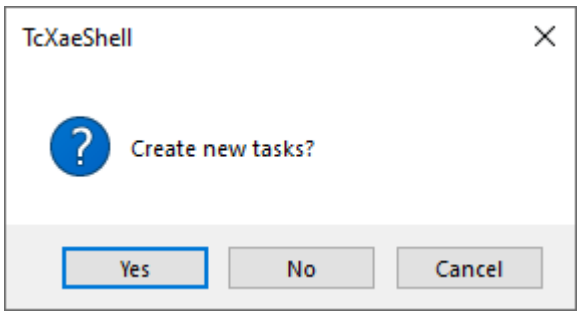
✓ A TwinCAT project with an MC project has been created;
see Creating an MC project [▶ 17].
A Motion project can consist of 1 to n task contexts. You can add an additional context to a Motion project by following the steps below.

1. In the Context [▶ 19] tab of the Motion project, click **Add**.



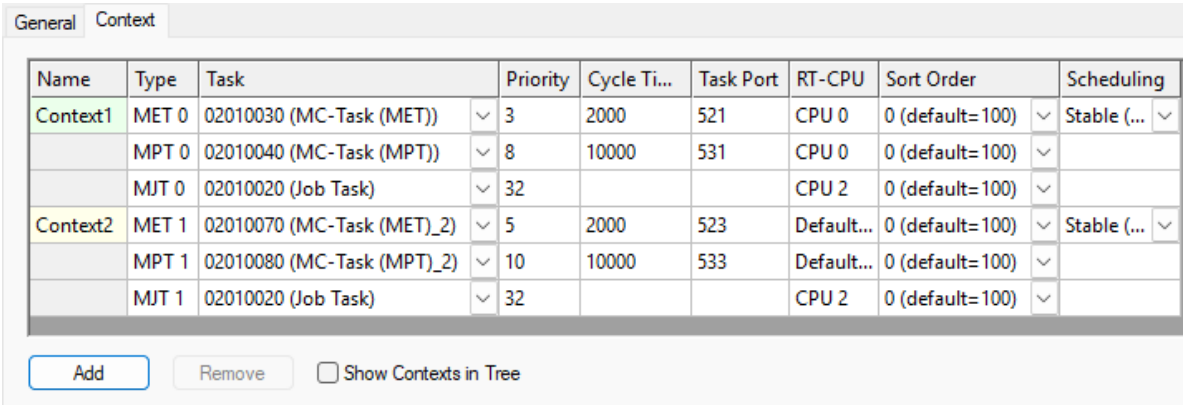
⇨ You are then prompted to specify whether tasks should be created for the new context. Each context requires its own MET and MPT task.

2. If no tasks have been created for the new context yet, click the **Yes** button to have TwinCAT create the tasks.



⇨ A new context is created.

⇨ When new tasks have been created, they are assigned directly to the context.

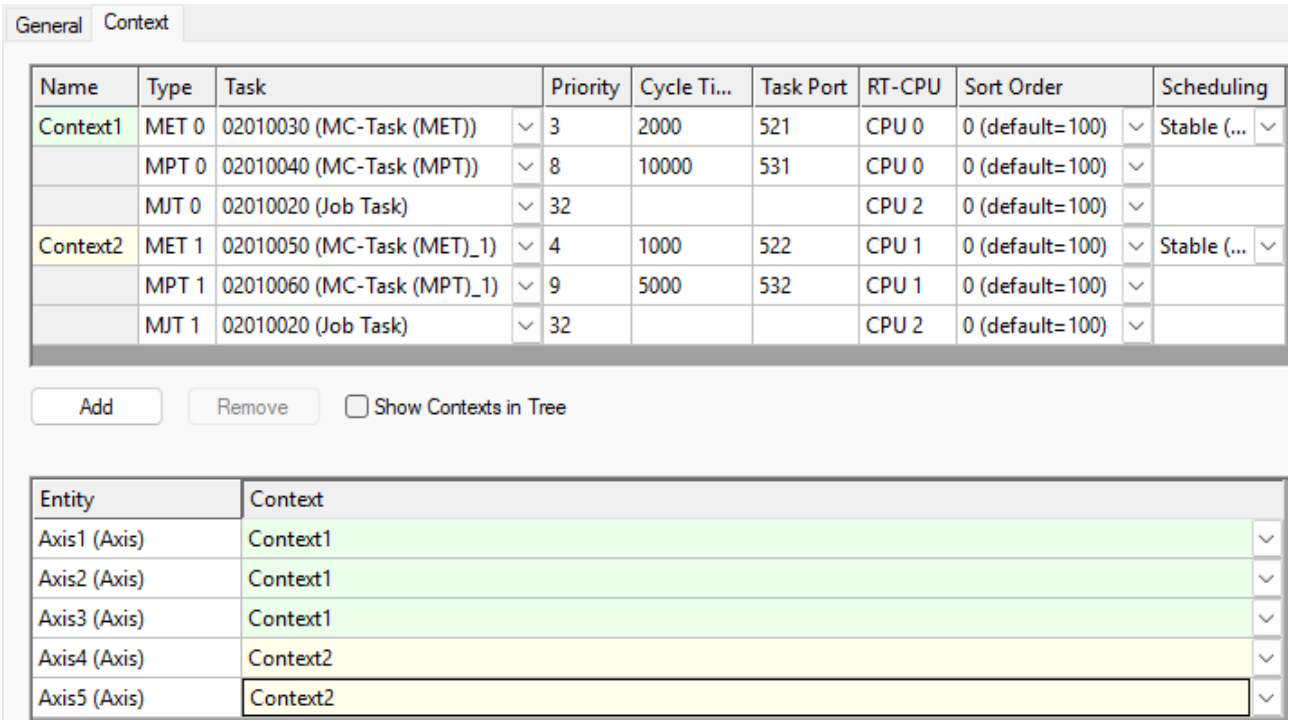


3. Assign tasks to the new context if no tasks have been created yet.
4. The tasks must be configured according to the intended use of the context and assigned to the appropriate CPU core.
5. In addition, axes must be assigned to the context; see Assign axes to a context [▶ 21].

3.2.1.3.2 Assign axes to a context

Requirements

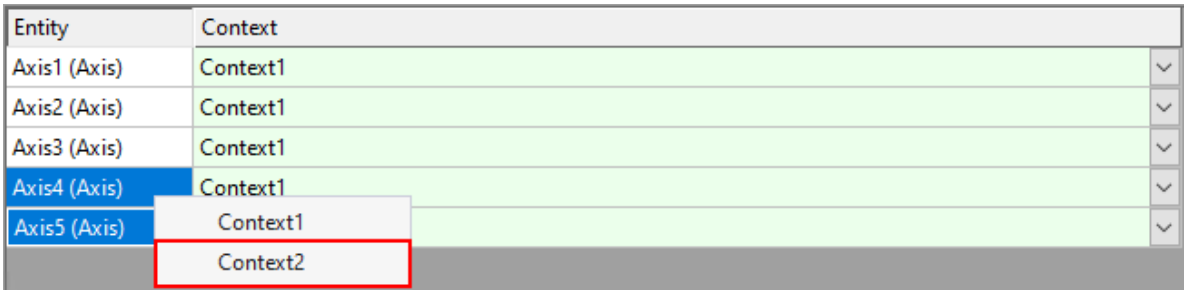
- An additional context has been added to the MC project; see Create a new context [▶ 20].
- There is at least one axis in the MC project; see Create axis object [▶ 25].
- Each axis must be assigned to a task context. By default, every Motion project contains one context, and all axes are assigned to this first context.
- If a Motion project contains more than one context, its axes can be explicitly assigned to different contexts. This setting can be applied to a single axis or to multiple axes simultaneously.



Option 1: Context tab - Multiple selection and right-click

1. In the axis list on the **Context** tab, select the axes that you want to assign to a different context.

2. Right-click the selection.
   ⇨ A context menu with the available contexts appears.

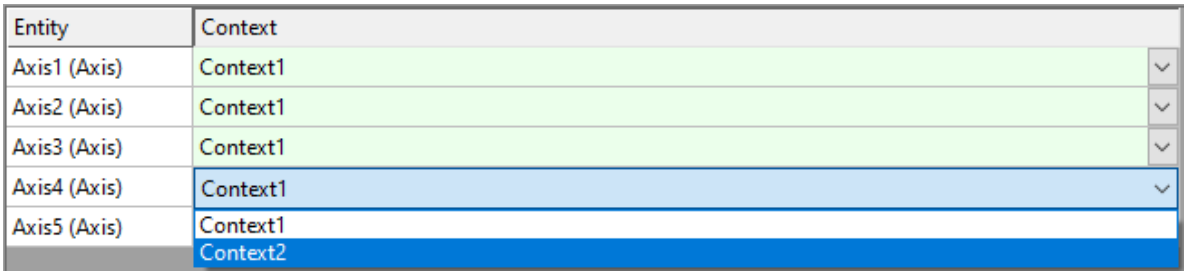


3. Select the required context.

⇨ All selected axes have been assigned to the selected context.

Option 2: Context tab - Individual selection using the combo box

1. In the axis list on the **Context** tab, double-click the context assigned to the axis that you want to change.
   ⇨ The combo box opens with a list of the available contexts.



2. Select the required context.

⇨ The axis has been assigned to the selected context. If multiple axes are to be assigned to the context, the process must be repeated for all axes.

Other variants:

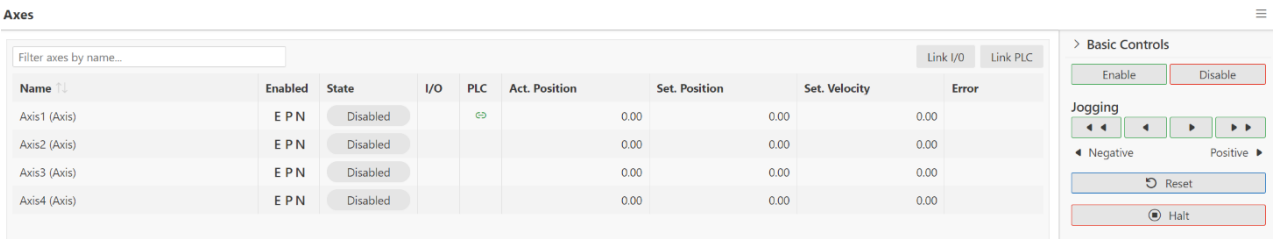
- Using the axis settings [▶ 27].

3.2.2 Axes

The **Axes** section displays an overview of all axes in the MC project. You can use this to enable axes and jog them.

If the project also includes movers, you can view an overview of all movers on the “Mover” tab. Further information on this topic can be found in the XPlanar documentation.

It is also possible to link the axis objects directly to the PLC and the hardware (I/O) on this page. Alternatively, you can link each axis under the Settings [▶ 27] tab.



Using the Axes UI

To enable or move axes from this view, select the desired axes. You can then control and operate the axes using the Sidebar.

You can minimize or collapse the Sidebar using the arrow in the upper-left corner and expand it to its normal size again.

Load Links

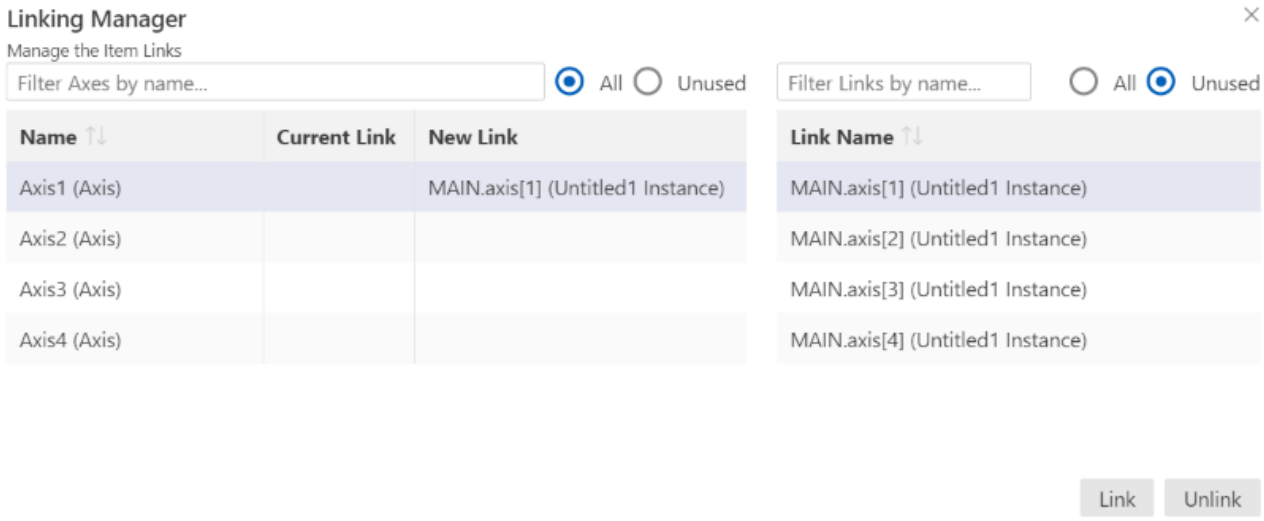
The **Load Links** button retrieves the current I/O and PLC links and displays them in the table.

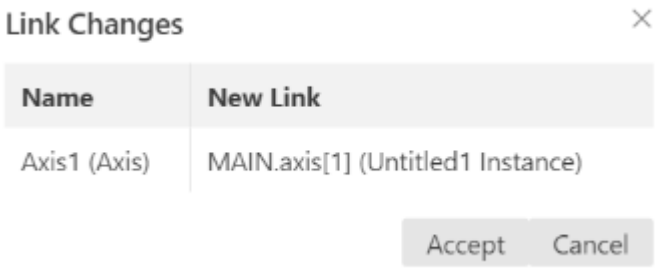
If an object is linked, the green  icon appears.

Link I/O and Link PLC – Linking Manager

Clicking on **Link I/O** or **Link PLC** opens the Link Manager. Using **Link PLC**, you can link axis objects to AXIS_REF instances in the PLC or unlink them again. **Link I/O** can be used to link axis objects to hardware components. The axes are linked in the order in which they are selected.

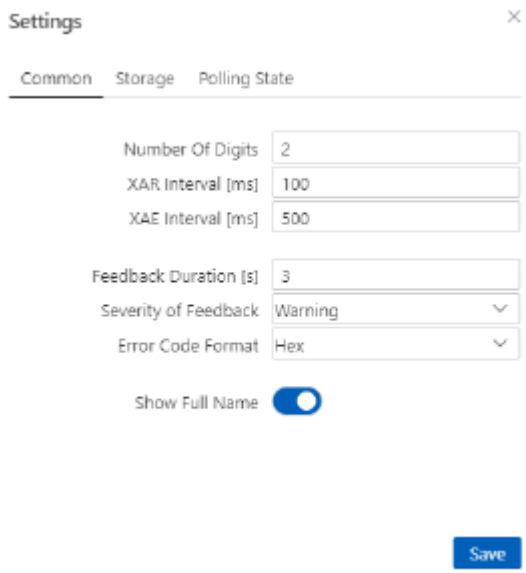
Example: In the system configuration (left table), select Axis2, Axis4, and Axis3. In the PLC configuration (right table), select MAIN.axis[4], MAIN.axis[2], and MAIN.axis[3]. This results in the following links: Axis2 + MAIN.axis[4], Axis4 + MAIN.axis[2], and Axis3 + MAIN.axis[3].





Settings

Under “Settings”, you can configure various settings for the Motion UI. Clicking **Save** saves the settings you have configured within the project.



Common

Selection	Description
Number Of Digits	Number of decimal places to display.
XAR Interval [ms]	Specifies the interval at which data is read from the runtime environment.
XAE Interval [ms]	Specifies the interval at which data is read from the engineering environment.
Feedback Duration [s]	Sets the duration for which the feedback window (pop-up) is displayed.
Severity of Feedback	Specifies from which level events are displayed in the feedback window.
Error Code Format	Options: Hex, Decimal Specifies the format in which the error code is displayed (as a hexadecimal or decimal number).
Show Full Name	If necessary, enlarges the axis name display so that it is shown in its entirety.

Storage unit

Selection	Description
Layouts and Inputs	Clicking **Clear** resets the layout to the default view and clears any entries made in the UI.
Settings	Clicking **Clear** clears the settings you have configured for this interface.

Polling State

The polling state displays data from the UI's last request to the Motion driver for diagnostic purposes.

Selection	Description
Polling State	Idle / Polling / No Connection / Error
Error message	Error message that appears if an error occurs in the communication between Motion UI and the driver.
Last Fetch Duration	Total duration of the UI's most recent request to the motion driver.
Last Bundled Response	Final response from the driver to the UI.

Zoom In/Out

You can zoom in and out by holding down the **[CTRL]** key on your keyboard and scrolling the mouse wheel at the same time.

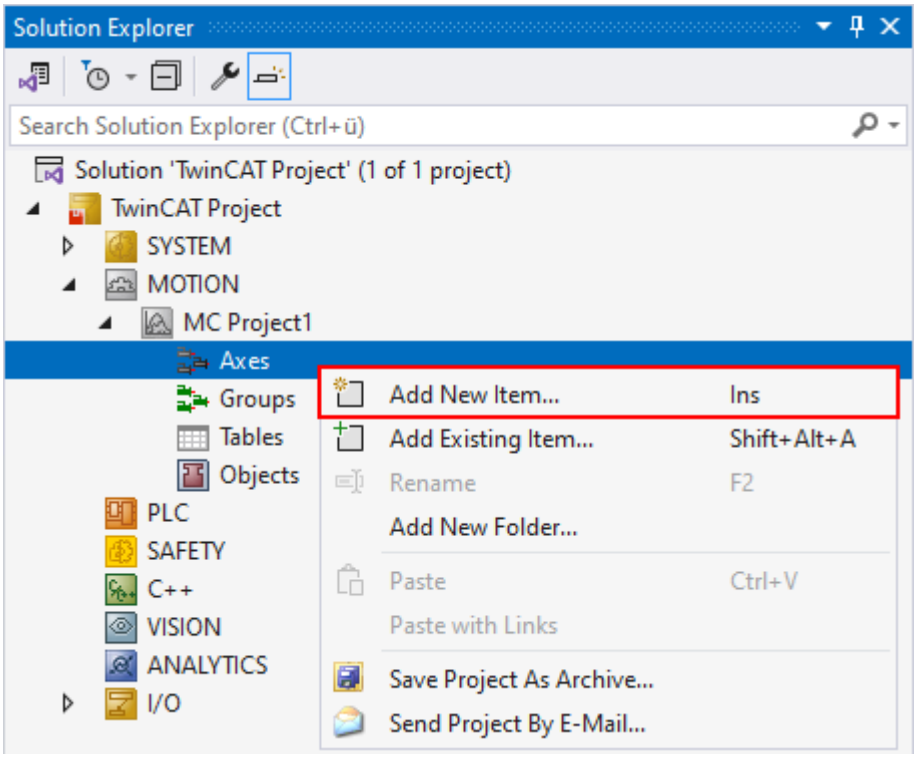
3.2.3 Default Axis

The following section contains information about the standard axis object. The various configuration options and available expansion modules are presented.

3.2.3.1 Create an axis object

✓ A TwinCAT project with an MC project has been created;
see Creating an MC project [▶ 17].

1. Right-click the node in the structure tree labeled **Axes > Add New Item…**



2. In the **Insert TcCOM Object** dialog box, select the axis type from the **Type** list and specify the desired number of axes under **Multiple**.

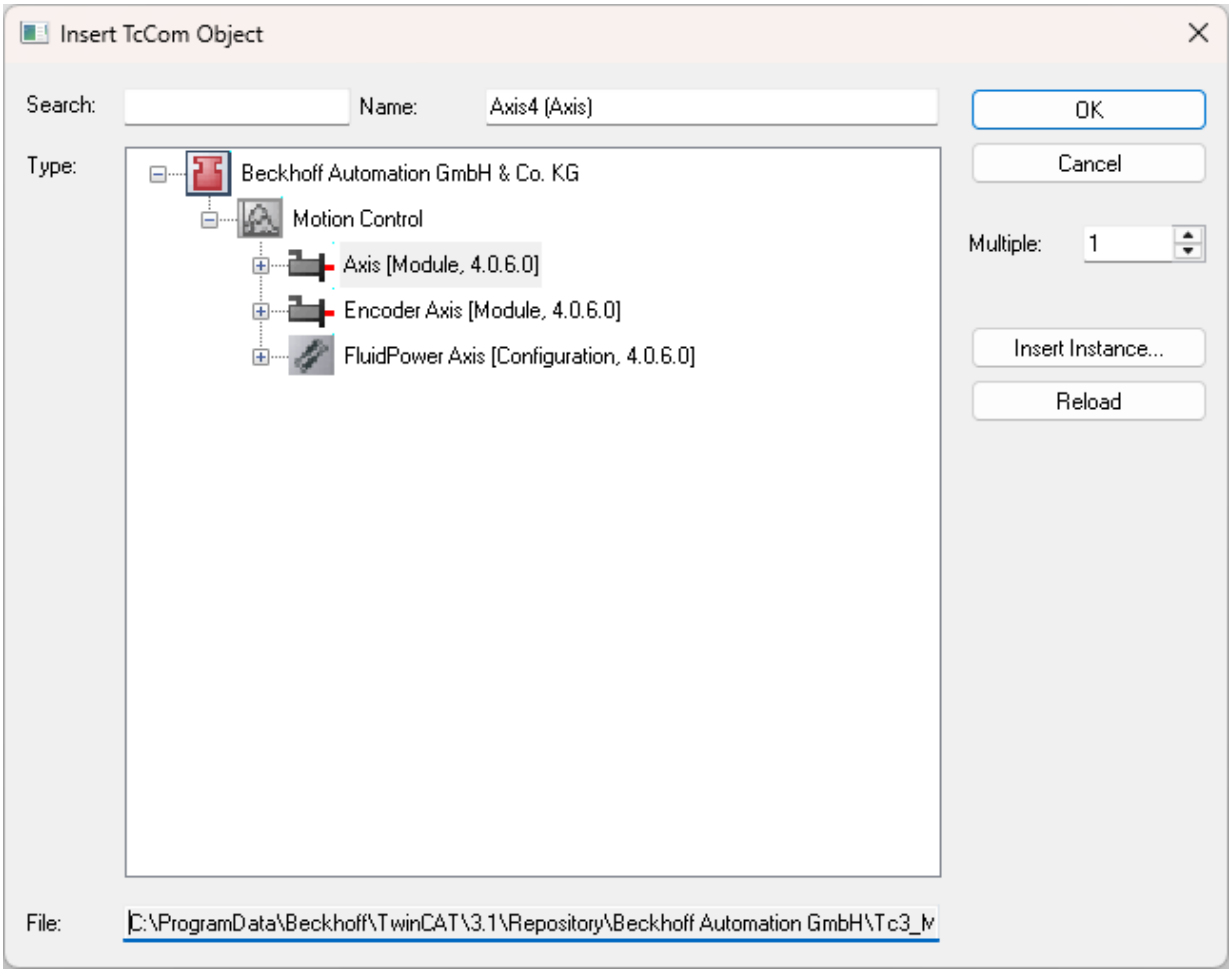
Axis
Standard axis

Encoder Axis
Encoder axis

FluidPower Axis
Special configuration of the standard axis, expanded with additional modules for use with FluidPower.

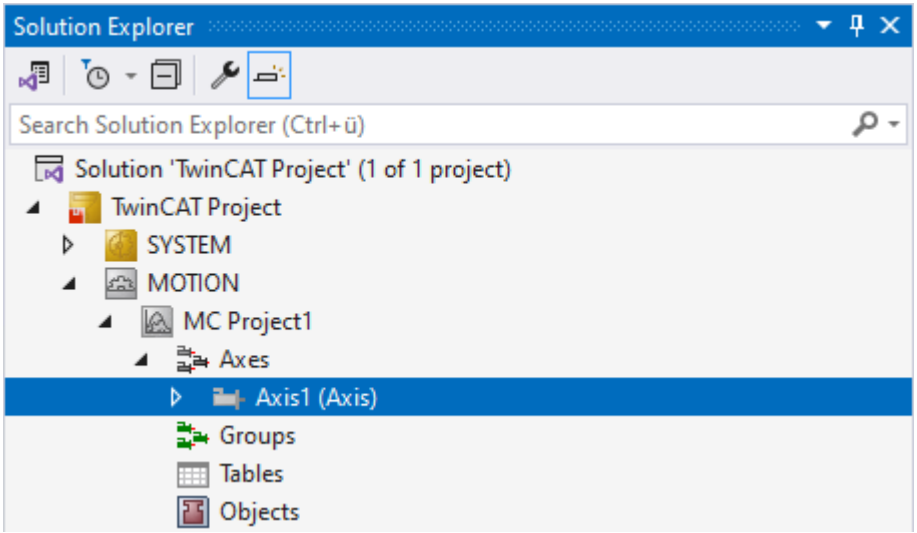
Planar Mover

For use with the Beckhoff XPlanar Mover



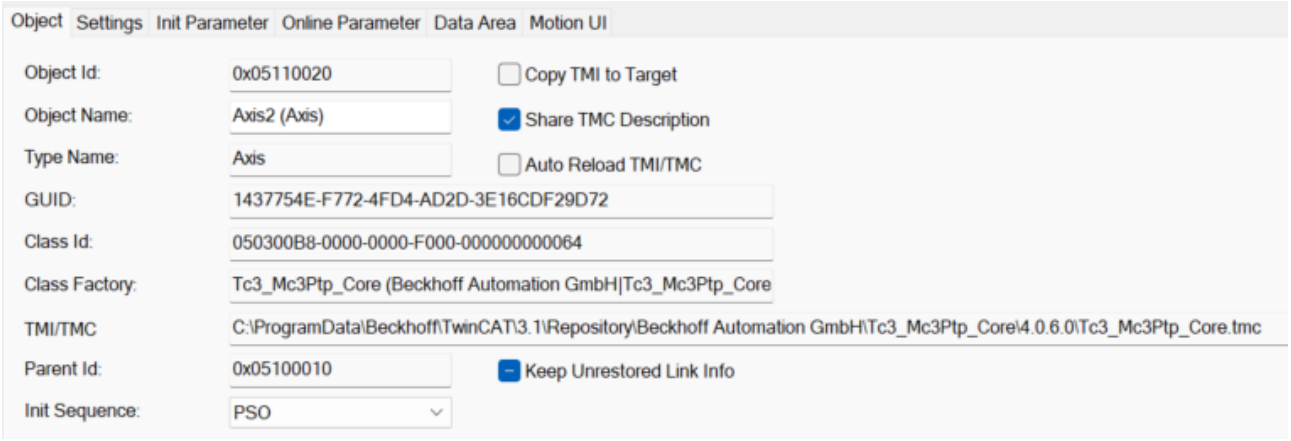
3. Confirm both by clicking the **OK** button.

⇨ The axis/mover has been created and is located below the Axes node. Until the axis type is selected, it is automatically a simulation axis , or, until it is linked to hardware, a simulation mover.
The axis type can be configured on the Settings tab [▶ 27] of the axis object.



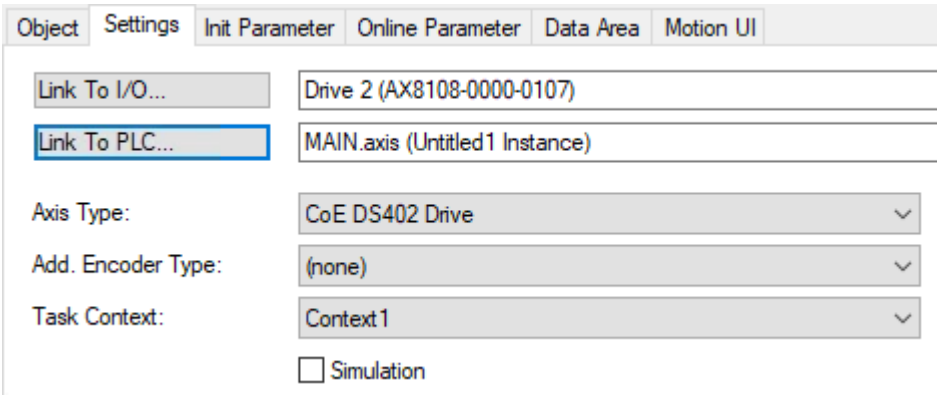
3.2.3.2 Object tab

The same properties apply to the Object tab as to the Instance node of the PLC.



3.2.3.3 Settings tab

In the Settings tab, you can configure key settings for the axis.



Property	Description
Link to I/O…	The Link button opens a dialog box for linking the MC axis to the drive hardware under the I/O node. The linked item is displayed in the field to the right of it.
Link to PLC…	The Link button opens a dialog box for linking the MC axis to a PLC instance of AXIS_REF. The linked item is displayed in the field to the right of it.
Axis type	Specifies the protocol for communicating with the connected drive hardware. Details on the axis types can be found in Drive modules [▶ 41].
Add. Encoder Type	Option to add an additional encoder to the axis. Details about the encoders can be found at Encoder modules [▶ 80]. Example: Use this when connecting an Analog Drive that does not have its own encoder system.
Task Context	Each axis must be assigned to a task context. If a MC Project [▶ 17] has more than one context, you can use the combo box to change the axis's context. New contexts are created and configured using the MC Project's Context tab [▶ 19].
Simulation	☑  If simulation mode is active, the link to the I/O is ignored, and a simulation object (drive and encoder) is used instead of the configured axis type.  If the Axis Type is set to (none), the axis is always a simulation axis, and the setting cannot be changed.

Property	Description		
	☐		The configured Axis Type and the link to the I/O are used.

3.2.3.4 Init Parameter tab

General

Name	Default value	Unit	Type
Simulation Mode	TRUE		BOOL
Position Unit	mm	[PosUnit]	MC.EPositionUnit [▶ 202]
Is Modulo Axis	FALSE		BOOL
Position Modulus	360.0	[PosUnit]	LREAL
Position Modulo Tolerance Window	0.0	[PosUnit]	LREAL

Simulation Mode

The Simulation Mode indicates whether the I/O connection to the drive and encoder should be active (`FALSE`) or simulated (`TRUE`). The parameter can be written at runtime if the axis is not enabled by the controller.

Position Unit

The Position Unit field is used for display purposes, such as on the commissioning interface. Inputs and outputs on function blocks are not converted.

Is Modulo Axis

Enable this parameter to use modulo functionality in function blocks for this axis. The setpoint and actual values are calculated as absolute and modulo values in the cyclic interface. To exchange Modulo information - for example, with the PLC - you must enable the Modulo data area [▶ 32] .

- **Position Modulus**
  Modulo value used to calculate the modulo position
- **Position Modulo Tolerance Window**
  Movement in the opposite direction to the specified direction is permitted, provided that the target position lies within this position tolerance of the command's starting position. This parameter affects the behavior of MC_MoveModulo [▶ 272]. It interacts with the setting selected at the `Direction` input of the function block.

Maximum Dynamics

Name	Default value	Unit	Type
Maximum Velocity	2000.0	[PosUnit]/s	LREAL
Maximum Acceleration	10000.0	[PosUnit]/s^2	LREAL
Maximum Deceleration	10000.0	[PosUnit]/s^2	LREAL
Maximum Jerk	100000.0	[PosUnit]/s^3	LREAL

Maximum Velocity

Maximum speed taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Velocity` input of a function block, the value defined here is used as the limit value for the velocity.

Maximum Acceleration

Maximum acceleration taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Acceleration` input of a function block, the value defined here is used as the limit value for acceleration.

Maximum Deceleration

Maximum delay taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Deceleration` input of a function block, the value defined here is used as the limit value for the delay.

Maximum Jerk

Maximum jerk taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Jerk` input of a function block, the value defined here is used as the limit value for jerk.

Default Dynamics

Name	Default value	Unit	Type
Default Acceleration	2000.0	[PosUnit]/s^2	LREAL
Default Deceleration	2000.0	[PosUnit]/s^2	LREAL
Default Jerk	4000.0	[PosUnit]/s^3	LREAL

Default Acceleration

This value is used as the acceleration at the `Acceleration` input on function blocks, unless it is explicitly set. If the constant `#Default` is specified at the input of a function block, the value defined here is used as the limit value for acceleration.

Default Deceleration

This value is used as the delay at the `Deceleration` input on function blocks, unless it is explicitly set. If the constant `#Default` is specified at the input of a function block, the value defined here is used as the limit value for the delay.

Default Jerk

This value is used as jerk at the `Jerk` input on function blocks, unless it is explicitly set. If the constant `#Default` is specified at the input of a function block, the value defined here is used as the limit value for jerk.

Manual Motion

Name	Default value	Unit	Type
Jog Velocity Slow	100.0	[PosUnit]/s	LREAL
Jog Velocity Fast	250.0	[PosUnit]/s	LREAL

Jog Velocity Slow

This target velocity is used for slow movement of an axis using manual control buttons. This procedure can be performed using `MC_Jog` as well as through the commissioning interface.

Jog Velocity Fast

This target velocity is used for rapid movement of an axis using manual control buttons. This procedure can be performed using `MC_Jog` as well as through the commissioning interface.

Monitoring

Name	Default value	Unit	Type
Position Limit Enforcement Enabled	FALSE		BOOL
Minimum Position	#Ignore	[PosUnit]	LREAL
Maximum Position	#Ignore	[PosUnit]	LREAL
Position Lag Monitoring Enabled	TRUE		BOOL
Maximum Position Lag Value	5.0	[PosUnit]	LREAL
Maximum Position Lag Filter Time	0.02	s	LREAL
Position Lag Calculation	HardwareProvidedIfAvailable		MC.EPositionLagSource [▶ 201]

Position Limit Enforcement Enabled

This parameter specifies whether position limits are taken into account when generating motion profiles. If the value is `TRUE`, the following two parameters must be set.

- **Minimum Position**
  : The minimum allowed position for the axis. Use `#Ignore` to disable enforcement of the position limit in this direction of motion. It is not possible to move an axis beyond this limit. If an axis is outside its position limits when TwinCAT starts up, it can move toward the valid position range. Commands with a target position outside this lower limit will be rejected.
- **Maximum Position**
  : The maximum allowable position for the axis. Use `#Ignore` to disable enforcement of the position limit in this direction of motion. It is not possible to move the axis beyond this limit. If an axis is outside its position limits when TwinCAT starts up, it can move toward the valid position range. Commands with a target position outside this lower limit will be rejected.

Position Lag Monitoring Enabled

This parameter specifies whether the position lag for the axis should be monitored. If the value is `TRUE`, the following two parameters are displayed for configuring position and time. If the parameterized limits are exceeded, a runtime error is output.

The calculation is performed as follows:

Position lag value = current target position - actual position

- **Maximum Position Lag Value and Maximum Position Lag Filter Time**
  The `Maximum Position Lag Value` is the position lag value that must not be exceeded for longer than the specified time (`Maximum Position Lag Filter Time`). Otherwise, the axis is stopped immediately by direct shutdown, and an error is generated.

Position Lag Calculation

This parameter specifies whether the position lag is calculated by the software from the MC axis (`InternallyCalculated`) or read directly from the linked hardware (`HardwareProvidedIfAvailable`). If `HardwareProvidedIfAvailable` is set but no hardware is available, the system falls back to the internally calculated value. For a simulation axis, `HardwareProvidedIfAvailable` uses data from the simulation drive.

Error event handling

Name	Default value	Unit	Type
Control Error Setpoint Mode	HardStop		MC.EControlErrorSetpointMode [▶ 196]

If there is a hardware error (e.g., in the control loop) and the MC no longer has control over the axis, you can configure the behavior of setpoint generation here. `HardStop` causes the axis to stop abruptly in the event of a fault.

Diagnostics

Name	Default value	Unit	Type
TraceLevel	tlWarning		TcTraceLevel

Specify which messages should be sent to the TwinCAT error list.

Additional Encoder

Name	Default value	Unit	Type
Main Position Encoder			OTCID_PositionEncoder
Load Control Type	None		MC.ELoadType [▶ 200]

Main Position Encoder

If an additional encoder is available, select its object ID here.

Load Control Type

Set the type of load to be controlled here.

3.2.3.5 Online Parameter tab

The following online parameters are available:

Monitoring

Name	Unit	Type
Minimum Position Lag	[PosUnit]	LREAL
Maximum Position Lag	[PosUnit]	LREAL
Trigger Minimum/Maximum Position Lag Reset		Trigger Minimum/Maximum Position Lag Reset

Minimum Position Lag

This read-only online parameter displays the minimum detected position lag for informational purposes.

Maximum Position Lag

This read-only online parameter displays the maximum detected position lag for informational purposes.

Trigger Minimum/Maximum Position Lag Reset

A rising edge at this parameter input resets the stored minimum and maximum position lags.

Additional Error Event Information

Name	Unit	Type
Originating Error Event Sender ID		OTCID
Original Error Id		HRESULT

Originating Error Event Sender ID

If a subsequent error occurs - for example, due to coupling - this parameter displays the object ID of the axis from which the error originated.

Original Error Id

If a subsequent error occurs - for example, due to a coupling - this online parameter displays the original error number.

Override

Name	Unit	Type
Velocity Override Factor		LREAL

Velocity Override Factor

The current velocity override for the axis is displayed here.

3.2.3.6 Data Area tab

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

NOTICE

Unlinked CDTs can cause a floating-point exception when accessed

Comparisons with NaN values can lead to an exception, which results in the runtime stopping and potentially causing machine damage.
Valid data can only be expected if `IsConnected = TRUE`. Otherwise, there is a risk of an inconsistent or invalid data set, which may lead to unpredictable consequences and unexpected behavior. Arithmetic operations using invalid values can cause floating-point exceptions in the processor's floating-point unit.

- Switch off the Floating Point Exceptions if NaN values are to be used and processed in the application.
- Before accessing a CDT (e.g., from the PLC), check the CDT's `IsConnected`.
- Familiarize yourself with how to handle named constants and NaN values.

PlcToMc (Input)

Enable (default)	Name	Type	Description
☑	Std	POINTER TO MC.CDT_PLCTOMC_AXIS_STD [▶ 210]	Standard cyclic data structure for enabling the axis and setting the velocity override.

McToPlc (Output)

Enable (default)	Name	Type	Description
☑	Std	POINTER TO MC.CDT_MCTOPLC_AXIS_STD [▶ 207]	Standard cyclic data structure containing the axis Oid and key status information.
☑	Set	POINTER TO MC.CDT_MCTOPLC_AXIS_SET [▶ 206]	Optional cyclic data structure containing information about the axis's current setpoints. ***Notice*** **Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)
☑	Act	POINTER TO MC.CDT_MCTOPLC_AXIS_ACT [▶ 204]	Optional cyclic data structure containing information about the axis's current actual values.

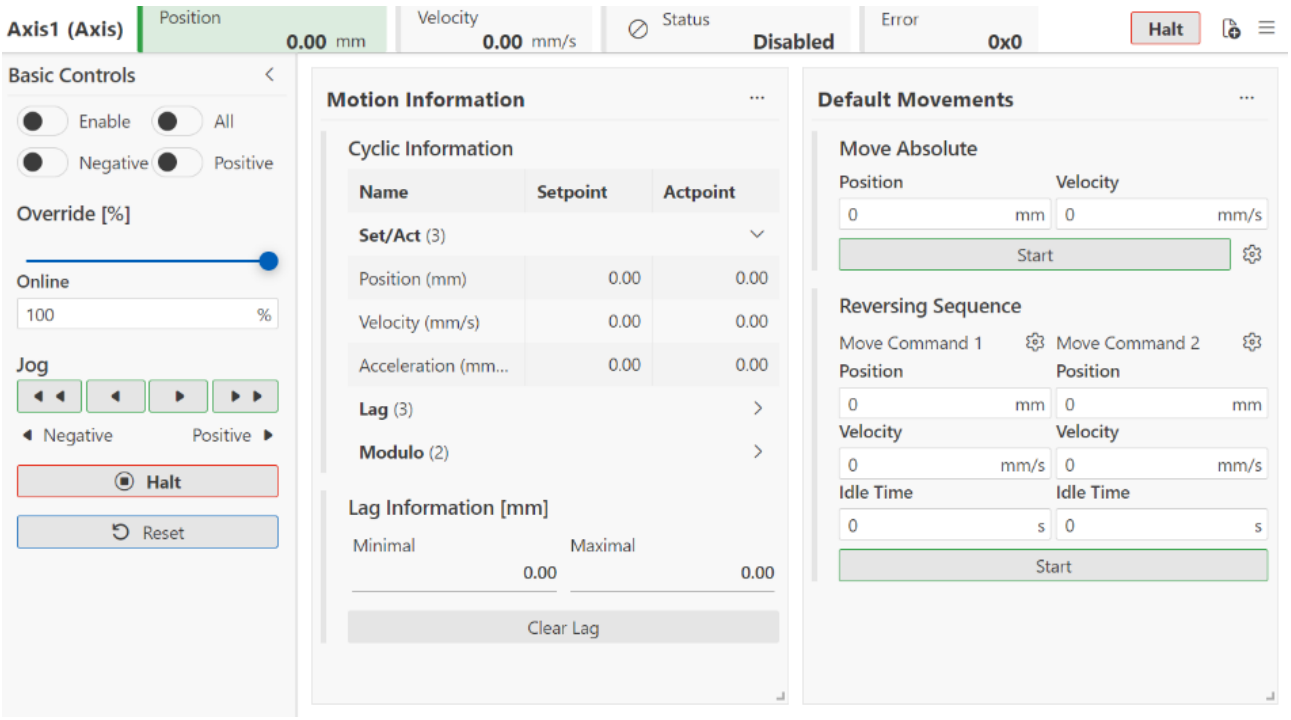
Enable (default)	Name	Type	Description
			Notice **Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)
☐	Modulo	POINTER TO MC.CDT_MCTOPLC_AXIS_MODULO [▶ 206]	Optional cyclic data structure containing information about the axis's module position. To calculate the module position accurately, set the **Is Modulo Axis** parameter to TRUE in the Axis Parameters (Init) [▶ 28] section, and configure the **Position Modulus** and **Position Modulo Tolerance Window** parameters accordingly. **Notice** **Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)
☐	LoadControl	POINTER TO MC.CDT_MCTOPLC_AXIS_LOADCONTROL [▶ 205]	Optional cyclic data structure. **Notice** **Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)
☐	Fluidpower	POINTER TO MC.CDT_MCTOPLC_AXIS_FLUIDPOWER [▶ 205]	Optional cyclic data structure. **Notice** **Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)

3.2.3.7 Motion UI tab

The **Motion UI** tab contains a commissioning interface (Motion UI) for the axis, intended for commissioning and testing purposes. It provides basic commands as well as information about the axis and its state. Motion UI can be customized to suit individual users.

In this documentation, Motion UI is divided into two sections:

1. Widget bar with status information and quick access icons at the top
2. Cards in the main section



3.2.3.7.1 Setting options

In the upper-right corner, there are controls that allow you to personalize and customize the interface.

Icon	Meaning
	You can use this icon to add new cards to the main section.
	This icon opens the menu with advanced settings.

Menu 

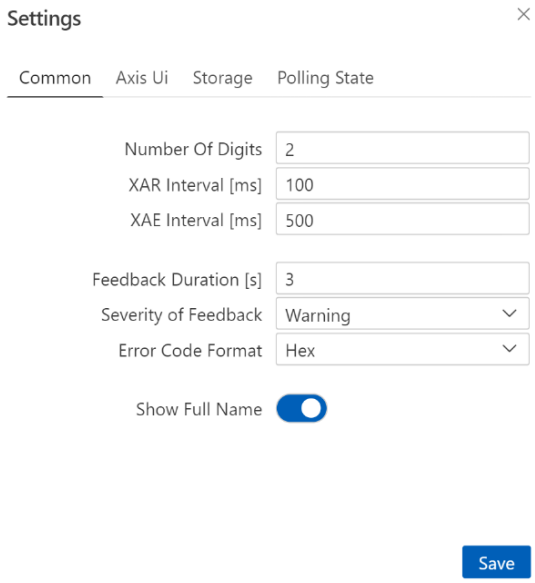
The menu offers various actions and settings:

Selection	Description
Reconnect	This reestablishes the connection between the UI and the object to be displayed and reloads the data.
Settings	This opens the settings.
Open Widget Catalog	Opens the widget catalog for the status bar.
Save Layout	Saves the current UI layout. You must enter a custom name. The saved layout can only be used in the current project.
Apply Layout	Applies an existing layout to the current view. The following layouts are predefined by default: • Default, • Modulo, • Fluid Power. Additional layouts that have been saved using **Save Layout** are also available here.
Layout Manager	Opens the Layout Manager.
Reset Layout	Resets the UI so that the default layout is used again.

Selection	Description
Theme	You can choose from the following themes for the color scheme: • System, • Light, • Dark.

Settings

Under “Settings”, you can configure various settings for the Motion UI. Clicking **Save** saves the settings you have configured within the project.



Common

Selection	Description
Number Of Digits	Number of decimal places to display.
XAR Interval [ms]	Specifies the interval at which data is read from the runtime environment.
XAE Interval [ms]	Specifies the interval at which data is read from the engineering environment.
Feedback Duration [s]	Sets the duration for which the feedback window (pop-up) is displayed.
Severity of Feedback	Specifies from which level events are displayed in the feedback window.
Error Code Format	Options: Hex, Decimal Specifies the format in which the error code is displayed (as a hexadecimal or decimal number).
Show Full Name	If necessary, enlarges the axis name display so that it is shown in its entirety.

Axis UI

Selection	Description
Allow PLC Override	This option is disabled by default. In that case, axis limits and overrides cannot be changed in the UI as long as the axis is controlled by the PLC. If this option is enabled, the user can override the axis enable settings and the override in the user interface.
Allow Keyboard Control	This option is disabled by default. In that case, axes cannot be moved using the keyboard. If this option is enabled, you can jog the axes using the arrow keys. Pressing the spacebar stops the axis.

Storage unit

Selection	Description
Layouts and Inputs	Clicking **Clear** resets the layout to the default view and clears any entries made in the UI.
Settings	Clicking **Clear** clears the settings you have configured for this interface.

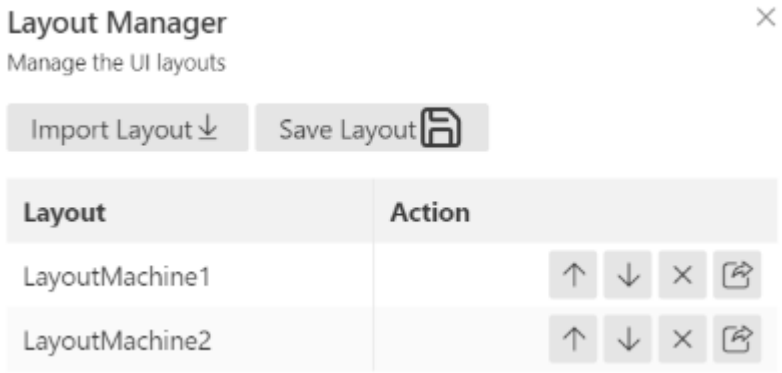
Polling State

The polling state displays data from the UI's last request to the Motion driver for diagnostic purposes.

Selection	Description
Polling State	Idle / Polling / No Connection / Error
Error message	Error message that appears if an error occurs in the communication between Motion UI and the driver.
Last Fetch Duration	Total duration of the UI's most recent request to the motion driver.
Last Bundled Response	Final response from the driver to the UI.

Layout Manager

The Layout Manager displays your saved layouts and offers additional options.



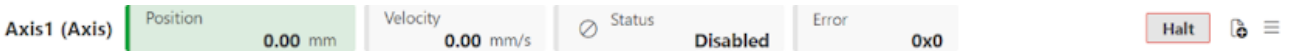
Selection	Description
Import Layout	Imports an existing layout into the project.
Save Layout	Saves the current UI layout within the project. Another window will open so you can enter a name. If an existing layout name is used, the existing layout will be overwritten.
↑ ↓	Changes the order of your layouts. You can use the arrow keys to move a layout up or down in the list.
×	Deletes the corresponding custom layout.
↥	Export the layout you created so that it can be used in other projects as well. Next, you must specify the storage location.

Zoom In/Out

You can zoom in and out by holding down the **[CTRL]** key on your keyboard and scrolling the mouse wheel at the same time.

3.2.3.7.2 Widget Bar

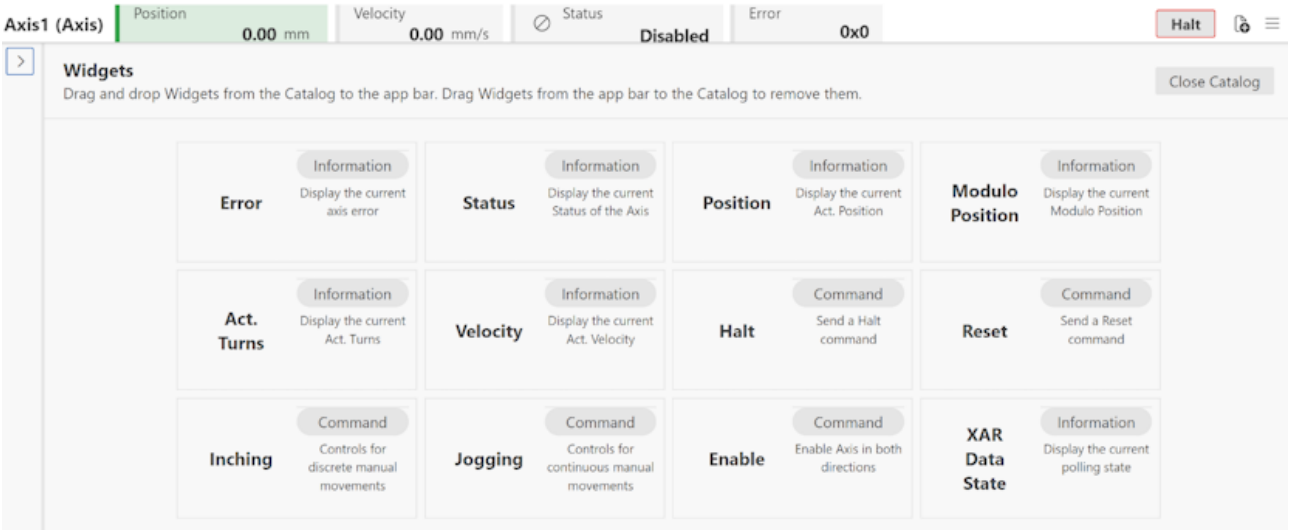
The widget bar is located at the top of the Motion UI. The standard layout already includes some information there, such as the axis name, axis position, and axis status. In addition, the stop button, which stops the axis, is located on the right.



The bar is divided into a left and a right half. When the available width is reached, the left half of the status bar is automatically expanded by one additional line.

Customizing the Widget Bar

You can customize the widget bar; both halves can be arranged as you like. To do this, select **Open Widget Catalog** from the menu. You can add and remove widgets from the toolbar using drag and drop.



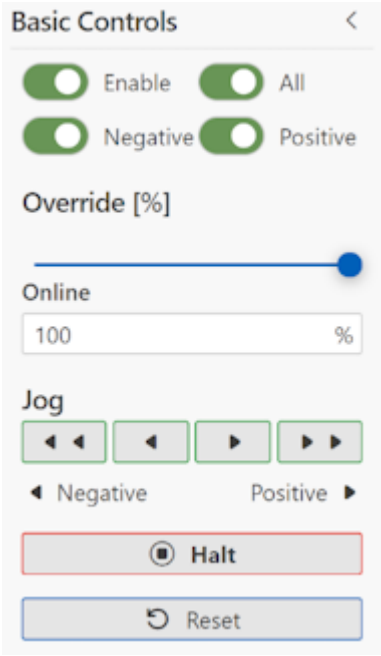
The following widgets are available:

Widget	Description
Error	Displays the current axis error.
Status	Displays the current axis status.
Position	Displays the current axis position (Act Pos).
Modulo position	Displays the current axis position in the modulo range.
Act. Turns	Displays the current revolutions (modulo).
Velocity	Displays the current axis velocity (Act Velo).
Halt	Pressing the **Stop button** aborts the current motion, and the axis transitions to standstill.
Reset	If an axis error is present, it can be cleared using **Reset**.
Inching	If this control is integrated, it can be used to manually initiate discrete (incremental) movements.
Jogging	If this control is integrated, it can be used to manually initiate continuous movements (= the axis is jogged).
Enable	The **Enable** option allows the axis to be enabled in both directions.
XAR Data State	Displays the current polling state. The polling state indicates whether Motion UI is receiving current values (polling) or whether the connection to the driver has been interrupted.

3.2.3.7.3 Side Bar

The sidebar is located on the left edge of Motion UI. It contains the Basic Controls and can be collapsed by clicking the small arrow labeled < in the upper-right corner.

The sidebar cannot be customized, and its content is predetermined.



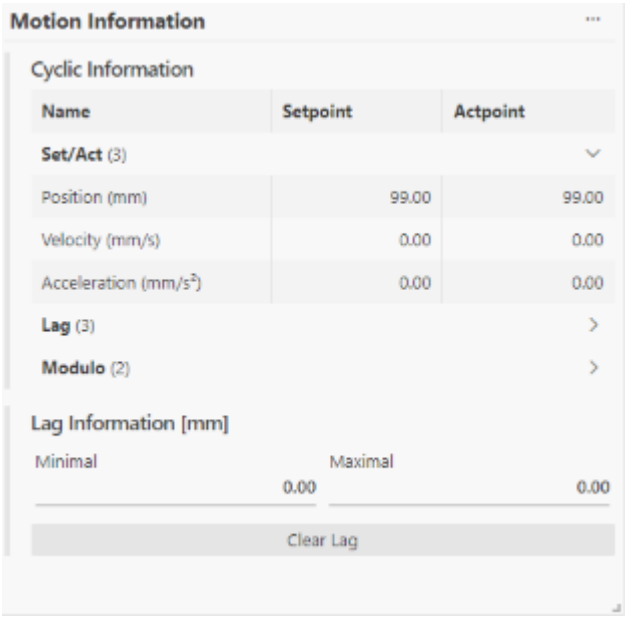
Manual movement via keyboard controls

You can also use the keyboard instead of the jog buttons. The user must first enable this option in the settings. The following keys are assigned to the actions:

Action	Optional keyboard operation
Slow movement in the positive direction using the manual control buttons	→ (Right arrow key)
Rapid movement in the positive direction using manual control buttons	↑ (Up arrow key)
Slow movement in the negative direction using the manual control buttons	← (Left arrow key)
Rapid movement in the negative direction using the manual control buttons	↓ (Down arrow key)
Halt	Space bar

3.2.3.7.4 Cards

The various dialogs in the main section of Motion UI are referred to as cards. In the standard layout, this allows you to set axis enable flags and perform simple motion sequences. In addition, information on the current dynamic values, for example, is listed.



Moving and resizing the cards

You can change the position of the cards on the page using drag and drop. To do this, click on the card's header/title bar and, while holding down the mouse button, select a new position.

You can also change the size of each card by dragging and dropping the bottom-right corner of the card.

Customizing card content

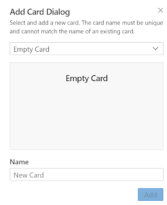
You can name and fill the cards however you like, and you can manually adjust the order of the content. To switch to edit mode for the cards, use the three dots in the upper-right corner to open the card menu

and select  . If this option is selected, the text in the menu is highlighted. The individual sections of the card can now be rearranged using the displayed arrows  . In addition, individual sections can be easily removed from the card using the trash can icon  .

You can also edit the card name by clicking the pencil icon  , which appears next to the card name in edit mode. To exit edit mode, click **Edit Card** again in the menu.

Add more cards

You can add more cards to the Motion UI via  . The user can choose between a predefined standard card type or a blank card, which they can then fill with content themselves. Use the arrow keys to switch between the predefined card types in the selection.



Under “Name”, the user can give the card a title.

3.2.3.7.5 Moving an axis using Motion UI

⚠ DANGER

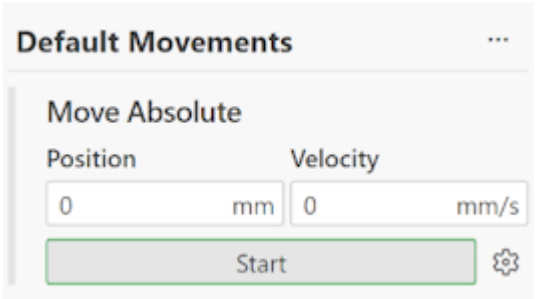
This example requires simulation axes - improper use of real axes can result in fatal or serious injury!

This example creates an axis movement. When using real axes, different safety requirements apply. There is a risk of fatal or serious injury due to unintended movement of real axes.

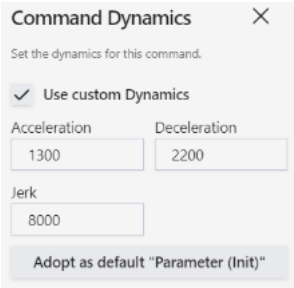
- In any case, make sure that neither you nor others can be harmed by the movement, e.g. by maintaining a suitable safety distance.
- Do not perform any action whose consequences you cannot estimate.

✓ An axis object has been created;
see Create an axis object [▶ 25].

1. Activate the TwinCAT project.
2. Enable the axis using **All** in the Enable dialog.
   ⇨ The sliders in the Enable dialog should now be highlighted in green. In addition, the axis status in the widget bar should be highlighted in green and display the status `Standstill`.
3. Move the axis by clicking and holding one of the jog buttons  in the sidebar.
   ⇨ While you hold down the button, the axis state `Discrete Motion` should be displayed. The displayed position changes as the object moves, and the current velocity is shown.
4. Now perform an absolute movement. To do this, you need to enter values in the Move Absolute dialog box. For example, enter the value “1000” for **Position** and the value “750” for **Velocity**.



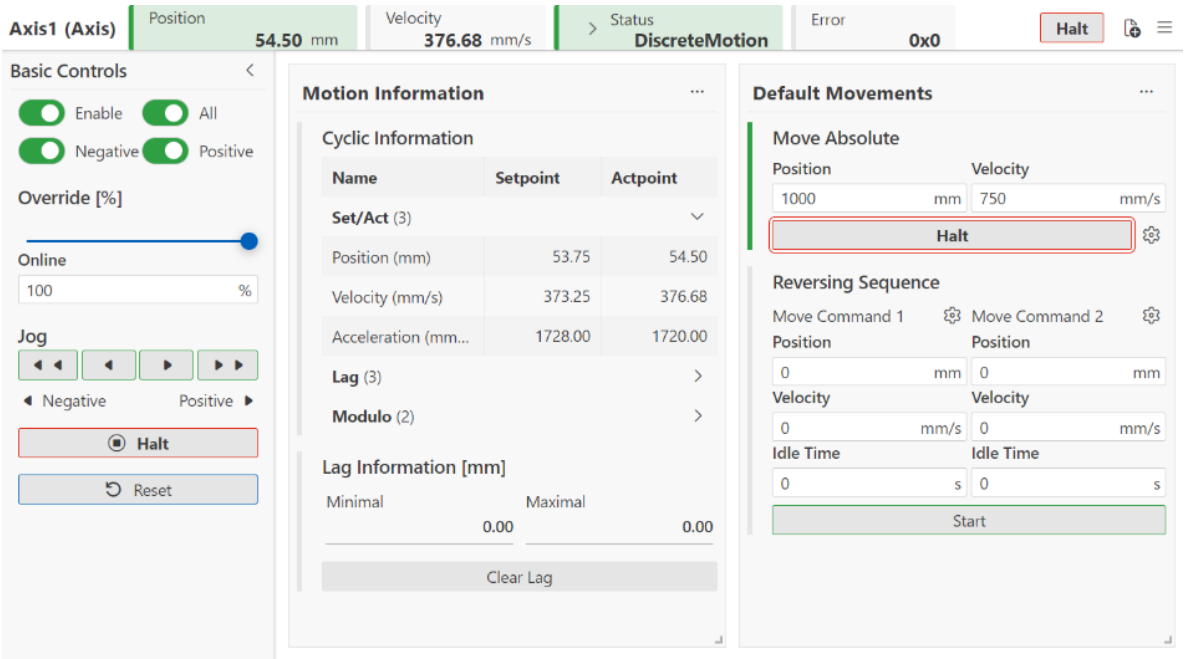
5. To specify motion dynamics, you can use the gear icon  to open additional settings and configure the parameters.



   ⇨ If user-defined parameters are stored for a command, the gear icon turns green:  . Clicking the **Adopt as default 'Init Parameter'** button applies the parameters defined here to the axis's parameter list, overwriting any settings previously configured there.

6. Click the **Start** button to begin the movement.

   ⇨ The green bar on the dialog box indicates that the movement is active. During the motion, the axis state is displayed as `Discrete Motion`.
   The current position changes, and the velocity is also displayed.



7. Click the **Stop** button to stop the axis from moving. This aborts the motion command.

3.2.3.8 Drive modules

Each axis is equipped with an Simulation Drive [▶ 42] module. This allows the axis to be used without real drive hardware. By default, this module is hidden. Configuration is always performed directly through the axis parameters.

As soon as an axis type is selected on the axis Settings tab [▶ 27], the axis has an additional drive module that serves as an interface to the real drive hardware. This drive module is also hidden by default. As long as it is hidden, parameterization is performed via the axis Init Parameter tab [▶ 28], and the cyclic interface to the hardware is configured via the axis Data Area tab [▶ 32].

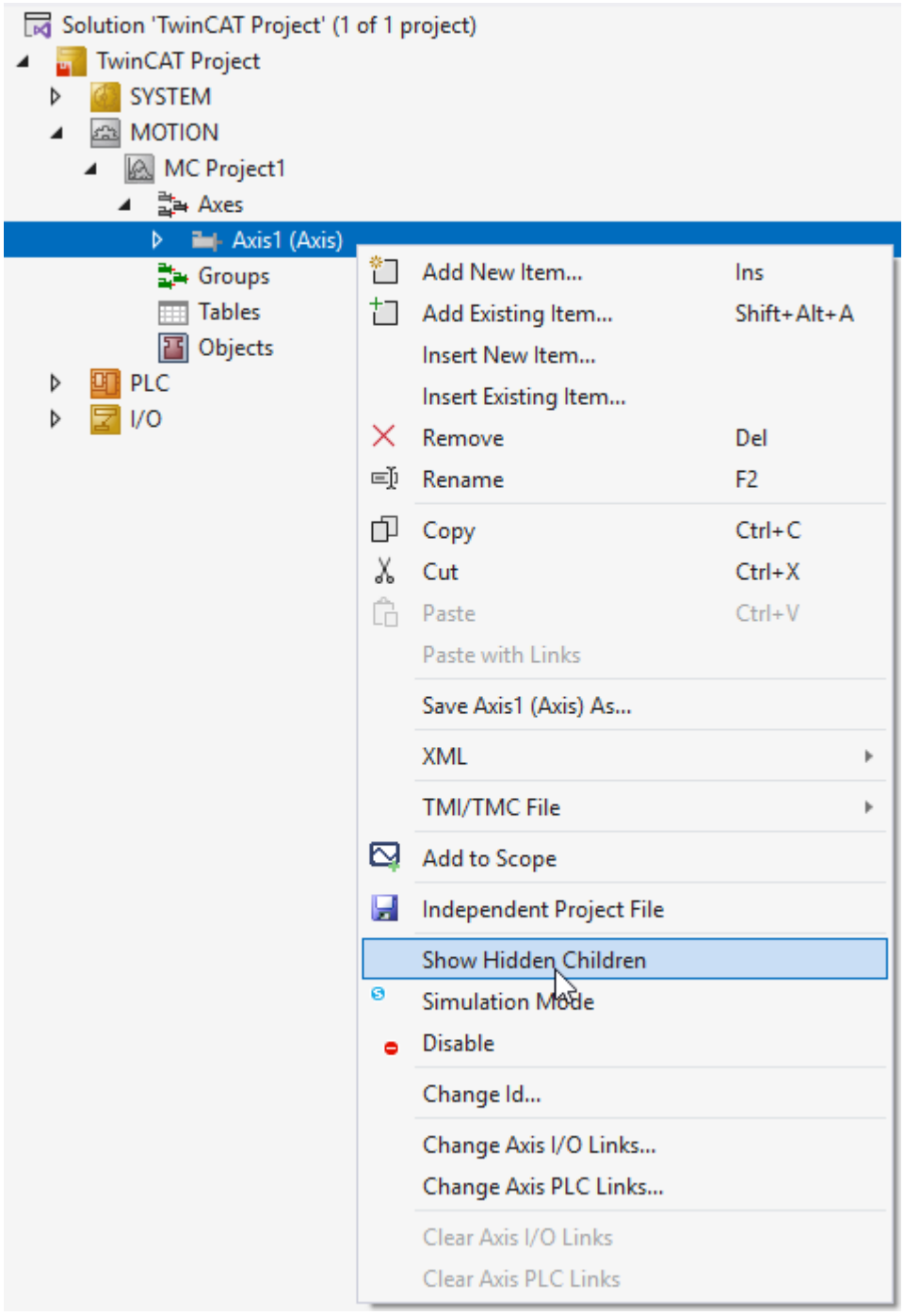
When the drive module is displayed, parameterization and configuration are performed via the drive module.

You can find the available drive modules in the following sections of this chapter.

Show drive modules

✓ The drive modules for an axis are hidden by default.

1. Right-click on **Axes > Axis > Show Hidden Children** to display them.



3.2.3.8.1 Simulation Drive

Each axis is equipped with a simulation drive module. This allows the axis to be used without real drive hardware. By default, this module is hidden. Configuration is always performed directly using the axis parameters.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.1.1 Init Parameter

Name	Default value	Unit	Type
Encoder Mounting Offset	0.0	[PosUnit]	LREAL

Encoder Mounting Offset

`Encoder Mounting Offset` specifies the permanent correction value for the physical encoder alignment. Set this value during commissioning.

3.2.3.8.1.2 Online Parameter

Scaling

Name	Unit	Type
Work Zero Offset	[PosUnit]	LREAL
Active Homing Offset	[PosUnit]	LREAL

Work Zero Offset

Non-persistent offset that can be set using the MC_SetPosition function block. If MC_SetPosition is executed multiple times, the individual offsets are combined, taking the sign into account.
The offset is active at runtime and is reset when TwinCAT is restarted.

```
Work Zero Offset = Vorgegebene neue Position - Aktuelle Sollposition
```

Active Homing Offset

Displays the temporary offset calculated during the last homing procedure. This value is not permanently saved.

3.2.3.8.2 MDP742 Drive

Module for linking an MC3 axis to a drive with the MDP742 drive profile.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.2.1 Init Parameter

General

Name	Default value	Unit	Type
Position Interpretation	Incremental		MC.EEncoderInterpretation [▶ 198]

Name		Default value	Unit	Type
	Maximum Absolute Counter Value (Visibility depends on position interpretation)	1048575	INC	UDINT
Position Data Range		FullRange		MC.EEncoderDataRange [▶ 198]
	Minimum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 0 • signed: -2147483648	INC	LINT
	Maximum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 4294967295 • signed: 2147483647	INC	LINT
Feedback Velocity Source		HardwareProvidedIfAvailable		MC.EEncoderFeedbackVeloSource [▶ 198]

Position Interpretation

The `Position Interpretation` parameter defines the semantics according to which the values are transmitted by the position encoder and interpreted by the MC3 axis.

`Incremental`

The `Incremental` setting does not use the absolute properties of the physical feedback system, but can be used with both incremental and absolute encoder systems. After starting the controller, homing is required to initialize the actual position of the axis (`IsPositionValid = FALSE`).

`AbsoluteFullRange`

The `AbsoluteFullRange` setting is typically used with multi-turn or single-turn absolute encoders. The MC3 axis takes into account the entire data range of the data type in the I/O mapping (FullRange). The maximum number of increments corresponds to the upper limit of the data type. The size of the data type is specified in the protocol used.

When the controller is powered on or restarted and position limits are enabled, the axis signals `IsPositionValid = TRUE`. Configuring and enabling the software-based position limits for the axis allows the axis position to be restored after a controller restart. This applies to a maximum of one overflow of the transmitted increments. The position limits must be selected such that the resulting possible travel range of the axis is less than the encoder's counting range (FullRange).

If the position limits are disabled, the axis signals this with `IsPositionValid = FALSE`. A homing run is not required for AbsoluteFullRange.

`AbsolutUserDataRange`

The same boundary conditions apply as for the setting. `AbsoluteFullRange`. In addition, the parameter `Maximum Absolute Counter Value` is displayed and taken into account:

- `Maximum Absolute Counter Value`
  Unique maximum number of increments that the encoder system transmits (within the limits of the data type used). For example, the default value is set to 2^20-1.

`AbsoluteModulo`

The position increments are transmitted by the I/O device using modulo logic. For example, if one revolution of the encoder system corresponds to the modulus of the MC3 axis, the position is unique within this range. The axis indicates `IsPositionValid = TRUE`.

Position Data Range

The `Position Data Range` parameter defines the value range of the encoder system that is used for position calculation. This refers to the raw value received from the drive or encoder via the process data (PDO mapping) and enables the detection and handling of overflows. The parameter can only be changed at configuration time, not at runtime.

`FullRange`

Position overflow is based on the data type limits of the respective encoder system. For a signed 32-bit encoder system (e.g., CoE DS402 Drive (AX8000), SoE Drive (AX5000), EtherCAT encoder), this implicitly means a value range from -2147483648 to 2147483647. For an unsigned 32-bit encoder system (e.g., MDP742 Drive (selected ELM72xx), SSI encoder), this implies a value range from 0 to 4294967295.

`UserDefined`

This parameter value allows you to manually enter the range limits used to determine an overflow. Two additional parameters for entering the upper and lower limits will be displayed:

- Minimum Counter Value
- Maximum Counter Value

Feedback Velocity Source

Specifies the source of the actual velocity (AXIS_REF.McToPlc.Act.Velocity). The velocity can either be obtained from the hardware being used or calculated internally by differentiating the actual position value.

`HardwareProvidedIfAvailable` (Default value)

Uses the velocity provided by the hardware. If no entry for the actual velocity is mapped in the process data, it is automatically calculated internally.

`InternallyCalculated`

Velocity is always calculated from the derivative of the actual positions. Actual hardware velocities are ignored, even if they are available.

`HardwareProvided`

The velocity is always taken directly from the hardware feedback. This requires linking the actual velocity via process data (PDO mapping). If the mapping is missing, this will result in an error when the configuration is started.

Scaling

Name	Default value	Unit	Type
Position Scaling Numerator from Hardware	360	[PosUnit]/INC	LREAL
Position Scaling Denominator from Hardware	1048576.0		LREAL
Encoder Mounting Offset	0.0	[PosUnit]	LREAL
Velocity Scaling Numerator to Hardware	1.0	INC/([PosUnit]/s)	LREAL
Velocity Scaling Denominator to Hardware	1.0		LREAL
Direction Inverted	`FALSE`		BOOL
Torque Scaling from Hardware	0.1	INC/([TorqueUnit])	LREAL

The position scaling factor is divided into a numerator and a denominator. It is used to convert the path increments into axis positions or to calculate a user unit from encoder increments.

Position Scaling Numerator from Hardware

Position Scaling Numerator From Hardware is the feed rate that the application applies when the output shaft has completed one revolution.

Position Scaling Denominator from Hardware

Position Scaling Denominator From Hardware is the number of increments that the drive outputs when the motor shaft has completed one revolution.

Example 1 on position scaling:

Motor without gear unit with 100 mm pinion on AX5000 with default settings:

- Numerator: 100 mm
- Denominator: 1048576

Example 2 on position scaling:

Motor with gear unit with i=10 on turntable on AX5000 with default settings:

- Numerator: 360° / 10 = 36°
- Denominator: 1048576

Encoder Mounting Offset

An offset used to correct the position of an encoder system or align it within the machine coordinate system, thereby defining the machine-specific zero point. This offset is restored when the controller is restarted.

Velocity Scaling Numerator to Hardware

This value is used to calculate the target and actual velocity values based on the encoder's pulse count.

Velocity Scaling Denominator to Hardware

This value is used to calculate the target and actual velocity values based on the encoder's pulse count.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Torque Scaling from Hardware

A factor used to scale the torque reading to the desired unit.

Connection

Name	Default value	Unit	Type
AMS address	0.0.0.0.0.0:0		
.netId	0.0.0.0.0.0		AMSNETID
.port	0x0000		WORD
Channel Number	0		UDINT
EtherCAT Slave Oid	00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

Dead times are automatically determined during EtherCAT initialization and taken into account in TwinCAT MC3. The calculated values generally do not require any manual adjustment. Nevertheless, a manual configuration can be necessary, for example to compensate the drive's internal dead times.

Name	Default value	Unit	Type
Dead Time Compensation from Hardware	NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
Additional Time Shift from Hardware	0.0	ms	LREAL
Dead Time Compensation to Hardware	NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation from Hardware

Set how the compensated current position should be calculated.

Additional Time Shift from Hardware

Additional offset of the automatically calculated dead time to compensate for the current position.

Dead Time Compensation to Hardware

Specify how the compensated target position is to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift to Hardware

Additional time shift of the automatically determined dead time for target position compensation.

Actual Position Filter

Name	Default value	Unit	Type
Filter Type	None		MC.EFilterType [▶ 199]
Filter Time Position	0.0	s	LREAL
Filter Time Velocity	0.0	s	LREAL
Filter Time Acceleration	0.01	s	LREAL
Filter Time	0.001	s	LREAL

Filter Type

`None`

No filter is applied.

`PT1-Filter`

This is a first-order delay element (digital low-pass filter, proportional element with first-order delay) used to smooth out noisy signals (actual value). The PT1 filter is applied **independently** to the three actual values: position, velocity, and acceleration. Each variable has its own filter time constant. The filter time constants

should be selected from the positive range of values. The value `0.0` disables filtering for the corresponding variable. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay. Because the variables are filtered separately, the output values are not necessarily consistent with each other.

`MovingAverage-Filter`

The moving-average filter calculates the arithmetic mean over a sliding time window of the last `N` input values and is thus a simple FIR low-pass filter (with a finite impulse response).
The moving-average filter is applied independently to the three actual values: position, velocity, and acceleration. Each variable has its own window length: `N`. The window length is calculated as the quotient of the parameterized filter time and the cycle time. The maximum window length is 1000 data points. The dead time is equal to half the window length.
Since the variables are filtered separately, the output values are not necessarily consistent with one another. With a filter time of `0.0`, the actual value is processed immediately and is not filtered. A long filter time results in significant smoothing and increases the dead time.

`KalmanFilter3x1`

The 3x1 Kalman filter estimates a set of position (s), velocity (v), and acceleration (a) values based on the continuous actual position values. On the one hand, this serves to reduce noise; on the other hand, the calculation of (s, v, a) is performed consistently, i.e., taking into account the interdependence of these variables. 3x1 indicates that the filter has three output variables (s, v, a) and one input variable (s).
It is often advantageous for (s, v, a) to be consistent with one another. In particular, the Kalman filter is recommended when coupling an axis to an encoder signal.
The 3x1 Kalman filter is configured using the “Filter Time” parameter, which defines the time scale used to calculate (s, v, a). The choice of this parameter represents a trade-off between noise reduction and dynamic response. A longer filter time results in greater noise reduction but a decrease in responsiveness, and vice versa (see the *Filter Time* parameter below).

Filter Time Position

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables position filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Velocity

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables velocity filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Acceleration

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables acceleration filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time

The parameter is active when the `Filter Type` is set to `KalmanFilter3x1`. The Kalman filter is adapted to the application using a simplified parameterization with a single filter time constant. The larger the filter time constant, the greater the degree of smoothing.

3.2.3.8.2.2 Online Parameter

Scaling

Name	Unit	Type
Work Zero Offset	[PosUnit]	LREAL
Active Homing Offset	[PosUnit]	LREAL

Work Zero Offset

Non-persistent offset that can be set using the MC_SetPosition function block. If MC_SetPosition is executed multiple times, the individual offsets are combined, taking the sign into account.
The offset is active at runtime and is reset when TwinCAT is restarted.

```
Work Zero Offset = Vorgegebene neue Position - Aktuelle Sollposition
```

Active Homing Offset

Displays the temporary offset calculated during the last homing procedure. This value is not permanently saved.

The following online parameters are available for TouchProbe2 as well as for TouchProbe1. For the sake of clarity and to avoid duplication, these parameters are listed here only for TouchProbe1.

TouchProbe1 Positive Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

TouchProbe1 Negative Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.8.2.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

MDP742 Drive / MDP742 Inputs

Enable (default)	Name	Type	Description
☑	Statusword	UINT	Status information
☑	Following error actual value	DINT	Position lag (position control deviation, i.e. target position minus actual position) of the I/O drive in increments.
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram.
☑	Position	UDINT	Actual position of the I/O drive in increments.
☐	Velocity actual value	DINT	Actual velocity of the I/O drive in increments.
☑	Torque actual value	INT	Actual torque/force value of the I/O drive in increments.
☑	TxPdoState	BIT	This element indicates whether the values measured by the encoder are in an invalid state.
☑	DcInputShift	DINT	Used for drive dead time compensation.
☑	DcOutputShift	DINT	Used for drive dead time compensation.

MDP742 Drive / MDP742 TouchProbe Inputs

In the default setting, this data range is disabled.

Enable (default)	Name	Type	Description
☑	TP1 Enable	BIT	Touch probe 1 is on
☑	TP1 Pos value stored	BIT	Value for touch probe 1 saved on the rising edge
☑	TP1 Neg value stored	BIT	Value for touch probe 1 saved on the falling edge
☑	TP1 Input	BIT	Digital input touch probe 1
☑	TP2 Enable	BIT	Touch probe 2 is on
☑	TP2 Pos value stored	BIT	Value for touch probe 2 saved on the rising edge
☑	TP2 Neg value stored	BIT	Value for touch probe 2 saved on the falling edge
☑	TP2 Input	BIT	Digital input touch probe 2
☑	TP1 Pos position	UDINT	Positive value of touch probe 1 **Unit**: the given value must be multiplied by the corresponding scaling factor [▶ 45].
☑	TP1 Neg position	UDINT	Negative value of touch probe 1 **Unit**: the given value must be multiplied by the corresponding scaling factor [▶ 45].
☑	TP1 Pos timestamp	UDINT	Timestamp for touch probe 1 when a rising edge event was detected.
☑	TP1 Neg timestamp	UDINT	Timestamp for touch probe 1 when a falling-edge event was detected.

Enable (default)	Name	Type	Description
☑	TP2 Pos position	UDINT	Positive value of touch probe 2 **Unit**: the given value must be multiplied by the corresponding scaling factor [▶ 45].
☑	TP2 Neg position	UDINT	Negative value of touch probe 2 **Unit**: the given value must be multiplied by the corresponding scaling factor [▶ 45].
☑	TP2 Pos timestamp	UDINT	Timestamp for touch probe 2 when a rising edge event was detected.
☑	TP2 Neg timestamp	UDINT	Timestamp for touch probe 2 when a falling-edge event was detected.

MDP742 Drive / MDP742 Outputs

Enable (default)	Name	Type	Description
☑	Controlword	UINT	Control information
☑	Target position	UDINT	Current target position in increments The target position of the MC3 axis in physical units, e.g. mm or degrees, is mathematically converted to an incremental value by MC3 and transferred to the drive. Overflows of the incremental value are hereby taken into account by the MC3 in the target position.
☐	Target velocity	DINT	Current target velocity in increments The target velocity of the MC3 axis in physical units, e.g. mm/s or degrees/s, is mathematically converted to an incremental value by MC3 and transferred to the drive.

MDP742 drive / MDP742 touch probe outputs

In the default setting, this data range is disabled.

Enable (default)	Name	Type	Description
☑	TP1 Enable	BIT	Turn on touch probe 1
☑	TP1 Continuous	BIT	One-time or cyclic events 0: first event detected after power-on 1: continuous detection
☑	TP1 Trigger mode	BIT2	Input 1 is triggered
☑	TP1 Enable pos edge	BIT	A rising edge at touch probe 1 is the trigger for an event
☑	TP1 Enable neg edge	BIT	A falling edge at touch probe 1 is the trigger for an event
☑	TP2 Enable	BIT	Turn on touch probe 2
☑	TP2 Continuous	BIT	One-time or cyclic events 0: first event detected after power-on 1: continuous detection
☑	TP2 Trigger mode	BIT2	Input 2 is triggered

Enable (default)	Name	Type	Description
☑	TP2 Enable pos edge	BIT	A rising edge at touch probe 2 is the trigger for an event
☑	TP2 Enable neg edge	BIT	A falling edge at touch probe 2 is the trigger for an event

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.8.3 CoE DS402 Drive

Module for linking an MC3 axis to a drive with the CoE DS402 drive profile.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.3.1 Init Parameter

General

Name	Default value	Unit	Type
Position Interpretation	Incremental		MC.EEncoderInterpretation [▶ 198]
Maximum Absolute Counter Value (Visibility depends on position interpretation)	1048575	INC	UDINT
Position Data Range	FullRange		MC.EEncoderDataRange [▶ 198]
Minimum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 0 • signed: -2147483648	INC	LINT
Maximum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 4294967295 • signed: 2147483647	INC	LINT
Feedback Velocity Source	HardwareProvidedIfAvailable		MC.EEncoderFeedbackVeloSource [▶ 198]

Position Interpretation

The `Position Interpretation` parameter defines the semantics according to which the values are transmitted by the position encoder and interpreted by the MC3 axis.

`Incremental`

The `Incremental` setting does not use the absolute properties of the physical feedback system, but can be used with both incremental and absolute encoder systems. After starting the controller, homing is required to initialize the actual position of the axis (`IsPositionValid = FALSE`).

`AbsoluteFullRange`

The `AbsoluteFullRange` setting is typically used with multi-turn or single-turn absolute encoders. The MC3 axis takes into account the entire data range of the data type in the I/O mapping (FullRange). The maximum number of increments corresponds to the upper limit of the data type. The size of the data type is specified in the protocol used.

When the controller is powered on or restarted and position limits are enabled, the axis signals `IsPositionValid = TRUE`. Configuring and enabling the software-based position limits for the axis allows the axis position to be restored after a controller restart. This applies to a maximum of one overflow of the transmitted increments. The position limits must be selected such that the resulting possible travel range of the axis is less than the encoder's counting range (FullRange).

If the position limits are disabled, the axis signals this with `IsPositionValid = FALSE`. A homing run is not required for AbsoluteFullRange.

`AbsolutUserDataRange`

The same boundary conditions apply as for the setting. `AbsoluteFullRange`. In addition, the parameter `Maximum Absolute Counter Value` is displayed and taken into account:

- `Maximum Absolute Counter Value`
  Unique maximum number of increments that the encoder system transmits (within the limits of the data type used). For example, the default value is set to 2^20-1.

`AbsoluteModulo`

The position increments are transmitted by the I/O device using modulo logic. For example, if one revolution of the encoder system corresponds to the modulus of the MC3 axis, the position is unique within this range. The axis indicates `IsPositionValid = TRUE`.

Position Data Range

The `Position Data Range` parameter defines the value range of the encoder system that is used for position calculation. This refers to the raw value received from the drive or encoder via the process data (PDO mapping) and enables the detection and handling of overflows. The parameter can only be changed at configuration time, not at runtime.

`FullRange`

Position overflow is based on the data type limits of the respective encoder system. For a signed 32-bit encoder system (e.g., CoE DS402 Drive (AX8000), SoE Drive (AX5000), EtherCAT encoder), this implicitly means a value range from -2147483648 to 2147483647. For an unsigned 32-bit encoder system (e.g., MDP742 Drive (selected ELM72xx), SSI encoder), this implies a value range from 0 to 4294967295.

`UserDefined`

This parameter value allows you to manually enter the range limits used to determine an overflow. Two additional parameters for entering the upper and lower limits will be displayed:

- Minimum Counter Value
- Maximum Counter Value

Feedback Velocity Source

Specifies the source of the actual velocity (AXIS_REF.McToPlc.Act.Velocity). The velocity can either be obtained from the hardware being used or calculated internally by differentiating the actual position value.

`HardwareProvidedIfAvailable` (Default value)

Uses the velocity provided by the hardware. If no entry for the actual velocity is mapped in the process data, it is automatically calculated internally.

`InternallyCalculated`

Velocity is always calculated from the derivative of the actual positions. Actual hardware velocities are ignored, even if they are available.

`HardwareProvided`

The velocity is always taken directly from the hardware feedback. This requires linking the actual velocity via process data (PDO mapping). If the mapping is missing, this will result in an error when the configuration is started.

Scaling

Name	Default value	Unit	Type
Position Scaling Numerator from Hardware	360	[PosUnit]/INC	LREAL
Position Scaling Denominator from Hardware	1048576.0		LREAL
Encoder Mounting Offset	0.0	[PosUnit]	LREAL
Velocity Scaling Numerator to Hardware	1.0	INC/([PosUnit]/s)	LREAL
Velocity Scaling Denominator to Hardware	1.0		LREAL
Direction Inverted	`FALSE`		BOOL
Torque Scaling from Hardware	0.1	INC/([TorqueUnit])	LREAL

The position scaling factor is divided into a numerator and a denominator. It is used to convert the path increments into axis positions or to calculate a user unit from encoder increments.

Position Scaling Numerator from Hardware

Position Scaling Numerator From Hardware is the feed rate that the application applies when the output shaft has completed one revolution.

Position Scaling Denominator from Hardware

Position Scaling Denominator From Hardware is the number of increments that the drive outputs when the motor shaft has completed one revolution.

Example 1 on position scaling:

Motor without gear unit with 100 mm pinion on AX5000 with default settings:

- Numerator: 100 mm
- Denominator: 1048576

Example 2 on position scaling:

Motor with gear unit with i=10 on turntable on AX5000 with default settings:

- Numerator: 360° / 10 = 36°
- Denominator: 1048576

Encoder Mounting Offset

An offset used to correct the position of an encoder system or align it within the machine coordinate system, thereby defining the machine-specific zero point. This offset is restored when the controller is restarted.

Velocity Scaling Numerator to Hardware

This value is used to calculate the target and actual velocity values based on the encoder's pulse count.

Velocity Scaling Denominator to Hardware

This value is used to calculate the target and actual velocity values based on the encoder's pulse count.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Torque Scaling from Hardware

A factor used to scale the torque reading to the desired unit.

Connection

Name		Default value	Unit	Type
AMS address		0.0.0.0.0.0:0		
	.netId	0.0.0.0.0.0		AMSNETID
	.port	0x0000		WORD
Channel Number		0		UDINT
EtherCAT Slave Oid		00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

Dead times are automatically determined during EtherCAT initialization and taken into account in TwinCAT MC3. The calculated values generally do not require any manual adjustment. Nevertheless, a manual configuration can be necessary, for example to compensate the drive's internal dead times.

Name		Default value	Unit	Type
Dead Time Compensation from Hardware		NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift from Hardware	0.0	ms	LREAL

Name		Default value	Unit	Type
Dead Time Compensation to Hardware		NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation from Hardware

Set how the compensated current position should be calculated.

Additional Time Shift from Hardware

Additional offset of the automatically calculated dead time to compensate for the current position.

Dead Time Compensation to Hardware

Specify how the compensated target position is to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift to Hardware

Additional time shift of the automatically determined dead time for target position compensation.

Touch probe

Name	Default value	Unit	Type
TouchProbe1 Source	DriveDefault		MC.ETouchProbeTriggerSource [▶ 203]
TouchProbe2 Source	DriveDefault		MC.ETouchProbeTriggerSource [▶ 203]

TouchProbe Source

Selection of the input for the touch probe.

Actual Position Filter

Name		Default value	Unit	Type
Filter Type		None		MC.EFilterType [▶ 199]
	Filter Time Position	0.0	s	LREAL
	Filter Time Velocity	0.0	s	LREAL
	Filter Time Acceleration	0.01	s	LREAL
	Filter Time	0.001	s	LREAL

Filter Type

`None`

No filter is applied.

`PT1-Filter`

This is a first-order delay element (digital low-pass filter, proportional element with first-order delay) used to smooth out noisy signals (actual value). The PT1 filter is applied **independently** to the three actual values: position, velocity, and acceleration. Each variable has its own filter time constant. The filter time constants should be selected from the positive range of values. The value `0.0` disables filtering for the corresponding variable. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay. Because the variables are filtered separately, the output values are not necessarily consistent with each other.

`MovingAverage-Filter`

The moving-average filter calculates the arithmetic mean over a sliding time window of the last `N` input values and is thus a simple FIR low-pass filter (with a finite impulse response).
The moving-average filter is applied independently to the three actual values: position, velocity, and acceleration. Each variable has its own window length: `N`. The window length is calculated as the quotient of the parameterized filter time and the cycle time. The maximum window length is 1000 data points. The dead time is equal to half the window length.
Since the variables are filtered separately, the output values are not necessarily consistent with one another. With a filter time of `0.0`, the actual value is processed immediately and is not filtered. A long filter time results in significant smoothing and increases the dead time.

`KalmanFilter3x1`

The 3x1 Kalman filter estimates a set of position (s), velocity (v), and acceleration (a) values based on the continuous actual position values. On the one hand, this serves to reduce noise; on the other hand, the calculation of (s, v, a) is performed consistently, i.e., taking into account the interdependence of these variables. 3x1 indicates that the filter has three output variables (s, v, a) and one input variable (s).
It is often advantageous for (s, v, a) to be consistent with one another. In particular, the Kalman filter is recommended when coupling an axis to an encoder signal.
The 3x1 Kalman filter is configured using the “Filter Time” parameter, which defines the time scale used to calculate (s, v, a). The choice of this parameter represents a trade-off between noise reduction and dynamic response. A longer filter time results in greater noise reduction but a decrease in responsiveness, and vice versa (see the *Filter Time* parameter below).

Filter Time Position

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables position filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Velocity

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables velocity filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Acceleration

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables acceleration filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time

The parameter is active when the `Filter Type` is set to `KalmanFilter3x1`. The Kalman filter is adapted to the application using a simplified parameterization with a single filter time constant. The larger the filter time constant, the greater the degree of smoothing.

3.2.3.8.3.2 Online Parameter

Scaling

Name	Unit	Type
Work Zero Offset	[PosUnit]	LREAL
Active Homing Offset	[PosUnit]	LREAL

Work Zero Offset

Non-persistent offset that can be set using the MC_SetPosition function block. If MC_SetPosition is executed multiple times, the individual offsets are combined, taking the sign into account.
The offset is active at runtime and is reset when TwinCAT is restarted.

```
Work Zero Offset = Vorgegebene neue Position - Aktuelle Sollposition
```

Active Homing Offset

Displays the temporary offset calculated during the last homing procedure. This value is not permanently saved.

The following online parameters are available for TouchProbe2 as well as for TouchProbe1. For the sake of clarity and to avoid duplication, these parameters are listed here only for TouchProbe1.

TouchProbe1 Positive Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

TouchProbe1 Negative Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.8.3.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

CoE DS402 Drive / CoE Inputs

Enable (default)	Name	Type	Description
☑	Statusword	WORD	Status information
☑	Following error actual value	DINT	Position lag (position control deviation, i.e. target position minus actual position) of the I/O drive in increments.

Enable (default)	Name	Type	Description
☑	Position actual value invalid	BIT	This element indicates whether the value measured by the position sensor is invalid.
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram.
☑	Position actual value	DINT	Actual position of the I/O drive in increments.
☐	Velocity actual value	DINT	Actual velocity of the I/O drive in increments.
☑	Torque/force actual value	INT	Actual torque/force value of the I/O drive in increments.
☑	DcInputShift	DINT	Used for drive dead time compensation.
☑	DcOutputShift	DINT	Used for drive dead time compensation.

CoE DS402 Drive / DS402 Touchprobe Inputs

In the default setting, this data range is disabled.

Enable (default)	Name	Type	Description
☑	Touch probe status	UINT	This element displays the status of the touch probe function.
☑	Touch probe 1 positive edge	DINT	This element returns the position value of touch probe 1 on the rising edge.
☑	Touch probe 1 negative edge	DINT	This element returns the position value of touch probe 1 on a falling edge.
☑	Touch probe time stamp 1 positive edge	UDINT	This element provides the timestamp when a rising edge event was detected at touch probe 1.
☑	Touch probe time stamp 1 negative edge	UDINT	This element provides the timestamp for when a falling-edge event was detected at touch probe 1.
☑	Touch probe 2 positive edge	DINT	This element returns the position value of touch probe 2 on the rising edge.
☑	Touch probe 2 negative edge	DINT	This element returns the position value of touch probe 2 on a falling edge.
☑	Touch probe time stamp 2 positive edge	UDINT	This element provides the timestamp when a rising edge event was detected at touch probe 2.
☑	Touch probe time stamp 2 negative edge	UDINT	This element provides the timestamp for when a falling-edge event was detected at touch probe 2.

CoE DS402 Drive / CoE Outputs

Enable (default)	Name	Type	Description
☑	Controlword	WORD	Control information
☑	Target position	DINT	Current target position in increments

Enable (default)	Name	Type	Description
			The target position of the MC3 axis in physical units, e.g. mm or degrees, is mathematically converted to an incremental value by MC3 and transferred to the drive. Overflows of the incremental value are hereby taken into account by the MC3 in the target position.
☐	Target velocity	DINT	Current target velocity in increments The target velocity of the MC3 axis in physical units, e.g. mm/s or degrees/s, is mathematically converted to an incremental value by MC3 and transferred to the drive.

CoE DS402 Drive / DS402 TouchProbe Outputs

In the default setting, this data range is disabled.

Enable (default)	Name	Type	Description
☑	Touch probe control	UINT	This element contains the configuration for the touch probe function.

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.8.4 SoE Drive

Module for linking an MC3 axis to a drive with the SoE drive profile.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.4.1 Init Parameter

General

Name	Default value	Unit	Type
Position Interpretation	Incremental		MC.EEncoderInterpretation [▶ 198]
Maximum Absolute Counter Value	1048575	INC	UDINT

Name		Default value	Unit	Type
	(Visibility depends on position interpretation)			
Position Data Range		FullRange		MC.EEncoderDataRange [▶ 198]
	Minimum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 0 • signed: -2147483648	INC	LINT
	Maximum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 4294967295 • signed: 2147483647	INC	LINT
Feedback Velocity Source		HardwareProvidedIfAvailable		MC.EEncoderFeedbackVeloSource [▶ 198]

Position Interpretation

The `Position Interpretation` parameter defines the semantics according to which the values are transmitted by the position encoder and interpreted by the MC3 axis.

`Incremental`

The `Incremental` setting does not use the absolute properties of the physical feedback system, but can be used with both incremental and absolute encoder systems. After starting the controller, homing is required to initialize the actual position of the axis (`IsPositionValid = FALSE`).

`AbsoluteFullRange`

The `AbsoluteFullRange` setting is typically used with multi-turn or single-turn absolute encoders. The MC3 axis takes into account the entire data range of the data type in the I/O mapping (FullRange). The maximum number of increments corresponds to the upper limit of the data type. The size of the data type is specified in the protocol used.

When the controller is powered on or restarted and position limits are enabled, the axis signals `IsPositionValid = TRUE`. Configuring and enabling the software-based position limits for the axis allows the axis position to be restored after a controller restart. This applies to a maximum of one overflow of the transmitted increments. The position limits must be selected such that the resulting possible travel range of the axis is less than the encoder's counting range (FullRange).

If the position limits are disabled, the axis signals this with `IsPositionValid = FALSE`. A homing run is not required for AbsoluteFullRange.

`AbsolutUserDataRange`

The same boundary conditions apply as for the setting. `AbsoluteFullRange`. In addition, the parameter `Maximum Absolute Counter Value` is displayed and taken into account:

- `Maximum Absolute Counter Value`
  Unique maximum number of increments that the encoder system transmits (within the limits of the data type used). For example, the default value is set to 2^20-1.

`AbsoluteModulo`

The position increments are transmitted by the I/O device using modulo logic. For example, if one revolution of the encoder system corresponds to the modulus of the MC3 axis, the position is unique within this range. The axis indicates `IsPositionValid = TRUE`.

Position Data Range

The `Position Data Range` parameter defines the value range of the encoder system that is used for position calculation. This refers to the raw value received from the drive or encoder via the process data (PDO mapping) and enables the detection and handling of overflows. The parameter can only be changed at configuration time, not at runtime.

`FullRange`

Position overflow is based on the data type limits of the respective encoder system. For a signed 32-bit encoder system (e.g., CoE DS402 Drive (AX8000), SoE Drive (AX5000), EtherCAT encoder), this implicitly means a value range from -2147483648 to 2147483647. For an unsigned 32-bit encoder system (e.g., MDP742 Drive (selected ELM72xx), SSI encoder), this implies a value range from 0 to 4294967295.

`UserDefined`

This parameter value allows you to manually enter the range limits used to determine an overflow. Two additional parameters for entering the upper and lower limits will be displayed:

- Minimum Counter Value
- Maximum Counter Value

Feedback Velocity Source

Specifies the source of the actual velocity (AXIS_REF.McToPlc.Act.Velocity). The velocity can either be obtained from the hardware being used or calculated internally by differentiating the actual position value.

`HardwareProvidedIfAvailable` (Default value)

Uses the velocity provided by the hardware. If no entry for the actual velocity is mapped in the process data, it is automatically calculated internally.

`InternallyCalculated`

Velocity is always calculated from the derivative of the actual positions. Actual hardware velocities are ignored, even if they are available.

`HardwareProvided`

The velocity is always taken directly from the hardware feedback. This requires linking the actual velocity via process data (PDO mapping). If the mapping is missing, this will result in an error when the configuration is started.

Scaling

Name	Default value	Unit	Type
Position Scaling Numerator from Hardware	360	[PosUnit]/INC	LREAL
Position Scaling Denominator from Hardware	1048576.0		LREAL
Encoder Mounting Offset	0.0	[PosUnit]	LREAL
Velocity Scaling Numerator to Hardware	1.0	INC/([PosUnit]/s)	LREAL
Velocity Scaling Denominator to Hardware	1.0		LREAL
Direction Inverted	`FALSE`		BOOL
Torque Scaling from Hardware	0.1	INC/([TorqueUnit])	LREAL

The position scaling factor is divided into a numerator and a denominator. It is used to convert the path increments into axis positions or to calculate a user unit from encoder increments.

Position Scaling Numerator from Hardware

Position Scaling Numerator From Hardware is the feed rate that the application applies when the output shaft has completed one revolution.

Position Scaling Denominator from Hardware

Position Scaling Denominator From Hardware is the number of increments that the drive outputs when the motor shaft has completed one revolution.

Example 1 on position scaling:

Motor without gear unit with 100 mm pinion on AX5000 with default settings:

- Numerator: 100 mm
- Denominator: 1048576

Example 2 on position scaling:

Motor with gear unit with i=10 on turntable on AX5000 with default settings:

- Numerator: 360° / 10 = 36°
- Denominator: 1048576

Encoder Mounting Offset

An offset used to correct the position of an encoder system or align it within the machine coordinate system, thereby defining the machine-specific zero point. This offset is restored when the controller is restarted.

Velocity Scaling Numerator to Hardware

This value is used to calculate the target and actual velocity values based on the encoder's pulse count.

Velocity Scaling Denominator to Hardware

This value is used to calculate the target and actual velocity values based on the encoder's pulse count.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Torque Scaling from Hardware

A factor used to scale the torque reading to the desired unit.

Connection

Name	Default value	Unit	Type
AMS address	0.0.0.0.0.0:0		
.netId	0.0.0.0.0.0		AMSNETID
.port	0x0000		WORD
Channel Number	0		UDINT
EtherCAT Slave Oid	00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name		Default value	Unit	Type
Dead Time Compensation from Hardware		NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift from Hardware	0.0	ms	LREAL
Dead Time Compensation		NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation from Hardware

Specify how the compensated actual position should be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift from Hardware

Time required to adjust the (additive), automatically determined dead time for compensating the actual position.

Dead Time Compensation

Specify how the compensated target position is to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift to Hardware

Time required to adjust the (additive), automatically determined dead time for compensating the target position.

Actual Position Filter

Name		Default value	Unit	Type
Filter Type		None		MC.EFilterType [▶ 199]
	Filter Time Position	0.0	s	LREAL
	Filter Time Velocity	0.0	s	LREAL
	Filter Time Acceleration	0.01	s	LREAL
	Filter Time	0.001	s	LREAL

Filter Type

`None`

No filter is applied.

`PT1-Filter`

This is a first-order delay element (digital low-pass filter, proportional element with first-order delay) used to smooth out noisy signals (actual value). The PT1 filter is applied **independently** to the three actual values: position, velocity, and acceleration. Each variable has its own filter time constant. The filter time constants should be selected from the positive range of values. The value `0.0` disables filtering for the corresponding variable. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay. Because the variables are filtered separately, the output values are not necessarily consistent with each other.

`MovingAverage-Filter`

The moving-average filter calculates the arithmetic mean over a sliding time window of the last `N` input values and is thus a simple FIR low-pass filter (with a finite impulse response).
The moving-average filter is applied independently to the three actual values: position, velocity, and acceleration. Each variable has its own window length: `N`. The window length is calculated as the quotient of the parameterized filter time and the cycle time. The maximum window length is 1000 data points. The dead time is equal to half the window length.
Since the variables are filtered separately, the output values are not necessarily consistent with one another. With a filter time of `0.0`, the actual value is processed immediately and is not filtered. A long filter time results in significant smoothing and increases the dead time.

`KalmanFilter3x1`

The 3x1 Kalman filter estimates a set of position (s), velocity (v), and acceleration (a) values based on the continuous actual position values. On the one hand, this serves to reduce noise; on the other hand, the calculation of (s, v, a) is performed consistently, i.e., taking into account the interdependence of these variables. 3x1 indicates that the filter has three output variables (s, v, a) and one input variable (s).
It is often advantageous for (s, v, a) to be consistent with one another. In particular, the Kalman filter is recommended when coupling an axis to an encoder signal.
The 3x1 Kalman filter is configured using the “Filter Time” parameter, which defines the time scale used to calculate (s, v, a). The choice of this parameter represents a trade-off between noise reduction and dynamic response. A longer filter time results in greater noise reduction but a decrease in responsiveness, and vice versa (see the *Filter Time* parameter below).

Filter Time Position

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables position filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Velocity

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables velocity filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Acceleration

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables acceleration filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time

The parameter is active when the `Filter Type` is set to `KalmanFilter3x1`. The Kalman filter is adapted to the application using a simplified parameterization with a single filter time constant. The larger the filter time constant, the greater the degree of smoothing.

3.2.3.8.4.2 Online Parameter

Scaling

Name	Unit	Type
Work Zero Offset	[PosUnit]	LREAL
Active Homing Offset	[PosUnit]	LREAL

Work Zero Offset

Non-persistent offset that can be set using the MC_SetPosition function block. If MC_SetPosition is executed multiple times, the individual offsets are combined, taking the sign into account.
The offset is active at runtime and is reset when TwinCAT is restarted.

```
Work Zero Offset = Vorgegebene neue Position - Aktuelle Sollposition
```

Active Homing Offset

Displays the temporary offset calculated during the last homing procedure. This value is not permanently saved.

TouchProbe1 Positive Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

TouchProbe1 Negative Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.8.4.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

SoE Drive / SoE Inputs

Enable (default)	Name	Type	Description
☑	Drive status word	UINT	Status information
☑	Position feedback value 1	DINT	Actual position of the I/O drive in increments.
☑	Following distance	DINT	Position lag (position control deviation, i.e. target position minus actual position) of the I/O drive in increments.
☐	Velocity feedback value 1	DINT	Actual velocity of the I/O drive in increments.
☑	Torque feedback value	INT	Actual torque/force value of the I/O drive in increments.
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram.
☑	DcInputShift	DINT	Used for drive dead time compensation.
☑	DcOutputShift	DINT	Used for drive dead time compensation.

SoE Drive / SoE Touch Probe Inputs

In the default setting, this data range is disabled.

Enable (default)	Name	Type	Description
☑	Probe value 1 positive edge	DINT	This element returns the position value of touch probe 1 on the rising edge.
☑	Probe value 1 negative edge	DINT	This element returns the position value of touch probe 1 on a falling edge.
☑	Probe 1 positive latched	UINT	Event detected on a rising edge for touch probe 1.
☑	Probe 1 negative latched	UINT	Event detected on the falling edge for touch probe 1.
☑	Probe 1 time positive edge	UDINT	This element provides the timestamp when a rising edge event was detected at touch probe 1.
☑	Probe 1 time negative edge	UDINT	This element provides the timestamp for when a falling-edge event was detected at touch probe 1.

SoE Drive / SoE Outputs

Enable (default)	Name	Type	Description
☑	Master control word	WORD	Control information
☑	Position command value	DINT	Current target position in increments The target position of the MC3 axis in physical units, e.g. mm or degrees, is mathematically converted to an incremental value by MC3 and transferred to the drive. Overflows of the incremental value are hereby taken into account by the MC3 in the target position.

Enable (default)	Name	Type	Description
☐	Velocity command value	DINT	Current target velocity in increments The target velocity of the MC3 axis in physical units, e.g. mm/s or degrees/s, is mathematically converted to an incremental value by MC3 and transferred to the drive.

SoE Drive / SoE Touch Probe Outputs

In the default setting, this data range is disabled.

Enable (default)	Name	Type	Description
☑	Probe 1 enable	UINT	Turn on touch probe 1.

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.8.5 CoE DS408 Drive (hydraulic)

Module for linking an MC3 axis to a drive with the CoE DS408 (hydraulic) drive profile.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.5.1 Init Parameter

Scaling

Name	Default value	Unit	Type
Velocity Scaling Numerator to Hardware	1.0	INC/([PosUnit]/s)	LREAL
Velocity Scaling Denominator to Hardware	1.0		LREAL
Direction Inverted	`FALSE`		BOOL
Output Zero Offset	0.0	%	LREAL

Velocity Scaling Numerator to Hardware

`Velocity Scaling Numerator To Hardware` is used to calculate the target and actual velocity values from the encoder revolutions.

Velocity Scaling Denominator to Hardware

`Velocity Scaling Denominator To Hardware` is used to calculate the target and actual velocity values from the encoder count.

Invert Direction

`Invert Direction` Specifies whether the velocity values to and from the hardware should be inverted.

Output Zero Offset

`Output Zero Offset` indicates the output at zero velocity as a percentage of the maximum output range (+-32767 for analog drives).

Connection

Name	Default value	Unit	Type
AMS address	0.0.0.0.0.0:0		
.netId	0.0.0.0.0.0		AMSNETID
.port	0x0000		WORD
Channel Number	0		UDINT
EtherCAT Slave Oid	00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name	Default value	Unit	Type
Dead Time Compensation	NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation

Specify how the compensated target position and actual position are to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift to Hardware

Additional time shift of the automatically determined dead time for target position compensation.

3.2.3.8.5.2 Online Parameter

Name	Unit	Type
Spool Set Position		LREAL
Spool Act Position		LREAL

Spool Set Position

Setpoint for the coil position ((-)1.0 = full power)

Spool Act Position

Current value of the coil position ((-)1.0 = full power)

3.2.3.8.5.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

CoE DS408 Drive (hydraulic) / DS408 Inputs

Enable (default)	Name	Type	Description
☑	Status word	UINT	Status information
☑	Valve Act Spool Position	INT	Current position of the valve slide
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram

CoE DS408 Drive (hydraulic) / DS408 Outputs

Enable (default)	Name	Type	Description
☑	Control word	UINT	Control information
☑	Valve Set Spool Position	INT	Target position of the valve spool

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.8.6 Analog Drive

Module for linking an MC3 axis to an analog drive.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.6.1 Init Parameter

Scaling

Name	Default value	Unit	Type
Velocity Scaling Numerator to Hardware	1.0	INC/([PosUnit]/s)	LREAL
Velocity Scaling Denominator to Hardware	1.0		LREAL
Direction Inverted	FALSE		BOOL
Output Zero Offset	0.0	%	LREAL

Velocity Scaling Numerator to Hardware

`Velocity Scaling Numerator To Hardware` is used to calculate the analog output value based on the velocity setpoint.

Velocity Scaling Denominator to Hardware

`Velocity Scaling Denominator To Hardware` is used to calculate the analog output value based on the velocity setpoint.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Output Zero Offset

`Output Zero Offset` indicates the output at zero velocity as a percentage of the maximum output range (+-32767 for analog drives).

Connection

Name	Default value	Unit	Type
AMS address	0.0.0.0.0.0:0		
.netId	0.0.0.0.0.0		AMSNETID
.port	0x0000		WORD
Channel Number	0		UDINT
EtherCAT Slave Oid	00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name	Default value	Unit	Type
Dead Time Compensation	NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation

Specify how the compensated target position and actual position are to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift to Hardware

Additional time shift of the automatically determined dead time for target position compensation.

3.2.3.8.6.2 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

Analog Drive / Analog Inputs

Enable (default)	Name	Type	Description
☑	WcState	BIT	Working counter

Analog Drive / Analog Outputs

Enable (default)	Name	Type	Description
☑	Output	INT	Initial value

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.8.7 PulseTrain Drive

Module for linking an MC3 axis to a drive using the PulseTrain drive profile.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.7.1 Init Parameter

Scaling

Name	Default value	Unit	Type
Position Scaling Numerator from Hardware	1.0	[PosUnit]/INC	LREAL
Position Scaling Denominator from Hardware	1.0		LREAL
Direction Inverted	FALSE		BOOL
Encoder Mounting Offset	0.0	[PosUnit]	LREAL

The position scaling factor is divided into a numerator and a denominator. It is used to convert the path increments into axis positions or to calculate a user unit from encoder increments.

Position Scaling Numerator from Hardware

Position Scaling Numerator From Hardware is the feed rate that the application applies when the output shaft has completed one revolution.

Position Scaling Denominator from Hardware

Position Scaling Denominator From Hardware is the number of increments that the drive outputs when the motor shaft has completed one revolution.

Example 1 on position scaling:

Motor without gear unit with 100 mm pinion on AX5000 with default settings:

- Numerator: 100 mm
- Denominator: 1048576

Example 2 on position scaling:

Motor with gear unit with i=10 on turntable on AX5000 with default settings:

- Numerator: 360° / 10 = 36°
- Denominator: 1048576

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Encoder Mounting Offset

An offset used to correct the position of an encoder system or align it within the machine coordinate system, thereby defining the machine-specific zero point. This offset is restored when the controller is restarted.

Connection

Name		Default value	Unit	Type
AMS address		0.0.0.0.0.0:0		
	.netId	0.0.0.0.0.0		AMSNETID
	.port	0x0000		WORD
Channel Number		0		UDINT
EtherCAT Slave Oid		00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name		Default value	Unit	Type
Dead Time Compensation		None		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation

Specify how the compensated setpoint and actual value should be calculated. The dead time is automatically determined during EtherCAT initialization.

Dead Time Compensation to Hardware

Time required to adjust the automatically calculated dead time for setpoint compensation.

Actual Position Filter

Name		Default value	Unit	Type
Filter Type		None		MC.EFilterType [▶ 199]
	Filter Time Position	0.0	s	LREAL

Name		Default value	Unit	Type
	Filter Time Velocity	0.0	s	LREAL
	Filter Time Acceleration	0.01	s	LREAL
	Filter Time	0.001	s	LREAL

Filter Type

`None`

No filter is applied.

`PT1-Filter`

This is a first-order delay element (digital low-pass filter, proportional element with first-order delay) used to smooth out noisy signals (actual value). The PT1 filter is applied **independently** to the three actual values: position, velocity, and acceleration. Each variable has its own filter time constant. The filter time constants should be selected from the positive range of values. The value `0.0` disables filtering for the corresponding variable. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay. Because the variables are filtered separately, the output values are not necessarily consistent with each other.

`MovingAverage-Filter`

The moving-average filter calculates the arithmetic mean over a sliding time window of the last `N` input values and is thus a simple FIR low-pass filter (with a finite impulse response).
The moving-average filter is applied independently to the three actual values: position, velocity, and acceleration. Each variable has its own window length: `N`. The window length is calculated as the quotient of the parameterized filter time and the cycle time. The maximum window length is 1000 data points. The dead time is equal to half the window length.
Since the variables are filtered separately, the output values are not necessarily consistent with one another. With a filter time of `0.0`, the actual value is processed immediately and is not filtered. A long filter time results in significant smoothing and increases the dead time.

`KalmanFilter3x1`

The 3x1 Kalman filter estimates a set of position (s), velocity (v), and acceleration (a) values based on the continuous actual position values. On the one hand, this serves to reduce noise; on the other hand, the calculation of (s, v, a) is performed consistently, i.e., taking into account the interdependence of these variables. 3x1 indicates that the filter has three output variables (s, v, a) and one input variable (s).
It is often advantageous for (s, v, a) to be consistent with one another. In particular, the Kalman filter is recommended when coupling an axis to an encoder signal.
The 3x1 Kalman filter is configured using the “Filter Time” parameter, which defines the time scale used to calculate (s, v, a). The choice of this parameter represents a trade-off between noise reduction and dynamic response. A longer filter time results in greater noise reduction but a decrease in responsiveness, and vice versa (see the *Filter Time* parameter below).

Filter Time Position

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables position filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Velocity

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables velocity filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Acceleration

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables acceleration filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time

The parameter is active when the `Filter Type` is set to `KalmanFilter3x1`. The Kalman filter is adapted to the application using a simplified parameterization with a single filter time constant. The larger the filter time constant, the greater the degree of smoothing.

3.2.3.8.7.2 Online Parameter

Scaling

Name	Unit	Type
Work Zero Offset	[PosUnit]	LREAL
Active Homing Offset	[PosUnit]	LREAL

Work Zero Offset

Non-persistent offset that can be set using the MC_SetPosition function block. If MC_SetPosition is executed multiple times, the individual offsets are combined, taking the sign into account.
The offset is active at runtime and is reset when TwinCAT is restarted.

```
Work Zero Offset = Vorgegebene neue Position - Aktuelle Sollposition
```

Active Homing Offset

Displays the temporary offset calculated during the last homing procedure. This value is not permanently saved.

3.2.3.8.7.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

PulseTrain Drive / PulseTrain Inputs

Enable (default)	Name	Type	Description
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram
☑	Counter value	UDINT	Meter reading
☑	ENC Sync error	BIT	The Sync error bit is required only for distributed clocks mode and indicates whether a synchronization error occurred during the previous cycle.
☑	ENC TxPDO State	BIT	Validity of the data in the associated TxPDO (0=valid, 1=invalid).
☑	ENC TxPDO Toggle	BIT	The TxPDO toggle is set by the slave when the data in the corresponding TxPDO has been updated.
☑	DcInputShift	DINT	Used for dead-time compensation of the drive.
☑	DcOutputShift	DINT	Used for dead-time compensation of the drive.

PulseTrain Drive / PulseTrain Outputs

Enable (default)	Name	Type	Description
☑	PTO Automatic direction	BIT	The terminal selects the direction of travel based on the position difference (actual and target values) in such a way that the shorter distance is preferred.
☑	Target counter value	UDINT	Target meter reading

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.8.8 Pulse Width Current Drive (MDP250) (both channels)

Module for linking an MC3 axis to a drive with the Pulse Width Current Drive (MDP250) drive profile (both channels).

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.8.8.1 Init Parameter

Scaling

Name	Default value	Unit	Type
Velocity Scaling Numerator to Hardware	1.0	INC/([PosUnit]/s)	LREAL
Velocity Scaling Denominator to Hardware	1.0		LREAL
Direction Inverted	FALSE		BOOL
Output Zero Offset	0.0	%	LREAL

Velocity Scaling Numerator to Hardware

`Velocity Scaling Numerator To Hardware` is used to calculate the analog output value based on the velocity setpoint.

Velocity Scaling Denominator to Hardware

`Velocity Scaling Denominator To Hardware` is used to calculate the analog output value based on the velocity setpoint.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Output Zero Offset

`Output Zero Offset` indicates the output at zero velocity as a percentage of the maximum output range (+-32767 for analog drives).

Connection

Name		Default value	Unit	Type
AMS address		0.0.0.0.0.0:0		
	.netId	0.0.0.0.0.0		AMSNETID
	.port	0x0000		WORD
Channel Number		0		UDINT
EtherCAT Slave Oid		00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name		Default value	Unit	Type
Dead Time Compensation		None		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift to Hardware	0.0	ms	LREAL

Dead Time Compensation

Specify how the compensated setpoint and actual value should be calculated. The dead time is automatically determined during EtherCAT initialization.

Dead Time Compensation to Hardware

Time required to adjust the automatically calculated dead time for setpoint compensation.

3.2.3.8.8.2 Online Parameter

Functionality

Name	Unit	Type
Dither Enabled		BOOL

Enables the drive's dither signal to reduce hysteresis and static friction effects.

3.2.3.8.8.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

Pulse Width Current Drive (MDP250) (both channels) / PWM Inputs

Enable (default)	Name	Type	Description
☑	Channel 1 Ready to enable	BIT	Channel 1: The terminal indicates that it is ready for operation.
☑	Channel 1 Warning	BIT	Channel 1: A warning has occurred.
☑	Channel 1 Error	BIT	Channel 1: An error has occurred, and the output drivers are disabled.
☑	Channel 2 Ready to enable	BIT	Channel 2: The terminal indicates that it is ready for operation.
☑	Channel 2 Warning	BIT	Channel 2: A warning has occurred.
☑	Channel 2 Error	BIT	Channel 2: An error has occurred, and the output drivers are disabled.
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram.

Pulse Width Current Drive (MDP250) (both channels) / PWM Outputs

Enable (default)	Name	Type	Description
☑	Channel 1 Enable dithering	BIT	Dithering for Channel 1 is "enabled."
☑	Channel 1 Enable	BIT	Enable dithering for channel 1.
☑	Channel 1 Reset	BIT	Resetting an error on Channel 1.
☑	Channel 1 PWM Output	INT	Output from the PWM terminal on channel 1
☑	Channel 2 Enable dithering	BIT	Dithering for Channel 2 is "enabled”.
☑	Channel 2 Enable	BIT	Enable dithering for channel 2.
☑	Channel 2 Reset	BIT	Resetting an error on Channel 2.
☑	Channel 2 PWM Output	INT	Output from the PWM terminal on channel 2.

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.9 Encoder modules

The following section contains information about the available encoder modules.

3.2.3.9.1 EtherCAT Encoder

Module for linking an MC3 axis to an EtherCAT encoder.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.9.1.1 Init Parameter

General

Name	Default value	Unit	Type
Position Interpretation	Incremental		MC.EEncoderInterpretation [▶ 198]
Maximum Absolute Counter Value (Visibility depends on position interpretation)	1048575	INC	UDINT
Position Data Range	FullRange		MC.EEncoderDataRange [▶ 198]
Minimum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 0 • signed: -2147483648	INC	LINT
Maximum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 4294967295 • signed: 2147483647	INC	LINT
Feedback Velocity Source	HardwareProvidedIfAvailable		MC.EEncoderFeedbackVeloSource [▶ 198]

Position Interpretation

The `Position Interpretation` parameter defines the semantics according to which the values are transmitted by the position encoder and interpreted by the MC3 axis.

`Incremental`

The `Incremental` setting does not use the absolute properties of the physical feedback system, but can be used with both incremental and absolute encoder systems. After starting the controller, homing is required to initialize the actual position of the axis (`IsPositionValid = FALSE`).

`AbsoluteFullRange`

The `AbsoluteFullRange` setting is typically used with multi-turn or single-turn absolute encoders. The MC3 axis takes into account the entire data range of the data type in the I/O mapping (FullRange). The maximum number of increments corresponds to the upper limit of the data type. The size of the data type is specified in the protocol used.

When the controller is powered on or restarted and position limits are enabled, the axis signals `IsPositionValid = TRUE`. Configuring and enabling the software-based position limits for the axis allows the axis position to be restored after a controller restart. This applies to a maximum of one overflow of the transmitted increments. The position limits must be selected such that the resulting possible travel range of the axis is less than the encoder's counting range (FullRange).

If the position limits are disabled, the axis signals this with `IsPositionValid = FALSE`. A homing run is not required for AbsoluteFullRange.

`AbsolutUserDataRange`

The same boundary conditions apply as for the setting. `AbsoluteFullRange`. In addition, the parameter `Maximum Absolute Counter Value` is displayed and taken into account:

- `Maximum Absolute Counter Value`
  Unique maximum number of increments that the encoder system transmits (within the limits of the data type used). For example, the default value is set to 2^20-1.

`AbsoluteModulo`

The position increments are transmitted by the I/O device using modulo logic. For example, if one revolution of the encoder system corresponds to the modulus of the MC3 axis, the position is unique within this range. The axis indicates `IsPositionValid = TRUE`.

Position Data Range

The `Position Data Range` parameter defines the value range of the encoder system that is used for position calculation. This refers to the raw value received from the drive or encoder via the process data (PDO mapping) and enables the detection and handling of overflows. The parameter can only be changed at configuration time, not at runtime.

`FullRange`

Position overflow is based on the data type limits of the respective encoder system. For a signed 32-bit encoder system (e.g., CoE DS402 Drive (AX8000), SoE Drive (AX5000), EtherCAT encoder), this implicitly means a value range from -2147483648 to 2147483647. For an unsigned 32-bit encoder system (e.g., MDP742 Drive (selected ELM72xx), SSI encoder), this implies a value range from 0 to 4294967295.

`UserDefined`

This parameter value allows you to manually enter the range limits used to determine an overflow. Two additional parameters for entering the upper and lower limits will be displayed:

- Minimum Counter Value
- Maximum Counter Value

Feedback Velocity Source

Specifies the source of the actual velocity (AXIS_REF.McToPlc.Act.Velocity). The velocity can either be obtained from the hardware being used or calculated internally by differentiating the actual position value.

`HardwareProvidedIfAvailable` (Default value)

Uses the velocity provided by the hardware. If no entry for the actual velocity is mapped in the process data, it is automatically calculated internally.

`InternallyCalculated`

Velocity is always calculated from the derivative of the actual positions. Actual hardware velocities are ignored, even if they are available.

`HardwareProvided`

The velocity is always taken directly from the hardware feedback. This requires linking the actual velocity via process data (PDO mapping). If the mapping is missing, this will result in an error when the configuration is started.

Scaling

Name	Default value	Unit	Type
Position Scaling Numerator from Hardware	360	[PosUnit]/INC	LREAL
Position Scaling Denominator from Hardware	1048576.0		LREAL
Encoder Mounting Offset	0.0	[PosUnit]	LREAL
Direction Inverted	`FALSE`		BOOL
Velocity Scaling Numerator from Hardware	1.0	[PosUnit]/(INC*s)	LREAL
Velocity Scaling Denominator from Hardware	1.0		LREAL

The position scaling factor is divided into a numerator and a denominator. It is used to convert the path increments into axis positions or to calculate a user unit from encoder increments.

Position Scaling Numerator from Hardware

Position Scaling Numerator From Hardware is the feed rate that the application applies when the output shaft has completed one revolution.

Position Scaling Denominator from Hardware

Position Scaling Denominator From Hardware is the number of increments that the drive outputs when the motor shaft has completed one revolution.

Example 1 on position scaling:

Motor without gear unit with 100 mm pinion on AX5000 with default settings:

- Numerator: 100 mm
- Denominator: 1048576

Example 2 on position scaling:

Motor with gear unit with i=10 on turntable on AX5000 with default settings:

- Numerator: 360° / 10 = 36°
- Denominator: 1048576

Encoder Mounting Offset

An offset used to correct the position of an encoder system or align it within the machine coordinate system, thereby defining the machine-specific zero point. This offset is restored when the controller is restarted.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Velocity Scaling Numerator from Hardware

Used to calculate the actual velocity value from the encoder pulses.

Velocity Scaling Denominator from Hardware

Used to calculate the actual velocity value from the encoder pulses.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name		Default value	Unit	Type
Dead Time Compensation		NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift from Hardware	0.0	ms	LREAL

Dead Time Compensation

Specify how the compensated target value and actual value are to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift from Hardware

Additional offset of the automatically calculated dead time to compensate for the current position.

Connection

Name		Default value	Unit	Type
AMS address		0.0.0.0.0.0:0		
	.netId	0.0.0.0.0.0		AMSNETID
	.port	0x0000		WORD
Channel Number		0		UDINT
EtherCAT Slave Oid		00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Status Mask

Name	Default value	Unit	Type
Status Word Error Mask	0x0008		WORD

Actual Position Filter

Name		Default value	Unit	Type
Filter Type		None		MC.EFilterType [▶ 199]
	Filter Time Position	0.0	s	LREAL
	Filter Time Velocity	0.0	s	LREAL
	Filter Time Acceleration	0.01	s	LREAL
	Filter Time	0.001	s	LREAL

Filter Type

`None`

No filter is applied.

`PT1-Filter`

This is a first-order delay element (digital low-pass filter, proportional element with first-order delay) used to smooth out noisy signals (actual value). The PT1 filter is applied **independently** to the three actual values: position, velocity, and acceleration. Each variable has its own filter time constant. The filter time constants should be selected from the positive range of values. The value `0.0` disables filtering for the corresponding variable. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay. Because the variables are filtered separately, the output values are not necessarily consistent with each other.

`MovingAverage-Filter`

The moving-average filter calculates the arithmetic mean over a sliding time window of the last `N` input values and is thus a simple FIR low-pass filter (with a finite impulse response).
The moving-average filter is applied independently to the three actual values: position, velocity, and acceleration. Each variable has its own window length: `N`. The window length is calculated as the quotient of the parameterized filter time and the cycle time. The maximum window length is 1000 data points. The dead time is equal to half the window length.
Since the variables are filtered separately, the output values are not necessarily consistent with one another. With a filter time of `0.0`, the actual value is processed immediately and is not filtered. A long filter time results in significant smoothing and increases the dead time.

`KalmanFilter3x1`

The 3x1 Kalman filter estimates a set of position (s), velocity (v), and acceleration (a) values based on the continuous actual position values. On the one hand, this serves to reduce noise; on the other hand, the calculation of (s, v, a) is performed consistently, i.e., taking into account the interdependence of these variables. 3x1 indicates that the filter has three output variables (s, v, a) and one input variable (s).
It is often advantageous for (s, v, a) to be consistent with one another. In particular, the Kalman filter is recommended when coupling an axis to an encoder signal.
The 3x1 Kalman filter is configured using the “Filter Time” parameter, which defines the time scale used to calculate (s, v, a). The choice of this parameter represents a trade-off between noise reduction and dynamic response. A longer filter time results in greater noise reduction but a decrease in responsiveness, and vice versa (see the *Filter Time* parameter below).

Filter Time Position

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables position filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Velocity

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables velocity filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Acceleration

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables acceleration filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time

The parameter is active when the `Filter Type` is set to `KalmanFilter3x1`. The Kalman filter is adapted to the application using a simplified parameterization with a single filter time constant. The larger the filter time constant, the greater the degree of smoothing.

3.2.3.9.1.2 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

EtherCAT Encoder Inputs

Enable (default)	Name	Type	Description
☑	Position	DINT	Actual position in increments.
☐	Velocity	DINT	Actual speed in increments
☑	Status	WORD	Current status of the slave
☑	WcState	BIT	Working counter. Indicates whether there was valid real-time communication during the last cycle.
☑	InputToggle	BIT	Level change upon receipt of a new telegram.
☐	DcInputShift	DINT	Used for dead-time compensation of the drive.

3.2.3.9.2 Analog Load Encoder

Module for linking an MC3 axis to an analog encoder.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.9.2.1 Init Parameter

General

Name	Default value	Unit	Type
Load Type	None		MC.ELoadType [▶ 200]

Load Type

Specifies which load variable this encoder measures (e.g., Force, PressureA, PressureB).

Scaling

Name	Default value	Unit	Type
Direction Inverted	FALSE		BOOL
Load Scaling Factor Numerator from Hardware	0.001	[[LoadUnit]/INC]	LREAL
Load Scaling Factor Denominator from Hardware	1.0		LREAL
Load Offset	0.0	[LoadUnit]	LREAL

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Load Scaling Factor Numerator from Hardware

This value is used to convert sensor increments into the actual load value.

Load Scaling Factor Denominator from Hardware

This value is used to convert sensor increments into the actual load value.

Load Offset

Offset used to specify a machine-dependent zero point.

Connection

Name		Default value	Unit	Type
AMS address		0.0.0.0.0.0:0		
	.netId	0.0.0.0.0.0		AMSNETID
	.port	0x0000		WORD
Channel Number		0		UDINT
EtherCAT Slave Oid		00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Actual Load Filter

Name		Default value	Unit	Type
Filter Type		None		MC.EFilterType [▶ 199]
	Filter Time Load	0.0	s	LREAL
	Filter Time Load Derivative	0.005	s	LREAL
	Filter Time Load Second Derivative	0.01	s	LREAL
	Filter Time	0.001	s	LREAL

Filter Type

Defines the filter algorithm. The following types are available:

- None: No filter is applied.
- PT1: A first-order low-pass filter.
- MovingAverage: Outputs the average value of the input signal over the filter time.
- KalmanFilter3x1: FIR low-pass filter

Filter Time Load

Increasing positive values increase the smoothing and delay of the actual load; zero means no filtering.

Filter Time Load Derivative

Increasing positive values increase the smoothing and delay of the actual load derivative; zero means no filtering.

Filter Time Load Second Derivative

Increasing positive values increase the smoothing and delay of the second derivative of the actual load; zero means no filtering.

Filter Time

Filter time for the Kalman filter. The value is squared internally to obtain the covariance of the observational noise (R_sigma_s). Higher values produce a smoother output signal, while lower values track the measured signal more closely.

3.2.3.9.2.2 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

Analog inputs

Enable (default)	Name	Type	Description
☑	Underrange	BIT	The analog input signal is below the lower permissible threshold for the terminal.
☑	Overrange	BIT	The analog input signal exceeds the upper permissible threshold for the terminal.
☑	Error	BIT	Data invalid

Enable (default)	Name	Type	Description
☑	Counter value	INT	Meter reading
☑	WcState	BIT	Working counter

3.2.3.9.3 SSI Encoder

Module for linking an MC3 axis to an SSI encoder.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.9.3.1 Init Parameter

General

Name	Default value	Unit	Type
Position Interpretation	Incremental		MC.EEncoderInterpretation [▶ 198]
Maximum Absolute Counter Value (Visibility depends on position interpretation)	1048575	INC	UDINT
Position Data Range	FullRange		MC.EEncoderDataRange [▶ 198]
Maximum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 4294967295 • signed: 2147483647	INC	LINT
Minimum Counter Value (Visibility depends on the position data range setting)	32-bit encoder system • unsigned: 0 • signed: -2147483648	INC	LINT

Position Interpretation

The `Position Interpretation` parameter defines the semantics according to which the values are transmitted by the position encoder and interpreted by the MC3 axis.

`Incremental`

The `Incremental` setting does not use the absolute properties of the physical feedback system, but can be used with both incremental and absolute encoder systems. After starting the controller, homing is required to initialize the actual position of the axis (`IsPositionValid = FALSE`).

`AbsoluteFullRange`

The `AbsoluteFullRange` setting is typically used with multi-turn or single-turn absolute encoders. The MC3 axis takes into account the entire data range of the data type in the I/O mapping (FullRange). The maximum number of increments corresponds to the upper limit of the data type. The size of the data type is specified in the protocol used.

When the controller is powered on or restarted and position limits are enabled, the axis signals `IsPositionValid = TRUE`. Configuring and enabling the software-based position limits for the axis allows the axis position to be restored after a controller restart. This applies to a maximum of one overflow of the transmitted increments. The position limits must be selected such that the resulting possible travel range of the axis is less than the encoder's counting range (FullRange).

If the position limits are disabled, the axis signals this with `IsPositionValid = FALSE`. A homing run is not required for AbsoluteFullRange.

`AbsolutUserDataRange`

The same boundary conditions apply as for the setting. `AbsoluteFullRange`. In addition, the parameter `Maximum Absolute Counter Value` is displayed and taken into account:

- `Maximum Absolute Counter Value`
  Unique maximum number of increments that the encoder system transmits (within the limits of the data type used). For example, the default value is set to 2^20-1.

`AbsoluteModulo`

The position increments are transmitted by the I/O device using modulo logic. For example, if one revolution of the encoder system corresponds to the modulus of the MC3 axis, the position is unique within this range. The axis indicates `IsPositionValid = TRUE`.

Position Data Range

The `Position Data Range` parameter defines the value range of the encoder system that is used for position calculation. This refers to the raw value received from the drive or encoder via the process data (PDO mapping) and enables the detection and handling of overflows. The parameter can only be changed at configuration time, not at runtime.

`FullRange`

Position overflow is based on the data type limits of the respective encoder system. For a signed 32-bit encoder system (e.g., CoE DS402 Drive (AX8000), SoE Drive (AX5000), EtherCAT encoder), this implicitly means a value range from -2147483648 to 2147483647. For an unsigned 32-bit encoder system (e.g., MDP742 Drive (selected ELM72xx), SSI encoder), this implies a value range from 0 to 4294967295.

`UserDefined`

This parameter value allows you to manually enter the range limits used to determine an overflow. Two additional parameters for entering the upper and lower limits will be displayed:

- Minimum Counter Value
- Maximum Counter Value

Scaling

Name	Default value	Unit	Type
Position Scaling Numerator from Hardware	360	[PosUnit]/INC	LREAL
Position Scaling Denominator from Hardware	1048576.0		LREAL
Encoder Mounting Offset	0.0	[PosUnit]	LREAL
Direction Inverted	`FALSE`		BOOL

The position scaling factor is divided into a numerator and a denominator. It is used to convert the path increments into axis positions or to calculate a user unit from encoder increments.

Position Scaling Numerator from Hardware

Position Scaling Numerator From Hardware is the feed rate that the application applies when the output shaft has completed one revolution.

Position Scaling Denominator from Hardware

Position Scaling Denominator From Hardware is the number of increments that the drive outputs when the motor shaft has completed one revolution.

Example 1 on position scaling:

Motor without gear unit with 100 mm pinion on AX5000 with default settings:

- Numerator: 100 mm
- Denominator: 1048576

Example 2 on position scaling:

Motor with gear unit with i=10 on turntable on AX5000 with default settings:

- Numerator: 360° / 10 = 36°
- Denominator: 1048576

Encoder Mounting Offset

An offset used to correct the position of an encoder system or align it within the machine coordinate system, thereby defining the machine-specific zero point. This offset is restored when the controller is restarted.

Direction Inverted

The `Direction Inverted` parameter can reverse the counting direction of the encoder or drive:

- `FALSE`: The polarity of the axis movement agrees with the counting direction of the acquisition hardware.
- `TRUE`: The polarity of the axis movement is in the opposite sense to that of the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the counting direction of the encoder and the motor polarity do not match, the axis will perform unexpected movements.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Dead Time Compensation

A requirement for high-precision conversion of positions into times and vice versa is precise dead time compensation of the axes.

Name		Default value	Unit	Type
Dead Time Compensation		NonlinearShift		MC.EDeadTimeCompensationMode [▶ 197]
	Additional Time Shift from Hardware	0.0	ms	LREAL

Dead Time Compensation

Specify how the compensated target value and actual value are to be calculated. The dead time is automatically determined during EtherCAT initialization.

Additional Time Shift from Hardware

Additional offset of the automatically calculated dead time to compensate for the current position.

Connection

Name	Default value	Unit	Type
AMS address	0.0.0.0.0.0:0		
.netId	0.0.0.0.0.0		AMSNETID
.port	0x0000		WORD
Channel Number	0		UDINT
EtherCAT Slave Oid	00000000		OTCID

AMS address

Information about the address of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

Channel Number

Information about the channel number of the assigned hardware drive (read-only); no manual configuration is required. It is used for communication purposes.

EtherCAT Slave Oid

Object ID of the EtherCAT slave.

Actual Position Filter

Name	Default value	Unit	Type
Filter Type	None		MC.EFilterType [▶ 199]
Filter Time Position	0.0	s	LREAL
Filter Time Velocity	0.0	s	LREAL
Filter Time Acceleration	0.01	s	LREAL
Filter Time	0.001	s	LREAL

Filter Type

`None`

No filter is applied.

`PT1-Filter`

This is a first-order delay element (digital low-pass filter, proportional element with first-order delay) used to smooth out noisy signals (actual value). The PT1 filter is applied **independently** to the three actual values: position, velocity, and acceleration. Each variable has its own filter time constant. The filter time constants should be selected from the positive range of values. The value `0.0` disables filtering for the corresponding variable. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay. Because the variables are filtered separately, the output values are not necessarily consistent with each other.

`MovingAverage-Filter`

The moving-average filter calculates the arithmetic mean over a sliding time window of the last `N` input values and is thus a simple FIR low-pass filter (with a finite impulse response).
The moving-average filter is applied independently to the three actual values: position, velocity, and acceleration. Each variable has its own window length: `N`. The window length is calculated as the quotient of the parameterized filter time and the cycle time. The maximum window length is 1000 data points. The dead time is equal to half the window length.
Since the variables are filtered separately, the output values are not necessarily consistent with one another. With a filter time of `0.0`, the actual value is processed immediately and is not filtered. A long filter time results in significant smoothing and increases the dead time.

`KalmanFilter3x1`

The 3x1 Kalman filter estimates a set of position (s), velocity (v), and acceleration (a) values based on the continuous actual position values. On the one hand, this serves to reduce noise; on the other hand, the calculation of (s, v, a) is performed consistently, i.e., taking into account the interdependence of these variables. 3x1 indicates that the filter has three output variables (s, v, a) and one input variable (s).
It is often advantageous for (s, v, a) to be consistent with one another. In particular, the Kalman filter is recommended when coupling an axis to an encoder signal.
The 3x1 Kalman filter is configured using the “Filter Time” parameter, which defines the time scale used to calculate (s, v, a). The choice of this parameter represents a trade-off between noise reduction and dynamic response. A longer filter time results in greater noise reduction but a decrease in responsiveness, and vice versa (see the *Filter Time* parameter below).

Filter Time Position

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables position filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Velocity

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables velocity filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time Acceleration

This parameter is enabled when the `Filter Type` is set to `PT1` or `MovingAverage`. The value `0.0` disables acceleration filtering. The filter time constant must be selected from the positive range of values. The larger the filter time constant, the more the actual value is smoothed, and the greater the delay.

Filter Time

The parameter is active when the `Filter Type` is set to `KalmanFilter3x1`. The Kalman filter is adapted to the application using a simplified parameterization with a single filter time constant. The larger the filter time constant, the greater the degree of smoothing.

3.2.3.9.3.2 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

SSI Inputs

Enable (default)	Name	Type	Description
☑	Counter Value	UDINT	Meter reading
☑	Data error	BIT	SSI Input Error
☑	Frame error	BIT	The data frame is incorrect; that is, the data frame was not terminated with a zero (possibly a broken wire in the clock line).
☑	Power failure	BIT	A sensor-related error is displayed if the sensor was previously enabled.
☑	Sync error	BIT	The synchronization error bit is only required in DC mode. It indicates whether a synchronization error occurred in the previous cycle. This means that a SYNC signal was triggered at the terminal even though no new process data was available.

Enable (default)	Name	Type	Description
☑	TxPDO state	BIT	This element indicates whether the values measured by the encoder are in an invalid state.
☑	TxPDO toggle	BIT	TxPDO Toggle is toggled by the slave when the data for the corresponding TxPDO is updated.
☑	WcState	BIT	Working counter
☑	InputToggle	BIT	Level change upon receipt of a new telegram.
☐	DcInputShift	DINT	Used for dead-time compensation of the drive.

3.2.3.10 Controller modules

The following section contains information about the available controller modules.

3.2.3.10.1 Load Controller

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.10.1.1 Init Parameter

General

Name	Default value	Unit	Type
Control Value Type	Velocity		MC.EControlValueType [▶ 196]

Control Value Type

Specifies the type of the controller's output variable. For example, `Velocity` means that the controller output is a velocity.

Control Parameter

Name	Default value	Unit	Type
Inverted Output	FALSE		BOOL
Proportional Gain (Kp)	0.0	[ControlVariableUnit]/[LoadUnit]	LREAL
Derivative Gain (Kd)	0.0	[ControlVariableUnit]/([LoadUnit]/s)	LREAL
Integral Gain (AI)	0.0	[ControlVariableUnit]/([LoadUnit]*s)	LREAL
Integrator Lag Window Minimum	#Ignore	[LoadUnit]	LREAL
Integrator Lag Window Maximum	#Ignore	[LoadUnit]	LREAL

Inverted Output

The parameter `Output Inverted` can reverse the sign of the output variable.

⚠ WARNING

Risk of injury from unexpected movements

If the controller's output is not properly matched to the controlled system, the axis may move unexpectedly.

- When commissioning the system, make sure that the drive's direction of rotation is set correctly.
- Always maintain a safe distance.

Proportional Gain

The proportional gain (Kp) is multiplied by the current control error, resulting in a term in the controlled variable.

Derivative Gain

The derived gain (Kd) is multiplied by the derivative of the error, resulting in a term in the controlled variable.

Integral Gain

The integral gain (Ki) is multiplied by the integral of the control error, resulting in a term in the controlled variable.

Integrator Lag Window Minimum

If the actual load delay is less than the `Integrator Lag Window Minimum`, the integrator does not change. The value must be greater than or equal to 0.0 and less than or equal to `Integrator Lag Window Maximum`.

Integrator Lag Window Maximum

If the actual load delay is greater than the `Integrator Lag Window Maximum`, the integrator does not change. The value must equal or greater 0.0.

Output Limitation

Name	Default value	Unit	Type
Integrator Minimum Output	#Ignore	[ControlVariableUnit]	LREAL
Integrator Maximum Output	#Ignore	[ControlVariableUnit]	LREAL
Minimum output	#Ignore	[ControlVariableUnit]	LREAL
Maximum output	#Ignore	[ControlVariableUnit]	LREAL

Integrator Minimum Output

Minimum output of the integrator. The value `Integrator Minimum Output` must be less than or equal to the value `Integrator Maximum Output`.

Integrator Maximum Output

Maximum output of the integrator. The `Integrator Maximum Output` value must be greater than or equal to the `Integrator Minimum Output` value.

Minimum output

Minimum output of the controller. The `Minimum Output` value must be less than or equal to the `Maximum Output` value.

Maximum output

Maximum output of the controller. The `Maximum Output` value must be greater than or equal to the `Minimum Output` value.

3.2.3.10.1.2 Online Parameter

Name	Unit	Type
Control Variable	[ControlVariableUnit]	LREAL
Control Variable Proportional Part	[ControlVariableUnit]	LREAL
Control Variable Derivative Part	[ControlVariableUnit]	LREAL
Control Variable Integral Part	[ControlVariableUnit]	LREAL

Control Variable

The controlled variable used to achieve the desired setpoint.

Control Variable Proportional Part

The proportional component of the controlled variable.

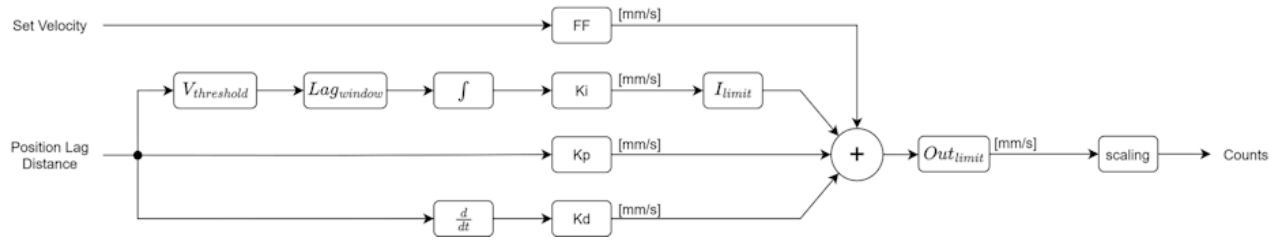
Control Variable Derivative Part

The derivative component of the controlled variable.

Control Variable Integral Part

The integral component of the control variable.

3.2.3.10.2 Position Controller



Name	Explanation
FF	Feed Forward - The velocity setpoint is multiplied by this factor and added to the controller's total output.
$V_{threshold}$	If the actual velocity is greater than the Integrator Threshold Maximum Velocity [▶ 96] parameter or less than the Integrator Threshold Minimum Velocity [▶ 96] parameter, the integrator's control component is no longer adjusted. The Integrator Threshold Minimum Velocity [▶ 96] is usually a negative value.
Lag_{window}	If the magnitude of the position lag is less than the Integrator Lag Window Minimum [▶ 96] parameter or greater than the Integrator Lag Window Maximum [▶ 96] parameter, the integrator's control component is no longer adjusted. Integrator Lag Window Minimum [▶ 96] and Integrator Lag Window Maximum [▶ 96] must be positive.
I_{Limit}	Limit on the integral component of the controller
Out_{Limit}	Limit on the controller's output variable

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.10.2.1 Init Parameter

General

Name	Default value	Unit	Type
Control Value Type	Velocity		MC.EControlValueType [▶ 196]
Enabled	TRUE		BOOL
Velocity Feed-Forward Factor	1.0		LREAL

Control Value Type

Specifies the type of the controller's output variable. For example, `Velocity` means that the controller output is a velocity.

Enabled

`Enabled` indicates whether the position controller is being used or not. The controller output is 0.0 when it is disabled.

Velocity Feed-Forward Factor

The product of `Velocity Feed Forward Factor` and the velocity setpoint is added to the sum of P, I, and D to produce the controlled variable.

Control Parameter

Name	Default value	Unit	Type
Proportional Gain (Kp)	1.0	[ControlVariableUnit] / [PosUnit]	LREAL
Derivative Gain (Kd)	0.0	[ControlVariableUnit]/ ([PosUnit] / s)	LREAL
Integral Gain (AI)	0.0	[ControlVariableUnit] / ([PosUnit] * s)	LREAL
Integrator Threshold Minimum Velocity	#Ignore	[PosUnit]/s	LREAL
Integrator Threshold Maximum Velocity	#Ignore	[PosUnit]/s	LREAL
Integrator Lag Window Minimum	#Ignore	[PosUnit]	LREAL
Integrator Lag Window Maximum	#Ignore	[PosUnit]	LREAL

Proportional Gain (Kp)

The proportional gain (Kp) is multiplied by the current control error, resulting in a term in the controlled variable.

Derivative Gain (Kd)

The derived gain (Kd) is multiplied by the derivative of the error, resulting in a term in the controlled variable.

Integral Gain (AI)

The integral gain (Ki) is multiplied by the integral of the control error, resulting in a term in the controlled variable.

Integrator Threshold Minimum Velocity

The integrator does not change if the actual velocity is less than the minimum velocity. The minimum velocity must be less than or equal to 0.0.

`Integrator Threshold Minimum Velocity ≤ 0,0`

Integrator Threshold Maximum Velocity

The integrator does not change if the current velocity is greater than the maximum velocity. The maximum velocity must be greater than 0.0.

`Integrator Threshold Maximum Velocity > 0,0`

Integrator Lag Window Minimum

The integrator does not change if the actual delay falls outside this interval. The value for the integrator's minimum delay must be greater than 0.0.

`Integrator Lag Window Minimum > 0,0`

Integrator Lag Window Maximum

The integrator does not change if the actual delay falls outside this interval. The value for the integrator's maximum delay must be greater than the minimum delay.

`Integrator Lag Window Maximum > Integrator Lag Window Minimum`

Output Limitation

Name	Default value	Unit	Type
Integrator Minimum Output	#Ignore	[ControlVariableUnit]	LREAL
Integrator Maximum Output	#Ignore	[ControlVariableUnit]	LREAL
Minimum output	#Ignore	[ControlVariableUnit]	LREAL
Maximum output	#Ignore	[ControlVariableUnit]	LREAL

Integrator Minimum Output

Minimum output of the integrator. The value `Integrator Minimum Output` must be less than or equal to the value `Integrator Maximum Output`.

Integrator Maximum Output

Maximum output of the integrator. The `Integrator Maximum Output` value must be greater than or equal to the `Integrator Minimum Output` value.

Minimum output

Minimum output of the controller. The `Minimum Output` value must be less than or equal to the `Maximum Output` value.

Maximum output

Maximum output of the controller. The `Maximum Output` value must be greater than or equal to the `Minimum Output` value.

3.2.3.10.2.2 Online Parameter

Name	Unit	Type
Control Value Final Output	[ControlVariableUnit]	LREAL
Control Value	[ControlVariableUnit]	LREAL
Control Variable Proportional Part	[ControlVariableUnit]	LREAL
Control Variable Derivative Part	[ControlVariableUnit]	LREAL
Control Variable Integral Part	[ControlVariableUnit]	LREAL

Control Value Final Output

The sum of the P-component, I-component, D-component, and FF-component.

Control Value

The controlled variable used to achieve the desired setpoint.

Control Variable Proportional Part

The proportional component of the controlled variable.

Control Variable Derivative Part

The derivative component of the controlled variable.

Control Variable Integral Part

The integral component of the control variable.

3.2.3.11 Fluid Power modules

The components of the FluidPower module are described below.

3.2.3.11.1 FluidPower Axis Parameters

FluidPower axis parameters.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.11.1.1 Init Parameter

Cylinder Dimensions

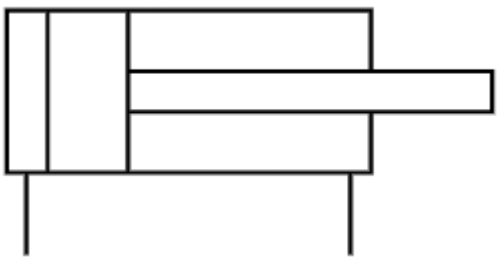
Name	Default value	Unit	Type
Cylinder Design	Differential		MC.ECylinderDesign
Cylinder Bore Diameter	`#Ignore`	mm	LREAL
Cylinder Rod Diameter	`#Ignore`	mm	LREAL

Cylinder Design

The `Cylinder Design` parameter selects the type of hydraulic cylinder and determines how the piston and piston rod diameters are used for force and area calculations.

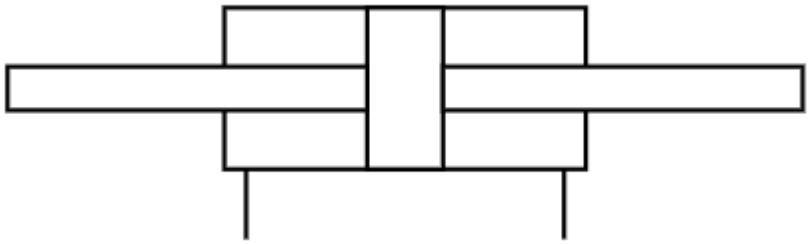
`Differential`

The differential cylinder is a double-acting hydraulic cylinder with a single-ended piston rod.



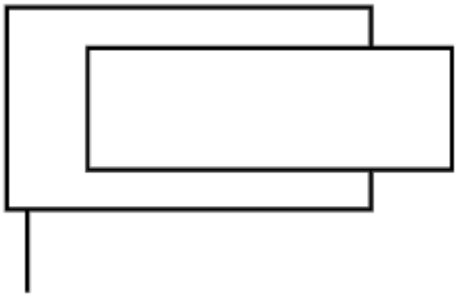
`Synchronous`

The synchronous cylinder is a double-acting hydraulic cylinder with a through-rod.



`Plunger`

The plunger cylinder is a single-acting hydraulic cylinder without a piston rod.



Cylinder Bore Diameter

Inner diameter of the hydraulic cylinder bore. Used to calculate the piston area and the force.

Cylinder Rod Diameter

Diameter of the cylinder rod. Relevant for the design of differential and synchronous cylinders when calculating the effective piston area on the rod side.

3.2.3.11.2 Valve Characteristic Curve

Settings and parameters on the Valve Characteristic Curve.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.11.2.1 Init Parameter

Name	Default value	Unit	Type
Valve Linearization Enabled	`TRUE`		BOOL

This parameter allows the drive output to be linearized in order to compensate for the valve's nonlinear behavior.

3.2.3.11.2.2 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

McToPlc (Output)

Name	Type	Description
Oid	OTCID	Object ID for linking to a valve characteristic curve in the PLC.

3.2.3.12 Touch probe modules

The use of touch probe modules allows the detection of an axis position at the time of a trigger event. This can be a digital signal that replicates the function of a touch probe. It is also possible to determine the timestamp at which the axis reaches a defined position.

Add a touch probe module

Follow the steps below to add *additional* touch probe modules. Drive modules, such as the DS402 (AX8xxx), MDP742 (ELM72xx), or Sercos (AX5xxx), already include touch probe functions. No additional touch probe modules need to be added; you simply need to enable the cyclic process data and link it to the appropriate I/O device (either manually or through configuration in Drive Manager). In the PLC, a corresponding `TRIGGER_REF` (note derived ref objects) can be linked.

1. Right-click **Axes > Axis > Add New Item...**

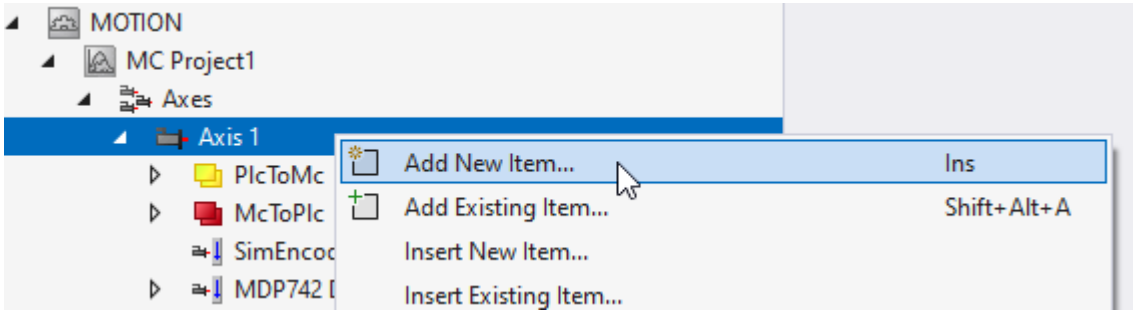
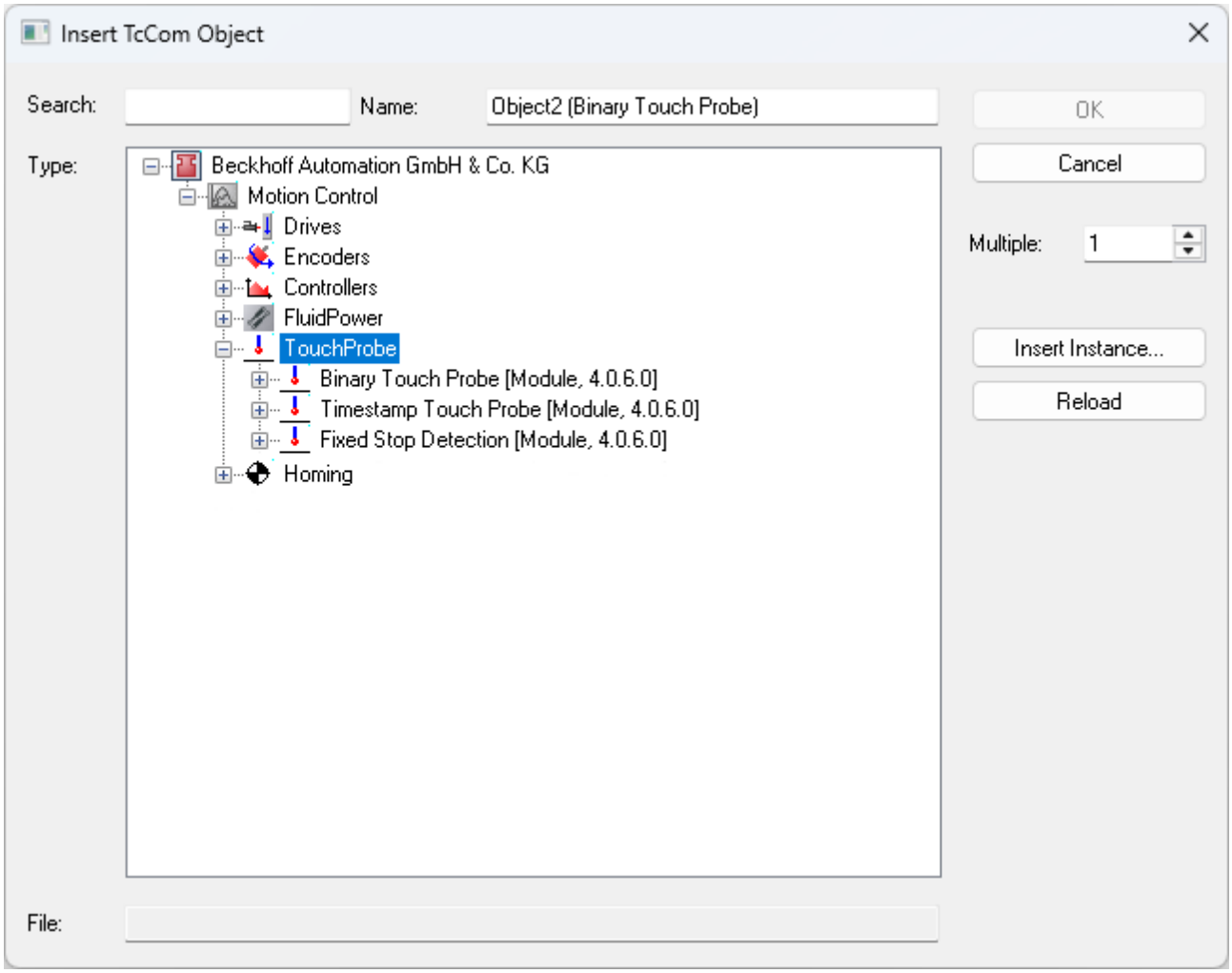


Fig. 1:

2. In the **Insert TcCOM Object** dialog box, navigate to the **TouchProbe** group under **Type** and the **Motion Control** group.
3. Expand this group if necessary.



4. Select the appropriate touch probe module and complete the process by clicking the **OK** button. Be sure to select the same module version as the one used for existing Motion Control objects in the Solution.

3.2.3.12.1 Binary Touch Probe

The Binary Touch Probe module records an axis position at the time of a digital signal (touch probe function). The position is typically not detected directly within the PLC environment, but rather via an external hardware signal. The signal input is evaluated within the context of the axis's MET task [▶ 19].

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.12.1.1 Init Parameter

Binary Touch Probe

Name	Default value	Unit	Type
Input Inverted	FALSE		BOOL

Invert Input

If the value of the `Input Inverted` parameter is set to `TRUE`, the MC's internal algorithm inverts the logic of the input Signal [▶ 102] for evaluation by the touch probe unit.

3.2.3.12.1.2 Online Parameter

Positive Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

Negative Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.12.1.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

Inputs

Enable (default)	Name	Type	Description
☑	Signal	BIT	Input for the trigger signal. For example, it can be linked to a variable from the PLC or to a digital input on an I/O terminal.
☐	WcState	BIT	Working counter

ToPlc

Enable (default)	Name	Type	Description
☑	Oid	OTCID	Object ID for linking to a touch probe object in the PLC

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.12.2 Timestamp Touch Probe

The Timestamp Touch Probe module determines an axis position at the time specified by an application-defined timestamp.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.12.2.1 Online Parameter

Touch Probe Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.12.2.2 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

Inputs

Enable (default)	Name	Type	Description
☑	Timestamp 32bit	UDINT	A timestamp whose value changes are evaluated by the touch probe algorithm.
☐	Timestamp 64bit	ULINT	A timestamp whose value changes are evaluated by the touch probe algorithm.
☐	WcState	BIT	Working counter

NOTICE

Warning in the error log window due to multiple selection of input elements

If both the 32-bit and 64-bit timestamp inputs are enabled and linked, this will result in a warning in the error log window.

- Select a timestamp element based on the hardware you are using or the timestamp you want to link to (especially based on its data type).

ToPlc

Enable (default)	Name	Type	Description
☑	Oid	OTCID	Object ID for linking to a touch probe object in the PLC

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.12.3 Fixed Stop Detection

The Fixed Stop Detection module determines an axis position, among other things, when a torque/force threshold value is reached. The reference value is the torque/force input of the axis's driven object. The torque/force is evaluated in the context of the axis's MET task [▶ 19].

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.12.3.1 Init Parameter

Detection Thresholds

Name	Default value	Unit	Type
Torque/Force Threshold	10.0	%	LREAL
Velocity Threshold	10.0	[PosUnit]/s	LREAL

Name	Default value	Unit	Type
Lag Threshold	2.0	[PosUnit]	LREAL

Torque/Force Threshold

Threshold value for the actual motor torque value. If the torque exceeds this value during evaluation, this serves as an indicator that a hard stop has been detected. The value `#Ignore` disables this threshold.

Velocity Threshold

Threshold value for the actual axis velocity during evaluation. The power/torque threshold is evaluated only if the actual velocity is below this threshold value. The value `#Ignore` disables this threshold.

Lag Threshold

Threshold value for the position lag. If the actual position lag exceeds this value during evaluation, this serves as an indicator that a hard stop has been detected. The value `#Ignore` disables this threshold.

Disabling all thresholds

If all thresholds are disabled, the module immediately reports a detected event. In the case of homing, this is instantly recognized as the encoder having reached the end stop.

- Enter at least a real number for the torque/force threshold or position lag threshold to enable hard stop detection.

Behavior of an accelerating axis with a velocity threshold

If, for example, the axis accelerates from a standstill and the touch probe module's evaluation is already active, the axis's velocity may still be below the set threshold value. Depending on the other threshold values, this may cause the touch probe module to report a detected event. In the case of homing, this can be detected as reaching the end stop.

- If necessary, start an axis movement before activating the touch probe unit.

3.2.3.12.3.2 Online Parameter

Touch Probe Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.12.3.3 Data Area

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

ToPlc

Enable (default)	Name	Type	Description
☑	Oid	OTCID	Object ID for linking to a touch probe object in the PLC

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

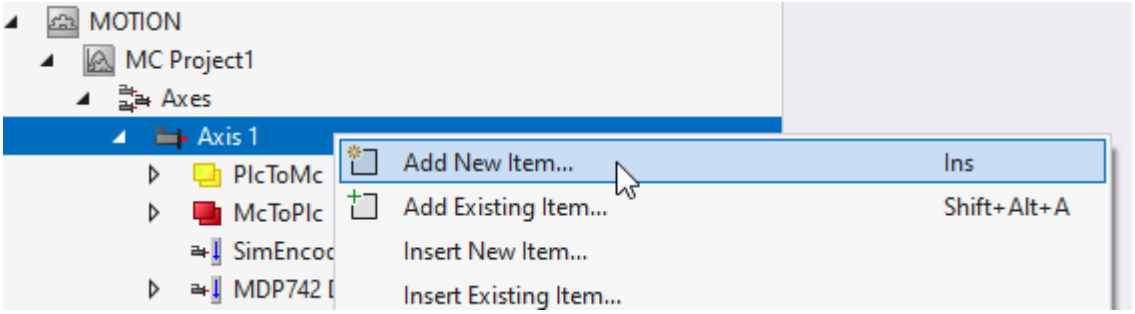
3.2.3.13 Homing modules

Homing refers to a process of referencing an axis in which the correct actual position is determined based on a reference signal. The process can be carried out using the MC. A successfully referenced axis signals `McToPlc.Std.IsPositionValid = TRUE.` in the cyclic interface.

The basic configuration is performed in TcCOM modules. The configuration includes the phases of a homing procedure (e.g., movement toward the reference signal or movement until a synchronization event occurs) as well as the associated boundary conditions in the form of object parameters. A homing module can optionally be added to an axis as a child object if the axis requires a homing procedure to operate. One homing module can be configured per axis.

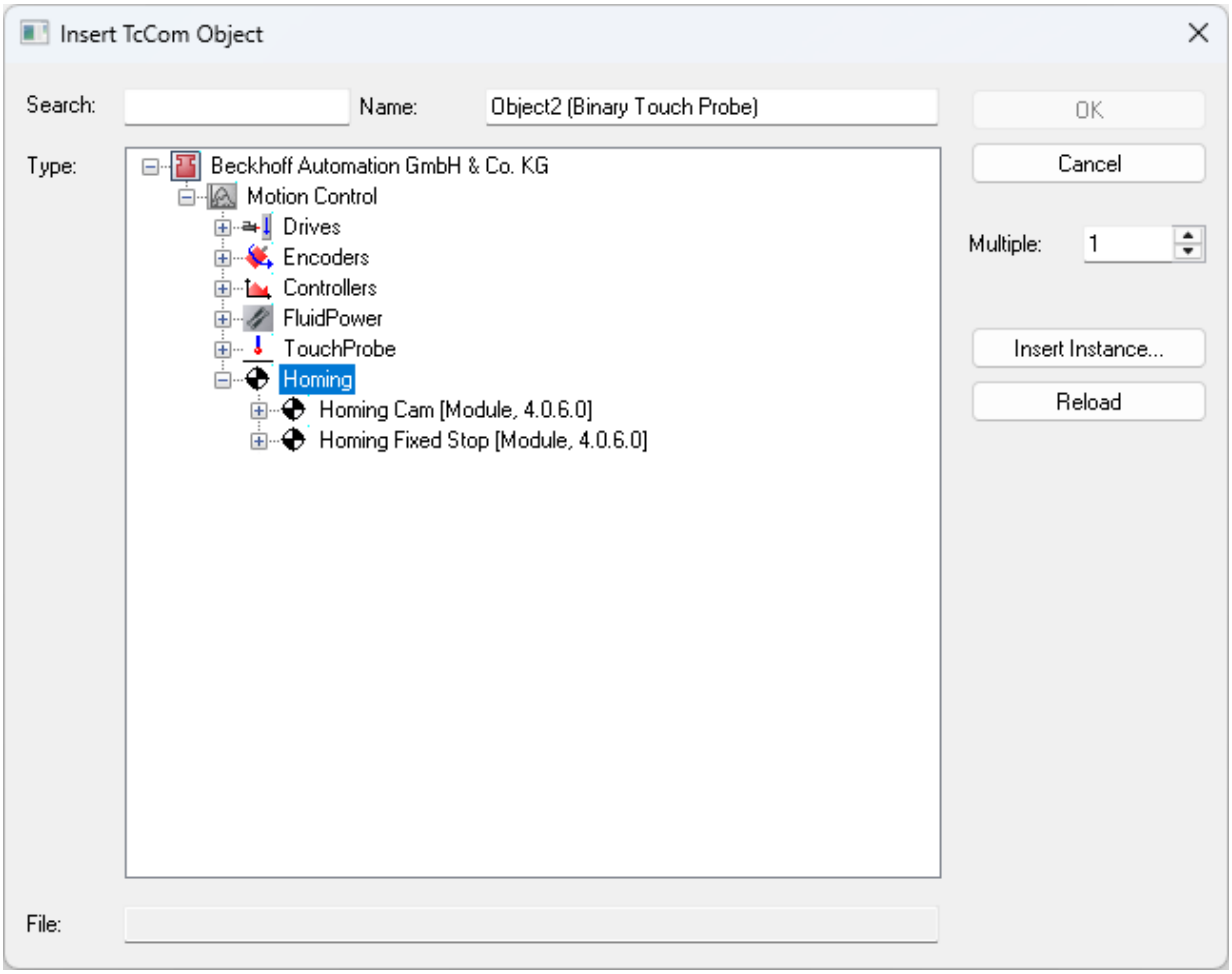
Add homing module

1. In the MC project, right-click on **Axes > Axis > Add New Item…** in the structure tree.



2. In the **Insert TcCOM Object** dialog box, navigate to the **Homing** group under **Type** and the **Motion Control** group.

3. Expand this group if necessary.

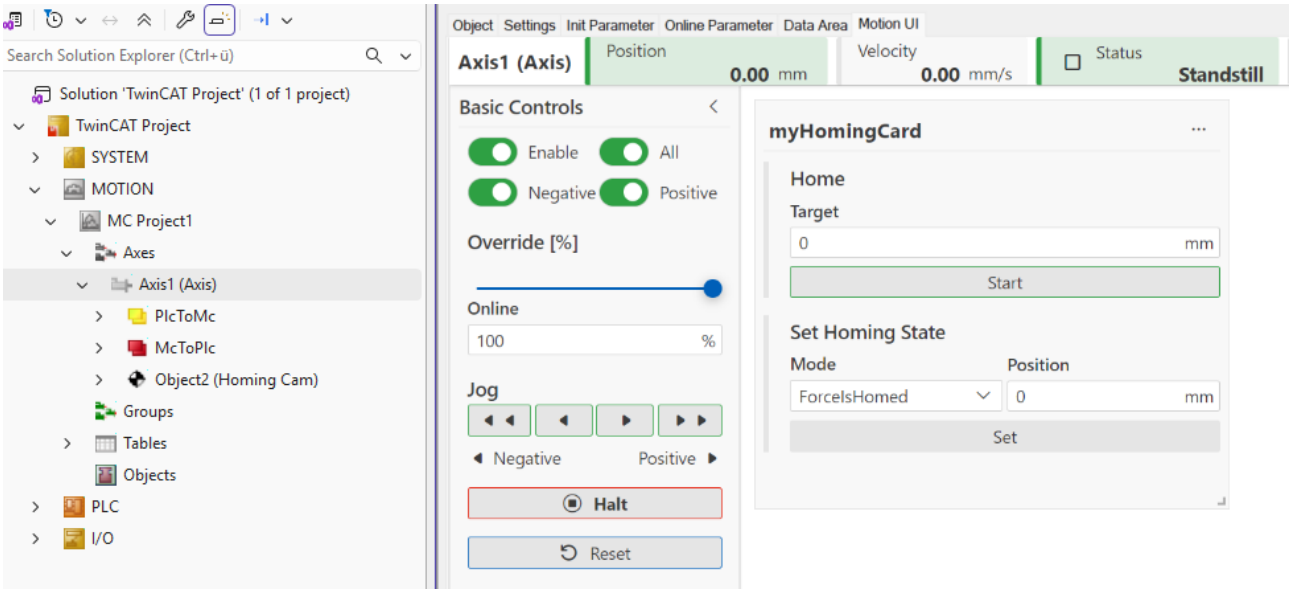


4. Select the appropriate homing module and complete the process by clicking the **OK** button.

Be sure to select the same module version as the one used for existing Motion Control objects in the Solution.

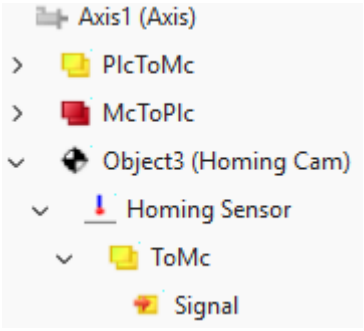
Start the homing procedure

The sequence configured in the homing module can be started in the PLC using the MC_Home [▶ 283] function block. Alternatively, this can also be done using Home Control within a Motion UI card.



3.2.3.13.1 Homing Cam

The Homing Cam object allows you to use a digital input as a reference signal for an initialization movement. To do this, a Binary Touch Probe object [▶ 101] is nested within the object. This is how the Homing Cam device is used, for example, to link the signal from a digital input terminal or a PLC output as a reference signal. Signal analysis takes place within the context of the MET task [▶ 19]. The homing procedure is configured in the Parameters (Init) [▶ 108] tab.



View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.13.1.1 Init Parameter

General

Homing Position

Name	Default value	Unit	Type
Homing Position	0.0	[PosUnit]	LREAL

The parameter `Homing Position` defines the absolute reference position. The axis is set to this position after a successful homing procedure. This parameter has an effect when the value `#Default` is used for the reference position input in the command to start a homing procedure (e.g., MC_Home).

Sequence

A homing process has several phases. The following parameters can be used to enable and configure individual phases.

Move to Cam Enabled

Name		Default value	Unit	Type
Move to Cam Enabled		FALSE		BOOL
	Move to Cam Distance	10.0	[PosUnit]	LREAL
	Move to Cam Velocity	30.0	[PosUnit]/s	LREAL

Enabling the `Move To Cam Enabled` parameter (value: `TRUE`) adds a sequence in the form of an axis movement to the initialization process. The system is searching for a reference signal. The axis continues to move until a corresponding value is detected at the input Signal [▶ 102]. The logic of the input depends on the Invert Input [▶ 110] parameter.

- `Move To Cam Distance`
  If the expected event is not detected at the Signal [▶ 102] input within this distance (measured from the starting position of the homing process), the homing process is aborted. The sign of the value determines the direction of movement of the axis.

- `Move To Cam Velocity`
  During the sequence, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.

Move from Cam Enabled

Name		Default value	Unit	Type
Move from Cam Enabled		FALSE		BOOL
	Move from Cam Distance	-10.0	[PosUnit]	LINT
	Move from Cam Velocity	30.0	[PosUnit]/s	LINT

Enabling the `Move From Cam Enabled` parameter (value: `TRUE`) adds a sequence in the form of an axis movement to the initialization process. The axis continues to move until the reference signal at the Signal [▶ 102] input returns to its inactive state. The logic of the input Signal [▶ 102] depends on the Invert Input [▶ 110] parameter.

- `Move From Cam Distance`
  If the expected event is not detected at the *Signal* input within this distance (measured from the starting position of the homing process), the homing process is aborted. The sign of the value determines the direction of movement of the axis.
- `Move From Cam Velocity`
  During the sequence, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.

Move to Sync Enabled

Name		Default value	Unit	Type
Move to Sync Enabled		FALSE		BOOL
	Move to Sync Distance	10.0	[PosUnit]	LREAL
	Move to Sync Velocity	30.0	[PosUnit]/s	LREAL
	Move to Sync Source	ZeroImpulseCoE		MC.EHomingSyncSource [▶ 200]
	Move to Sync Edge	Positive		MC.ETouchProbeTriggerSource [▶ 203]

Enabling the `Move To Sync Enabled` parameter (value: `TRUE`) adds a sequence in the form of an axis movement to the initialization process. The axis continues to move until a configured event is detected in touch probe unit 1 of the drive. The touch probe unit can be configured for the homing process using the parameter `Move To Sync Source`.

- `Move To Sync Distance`
  If the expected event is not detected in touch probe unit 1 within this distance (measured from the starting position of the homing process), the homing process is aborted. The sign of the value determines the direction of movement of the axis
- `Move To Sync Velocity`
  During the sequence, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.
  The direction of movement is determined by the `Move To Sync Distance`.
- `Move To Sync Source`
  During the homing procedure, this parameter affects the configuration of touch probe unit 1. The homing algorithm evaluates the touch probe 1 input of the axis or the subordinate drive object (PDO link).
  - `HardwareLatch1`
    Touch probe unit 1 of the drive is being evaluated.
  - `ZeroImpulseCoE`
    The encoder's zero pulse is evaluated.

- ◦ `HardwareLatch1CoEDriveDefined`
  The input on the drive to be evaluated is configured in the CoE object `0x60D0:01` .
- ◦ `SoftwareSync`
  Software-based singleturn zero crossing detection for homing synchronization.

Move to Initial Position Enabled

Name		Default value	Unit	Type
Move to Initial Position Enabled		FALSE		BOOL
	Move to Initial Position	10.0	[PosUnit]	LREAL
	Move to Initial Position Velocity	30.0	[PosUnit]/s	LREAL

Optionally, a homing process can be completed with a movement to an absolute target position if the parameter `Move To Initial Point Enabled` is set to `TRUE`.

- `Move To Initial Position`
  After successful homing, a motion command is initiated to the specified absolute target position. If the constant #Default is specified, the `Homing Position` is used as the position value.
- `Move To Initial Position Velocity`
  During the final motion command, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

The following parameters originate from the nested Touch Probe object [▶ 110].

Binary Touch Probe

Name	Default value	Unit	Type
Input Inverted	FALSE		BOOL

Invert Input

If the value of the `Input Inverted` parameter is set to `TRUE`, the MC's internal algorithm inverts the logic of the input Signal [▶ 102] for evaluation by the touch probe unit.

3.2.3.13.1.2 Online Parameter

The (online) parameters originate from the nested Touch Probe object [▶ 102].

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

Positive Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]

Name	Type
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

Negative Edge Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.13.1.3 Data Area

The data areas originate from the nested Touch Probe object [▶ 102].

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

Inputs

Enable (default)	Name	Type	Description
☑	Signal	BIT	Input for the trigger signal. For example, it can be linked to a variable from the PLC or to a digital input on an I/O terminal.
☐	WcState	BIT	Working counter

ToPlc

Enable (default)	Name	Type	Description
☑	Oid	OTCID	Object ID for linking to a touch probe object in the PLC

NOTICE

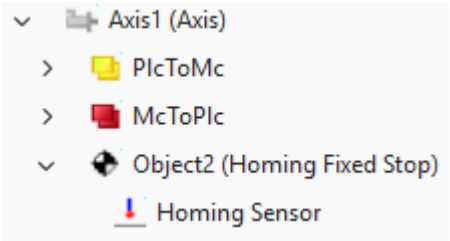
Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

3.2.3.13.2 Homing Fixed Stop

The Homing Fixed Stop feature allows for a homing process using a physical object that mechanically blocks the axis's movement. To this end, a Fixed Stop Detection Touch Probe object [▶ 104] is nested within the Homing Fixed Stop object. The Homing Fixed Stop object should be used, for example, to reference the axis using a fixed stop. Signal analysis takes place within the context of the MET task [▶ 19]. The homing procedure is configured in the Parameters (Init) [▶ 112] tab.



View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

3.2.3.13.2.1 Init Parameter

General

Homing Position

Name	Default value	Unit	Type
Homing Position	0.0	[PosUnit]	LREAL

The parameter `Homing Position` defines the absolute reference position. The axis is set to this position after a successful homing procedure. This parameter has an effect when the value `#Default` is used for the reference position input in the command to start a homing procedure (e.g., MC_Home).

Sequence

A homing process has several phases. The following parameters can be used to enable and configure individual phases.

Move to Fixed Stop Enabled

Name		Default value	Unit	Type
Move to Fixed Stop Enabled		FALSE		BOOL
	Move to Fixed Stop Distance	10.0	[PosUnit]	LREAL
	Move to Fixed Stop Velocity	30.0	[PosUnit]/s	LREAL
	Torque Limit	30.0	%	LREAL

Enabling the `Move To Fixed Stop Enabled` parameter (value: `TRUE`) adds a sequence in the form of an axis movement to the initialization process.

- `Move To Fixed Stop Distance`
  If the expected force/torque threshold is not reached within this distance (measured from the starting position of the homing process), the homing process is aborted. The sign of the value determines the direction of movement of the axis.
- `Move To Fixed Stop Velocity`
  During the sequence, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.
- `Torque Limit`
  Torque/force limitation during motion toward the hard stop.

Move from Fixed Stop Enabled

Name		Default value	Unit	Type
Move from Fixed Stop Enabled		FALSE		BOOL
	Move from Fixed Stop Velocity	30.0	[PosUnit]/s	LINT

Enabling the `Move From Fixed Stop Enabled` parameter (value: `TRUE`) adds a sequence in the form of an axis movement to the initialization process. The axis continues to move until the force/torque level falls below the expected threshold again.

- `Move From Fixed Stop Velocity`
  During the sequence, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.

Move to Sync Enabled

Name		Default value	Unit	Type
Move to Sync Enabled		FALSE		BOOL
	Move to Sync Distance	10.0	[PosUnit]	LREAL
	Move to Sync Velocity	30.0	[PosUnit]/s	LREAL
	Move to Sync Source	ZeroImpulseCoE		MC.EHomingSyncSource [▶ 200]
	Move to Sync Edge	Positive		MC.ETouchProbeTriggerSource [▶ 203]

Enabling the `Move To Sync Enabled` parameter (value: `TRUE`) adds a sequence in the form of an axis movement to the initialization process. The axis continues to move until a configured event is detected in touch probe unit 1 of the drive. The touch probe unit can be configured for the homing process using the parameter `Move To Sync Source`.

- `Move To Sync Distance`
  If the expected event is not detected in touch probe unit 1 within this distance (measured from the starting position of the homing process), the homing process is aborted. The sign of the value determines the direction of movement of the axis

- `Move To Sync Velocity`
  During the sequence, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.
  The direction of movement is determined by the `Move To Sync Distance`.
- `Move To Sync Source`
  During the homing procedure, this parameter affects the configuration of touch probe unit 1. The homing algorithm evaluates the touch probe 1 input of the axis or the subordinate drive object (PDO link).
  - `HardwareLatch1`
    Touch probe unit 1 of the drive is being evaluated.
  - `ZeroImpulseCoE`
    The encoder's zero pulse is evaluated.
  - `HardwareLatch1CoEDriveDefined`
    The input on the drive to be evaluated is configured in the CoE object `0x60D0:01`.
  - `SoftwareSync`
    Software-based singleturn zero crossing detection for homing synchronization.

Move to Initial Position Enabled

Name		Default value	Unit	Type
Move to Initial Position Enabled		FALSE		BOOL
	Move to Initial Position	10.0	[PosUnit]	LREAL
	Move to Initial Position Velocity	30.0	[PosUnit]/s	LREAL

Optionally, a homing process can be completed with a movement to an absolute target position if the parameter `Move To Initial Point Enabled` is set to `TRUE`.

- `Move To Initial Position`
  After successful homing, a motion command is initiated to the specified absolute target position. If the constant #Default is specified, the `Homing Position` is used as the position value.
- `Move To Initial Position Velocity`
  During the final motion command, the axis moves at no more than the velocity set in this parameter. The value must be set to >0.0.

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

The following parameters originate from the nested Touch Probe object [▶ 104].

Detection Thresholds

Name	Default value	Unit	Type
Torque/Force Threshold	10.0	%	LREAL
Velocity Threshold	10.0	[PosUnit]/s	LREAL
Lag Threshold	2.0	[PosUnit]	LREAL

Torque/Force Threshold

Threshold value for the actual motor torque value. If the torque exceeds this value during evaluation, this serves as an indicator that a hard stop has been detected. The value `#Ignore` disables this threshold.

Velocity Threshold

Threshold value for the actual axis velocity during evaluation. The power/torque threshold is evaluated only if the actual velocity is below this threshold value. The value `#Ignore` disables this threshold.

Lag Threshold

Threshold value for the position lag. If the actual position lag exceeds this value during evaluation, this serves as an indicator that a hard stop has been detected. The value `#Ignore` disables this threshold.

Disabling all thresholds

If all thresholds are disabled, the module immediately reports a detected event. In the case of homing, this is instantly recognized as the encoder having reached the end stop.

- Enter at least a real number for the torque/force threshold or position lag threshold to enable hard stop detection.

Behavior of an accelerating axis with a velocity threshold

If, for example, the axis accelerates from a standstill and the touch probe module's evaluation is already active, the axis's velocity may still be below the set threshold value. Depending on the other threshold values, this may cause the touch probe module to report a detected event. In the case of homing, this can be detected as reaching the end stop.

- If necessary, start an axis movement before activating the touch probe unit.

3.2.3.13.2.2 Online Parameter

The (online) parameters originate from the nested Touch Probe object [▶ 105].

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

Touch Probe Status

Name	Type
State	MC.ETouchProbeTriggerState [▶ 203]
Last Position	LREAL
Last DC timestamp	LINT

State

Current state of the touch probe unit.

Last Position

The position where the configured event was last detected. The value `#Invalid` indicates that no event has been recorded yet.

Last DC timestamp

DC Time in nanoseconds when the configured event was last recorded. The value `0` indicates that no event has been recorded yet.

3.2.3.13.2.3 Data Area

The data areas originate from the nested Touch probe object [▶ 105].

View of subordinate objects

Depending on the *Show Hidden Children* setting in the context menu of an overlying parent object, child objects are hidden. Parameters and data areas of the child object are visible in the parent object when this setting is disabled.

- To display child objects, **right-click** the **parent object** (context menu) > and click **Show Hidden Children.**

ToPlc

Enable (default)	Name	Type	Description
☑	Oid	OTCID	Object ID for linking to a touch probe object in the PLC

NOTICE

Warning in the error log window caused by selected entries that are not linked to an I/O device

Entries in the data areas of drive and encoder objects that are enabled (Enable = True) but are not linked to the I/O drive object result in a warning in the error log window. If the link for a function used at runtime is missing, this results in a runtime error. For example, an axis cannot be started if the status word from the I/O drive object is not linked.

- Link the required entries for the interface between the axis and the I/O device
- Deselect the entry that is not required to avoid generating a warning.

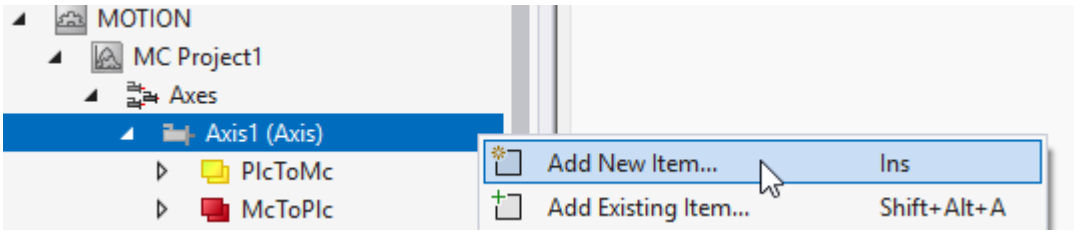
3.2.3.14 Expand the axis object with an additional module

The axis object can be expanded to include an additional module.

✓ A TwinCAT project with an MC project has been created;
see Creating an MC project [▶ 17].

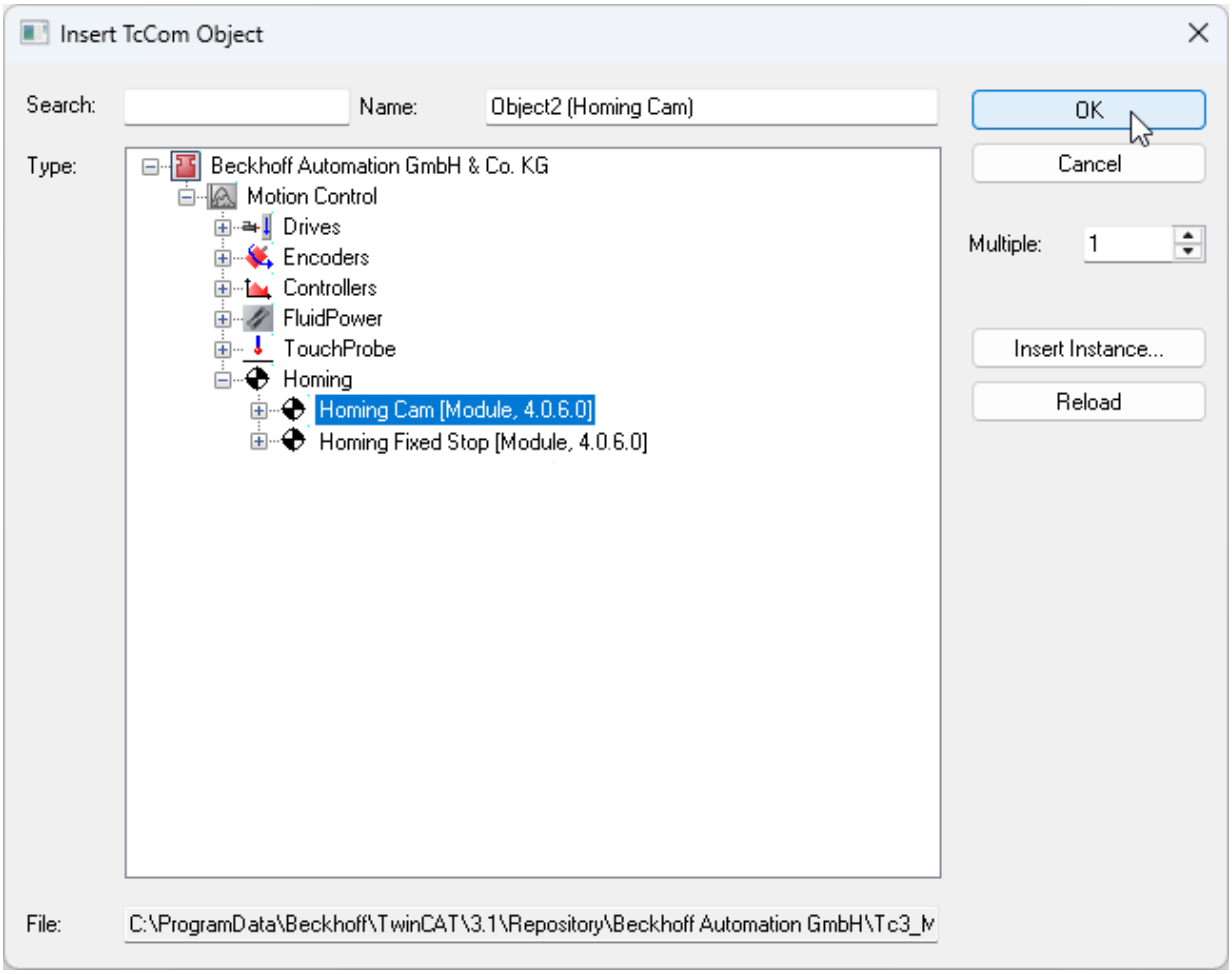
✓ An MC3 axis has been added to the TwinCAT project;
see Create an axis object [▶ 25]

1. Right-click **Axes > Axis > Add New Item...**

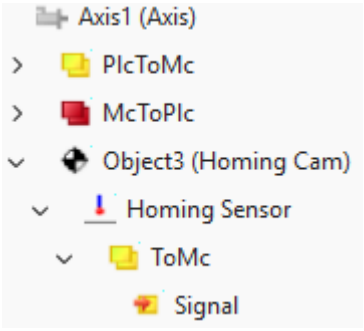


2. In the **Insert TcCOM Object** dialog box, select an object under **Motion Control** (e.g., **Homing > Homing Cam**).

3. Enter the desired number in the **Multiple** field and confirm both by clicking the **OK** button.



⇨ An additional module was successfully inserted below the axis object.



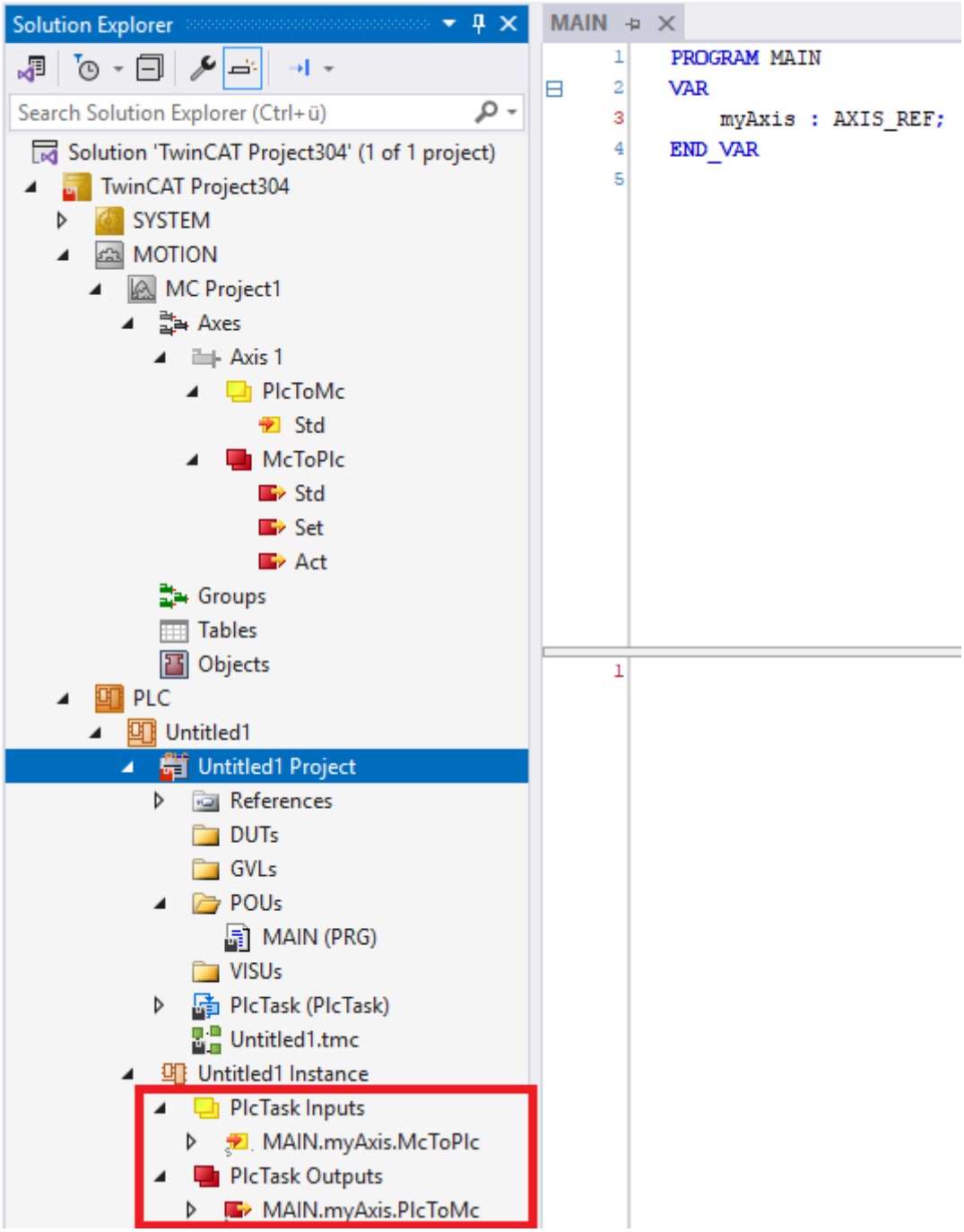
3.2.3.15 Link a PLC to an axis object

✓ An axis object has been created;
see Create an axis object [▶ 25].

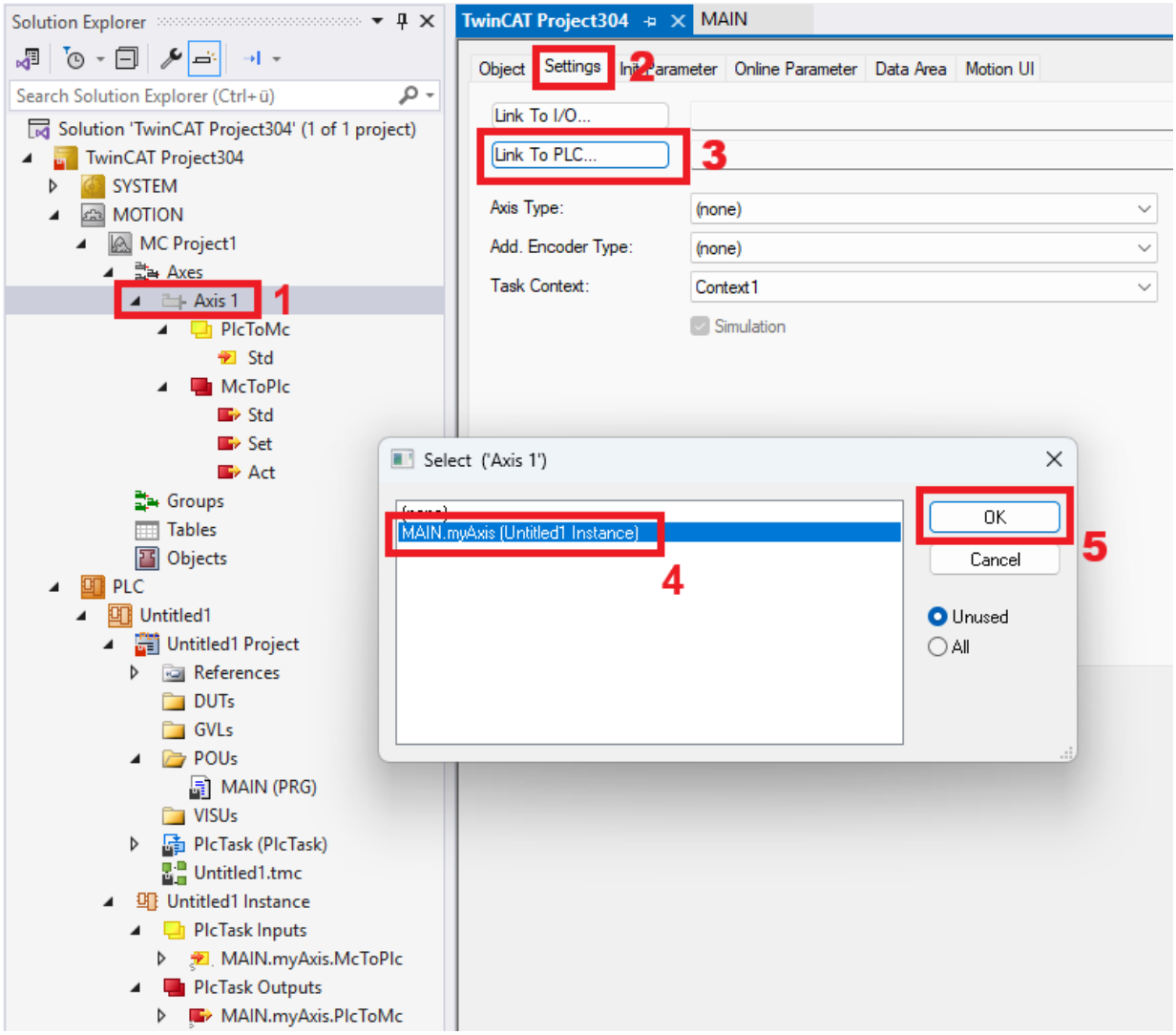
✓ A PLC project containing an instance of Tc3_Mc3Ptp.AXIS_REF has been created.

1. Build the PLC project.

⇨ The `axis` instance of the `Tc3_Mc3Ptp.AXIS_REF` should now be displayed under the PLC instances in the Solution Explorer tree.



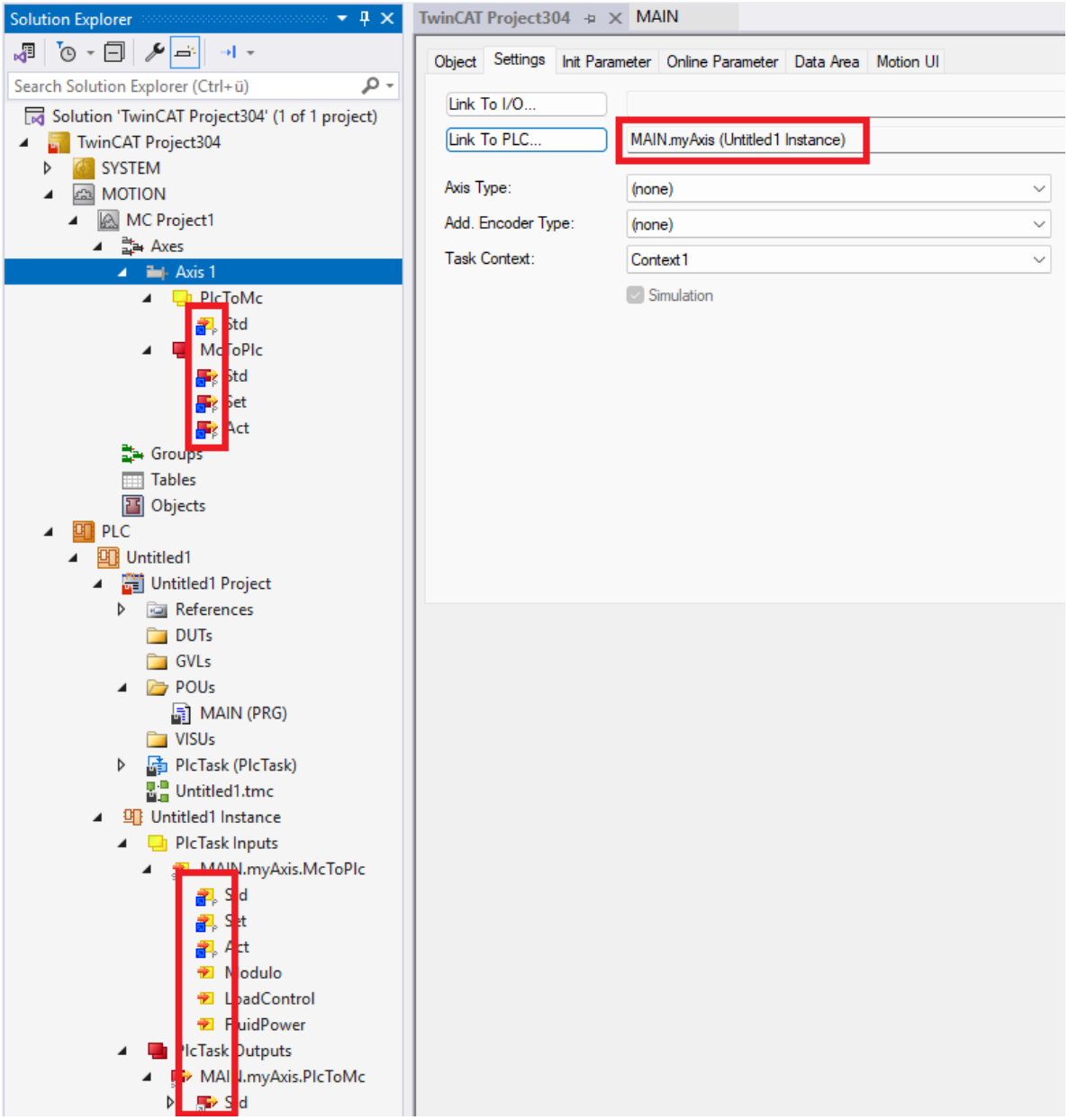
2. Link the PLC instance of the `Tc3_Mc3Ptp.AXIS_REF` to the MC3 axis instance (steps 1–5).



⇨ The axis is now linked to the PLC.

⇨ The text field next to the Link to PLC… button displays the linked instance of AXIS_REF in the PLC.

⇨ In the configuration tree in Solution Explorer, a linked variable from the cyclic interface is indicated by a blue or white icon.

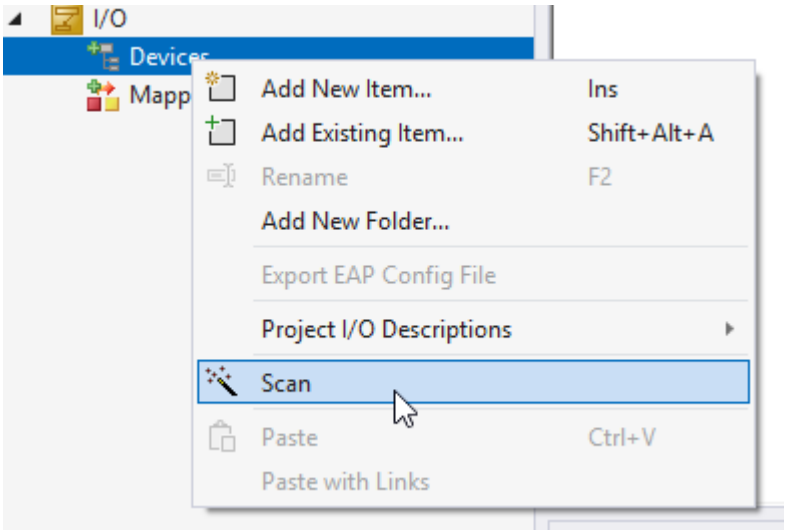


3. Activate the TwinCAT project.

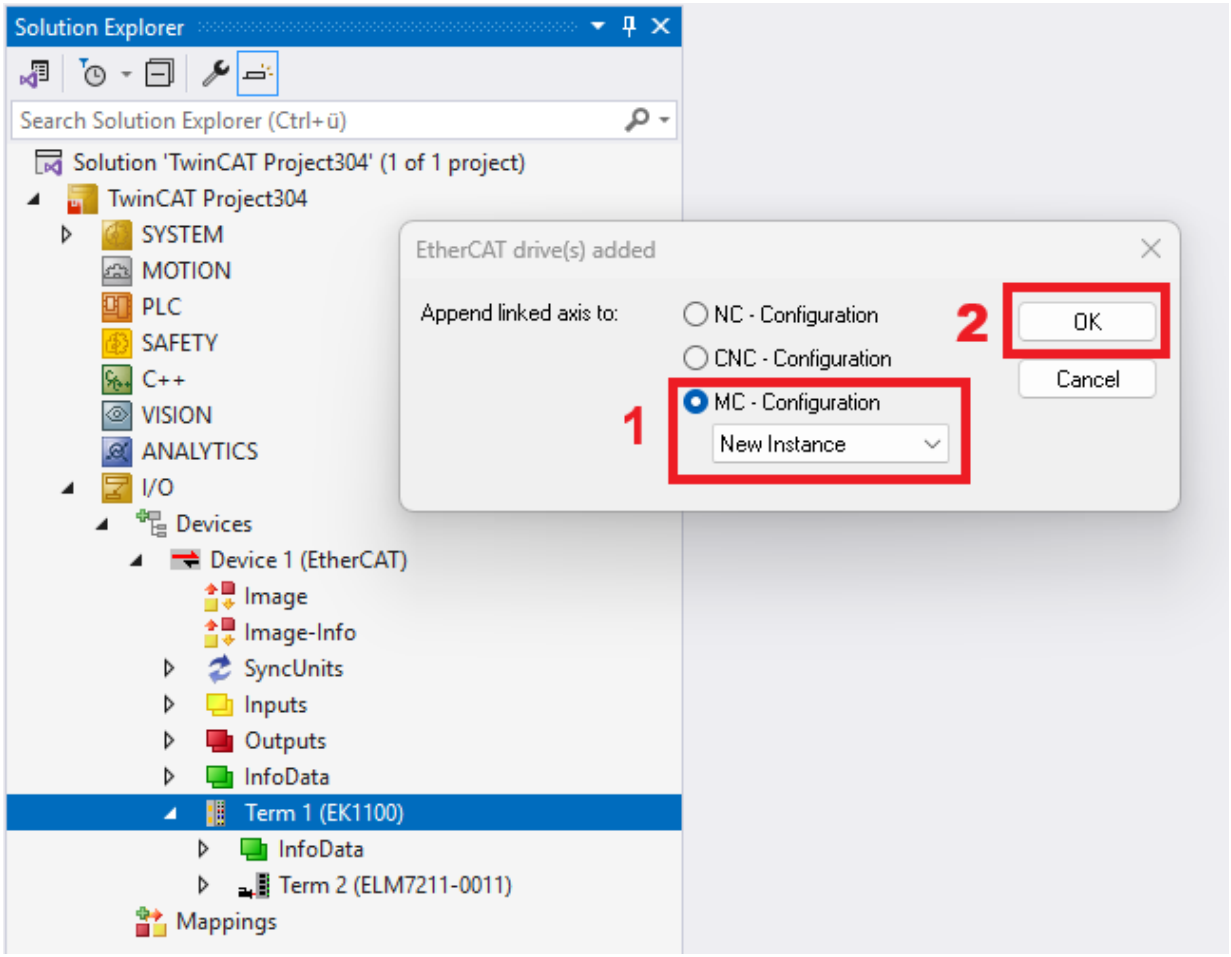
3.2.3.16 Link an axis object to hardware

Automatic creation and linking of an axis when scanning the EtherCAT configuration

1. Scan the EtherCAT fieldbus configuration: I/O Nodes > Devices > Context menu (right-click) > **Scan**



2. In the “EtherCAT drives added” dialog box that opens, select the **MC – Configuration** (1) option. If a motion configuration has already been created in the TwinCAT project, you can select the desired instance from the drop-down menu. Selecting **New Instance** creates a new motion configuration node. Confirm the dialog with **OK** (2).



⇨ In the TwinCAT project in Solution Explorer, a new motion configuration and an axis are created. It is automatically linked to the hardware.

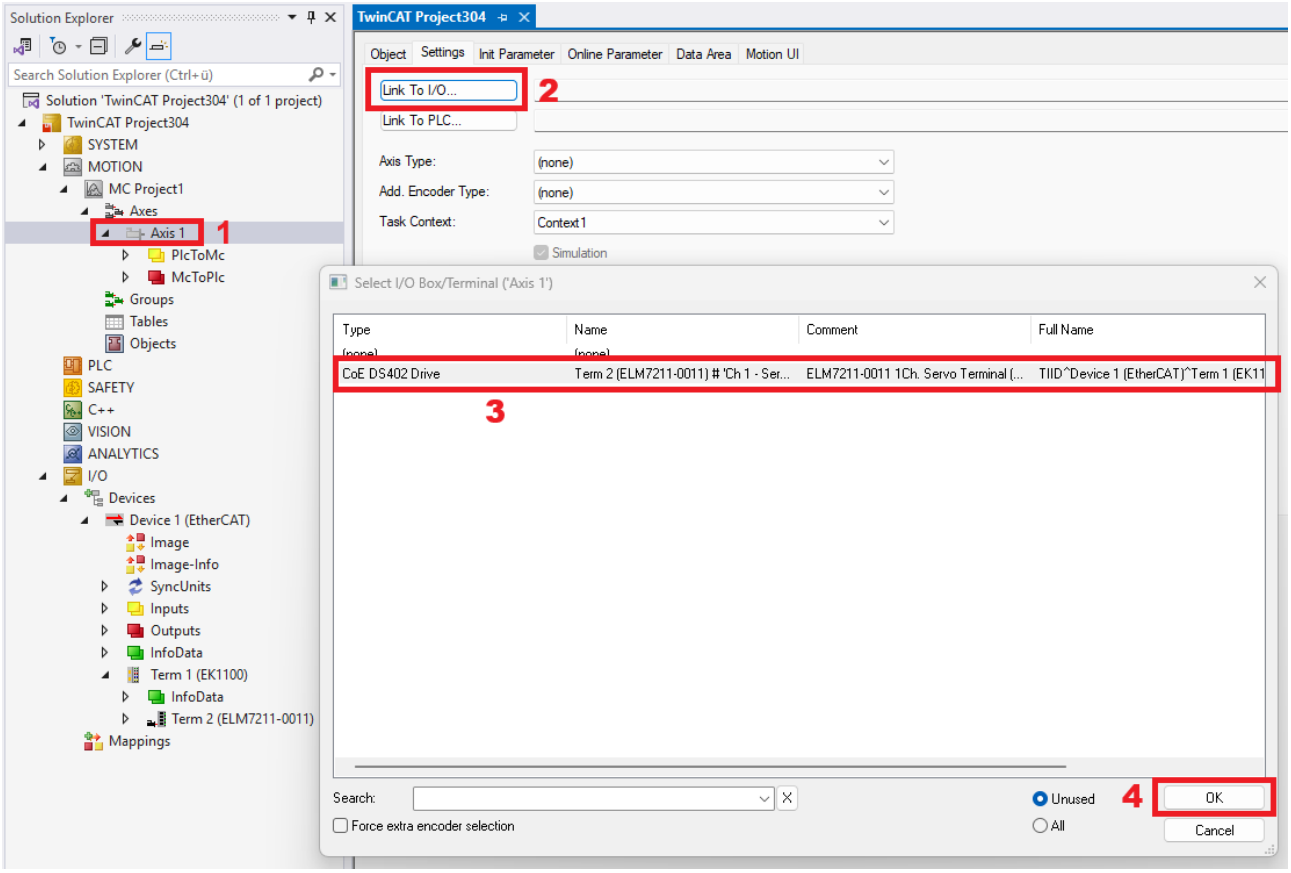
Link to I/O – Manually link an axis

Procedure – Example for EL72xx/ELM72xx, AX5000, AX8000

✓ The hardware configuration has been created.

✓ A motion configuration with one MC3 axis has been created.

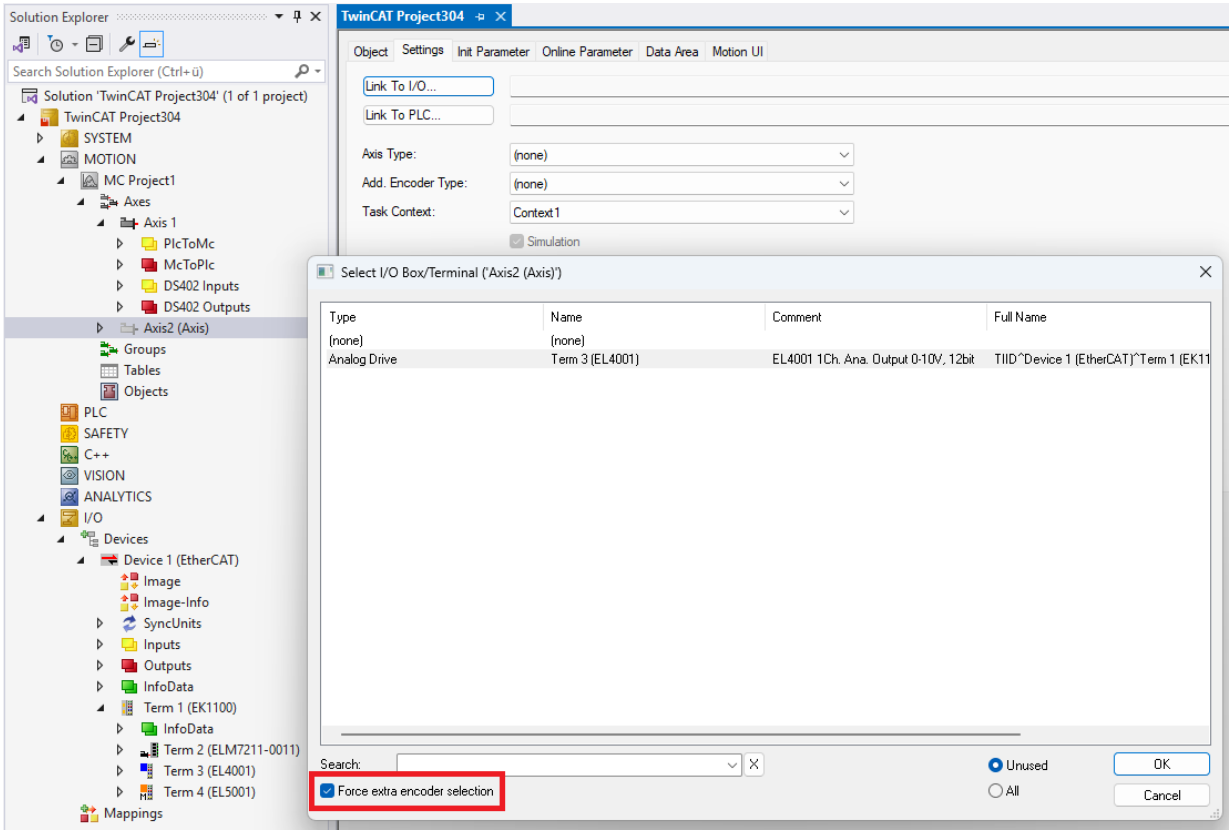
3. Select the axis you want to link (1) > **Settings** tab.
4. Click on **Link to I/O...**(2).
5. Select the drive hardware (3) you want to link.
6. Confirm your selection with **OK** (4).



Procedure – Example: Analog drive with an additional encoder

✓ In this case, the same steps as in the previous section apply initially. In the “Select I/O Box/Terminal” dialog box, note the following selection:

7. Select the **Force extra encoder selection** option before clicking **OK** to close the dialog box.



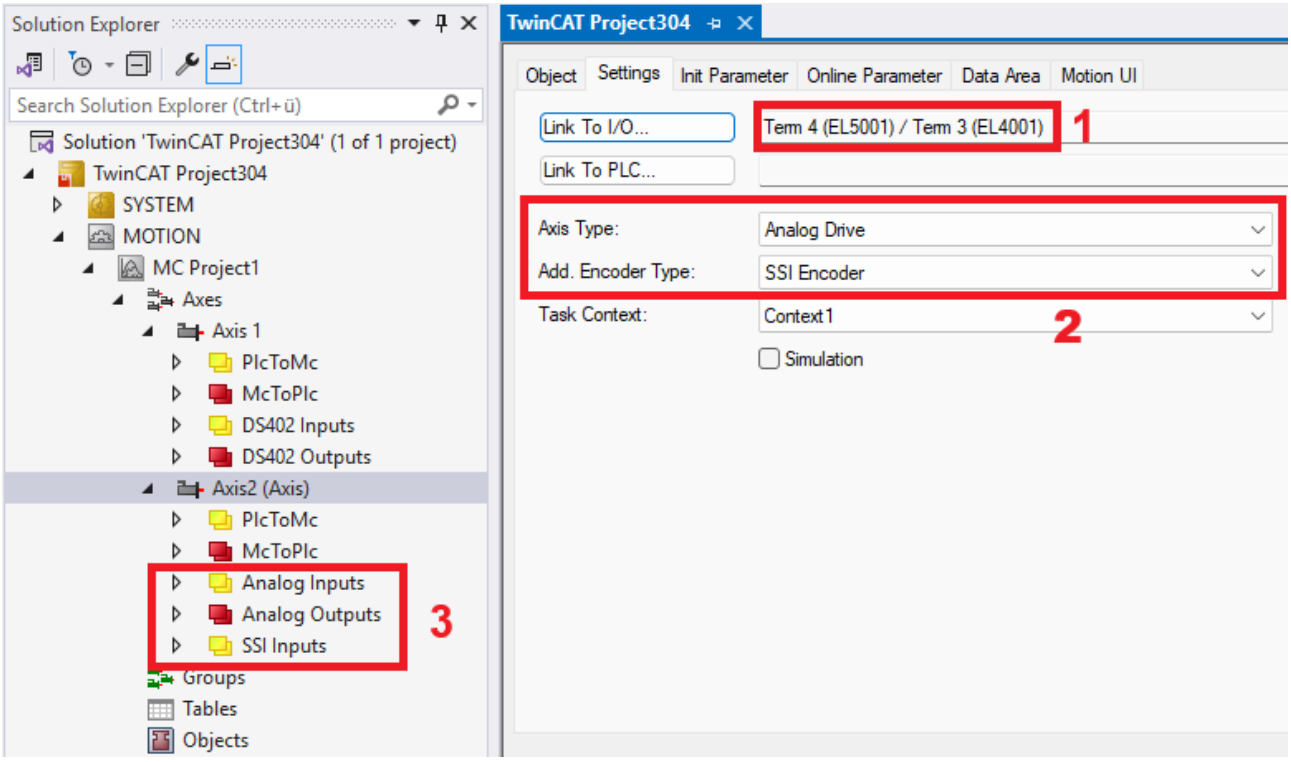
⇨ Another dialog opens.

⇨ Select an encoder and confirm your selection by clicking the **OK** button.

⇨ The text field next to the **Link to I/O…** button displays the linked hardware devices (1).

⇨ The type of axis and the additional encoder are displayed (2).

⇨ The cyclic interfaces between the MC3 axis and the I/O devices are visible (3).



3.2.3.17 Migrating from NC2 to MC3

This guide provides step-by-step instructions on how to migrate a TwinCAT NC2 axis (classic NC PTP) to an MC3 axis.

Requirements

- TwinCAT 3 Motion configuration with one or multiple NC2 axes
- Installation of the MC3 packages is complete (see the Installation [▶ 8] chapter)
- It is recommended that you create a backup copy of the original project.

Procedure

Creating MC3 projects and axes

1. Create a new MC3 configuration in the existing project folder (see the chapter Creating an MC Project [▶ 17]).
2. Add new axis objects to the MC project according to the number of axes to be migrated (see the chapter Creating an Axis Object [▶ 25]).
3. If necessary, assign an application-specific name.
4. Distinguish between the virtual axis and the physical axis with its link to the hardware:

✓ Virtual axis not linked to the I/O configuration:

5. Continue with the migration of the axis parameters.

✓ Axis linked to the I/O configuration:

6. Remove the link between the NC2 axis to be migrated and the I/O configuration.
7. Link the MC3 axis to the I/O device (see the chapter Link an axis object to hardware [▶ 121] Linking an axis object to hardware).

Advanced axis configuration (optional)

The modularity of the MC3 configuration allows for a clearer overview and better organization of the parameters. If a homing procedure or a position controller is required in the MC3 software for the axis to be migrated, additional objects must be inserted below the MC3 axis. The same applies when using the touch probe function.

References:

- Expand the axis object with an additional module [▶ 116]
- Referencing cams (homing modules) [▶ 106]
- Probe modules (touch probe modules) [▶ 100]
- Controller objects (controller modules) [▶ 93]

Transfer axis parameters

The parameters for the MC3 axis can be found in the **Init Parameter** tab.

NC2 Parameter	MC3 Parameter	To be observed
Simulation mode in the Solution Explorer context menu	Simulation mode in the Solution Explorer context menu and as the `Simulation Mode` axis parameter	The parameter can also be written to an MC3 axis during runtime. The prerequisite is that the axis is not enabled.
Unit (Settings tab)	Position Unit (Init Parameter tab)	In MC3, selecting a value from an enumeration type.
Maximum Dynamics		

NC2 Parameter	MC3 Parameter	To be observed
Reference Velocity Default: 2200.0 Maximum Velocity Default: 2000.0	Maximum Velocity Default: 2000.0	If the Reference Velocity is used in NC2, the minimum value between the Reference Velocity (NC2) and the Maximum Velocity (NC2) can be calculated for MC3, and this value can be entered as the Maximum Velocity (MC3).
Maximum Acceleration Default: 15000.0	Maximum Acceleration Default: 10000.0	Changing the default value. The value can be copied from an existing NC2 axis.
Maximum Deceleration Default: 15000.0	Maximum Deceleration Default: 10000.0	Changing the default value. The value can be copied from an existing NC2 axis.
Default Dynamics		
Default Acceleration Default: 1500.0	Default Acceleration Default: 2000.0	Changing the default value. The value can be copied from an existing NC2 axis.
Default Deceleration Default: 1500.0	Default Deceleration Default: 2000.0	Changing the default value. The value can be copied from an existing NC2 axis.
Default Jerk Default: 2250.0	Default Jerk Default: 4000.0	Changing the default value. The value can be copied from an existing NC2 axis.
Fast Axis Stop	-	
Fast Axis Stop Signal Type (optional)	-	
Fast Acceleration (optional)	-	
Fast Deceleration (optional)	-	
Fast Jerk (optional)	-	
Manual Motion and Homing		
Homing Velocity (towards plc cam)	Homing object [▶ 106]	Central parameterization in an optional Homing object [▶ 106] as a child object of the axis.
Homing Velocity (off plc cam)	Homing object [▶ 106]	Central parameterization in an optional Homing object [▶ 106] as a child object of the axis.
Manual Velocity (Slow) Default: 100.0	Jog Velocity Slow Default: 100.0	Changing the parameter name. The value can be copied from an existing NC2 axis.
Manual Velocity (Fast) Default: 600.0	Jog Velocity Fast Default: 250.0	Changing the default value and the parameter name. The value can be copied from an existing NC2 axis.
Jog Increment (Forward)	MC_Jog.Distance	Available via a function block in “Inching” mode
Jog Increment (Backward)	MC_Jog.Distance	Available via a function block in “Inching” mode
Limit Switches		
Soft Position Limit Minimum Monitoring	Position Limit Enforcement Enabled	A general parameter for enabling monitoring

NC2 Parameter		MC3 Parameter	To be observed
	Minimum Position	Minimum Position	The parameter appears when `Position Limit Enforcement Enabled = TRUE` is set. `#Ignore` can be used to disable monitoring of the software limit switches in one direction.
Soft Position Limit Maximum Monitoring		Position Limit Enforcement Enabled	A general parameter for enabling monitoring
	Maximum Position	Maximum Position	The parameter appears when `Position Limit Enforcement Enabled = TRUE` is set. `#Ignore` can be used to disable monitoring of the software limit switches in one direction.
Monitoring			
Position Lag Monitoring		Position Lag Monitoring Enabled	Changing the parameter name
	Maximum Position Lag Value	Maximum Position Lag Value	The parameter appears when `Position Lag Monitoring Enabled = TRUE` is set.
	Maximum Position Lag Filter Time	Maximum Position Lag Filter Time	The parameter appears when `Position Lag Monitoring Enabled = TRUE` is set.
Position Range Monitoring		-	
	Position Range Window	-	
Target Position Monitoring		MC_WaitForStableActualPosition	**NC2**: `MC_MoveAbsolute.Done = TRUE` when ActPos is within the window specified by the `TargetPositionMonitoring` and `TargetPositionWindow` parameters. **MC3**: The function block can be buffered after motion commands of the DiscreteMotion type (PLCopen state). `MC_MoveAbsolute.Done = TRUE` when the setpoint profile reaches the target and is therefore independent of target position monitoring. A clear separation of functionality is established.
	Target Position Window		
	Target Position Monitoring Time		
In-Target Alarm			
	In-Target Timeout		
Motion Monitoring		-	
	Motion Monitoring Window	-	
	Motion Monitoring Time	-	
Setpoint Generator		-	
Setpoint Generator Type		-	
Velocity Override Type		-	
NCI Parameter		-	
Rapid Traverse Velocity (G0)		-	

NC2 Parameter		MC3 Parameter	To be observed
Velo Jump Factor		-	
Tolerance ball auxiliary axis		-	
Max. position deviation, aux. axis		-	
Other Settings			
Position Correction		-	
	Filter Time Position Correction (P-T1)	-	
Backlash		-	

Error Propagation Mode		Example:
		`MC_CamIn.ReactionToMasterError`
		`MC_CamIn.ReactionToSlaveError`
			Coupling function blocks such as MC_GearIn or MC_CamIn have additional inputs. Couplings are command-based. A separate error propagation setting can be configured for each coupling/ command.

	Error Propagation Delay	-	Previous error responses and propagation can be implemented using the settings of coupling function blocks and axis parameters. Example: An error has occurred on the slave axis. The error must not be propagated to the master axis (`MC_CamIn.ReactionToSlaveError := NoReaction`), and the slave axis must be decoupled from the master axis. In the MC3, automatic decoupling occurs in this case, and the slave axis initiates its error response. If it is still possible to control the axis, a deceleration ramp following an MC_Reset can be replaced by a user-defined command with the same dynamic limits (this can be done within a PLC cycle).

Couple slave to actual values if not enabled		Example:
		`MC_CamIn.ReactionToMasterError`
		`MC_CamIn.ReactionToSlaveError`
			Coupling function blocks such as MC_GearIn or MC_CamIn have additional inputs. Couplings are command-based. A separate error propagation setting can be configured for each coupling/ command.

	Velocity Window	-	
	Filter Time for Velocity Window	-	
Allow motion commands to slave axis (Default `TRUE`)		No separate parameter - MC3 axes allow motion commands to be sent to the slave axis (default `TRUE`)	Slave axes can be decoupled from their master axis using a motion command (`BufferMode := Aborting`)
Allow motion commands to external setpoint axis		No separate parameter - MC3 axes allow motion commands on axes with external setpoint generation enabled (default `TRUE`)	

NC2 Parameter	MC3 Parameter	To be observed
Dead Time Compensation (Delay Velo and Position)	Dead Time Compensation from Hardware	As a rule, dead times are automatically determined during EtherCAT fieldbus initialization and compensated for internally in MC3. If a manual correction is nevertheless necessary, it can be entered in the `Additional Time Shift From Hardware` and `Additional Time Shift To Hardware` parameters.
	Additional Time Shift from Hardware	
	Dead Time Compensation to Hardware	
	Additional Time Shift to Hardware	
Data Persistence	-	

Transfer drive parameters

The parameters for the MC3 axis can be found in the **Init Parameter** tab. It is important to note whether the *Show Hidden Children* option is enabled in the axis's context menu (disabled by default; Axis > right-click > Show Hidden Children). If the function is enabled, the following parameters are displayed in the child object of the axis, which then becomes visible. Otherwise, all parameters can be accessed centrally via the axis object.

NC2 Parameter	MC3 Parameter	To be observed
Output Settings		
Invert Motor Polarity	Drive Type / Scaling > Direction Inverted	In the MC3, the inversion of the motor and encoder orientation is a parameter of the drive object. From the MC's perspective, there is often only one hardware object (e.g., AX8000). In this case, using two parameters with different settings is not advisable, as it will result in a position lag and the control loop will not function as expected. An additional encoder TcCOM object as a child object of the axis also has its own parameter.
Reference Velocity	-	See the description of the axis parameters
Position and Velocity Scaling		
Output Scaling Factor (Position)	Drive Type / Scaling > Position Scaling Numerator/Denominator from Hardware	Not directly applicable. See the MC3 parameters for scaling the target and actual positions. If the MC considers it to be a hardware object, the parameters can be found under the Drive object. Otherwise, the drive and encoder have their own scaling parameters.
Output Scaling Factor (Velocity)	Drive Type / Scaling > Velocity Scaling Numerator to Hardware	In this procedure, the Velocity Scaling Denominator must be set to 1. Used to scale the target and actual velocities for the drive and from the encoder, respectively.
Minimum Drive Output Limitation [-1.0 ... 1.0]	-	
Maximum Drive Output Limitation [-1.0 ... 1.0]	-	
Output Delay	-	

NC2 Parameter	MC3 Parameter	To be observed
Torque and Acceleration Scaling		
Input Scaling Factor (Actual Torque)	Drive Type / Scaling > Torque Scaling from Hardware	The value can be copied directly.
Input P-T1 Filter Time (Actual Torque)	Drive Type / Actual Torque Filter / Settings > Filter Time Torque	A standalone TcCOM object as a child object under the Drive object. Enable *Show Hidden Children* to add a filter object.
Input P-T1 Filter (Actual Torque Derivative)	Drive Type / Actual Torque Filter / Settings > Filter Time Torque Derivative	A standalone TcCOM object as a child object under the Drive object. Enable *Show Hidden Children* to add a filter object.
Output Scaling Factor (Torque Setpoint)	-	
Output Scaling Factor (Torque Offset)	-	
Output Delay (Torque Offset)	-	
Output Scaling Factor (Acceleration)	-	
Output Delay (Acceleration)	-	
Optional Position Command Output Smoothing Filter		
Smoothing Filter Type	-	
Smoothing Filter Time	-	
Smoothing Filter Order (P-Tn only)	-	
Other Settings		
Drive Mode	-	
Drift Compensation (DAC-Offset)	-	
Following Error Calculation	Axis > Monitoring > Position Lag Calculation	Internal – InternallyCalculated External - HardwareProvided
Error Tolerance	-	

Transfer encoder parameters

The parameters for the MC3 axis can be found in the **Init Parameter** tab. It is important to note whether the *Show Hidden Children* option is enabled in the axis's context menu (disabled by default; Axis > right-click > Show Hidden Children). If the function is enabled, the following parameters are displayed in the child object of the axis, which then becomes visible. Otherwise, all parameters can be accessed centrally via the axis object.

From the MC's perspective, there is often only one hardware object (e.g., AX8000). In this case, for example, the MC3 provides only one parameter for reversing the direction of rotation. An additional encoder TcCOM object as a child object of the axis also has its own parameter.

If two options are listed in the “MC3 Parameter” column in the table below, the parameter in question is available for a drive object and, optionally, for an additional encoder.

NC2 Parameter	MC3 Parameter	To be observed
Encoder Evaluation		
Invert Encoder Counting Direction	Axis > Drive Type / Scaling > Direction Inverted or Axis > Encoder Type / Scaling > Direction Inverted	In the MC3, the inversion of the motor and encoder orientation is *a* parameter of the drive object. In the case of a hardware object, having two parameters with

NC2 Parameter	MC3 Parameter	To be observed
		different settings does not make sense, as this results in a position lag and the control loop does not function as expected.
Scaling Factor Numerator	Axis > Drive Type / Scaling > Position Scaling Numerator from Hardware or Axis > Encoder Type / Scaling > Position Scaling Numerator from Hardware	
Scaling Factor Denominator	Axis > Drive Type / Scaling > Position Scaling Denominator from Hardware or Axis > Encoder Type / Scaling > Position Scaling Denominator from Hardware	
Position Bias	Axis > Drive Type / Scaling > Encoder Mounting Offset or Axis > Encoder Type / Scaling > Encoder Mounting Offset	
Modulo Factor	Axis > General > Position Modulus	The axis parameter is visible as soon as the parameter is set to `Is Modulo Axis = TRUE` .
Tolerance Window for Modulo Start	Axis > General > Position Modulo Tolerance Window	
Encoder Mask (maximum encoder value)	Axis > Drive Type / General > Maximum Counter Value + Minimum Counter Value or Axis > Encoder Type / General > Maximum Counter Value + Minimum Counter Value	The two MC3 parameters are visible when the `Position Data Range` parameter is set to `UserDefined`. A manual check of the value is necessary. In MC3, the value range is signed. `FullRange`: Refers to the maximum range of values that can be represented by the data type, e.g., 0 and 2^32 for UDINT, and -2147483648 to 2147483647 for DINT. `UserDefined`: Manually enter the value range for the value from the PDO link.
Encoder Sub Mask	Axis > Drive Type / General > Maximum Absolute Counter Value or Axis > Encoder Type / General > Maximum Absolute Counter Value	The two MC3 parameters are visible when the `Position Interpretation` parameter is set to `AbsoluteUserDataRange`. Number of unique increments per revolution (2^20-1 for a single-turn absolute encoder).
Reference System	Axis > Drive Type / General > Position Interpretation or Axis > Encoder Type / General > Position Interpretation	• `INCREMENTAL` corresponds to `Incremental` • `INCREMENTAL` (single-turn absolute) corresponds to `Incremental` • `ABSOLUTE` corresponds to `AbsoluteFullRange` • `ABSOLUTE MULTITURN RANGE (with single overflow)` corresponds to `AbsoluteFullRange`

NC2 Parameter	MC3 Parameter	To be observed
		• `ABSOLUTE SINGLETURN RANGE (with single overflow)` corresponds to `AbsoluteFullRange` • `ABSOLUTE (modulo)` corresponds to `AbsoluteModulo`
Limit Switches		
Soft Position Limit Minimum Monitoring	Position Limit Enforcement Enabled	A general parameter for enabling monitoring
Minimum Position	Minimum Position	The parameter appears when `Position Limit Enforcement Enabled = TRUE` is set. `#Ignore` can be used to disable monitoring of the software limit switches in one direction.
Soft Position Limit Maximum Monitoring	Position Limit Enforcement Enabled	A general parameter for enabling monitoring
Maximum Position	Maximum Position	The parameter appears when `Position Limit Enforcement Enabled = TRUE` is set. `#Ignore` can be used to disable monitoring of the software limit switches in one direction.
Filter		
Filter Time for Actual Position (P-T1)	Axis > Drive Type / Actual Position Filter / Settings > Filter Type = PT1 > Filter Time Position or Axis > Encoder Type / Actual Position Filter / Settings > Filter Type = PT1 > Filter Time Position	An additional TcCOM filter object is a child object of each drive and/or encoder object by default. The `Filter Typ` should be set to PT1. The additional parameters for setting the filter times will then become visible.
Filter Time for Actual Velocity (P-T1)	Axis > Drive Type / Actual Position Filter / Settings > Filter Type = PT1 > Filter Time Velocity or Axis > Encoder Type / Actual Position Filter / Settings > Filter Type = PT1 > Filter Time Velocity	
Filter Time for Actual Acceleration (P-T1)	Axis > Drive Type / Actual Position Filter / Settings > Filter Type = PT1 > Filter Time Acceleration or Axis > Encoder Type / Actual Position Filter / Settings > Filter Type = PT1 > Filter Time Acceleration	
Homing		
Invert Direction for Homing Sensor Search	Homing object [▶ 106]	Central parameterization in an optional Homing object [▶ 106] as a child object of the axis.
Invert Direction for Sync Impulse Search		

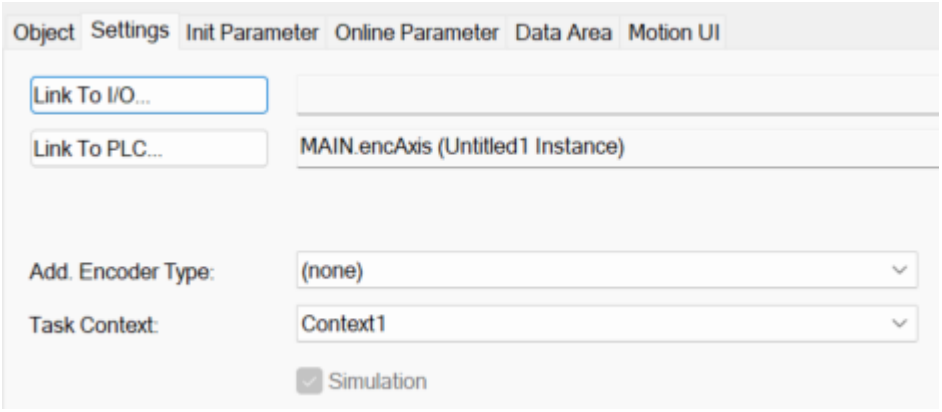
NC2 Parameter	MC3 Parameter	To be observed
Home Position (Calibration Value)		
Reference Mode (Sync condition)		
Homing Sensor Source		

3.2.4 Encoder Axis

An encoder axis in TwinCAT means that a pure encoder system is to be integrated into the system as an axis object. The settings and parameters for the axis encoder are described in the following chapter.

3.2.4.1 Settings tab

In the **Settings** tab, you can configure key settings for the encoder axis.



Property	Description		
Link to I/O…	The Link button opens a dialog box for linking the MC encoder axis to the drive hardware under the I/O node. The linked item is displayed in the field to the right of it.		
Link to PLC…	The Link button opens a dialog for linking the MC encoder axis to a PLC instance of ENCODER_AXIS_REF. The linked item is displayed in the field to the right of it.		
Add. Encoder Type	Select the Encoder Type to be used here. Details about the encoders can be found at Encoder modules [▶ 80]. The value is usually determined automatically when linking to a hardware element (Link to I/O).		
Task Context	Each encoder axis must be assigned to a task context. If a MC Project [▶ 17] has more than one context, you can use the combo box to change the axis's context. New contexts are created and configured using the MC project's Context tab [▶ 19].		
Simulation	☑		When simulation mode is active, the link to the I/O is ignored. Instead of the configured Encoder Type, you must select the data source for the simulation under **Init Parameter > General > Simulation Data Source**.
			If Encoder Type (none) is selected, the axis is always a simulation encoder axis, and the setting cannot be changed.
	☐		The configured Encoder Type and link to the I/O are used.

3.2.4.2 Init Parameter tab

General

Name	Default value	Unit	Type
Simulation Mode	TRUE		BOOL
Simulation Data Source	00000000		OTCID_EncoderAxisSimulationDataSource
Position Unit	mm	[PosUnit]	MC.EPositionUnit
Is Modulo Axis	FALSE		BOOL
Position Modulus	360.0	[PosUnit]	LREAL

Simulation Mode

The Simulation Mode indicates whether the I/O connection to the drive and encoder should be active (`FALSE`) or simulated (`TRUE`). The parameter can be written at runtime if the axis is not enabled by the controller.

Simulation Data Source

Simulation Data Source specifies the object ID of the axis used for the simulation. If Simulation Mode is active and the Simulation Data Source is set, the encoder axis reads the setpoints from the simulation source as its own actual values.

Position Unit

The Position Unit field is used for display purposes, such as on the commissioning interface. Inputs and outputs on function blocks are not converted.

Is Modulo Axis

Enable this parameter to use modulo functionality in function blocks for this axis. The setpoint and actual values are calculated as absolute and modulo values in the cyclic interface. To exchange Modulo information - for example, with the PLC - you must enable the Modulo data area [▶ 32] .

- **Position Modulus**
  Modulo value used to calculate the modulo position

Diagnostics

Name	Default value	Unit	Type
TraceLevel	tlWarning		TcTraceLevel

Specify which messages should be sent to the TwinCAT error list.

3.2.4.3 Online Parameter tab

An encoder axis has the following online parameters:

Additional Error Event Information

Name	Unit	Type
Originating Error Event Sender ID		OTCID
Original Error Id		HRESULT

Originating Error Event Sender ID

If a subsequent error occurs - for example, due to coupling - this parameter displays the object ID of the axis from which the error originated.

Original Error Id

If a subsequent error occurs - for example, due to a coupling - this online parameter displays the original error number.

3.2.4.4 Data Area tab

In general, TcCOM objects can contain data areas that can be used by the environment (e.g., by a PLC instance or by the TwinCAT I/O area). For TwinCAT MC3 TcCOM objects, you have a choice of data structures to use via the Data Area tab.

NOTICE

Unlinked CDTs can cause a floating-point exception when accessed

Comparisons with NaN values can lead to an exception, which results in the runtime stopping and potentially causing machine damage.
Valid data can only be expected if `IsConnected = TRUE`. Otherwise, there is a risk of an inconsistent or invalid data set, which may lead to unpredictable consequences and unexpected behavior. Arithmetic operations using invalid values can cause floating-point exceptions in the processor's floating-point unit.

- Switch off the Floating Point Exceptions if NaN values are to be used and processed in the application.
- Before accessing a CDT (e.g., from the PLC), check the CDT's `IsConnected`.
- Familiarize yourself with how to handle named constants and NaN values.

McToPlc (Output)

Enable (default)	Name	Type	Description
☑	Std	POINTER TO MC.CDT_MCTOPLC_ENCODER_AXIS_STD	Standard cyclic data structure containing the Oid of the encoder axis and key status information.
☑	Act	POINTER TO MC.CDT_MCTOPLC_ENCODER_AXIS_ACT	Optional cyclic data structure containing information about the current actual values of the encoder axis. **Notice Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)
☐	Modulo	POINTER TO MC.CDT_MCTOPLC_ENCODER_AXIS_MODULO	Optional cyclic data structure containing information about the module position of the encoder axis. To calculate the module position correctly, set the **Is Modulo Axis** parameter to TRUE in the Init Parameter [▶ 28] tab. **Notice Valid data can only be expected if IsConnected = TRUE. Valid data must be validated in accordance with the following chapter:** Constants (Named Constants)

3.2.4.5 Use the encoder axis

✓ Adding a motion object to an existing project folder is explained in the section Create an axis object [▶ 25].

1. Select the **Encoder Axis** module to add at least one encoder axis to the motion configuration.
2. Alternatively, you can automatically add and link an encoder axis when scanning the I/O configuration using the project tree in Solution Explorer.

Option 1: Operation in simulation mode

If the encoder axis is to be operated exclusively in simulation mode - or even during operation - the following two parameters must be set:

- Init Parameter > Simulation Mode (set by default by the development environment; if no hardware is connected, it can be written at runtime),
- Init Parameter > Simulation Data Source.

3. In the **Simulation Data Source** parameter, select the object ID of the axis whose setpoint values are to be output as actual values for the encoder axis.
4. If necessary, create an additional simulation axis.

Option 2: Operation with hardware

If the encoder axis is to be used with hardware, it must be linked to the appropriate device in the I/O configuration. If the encoder axis has not yet been automatically linked in the tree view in Solution Explorer when scanning the I/O configuration, you can do so by following the steps in the section Link an axis object to hardware [▶ 121].

Add the encoder axis to the PLC program and link it

✓ A PLC project has been created, and the Tc3_Mc3Ptp library has been added as a reference.

5. Insert an instance of the type `ENCODER_AXIS_REF` into your PLC program.

Example:

```
PROGRAM MAIN
VAR
    myEncoder : Tc3_Mc3Ptp.ENCODER_AXIS_REF;
END_VAR
```

6. Follow a similar procedure to that described in the Link a PLC to an axis object [▶ 117] section to establish a link between the PLC and the motion object.

   ⇨ An encoder axis has been added to the motion node in the tree view and linked to the PLC.

Create a PLC program with an encoder axis

When using MC3 in the PLC, note that a distinction is made between a classic axis of the type `AXIS_REF` and an encoder axis of the type `ENCODER_AXIS_REF`. This makes it possible to determine in XAE, at compile time, whether a programmed function is supported (see also Overview: MC3 types [▶ 214]). For example, motion commands (MC_MoveAbsolute, MC_Halt, …) and certain administrative commands (MC_Power) cannot be used with an encoder axis.

Use of unsupported function blocks

Programming unsupported function blocks results in error messages during compilation.

3.2.5 FluidPower Axis

A hydraulic axis can be designed as either a valve-controlled or a pump-controlled drive. To solve this, a special configuration of the standard axis [▶ 25] is used in the MC3. The hydraulic axis can operate with several types of position measuring systems and actuators.

Adding a hydraulic axis works on the same principle as a standard electric axis:

- Adding an axis [▶ 136]

In general, the FluidPower axis is capable of controlling both the position and the force or pressure of the hydraulic axis. The block diagrams and a supplementary description for each control system can be found here:

- Position controller [▶ 146]
- Pressure/force controller [▶ 147]

Force and pressure control requires the integration of pressure and force sensors. Details can be found in the following section:

- Pressure sensor A/B [▶ 147]
- Virtual force sensor [▶ 150]

What makes the solution integrated into MC3 unique is the ability to implement and automatically record the system characteristic curve. This is connected downstream of the position or force/pressure controller, thereby eliminating the system's nonlinear properties. Details on the system characteristic curve can be found in the following section:

- Valve characteristics [▶ 151]

The FluidPower axis also has several properties and parameters that are unique to this model and not found on the standard axis:

- Axis parameters [▶ 141]

The following “How-to” guides describe the process of commissioning a:

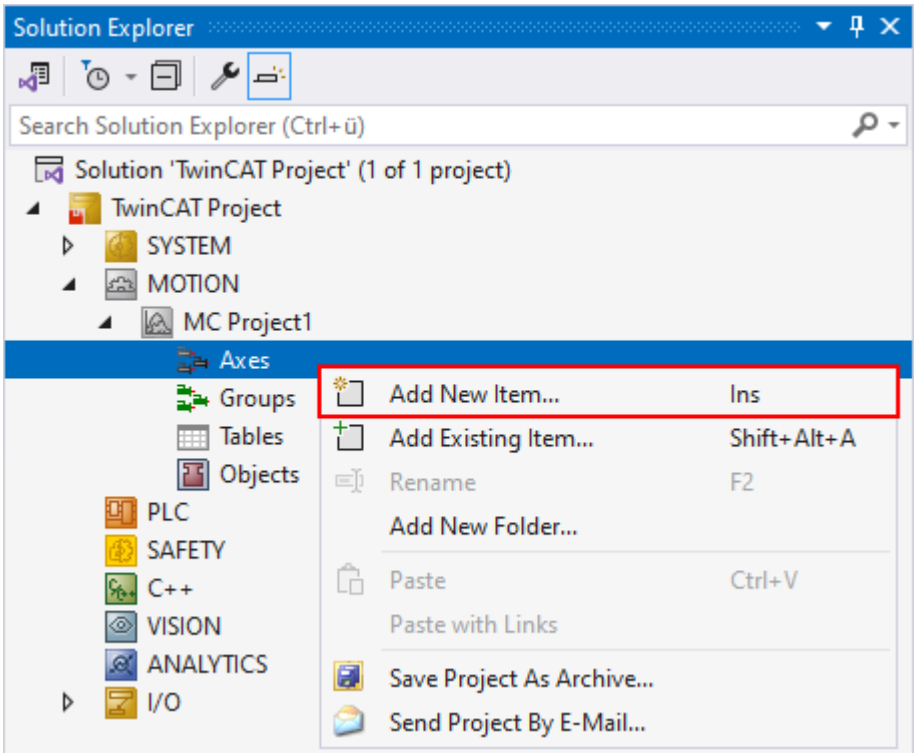
- Valve-controlled drive,
  see: Commissioning the FluidPower axis (valve-controlled axis) [▶ 155]
- Pump-controlled drive,
  see:

3.2.5.1 Adding an axis

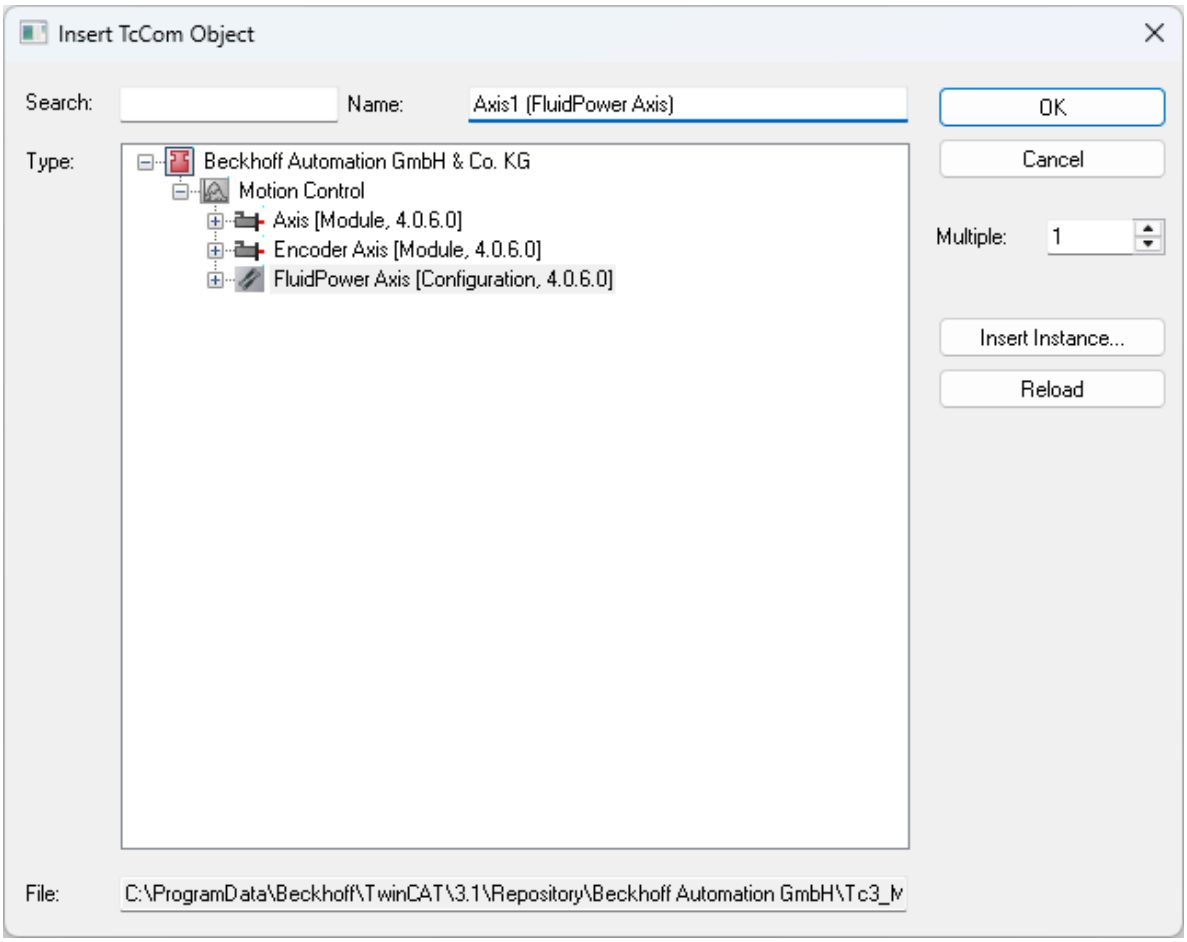
3.2.5.1.1 Create axis object/add axis

✓ A TwinCAT project with an MC project has been created;
see Creating an MC project [▶ 17].

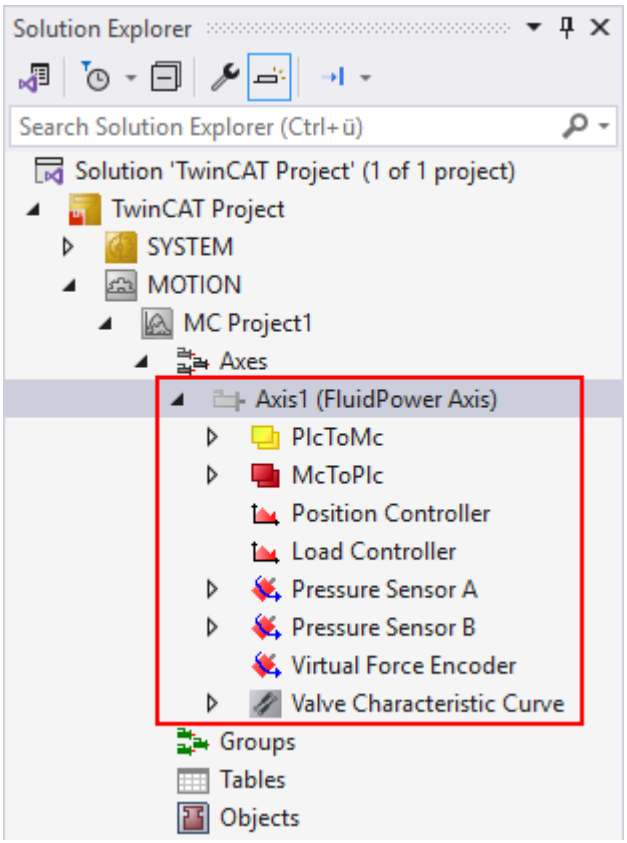
1. Right-click **Axes > Add New Item...**



⇨ In the **Insert TcCOM Object** dialog box, select the **FluidPower Axis** axis type under **Type** and click **OK**.



The FluidPower axis has been created with the required modules and is located below the Axes node.



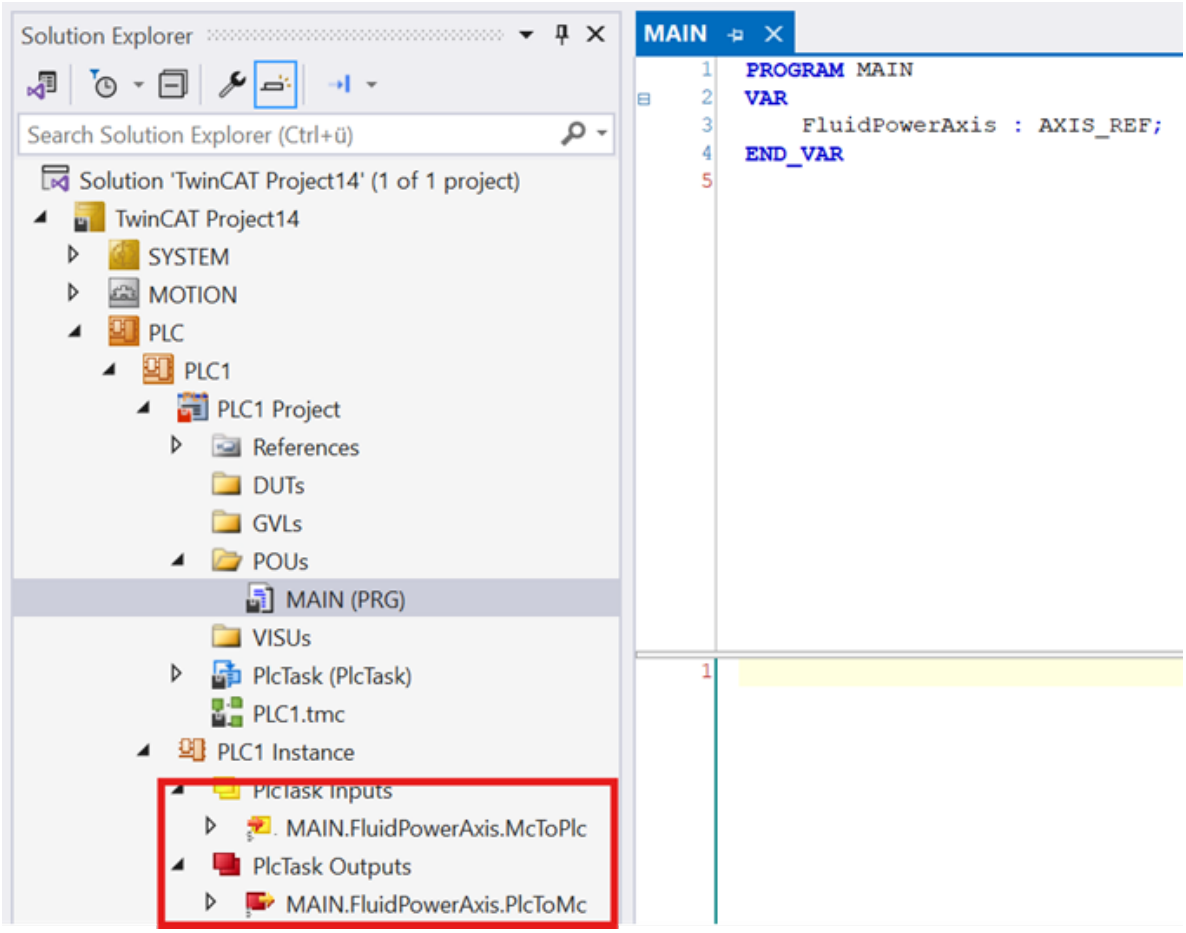
3.2.5.1.2 Link the axis object to the PLC

✓ An axis object has been created;
see Create axis object/add axis [▶ 136].

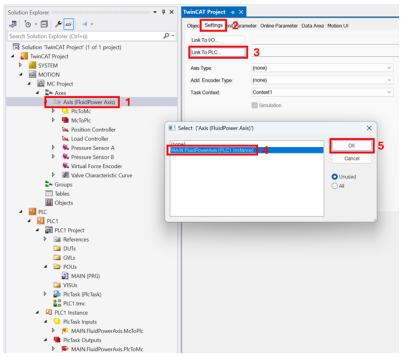
✓ A PLC project containing an instance of Tc3_Mc3Ptp.AXIS_REF was created.

1. Compile the PLC project.

⇨ The `FluidPowerAxis` instance of the `Tc3_Mc3Ptp.AXIS_REF` is now displayed under the PLC instances in the Solution Explorer.



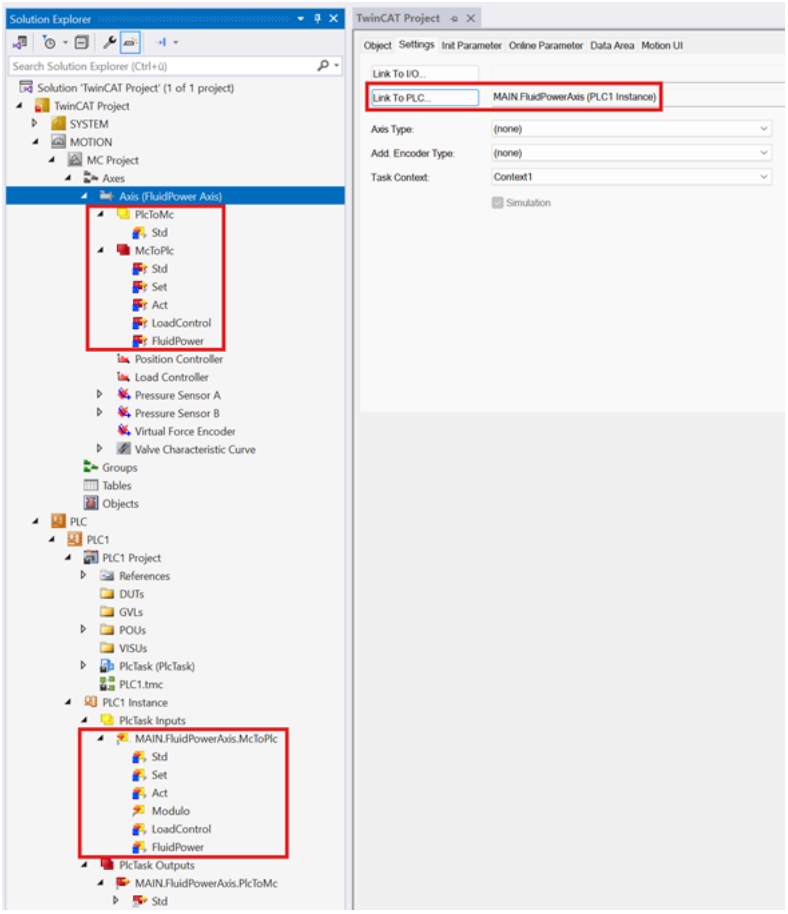
2. Link the PLC instance of the `Tc3_Mc3Ptp.AXIS_REF` to the MC3 axis instance (steps 1–5).



⇨ The axis is now linked to the PLC.

⇨ The text field next to the Link to PLC… button displays the linked instance of AXIS_REF in the PLC.

⇨ In the Solution Explorer configuration within the project tree, a linked variable from the cyclic interface is indicated by a blue or white icon.



3. Activate the TwinCAT project.

3.2.5.1.3 Link the axis object to the hardware

Procedure

- For instructions on linking an axis object to the hardware - both automatically and manually - see Link an axis object to hardware [▶ 121].
- In the axis settings, you can specify the axis type, the additional encoder type, and the task context. Depending on the selection, the corresponding structure in the axis object changes.
  - Details on the axis types can be found in Drive modules [▶ 41].
  - Details about the encoders can be found at Encoder modules [▶ 80].
  - Details about the task context can be found at Context tab [▶ 19].
- In hydraulic systems, the following axis types are typically used:
  - Analog Drive [▶ 70] – is used whenever an analog proportional valve is installed.
  - CoE DS408 Drive (hydraulic) [▶ 68] – is used whenever a proportional valve with a DS408 profile is installed.
  - Pulse Width Current Drive (MDP250) (both channels) [▶ 77] – is used whenever a proportional valve with pulse width modulation (PWM) is installed.
- In hydraulic systems, the following encoder types are typically used:
  - EtherCAT Encoder [▶ 80] – is used when a position measuring system with an EtherCAT interface is installed.
  - SSI Encoder [▶ 88] - is used when a position measuring system with an SSI interface is installed.

- In hydraulic systems, analog sensors are typically used for force and pressure control, and their load type and scaling must be adjusted:
  - Analog Load Encoder [▶ 85] – is used when analog (voltage/current) pressure sensors are installed.

3.2.5.2 Axis parameters

A general distinction is made between parameters that can be adjusted only during initialization and those that can also be adjusted online.

Furthermore, the parameters of a FluidPower axis can be divided into two categories:

- Standard parameters corresponding to those of an electric axis
  - Init Parameter, see: Init Parameter tab [▶ 141].
  - Online Parameter, see: Online Parameter tab [▶ 144].
- Specific parameters of a FluidPower axis
  - Init Parameter, see: Init Parameter [▶ 145].

3.2.5.2.1 Init Parameter tab

General

Name	Default value	Unit	Type
Simulation Mode	TRUE		BOOL
Position Unit	mm	[PosUnit]	MC.EPositionUnit [▶ 202]
Is Modulo Axis	FALSE		BOOL
Position Modulus	360.0	[PosUnit]	LREAL
Position Modulo Tolerance Window	0.0	[PosUnit]	LREAL

Simulation Mode

The Simulation Mode indicates whether the I/O connection to the drive and encoder should be active (`FALSE`) or simulated (`TRUE`). The parameter can be written at runtime if the axis is not enabled by the controller.

Position Unit

The Position Unit field is used for display purposes, such as on the commissioning interface. Inputs and outputs on function blocks are not converted.

Is Modulo Axis

Enable this parameter to use modulo functionality in function blocks for this axis. The setpoint and actual values are calculated as absolute and modulo values in the cyclic interface. To exchange Modulo information - for example, with the PLC - you must enable the Modulo data area [▶ 32] .

- **Position Modulus**
  Modulo value used to calculate the modulo position
- **Position Modulo Tolerance Window**
  Movement in the opposite direction to the specified direction is permitted, provided that the target position lies within this position tolerance of the command's starting position. This parameter affects the behavior of MC_MoveModulo [▶ 272]. It interacts with the setting selected at the `Direction` input of the function block.

Maximum Dynamics

Name	Default value	Unit	Type
Maximum Velocity	2000.0	[PosUnit]/s	LREAL

Name	Default value	Unit	Type
Maximum Acceleration	10000.0	[PosUnit]/s^2	LREAL
Maximum Deceleration	10000.0	[PosUnit]/s^2	LREAL
Maximum Jerk	100000.0	[PosUnit]/s^3	LREAL

Maximum Velocity

Maximum speed taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Velocity` input of a function block, the value defined here is used as the limit value for the velocity.

Maximum Acceleration

Maximum acceleration taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Acceleration` input of a function block, the value defined here is used as the limit value for acceleration.

Maximum Deceleration

Maximum delay taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Deceleration` input of a function block, the value defined here is used as the limit value for the delay.

Maximum Jerk

Maximum jerk taken into account during profile generation. Input values for function blocks are automatically reduced to this limit if values that are too high are specified. If the constant `#Maximum` is specified at the `Jerk` input of a function block, the value defined here is used as the limit value for jerk.

Default Dynamics

Name	Default value	Unit	Type
Default Acceleration	2000.0	[PosUnit]/s^2	LREAL
Default Deceleration	2000.0	[PosUnit]/s^2	LREAL
Default Jerk	4000.0	[PosUnit]/s^3	LREAL

Default Acceleration

This value is used as the acceleration at the `Acceleration` input on function blocks, unless it is explicitly set. If the constant `#Default` is specified at the input of a function block, the value defined here is used as the limit value for acceleration.

Default Deceleration

This value is used as the delay at the `Deceleration` input on function blocks, unless it is explicitly set. If the constant `#Default` is specified at the input of a function block, the value defined here is used as the limit value for the delay.

Default Jerk

This value is used as jerk at the `Jerk` input on function blocks, unless it is explicitly set. If the constant `#Default` is specified at the input of a function block, the value defined here is used as the limit value for jerk.

Manual Motion

Name	Default value	Unit	Type
Jog Velocity Slow	100.0	[PosUnit]/s	LREAL
Jog Velocity Fast	250.0	[PosUnit]/s	LREAL

Jog Velocity Slow

This target velocity is used for slow movement of an axis using manual control buttons. This procedure can be performed using `MC_Jog` as well as through the commissioning interface.

Jog Velocity Fast

This target velocity is used for rapid movement of an axis using manual control buttons. This procedure can be performed using `MC_Jog` as well as through the commissioning interface.

Monitoring

Name	Default value	Unit	Type
Position Limit Enforcement Enabled	FALSE		BOOL
Minimum Position	#Ignore	[PosUnit]	LREAL
Maximum Position	#Ignore	[PosUnit]	LREAL
Position Lag Monitoring Enabled	TRUE		BOOL
Maximum Position Lag Value	5.0	[PosUnit]	LREAL
Maximum Position Lag Filter Time	0.02	s	LREAL
Position Lag Calculation	HardwareProvidedIfAvailable		MC.EPositionLagSource [▶ 201]

Position Limit Enforcement Enabled

This parameter specifies whether position limits are taken into account when generating motion profiles. If the value is `TRUE`, the following two parameters must be set.

- **Minimum Position**
  : The minimum allowed position for the axis. Use `#Ignore` to disable enforcement of the position limit in this direction of motion. It is not possible to move an axis beyond this limit. If an axis is outside its position limits when TwinCAT starts up, it can move toward the valid position range. Commands with a target position outside this lower limit will be rejected.
- **Maximum Position**
  : The maximum allowable position for the axis. Use `#Ignore` to disable enforcement of the position limit in this direction of motion. It is not possible to move the axis beyond this limit. If an axis is outside its position limits when TwinCAT starts up, it can move toward the valid position range. Commands with a target position outside this lower limit will be rejected.

Position Lag Monitoring Enabled

This parameter specifies whether the position lag for the axis should be monitored. If the value is `TRUE`, the following two parameters are displayed for configuring position and time. If the parameterized limits are exceeded, a runtime error is output.

The calculation is performed as follows:

Position lag value = current target position - actual position

- **Maximum Position Lag Value and Maximum Position Lag Filter Time**
  The `Maximum Position Lag Value` is the position lag value that must not be exceeded for longer than the specified time (`Maximum Position Lag Filter Time`). Otherwise, the axis is stopped immediately by direct shutdown, and an error is generated.

Position Lag Calculation

This parameter specifies whether the position lag is calculated by the software from the MC axis (`InternallyCalculated`) or read directly from the linked hardware (`HardwareProvidedIfAvailable`). If `HardwareProvidedIfAvailable` is set but no hardware is available, the system falls back to the internally calculated value. For a simulation axis, `HardwareProvidedIfAvailable` uses data from the simulation drive.

Error event handling

Name	Default value	Unit	Type
Control Error Setpoint Mode	HardStop		MC.EControlErrorSetpoint Mode [▶ 196]

If there is a hardware error (e.g., in the control loop) and the MC no longer has control over the axis, you can configure the behavior of setpoint generation here. `HardStop` causes the axis to stop abruptly in the event of a fault.

Diagnostics

Name	Default value	Unit	Type
TraceLevel	tlWarning		TcTraceLevel

Specify which messages should be sent to the TwinCAT error list.

Additional Encoder

Name	Default value	Unit	Type
Main Position Encoder			OTCID_PositionEncoder
Load Control Type	None		MC.ELoadType [▶ 200]

Main Position Encoder

If an additional encoder is available, select its object ID here.

Load Control Type

Set the type of load to be controlled here.

3.2.5.2.2 Online Parameter tab

The following online parameters are available:

Monitoring

Name	Unit	Type
Minimum Position Lag	[PosUnit]	LREAL
Maximum Position Lag	[PosUnit]	LREAL
Trigger Minimum/Maximum Position Lag Reset		Trigger Minimum/Maximum Position Lag Reset

Minimum Position Lag

This read-only online parameter displays the minimum detected position lag for informational purposes.

Maximum Position Lag

This read-only online parameter displays the maximum detected position lag for informational purposes.

Trigger Minimum/Maximum Position Lag Reset

A rising edge at this parameter input resets the stored minimum and maximum position lags.

Additional Error Event Information

Name	Unit	Type
Originating Error Event Sender ID		OTCID
Original Error Id		HRESULT

Originating Error Event Sender ID

If a subsequent error occurs - for example, due to coupling - this parameter displays the object ID of the axis from which the error originated.

Original Error Id

If a subsequent error occurs - for example, due to a coupling - this online parameter displays the original error number.

Override

Name	Unit	Type
Velocity Override Factor		LREAL

Velocity Override Factor

The current velocity override for the axis is displayed here.

3.2.5.2.3 Init Parameter

Cylinder Dimensions

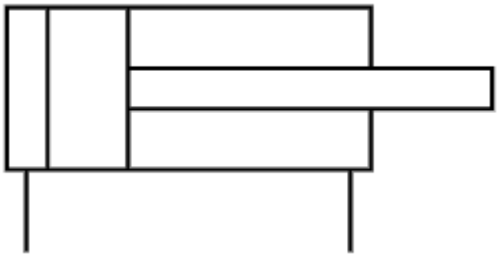
Name	Default value	Unit	Type
Cylinder Design	Differential		MC.ECylinderDesign
Cylinder Bore Diameter	`#Ignore`	mm	LREAL
Cylinder Rod Diameter	`#Ignore`	mm	LREAL

Cylinder Design

The `Cylinder Design` parameter selects the type of hydraulic cylinder and determines how the piston and piston rod diameters are used for force and area calculations.

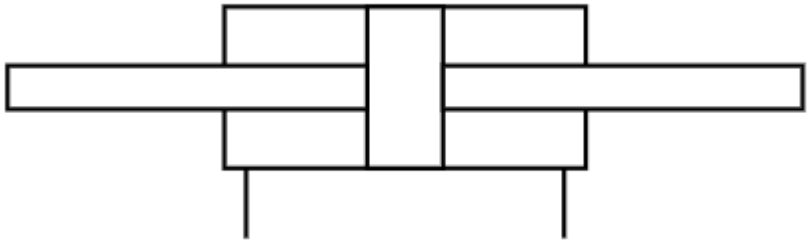
`Differential`

The differential cylinder is a double-acting hydraulic cylinder with a single-ended piston rod.



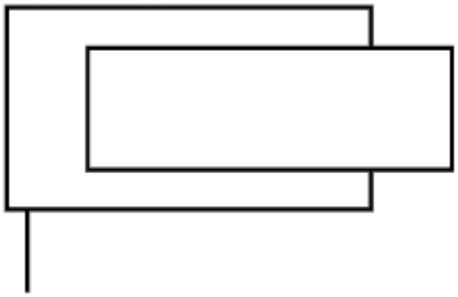
`Synchronous`

The synchronous cylinder is a double-acting hydraulic cylinder with a through-rod.



Plunger

The plunger cylinder is a single-acting hydraulic cylinder without a piston rod.



Cylinder Bore Diameter

Inner diameter of the hydraulic cylinder bore. Used to calculate the piston area and the force.

Cylinder Rod Diameter

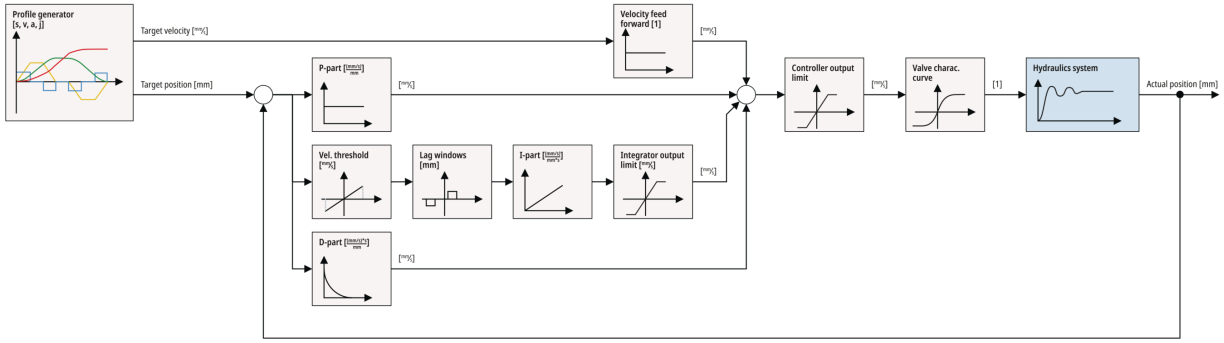
Diameter of the cylinder rod. Relevant for the design of differential and synchronous cylinders when calculating the effective piston area on the rod side.

3.2.5.3 Position controller

The setpoint generator is based on a **7-phase profile** for defining a trajectory that takes into account acceleration, velocity, and deceleration setpoints. Control is achieved using a **conventional PID controller**, with the **integral component (I)** adjusted in specific ways to prevent integral overshoot, oscillation, and drift.

To improve dynamic performance, a **Velocity feedforward control** is implemented, which reduces the load on the controller and shortens the response time to setpoint changes. This is followed by **downstream controller limiting** and the **integration of the valve characteristics** as nonlinear control value compensation. This also integrates the characteristic flow parameters of the valve being used into the control system, ensuring a linearized control response and higher accuracy across the entire operating range.

The following block diagram shows a graphical representation of the position controller:



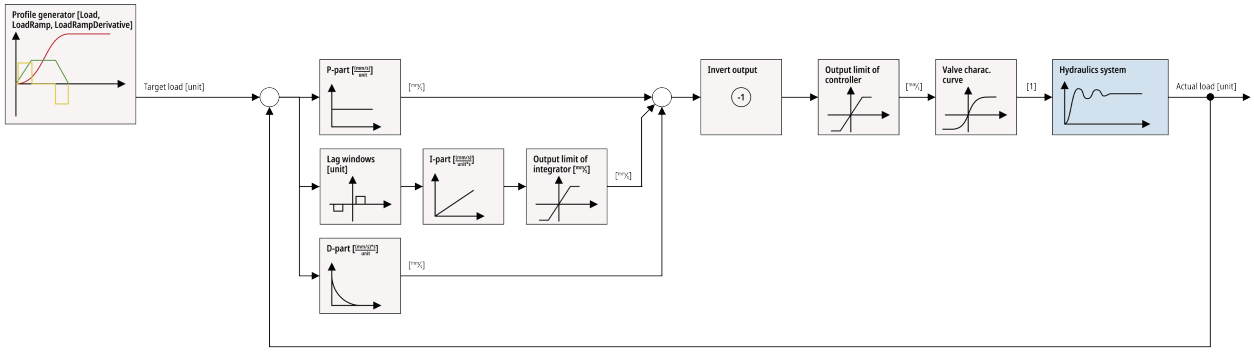
- For explanations of the control parameters, see Init Parameter [▶ 96].
- For explanations of the online parameters, see Online Parameter [▶ 97]

3.2.5.4 Pressure/force controller

The setpoint generator is based on a **5-phase profile** for defining the trajectory, that takes into account pressure ramps and their derivatives. Control is achieved using a **conventional PID controller**, with the **integral component (I)** adjusted in specific ways to prevent integral overshoot, oscillation, and drift.

Following the controller, an optional **controller inversion** is performed, along with **downstream controller limiting** and the **integration of the valve characteristics** as nonlinear control value compensation. This also integrates the characteristic flow parameters of the valve being used into the control system, ensuring a linearized control response and higher accuracy across the entire operating range.

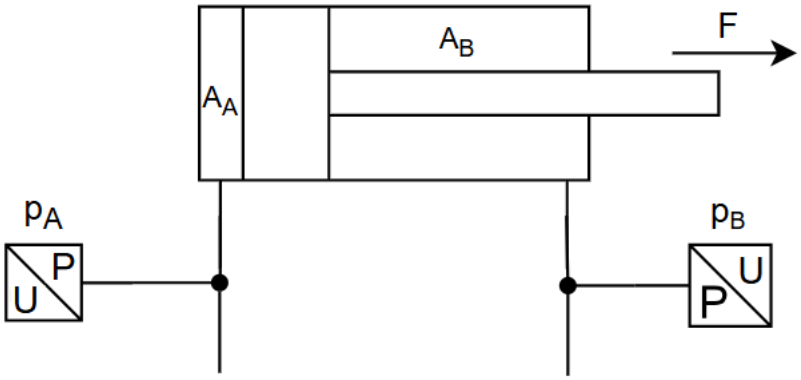
The block diagram shows a graphical representation of the force/pressure controller:



- For explanations of the control parameters, see here: Init Parameter [▶ 93]
- You can find explanations of the online parameters here: Online Parameter [▶ 95]

3.2.5.5 Pressure sensor A/B

Pressure sensors are used to measure pressures within a hydraulic system. The FluidPower axis uses pressure sensors A and B for this purpose, with pressure sensor A measuring the pressure on the piston side and sensor B measuring the pressure on the rod side of a hydraulic cylinder.



You can find the parameters for configuring the pressure sensors here: Analog Load Encoder [▶ 85]

3.2.5.5.1 Linking | Analog inputs

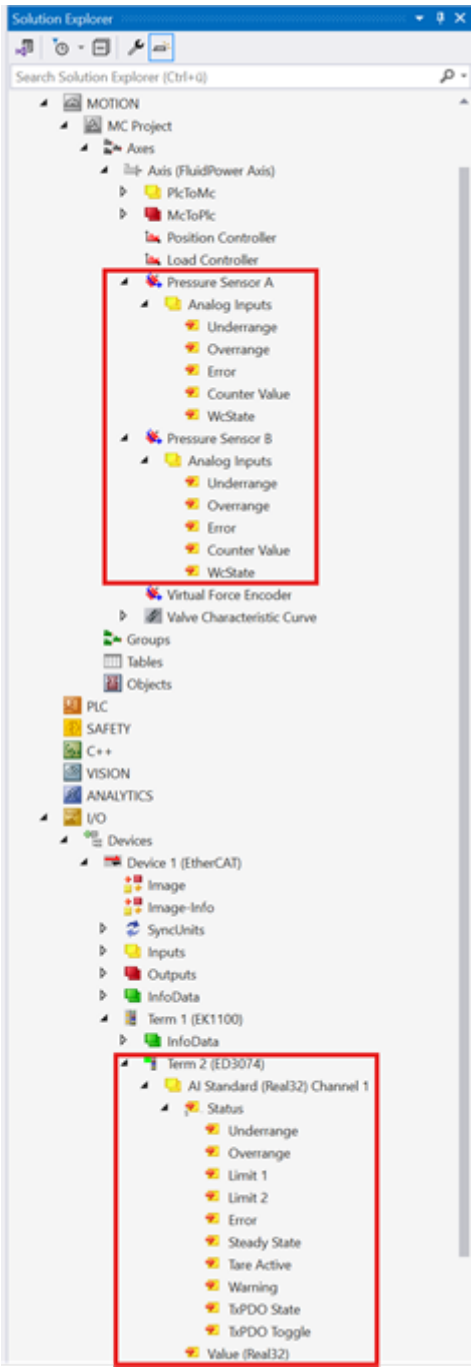
Requirements

- An MC3 FluidPower axis was added to the TwinCAT project;
  see Create axis object/add axis [▶ 136], or
- An MC3 axis has been extended to include an Analog Load Encoder object;
  see Expand the axis object with an additional module [▶ 116]

Initial situation

- The hardware configuration has been created.

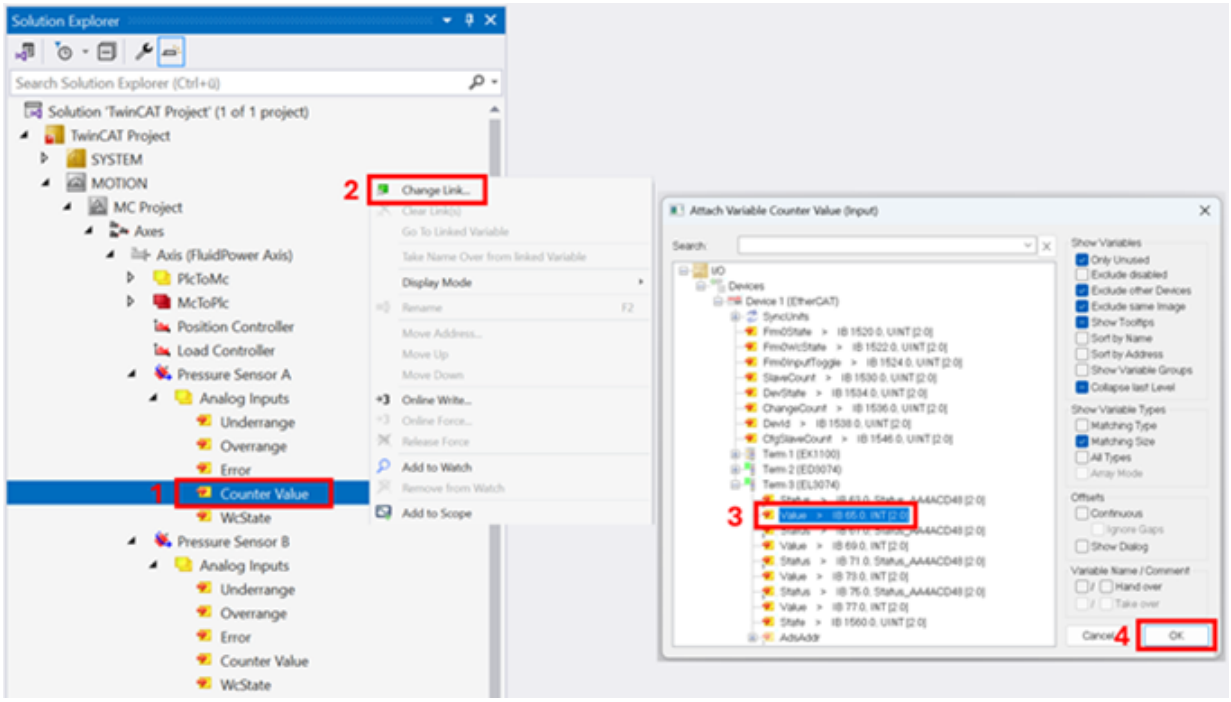
- Pressure sensor A/B has been created.



Procedure: Example for EL307x, ED307x

1. Select the link you want to link to.
2. Right-click on **Change Link…**
3. Select the hardware link you want to link.

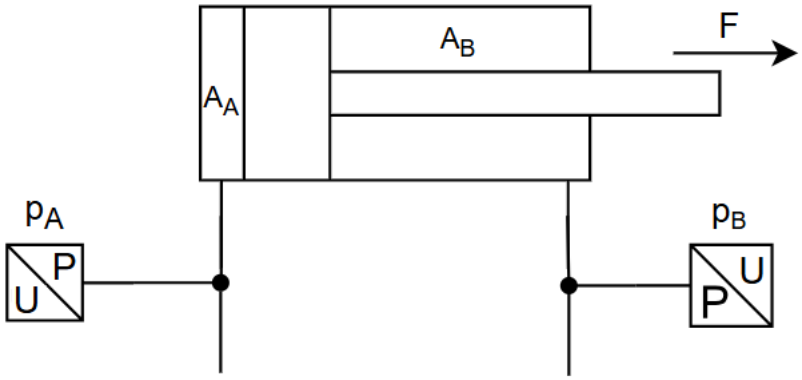
4. Confirm the selection with the **OK** button.



3.2.5.6 Virtual force sensor

A **virtual force sensor** is used to calculate force when configuring a FluidPower axis.

The force is calculated using the parameterized cylinder data Init Parameter [▶ 145] and the corresponding pressures A/B Pressure sensor A/B [▶ 147].



The force F is calculated from the cylinder areas A_A and A_B and the pressures p_A and p_B as follows.

$F = A_A * p_A – A_B * p_B$

3.2.5.6.1 Virtual Force Encoder - Initialization Parameters

Scaling

Name	Default value	Unit	Type
Direction Inverted	FALSE		BOOL

Direction Inverted

The `Direction Inverted` parameter can adjust the force counting direction as follows:

- `FALSE`: The force polarity matches the counting direction of the acquisition hardware.
- `TRUE`: The force polarity is inverted relative to the counting direction of the acquisition hardware.

⚠ WARNING

Risk of injury from unexpected movements

If the encoder's counting direction and the actuator's polarity do not match, the axis may move unexpectedly.

- When commissioning the system, make sure that the encoder's counting direction is set correctly.
- Always maintain a safe distance.

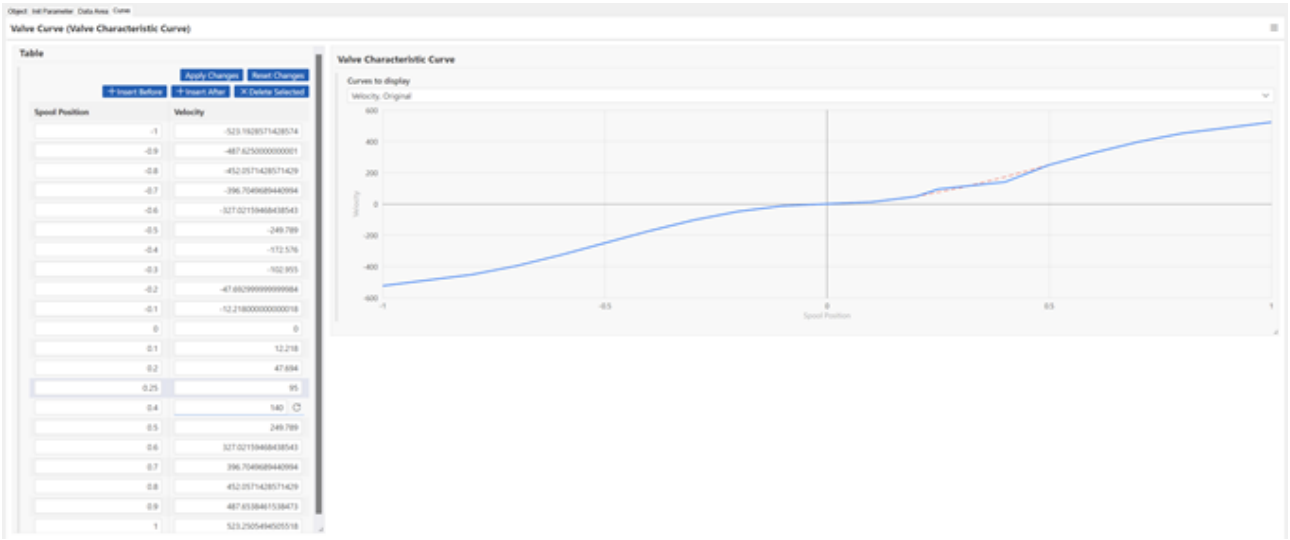
3.2.5.7 Valve characteristics

The MC3 FluidPower axis enables the implementation and automatic recording of the valve characteristics. This is connected downstream of the position or force/pressure controller.

By determining the characteristic curve in the moving system, all nonlinear properties of the hydraulic system are identified and eliminated. In other words, in addition to the valve's flow characteristic curve, the characteristic curve also includes the nonlinear effects of oil flow through hydraulic blocks, hoses, and pipes, as well as those of the hydraulic cylinder.

Curve

In the **Curve** tab of the valve characteristics object, an existing valve characteristic is displayed in tabular form and as a graph.



Table

In the “Table” card, you can make changes to existing points on the valve characteristics curve. Likewise, using the buttons

- Insert Before
- Insert After
- Delete Selected

points can be added or removed.
Changes are, using the buttons

- Apply changes
- Reset changes

saved or reset.

Valve Characteristic Curve

The “Valve characteristic curve” card displays a graphical representation of the current valve characteristics. The “Curves to display” selection provides the following options:

- Velocity
- Original

By default, “Velocity” is enabled. This selection displays the currently configured curve.
Selecting the “Original” checkbox also displays the most recently saved valve characteristic curve.

For information on the parameters of the valve characteristics object, see Valve Characteristic Curve [▶ 99].

For an explanation of the menu, see Setting options [▶ 34].

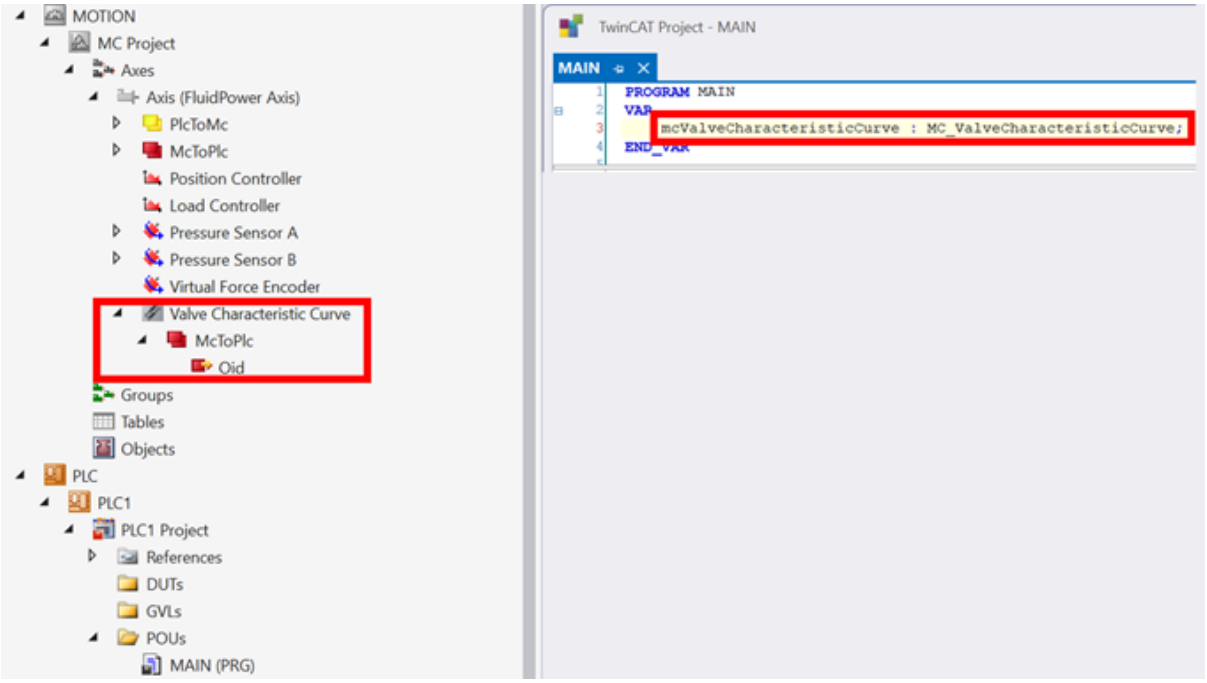
3.2.5.7.1 Linking MC_ValveCharacteristicCurve

Requirements

- An MC3 FluidPower axis was added to the TwinCAT project;
  see Create an axis object [▶ 25], or
- An MC3 axis has been extended to include a Valve Characteristic Curve object;
  see Expand the axis object with an additional module [▶ 116]
- A PLC project containing an instance of the function block Tc3_Mc3FluidPower.MC_ValveCharacteristicCurve was created;
  see MC_ValveCharacteristicCurve

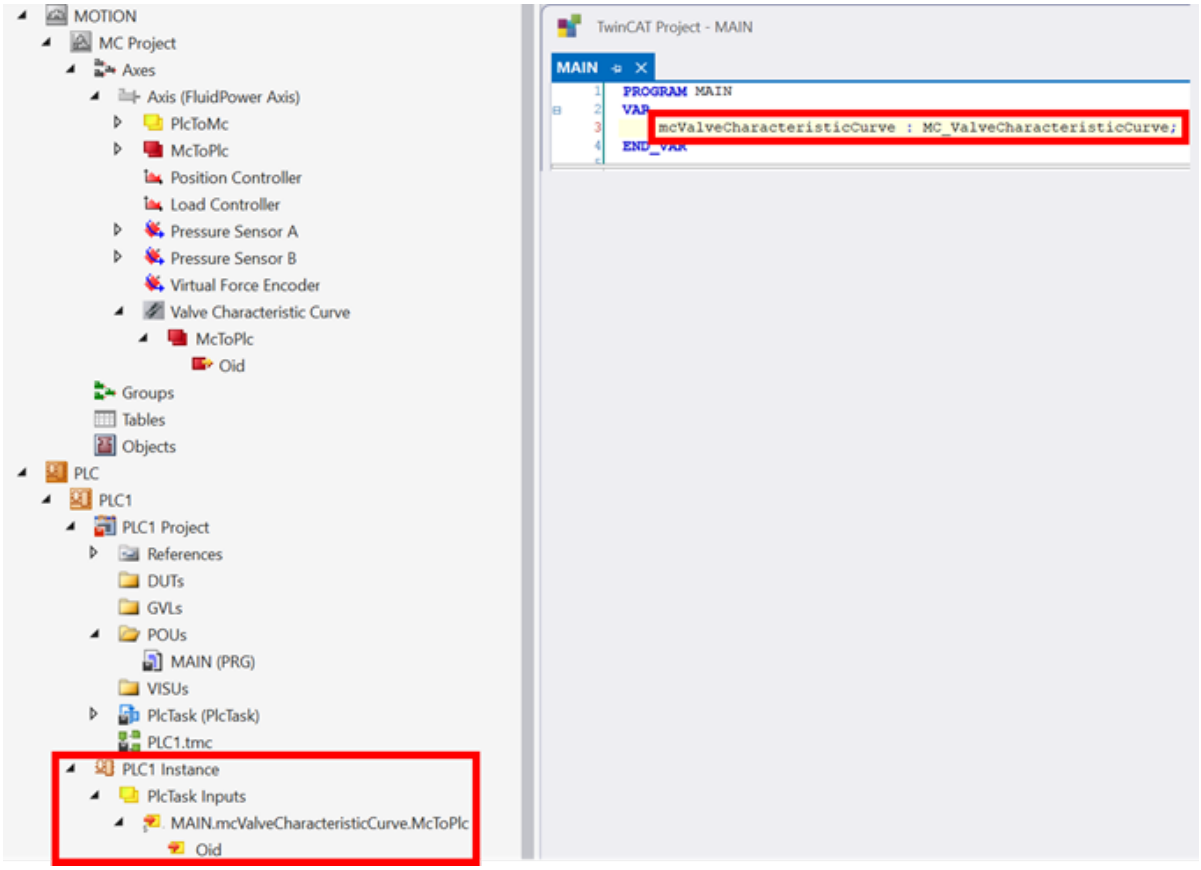
Initial situation

- Valve Characteristic Curve object created
- Instance of Tc3_Mc3FluidPower.MC_ValveCharacteristicCurve created

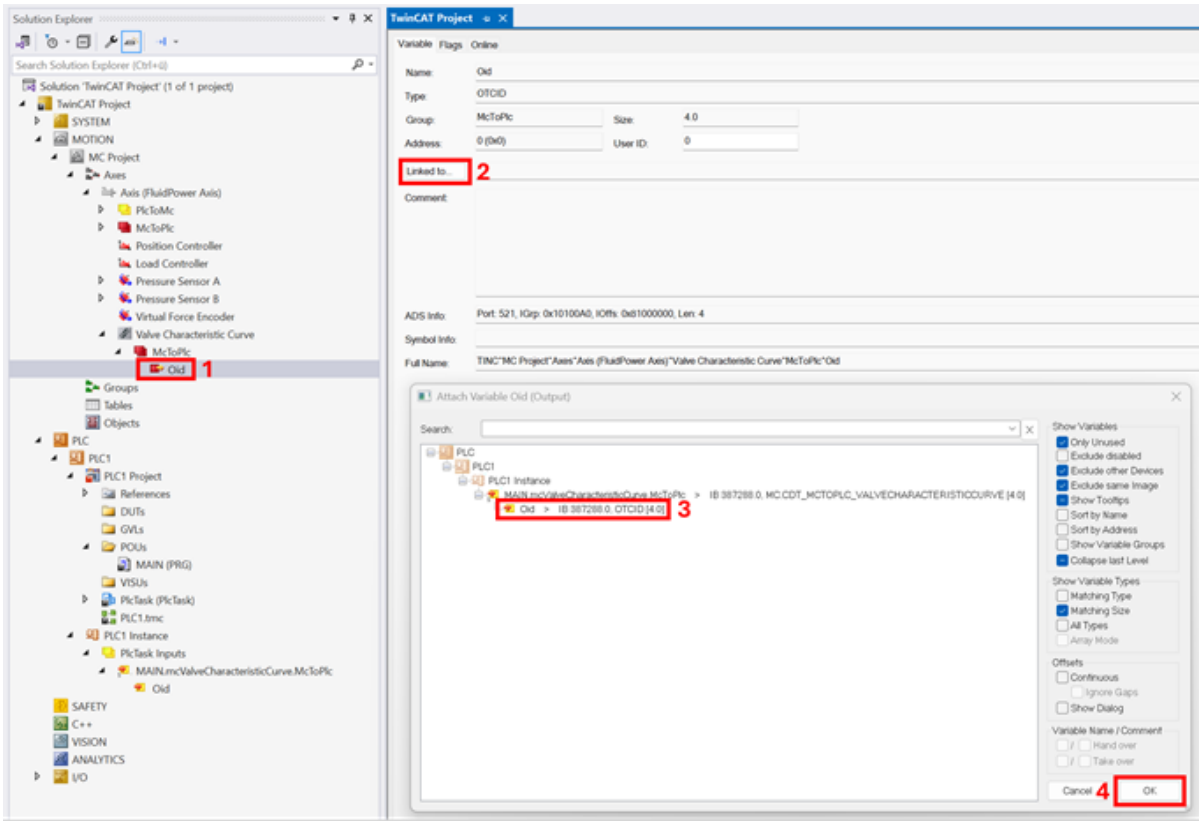


How to

1. Create the PLC project
   → The `Tc3_Mc3FluidPower.MC_ValveCharacteristicCurve` instance appears under PLC Instances in Solution Explorer.



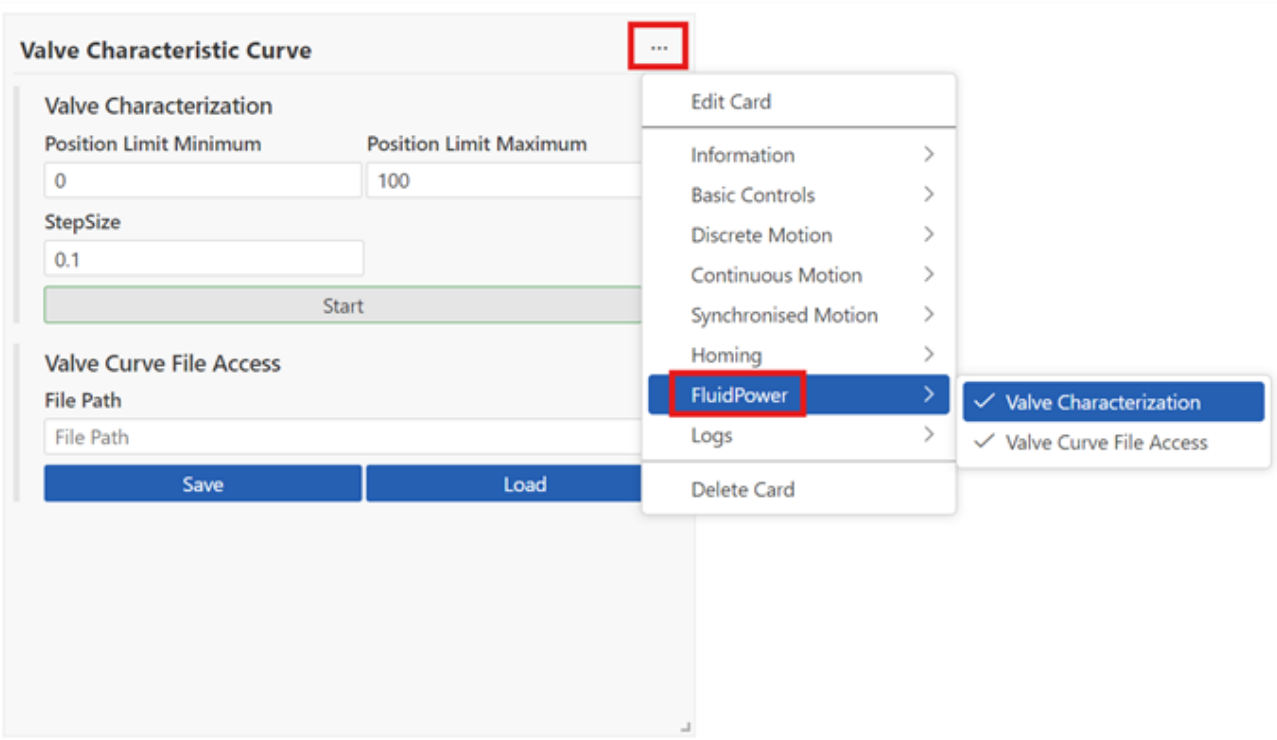
2. Link the PLC instance to the Valve Characteristic Curve instance in MC3.



3. Activate the TwinCAT project.

3.2.5.7.2 Motion UI

By clicking on “FluidPower” in the Cards [▶ 38] menu on the Motion UI tab [▶ 33], options for identifying and handling the valve characteristics can be displayed on a card.



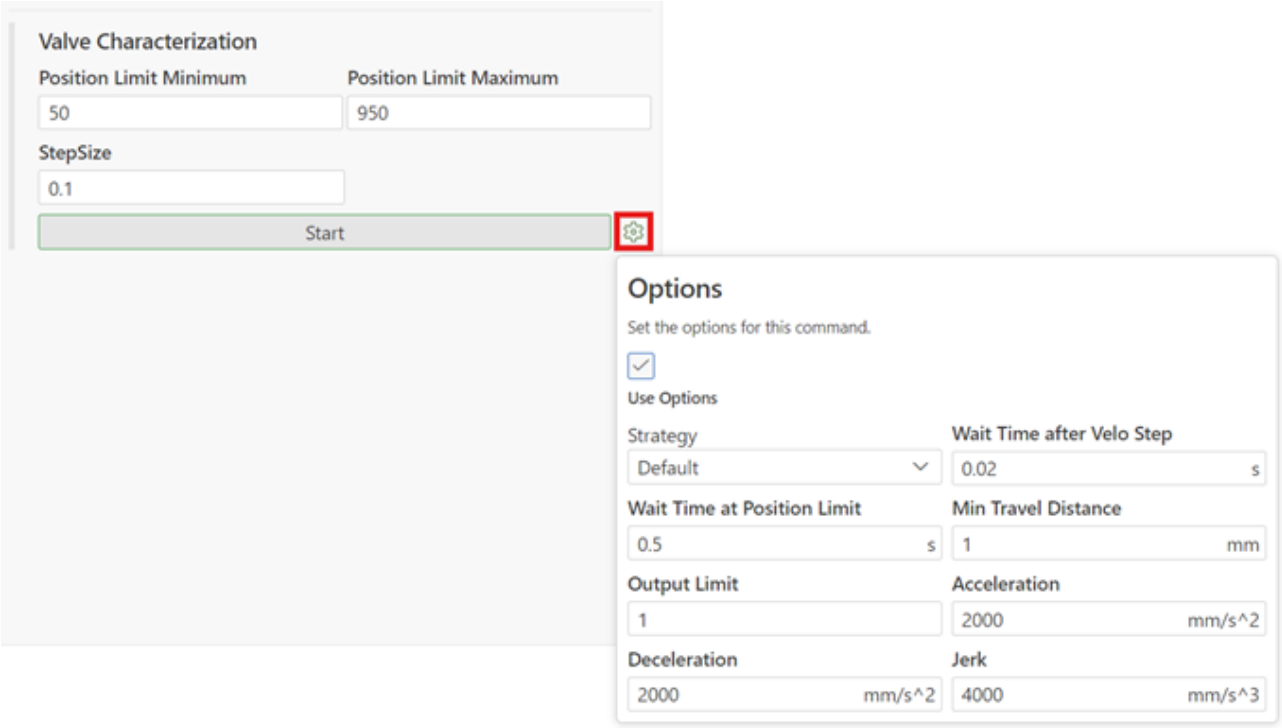
3.2.5.7.2.1 Identification of the valve characteristics

Requirements

- The axis and encoder types were configured in the MC3 axis settings; see Settings tab [▶ 27]
- The velocity feedforward control of the position controller must be set to 1.0; see Init Parameter [▶ 96]

Procedure

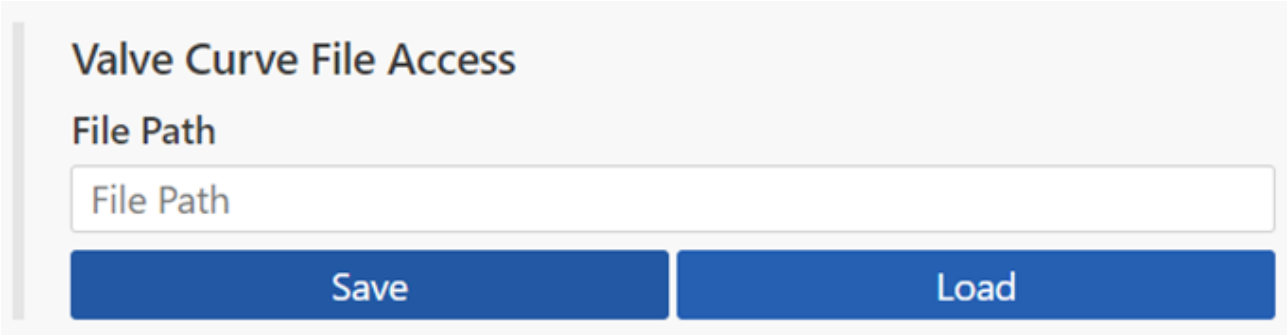
Once the MC3 axis is configured, the valve characteristic curve can be identified using the “Valve Characterization” option via the Motion UI, without the need for a PLC project.



1. Set the desired parameters.
2. Perform the identification by clicking “Start”.
   → The button changes to “Stop”.
   → Once the valve characteristics have been successfully identified, the button changes back to “Start”.
3. To save the characteristic curve, see File access [▶ 155]

3.2.5.7.2.2 File access

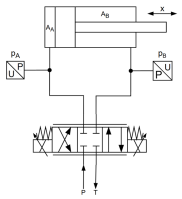
The “Valve Characteristics File Access” option allows you to save the valve characteristics to a file or load them from a file.



- Clicking “Save” saves the current valve characteristics to a file on the target system.
- Clicking “Load” loads an existing valve characteristic from a file on the target system.
- “File Path” specifies the file path on the target system.
  → If nothing is entered, the default path is: “C:\ProgramData\Beckhoff\TwinCAT\3.1\Boot\TcCOM”.

3.2.5.8 Commissioning the FluidPower axis (valve-controlled axis)

The following figure shows an example schematic of a differential cylinder whose position and pressure/force are to be controlled via a 4/3-way proportional valve.



The following section provides a step-by-step description of how to perform commissioning of the position and pressure control of such an axis, using an example with the following components.

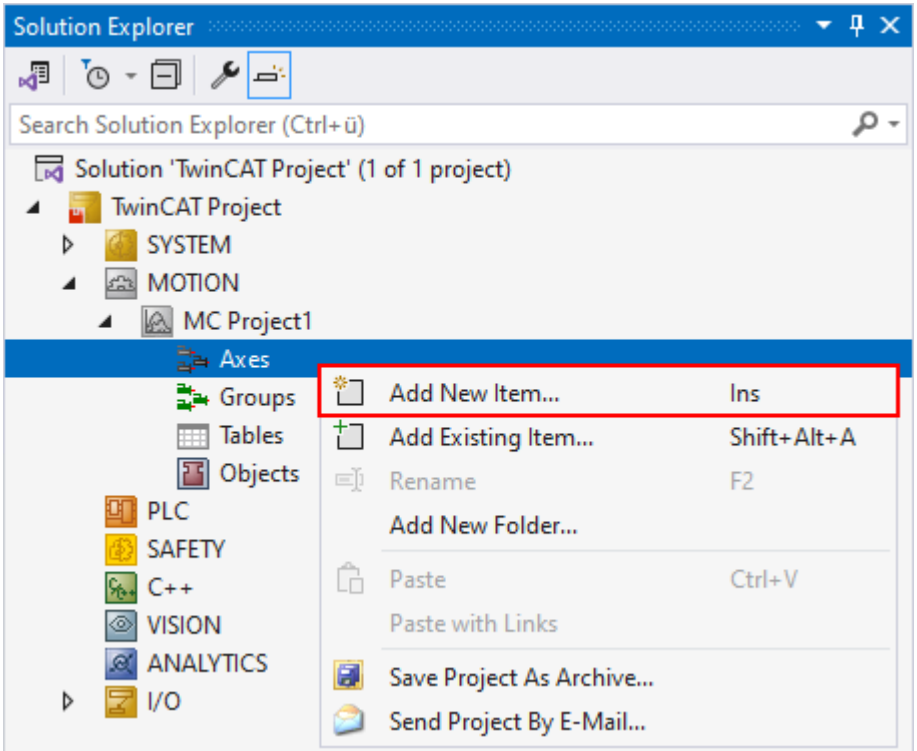
- 50/32 mm hydraulic differential cylinder
- Absolute SSI encoder, 0.1 µm resolution
- Analog 4/3-way proportional valve with analog output terminal
- Pressure sensors A/B, 0–400 bar with an analog input terminal

The first step is to create a FluidPower axis;
see Adding an axis [▶ 156]

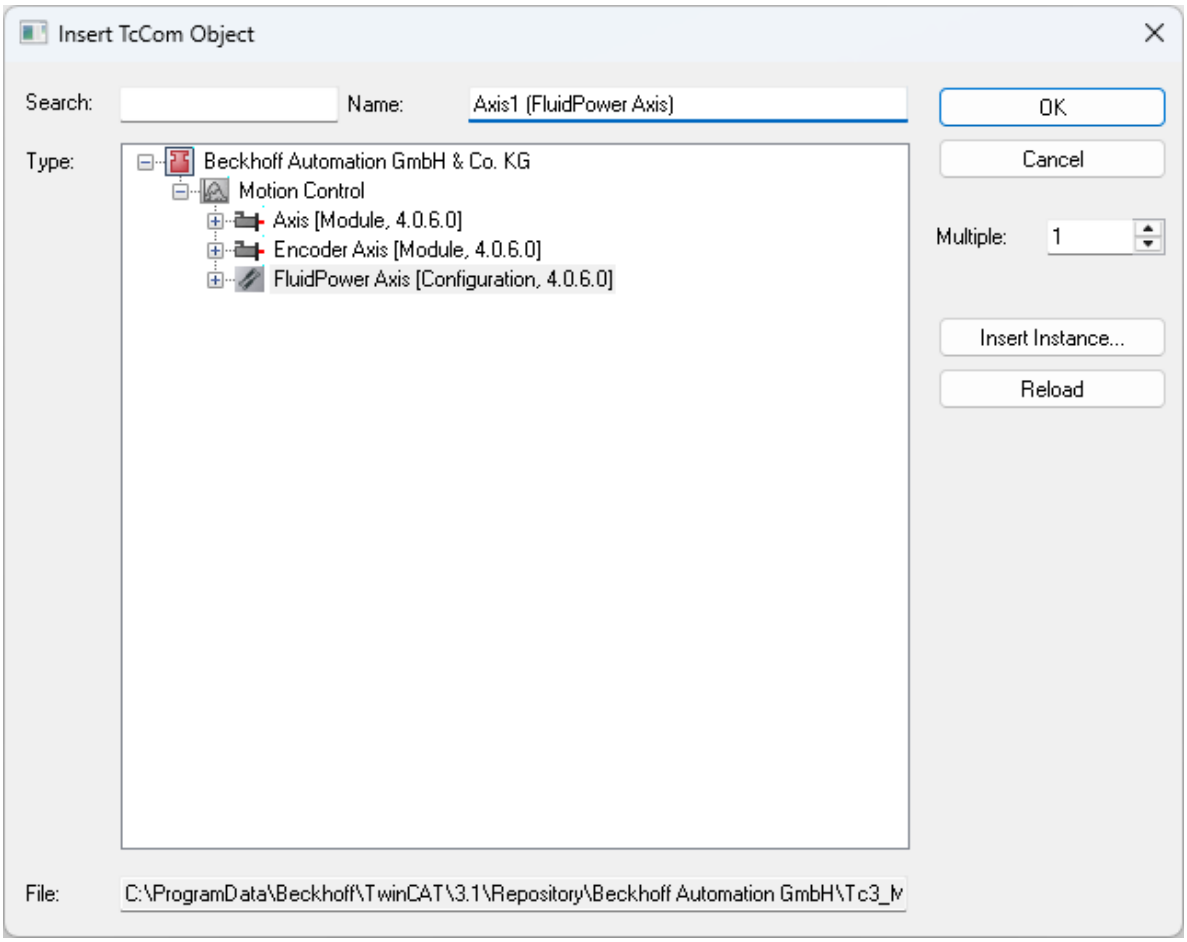
3.2.5.8.1 Adding an axis

✓ A TwinCAT project with an MC project has been created;
see Creating an MC project [▶ 17].

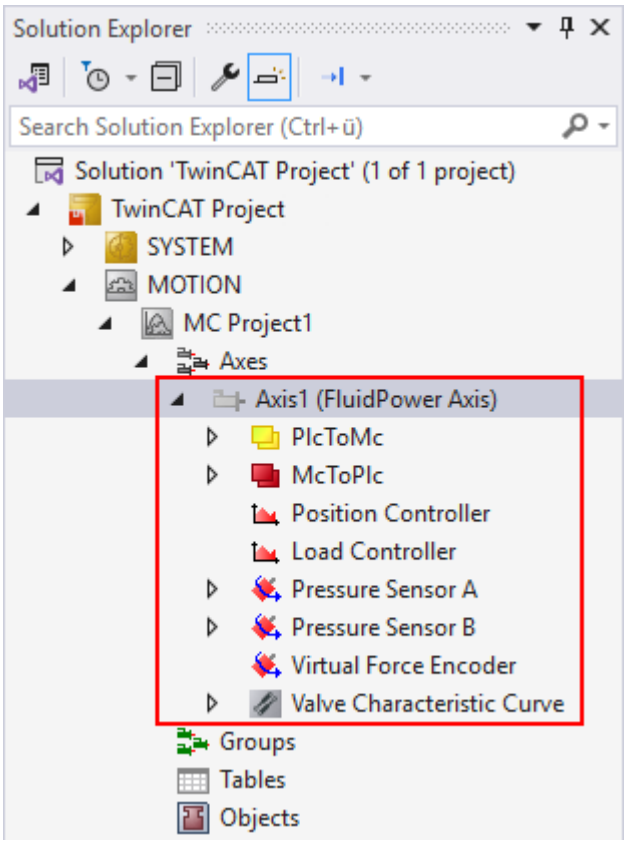
1. Right-click **Axes > Add New Item...**



⇨ In the **Insert TcCOM Object** dialog box, select the **FluidPower Axis** axis type under **Type** and click **OK**.

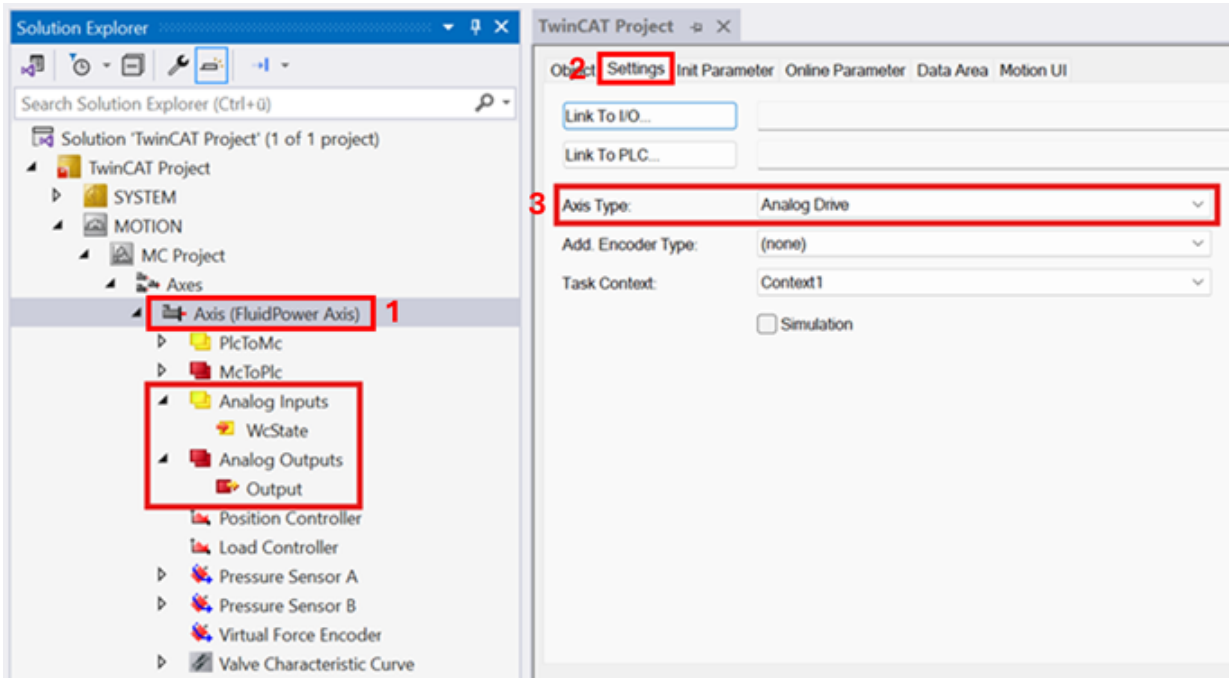


The FluidPower axis has been created with the required modules and is located below the Axes node.



Select axis type

2. Selection of axis type via the Settings tab [▶ 27] of the axis object.

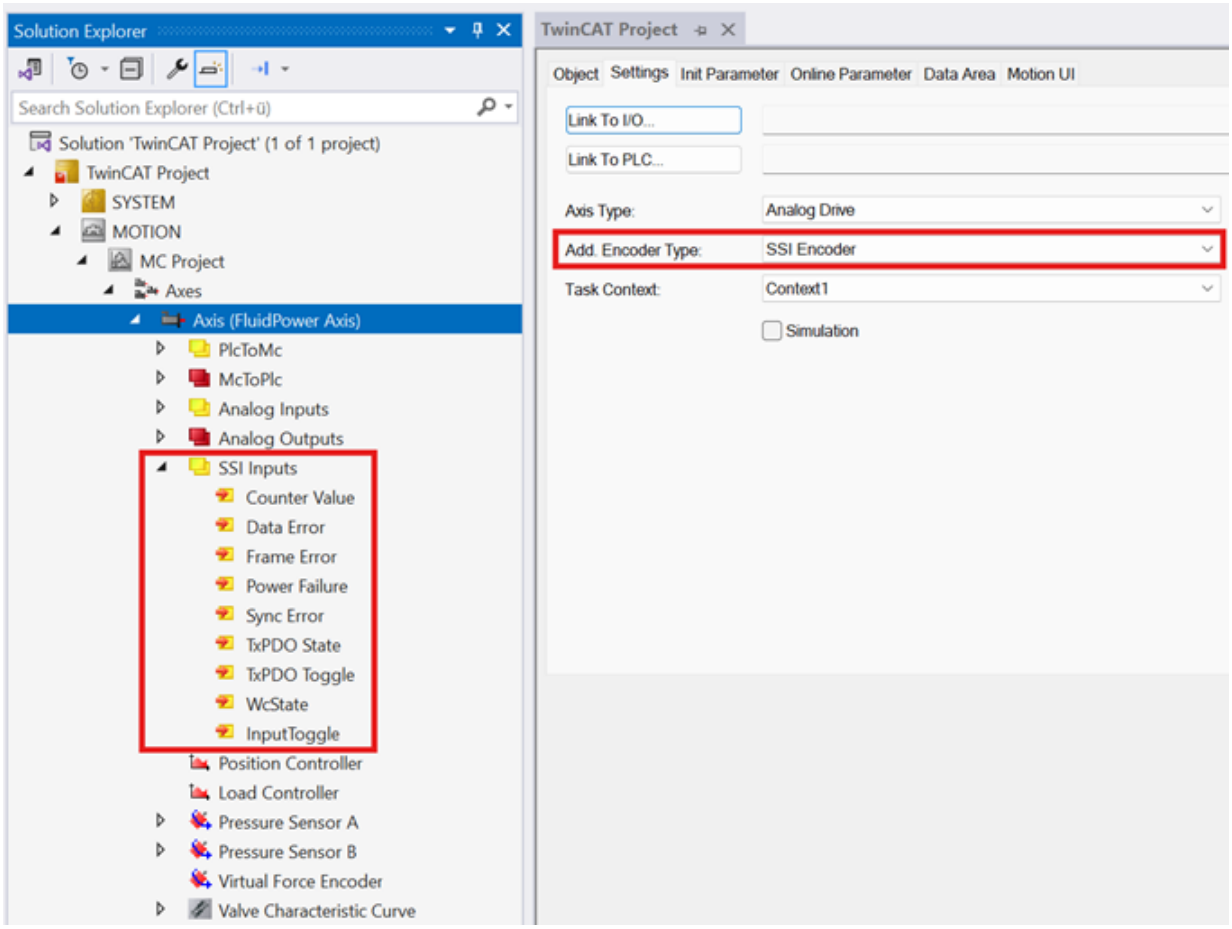


1. Double-click the **FluidPower Axis** object
2. Open the **Settings** tab
3. Selection of an **Analog Drive** to control an analog proportional valve

→ The analog modules, including the Data Area, have been created below the FluidPower axis.

Select encoder type

3. Selection of the **SSI encoder** type on the Settings tab [▶ 27] of the axis object



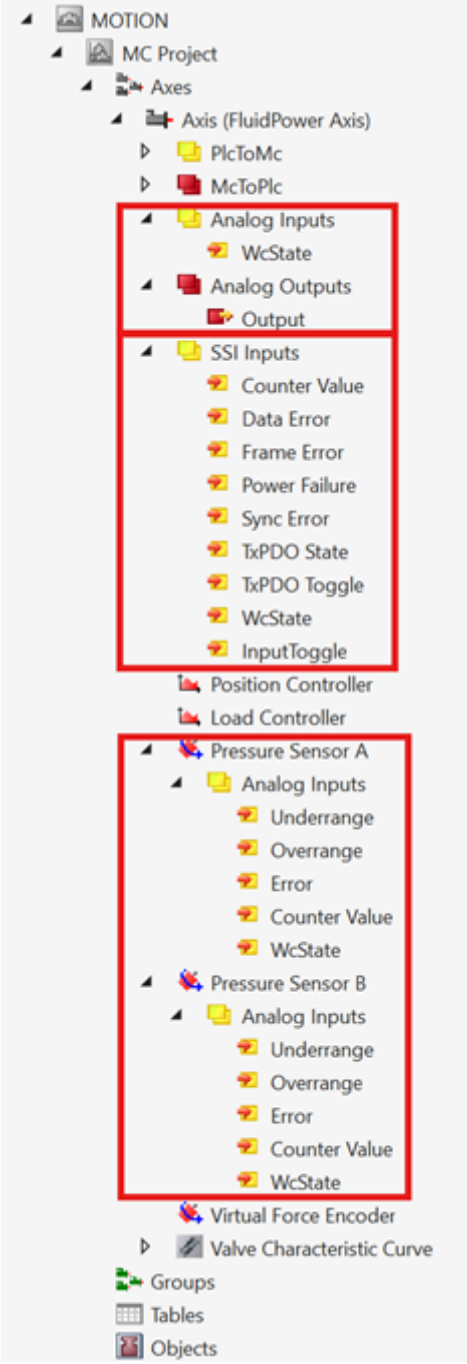
⇨ The SSI Inputs module, including the Data Area, was created below the FluidPower axis.

⇨ In the next step, the axis objects are linked to the hardware;
see Link axis objects to hardware [▶ 159].

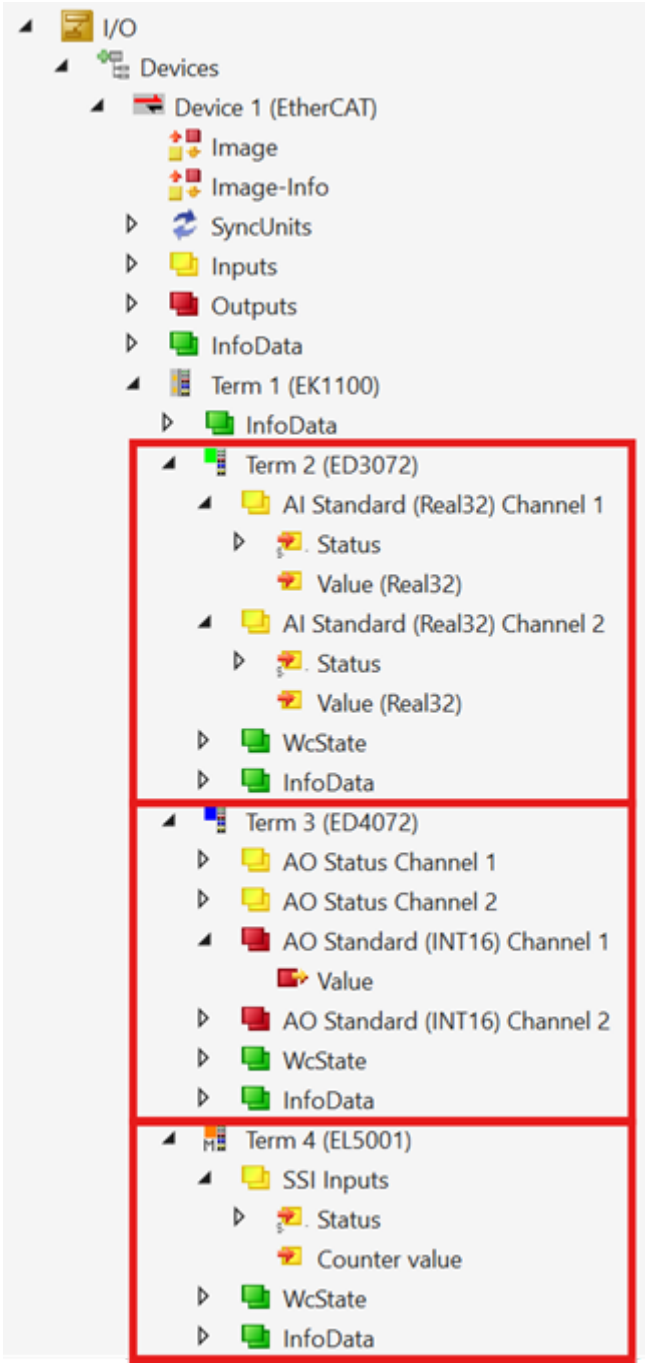
3.2.5.8.2 Link axis objects to hardware

- An MC3 axis was created, and its axis and encoder types were configured;
see Adding an axis [▶ 156].
- A hardware configuration has been created;
see Adding an I/O Device.

- An axis configuration has been created (pressure sensors A/B, axis type: analog drive, encoder type: SSI encoder).

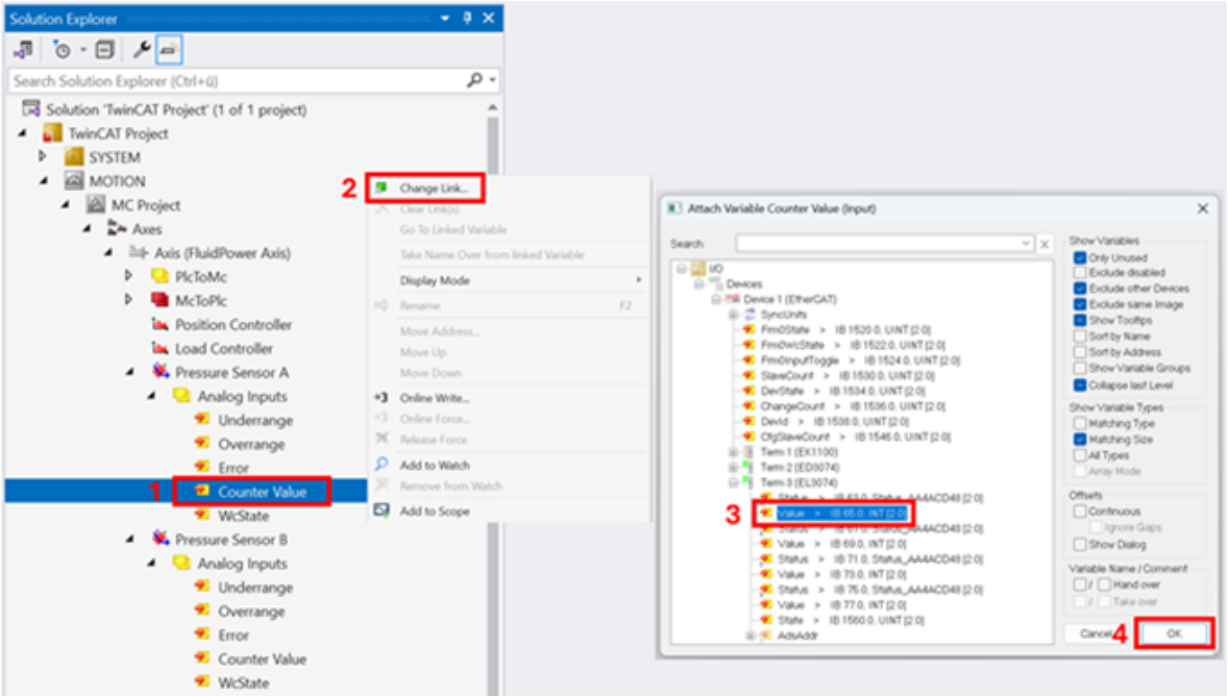


- A hardware configuration has been created (Ain, Aout, and SSI terminals).



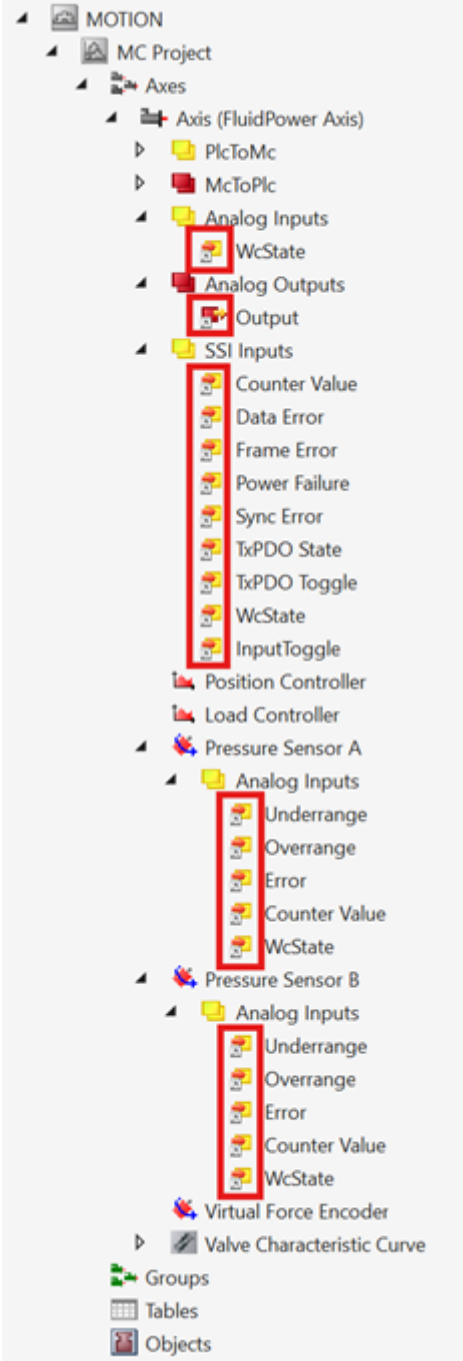
1. Select the link you want to connect from the axis configuration,
   for example, Pressure Sensor A → Counter Value to link the counter input of pressure sensor A.
2. Right-click on **Change Link…**
3. Select the hardware link you want to connect,
   for example, EL3074 → Value to connect analog input 1.

4. Confirm your selection with **OK**.



5. Repeat the process for all links available in the axis and hardware configuration.

⇨ In the Solution Explorer configuration in the project tree, all variables from the cyclic I/O interface are linked and marked with a white icon.



6. Activate the TwinCAT project.

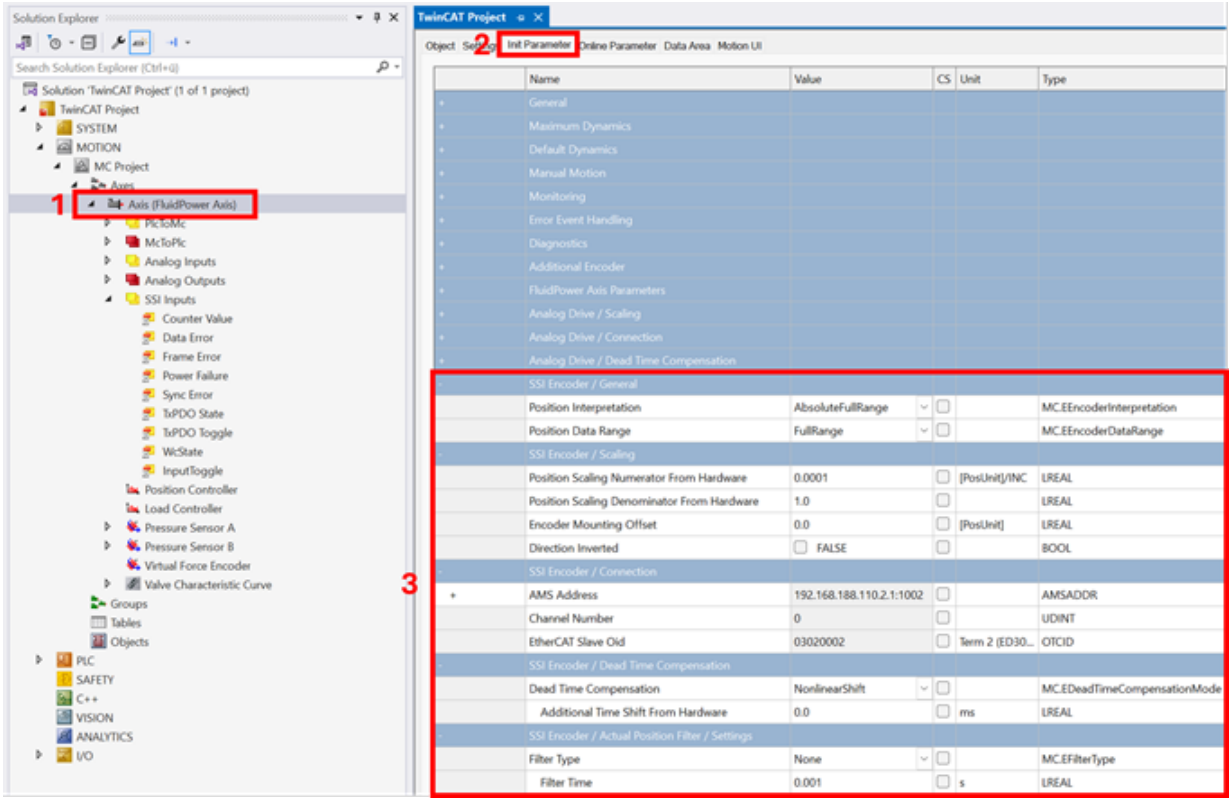
The next step is the parameterization of the SSI encoder;
see Set the position measuring system [▶ 163].

3.2.5.8.3 Set the position measuring system

✓ An MC3 axis was created and an SSI encoder was configured;
see Adding an axis [▶ 156].

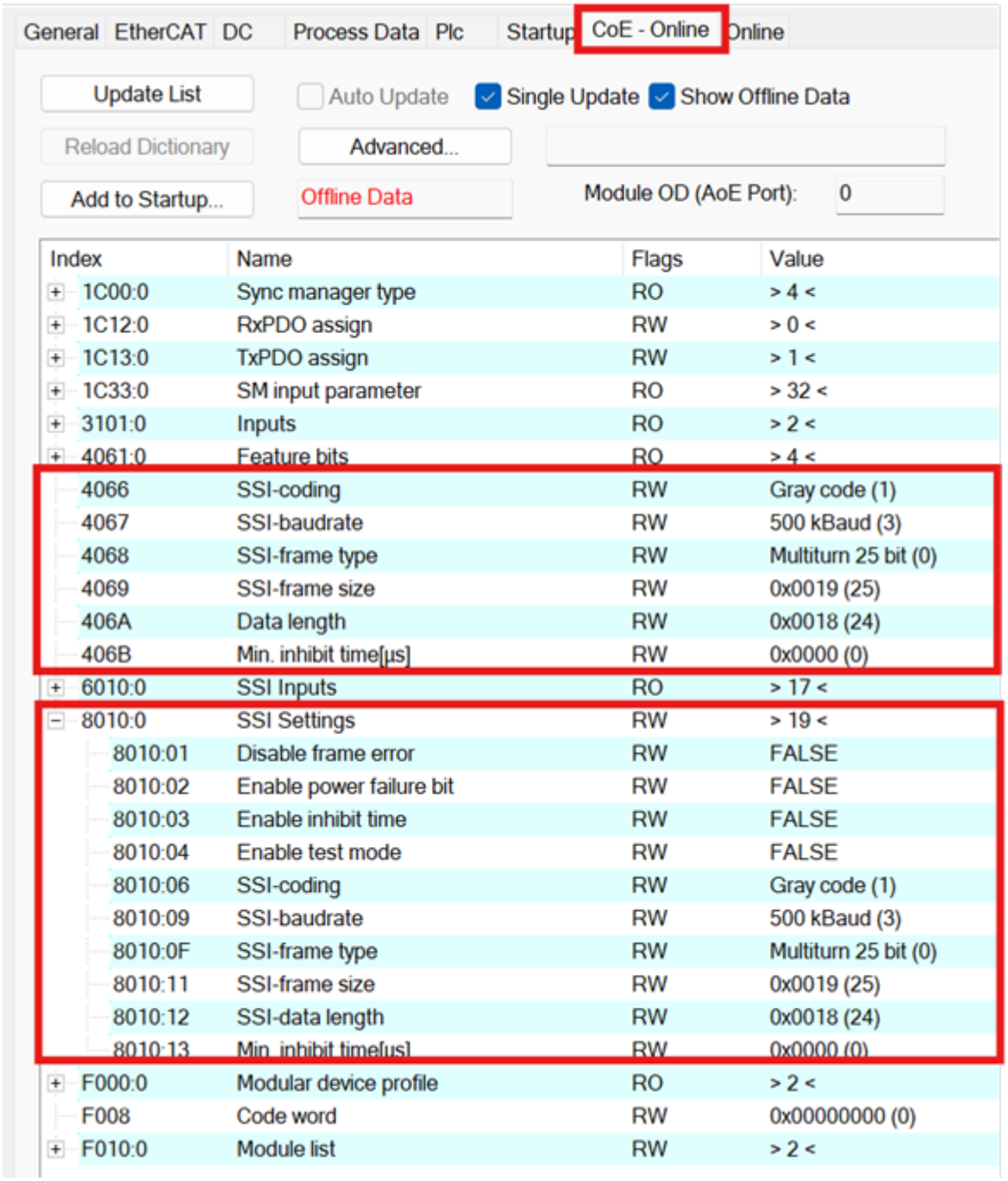
1. Double-click the **FluidPower Axis** object.
2. Open the **Init Parameter** tab.

3. Configure the **SSI encoder** groupings using the data sheet for the encoder you are using, e.g., resolution scaling of 0.1 µm.



Explanations of the parameters can be found here: Init Parameter [▶ 88]

4. Parameterize the remaining parameters for the SSI encoder in the terminal under CoE – Online; see https://infosys.beckhoff.com/content/1033/el500x/2247264523.html



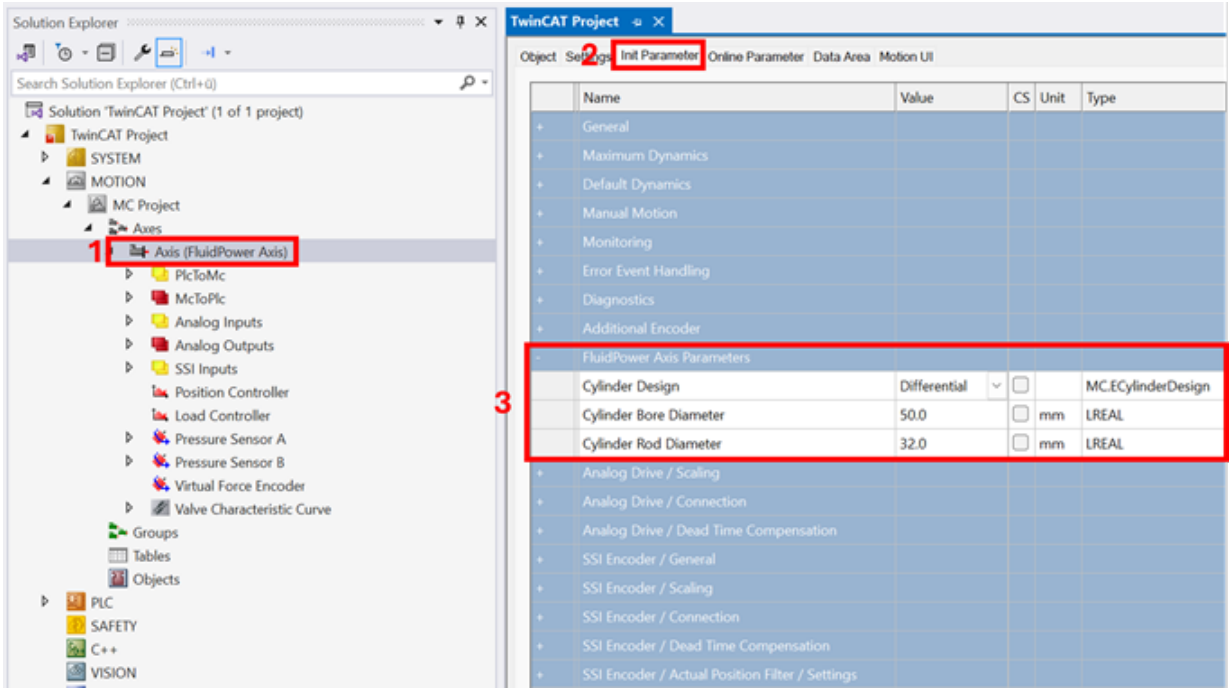
The next step is the parameterization of the cylinder data;
see Set cylinder data [▶ 165].

3.2.5.8.4 Set cylinder data

✓ A FluidPower axis has been created;
see Adding an axis [▶ 156].

1. Double-click the **FluidPower Axis** object.
2. Open the **Init Parameter** tab.

3. Enter the cylinder data in the **FluidPower Axis Parameters** groups, for example, Differential Cylinder 50/32.



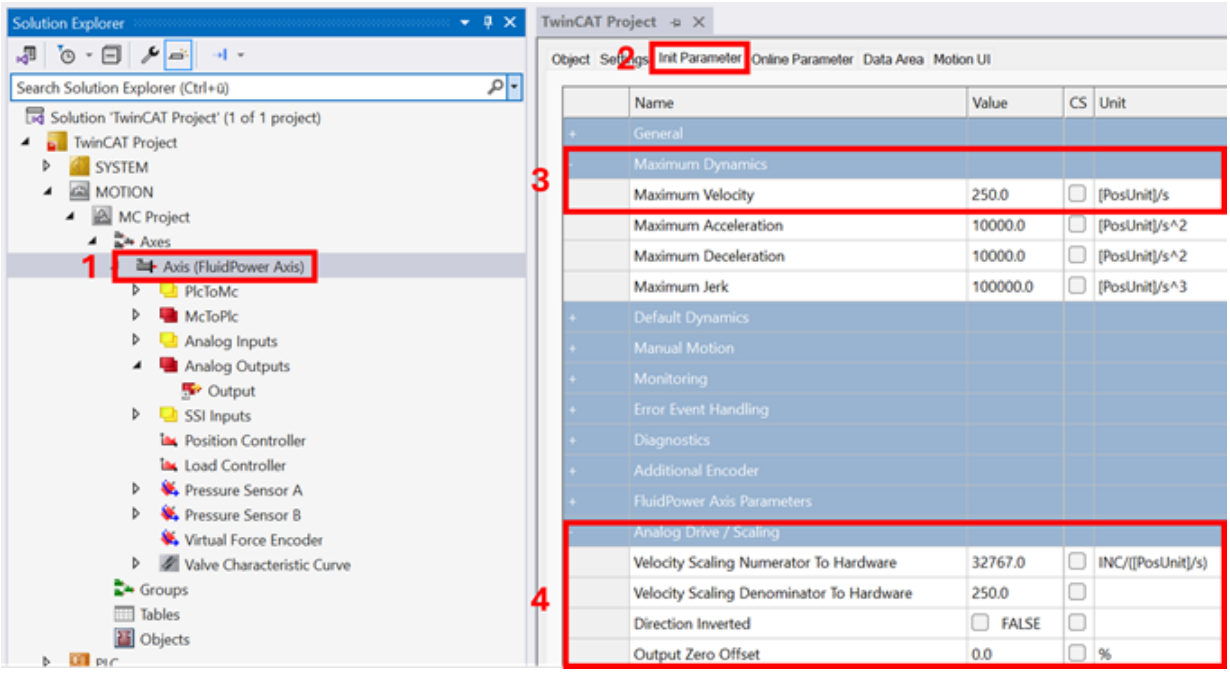
Explanations of the parameters can be found here: Init Parameter [▶ 145]

The next step is the parameterization of the output signal;
see Set the valve output [▶ 166].

3.2.5.8.5 Set the valve output

✓ An MC3 axis was created and an Analog Drive was configured as the axis type;
see Adding an axis [▶ 156].

1. Double-click the **FluidPower Axis** object.
2. Open the **Init Parameter** tab.
3. Set the **Maximum Velocity** in the **Maximum Dynamics** group.
   The Maximum Velocity is the reference velocity of the position controller. It represents the maximum velocity that the hydraulic cylinder can reach when the valve is 100% open. This value must be calculated taking into account the cylinder dimensions and the valve size.
   This is also the velocity limit for profile generation.
4. Configure the **Analog Drive/Scaling** group.
   In this example, **Velocity Scaling Numerator** corresponds to the increments of the analog output module, and **Velocity Scaling Denominator** corresponds to the maximum velocity.



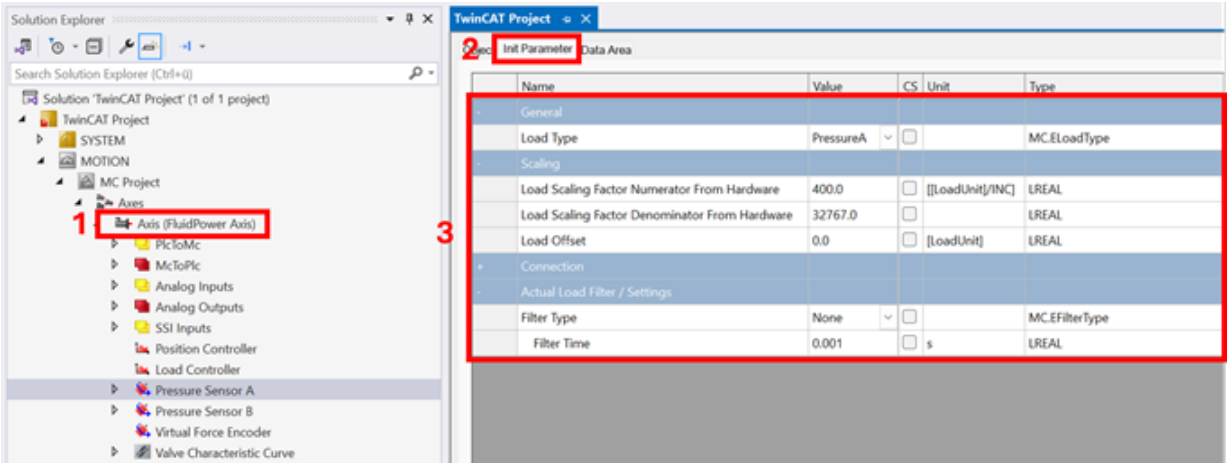
Explanations of the parameters can be found here: Init Parameter [▶ 71]

The next step is the parameterization of the pressure sensors;
see Set the pressure sensors [▶ 167].

3.2.5.8.6 Set the pressure sensors

✓ An MC3 FluidPower axis has been created;
see Adding an axis [▶ 156].

1. Double-click the **Pressure Sensor A/B** object.
2. Open the **Init Parameter** tab.
3. Set the parameters:
e.g., pressure sensor pA, 400 bar / 32,767-increment analog module.



Explanations of the parameters can be found here: Init Parameter [▶ 85]

4. Activate the TwinCAT project.
→ Configure the **Load Offset** to zero the sensor when the hydraulic system is depressurized.

Once parameterization is complete, move the axis in manual operation to check the direction of rotation;
see Move the axis in manual operation using Motion UI [▶ 168].

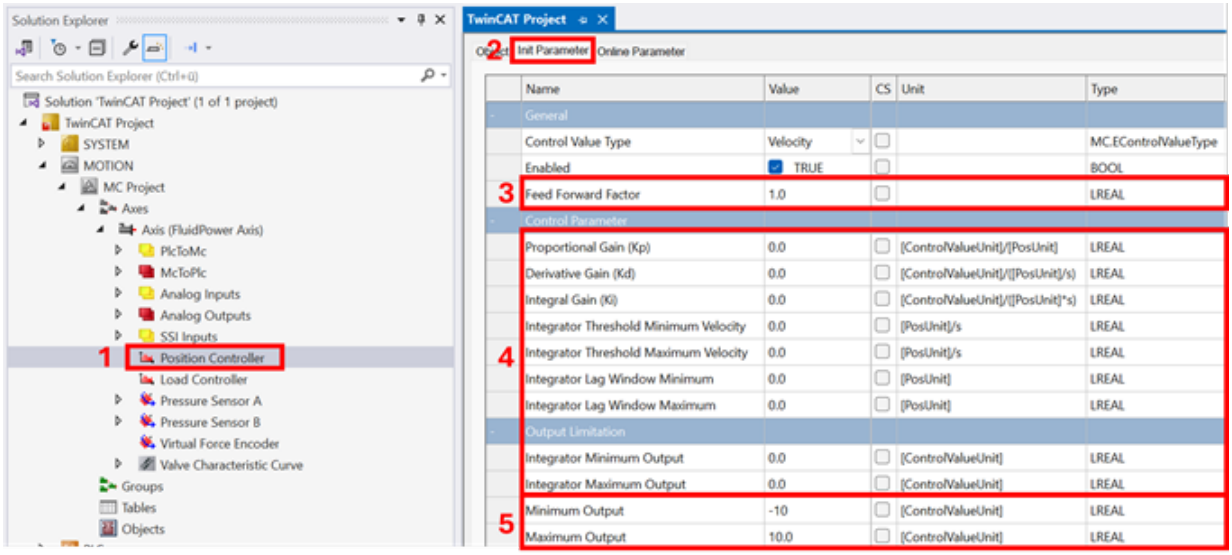
3.2.5.8.7 Move the axis in manual operation using Motion UI

Requirements

- An MC3 FluidPower axis was created, and its axis and encoder types were configured; see Adding an axis [▶ 156].
- The axis objects were linked to the hardware; see Link axis objects to hardware [▶ 159].
- The position measuring system has been configured; see Set the position measuring system [▶ 163].
- The valve output has been configured; see Set the valve output [▶ 166].

Control signal limitation

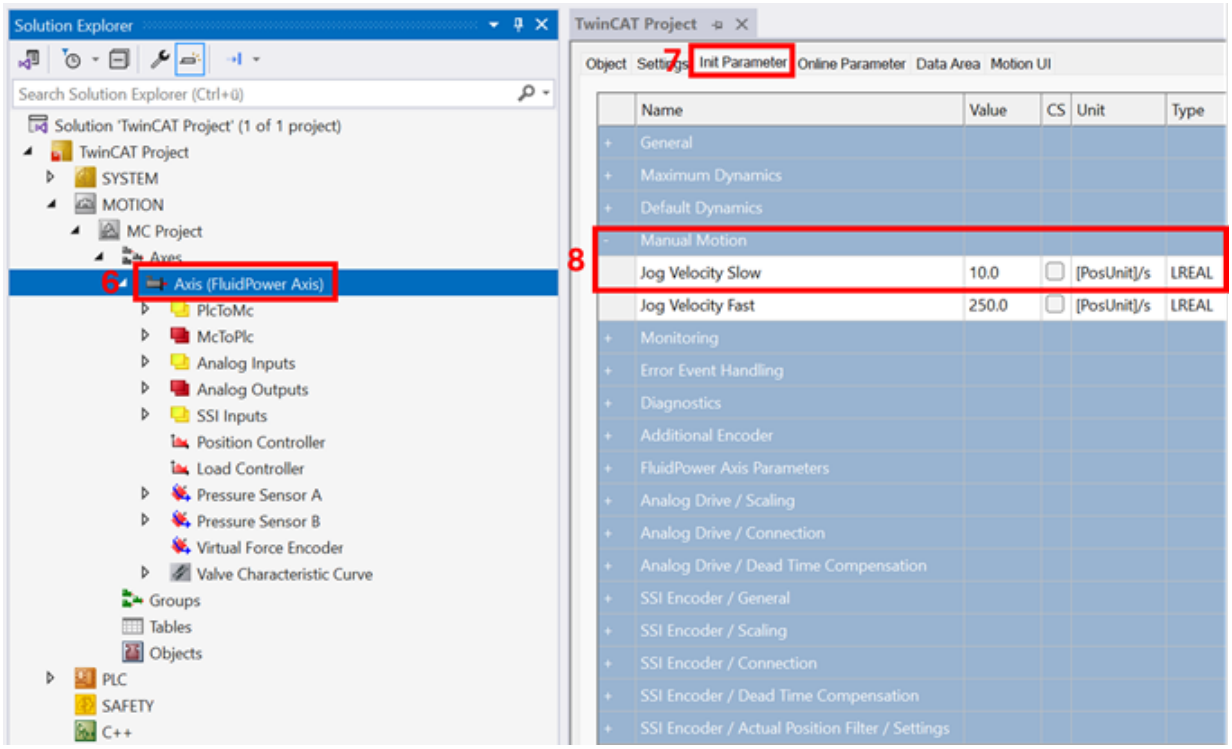
1. Double-click the **Position Controller** object.
2. Open the **Init Parameter** tab.
3. Set the **Velocity Feed Forward Factor = 1.0** in the **General** group.
4. Disable the position controller by setting all **Control parameters = 0.0** in the **Controller Parameters** and **Output Limitation** groups.
5. Limit the control signal using the **Minimum/Maximum Output** to prevent uncontrolled, rapid movements.



Select manual velocity

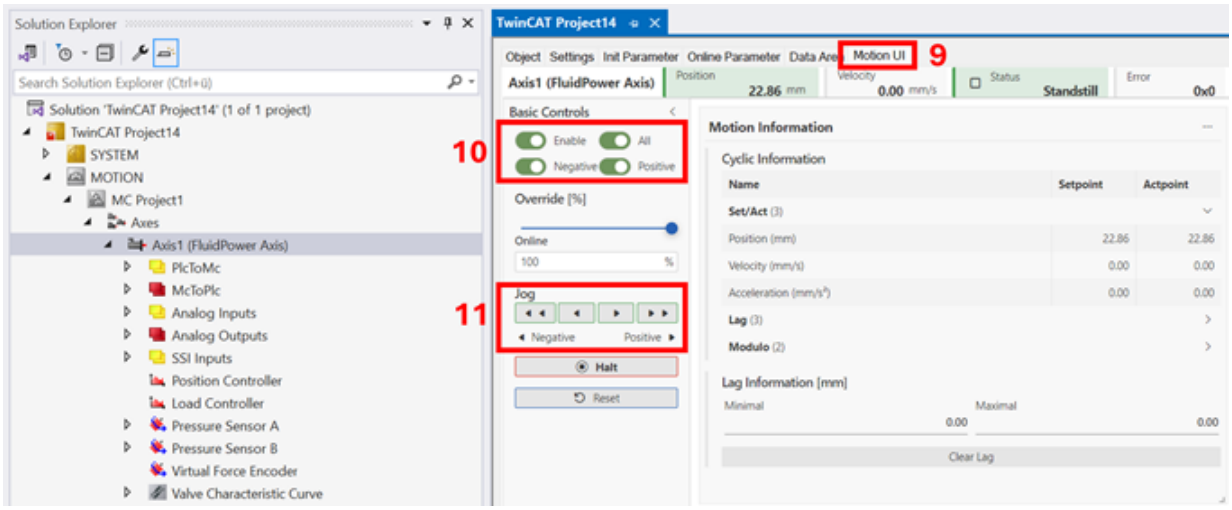
6. Double-click the **Axis** object.
7. Open the **Init Parameter** tab.

8. Reduce the Jog Velocity Slow setting in the **Manual Motion** group.



Move in manual operation

9. Click **Motion UI**.
10. Click **Enable All** to power the axis.
11. Move the axis by clicking on **Jog Slow Positive / Negative.**



→ Check the counting direction of the position measurement system and the valve control based on the triggered movement.

→ If the position measurement system malfunctions, check the wiring or set the inversion (see Set the position measuring system [▶ 163]). Repeat the check.
→ If the valve is malfunctioning, check the wiring or set the required inversion (see Set the valve output [▶ 166]).
→ If the axis is drifting significantly, set a corrective offset for the output signal (see Set the valve output [▶ 166]).

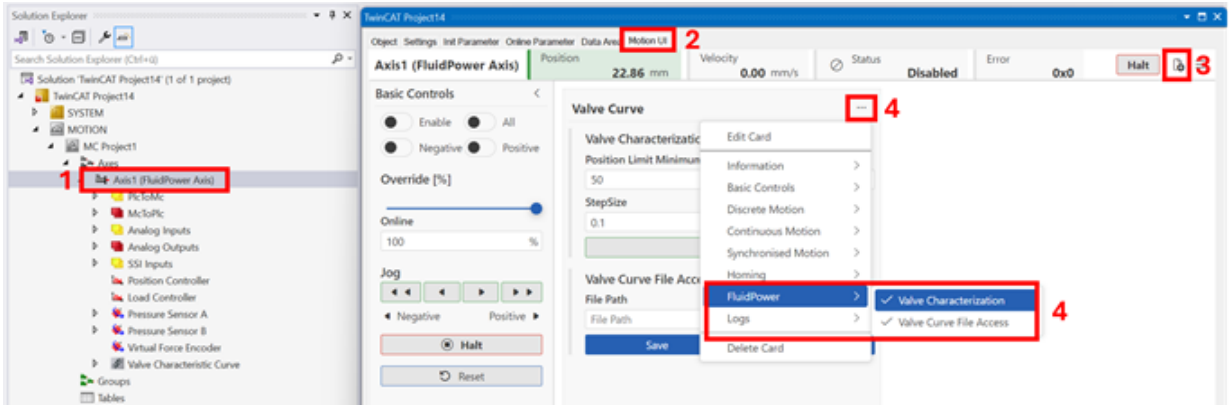
The next step is to identify the valve characteristics;
see Measuring the valve characteristics [▶ 170].

3.2.5.8.8 Measuring the valve characteristics

Add a FluidPower card

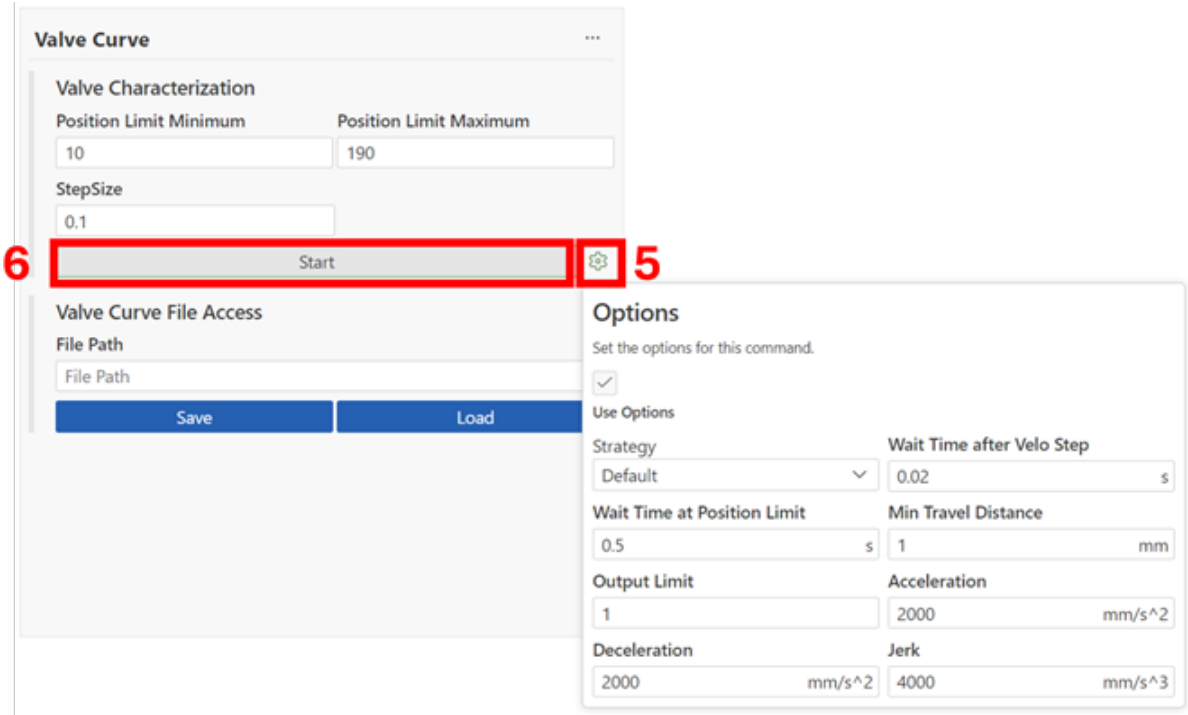
✓ The MC3 FluidPower axis was moved during manual operation; see Move the axis in manual operation using Motion UI [▶ 168].

1. Double-click the **FluidPower Axis** object
2. Opening the **Motion UI** tab
3. Add a card (see Cards [▶ 38])
4. Add the **FluidPower → Valve Characterization** and **FluidPower → Valve Curve File Access** options to the new card (see Motion UI [▶ 154])

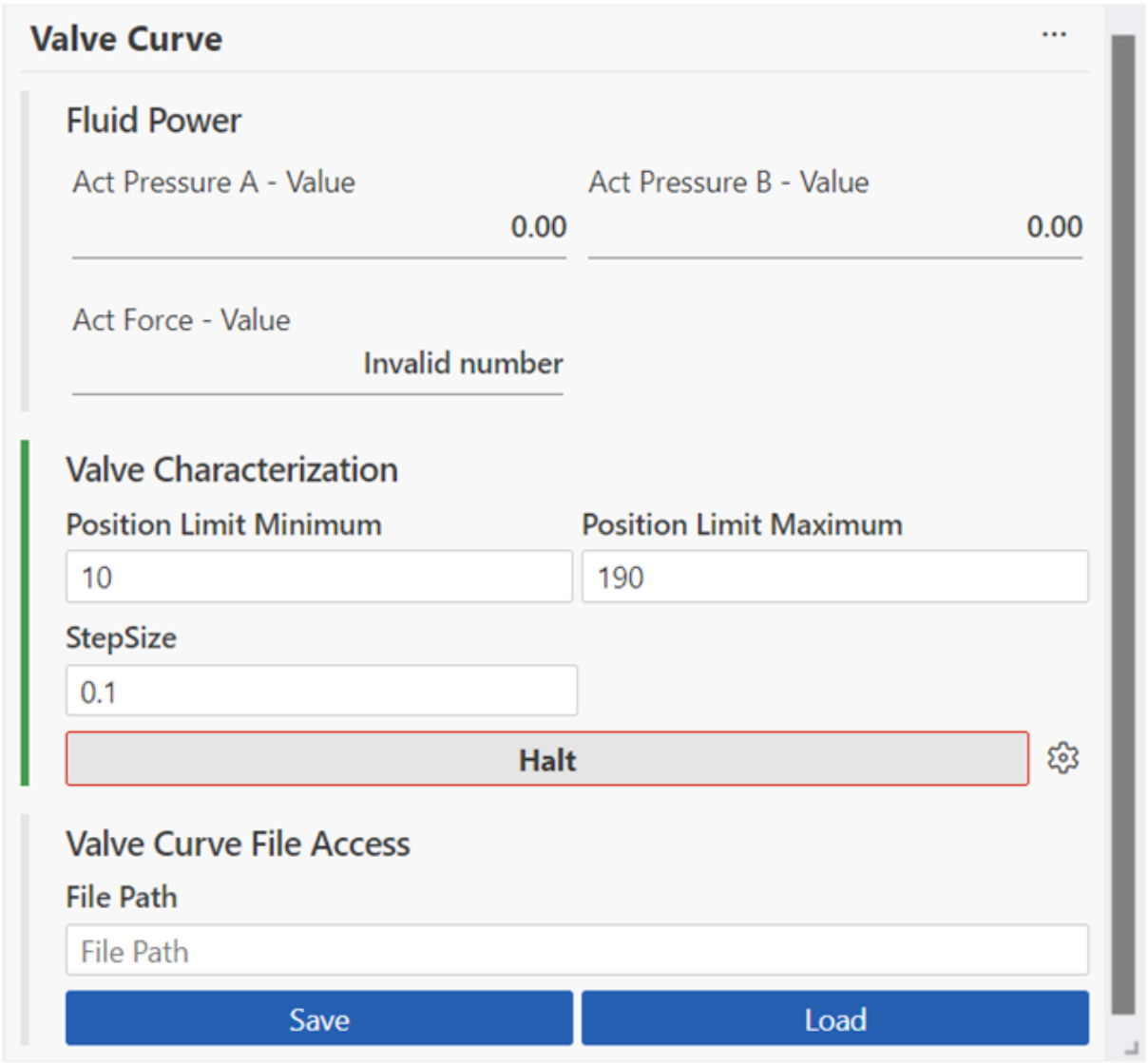


Starting the identification

5. Set the appropriate **limits** for your hydraulic system (see Identification of the valve characteristics [▶ 154])
6. Start identifying the valve characteristics by clicking **Start**



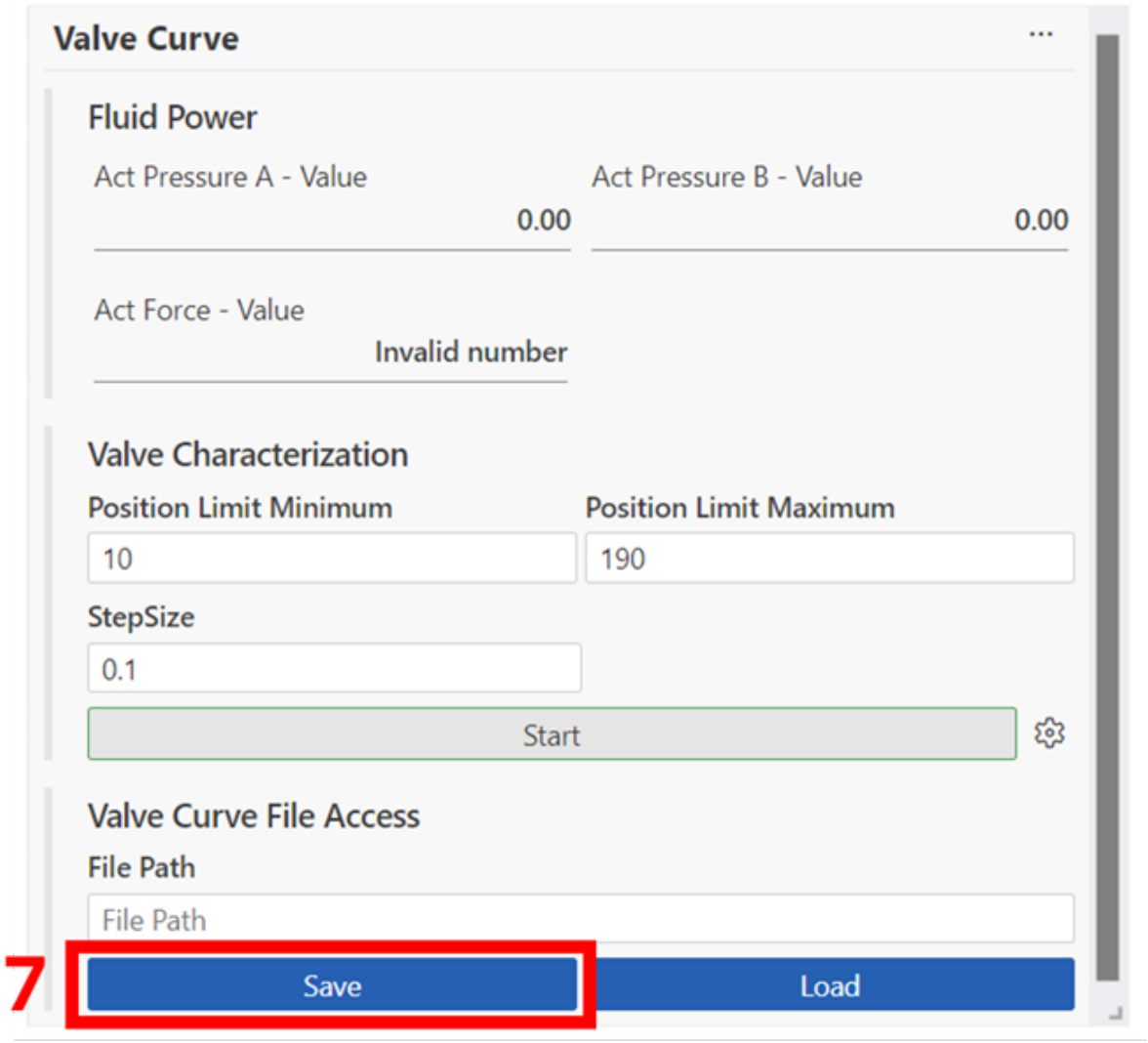
→ The “Start” button changes to “Stop”



→ Once the valve characteristics have been successfully identified, the button changes back to “Start”.

Save Valve Characteristics

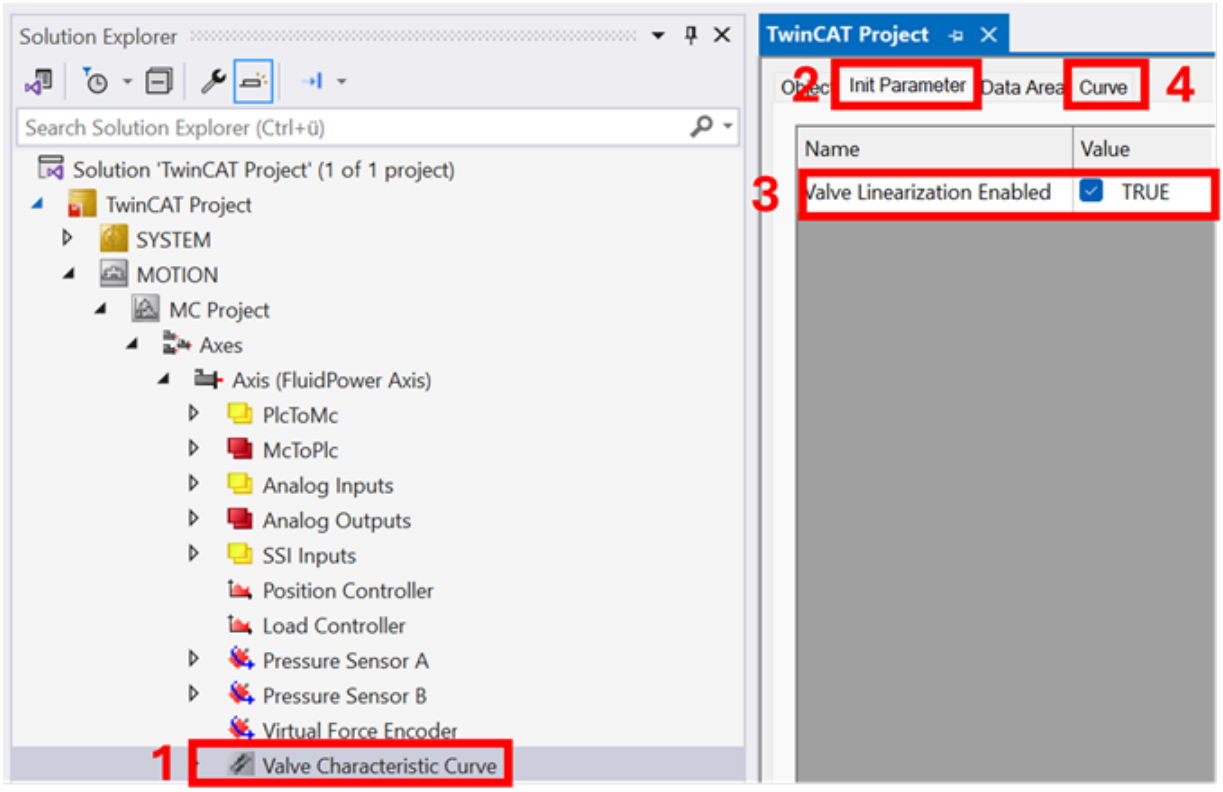
7. Click Save to save the characteristic curve on the target device (see File access [▶ 155])



Enable Valve Characteristics

8. Double-click the **Valve Characteristic Curve** object
9. Click on **Init Parameter**
10. Activate the valve characteristics

11. Click on **Curve** to view and adjust the characteristic curve.



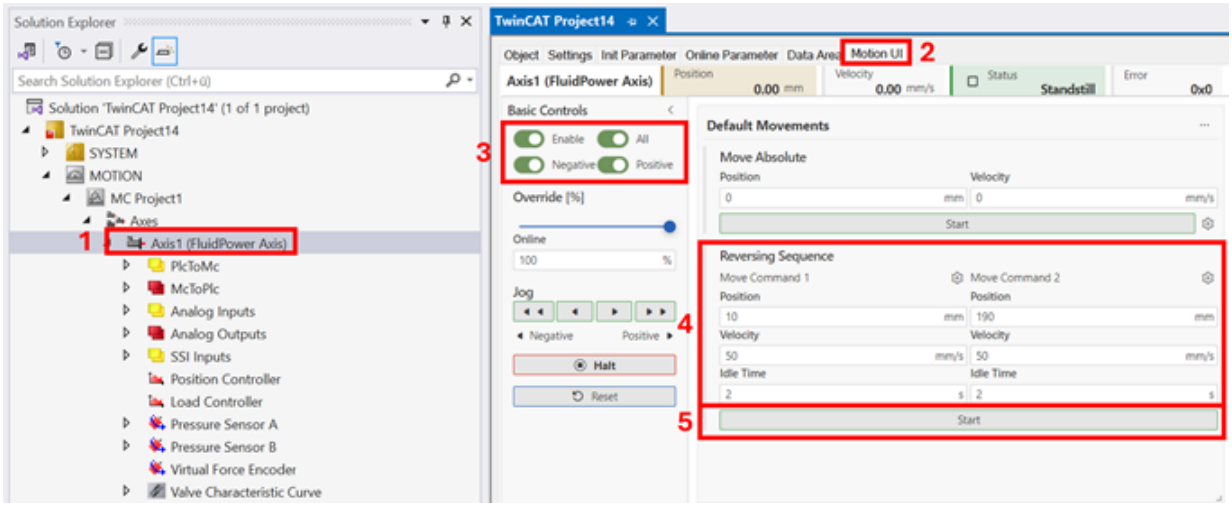
The next step is to optimize the position controller;
see Set control parameters for position control [▶ 173].

3.2.5.8.9 Set control parameters for position control

Start test movement

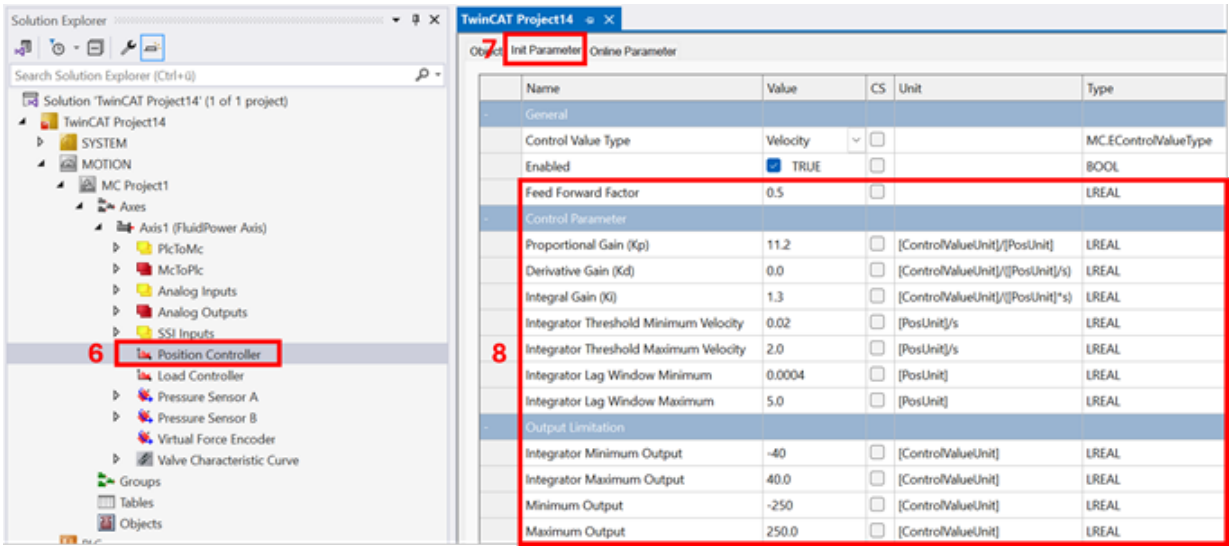
✓ The MC3 FluidPower axis was moved during manual operation;
see Move the axis in manual operation using Motion UI [▶ 168].

1. Double-click the **Axis** object
2. Opening the **Motion UI** tab
3. Apply power to the axis by clicking **Enable All**
4. In the **Default Movements** card, set the reverse operation parameters to match your motion profile.
5. **Start** of the movement



Set the controller

6. Double-click the **Position Controller** object
7. Open the **Init Parameter** tab
8. Optimization of Control Parameters (see also Init Parameter [▶ 96])



- If necessary, increase any previously set limits on the control signal **Minimum/Maximum Output** to values permitted for the hydraulic system.
- Set **Integral Gain** and **Derivative Gain** to 0.0
- Increase the **Proportional Gain** until the output signal begins to oscillate
- Slight reduction of the **Proportional Gain**
- If necessary, adjust the **Feed Forward Factor** to bring the actual velocity closer to the target velocity
- Set **Integral Gain** to reduce the residual error at the target position
- Limiting the velocity windows: **Integrator Threshold, Minimum/Maximum Velocity** of the I-component to prevent over-integration during motion and the resulting overshoot.
- Limiting the **Integrator Lag Window Minimum / Maximum** of the I-component to prevent residual oscillation at the target position.
  The recommended value for the **Integrator Lag Window Minimum** is 4 times the resolution of the position measurement system.
  The **Integrator Lag Window Maximum** is related to the residual position lag in pure P-control.

The next step is the integration of the FluidPower axis with the PLC;
see Link axis objects to a PLC [▶ 174]

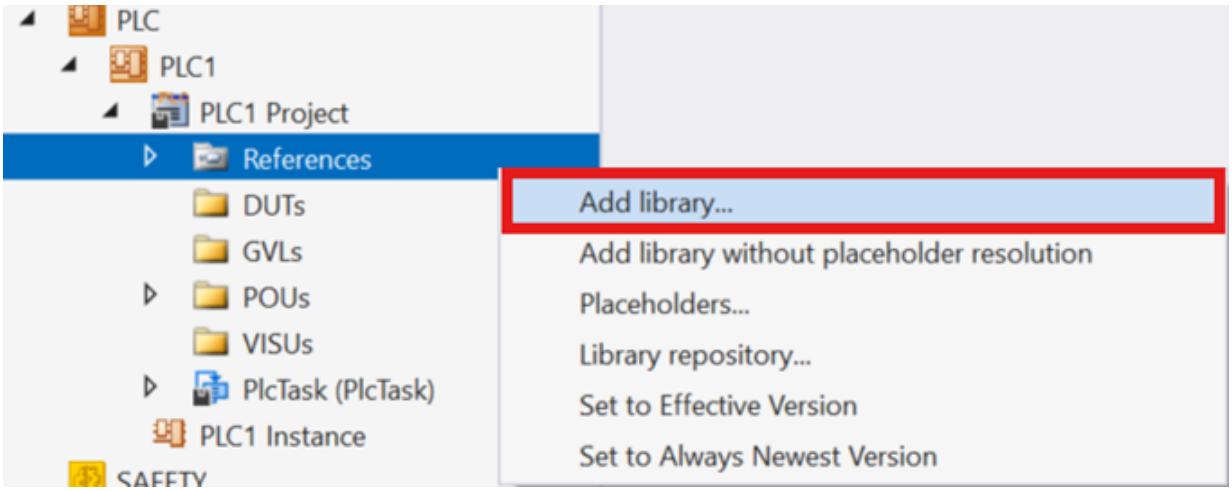
3.2.5.8.10 Link axis objects to a PLC

Adding the `Tc3_Mc3Ptp` PLC library

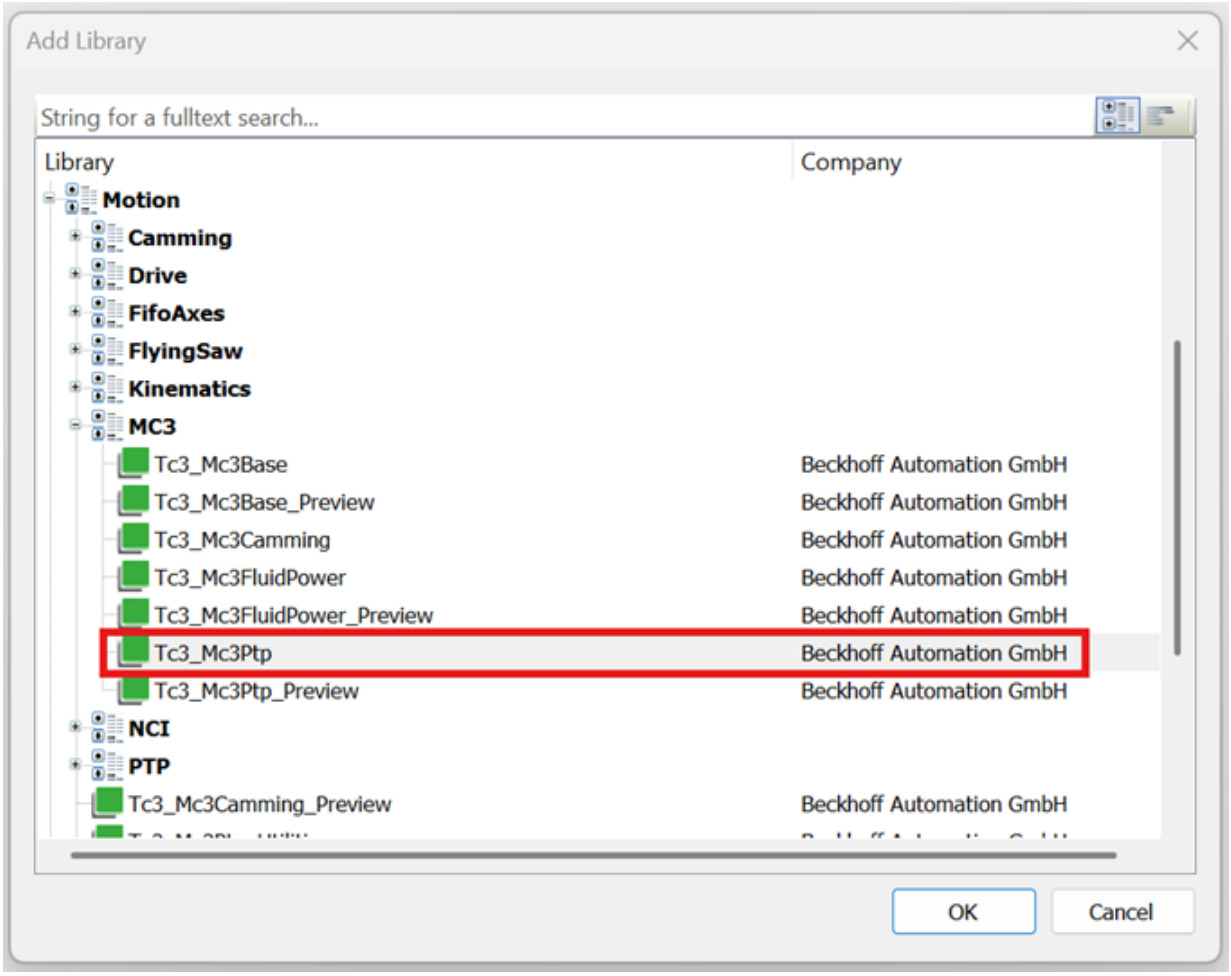
✓ A FluidPower axis object has been created and the position controller has been optimized;
see Set control parameters for position control [▶ 173]

✓ A PLC project has been created;
see Creating the project and selecting the PLC device

1. Right-click the References node > **Add library...**

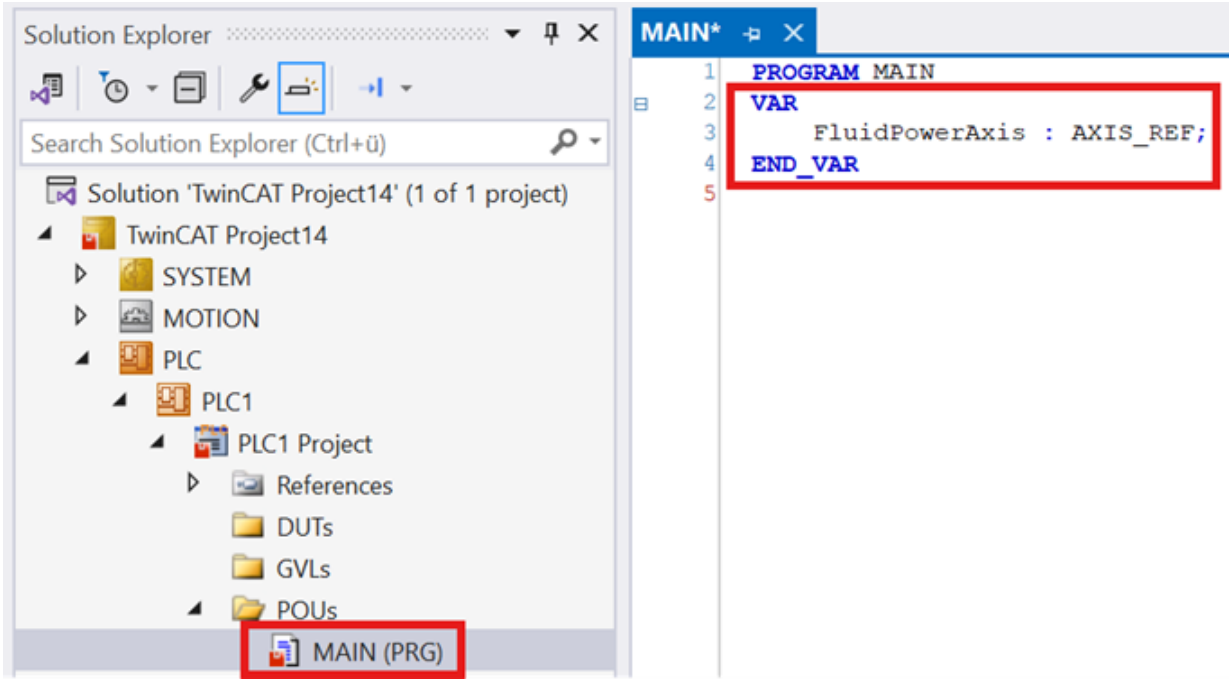


2. In the **Add Library** dialog under **Motion→MC3**, make the choice of the **Tc3_Mc3Ptp** library and click **OK**.

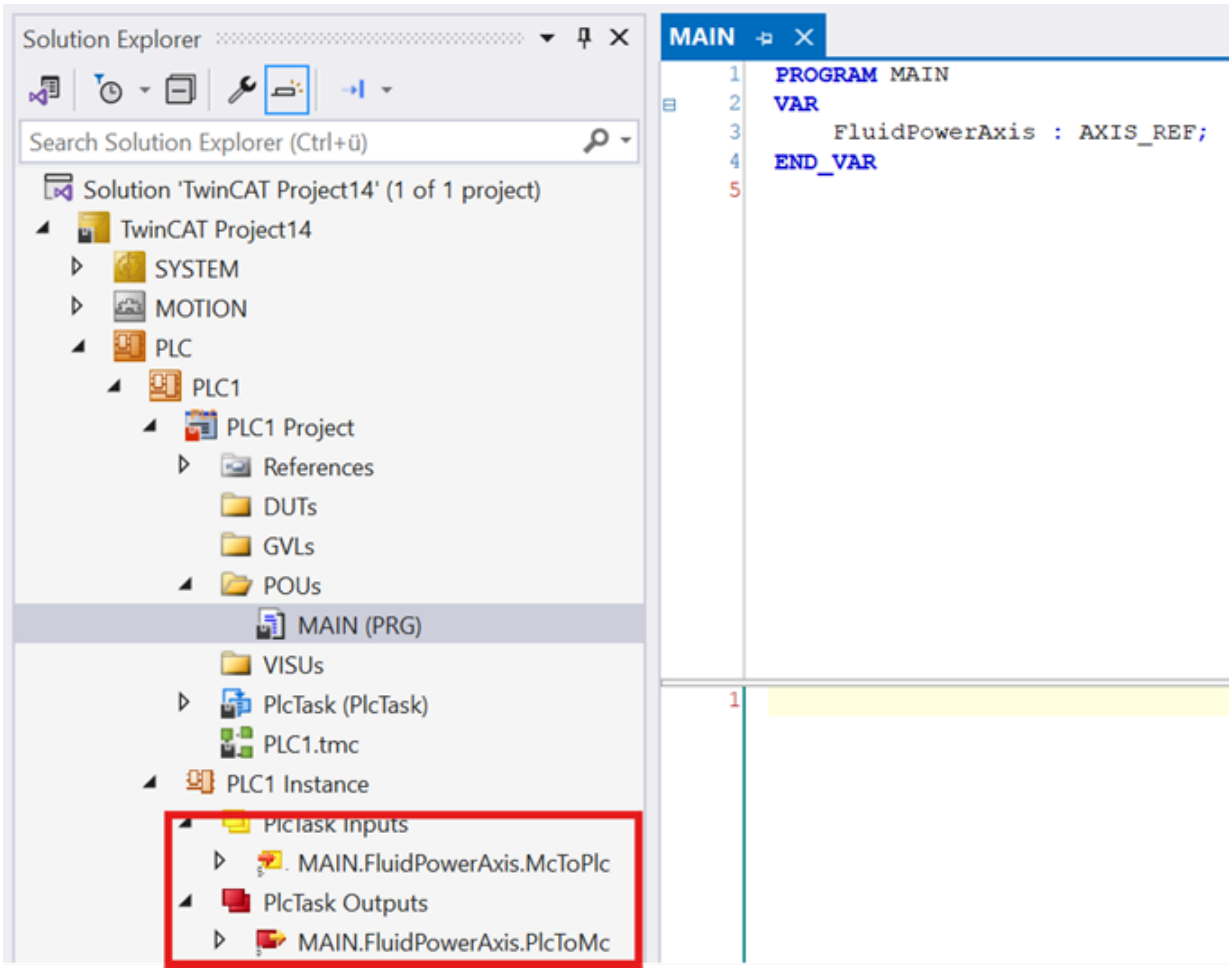


Create an instance of `AXIS_REF`

3. Double-click on **MAIN** and create an instance of **Tc3_Mc3Ptp.AXIS_REF**

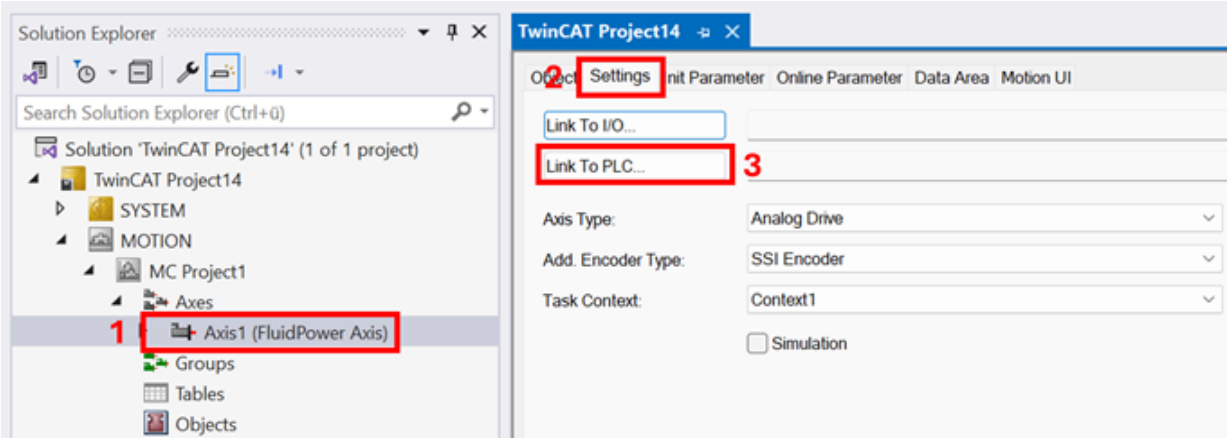


4. Create the PLC project
   → The instance of `Tc3_Mc3Ptp.AXIS_REF` will appear under PLC Instances in Solution Explorer.

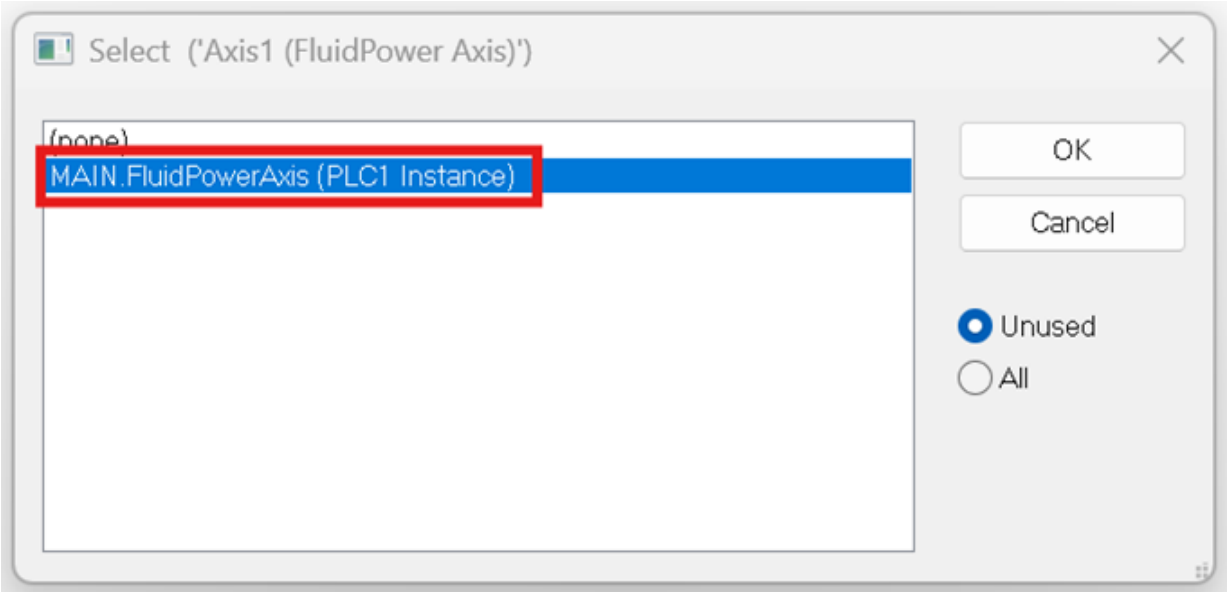


Connect the FluidPower axis to the PLC

5. Double-click the **FluidPower Axis** object
6. Opening the **Settings** tab
7. Click on the **Link to PLC link…**



8. In **Selection** dialog box, select the instance of **AXIS_REF** and click **OK** to confirm



Activating a TwinCAT project

1. Activate the TwinCAT project

The next step involves moving the FluidPower axis using a PLC;
see Programming a PLC using function blocks [▶ 177]

3.2.5.8.11 Programming a PLC using function blocks

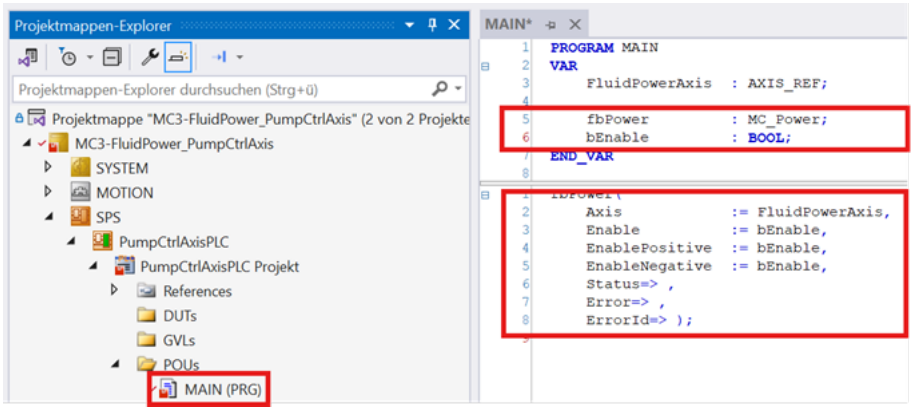
Enable the axis

✓ A FluidPower axis object has been created and the position controller has been optimized;
see Set control parameters for position control [▶ 173]

✓ The FluidPower axis object was linked to the PLC;
see Link axis objects to a PLC [▶ 174]

1. Double-click on **MAIN** and create an instance of **Tc3_Mc3Ptp.MC_Power**;
see MC_Power

2. Implementation of a cyclic call to the created instance

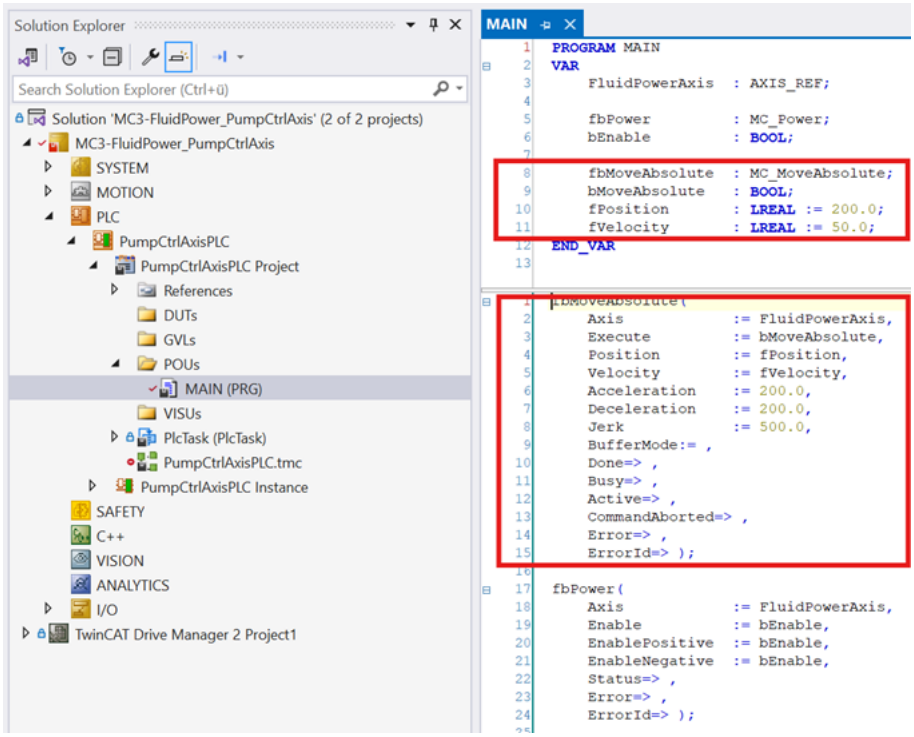



```
VAR
    FluidPowerAxis      : AXIS_REF;

    fbPower             : MC_Power;
    bEnable             : BOOL;
END_VAR
```

```
fbPower(
    Axis               := FluidPowerAxis,
    Enable             := bEnable,
    EnablePositive     := bEnable,
    EnableNegative     := bEnable,
    Status=> ,
    Error=> ,
    ErrorId=> );
```

Absolute positioning

3. Creating an instance of **Tc3_Mc3Ptp.MC_MoveAbsolute**; see MC_MoveAbsolute
4. Implementation of a cyclic call to the created instance



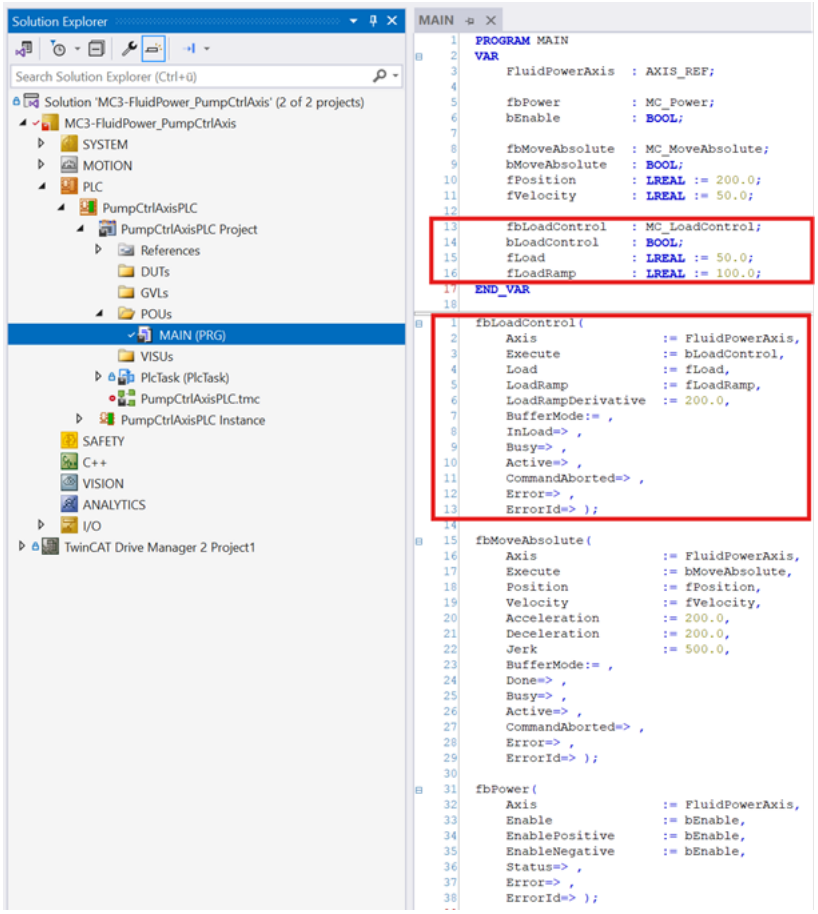
→ The axis can now be powered using the PLC and moved to its end position.

```
    fbMoveAbsolute      : MC_MoveAbsolute;
    bMoveAbsolute       : BOOL;
    fPosition           : LREAL := 200.0;
    fVelocity           : LREAL := 50.0;
```

```
fbMoveAbsolute(
    Axis              := FluidPowerAxis,
    Execute           := bMoveAbsolute,
    Position          := fPosition,
    Velocity          := fVelocity,
    Acceleration      := 200.0,
    Deceleration      := 200.0,
    Jerk              := 500.0,
    BufferMode:= ,
    Done=> ,
    Busy=> ,
    Active=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorId=> );
```

Pressure/force control

5. To create an instance of **Tc3_Mc3Ptp.MC_LoadControl**, see MC_LoadControl
6. Implementation of a cyclic call to the created instance



→ The pressure/force control of the axis can now be triggered via the PLC.

```
    fbLoadControl     : MC_LoadControl;
    bLoadControl      : BOOL;
    fLoad             : LREAL := 50.0;
    fLoadRamp         : LREAL := 100.0;
```

```
fbLoadControl(
    Axis                 := FluidPowerAxis,
    Execute              := bLoadControl,
    Load                 := fLoad,
    LoadRamp             := fLoadRamp,
    LoadRampDerivative   := 200.0,
    BufferMode:= ,
    InLoad=> ,
    Busy=> ,
    Active=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorId=> );
```

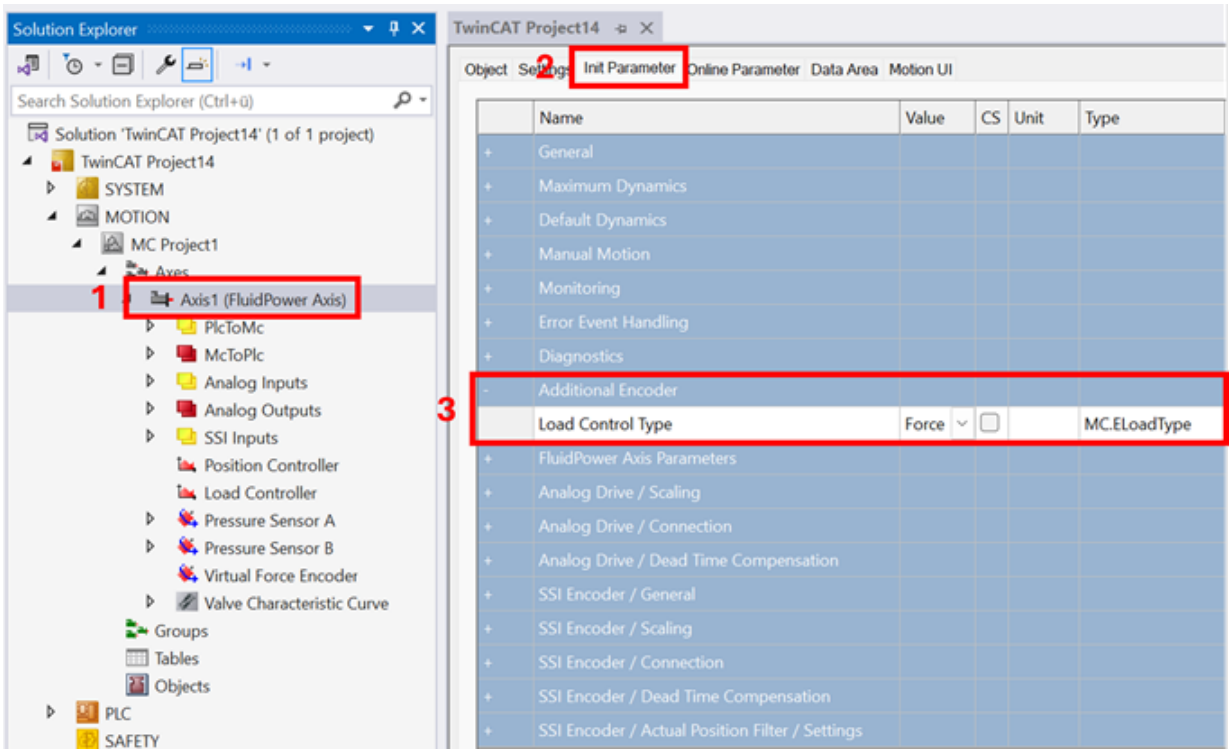
The next step is to optimize the pressure controller;
see Set the pressure controller [▶ 180]

3.2.5.8.12 Set the pressure controller

Select load type

✓ Pressure/force control can be activated via the PLC;
see Programming a PLC using function blocks [▶ 177]

1. Double-click the **Axis** object (1).
2. Open the **Init Parameter** (2) tab.
3. Select **Load Control Type** (3) in the **Additional Encoder** group.



For explanations of the parameters, see Values.

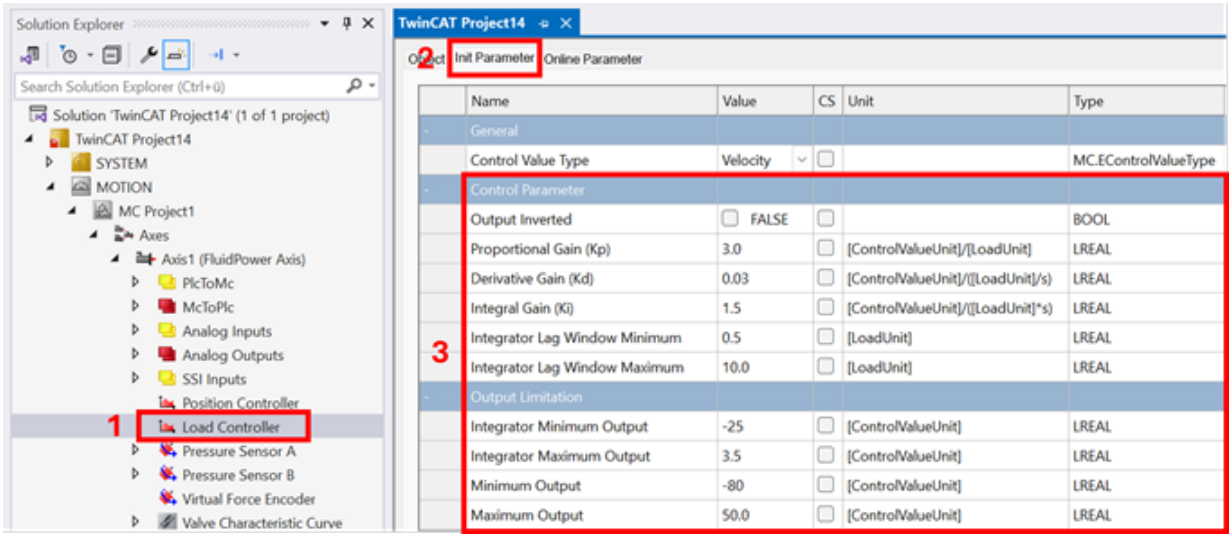
Start pressure/force control via PLC

1. Enable the axis.
2. Move the axis to its end position or until it reaches the point of contact with the opposing force.
3. Start the pressure/force control.

Set the controller

4. Double-click the **Load Controller** object (1).
5. Open the **Init Parameter** (2) tab.

6. Optimize the control parameters (see also Init Parameter [▶ 93]).



Possible measures:

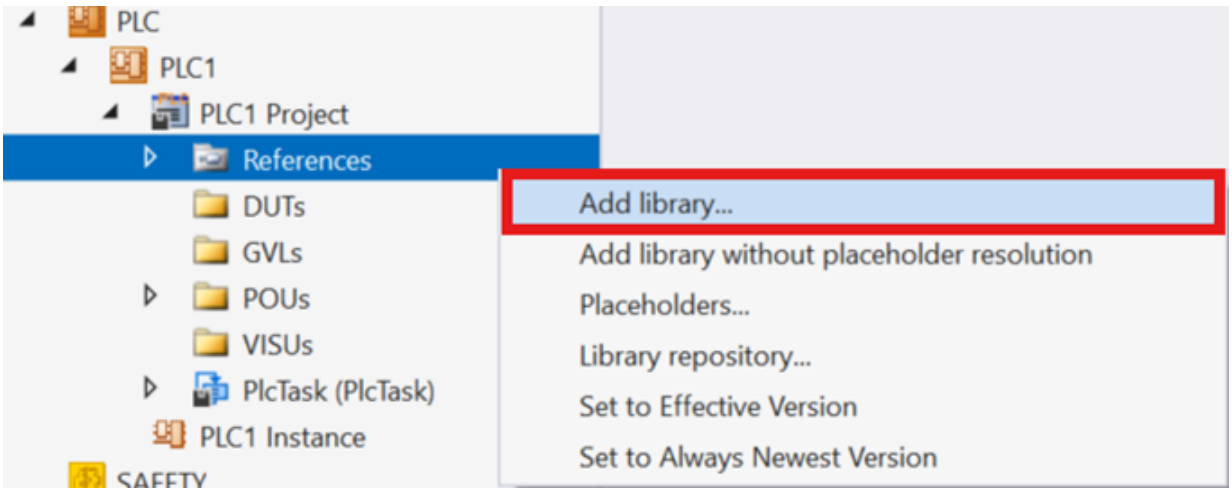
- If necessary, invert the control signal using **Output Inverted**.
- Set **Integral Gain** and **Derivative Gain** to 0.0
- Increase the **Proportional Gain** until the output signal begins to oscillate
- Slight reduction of the **Proportional Gain**
- If necessary, adjust the **Derivative Gain** to attenuate the control signal during setpoint jumps
- Set **Integral Gain** to reduce the residual error in the target load
- Limiting the **Integrator Lag Window Minimum / Maximum** of the I-component to prevent residual oscillation when holding the target load.
  The recommended value for the **Integrator Lag Window Minimum** is 4 times the resolution of the pressure/force sensor.
  The **Integrator Lag Window Maximum** is related to the residual lag error in pure P-control.

3.2.5.9 Identify the valve characteristics via PLC

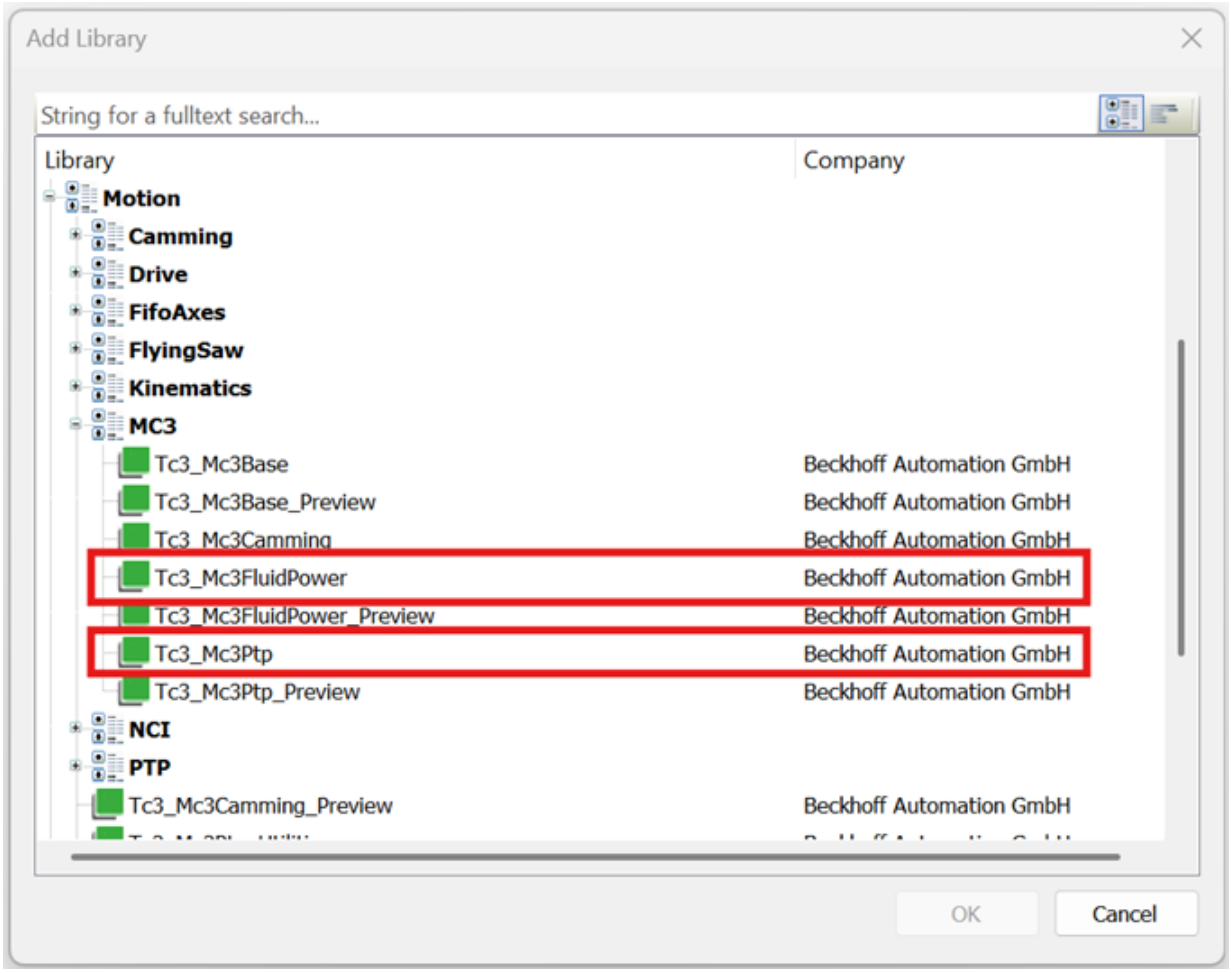
Adding the `Tc3_Mc3FluidPower` PLC library

✓ An MC3 FluidPower axis was moved during manual operation;
see Move the axis in manual operation using Motion UI [▶ 168]

1. Right-click the References node > **Add library...**



2. In the **Add Library** dialog under **Motion→MC3**, make the choice of the **Tc3_Mc3FluidPower** and **Tc3_Mc3Ptp** libraries and click **OK** to confirm.



PLC implementation

1. Create one instance each of **Tc3_Mc3FluidPower.MC_ValveCharacteristicCurve** (see MC_ValveCharacteristicCurve), **Tc3_Mc3FluidPower.MC_ValveCharacterization** (see MC_ValveCharacterization), **Tc3_Mc3FluidPower.ValveCharacterizationOptions** (see ValveCharacterizationOptions), and the desired number of characteristic curve points **numberOfValveCharacteristicCurvePoints**.
2. Make a connection between the reference **Tc3_Mc3FluidPower.MC_ValveCharacteristicCurve** and the Valve Characteristic Curve object of the FluidPower axis
3. Powering the FluidPower axis
4. Setting the options and starting the characterization of the valve characteristics

5. After successfully identifying the characteristic curve, activate the characteristic curve

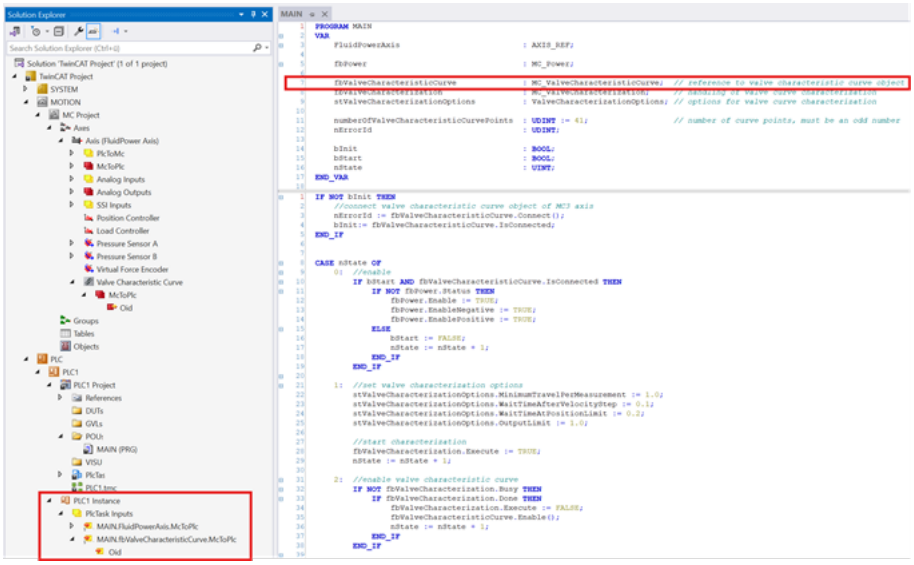

```
VAR
    FluidPowerAxis                          : AXIS_REF;

    fbPower                                 : MC_Power;

    fbValveCharacteristicCurve              : MC_ValveCharacteristicCurve;    // reference to
valve characteristic curve object
    fbValveCharacterization                 : MC_ValveCharacterization;       // handling of valve
curve characterization
    stValveCharacterizationOptions          : ValveCharacterizationOptions;   // options for valve
curve characterization

    numberOfValveCharacteristicCurvePoints  : UDINT := 41;                    // number of curve
points, must be an odd number
    nErrorId                                : UDINT;

    bInit                                   : BOOL;
    bStart                                  : BOOL;
    nState                                  : UINT;
END_VAR
```

```
IF NOT bInit THEN
    //connect valve characteristic curve object of MC3 axis
    nErrorId := fbValveCharacteristicCurve.Connect();
    bInit:= fbValveCharacteristicCurve.IsConnected;
END_IF
```

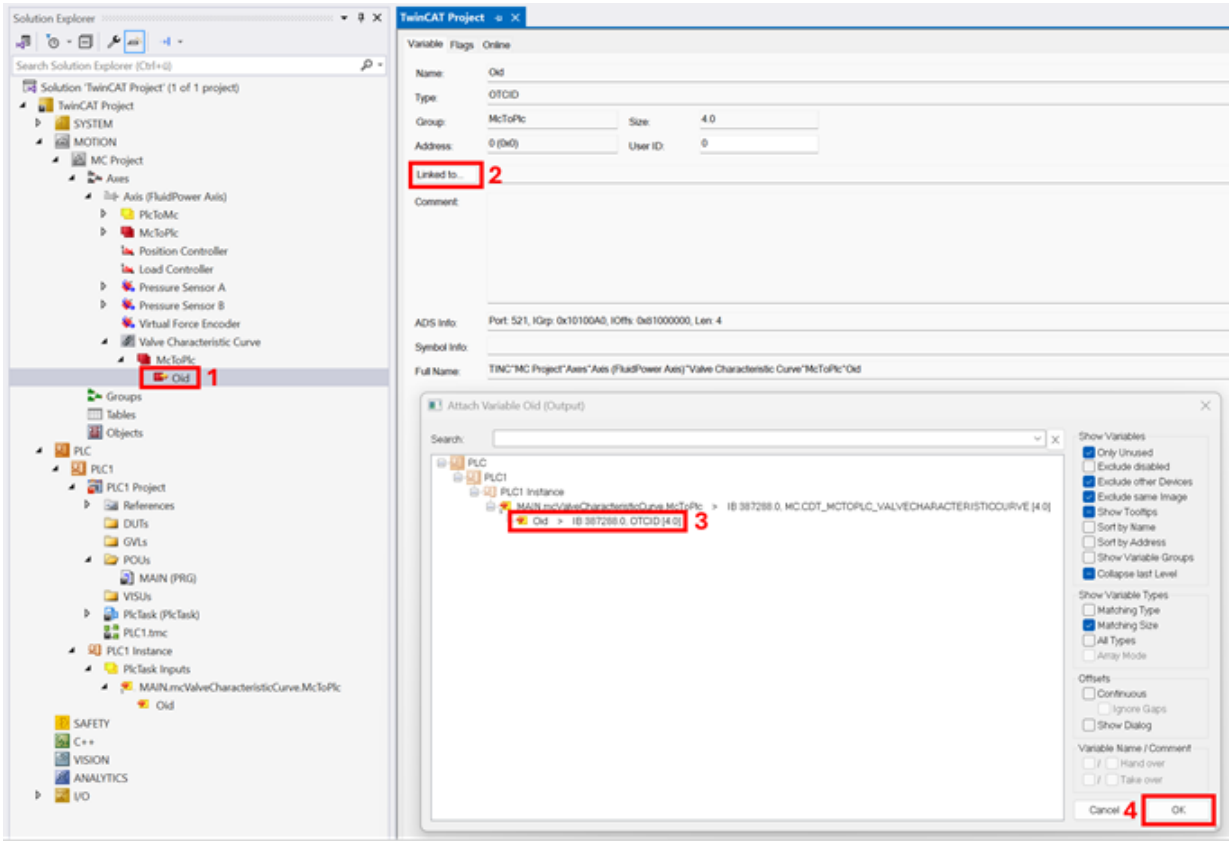
```
CASE nState OF
    0:  //enable
        IF bStart AND fbValveCharacteristicCurve.IsConnected THEN
            IF NOT fbPower.Status THEN
                fbPower.Enable := TRUE;
                fbPower.EnableNegative := TRUE;
                fbPower.EnablePositive := TRUE;
            ELSE
                bStart := FALSE;
                nState := nState + 1;
            END_IF
        END_IF

    1:  //set valve characterization options
        stValveCharacterizationOptions.MinimumTravelPerMeasurement := 1.0;
        stValveCharacterizationOptions.WaitTimeAfterVelocityStep := 0.1;
        stValveCharacterizationOptions.WaitTimeAtPositionLimit := 0.2;
        stValveCharacterizationOptions.OutputLimit := 1.0;

        //start characterization
        fbValveCharacterization.Execute := TRUE;
        nState := nState + 1;

    2:  //enable valve characteristic curve
        IF NOT fbValveCharacterization.Busy THEN
            IF fbValveCharacterization.Done THEN
                fbValveCharacterization.Execute := FALSE;
                fbValveCharacteristicCurve.Enable();
                nState := nState + 1;
            END_IF
        END_IF

    3:  //done
        nState := 0;
END_CASE


//call FBs
fbValveCharacterization(
    Axis                    := FluidPowerAxis,
    OutputStepsize          := 2.0 / TO_LREAL(numberOfValveCharacteristicCurvePoints - 1),
    PositionLimitMinimum    := 50.0,
    PositionLimitMaximum    := 450.0,
    Options                 := stValveCharacterizationOptions);

fbPower(Axis:= FluidPowerAxis);
```

Linking MC3 axis/PLC

3. Create the PLC project
   → The `Tc3_Mc3FluidPower.MC_ValveCharacteristicCurve` instance appears under PLC Instances in Solution Explorer.



4. Link the PLC instance to the Valve Characteristic Curve instance in MC3.



5. Activate the TwinCAT project.

3.2.6 Planar Mover

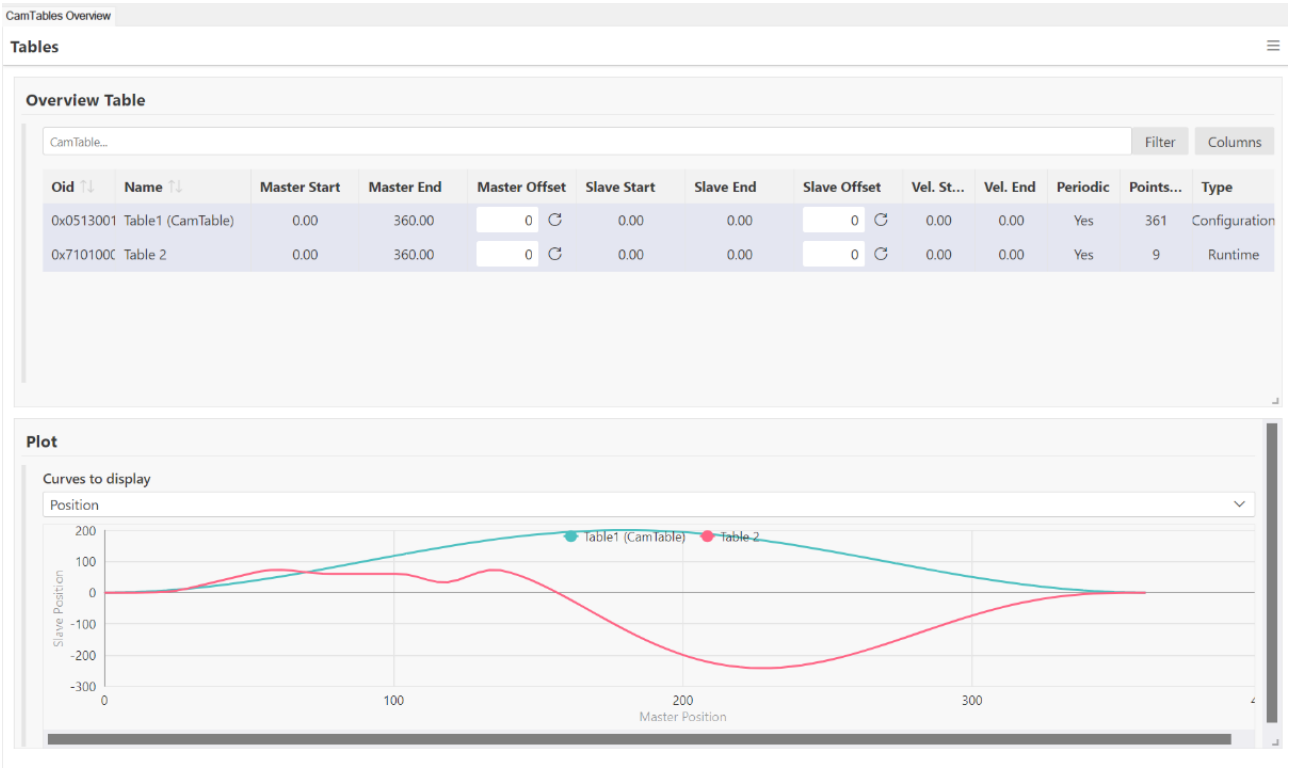
Special axis object for use with a Beckhoff XPlanar mover. For more details, please refer to the TF5430 | TwinCAT Planar Motion documentation.

3.2.7 Groups

Currently, this can only be used in connection with the XPlanar; see TF5430 | TwinCAT Planar Motion Documentation.

3.2.8 Tables

Under Tables, you'll find an overview of all available cam plates. If you click on a table in the upper part of the user interface to select it, its representation will be displayed in the lower half.



Filter

Use the filter to display the tables according to the following two options:

- Configuration CamTables,
- Runtime CamTables.

“Configuration CamTables” describes the cam plates that are created as cam objects in the system tree. “Runtime CamTables” are tables that are not created as objects in the system tree but are generated by the PLC at runtime.

Menu

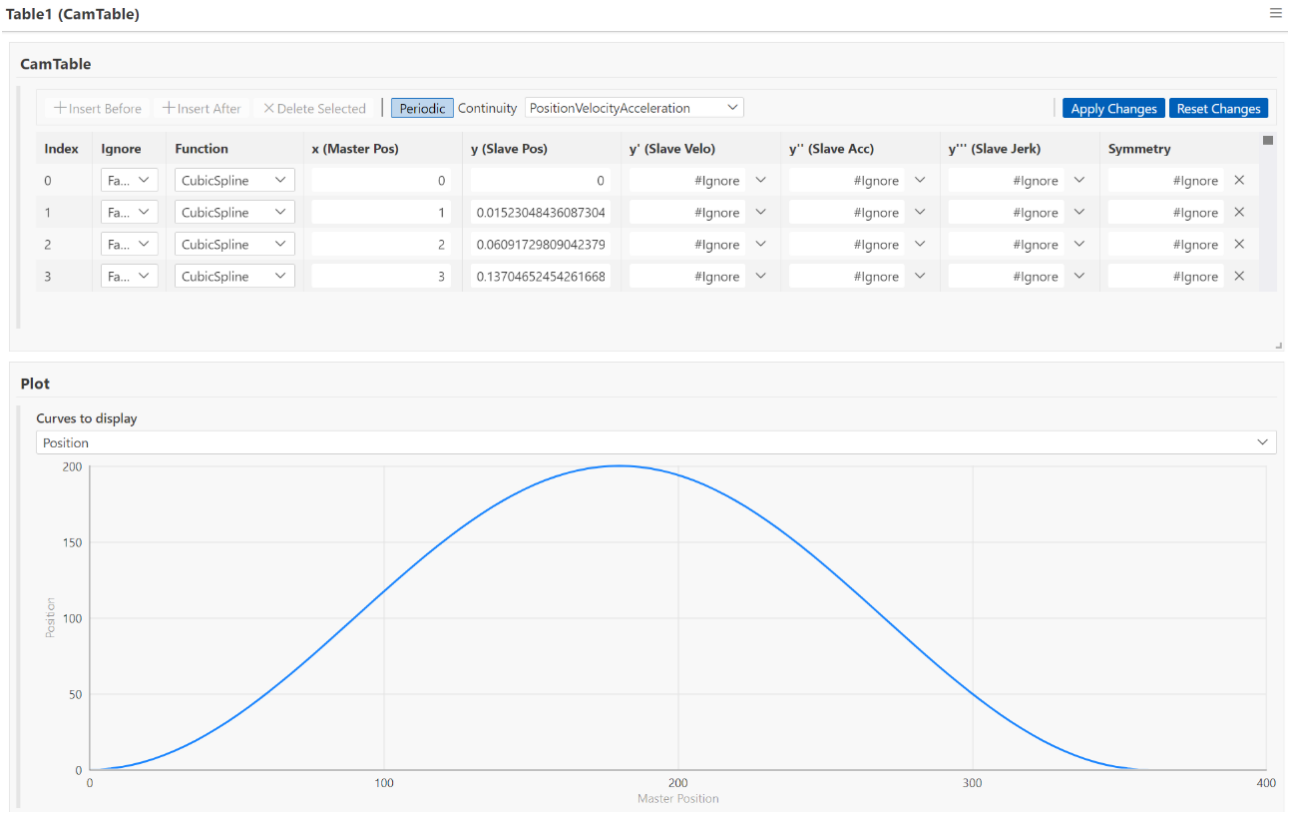
You can open the menu in the upper-right corner.

The menu offers various actions and settings:

Selection	Description
Reconnect	This reestablishes the connection between the UI and the object to be displayed and reloads the data.
Settings	This opens the settings.
Save Layout	Saves the current UI layout. You must enter a custom name. The saved layout can only be used in the current project.
Apply Layout	Applies an existing layout to the current view. The “default” layout is set by default. Additional layouts that have been saved using “Save Layout” are also available here.
Layout Manager	Opens the Layout Manager.
Theme	You can choose from the following themes for the color scheme: • System, • Light, • Dark.

3.2.8.1 Camming UI

The **Camming UI** tab is used to display and edit the selected cam plate.



CamTable

The “CamTable” card contains the individual support points for the current cam plate. Changes can be made to the table, which are represented directly on the “Plot” card. Click **Apply Changes** to save the changes. **Reset Changes**, on the other hand, resets any unsaved changes to their original values. If the “Periodic” checkbox is selected, the cam plate is treated as a periodic table.

Plot

The “Plot” card displays a graphical representation of the current cam plate. In addition to the position, checkboxes can be used to display the velocity and acceleration curves. You can also use the “Show Original” checkbox to display the original curves for position, velocity, and acceleration. The original curves are the ones that are currently active. In contrast, the curves shown are the fitted curves, which have not yet been saved.

Menu

You can open the menu in the upper-right corner.

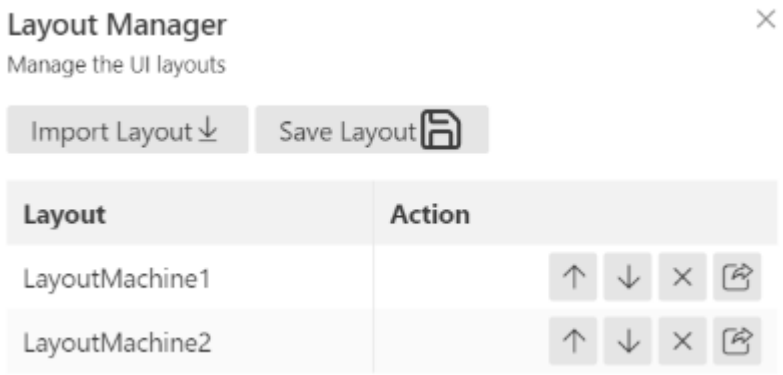
The menu offers various actions and settings:

Selection	Description
Reconnect	This reestablishes the connection between the UI and the object to be displayed and reloads the data.
Settings	This opens the settings.
Import	Opens the File Manager to import cam plates.
Export	Opens options for exporting the cam plate.

Selection	Description
Save Layout	Saves the current UI layout. You must enter a custom name. The saved layout can only be used in the current project.
Apply Layout	Applies an existing layout to the current view. The “default” layout is set by default. Additional layouts that have been saved using “Save Layout” are also available here.
Layout Manager	Opens the Layout Manager.
Reset Layout	Resets the UI so that the default layout is used again.
Theme	You can choose from the following themes for the color scheme: • System, • Light, • Dark.

Layout Manager

The Layout Manager displays your saved layouts and offers additional options.



Selection	Description
Import Layout	Imports an existing layout into the project.
Save Layout	Saves the current UI layout within the project. Another window will open so you can enter a name. If an existing layout name is used, the existing layout will be overwritten.
↑ ↓	Changes the order of your layouts. You can use the arrow keys to move a layout up or down in the list.
×	Deletes the corresponding custom layout.
↥	Export the layout you created so that it can be used in other projects as well. Next, you must specify the storage location.

Settings

Under “Settings”, you can configure various settings for the Camming UI. Clicking the **Save** button saves the settings you have configured within the project.

Common

Selection	Description
Number Of Digits	Number of decimal places to display.
XAR Interval [ms]	Specifies the interval at which data is read from the runtime environment.
XAE Interval [ms]	Specifies the interval at which data is read from the engineering environment.

Selection	Description
Feedback Duration [s]	Sets the duration for which the feedback window (pop-up) is displayed.
Severity of Feedback	Specifies from which level events are displayed in the feedback window.
Error Code Format	Options: Hex, Decimal Specifies the format in which the error code is displayed (as a hexadecimal or decimal number).
Show Full Name	If necessary, enlarges the axis name display so that it is shown in its entirety.

Polling State

The polling state displays data from the UI's last request to the Motion driver for diagnostic purposes.

Selection	Description
Polling State	Idle / Polling / No Connection / Error
Error message	Error message that appears if an error occurs in the communication between Motion UI and the driver.
Last Fetch Duration	Total duration of the UI's most recent request to the motion driver.
Last Bundled Response	Final response from the driver to the UI.

Storage unit

Selection	Description
Layouts and Inputs	Clicking **Clear** resets the layout to the default view and clears any entries made in the UI.
Settings	Clicking **Clear** clears the settings you have configured for this interface.

3.2.9 Objects

Currently, this can only be used in connection with the XPlanar; see TF5430 | TwinCAT Planar Motion Documentation.

3.2.10 Extensions

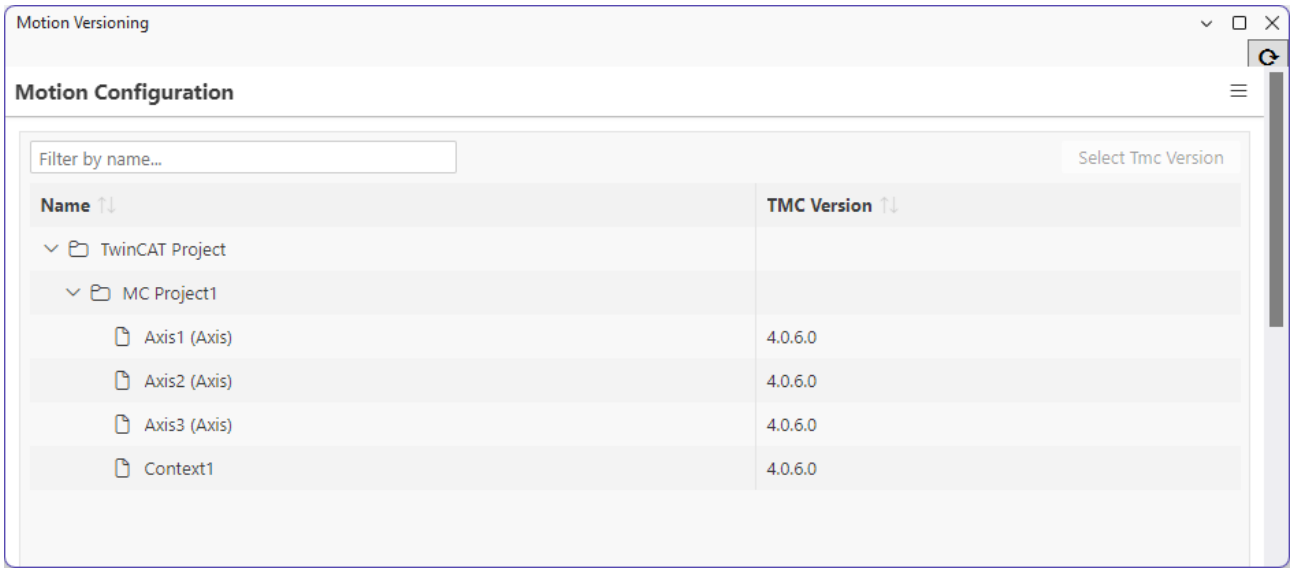
✓ Open motion control views from the menu bar.

1. **TwinCAT** > **MC3**
2. Make a choice of the desired view.

⇨ The following views are available.

Page	Description
Motion Versioning	Managing the TMC Versions of MC3 motion objects
Motion Axis View	Detailed view and control of a single axis
Motion Axes List View	List of all axes
Motion Cam Table View	Displaying the cam plate tables

Motion Versioning



The **Motion Configuration** view is used to manage and provide an overview of all motion projects and their components within a TwinCAT project. It displays the hierarchical structure of the configured axes, cam plates, and contexts, and allows you to manage the associated TMC versions.

There are two controls:

- **Select TMC Version**: Button in the upper-right corner to select or change the version of the selected items. It becomes active when one or more items are selected in the tree structure.
- **Filter by name**: Text field in the upper-left corner for filtering the tree structure by name. Entering a search term hides all nonmatching entries.

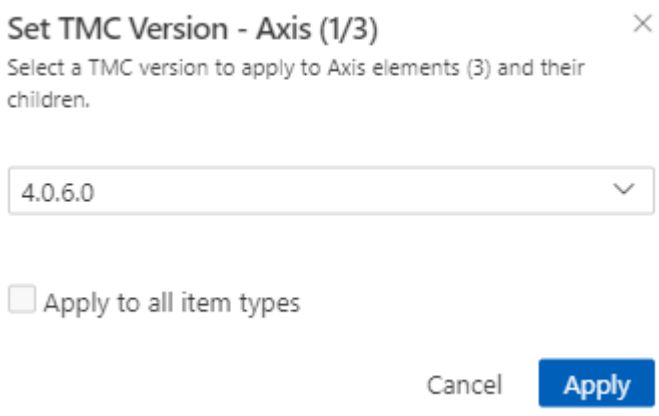
Tree structure (project tree)

The project tree, along with all associated motion objects, is displayed in a table.

Column	Description
Name	Hierarchical name of the element (sortable)
TMC version	Assigned version of the element (sortable) The **TMC Version** column shows the version of the TwinCAT Motion component used to configure the respective element. You can change the version using **Select TMC Version** if necessary, for example after an update.

Procedure for changing the version

Option 1 (recommended approach): Changing the versions for all components



To update the version of all child components at once, select the top-level element in the hierarchy (e.g., MC Project). Next, use the **Select TMC Version** option in the pop-up window to select the desired target version for each component.

If the **Apply to all item types** checkbox is selected, the new version is automatically applied to all subordinate elements (axes, cam plates, contexts) when you click **Apply** to confirm. This is particularly advantageous because not every component has to be reconfigured individually. This makes it quick and easy to ensure that all components of a project are consistently updated to the same version, especially after an update.

If the **Apply to all item types** checkbox is not selected, the version number must be chosen and confirmed for each object of the same type (e.g., for all axes).

Option 2: Changing the versions for individual components

It is technically possible to change the version for individual components separately. To do this, select the desired element (e.g., a single axis) in the tree structure and use **Select TMC Version** to assign a new version to only that element.

Preventing project startup: Avoiding multiple versions within an MC project

Using different versions within an MC project - that is, different versions for axes, cam plates, and contexts - is expressly not recommended. All components of an MC project should always use the same version. Different versions may prevent the project from starting.

Motion Axis View

This view essentially corresponds to the view in the Motion UI tab [▶ 33] tab of the individual axis. Advantage: This page can be opened multiple times as an extension, allowing different individual axes to be viewed and controlled simultaneously in separate windows.

Motion Axes List View

This view essentially corresponds to the axis overview Axes [▶ 22] in a Motion project. Advantage: This page can be opened multiple times as an extension, allowing you to view and operate different axis overviews simultaneously.

Motion Cam Table View

This view essentially corresponds to the view in the Camming UI [▶ 187] tab of the cam table. Advantage: This page can be opened multiple times as an extension, allowing you to view different cam plates side by side in separate windows.

3.3 Appendix

Important links

- Display MOTION nodes [▶ 16]

MC Project

- Creating an MC project [▶ 17]
- Create a new context [▶ 20]
- Assign axes to a context [▶ 21]

Axis

- Create an axis object [▶ 25]
- Expand the axis object with an additional module [▶ 116]
- Link a PLC to an axis object [▶ 117]
- Link an axis object to hardware [▶ 121]

- Version control [▶ 14]

4 Global data types and MC3 types

Cyclic Data Types (CDTs)

CDTs define structures for the cyclic exchange between MC3 and PLC. The Data Area [▶ 32] tab for each axis specifies which CDTs are mapped to the PLC for that axis and are therefore cyclically populated with data.

If there is a link between the MC3 axis and the PLC Tc3_Mc3PTP.AXIS_REF, then CDT_PLCTOMC_AXIS_STD and CDT_MCTOPLC_AXIS_STD are always linked. All other CDTs are optional. For optional CDTs, the `IsConnected` must be checked before any read or write operation. Only when `IsConnected = TRUE` may the PLC access the remaining structure.

NOTICE

Unlinked CDTs can cause a floating-point exception when accessed

Comparisons with NaN values can lead to an exception, which results in the runtime stopping and potentially causing machine damage.
Valid data can only be expected if `IsConnected = TRUE`. Otherwise, there is a risk of an inconsistent or invalid data set, which may lead to unpredictable consequences and unexpected behavior. Arithmetic operations using invalid values can cause floating-point exceptions in the processor's floating-point unit.

- Switch off the Floating Point Exceptions if NaN values are to be used and processed in the application.
- Before accessing a CDT (e.g., from the PLC), check the CDT's `IsConnected`.
- Familiarize yourself with how to handle named constants and NaN values.

4.1 Data Types

4.1.1 Enums

4.1.1.1 EAxisLicenseSetting

Syntax

Definition:

```
TYPE EAxisLicenseSetting :
(
    AutoBeckhoff    := 1,
    AutoMultiVendor := 3,
    Beckhoff        := 5,
    MultiVendor     := 7
)UDINT;
END_TYPE
```

Values

Name	Description
AutoBeckhoff	
AutoMultiVendor	
Beckhoff	
MultiVendor	

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.2 EAxisState

Syntax

Definition:

```
TYPE EAxisState :
(
    Invalid            := 0,
    Disabled           := 1,
    Standstill         := 2,
    Homing             := 3,
    ErrorStop          := 4,
    Stopping           := 5,
    DiscreteMotion     := 6,
    ContinuousMotion   := 7,
    SynchronizedMotion := 8,
    ResetPending       := 10,
    Enabling           := 11,
    Disabling          := 12,
    NotReady           := 13
)UINT;
END_TYPE
```

Values

Name	Description
Invalid	Axis object is not initialized or in an undefined state.
Disabled	Axis is disabled and the drive power stage is off.
Standstill	Axis is enabled and at standstill, ready to accept motion commands.
Homing	Axis is performing a homing procedure to establish a reference position.
ErrorStop	Axis has stopped due to an error. Error must be reset before further motion.
Stopping	Axis is decelerating to standstill during a stop command and remains in this state until Execute for stop command is released.
DiscreteMotion	Axis is executing a point-to-point motion command with a defined target position.
ContinuousMotion	Axis is executing a continuous velocity motion without a defined end position.
SynchronizedMotion	Axis is coupled to a master axis and following synchronized setpoints (e.g. gearing, camming).
ResetPending	An error reset has been requested and is being processed.
Enabling	Axis is in the process of being enabled (drive power stage is starting up).
Disabling	Axis is in the process of being disabled (drive power stage is shutting down).
NotReady	Axis is not ready for operation, e.g. waiting for required sub-modules to initialize or to start up.

Meaning of the states

Name	Description
Invalid	• Invalid axis state • The axis is not ready for operation. • A possible cause could be that AXIS_REF is not linked to an MC3 axis.
Disabled	• Basic state of an axis. • The axis is not released.

Name	Description
	• There is no axis error.
Standstill	• The axis is enabled. • No command has been issued to the axis.
Homing	• The axis is enabled. • The axis is homed (e.g., MC_Home).
ErrorStop	• An axis error has occurred. • A reset is required to exit this state.
Stopping	• The axis is enabled. • The axis is on a stop ramp (MC_Stop). • MC_Stop.Execute is set and blocks new motion commands.
DiscreteMotion	• The axis is enabled. • The axis performs a point-to-point movement (e.g., MC_MoveAbsolute, MC_MoveRelative, MC_MoveModulo, MC_Halt).
ContinuousMotion	• The axis is enabled. • The axis performs a continuous motion (e.g., MC_MoveVelocity).
SynchronizedMotion	• The axis is enabled. • The axis is linked to another axis (e.g., MC_CamIn, MC_GearIn, MC_GearInPos).
ResetPending	• Internal state
Enabling	• Extended axis status for representation of hardware conditions. • The command to enable the drive hardware's output stage has already been sent or is currently being sent, and the MC3 axis is waiting for the drive hardware to reach the “operational” state
Disabling	• Extended axis status for representation of hardware conditions. • The command to deactivate the drive hardware's output stage has already been sent or is currently being sent, and the MC3 axis is waiting for the drive hardware to enter the “not operational” state.
NotReady	• Extended axis status for representation of hardware conditions. • The axis is not ready for operation. • There is no active communication with the hardware (e.g., WcState), or the drive's state machine reports “not ready”.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.3 EContinuity

Syntax

Definition:

```
TYPE EContinuity :
(
    Position                        := 0,
    PositionVelocity                := 1,
    PositionVelocityAcceleration    := 2,
    PositionVelocityAccelerationJerk := 3
)UINT;
END_TYPE
```

Values

Name	Description
Position	Position profile is continuous.
PositionVelocity	Position and velocity profiles are continuous.
PositionVelocityAcceleration	Position, velocity and acceleration profiles are continuous.
PositionVelocityAccelerationJerk	Position, velocity, acceleration and jerk profiles are continuous.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.4 EControlErrorSetpointMode

Syntax

Definition:

```
TYPE EControlErrorSetpointMode :
(
    HardStop := 1
)UDINT;
END_TYPE
```

Values

Name	Description
HardStop	The dynamics of the axis are set to zero.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.5 EControlValueType

Syntax

Definition:

```
TYPE EControlValueType :
(
    Velocity := 0x1
)UDINT;
END_TYPE
```

Values

Name	Description
Velocity	Controller output is a velocity setpoint.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.6 ECylinderDesign

Syntax

Definition:

```
TYPE ECylinderDesign :
(
    Differential := 0,
    Synchronous  := 1,
    Plunger      := 2
)UINT;
END_TYPE
```

Values

Name	Description
Differential	Differential cylinder with different piston areas on extend and retract sides.
Synchronous	Synchronous cylinder with equal piston areas on both sides.
Plunger	Plunger cylinder with a single-acting piston (no rod-side area).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.7 EDeadTimeCompensationMode

Syntax

Definition:

```
TYPE EDeadTimeCompensationMode :
(
    None                     := 0,
    LastVelocity             := 1,
    LastVelocityAcceleration := 2,
    NonlinearShift           := 10
)UDINT;
END_TYPE
```

Values

Name	Description
None	No dead time compensation applied.
LastVelocity	Compensation uses the last known velocity to extrapolate the actual position.
LastVelocityAcceleration	Compensation uses the last known velocity and acceleration for more accurate extrapolation.
NonlinearShift	Compensation applies a non-linear time shift method for improved accuracy in dynamic motion. The history of the last actual values is taken into account.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.8 EEncoderDataRange

Syntax

Definition:

```
TYPE EEncoderDataRange :
(
    FullRange   := 1,
    UserDefined := 2
)UDINT;
END_TYPE
```

Values

Name	Description
FullRange	Position feedback overflow position based on CDT limits.
UserDefined	Manual entry of the overflow position.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.9 EEncoderFeedbackVeloSource

Syntax

Definition:

```
TYPE EEncoderFeedbackVeloSource :
(
    HardwareProvidedIfAvailable := 0x0,
    InternallyCalculated        := 1,
    HardwareProvided            := 2
)UDINT;
END_TYPE
```

Values

Name	Description
HardwareProvidedIfAvailable	Uses hardware-provided velocity feedback if available, otherwise falls back to internal calculation (default).
InternallyCalculated	Velocity is always calculated internally by differentiating the position feedback.
HardwareProvided	Velocity is always taken from hardware feedback. Requires hardware support.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.10 EEncoderInterpretation

Syntax

Definition:

```
TYPE EEncoderInterpretation :
(
    AbsoluteFullRange     := 0,
```

```
    AbsoluteUserDataRange := 2,
    Incremental           := 4,
    AbsoluteModulo        := 5
)UDINT;
END_TYPE
```

Values

Name	Description
AbsoluteFullRange	Absolute encoder using the full data range defined by the data type of mapping element (default).
AbsoluteUserDataRange	Absolute encoder with user-defined absolute data range and overflow position.
Incremental	Incremental encoder providing relative position changes only. For example, if position recovery after a power cycle is not possible. A homing sequence is necessary.
AbsoluteModulo	Absolute encoder interpreted as modulo position within the configured range if the modulo operating mode is set on drive/encoder device.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.11 EFilterType

Syntax

Definition:

```
TYPE EFilterType :
(
    None            := 0,
    PT1             := 1,
    MovingAverage   := 2,
    KalmanFilter3x1 := 3
)UDINT;
END_TYPE
```

Values

Name	Description
None	No filtering applied.
PT1	First-order low-pass filter.
MovingAverage	Moving average filter that smooths over a configurable number of samples.
KalmanFilter3x1	Kalman filter with 3-state model.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.12 EHomingObjectType

Syntax

Definition:

```
TYPE EHomingObjectType :
(
    Invalid   := 0,
    Cam       := 1,
    FixedStop := 2
)UDINT;
END_TYPE
```

Values

Name	Description
Invalid	No valid homing object configured.
Cam	Homing uses a reference cam or proximity switch to locate the home position.
FixedStop	Homing drives against a mechanical fixed stop to determine the home position.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.13 EHomingSyncSource

Homing Sync Source

Syntax

Definition:

```
TYPE EHomingSyncSource :
(
    HardwareLatch1                := 0,
    ZeroImpulseCoE                := 1,
    HardwareLatch1CoEDriveDefined := 2,
    SoftwareSync                  := 3
)UDINT;
END_TYPE
```

Values

Name	Description
HardwareLatch1	Synchronization to reference position via hardware latch 1 input on the EtherCAT slave.
ZeroImpulseCoE	Synchronization to reference position via encoder zero impulse (index signal) through CoE drive interface.
HardwareLatch1CoEDriveDefined	Synchronization to reference position via hardware latch 1 with drive-defined CoE configuration.
SoftwareSync	Software-based singleturn zero crossing detection for homing synchronization.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.14 ELoadType

Indicates the type of the currently selected load of the axis.

Syntax

Definition:

```
TYPE ELoadType :
(
    None                 := 0,
    Force                := 2,
    PressureA            := 3,
    PressureB            := 4,
    DirectVelocityOutput := 5
)UDINT;
END_TYPE
```

Values

Name	Description
None	No load control active.
Force	Load control based on force.
PressureA	Load control based on pressure at area A of the cylinder.
PressureB	Load control based on pressure at area B of the cylinder.
DirectVelocityOutput	Velocity setpoint is written directly to the drive output.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.15 EMcTaskInternalSequence

Syntax

Definition:

```
TYPE EMcTaskInternalSequence :
(
    FastReaction := 0,
    Stable       := 1
)UDINT;
END_TYPE
```

Values

Name	Description
FastReaction	Execution order: Commands - Exec - Output. Processes new commands before output update, providing the fastest reaction to new commands.
Stable	Execution order: Output - Commands - Exec. Output update is executed first, then new commands are processed, providing less jitter for output update.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.16 EPositionLagSource

Syntax

Definition:

```
TYPE EPositionLagSource :
(
    HardwareProvidedIfAvailable := 0,
    InternallyCalculated        := 1,
    HardwareProvided            := 2
)UDINT;
END_TYPE
```

Values

Name	Description
HardwareProvidedIfAvailable	Uses hardware-provided position lag if available, otherwise falls back to internal calculation.
InternallyCalculated	Position lag is always calculated internally from position of setpoint and actual position.
HardwareProvided	Position lag is always taken from hardware. Requires hardware support and mapping.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.17 EPositionUnit

Syntax

Definition:

```
TYPE EPositionUnit :
(
    um  := 0,
    mm  := 1,
    m   := 2,
    in  := 3,
    deg := 11,
    rad := 12,
    gon := 13
)UDINT;
END_TYPE
```

Values

Name	Description
um	Micrometers
mm	Millimeters
m	Meters
in	Inches
deg	Degrees (360 per revolution)
rad	Radians (2*pi per revolution)
gon	Gradians (400 per revolution)

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.18 ETouchProbeTriggerEdge

Touchprobe Trigger Edge

Syntax

Definition:

```
TYPE ETouchProbeTriggerEdge :
(
    Positive := 1,
    Negative := 2
)UDINT;
END_TYPE
```

Values

Name	Description
Positive	Trigger on rising edge of the input signal.
Negative	Trigger on falling edge of the input signal.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.19 ETouchProbeTriggerSource

Selection for CoE Drive touch probe input

Syntax

Definition:

```
TYPE ETouchProbeTriggerSource :
(
    DriveDefault := 0,
    ZeroImpulse  := 1,
    DriveDefined := 2
)UDINT;
END_TYPE
```

Values

Name	Description
DriveDefault	Trigger on the touch probe default input (i.e., the digital input 1/2 mapped to TouchProbe1/ TouchProbe2 functionality in the drive).
ZeroImpulse	Trigger on the main encoder index signal (encoder Z/ index). Only valid if the drive exposes the index signal to the touch probe function.
DriveDefined	Trigger source is defined by object 0x60D0/0x8001 (DS402/MDP742).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.1.20 ETouchProbeTriggerState

Touchprobe Trigger State Machine

Syntax

Definition:

```
TYPE ETouchProbeTriggerState :
(
    Init       := 0,
    Armed      := 1,
    Latched    := 2,
    Continuous := 3,
    Aborted    := 4
)UDINT;
END_TYPE
```

Values

Name	Description
Init	Touch probe is initialized but not yet armed.
Armed	Touch probe is armed and waiting for a trigger event.
Latched	Touch probe has been triggered and the position has been latched.
Continuous	Touch probe is in continuous mode, latching positions on every trigger event.
Aborted	Touch probe operation was aborted before a trigger event occurred.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2 Structs

4.1.2.1 CDT_MCTOPLC_AXIS_ACT

CDT provides current act values.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_AXIS_ACT :
STRUCT
    IsConnected  : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    PositionLag  : LREAL;
    Torque       : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is beeing received from MC.
Position	LREAL	Current actual position of the axis.
Velocity	LREAL	Current actual velocity of the axis.
Acceleration	LREAL	Current actual acceleration of the axis.
PositionLag	LREAL	Difference between set position and actual position (position following error).
Torque	LREAL	Current actual torque of the axis.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.2 CDT_MCTOPLC_AXIS_FLUIDPOWER

CDT provides fluid power specific setpoint and actpoint.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_AXIS_FLUIDPOWER :
STRUCT
    IsConnected  : BOOL;
    ActPressureA : LoadData;
    ActPressureB : LoadData;
    ActForce     : LoadData;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is being received from MC.
ActPressureA	LoadData [▶ 210]	PressureA actpoint
ActPressureB	LoadData [▶ 210]	PressureB actpoint
ActForce	LoadData [▶ 210]	Force actpoint

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.3 CDT_MCTOPLC_AXIS_LOADCONTROL

CDT provides load setpoint and actpoint.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_AXIS_LOADCONTROL :
STRUCT
    IsConnected  : BOOL;
    IsInLoadMode : BOOL;
    LoadType     : ELoadType;
    Set          : LoadData;
    Act          : LoadData;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is beeing received from MC.
IsInLoadMode	BOOL	True if the axis is producing load setpoints, false otherwise.
LoadType	ELoadType [▶ 200]	Load selected for load control.

Name	Type	Description
Set	LoadData [▶ 210]	Current load setpoint.
Act	LoadData [▶ 210]	Current load actpoint.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.4 CDT_MCTOPLC_AXIS_MODULO

CDT provides information about modulo positioning.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_AXIS_MODULO :
STRUCT
    IsConnected : BOOL;
    SetPosition : LREAL;
    SetTurns    : LINT;
    ActPosition : LREAL;
    ActTurns    : LINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is beeing received from MC.
SetPosition	LREAL	Modulo set position within one modulo period.
SetTurns	LINT	Number of complete modulo turns for the set position.
ActPosition	LREAL	Modulo actual position within one modulo period.
ActTurns	LINT	Number of complete modulo turns for the actual position.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.5 CDT_MCTOPLC_AXIS_SET

CDT provides current set values.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_AXIS_SET :
STRUCT
    IsConnected  : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is beeing received from MC.
Position	LREAL	Current position setpoint of the axis.
Velocity	LREAL	Current velocity setpoint of the axis.
Acceleration	LREAL	Current acceleration setpoint of the axis.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.6 CDT_MCTOPLC_AXIS_STD

Syntax

Definition:

```
TYPE CDT_MCTOPLC_AXIS_STD :
STRUCT
    AxisOid         : OTCID;
    AxisState       : EAxisState;
    IsInControl     : BOOL;
    HasError        : BOOL;
    ErrorId         : UDINT;
    IsPositionValid : BOOL;
    DcTime          : LINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
AxisOid	OTCID	Object ID of the axis for identification.
AxisState	EAxisState [▶ 194]	Current state of the axis state machine.
IsInControl	BOOL	TRUE if the axis controller is actively controlling the drive (control loop closed).
HasError	BOOL	TRUE if the axis is in an error state.
ErrorId	UDINT	Error code of the current axis error (0 if no error).
IsPositionValid	BOOL	TRUE if the reported position values are valid (e.g. after homing or with absolute encoder and drive/ encoder signals a valid position).
DcTime	LINT	Distributed clock timestamp of the cyclic data exchange.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.7 CDT_MCTOPLC_CAMTABLE

Syntax

Definition:

```
TYPE CDT_MCTOPLC_CAMTABLE :
STRUCT
    Oid : OTCID;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Oid	OTCID	Object ID of the cam table instance. Must be mapped to an MC_CamObject instance in the PLC.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.8 CDT_MCTOPLC_ENCODER_AXIS_ACT

CDT provides current actual values from encoder axis.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_ENCODER_AXIS_ACT :
STRUCT
    IsConnected  : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is being received from MC.
Position	LREAL	Current actual position of the encoder axis.
Velocity	LREAL	Current actual velocity of the encoder axis.
Acceleration	LREAL	Current actual acceleration of the encoder axis.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.9 CDT_MCTOPLC_ENCODER_AXIS_MODULO

CDT provides modulo positioning information from encoder axis.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_ENCODER_AXIS_MODULO :
STRUCT
    IsConnected : BOOL;
    Position    : LREAL;
    Turns       : LINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
IsConnected	BOOL	TRUE if the data area is connected and data is being received from MC.
Position	LREAL	Modulo position within one modulo period.
Turns	LINT	Number of complete modulo turns.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.10 CDT_MCTOPLC_ENCODER_AXIS_STD

Standard cyclic data from MC encoder axis to PLC.

Syntax

Definition:

```
TYPE CDT_MCTOPLC_ENCODER_AXIS_STD :
STRUCT
    EncoderAxisOid  : OTCID;
    HasError        : BOOL;
    IsPositionValid : BOOL;
    ErrorId         : UDINT;
    DcTime          : LINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
EncoderAxisOid	OTCID	Object ID of the encoder axis for identification.
HasError	BOOL	TRUE if the encoder axis is in an error state.
IsPositionValid	BOOL	TRUE if the reported position values are valid.
ErrorId	UDINT	Error code of the current encoder axis error (0 if no error).
DcTime	LINT	Distributed clock timestamp of the cyclic data exchange.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.11 CDT_MCTOPLC_VALVECHARACTERISTICCURVE

Syntax

Definition:

```
TYPE CDT_MCTOPLC_VALVECHARACTERISTICCURVE :
STRUCT
    Oid : OTCID;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Oid	OTCID	Object ID of the valve characteristic curve instance.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.12 CDT_PLCTOMC_AXIS_STD

Syntax

Definition:

```
TYPE CDT_PLCTOMC_AXIS_STD :
STRUCT
    Enable         : BOOL;
    EnablePositive : BOOL;
    EnableNegative : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Enable	BOOL	Enables the axis power stage.
EnablePositive	BOOL	Allows motion in positive direction when TRUE.
EnableNegative	BOOL	Allows motion in negative direction when TRUE.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.1.2.13 LoadData

Syntax

Definition:

```
TYPE LoadData :
STRUCT
    Value      : LREAL;
    Derivative : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Value	LREAL	Current load value in the configured load unit.
Derivative	LREAL	First time derivative of the load value.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.2 Interfaces

4.2.1 ITcMcExternalSetpointGenerator

Interface for implementing custom external setpoint generators.

Inheritance Hierarchy

ITcUnknown
  ITcMcExternalSetpointGenerator

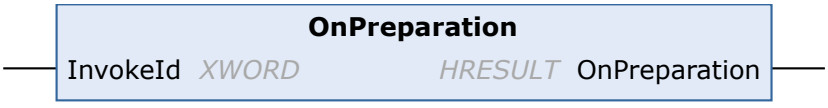
Methods

Name	Origin	Description
TcAddRef	ITcUnknown	Increments the reference counter.
TcQueryInterface	ITcUnknown	Get a reference to an implemented interface.
TcRelease	ITcUnknown	Decrements the reference counter.
OnPreparation [▶ 211]	ITcMcExternalSetpointGenerator	Called once when the external setpoint generator is being prepared for activation.
OnStartPointInitialization [▶ 212]	ITcMcExternalSetpointGenerator	Called once to provide the initial motion state before cyclic setpoint generation begins.
OnCyclicSetpointGeneration [▶ 213]	ITcMcExternalSetpointGenerator	Called every cycle to compute the next setpoint values for position, velocity, and acceleration.
OnDeactivation [▶ 213]	ITcMcExternalSetpointGenerator	Called when the external setpoint generator is deactivated/aborted. Use this method to tidy up, e. g. release created objects.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

4.2.1.1 OnPreparation



Called once when the external setpoint generator is being prepared for activation.

Syntax

Definition:

```
METHOD OnPreparation : HRESULT
VAR_INPUT
    InvokeId : XWORD;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	Unique identifier, the interpretation is user specific.

Return value

HRESULT

4.2.1.2 OnStartPointInitialization

Called once to provide the initial motion state before cyclic setpoint generation begins.

Syntax

Definition:

```
METHOD OnStartPointInitialization : HRESULT
VAR_INPUT
    InvokeId     : XWORD;
    StartDcTime  : LINT;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	Unique identifier, the interpretation is user specific.
StartDcTime	LINT	DC time at which the generation starts.
Position	LREAL	Start position of the axis.
Velocity	LREAL	Start velocity of the axis.
Acceleration	LREAL	Start acceleration of the axis.

Return value

HRESULT

4.2.1.3 OnCyclicSetpointGeneration

Called every cycle to compute the next setpoint values for position, velocity, and acceleration.

Syntax

Definition:

```
METHOD OnCyclicSetpointGeneration : HRESULT
VAR_INPUT
    InvokeId         : XWORD;
    TimeInGeneration : LREAL;
    Position         : Reference To LREAL;
    Velocity         : Reference To LREAL;
    Acceleration     : Reference To LREAL;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	Unique identifier, the interpretation is user specific.
TimeInGeneration	LREAL	Elapsed time since the start of setpoint generation (input from PLC side of view).
Position	Reference To LREAL	Reference to receive the computed position setpoint. Value must be set during method call (output from PLC side of view).
Velocity	Reference To LREAL	Reference to receive the computed velocity setpoint. Value must be set during method call (output from PLC side of view).
Acceleration	Reference To LREAL	Reference to receive the computed acceleration setpoint. Value must be set during method call (output from PLC side of view).

Return value

HRESULT

4.2.1.4 OnDeactivation

Called when the external setpoint generator is deactivated/aborted. Use this method to tidy up, e. g. release created objects.

Syntax

Definition:

```
METHOD OnDeactivation : HRESULT
VAR_INPUT
    InvokeId : XWORD;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	Unique identifier, the interpretation is user specific.

Return value

HRESULT

4.3 Overview: MC3 types

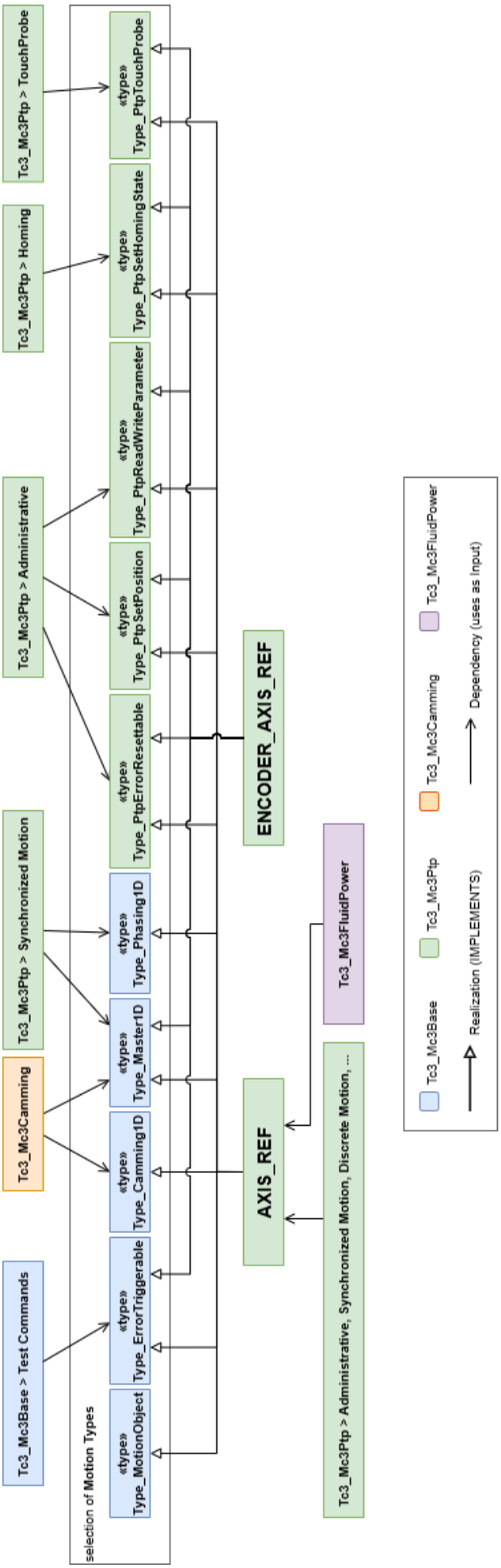
MC3 function blocks use typed axis references to define the required capabilities of an axis. Each type represents a specific motion ability. This capability could, for example, be the ability to act as a coupling master or the axis's ability to receive a position setpoint. Each motion object (`AXIS_REF`, `ENCODER_AXIS_REF`) implements only the capabilities that it actually supports.

The compiler checks whether the motion object implements the types required by the function block. If this is not the case, the program cannot be compiled. This prevents operations from being called on motion objects that do not support them. The function block only queries skills, not the motion object itself. The compiler checks whether the motion objects implement this capability.

The diagram outlines the dependencies and origins of the types (`Type_*`) and motion objects (`AXIS_REF`, `ENCODER_AXIS_REF`). The key levels can be seen in the center:

- Types: `Type_Master1D`, etc., which define what a motion object can do.
- Motion objects: `AXIS_REF` and `ENCODER_AXIS_REF`, which are declared in the PLC program.

The MC3 function blocks use the specialized motion objects or the types directly.


Tc3_Mc3Ptp > TouchProbe
Tc3_Mc3Ptp > Homing
Tc3_Mc3Ptp > Administrative
Tc3_Mc3Ptp > Synchronized Motion
Tc3_Mc3Camming
Tc3_Mc3Base > Test Commands
selection of Motion Types
«type» Type_MotionObject
«type» Type_ErrorTriggerable
«type» Type_Camming1D
«type» Type_Master1D
«type» Type_Phasing1D
«type» Type_PtpErrorResettable
«type» Type_PtpSetPosition
«type» Type_PtpReadWriteParameter
«type» Type_PtpSetHomingState
«type» Type_PtpTouchProbe
AXIS_REF
ENCODER_AXIS_REF
Tc3_Mc3FluidPower
Tc3_Mc3Ptp > Administrative, Synchronized Motion, Discrete Motion, ...
Tc3_Mc3Base
Tc3_Mc3Ptp
Tc3_Mc3Camming
Tc3_Mc3FluidPower
Realization (IMPLEMENTS)
Dependency (uses as Input)

Type	AXIS_REF	ENCODER_AXIS_REF
Type_PtpErrorResettable	X	X
Type_PtpReadWriteParameter	X	X
Type_PtpSetHomingState	X	X
Type_PtpSetPosition	X	X
Type_PtpTouchProbe	X	X
Type_Camming1D	X	
Type_ErrorTriggerable	X	X
Type_Master1D	X	X
Type_MotionObject	X	
Type_Phasing1D	X	

Use

The function blocks Tc3_Mc3Base, Tc3_Mc3Ptp, Tc3_Mc3Camming, and Tc3_Mc3FluidPower require parameterization using specialized types (`Type_*`) or more general motion objects (`AXIS_REF`). The diagram above shows that the more general motion objects implement selected types.

1. In the PLC, create an instance of the `AXIS_REF` type for a PTP or hydraulic axis as usual. The specialized types are not suitable for use as standalone instances.
2. In addition to `AXIS_REF`, other motion objects with various capabilities can be instantiated (e.g., a position sensor as `ENCODER_AXIS_REF`).
   ⇨ Functions blocks can only be used with motion objects that provide the necessary capabilities. It is not possible to run a `MC_MoveAbsolute` with a `ENCODER_AXIS_REF`. This causes a compiler error.

Using these types - for example, with coupling function blocks - allows the function blocks to be reused with different Motion objects. For a function block input of type `Type_Camming1D`, it is clear that this parameter requires an object that can act as a slave object for cam plates. Incorrect use of motion objects is detected during compilation, not just at runtime of the user program.

The table shows a choice of function blocks and the motion objects or types used.

Function block	Parameter	Required type	AXIS_REF	ENCODER_AXIS_REF
Basic Commands				
MC_Power	Axis	AXIS_REF	X	-
MC_Home	Axis	AXIS_REF	X	-
MC_SetHomingState	Axis	Type_PtpSetHomingState	X	X
MC_Stop	Axis	AXIS_REF	X	-
MC_Halt	Axis	AXIS_REF	X	-
MC_Reset	Axis	Type_PtpErrorResettable	X	X
MC_MoveAbsolute	Axis	AXIS_REF	X	-
MC_MoveRelative	Axis	AXIS_REF	X	-
MC_MoveVelocity	Axis	AXIS_REF	X	-
MC_MoveModulo	Axis	AXIS_REF	X	-
MC_SetOverride	Axis	AXIS_REF	X	-
MC_SetPosition	Axis	Type_PtpSetPosition	X	X
MC_ReadParameter	Axis	Type_PtpReadWriteParameter	X	X
MC_WriteParameter	Axis	Type_PtpReadWriteParameter	X	X
MC_ReadBoolParameter	Axis	Type_PtpReadWriteParameter	X	X

Function block	Parameter	Required type	AXIS_REF	ENCODER_AXIS_REF
MC_WriteBoolParameter	Axis	Type_PtpReadWriteParameter	X	X
MC_ReadHardwareParameter	Axis	AXIS_REF	X	X
MC_WriteHardwareParameter	Axis	AXIS_REF	X	X
Gear coupling				
MC_GearIn	Master	Type_Master1D	X	X
	Slave	AXIS_REF	X	-
MC_GearInPos	Master	Type_Master1D	X	X
	Slave	AXIS_REF	X	-
Cam plates				
MC_CamIn	Master	Type_Master1D	X	X
	Slave	Type_Camming1D	X	-
MC_CamTableSelect	Slave	Type_Camming1D	X	-
Phasing				
MC_PhasingAbsolute	Master	Type_Master1D	X	X
	Slave	Type_Phasing1D	X	-
MC_PhasingRelative	Master	Type_Master1D	X	X
	Slave	Type_Phasing1D	X	-
Touch Probe				
MC_TouchProbe	Axis	Type_PtpTouchProbe	X	X
	TriggerInput	TRIGGER_REF	separate type	
MC_AbortTrigger	Axis	Type_PtpTouchProbe	X	X
	TriggerInput	TRIGGER_REF	separate type	

Example

Declaration:

```
VAR
    myAxis1        : AXIS_REF;
    myAxis2        : AXIS_REF;
    myEncoder      : ENCODER_AXIS_REF;
    myGearIn       : MC_GearIn;
    myTypeInstance : Type_Master1D;
END_VAR
```

Implementation:

```
// valid call
myGearIn(
    Master:= myAxis1,
    Slave:= myAxis2,
    Execute:= TRUE);
```

```
// valid call
myGearIn(
    Master:= myEncoder,
    Slave:= myAxis2,
    Execute:= TRUE);
```

```
myGearIn(
    Master:= myAxis1,
    Slave:= myEncoder, // Compiler error
    Execute:= TRUE);
```

```
myGearIn(
    Master:= myTypeInstance, // FB runtime error 16#8145
    Slave:= myAxis2,
    Execute:= TRUE);
```

The MC_GearIn command expects an object of type AXIS_REF at the Slave input. Therefore, it is not possible to use an encoder axis of the type ENCODER_AXIS_REF; doing so will result in a compiler error. The same applies, for example, to the command MC_CamIn, since ENCODER_AXIS_REF does not offer the type Type_Camming1D.

MC3 types do not contain any implementation code. They are not available in Solution Explorer for creating a link between an MC3 axis and an instance in the PLC. In the last example, an instance of the type Type_Master1D is assigned to the Master input. This does not result in a compiler error, since the type is essentially correct. However, the function block's logic detects that a link to an axis is missing and reports this with error 16#8145.

Migrating from NC2 to MC3

NC2	MC3	Required action
Axis : Tc2_MC2.AXIS_REF	Axis : Tc3_Mc3Ptp.AXIS_REF	**None** – In MC3, the function blocks use the motion objects or types as input. Since the motion objects provide the appropriate types, there is no need to change the input to the function block or the declaration.
Encoder : Tc2_MC2.AXIS_REF	Encoder : Tc3_Mc3Ptp.ENCODER_AXIS_REF	For encoder axes, select the standalone ENCODER_AXIS_REF in MC3.

5 PLC libraries

5.1 Named constants

Named constants

Special input values can be used for inputs on function blocks of the type `MC_LREAL` (see the chapter MC_LREAL/Special input values). The MC3 uses only the standard data type `LREAL` instead of `MC_LREAL` and supports the use of so-called *named constants*.

Named constants are silent NaN values according to IEEE 754, used to indicate uninitialized or invalid data. The mantissa is encoded in a Beckhoff-specific bit pattern. The advantage is improved readability, for example, when viewed online. For example, the previous default value of 0 can be interpreted as a placeholder for the default dynamics or as the actual value 0. In this case, misinterpretations or incorrect mathematical operations are possible.

The following points rank among the main properties of NaN values:

- All arithmetic operations that use NaN as input data return NaN as the result.
- All relational operators =, !=, > < >= <= always return the value False if at least one of the operands is NaN.
- The Standard C functions `isnan()` and `_isnan()`, as well as the PLC functions LrealIsNaN() and RealIsNaN() (Tc2_Utilities library), return True if the argument has the value NaN.
- The expression `isnan(a)` is equivalent to the expression `!(a == a)` or `NOT(a = a)`.

The fact that NaN values propagate when used in further calculations has the advantage that invalid values cannot be overlooked.

⚠ CAUTION

Software malfunctions

NaN values can lead to potentially dangerous malfunctions of the software concerned!

- Only use expressly approved NaN values, specifically as control values in motion control and drive control functions.

For further information, see the chapter on NaN values.

The syntax for use in the PLC is `STRING_TO_LREAL('[Wert]')` or the shorthand form `TO_LREAL('[Wert]')`. The following constants can be used for the MC3:

Value	Meaning	Example
#Invalid	Invalid value	Default input value for `MC_MoveVelocity.Velocity`, since this input must be defined when using the function block.
#Default	Can be used, for example, to use the axis's default dynamics as input for function blocks.	`MC_MoveAbsolute.Velocity := TO_LREAL('#Default');`
#Ignore	Can be used to disable inputs or boundary conditions.	Disabling the software limit switches on one side. Define the position derivatives as input or boundary conditions for the MotionFunctionPoints (camming) to be ignored.
#Maximum	Can be used, for example, to use the maximum dynamics of the axis as input for function blocks.	`MC_MoveVelocity.Velocity := TO_LREAL('#Maximum');`

Value	Meaning	Example
#MaximumNegative	Can be used, for example, to use the inverted maximum dynamics of the axis as input for function blocks.	`MC_MoveVelocity.Velocity := TO_LREAL('#MaximumNegative');`
#DefaultNegative	Can be used, for example, to use the inverted default dynamics of the axis as input for function blocks.	`MC_MoveVelocity.Velocity := TO_LREAL('#DefaultNegative');`

The syntax and usage of named constants have been optimized in TwinCAT 3.1 Build 4026.25 and are described below.

Available from TwinCAT 3.1 Build 4026.25

Special values, such as NaN (Not a Number), are now provided as type-safe constants. The LREAL_CONST namespace provides access to the following variables:

Constant	Meaning
DEFAULT	Application-specific default value (positive)
IGNORE	Marker value “do not evaluate”
INVALID	Marker value for invalid size
MAXIMUM	Largest finite positive value
DEFAULT_NEGATIVE	Application-specific default value (negative)
MAXIMUM_NEGATIVE	Largest finite negative value
NAN	Not a Number
INFINITY	Positive infinity
INFINITY_NEGATIVE	Negative infinity

Example: Using

```
VAR
    fNumerator : LREAL;
    fDenominator : LREAL;
    fResult : LREAL
END_VAR
IF fDenominator = 0 THEN
    fDenominator := LREAL_CONST.NAN;// invalid: division by zero
END_IF
    fResult := fNumerator / fDenominator;
```

5.2 Tc3_Mc3Base

The Tc3_Mc3Base library contains basic data types that are used across the other MC3 libraries.

5.2.1 Data Types

5.2.1.1 Enums

5.2.1.1.1 EBufferMode

Buffer mode between buffered commands.

Syntax

Definition:

```
TYPE EBufferMode :
(
    Aborting := 0,
    Buffered := 1
)UINT;
END_TYPE
```

Values

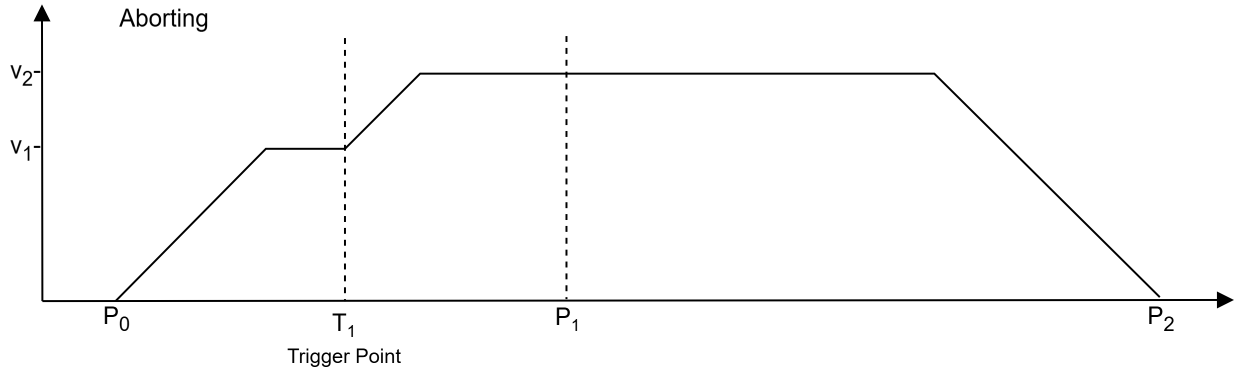
Name	Description
Aborting	The previous command is aborted and a transition to the new command is initiated immediately.
Buffered	The previous command is executed to completion. The new command affects the motion object as soon as the previous movement is 'Done'/'InVelocity'/'InSync'/'InGear'. There is no blending.

Further information

The data type `EBufferMode` is used in motion commands to specify how they replace the previous motion command.

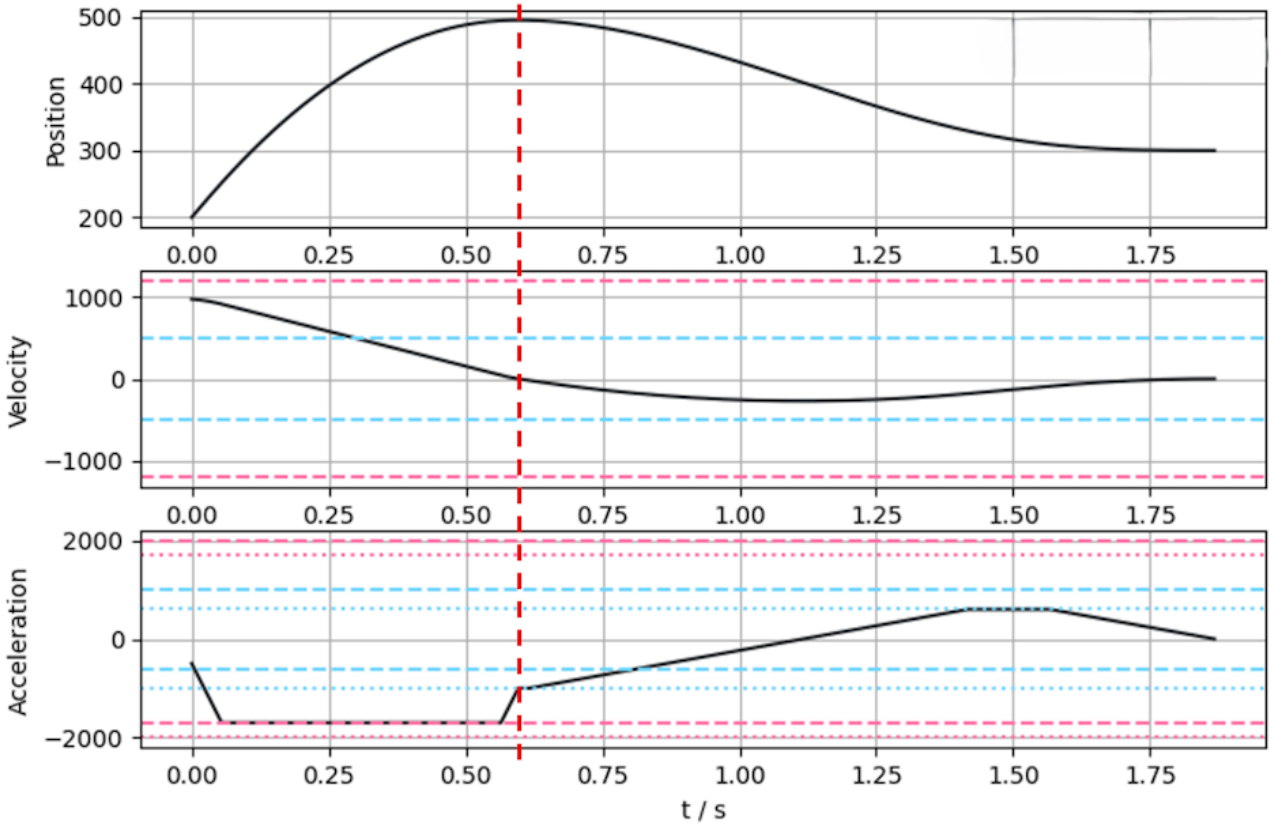
Aborting

The previous command is canceled immediately and indicates `CommandAborted = TRUE`. If higher dynamics are specified for the new command, they are immediately adopted as the limit for axis movement.



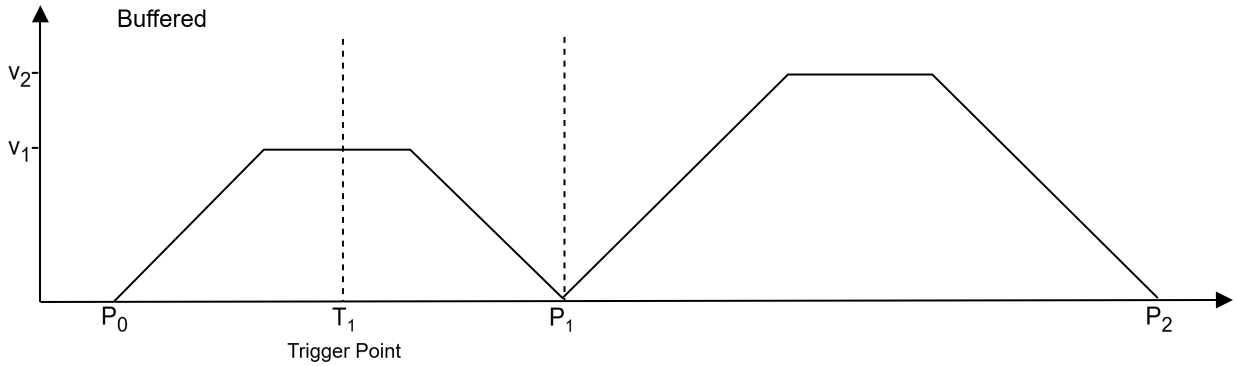
If lower dynamics are specified for the new command, a transition phase may occur. The axis is moved to the target state or within the range of the new dynamic limits as quickly as possible, while overshoots and undershoots are suppressed (e.g., overshooting the target position specified by the previous command). If this is also possible with the reduced dynamics, they are applied immediately. Otherwise, the axis will be decelerated using the higher dynamics from the previous command. This braking phase remains active until the movement is directed toward the target state.

The figure below shows a reversal movement. The original target position is 500; the aborting command is configured with lower dynamic parameters and a target position of 300. If the lower dynamics were applied instantly, the original target position of 500 would be exceeded. On the other hand, the axis would move further away from its new target state. The higher dynamics from the previous command remain active up to the red vertical marker. From the perspective of the new command, the overshoot relative to position 300 is kept as small as possible, taking command dynamics into account. In addition, the original target position is not exceeded.



Buffered

The previous command is executed to completion. The new command affects the moving object as soon as the previous movement is complete (Done/InVelocity/InSync/InGear). No blending takes place.



Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.1.1.2 EReactionToMasterError

Reaction type to master errors in master/slave couplings.

Syntax

Definition:

```
TYPE EReactionToMasterError :
(
    TriggerSlaveError         := 0,
```

```
    KeepCouplingWithSetValues := 1
)UINT;
END_TYPE
```

Values

Name	Description
TriggerSlaveError	In case of a master error, a slave error is triggered.
KeepCouplingWithSetValues	In case of a master error the coupling is kept with set values and no slave error is triggered.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.1.1.3 EReactionToSlaveError

Reaction type to slave errors in master/slave couplings.

Syntax

Definition:

```
TYPE EReactionToSlaveError :
(
    TriggerMasterError := 0,
    NoReaction         := 1
)UINT;
END_TYPE
```

Values

Name	Description
TriggerMasterError	In case of a slave error a master error is triggered.
NoReaction	No reaction on the master in case of a slave error.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.1.1.4 ESyncDirection

Directions in which a slave is allowed to move during synchronizing phase.

Syntax

Definition:

```
TYPE ESyncDirection :
(
    Positive                := 1,
    Negative                := 2,
    Both                    := 3,
    TowardsSlaveSyncPosition := 4
)UINT;
END_TYPE
```

Values

Name	Description
Positive	Movement is allowed only in positive direction while synchronizing.

Name	Description
Negative	Movement is allowed only in negative direction while synchronizing.
Both	Movement is allowed in any direction while synchronizing.
TowardsSlaveSyncPosition	Movement is allowed only in direction of the SlaveSyncPosition while synchronizing.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.2 Interfaces

5.2.2.1 Types

5.2.2.1.1 Type_Camming1D

Type for a motion control object that can operate as a 1D camming slave object (like slave axis for MC_CamIn, e. g. supported by AXIS_REF).

Inheritance Hierarchy

Type_Camming1D

Methods

Name	Origin	Description
GetCamming1DOid [▶ 224]	Type_Camming1D	

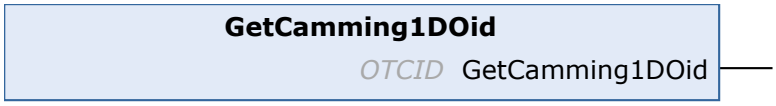
Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.2.1.1.1 GetCamming1DOid



Syntax

Definition:

```
METHOD GetCamming1DOid : OTCID
```

Return value

OTCID

5.2.2.1.2 Type_ErrorTriggerable

Type for a motion control object that supports error triggering (e. g. supported by AXIS_REF).

Inheritance Hierarchy

Type_ErrorTriggerable

Methods

Name	Origin	Description
GetErrorTriggerableOid [▶ 225]	Type_ErrorTriggerable	

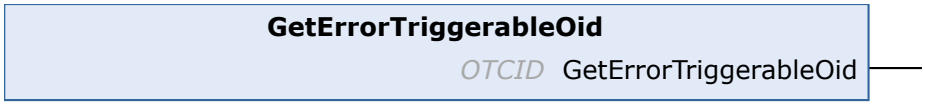
Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.2.1.2.1 GetErrorTriggerableOid



Syntax

Definition:

```
METHOD GetErrorTriggerableOid : OTCID
```

Return value

OTCID

5.2.2.1.3 Type_Master1D

Type for a motion control object that can operate as a 1D master object (like master axis for MC_GearIn, e. g. supportet by AXIS_REF).

Inheritance Hierarchy

Type_Master1D

Methods

Name	Origin	Description
GetMaster1DOid [▶ 226]	Type_Master1D	

Uses of the type

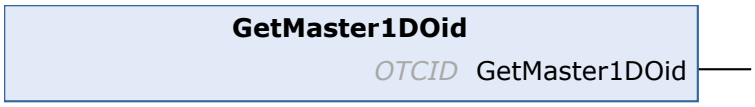
- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1

- TF5500 MC3 Base >= v4.0.6

5.2.2.1.3.1 GetMaster1DOid



Syntax

Definition:

```
METHOD GetMaster1DOid : OTCID
```

Return value

OTCID

5.2.2.1.4 Type_MotionObject

Inheritance Hierarchy

Type_MotionObject

Methods

Name	Origin	Description
GetMotionObjectOid [▶ 226]	Type_MotionObject	

Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.2.1.4.1 GetMotionObjectOid

Syntax

Definition:

```
METHOD GetMotionObjectOid : OTCID
```

Return value

OTCID

5.2.2.1.5 Type_Phasing1D

Type for a motion control object that supports 1D phasing (MC_PhasingAbsolute, e. g. supported by AXIS_REF).

Inheritance Hierarchy

Type_Phasing1D

Methods

Name	Origin	Description
GetPhasing1DOid [▶ 227]	Type_Phasing1D	

Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.2.2.1.5.1 GetPhasing1DOid

Syntax

Definition:

```
METHOD GetPhasing1DOid : OTCID
```

Return value

OTCID

5.2.3 Function Blocks

5.2.3.1 TestCommands

5.2.3.1.1 MC_TriggerError

This function block triggers an error in the target motion object.

Syntax

Definition:

```
FUNCTION_BLOCK MC_TriggerError
VAR_INPUT
    MotionObject : Type_ErrorTriggerable;
    Execute      : BOOL;
    ErrorCode    : UDINT;
    ControlError : BOOL;
END_VAR
VAR_OUTPUT
    Done         : BOOL;
```

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

Inputs

Name	Type	Description
MotionObject	Type_ErrorTriggerable [▶ 225]	Interface pointer to the object on which the error should be triggered (e.g. Axis).
Execute	BOOL	Trigger the command with rising edge.
ErrorCode	UDINT	Error code to be triggered
ControlError	BOOL	Trigger a ControlError (MC has no control over the axis, i.e. hardware or communication fault).

Outputs

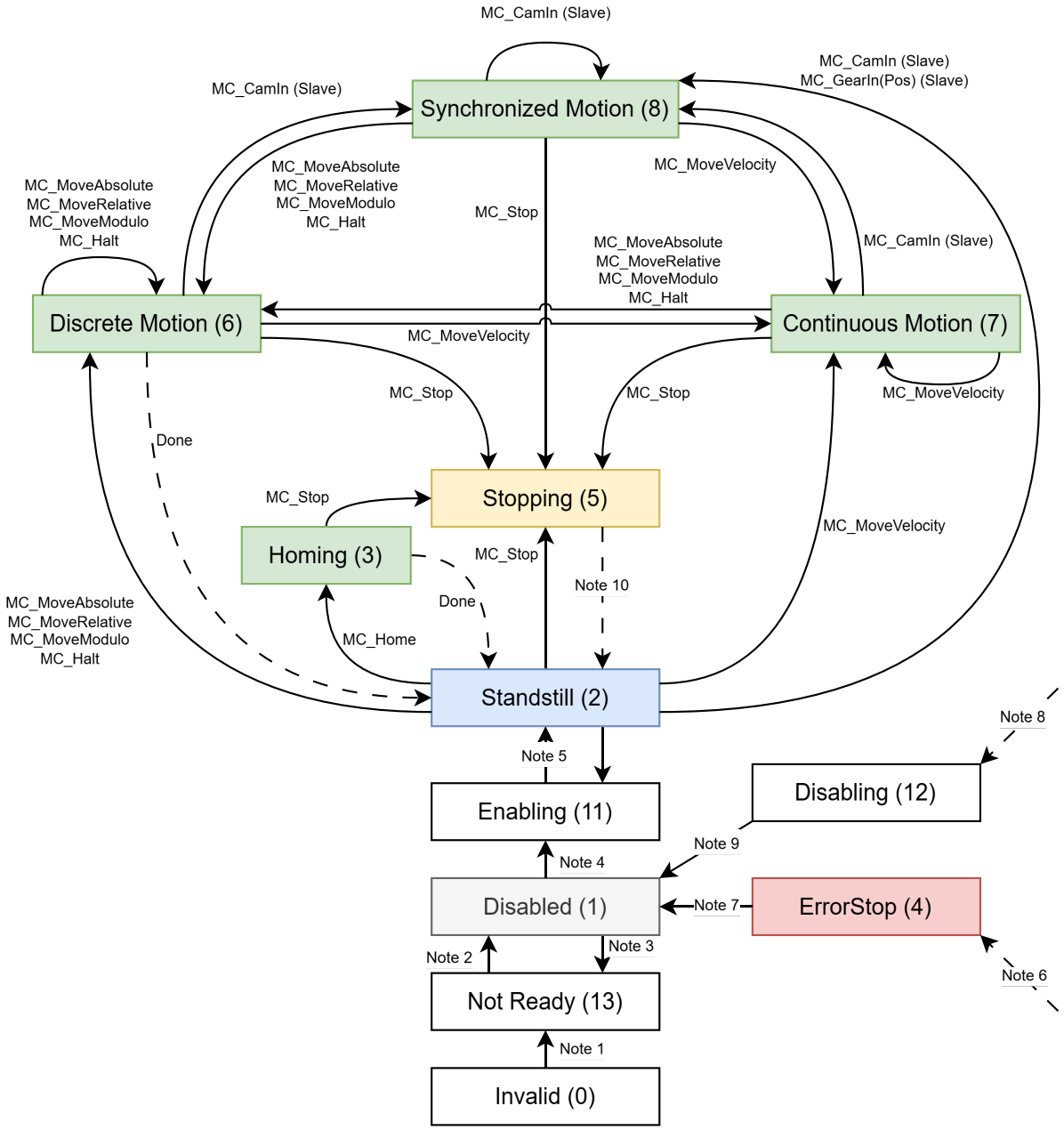
Name	Type	Description
Done	BOOL	Error has been successfully triggered.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3 Tc3_Mc3Ptp

5.3.1 State diagram



Note	Description
Note 1	State transition during XAR startup and axis initialization. This requires a link between `AXIS_REF` and the TcCOM axis object.
Note 2	Drive: `WcState` = 0, `TxPdoState` = 0 (if available), `Position actual value invalid` = 0 (if available), status word indicates readiness for controller enable (e.g., `ReadyToSwitchOn` for DS402).
Note 3	The drive's state changes, preventing the controller from being enabled (e.g., `SwitchOnDisabled` if there is no supply voltage to the power section).
Note 4	`MC_Power.Enable = TRUE`
Note 5	`MC_Power.Status = TRUE` and the drive controller is active (e.g., `Operation Enabled` for the DS402).

Note	Description
Note 6	From any state in which an error occurs. This also applies, for example, to `WcState` and `Position actual value invalid` if they initially reported a valid state after startup and then lost that state.
Note 7	`MC_Reset` (forward the call immediately if `MC_Power.Enable = TRUE`)
Note 8	From any state, provided that `MC_Power.Enable = FALSE` and the axis has no error.
Note 9	The drive controller is no longer active, and the status word indicates that the controller is ready to be enabled (e.g., `ReadyToSwitchOn` for the DS402).
Note 10	`MC_Stop.Done = TRUE und MC_Stop.Execute = FALSE`

Motion commands are always executed sequentially. All commands operate within the state diagram described. In the PLC, the states are contained in the enumeration type EAxisState. The numerical values stored there are listed in parentheses next to the states in the diagram (e.g., “Standstill (2)”).

The axis is always in one of the defined states. Each motion command that causes a transition changes the state of the axis and thus the motion profile. The state diagram is an abstraction layer that reflects the real axis state, comparable to the process image for I/O points. The axis state changes immediately when the command is issued.

The state diagram refers to single axes. Multi-axis blocks such as MC_CamIn or MC_GearIn influence the states of several axes, which can always be traced back to individual axis states of the axes involved in the process. For example, a cam plate master can be in the `Continuous Motion` state, while the associated slave is in the `Synchronized Motion` state. Coupling a slave has no effect on the state of the master.

The `Disabled` state is the default state of an axis. In this state, the axis cannot be moved by any function block, but it is ready for operation. When the MC_Power function block is called with `Enable = TRUE`, the axis changes to the `Standstill` state or, in the event of an error, to the `ErrorStop` state. If the MC_Power function block is called with `Enable = FALSE`, the state changes to `Disabled`.

The purpose of the `ErrorStop` state is to stop the axis and then reject any further commands until a reset has been triggered. The `Error` state transition only refers to actual axis errors and not to function block execution errors. Axis errors can also be indicated at the error output of a function block.

The `Stopping` state indicates that the axis is in a stop ramp. The state changes to the `Standstill` state once the axis has come to a complete stop.

Motion commands such as MC_MoveAbsolute that exit the `Synchronized Motion` state can be used without additional parameterization.

Function blocks that are not listed in the state diagram do not affect the state of the axis (e.g. MC_ReadStatus, MC_ReadParameter, MC_WriteParameter, ...).

Meaning of the states

Name	Description
Invalid	• Invalid axis state • The axis is not ready for operation. • A possible cause could be that AXIS_REF is not linked to an MC3 axis.
Disabled	• Basic state of an axis. • The axis is not released. • There is no axis error.
Standstill	• The axis is enabled. • No command has been issued to the axis.
Homing	• The axis is enabled. • The axis is homed (e.g., MC_Home).
ErrorStop	• An axis error has occurred. • A reset is required to exit this state.
Stopping	• The axis is enabled.

Name	Description
	• The axis is on a stop ramp (MC_Stop). • MC_Stop.Execute is set and blocks new motion commands.
DiscreteMotion	• The axis is enabled. • The axis performs a point-to-point movement (e.g., MC_MoveAbsolute, MC_MoveRelative, MC_MoveModulo, MC_Halt).
ContinuousMotion	• The axis is enabled. • The axis performs a continuous motion (e.g., MC_MoveVelocity).
SynchronizedMotion	• The axis is enabled. • The axis is linked to another axis (e.g., MC_CamIn, MC_GearIn, MC_GearInPos).
ResetPending	• Internal state
Enabling	• Extended axis status for representation of hardware conditions. • The command to enable the drive hardware's output stage has already been sent or is currently being sent, and the MC3 axis is waiting for the drive hardware to reach the “operational” state
Disabling	• Extended axis status for representation of hardware conditions. • The command to deactivate the drive hardware's output stage has already been sent or is currently being sent, and the MC3 axis is waiting for the drive hardware to enter the “not operational” state.
NotReady	• Extended axis status for representation of hardware conditions. • The axis is not ready for operation. • There is no active communication with the hardware (e.g., WcState), or the drive's state machine reports “not ready”.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.2 Moving the axis via the PLC

⚠ DANGER

This example requires simulation axes - using real axes poses a risk of injury!

This example creates an axis movement. When using real axes, there is a risk of injury from the movement of the axes.

- Make sure that neither you nor others are harmed by the movement, e.g. by maintaining a suitable safety distance.
- Do not perform any action whose consequences you cannot estimate.

Requirements

- A TwinCAT project with an MC project has been created;
  see Creating an MC project [▶ 17].
- A simulation axis was created in the MC project;
  see Create axis object [▶ 25].
- A PLC project was created in the TwinCAT project.

Including the PLC library

1. Add the Tc3_Mc3Ptp library to the PLC project. To do this, right-click on **References** in the PLC project and select **Add library…**

2. In the **Add Library** dialog, select the **Tc3_Mc3Ptp** library and click **OK** to confirm.

   ⇨ The Tc3_Mc3Ptp library has been added to the PLC project. The library's function blocks and data types can now be used.

Writing a PLC program

All necessary data types and function blocks for a simple PLC motion program with MC3 are contained in the Tc3_Mc3Ptp library.

AXIS_REF

For each axis one instance of the data type AXIS_REF [▶ 287] is required, which is the interface between PLC and MC3. It contains all the information about the axis that is given to the MC function blocks as a reference.

MC_Power

The MC_Power [▶ 248] function block is used to enable the axis and specify its direction of motion.

MC_SetOverride

The MC_SetOverride [▶ 256] function block can be used to change the axis's velocity override. By default, the factor is set to 1 (=100%), so changing the factor is optional.

MC_Reset

The function block MC_Reset [▶ 255] can be used to reset an error on an axis.

MC_MoveRelative

The MC_MoveRelative [▶ 274] function block can be used to move the axis by a specified distance.

Simple programming

3. Open MAIN(PRG) under POUs in the PLC project.
4. Insert the following declarations into MAIN(PRG).

```
PROGRAM MAIN
VAR
    axis           : AXIS_REF;
    power          : MC_Power;
    setOverride    : MC_SetOverride;
    moveRelative   : MC_MoveRelative;
    reset          : MC_Reset;
    stop           : MC_Stop;
    enableAxis     : BOOL;
    enableOverride : BOOL;
    executeStop    : BOOL;
    executeReset   : BOOL;
    executeMove    : BOOL;
    velFactor      : LREAL := 1; // adapt override factor if necessary(1 corresponds to 100 %)
    distance       : LREAL := ???; // ToDo: set to a reachable position
    velocity       : LREAL := ???; // ToDo: set velocity for move

END_VAR
```

5. Insert the following program code into MAIN(PRG).

```
power(
        Axis           := axis,
        Enable         := enableAxis,
        EnablePositive := enableAxis,
        EnableNegative := enableAxis,
        Status         => ,
        Error          => ,
        ErrorId        => );

setOverride(
        Axis           := axis,
        Enable         := enableOverride,
        VelocityFactor := velFactor,
```

```
        Enabled=> ,
        Error=> ,
        ErrorId=> );

stop(
        Axis            := axis,
        Execute         := executeStop, // The command is executed with a positive edge.
        Deceleration    := , // If the value is <= 0, the deceleration
                            // parameterized with the last Move command is used.
        Jerk            := , // If the value is <= 0, the jerk parameterized
                            // with the last Move command is used.
        Done            => ,
        Busy            => ,
        Active          => ,
        CommandAborted  => ,
        Error           => ,
        ErrorId         => );

reset(
        Axis            := axis,
        Execute         := executeReset, // The command is executed with a positive edge.
        Done            => ,
        Busy            => ,
        Error           => ,
        ErrorId         => );

moveRelative(
        Axis            := axis,
        Execute         := executeMove, // The command is executed with a positive edge.
        Distance        := distance,
        Velocity        := velocity,
        Acceleration    := LREAL_CONST.DEFAULT, // If the value is LREAL_CONST.DEFAULT,
                            // the standard acceleration from the axis configuration is used.
        Deceleration    := LREAL_CONST.DEFAULT, // If the value is LREAL_CONST.DEFAULT,
                            // the standard deceleration from the axis configuration is used.
        Jerk            := LREAL_CONST.DEFAULT, // If the value is LREAL_CONST.DEFAULT,
                            // the standard jerk from the axis configuration is applied.
        BufferMode      := Tc3_Mc3Base.EBufferMode.Buffered,
        Done            => ,
        Busy            => ,
        Active          => ,
        CommandAborted  => ,
        Error           => ,
        ErrorId         => );
```

6. When calling `moveRelative`, adjust the distance and dynamics to match your actual axis if you are not using a simulation axis.
7. Build the PLC project.
   ⇨ The `axis` instance of the `AXIS_REF` should now be displayed among the PLC instances in the Solution Explorer.
8. Link the PLC instance of the `AXIS_REF` to the axis object in the MC project (see Link axis object [▶ 117]).
9. Activate the TwinCAT project  . Log in to the PLC  and start it  .
10. Check that the distance specified for `moveRelative` can be traveled safely with the specified dynamics. Adjust them as needed.
11. If you are sure that the movement specified for `moveRelative` does not pose any danger, you can enable the controller for the axis by setting the `enableAxis` variable online to `TRUE`.
12. Check whether the controller was successfully enabled (`power.Status = TRUE`) and follow the instructions regarding the axis's operational readiness [▶ 249]. You can then activate the motion command with `executeMove`.
    ⇨ You can enable continuous writing of the override by setting `setOverride.Enable := TRUE`. The override factor is adjusted via `velFactor`.
    ⇨ Via `stop.Execute = TRUE` the motion command can be stopped prematurely if required.
    ⇨ If an axis error occurs, it can be reset via `reset.Execute := TRUE`.

5.3.3 Data Types

5.3.3.1 Enums

5.3.3.1.1 EAxisParameterId

Parameter IDs for reading and writing axis parameters via MC_ReadParameter / MC_WriteParameter (MC_ReadBoolParameter / MC_WriteBoolParameter).

Syntax

Definition:

```
TYPE EAxisParameterId :
(
    MaximumVelocity                             := 0x0000000005093009,
    MaximumAcceleration                         := 0x000000000509300A,
    MaximumDeceleration                         := 0x000000000509300B,
    DefaultAcceleration                         := 0x000000000509300C,
    DefaultDeceleration                         := 0x000000000509300D,
    DefaultJerk                                 := 0x000000000509300E,
    MaximumJerk                                 := 0x0000000005093015,
    MinimumPosition                             := 0x0000000005093021,
    MaximumPosition                             := 0x0000000005093023,
    PositionModulus                             := 0x0000000005030107,
    PositionModuloToleranceWindow               := 0x0000000005030108,
    MaximumPositionLagValue                     := 0x000000000503011B,
    MaximumPositionLagFilterTime                := 0x000000000503011C,
    JogVelocitySlow                             := 0x000000000503019E,
    JogVelocityFast                             := 0x000000000503019F,
    MinimumPositionLag                          := 0x00000000050301A1,
    MaximumPositionLag                          := 0x00000000050301A2,
    VelocityOverrideFactor                      := 0x0000000005030231,
    SimulationMode                              := 0x00000000050300B1,
    IsModuloAxis                                := 0x0000000005030101,
    PositionLagMonitoringEnabled                := 0x000000000503011A,
    PositionLimitEnforcementEnabled             := 0x000000000503015D,
    DriveVelocityScalingNumeratorToHardware     := 0x0000000105030122,
    DriveVelocityScalingDenominatorToHardware   := 0x00000001050301A7,
    DriveTorqueScalingFromHardware              := 0x000000010503014F,
    DriveTimeoutForEnableAndReset               := 0x000000010503D004,
    DriveActiveHomingOffset                     := 0x0000000105030224,
    DriveDirectionInverted                      := 0x00000001050301B2,
    EncoderPositionScalingNumeratorFromHardware := 0x0000000205030103,
    EncoderPositionScalingDenominatorFromHardware := 0x0000000205030104,
    EncoderMountingOffset                       := 0x0000000205030106,
    EncoderWorkZeroOffset                       := 0x00000002050301D1,
    EncoderDirectionInverted                    := 0x00000002050301B2,
    EncoderIsHomed                              := 0x000000020503D070,
    EncoderIsAbsolutePositionValid              := 0x000000020503D071
)ULINT;
END_TYPE
```

Values

Name	Description
MaximumVelocity	Use MC_ReadParameter, MC_WriteParameter
MaximumAcceleration	Use MC_ReadParameter, MC_WriteParameter
MaximumDeceleration	Use MC_ReadParameter, MC_WriteParameter
DefaultAcceleration	Use MC_ReadParameter, MC_WriteParameter
DefaultDeceleration	Use MC_ReadParameter, MC_WriteParameter
DefaultJerk	Use MC_ReadParameter, MC_WriteParameter
MaximumJerk	Use MC_ReadParameter, MC_WriteParameter
MinimumPosition	Use MC_ReadParameter, MC_WriteParameter
MaximumPosition	Use MC_ReadParameter, MC_WriteParameter
PositionModulus	Use MC_ReadParameter, MC_WriteParameter
PositionModuloToleranceWindow	Use MC_ReadParameter, MC_WriteParameter

Name	Description
MaximumPositionLagValue	Use MC_ReadParameter, MC_WriteParameter
MaximumPositionLagFilterTime	Use MC_ReadParameter, MC_WriteParameter
JogVelocitySlow	Use MC_ReadParameter, MC_WriteParameter
JogVelocityFast	Use MC_ReadParameter, MC_WriteParameter
MinimumPositionLag	Use MC_ReadParameter, MC_WriteParameter
MaximumPositionLag	Use MC_ReadParameter, MC_WriteParameter
VelocityOverrideFactor	Use MC_ReadParameter, MC_WriteParameter
SimulationMode	Use MC_ReadBoolParameter, MC_WriteBoolParameter
IsModuloAxis	Use MC_ReadBoolParameter, MC_WriteBoolParameter
PositionLagMonitoringEnabled	Use MC_ReadBoolParameter, MC_WriteBoolParameter
PositionLimitEnforcementEnabled	Use MC_ReadBoolParameter, MC_WriteBoolParameter
DriveVelocityScalingNumeratorToHardware	Use MC_ReadParameter, MC_WriteParameter
DriveVelocityScalingDenominatorToHardware	Use MC_ReadParameter, MC_WriteParameter
DriveTorqueScalingFromHardware	Use MC_ReadParameter, MC_WriteParameter
DriveTimeoutForEnableAndReset	Use MC_ReadParameter, MC_WriteParameter
DriveActiveHomingOffset	Use MC_ReadParameter, MC_WriteParameter
DriveDirectionInverted	Use MC_ReadBoolParameter, MC_WriteBoolParameter
EncoderPositionScalingNumeratorFromHardware	Use MC_ReadParameter, MC_WriteParameter
EncoderPositionScalingDenominatorFromHardware	Use MC_ReadParameter, MC_WriteParameter
EncoderMountingOffset	Use MC_ReadParameter, MC_WriteParameter
EncoderWorkZeroOffset	Use MC_ReadParameter, MC_WriteParameter
EncoderDirectionInverted	Use MC_ReadBoolParameter, MC_WriteBoolParameter
EncoderIsHomed	Use MC_ReadBoolParameter, MC_WriteBoolParameter
EncoderIsAbsolutePositionValid	Use MC_ReadBoolParameter, MC_WriteBoolParameter

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.2 ECoeTouchProbeInput

Selection for CoE Drive touch probe input.

Syntax

Definition:

```
TYPE ECoeTouchProbeInput :
(
    DriveDefault     := 0,
    ZeroImpulse      := 1,
    DriveDefined     := 2,
    ConfiguredInAxis := 255
)UDINT;
END_TYPE
```

Values

Name	Description
DriveDefault	Trigger on the touch probe default input (i.e., the digital input 1/2 mapped to TouchProbe1/ TouchProbe2 functionality in the drive).
ZeroImpulse	Trigger on the main encoder index signal (encoder Z/ index). Only valid if the drive exposes the index signal to the touch probe function.
DriveDefined	Trigger source is defined by object 0x60D0/0x8001 (DS402/MDP742).
ConfiguredInAxis	The touch probe input source is configured in the axis (TCOM object). The value specified at the CoeDriveSettings input is ignored.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.3 EGearInPosSyncState

Synchronization state reported by MC_GearInPos for the PhysicalAxis. Combines whether the slave is synchronized to the master with whether the master has already passed its master sync position. Reports 'Waiting' while the slave cannot move yet due to a sync direction restriction.

Syntax

Definition:

```
TYPE EGearInPosSyncState :
(
    Invalid                              := 0,
    Waiting                              := 1,
    Synchronizing                        := 2,
    SynchronizingAfterMasterSyncPosition := 3,
    InSyncBeforeMasterSyncPosition       := 4,
    InSync                               := 5
)UDINT;
END_TYPE
```

Values

Name	Description
Invalid	Default value before the function block has been activated.
Waiting	The slave cannot start to move yet because the master is moving against the restricted sync direction.
Synchronizing	The synchronization is not yet complete and the master has not yet passed its master sync position.
SynchronizingAfterMasterSyncPosition	The synchronization is not yet complete; the master has already passed its master sync position.
InSyncBeforeMasterSyncPosition	The slave is locked in to the master, but the master has not yet passed its master sync position.
InSync	The slave is locked in to the master and the master has passed its master sync position.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.4 EHomingMode

Mode selection for functional configuration of MC_SetHomingState command.

Syntax

Definition:

```
TYPE EHomingMode :
(
    ForceIsHomed           := 0,
    ResetIsHomed           := 1,
    ForceIsHomedAtPosition := 2
)UDINT;
END_TYPE
```

Values

Name	Description
ForceIsHomed	Sets the homing state to homed (IsPositionValid = TRUE) without performing any movement or position change.
ResetIsHomed	Resets the homing state to not homed (IsPositionValid = FALSE).
ForceIsHomedAtPosition	Sets the current position to the specified homing position and marks the axis as homed (IsPositionValid = TRUE).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.5 EIncrementalEncoderTouchProbeSource

Selection for incremental encoder touch probe latch source.

Syntax

Definition:

```
TYPE EIncrementalEncoderTouchProbeSource :
(
    LatchC              := 0,
    LatchExternPositive := 1,
    LatchExternNegative := 2,
    ConfiguredInAxis    := 255
)UDINT;
END_TYPE
```

Values

Name	Description
LatchC	Internal latch (encoder C-track / index signal).
LatchExternPositive	External latch input, capture on rising edge.
LatchExternNegative	External latch input, capture on falling edge.
ConfiguredInAxis	The touch probe latch source is configured in the axis (TCOM object). The value specified at the IncrementalEncoderSettings input is ignored.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.6 EJogMode

Selects the jog operation mode.

Syntax

Definition:

```
TYPE EJogMode :
(
    StandardSlow := 1,
    StandardFast := 2,
    Continuous   := 3,
    Inching      := 4
)UINT;
END_TYPE
```

Values

Name	Description
StandardSlow	Jog at low velocity using the configured slow jog speed.
StandardFast	Jog at high velocity using the configured fast jog speed.
Continuous	Jog continuously at the commanded velocity while the jog input is active.
Inching	Move a fixed incremental distance for each jog activation.

Further information

The EJogMode Enum is used in connection with the MC_Jog [▶ 265] function block to define its operating mode.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.7 EModuloDirection

Direction of travel for modulo positioning.

Syntax

Definition:

```
TYPE EModuloDirection :
(
    ShortestDistance  := 0,
    PositiveDirection := 1,
    NegativeDirection := 2
)UINT;
END_TYPE
```

Values

Name	Description
ShortestDistance	Move in the direction that results in the shortest travel distance.
PositiveDirection	Always move in the positive direction to reach the target.
NegativeDirection	Always move in the negative direction to reach the target.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.1.8 EReadMode

Selects the read operation mode.

Syntax

Definition:

```
TYPE EReadMode :
(
    Once   := 1,
    Cyclic := 2
)UDINT;
END_TYPE
```

Values

Name	Description
Once	Read the parameter value once.
Cyclic	Read the parameter value cyclically.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2 Structs

5.3.3.2.1 AXIS_REF_TOMC

Contains the information of the AXIS_REF passed from PLC to MC.

Syntax

Definition:

```
TYPE AXIS_REF_TOMC :
STRUCT
    Std : CDT_PLCTOMC_AXIS_STD;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Std	CDT_PLCTOMC_AXIS_STD	Standard data from PLC to MC.

Further information

Cyclic Data Types (CDTs) are defined globally and are configured in the Data Area [▶ 32] of a Motion object.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.2 AXIS_REF_TOPLC

Contains the information of the AXIS_REF passed from MC to PLC.

Syntax

Definition:

```
TYPE AXIS_REF_TOPLC :
STRUCT
    Std         : Reference To CDT_MCTOPLC_AXIS_STD;
    Set         : Reference To CDT_MCTOPLC_AXIS_SET;
    Act         : Reference To CDT_MCTOPLC_AXIS_ACT;
    Modulo      : Reference To CDT_MCTOPLC_AXIS_MODULO;
    LoadControl : Reference To CDT_MCTOPLC_AXIS_LOADCONTROL;
    FluidPower  : Reference To CDT_MCTOPLC_AXIS_FLUIDPOWER;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Std	Reference To CDT_MCTOPLC_AXIS_STD	Reference to standard data from MC to PLC.
Set	Reference To CDT_MCTOPLC_AXIS_SET	Reference to setpoint data from MC
Act	Reference To CDT_MCTOPLC_AXIS_ACT	Reference to actpoint data from MC
Modulo	Reference To CDT_MCTOPLC_AXIS_MODULO	Reference to modulo positions of a modulo axis
LoadControl	Reference To CDT_MCTOPLC_AXIS_LOADCONTROL	Reference to loadcontrol data from MC
FluidPower	Reference To CDT_MCTOPLC_AXIS_FLUIDPOWER	Reference to fluidpower data from MC

Further information

Cyclic Data Types (CDTs) are defined globally and are configured in the Data Area [▶ 32] of a Motion object.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.3 CoeTouchProbeSettings

CoE drive-specific touch probe source settings.

Syntax

Definition:

```
TYPE CoeTouchProbeSettings :
STRUCT
    TouchProbe1Source : ECoeTouchProbeInput;
    TouchProbe2Source : ECoeTouchProbeInput;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
TouchProbe1Source	ECoeTouchProbeInput [▶ 235]	ConfiguredInAxis	Input source for touch probe 1.
TouchProbe2Source	ECoeTouchProbeInput [▶ 235]	ConfiguredInAxis	Input source for touch probe 2.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.4 ENCODER_AXIS_REF_TOPLC

Contains the information of the ENCODER_AXIS_REF passed from MC to PLC.

Syntax

Definition:

```
TYPE ENCODER_AXIS_REF_TOPLC :
STRUCT
    Std    : Reference To CDT_MCTOPLC_ENCODER_AXIS_STD;
    Act    : Reference To CDT_MCTOPLC_ENCODER_AXIS_ACT;
    Modulo : Reference To CDT_MCTOPLC_ENCODER_AXIS_MODULO;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Std	Reference To CDT_MCTOPLC_ENCODER_AXIS_STD	Reference to standard data from MC to PLC.
Act	Reference To CDT_MCTOPLC_ENCODER_AXIS_ACT	Reference to actual data from MC.
Modulo	Reference To CDT_MCTOPLC_ENCODER_AXIS_MODULO	Reference to modulo positions from encoder axis.

Further information

Cyclic Data Types (CDTs) are defined globally and are configured in the Data Area [▶ 32] of a Motion object.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.5 FixedStopTouchProbeSettings

Configure up to three thresholds for fixed stop detection (Torque, Velocity, Lag). Enter #Default to use the axis default value, or #Ignore to disable a threshold. Typical detection criteria: combination of high torque, low velocity, and/or increasing lag.

Syntax

Definition:

```
TYPE FixedStopTouchProbeSettings :
STRUCT
    TorqueThreshold   : LREAL;
    VelocityThreshold : LREAL;
    LagThreshold      : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
TorqueThreshold	LREAL	#Default	Threshold for the actual motor torque. If the torque exceeds this value during evaluation, it contributes to a fixed stop decision. Enter #Ignore to disable this threshold.
VelocityThreshold	LREAL	#Default	Upper limit for the actual axis velocity during evaluation. Fixed stop detection is evaluated only while the absolute velocity is below this value. Enter #Ignore to disable this threshold.
LagThreshold	LREAL	#Default	Threshold for the position following error (command position minus actual position). If the absolute following error exceeds this value during evaluation, it contributes to a fixed stop decision. Enter #Ignore to disable this threshold.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.6 IncrementalEncoderTouchProbeSettings

Touch probe settings for incremental encoder terminals.

Syntax

Definition:

```
TYPE IncrementalEncoderTouchProbeSettings :
STRUCT
    Source : EIncrementalEncoderTouchProbeSource;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
Source	EIncrementalEncoderTouchProbeSource [▶ 237]	ConfiguredInAxis	Latch source override for position capture.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.7 TouchProbeOptions

MC_TouchProbe Options

Syntax

Definition:

```
TYPE TouchProbeOptions :
STRUCT
    WindowFirstPosition     : LREAL;
    WindowLastPosition      : LREAL;
    PositionsAsModuloValues : BOOL;
    Continuous              : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
WindowFirstPosition	LREAL	#Ignore	Start position from where trigger events are accepted ([PosUnit], included in the position window). It can be left as 'Ignore' if no lower limit is desired. Must have value <= WindowLastPosition otherwise.
WindowLastPosition	LREAL	#Ignore	Stop position up to where trigger events are accepted ([PosUnit], included in the position window). It can be left as 'Ignore' if no upper limit is desired. Must have value >= WindowFirstPosition otherwise.
PositionsAsModuloValues	BOOL	false	Calculate RecordedPosition as modulo value.
Continuous	BOOL	false	Continuous recording mode

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.8 TouchProbeUnit

Contains trigger references for positive and negative edge touch probes.

Syntax

Definition:

```
TYPE TouchProbeUnit :
STRUCT
    PositiveTrigger : TRIGGER_REF;
    NegativeTrigger : TRIGGER_REF;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
PositiveTrigger	TRIGGER_REF [▶ 244]	Trigger reference for positive edge events.
NegativeTrigger	TRIGGER_REF [▶ 244]	Trigger reference for negative edge events.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.3.2.9 TRIGGER_REF

Syntax

Definition:

```
TYPE TRIGGER_REF :
STRUCT
    Target                    : UDINT;
    ProbeId                   : UINT;
    Edge                      : UDINT;
    ObjId                     : PVOID;
    SettingsCoe               : PVOID;
    SettingsFixedStop         : PVOID;
    SettingsIncrementalEncoder : PVOID;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Target	UDINT	
ProbeId	UINT	
Edge	UDINT	
ObjId	PVOID	
SettingsCoe	PVOID	
SettingsFixedStop	PVOID	
SettingsIncrementalEncoder	PVOID	

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.4 Interfaces

5.3.4.1 Types

5.3.4.1.1 Type_PtpErrorResettable

Type for a motion control object whose errors can be reset (e. g. supported by AXIS_REF).

Inheritance Hierarchy

Type_PtpErrorResettable

Methods

Name	Origin	Description
GetPtpErrorResettableOid [▶ 245]	Type_PtpErrorResettable	

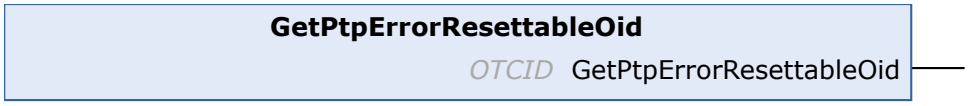
Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.4.1.1.1 GetPtpErrorResettableOid



Syntax

Definition:

```
METHOD GetPtpErrorResettableOid : OTCID
```

Return value

OTCID

5.3.4.1.2 Type_PtpReadWriteParameter

Type for a motion control object that supports parameter read/write access (e.g. supported by AXIS_REF and ENCODER_AXIS_REF).

Inheritance Hierarchy

Type_PtpReadWriteParameter

Methods

Name	Origin	Description
GetPtpReadWriteParameterOid [▶ 246]	Type_PtpReadWriteParameter	

Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.4.1.2.1 GetPtpReadWriteParameterOid

Syntax

Definition:

```
METHOD GetPtpReadWriteParameterOid : OTCID
```

Return value

OTCID

5.3.4.1.3 Type_PtpSetHomingState

Type for a motion control object that supports setting the homing state (e. g. supported by AXIS_REF and ENCODER_AXIS_REF).

Inheritance Hierarchy

Type_PtpSetHomingState

Methods

Name	Origin	Description
GetPtpSetHomingStateOid [▶ 246]	Type_PtpSetHomingState	

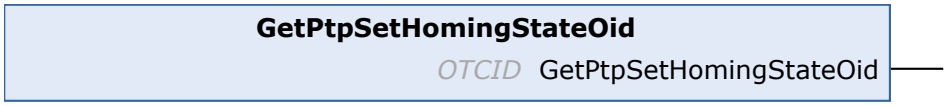
Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.4.1.3.1 GetPtpSetHomingStateOid



Syntax

Definition:

```
METHOD GetPtpSetHomingStateOid : OTCID
```

Return value

OTCID

5.3.4.1.4 Type_PtpSetPosition

Type for a motion control object that supports setting position (e. g. supported by AXIS_REF).

Inheritance Hierarchy

Type_PtpSetPosition

Methods

Name	Origin	Description
GetPtpSetPositionOid [▶ 247]	Type_PtpSetPosition	

Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.4.1.4.1 GetPtpSetPositionOid

Syntax

Definition:

```
METHOD GetPtpSetPositionOid : OTCID
```

Return value

OTCID

5.3.4.1.5 Type_PtpTouchProbe

Type for a motion control object that supports touch probe commands (e. g. supported by AXIS_REF and ENCODER_AXIS_REF).

Inheritance Hierarchy

Type_PtpTouchProbe

Methods

Name	Origin	Description
GetPtpTouchProbeOid [▶ 248]	Type_PtpTouchProbe	

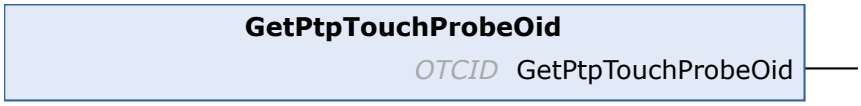
Uses of the type

- Overview: MC3 types [▶ 214]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.4.1.5.1 GetPtpTouchProbeOid



Syntax

Definition:

```
METHOD GetPtpTouchProbeOid : OTCID
```

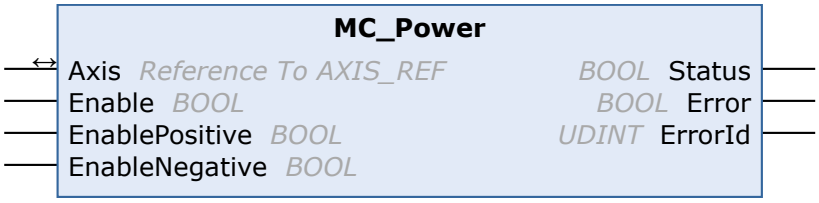
Return value

OTCID

5.3.5 Function Blocks

5.3.5.1 Administrative

5.3.5.1.1 MC_Power



This function block controls the power stage (on or off).

Syntax

Definition:

```
FUNCTION_BLOCK MC_Power
VAR_IN_OUT
    Axis           : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Enable         : BOOL;
    EnablePositive : BOOL;
    EnableNegative : BOOL;
END_VAR
VAR_OUTPUT
    Status         : BOOL;
    Error          : BOOL;
    ErrorId        : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Enable	BOOL		General software enable for the axis
EnablePositive	BOOL	false	Feed enable in positive direction. Only effective if Enable = TRUE
EnableNegative	BOOL	false	Feed enable in negative direction. Only effective if Enable = TRUE

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Status	BOOL	Axis is ready for motion commands.
Error	BOOL	Error occurred within function block or axis already signaling an error.
ErrorId	UDINT	Error identifier of function block or axis

Further information

The MC_Power function block is used to enable an axis in the software. The enable can be applied to both directions or only to a specific direction. A `TRUE` at the `Status` output indicates that the axis is ready for operation.

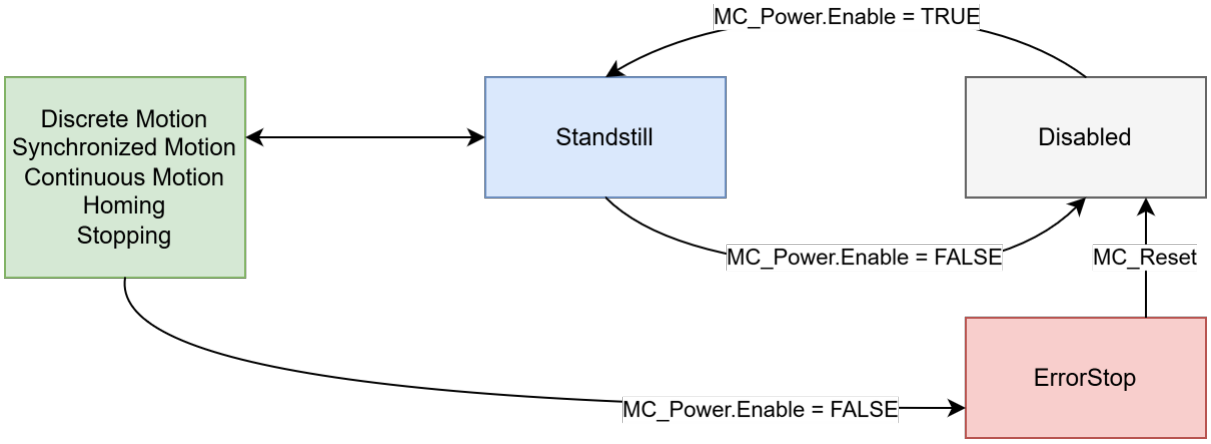
Depending on the drive type, `Status` also indicates that the drive is ready for operation. Digital drives provide feedback on operational readiness, while analog drives are unable to indicate their operational readiness. In the latter case Status only indicated operational readiness of the control side.

Hardware enable signal

In addition to software enable it may be necessary to activate a hardware enable signal in order to enable a drive. This signal is not influenced by MC_Power and must be activated separately by the PLC.

Effects on the axis state

MC_Power.Enable affects the axis state as follows:



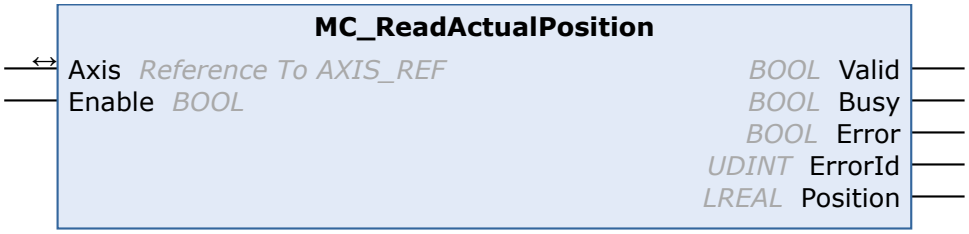
Note on the migration from NC2 to MC3

For MC3 axes, the override is set to 100% by default. The MC_SetOverride [▶ 256] function block is available for setting the override.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.2 MC_ReadActualPosition



This function block reads the current axis actual position.
In case of an invalid actual position value, the Position output is set to 0 and the Valid output is set to FALSE.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadActualPosition
VAR_IN_OUT
    Axis     : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Enable   : BOOL;
END_VAR
VAR_OUTPUT
    Valid    : BOOL;
    Busy     : BOOL;
    Error    : BOOL;
    ErrorId  : UDINT;
    Position : LREAL;
END_VAR
```

Inputs

Name	Type	Description
Enable	BOOL	Read the actual position continuously while Enable = TRUE.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
Position	LREAL	Actual position ([PosUnit])

Further information

Alternatively, the actual position can be read from the PLC using the AXIS_REF structure.

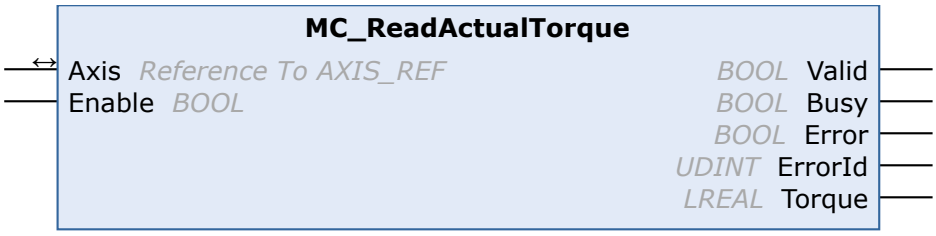
```
myAxis.McToPlc.Act.Position
```

Version information

- TwinCAT Standard >= v3.1.4026.23.1

- TF5500 MC3 Base >= v4.0.6

5.3.5.1.3 MC_ReadActualTorque



This function block reads the current axis actual torque.
In case of an invalid actual torque value, the Torque output is set to 0 and the Valid output is set to FALSE.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadActualTorque
VAR_IN_OUT
    Axis    : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Enable  : BOOL;
END_VAR
VAR_OUTPUT
    Valid   : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorId : UDINT;
    Torque  : LREAL;
END_VAR
```

Inputs

Name	Type	Description
Enable	BOOL	Read the actual torque continuously while Enable = TRUE.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
Torque	LREAL	Actual torque ([TorqueUnit])

Further information

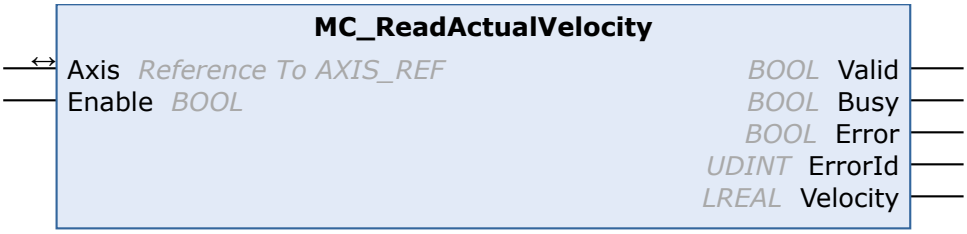
Alternatively, the torque can be read from the PLC using the AXIS_REF structure.

```
myAxis.McToPlc.Act.Torque
```

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.4 MC_ReadActualVelocity



This function block reads the current axis actual velocity.
In case of an invalid actual velocity value, the Velocity output is set to 0 and the Valid output is set to FALSE.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadActualVelocity
VAR_IN_OUT
    Axis     : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Enable   : BOOL;
END_VAR
VAR_OUTPUT
    Valid    : BOOL;
    Busy     : BOOL;
    Error    : BOOL;
    ErrorId  : UDINT;
    Velocity : LREAL;
END_VAR
```

Inputs

Name	Type	Description
Enable	BOOL	Read the actual velocity continuously while Enable = TRUE.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
Velocity	LREAL	Actual velocity ([PosUnit]/s)

Further information

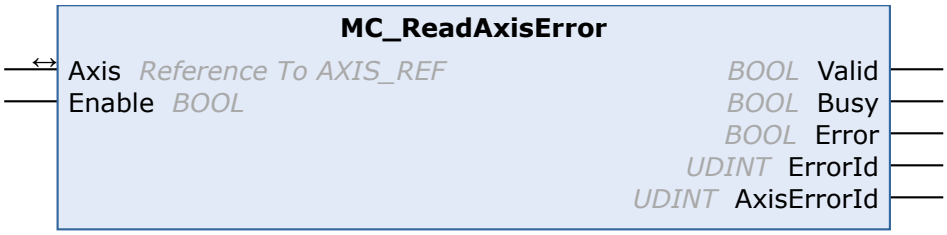
Alternatively, the actual velocity can be read from the PLC using the AXIS_REF structure.

```
myAxis.McToPlc.Act.Velocity
```

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.5 MC_ReadAxisError



This function block reads the current axis error identifier.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadAxisError
VAR_IN_OUT
    Axis        : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Enable      : BOOL;
END_VAR
VAR_OUTPUT
    Valid       : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorId     : UDINT;
    AxisErrorId : UDINT;
END_VAR
```

Inputs

Name	Type	Description
Enable	BOOL	Read the axis error identifier continuously while Enable = TRUE.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
AxisErrorId	UDINT	Axis error identifier

Further information

Alternatively, the axis status can be read from the PLC using the AXIS_REF structure.

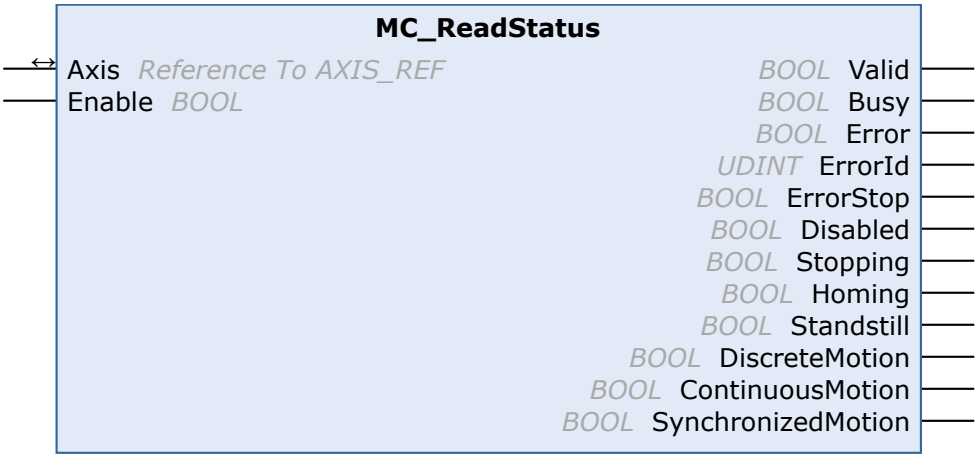
```
myAxis.McToPlc.Std.AxisState
myAxis.McToPlc.Std.HasError
myAxis.McToPlc.Std.ErrorId
```

Information about the individual states is summarized in the section State diagram [▶ 229].

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.6 MC_ReadStatus



This function block reads the current axis state.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadStatus
VAR_IN_OUT
    Axis                : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Enable              : BOOL;
END_VAR
VAR_OUTPUT
    Valid               : BOOL;
    Busy                : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
    ErrorStop           : BOOL;
    Disabled            : BOOL;
    Stopping            : BOOL;
    Homing              : BOOL;
    Standstill          : BOOL;
    DiscreteMotion      : BOOL;
    ContinuousMotion    : BOOL;
    SynchronizedMotion  : BOOL;
END_VAR
```

Inputs

Name	Type	Description
Enable	BOOL	Read the axis state continuously while Enable = TRUE.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
ErrorStop	BOOL	Axis is in state ErrorStop.
Disabled	BOOL	Axis is in state Disabled.
Stopping	BOOL	Axis is in state Stopping.
Homing	BOOL	Axis is in state Homing.
Standstill	BOOL	Axis is in state Standstill.
DiscreteMotion	BOOL	Axis is in state DiscreteMotion.
ContinuousMotion	BOOL	Axis is in state ContinuousMotion.
SynchronizedMotion	BOOL	Axis is in state SynchronizedMotion.

Further information

Alternatively, the axis status can be read from the PLC using the AXIS_REF structure.

```
myAxis.McToPlc.Std.AxisState
```

Information about the individual states is summarized in the section State diagram [▶ 229].

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.7 MC_Reset

This function block resets all motion object-related errors and transitions the motion object out of state ErrorStop.
The resulting state is Disabled, if no Enable is requested for the motion object (e.g. via MC_Power).
It results in DiscreteMotion, if the motion object has a closed control loop and a controlled error reaction is still ongoing.
Otherwise, Standstill is the resulting state. This function block does not affect the output of motion function block instances.

Syntax

Definition:

```
FUNCTION_BLOCK MC_Reset
VAR_INPUT
    Axis    : Type_PtpErrorResettable;
    Execute : BOOL;
END_VAR
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorId : UDINT;
END_VAR
```

Inputs

Name	Type	Description
Axis	Type_PtpErrorResettable [▶ 244]	Interface pointer to the object on which the command should be performed (e.g. Axis, Linear Mover).
Execute	BOOL	Trigger the command with rising edge.

Outputs

Name	Type	Description
Done	BOOL	Potential axis and/or drive error has been reset. State Standstill, Disabled or DiscreteMotion is reached.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Notes on the migration from NC2 to MC3

The MC3 function block MC_Reset only takes effect in the event of an error.

The MC3 MC_Reset function block does not reset an existing axis position lag.

If the connected drive hardware is also in an error state, a reset command is automatically sent to the hardware when the MC3 is reset. It is not necessary to perform a manual FB_SoEReset.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.8 MC_SetOverride

This function block sets the velocity override factor that is applied to movement commands of the axis. The override factor does not apply to movement in the synchronized motion state (e.g. movement induced by MC_GearIn, MC_GearInPos, ...).

Syntax

Definition:

```
FUNCTION_BLOCK MC_SetOverride
VAR_IN_OUT
    Axis           : Reference To AXIS_REF;
END_VAR
```

```
VAR_INPUT
    Enable         : BOOL;
    VelocityFactor : LREAL;
END_VAR
VAR_OUTPUT
    Enabled        : BOOL;
    Error          : BOOL;
    ErrorId        : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Enable	BOOL		Write the velocity override factor continuously while Enable = TRUE.
VelocityFactor	LREAL	1.0	Applies to commanded velocities. Valid range is [0,1]. Use STRING_TO_LREAL('#Ignore') to NOT overwrite the previous value.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Enabled	BOOL	The commanded override is set.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

The MC3 axis supports a velocity override ranging from 0 to 100%, which is normalized to a range of 0 to 1. By default, the velocity override (`VelocityFactor`) is set to 1, which corresponds to 100%. The MC_SetOverride function block can be used to adjust the velocity override within the permissible range.

The velocity override allows you to adjust the velocity of an axis without changing the commanded dynamics (velocity, acceleration, and deceleration). This does not change the underlying motion curve either.

When using couplings, keep in mind that only the master's override is taken into account.

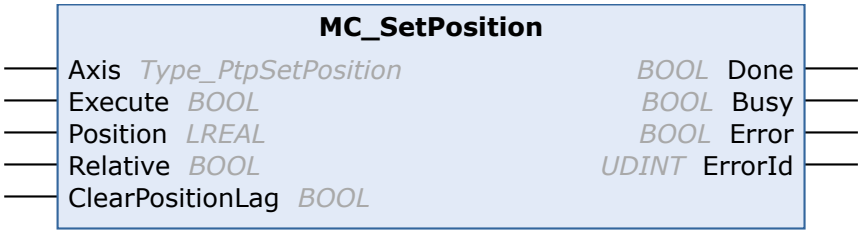
Note on the migration from NC2 to MC3

NC2 and MC3 support a velocity override ranging from 0 to 100%.
In the NC2, the velocity override is specified as a percentage and is set to 0% by default.
In contrast, in the MC3, the value range is normalized to 0 to 1, and the default value is 1, which corresponds to 100%.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.1.9 MC_SetPosition



This function block shifts the coordinate system of an axis by manipulating both the setpoint position as well as the actual position of an axis with the same value without any movement caused (Re-calibration with same following error). This can be used for instance for a reference situation.

Syntax

Definition:

```
FUNCTION_BLOCK MC_SetPosition
VAR_INPUT
    Axis             : Type_PtpSetPosition;
    Execute          : BOOL;
    Position         : LREAL;
    Relative         : BOOL;
    ClearPositionLag : BOOL;
END_VAR
VAR_OUTPUT
    Done             : BOOL;
    Busy             : BOOL;
    Error            : BOOL;
    ErrorId          : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Axis	Type_PtpSetPosition [▶ 246]		Reference to the axis
Execute	BOOL		Trigger the command with rising edge.
Position	LREAL	#Invalid	New position value ([PosUnit]) or distance if Relative = TRUE
Relative	BOOL	false	Shift axis position by value
ClearPositionLag	BOOL		Resets position lag to zero.

Outputs

Name	Type	Description
Done	BOOL	Position has new value.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

The MC_SetPosition function block sets the current axis position to a parameterizable value. To do this, the axis must be at a standstill.

In the default case (input `Relative = FALSE`), the actual position is set to the parameterized absolute value of input `Position`. In the relative case (input `Relative = TRUE`), the actual position is offset by the value of the `Position` input. In both cases, the target position of the axis is set so that any potential position lag is retained. If the position lag is to be cleared, this can be done via the `Options.ClearPositionLag` input.

Note on the migration from NC2 to MC3

The `Relative` input corresponds to the `Mode` input of the `Tc2_MC2.MC_SetPosition` function block.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.2 Administrative/Parameter

5.3.5.2.1 MC_ReadBoolParameter

This function block reads a boolean parameter value from the axis.
The command is typically processed synchronously.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadBoolParameter
VAR_INPUT
    Axis        : Type_PtpReadWriteParameter;
    Enable      : BOOL;
    ParameterId : EAxisParameterId;
    Mode        : EReadMode;
END_VAR
VAR_OUTPUT
    Valid       : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorId     : UDINT;
    Value       : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Axis	Type_PtpReadWriteParameter [▶ 245]		Reference to the axis
Enable	BOOL		Triggers a read on rising edge (EReadMode.Once) or every cycle while TRUE (EReadMode.Cyclic).
ParameterId	EAxisParameterId [▶ 234]		Identifier of the parameter to read.
Mode	EReadMode [▶ 239]	Once	Once: read once at rising edge of Enable. Cyclic: read every cycle while Enable is TRUE.

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block.
Busy	BOOL	Function block is not finished. Always FALSE with ReadMode Once.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
Value	BOOL	Last successfully read value of the parameter.

Further information

Alternatively, the axis status can be read from the PLC using the AXIS_REF structure.

```
myAxis.McToPlc.Std.AxisState
```

Information about the individual states is summarized in the section State diagram [▶ 229].

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.2.2 MC_ReadHardwareParameter

This function block allows to read parameters from the hardware drive asynchronously via ADS. The function block supports CoE and SoE drives.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadHardwareParameter
VAR_IN_OUT
    Axis               : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute            : BOOL;
    Index              : UINT;
    SubIndex           : USINT;
    DestinationAddress : PVOID;
    DataSize           : UDINT;
END_VAR
VAR_OUTPUT
    Done               : BOOL;
    Busy               : BOOL;
    Active             : BOOL;
    Error              : BOOL;
    ErrorId            : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.

Name	Type	Default	Description
Index	UINT	0	CoE object index or SoE IDN
SubIndex	USINT	0	CoE sub index or SoE element
DestinationAddress	PVOID	0	Contains the address of the destination buffer.
DataSize	UDINT	0	Contains the size of the destination buffer.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

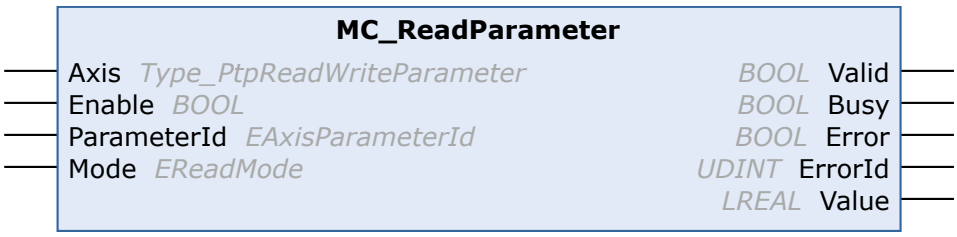
Outputs

Name	Type	Description
Done	BOOL	Value is successfully read from the target.
Busy	BOOL	Function block is not finished.
Active	BOOL	Function block is active.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.2.3 MC_ReadParameter



This function block reads a numeric (LREAL) parameter value from the axis.
The command is typically processed synchronously.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ReadParameter
VAR_INPUT
    Axis        : Type_PtpReadWriteParameter;
    Enable      : BOOL;
    ParameterId : EAxisParameterId;
    Mode        : EReadMode;
END_VAR
VAR_OUTPUT
    Valid       : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorId     : UDINT;
    Value       : LREAL;
END_VAR
```

Inputs

Name	Type	Default	Description
Axis	Type_PtpReadWriteParameter [▶ 245]		Reference to the axis
Enable	BOOL		Triggers a read on rising edge (EReadMode.Once) or every cycle while TRUE (EReadMode.Cyclic).
ParameterId	EAxisParameterId [▶ 234]		Identifier of the parameter to read.
Mode	EReadMode [▶ 239]	Once	Once: read once at rising edge of Enable. Cyclic: read every cycle while Enable is TRUE.

Outputs

Name	Type	Description
Valid	BOOL	Valid output available at function block.
Busy	BOOL	Function block is not finished. Always FALSE with ReadMode Once.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
Value	LREAL	Last successfully read value of the parameter.

Further information

Alternatively, the axis status can be read from the PLC using the AXIS_REF structure.

```
myAxis.McToPlc.Std.AxisState
```

Information about the individual states is summarized in the section State diagram [▶ 229].

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.2.4 MC_WriteBoolParameter

This function block writes a boolean parameter value to the axis.
The command is typically processed synchronously.

Syntax

Definition:

```
FUNCTION_BLOCK MC_WriteBoolParameter
VAR_INPUT
    Axis        : Type_PtpReadWriteParameter;
    Execute     : BOOL;
    ParameterId : EAxisParameterId;
    Value       : BOOL;
END_VAR
VAR_OUTPUT
```

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

Inputs

Name	Type	Description
Axis	Type_PtpReadWriteParameter [▶ 245]	Reference to the axis
Execute	BOOL	Trigger the command with rising edge.
ParameterId	EAxisParameterId [▶ 234]	Identifier of the parameter to write.
Value	BOOL	Value of the parameter to write.

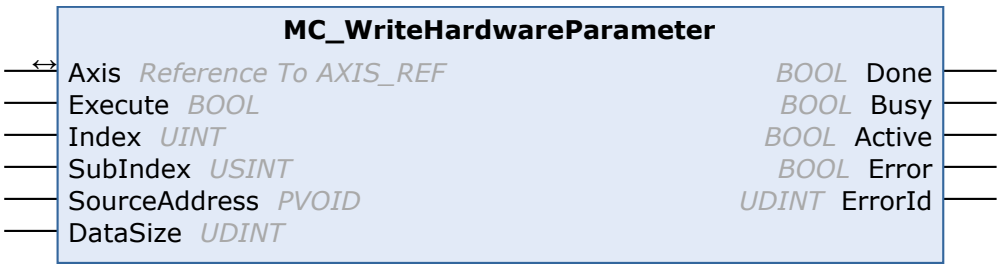
Outputs

Name	Type	Description
Done	BOOL	Parameter value has been successfully written.
Busy	BOOL	Function block is not finished.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.2.5 MC_WriteHardwareParameter



This function block allows to write parameters to the hardware drive asynchronously via ADS. It supports CoE and SoE drives.

Syntax

Definition:

```
FUNCTION_BLOCK MC_WriteHardwareParameter
VAR_IN_OUT
    Axis          : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute       : BOOL;
    Index         : UINT;
    SubIndex      : USINT;
    SourceAddress : PVOID;
    DataSize      : UDINT;
END_VAR
VAR_OUTPUT
    Done          : BOOL;
    Busy          : BOOL;
    Active        : BOOL;
```

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

Inputs

Name	Type	Default	Description
Execute	BOOL		Rising edge triggers the write operation.
Index	UINT	0	CoE object index or SoE IDN
SubIndex	USINT	0	CoE sub index or SoE element
SourceAddress	PVOID	0	Contains the address of the data.
DataSize	UDINT	0	Contains the size of the data.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Value is successfully sent to the target.
Busy	BOOL	Function block is not finished.
Active	BOOL	Function block is active.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.2.6 MC_WriteParameter

This function block writes a numeric (LREAL) parameter value to the axis.
The command is typically processed synchronously.

Syntax

Definition:

```
FUNCTION_BLOCK MC_WriteParameter
VAR_INPUT
    Axis        : Type_PtpReadWriteParameter;
    Execute     : BOOL;
    ParameterId : EAxisParameterId;
    Value       : LREAL;
END_VAR
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
```

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

Inputs

Name	Type	Description
Axis	Type_PtpReadWriteParameter [▶ 245]	Reference to the axis
Execute	BOOL	Trigger the command with rising edge.
ParameterId	EAxisParameterId [▶ 234]	Identifier of the parameter to write.
Value	LREAL	Value of the parameter to write.

Outputs

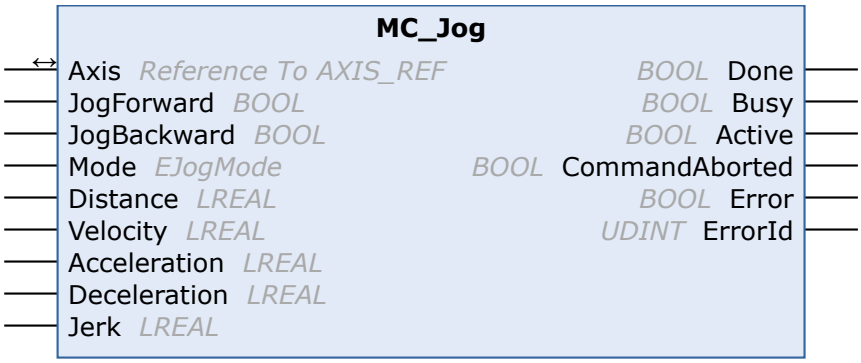
Name	Type	Description
Done	BOOL	Parameter value has been successfully written.
Busy	BOOL	Function block is not finished.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.3 Continuous Motion

5.3.5.3.1 MC_Jog



This function block enables an axis to be moved via manual keys in different modi.

Syntax

Definition:

```
FUNCTION_BLOCK MC_Jog
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    JogForward      : BOOL;
    JogBackward     : BOOL;
    Mode            : EJogMode;
    Distance        : LREAL;
    Velocity        : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
```

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

Inputs

Name	Type	Default	Description
JogForward	BOOL		The command is executed with a rising edge, and the axis is moved in positive direction. Depending on the mode, the axis moves as long as the signal remains TRUE, or it stops automatically after a specified distance.
JogBackward	BOOL		The command is executed with a rising edge, and the axis is moved in negative direction. Depending on the mode, the axis moves as long as the signal remains TRUE, or it stops automatically after a specified distance.
Mode	EJogMode [▶ 238]	StandardSlow	Determines the operation mode in which the manual function is executed.
Distance	LREAL	#Ignore	Distance for movements in Mode Inching
Velocity	LREAL	#Ignore	Jog velocity ([PosUnit]/s, positive). Required for Continuous and Inching modes; StandardSlow and StandardFast use axis defaults.
Acceleration	LREAL	#Default	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Function block has succeeded.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

The operating mode of the MC_Jog function block is defined via the input `Mode` of type E_JogMode [▶ 238]:

`E_JogMode.StandardSlow`

The axis moves as long as the signal at either the `JogForward` or `JogBackward` `TRUE` input is TRUE. The velocity specified by the `Jog Velocity Slow` axis parameter and the dynamics specified by the `Acceleration`, `Deceleration`, and `Jerk` inputs are used.

The `Distance` and `Velocity` function block inputs have no effect in this operating mode.

`E_JogMode.StandardFast`

The axis moves as long as the signal at either the `JogForward` or `JogBackward` `TRUE` input is TRUE. The velocity specified by the `Jog Velocity Fast` axis parameter and the dynamics specified by the `Acceleration`, `Deceleration`, and `Jerk` inputs are used.

The `Distance` and `Velocity` function block inputs have no effect in this operating mode.

`E_JogMode.Continuous`

The axis moves as long as the signal at either the `JogForward` or `JogBackward` `TRUE` input is TRUE.

The function block input `Distance` has no effect in this operating mode.

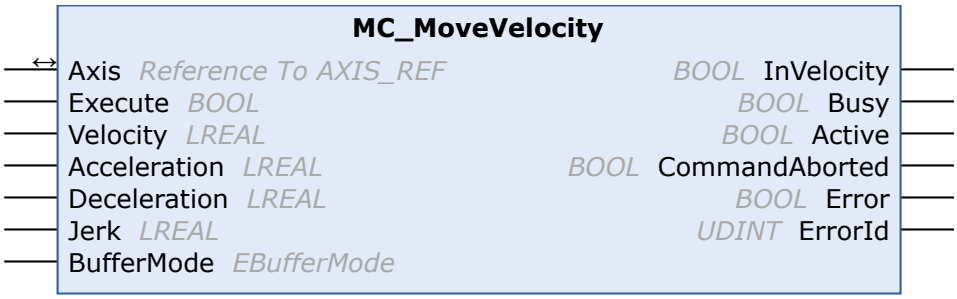
`E_JogMode.Inching`

The axis moves by the distance specified at the `Distance` input of the function block as soon as a rising edge is detected at the `JogForward` or `JogBackward` input.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.3.2 MC_MoveVelocity



This function block commands an endless movement with a specified velocity.

Syntax

Definition:

```
FUNCTION_BLOCK MC_MoveVelocity
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Velocity        : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
    BufferMode      : EBufferMode;
END_VAR
VAR_OUTPUT
    InVelocity      : BOOL;
```

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

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
Velocity	LREAL	#Invalid	Commanded velocity ([PosUnit]/s, the sign of this parameter determines the direction of movement).
Acceleration	LREAL	#Default	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)
BufferMode	EBufferMode	Aborting	Chronological sequence of the command

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
InVelocity	BOOL	Commanded velocity reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

The function block MC_MoveVelocity is used to start an endless travel with a specified velocity and direction. As soon as the specified constant velocity is reached, the `InVelocity` output is set, and the function block has completed its operation. The function block does not perform any further monitoring of the motion. If the command is aborted during the acceleration phase, the `CommandAborted` output or, in the event of an error, the `Error` output is set.

The motion can be paused using MC_Halt [▶ 269] and MC_Stop [▶ 275], or it can be canceled by another motion command.

If a motion command with the input `BufferMode = Tc3_Mc3Base.EBufferMode.Aborting` is applied to a coupled slave axis, this causes the axis to be automatically decoupled, followed by the execution of the command.

Note on the migration from NC2 to MC3

Unlike the NC2 function block of the same name, the MC3 function block MC_MoveVelocity has a signed `Velocity` input. The sign indicates the direction of motion.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.4 Discrete Motion

5.3.5.4.1 MC_Halt

This function block commands a motion stop. The axis is transferred to the state DiscreteMotion, until zero dynamics is reached.

Syntax

Definition:

```
FUNCTION_BLOCK MC_Halt
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Zero dynamics has been reached, axis state is transferred to Standstill.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

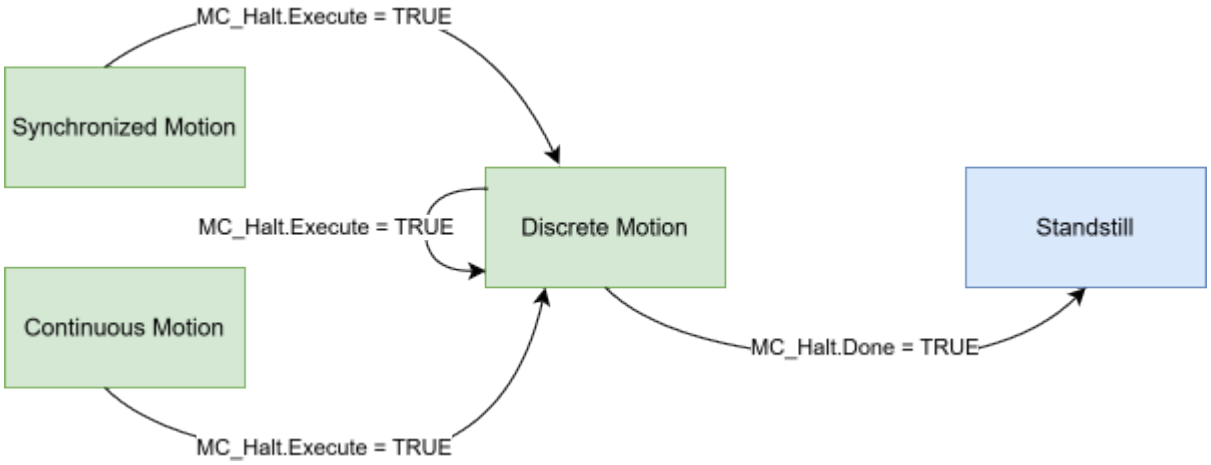
Further information

The function block MC_Halt is used to stop an axis with a defined deceleration ramp.

Unlike MC_Stop [▶ 275], the axis is not locked against further motion commands. This means that the MC_Halt command can be canceled either during the motion ramp or immediately afterward by another motion command.

If MC_Halt is applied to a coupled slave axis, the axis is automatically decoupled, and the halt command is then executed.

Effects on the axis state



Note on the migration from NC2 to MC3

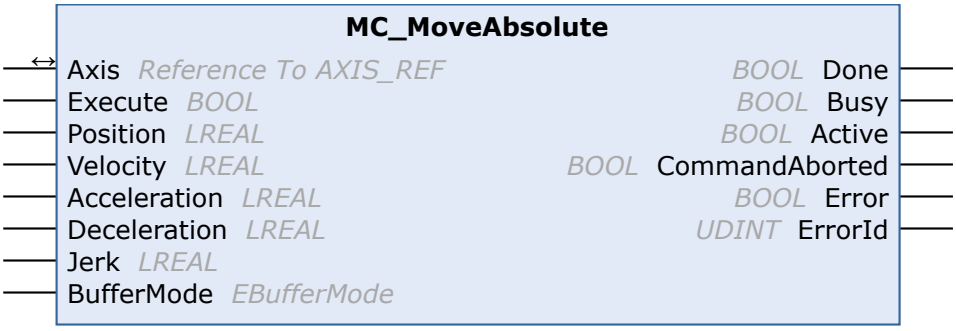
Unlike the NC2 function block of the same name, the MC3 function block MC_Halt does not have an input that can be used to set the buffer mode. In MC3, any previous command is always aborted immediately by MC_Halt, which internally corresponds to the Aborting buffer mode.

In MC3, the parameterization of the dynamics is not possible with a value of 0 (implicit use of the last commanded dynamics).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.4.2 MC_MoveAbsolute



This function block commands a motion to a specified absolute position.

Syntax

Definition:

```
FUNCTION_BLOCK MC_MoveAbsolute
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL;
    Velocity        : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
    BufferMode      : EBufferMode;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
Position	LREAL	#Invalid	Commanded position ([PosUnit])
Velocity	LREAL	#Invalid	Maximum velocity ([PosUnit]/s, positive)
Acceleration	LREAL	#Default	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)
BufferMode	EBufferMode	Aborting	Chronological sequence of the command

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Commanded position finally reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

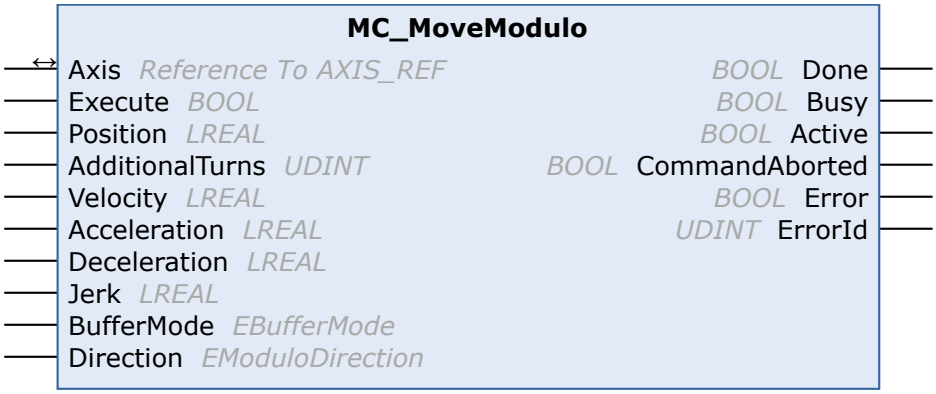
The function block MC_MoveAbsolute starts the positioning to an absolute target position and monitors the axis movement over the entire travel path.

If a motion command with the input `BufferMode = Tc3_Mc3Base.EBufferMode.Aborting` is applied to a coupled slave axis, this causes the axis to be automatically decoupled, followed by the execution of the command.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.4.3 MC_MoveModulo



This function block commands a motion to a specified modulo position.

Syntax

Definition:

```
FUNCTION_BLOCK MC_MoveModulo
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL;
    AdditionalTurns : UDINT;
    Velocity        : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
    BufferMode      : EBufferMode;
    Direction       : EModuloDirection;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
```

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

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
Position	LREAL	#Invalid	Commanded modulo position ([PosUnit])
AdditionalTurns	UDINT	0	Number of additional modulo turns in the commanded direction. A value > 0 requires Direction <> EModuloDirection.ShortestDistance.
Velocity	LREAL	#Invalid	Maximum velocity ([PosUnit]/s, positive)
Acceleration	LREAL	#Default	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)
BufferMode	EBufferMode	Aborting	Chronological sequence of the command
Direction	EModuloDirection [▶ 238]	ShortestDistance	Commanded direction. Must have value <> EModuloDirection.ShortestDistance if AdditionalTurns has value > 0.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Commanded position finally reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

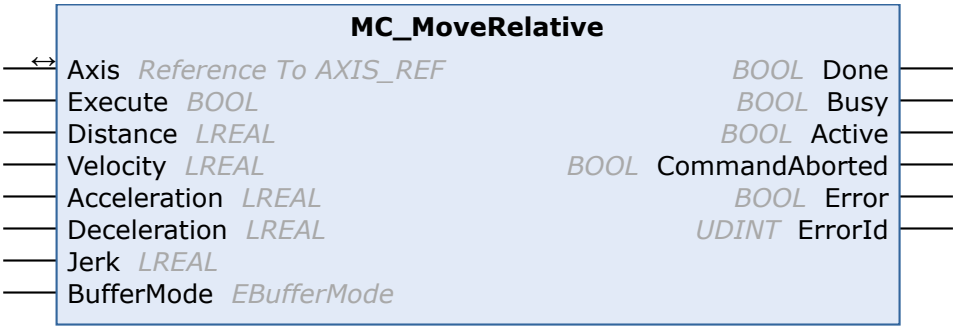
If the Direction input is set to PositiveDirection or NegativeDirection, the AdditionalTurns input is also taken into account. The AdditionalTurns input can be used to specify additional turns until the target position is reached.

Further information on Modulo positioning [▶ 303].

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.4.4 MC_MoveRelative



This function block commands a motion of a specified distance relative to the set position at the time the command gets active.

Syntax

Definition:

```
FUNCTION_BLOCK MC_MoveRelative
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Distance        : LREAL;
    Velocity        : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
    BufferMode      : EBufferMode;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
Distance	LREAL	#Invalid	Relative distance ([PosUnit])
Velocity	LREAL	#Invalid	Maximum velocity ([PosUnit]/s, positive)
Acceleration	LREAL	#Default	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)
BufferMode	EBufferMode	Aborting	Chronological sequence of the command

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

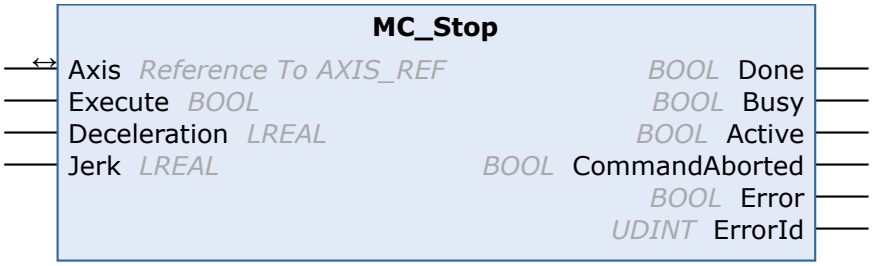
Outputs

Name	Type	Description
Done	BOOL	Commanded distance reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.4.5 MC_Stop



This function block commands a motion stop and transfers the axis to the state Stopping. It aborts any ongoing function block execution. While the axis is in state Stopping, no other function block can perform any motion on the same axis. The Done output is set immediately after zero dynamics is reached. The axis remains in the state Stopping as long as Execute is TRUE or zero dynamics is not yet reached. As soon as Done is SET and Execute is FALSE the axis goes to state Standstill.

Syntax

Definition:

```
FUNCTION_BLOCK MC_Stop
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge. While this is TRUE, other function blocks will be rejected for the axis.

Name	Type	Default	Description
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Zero dynamics has been reached.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

The function block MC_Stop is used to stop an axis with a defined deceleration ramp and to lock the axis against other motion commands. Only when the axis is at a standstill (`MC_Stop.Done = TRUE`) and the input `MC_Stop.Execute = FALSE` is the axis unlocked for other motion commands.

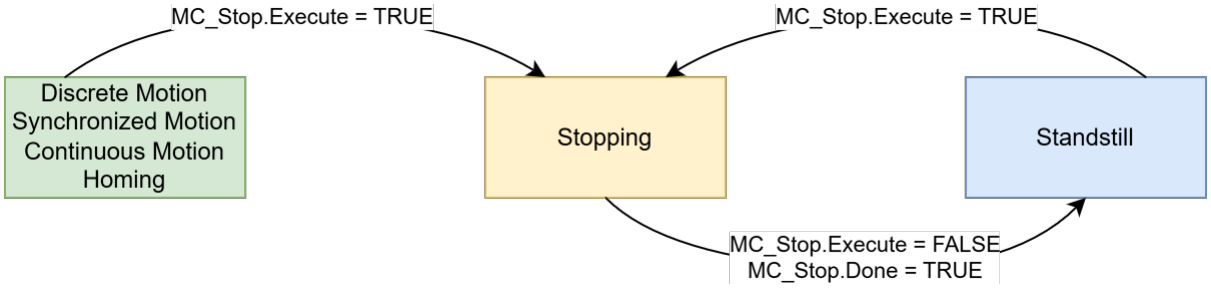
For the braking ramp, the maximum values of deceleration and jerk from the previous command and those specified in the function block are used.

Alternatively, an axis can be stopped with MC_Halt [▶ 269] without locking.

If MC_Stop is applied to a coupled slave axis, the axis is automatically decoupled, and the stop command is then executed.

Effects on the axis state

The MC_Stop function block can be called in any motion state of the axis. This sets the axis to the `Stopping` state. In addition to bringing the axis to a stop using a defined braking ramp, the `Stopping` state locks the axis against other motion commands. Therefore, it is also possible to call MC_Stop while in the `Standstill` state to lock the axis. Only when the axis is no longer moving and the input `MC_Stop.Execute = FALSE` is the axis set to the `Standstill` state, thereby releasing the axis lock.



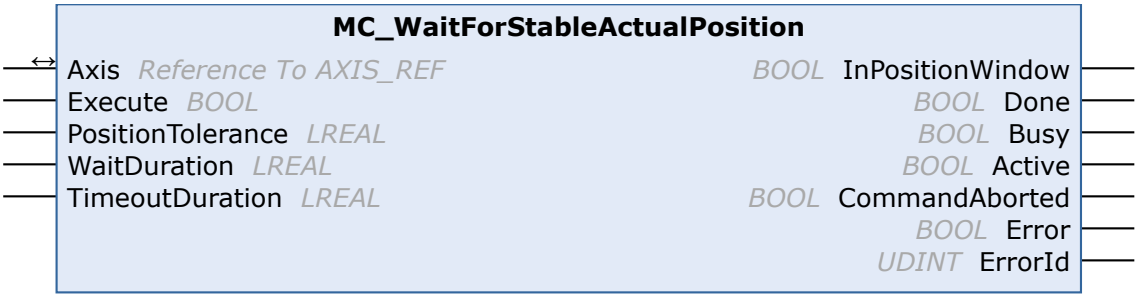
Note on the migration from NC2 to MC3

In MC3, the parameterization of the dynamics is not possible with a value of `0` (implicit use of the last commanded dynamics).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.4.6 MC_WaitForStableActualPosition



This function block waits for a stable actual position. The command is always buffered and may only be triggered when the previous command ends with zero dynamics.

Syntax

Definition:

```
FUNCTION_BLOCK MC_WaitForStableActualPosition
VAR_IN_OUT
    Axis               : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute            : BOOL;
    PositionTolerance  : LREAL;
    WaitDuration       : LREAL;
    TimeoutDuration    : LREAL;
END_VAR
VAR_OUTPUT
    InPositionWindow   : BOOL;
    Done               : BOOL;
    Busy               : BOOL;
    Active             : BOOL;
    CommandAborted     : BOOL;
    Error              : BOOL;
    ErrorId            : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
PositionTolerance	LREAL	#Invalid	Position tolerance around the set position ([PosUnit], positive). The actual position must remain within this window for the command to succeed.
WaitDuration	LREAL	#Invalid	Duration (in seconds) that the actual position needs to consecutively stay inside the position window. Must have a value >= 0.
TimeoutDuration	LREAL	#Ignore	Duration (in seconds) after which an error is triggered if the actual position is not inside the position window. It can be left as 'Ignore' if no timeout is desired, otherwise must have a value > WaitDuration.

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

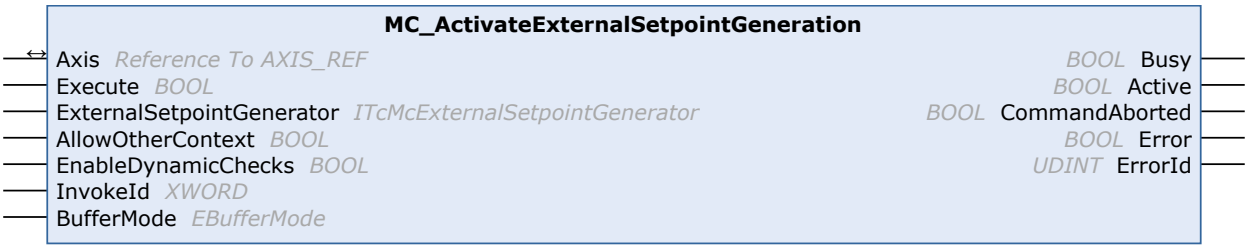
Name	Type	Description
InPositionWindow	BOOL	Actual position is currently inside the position window.
Done	BOOL	Actual position has been consecutively inside the position window for at least WaitDuration.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis. While Active is TRUE, the set position remains on the final position of the previous command and the set dynamics remain zero.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.5 External Setpoint Generator

5.3.5.5.1 MC_ActivateExternalSetpointGeneration



This function block activates external setpoint generation. Setpoints are generated by a function block, which is called through the ITcMcExternalSetpointGenerator interface from the MC context. Additionally custom startup data can be passed through the generator.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ActivateExternalSetpointGeneration
VAR_IN_OUT
    Axis                     : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute                  : BOOL;
    ExternalSetpointGenerator : ITcMcExternalSetpointGenerator;
    AllowOtherContext        : BOOL;
    EnableDynamicChecks      : BOOL;
    InvokeId                 : XWORD;
    BufferMode               : EBufferMode;
END_VAR
```

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

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
ExternalSetpointGenerator	ITcMcExternalSetpointGenerator		Function block that implements (cyclic) methods for external setpoint generation
AllowOtherContext	BOOL	false	Allow the calling task to differ from the MC task (MET task). When TRUE, ensure compatible update rates and safe data access in the ITcMcExternalSetpointGenerator methods.
EnableDynamicChecks	BOOL	true	Trigger error on inconsistent setpoint values (position, velocity, acceleration) if EnableDynamicChecks = TRUE.
InvokeId	XWORD	0	Invocation identifier, passed through to the external setpoint generator. Can be used to distinguish multiple activations of the same generator, or to pass custom data (e.g. a pointer).
BufferMode	EBufferMode	Aborting	Chronological sequence of the command

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

- ITcMcExternalSetpointGenerator [▶ 211]
- Appendix > External setpoint generation [▶ 302]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.5.2 MC_ExternalSetpointGenerator

This function block provides the API to implement an external setpoint generator from the PLC environment. The included callback methods are called from MC task context.
At least the OnCyclicSetpointGeneration method must be implemented.

Requirements for deriving function block:
- The function block must implement ITcMcExternalSetpointGenerator interface.
- The function block and all overridden ITcMcExternalSetpointGenerator methods must have {attribute 'c++_compatible'}.
- The function block must override FB_init.

Notes:
For online change compatibility the function block must have a FB_reinit method, in which MC_ExternalSetpointGenerator.FB_reinit() is called.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
FB_reinit [▶ 280]	
OnPreparation [▶ 281]	Called once when the external setpoint generator is being prepared for activation.
OnStartPointInitialization [▶ 281]	Called to initialize the starting point before cyclic setpoint generation begins.
OnCyclicSetpointGeneration [▶ 282]	Called every MC cycle to compute the next setpoint values. Must be overridden.
OnDeactivation [▶ 282]	Called when the external setpoint generator is deactivated/aborted.

Further information

- ITcMcExternalSetpointGenerator [▶ 211]
- Appendix > External setpoint generation [▶ 302]

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.5.2.1 FB_reinit

Syntax

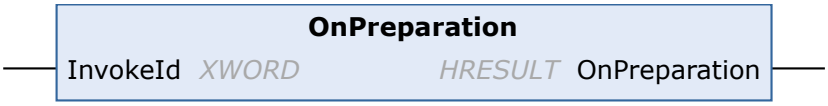
Definition:

```
METHOD FB_reinit : BOOL
```

Return value

BOOL

5.3.5.5.2.2 OnPreparation



Called once when the external setpoint generator is being prepared for activation.

Syntax

Definition:

```
METHOD OnPreparation : HRESULT
VAR_INPUT
    InvokeId : XWORD;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	

Return value

HRESULT

5.3.5.5.2.3 OnStartPointInitialization

Called to initialize the starting point before cyclic setpoint generation begins.

Syntax

Definition:

```
METHOD OnStartPointInitialization : HRESULT
VAR_INPUT
    InvokeId     : XWORD;
    StartDcTime  : LINT;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	
StartDcTime	LINT	
Position	LREAL	
Velocity	LREAL	
Acceleration	LREAL	

Return value

HRESULT

5.3.5.5.2.4 OnCyclicSetpointGeneration

Called every MC cycle to compute the next setpoint values. Must be overridden.

Syntax

Definition:

```
METHOD OnCyclicSetpointGeneration : HRESULT
VAR_INPUT
    InvokeId         : XWORD;
    TimeInGeneration : LREAL;
    Position         : Reference To LREAL;
    Velocity         : Reference To LREAL;
    Acceleration     : Reference To LREAL;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	
TimeInGeneration	LREAL	
Position	Reference To LREAL	
Velocity	Reference To LREAL	
Acceleration	Reference To LREAL	

Return value

HRESULT

5.3.5.5.2.5 OnDeactivation

Called when the external setpoint generator is deactivated/aborted.

Syntax

Definition:

```
METHOD OnDeactivation : HRESULT
VAR_INPUT
    InvokeId : XWORD;
END_VAR
```

Inputs

Name	Type	Description
InvokeId	XWORD	

Return value

HRESULT

5.3.5.6 Homing

5.3.5.6.1 MC_Home

This function block commands the axis to perform the homing sequence. The homing sequence is added via the homing TCOM object. The calculated offset is stored in 'Active Homing Offset'. Offsets from MC_SetPosition are discarded.

Syntax

Definition:

```
FUNCTION_BLOCK MC_Home
VAR_IN_OUT
    Axis            : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Trigger the command with rising edge.
Position	LREAL	#Default	Absolute position at the reference signal ([PosUnit])

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	'Search home' sequence succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Effects on the axis state

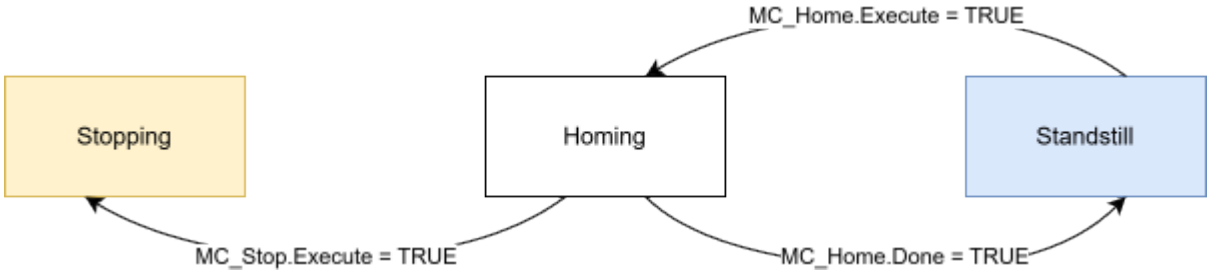


Fig. 2:

Notes on the migration from NC2 to MC3

In MC3, the MC_Home function block has fewer inputs than in NC2. The bCalibrationCam input, familiar from NC2, is now included as the input variable `Signal` in the homing object and can be linked directly there.

- Homing module [▶ 106] documentation

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.6.2 MC_SetHomingState

This function block allows setting and resetting the IsPositionValid flag in the CDT_MCTOPLC_AXIS_STD. Optionally, it can also set the Position.

Syntax

Definition:

```
FUNCTION_BLOCK MC_SetHomingState
VAR_INPUT
    Axis            : Type_PtpSetHomingState;
    Execute         : BOOL;
    Position        : LREAL;
    Mode            : EHomingMode;
```

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

Inputs

Name	Type	Default	Description
Axis	Type_PtpSetHomingState [▶ 246]		Reference to the axis
Execute	BOOL		Trigger the command with rising edge.
Position	LREAL	#Invalid	New position value ([PosUnit])
Mode	EHomingMode [▶ 237]	ForceIsHomed	Homing mode selection

Outputs

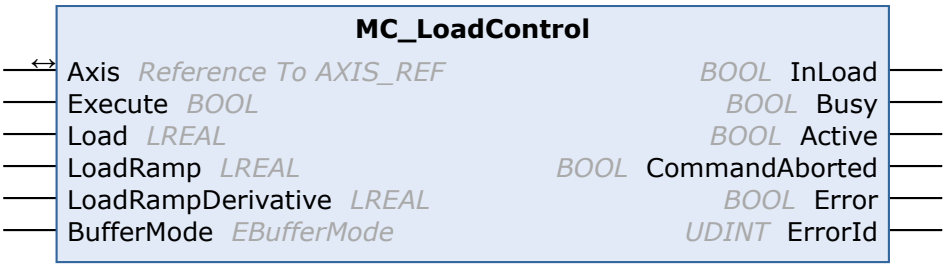
Name	Type	Description
Done	BOOL	Homing succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.7 Load Control

5.3.5.7.1 MC_LoadControl



This function block continuously exerts a torque, force or pressure of the specified value.
Positive torque, force and differential pressure is in the positive direction of velocity, pressure is physically unsigned. LoadUnit must match the unit of the value measured by the corresponding sensor.

Syntax

Definition:

```
FUNCTION_BLOCK MC_LoadControl
VAR_IN_OUT
    Axis              : Reference To AXIS_REF;
```

```
END_VAR
VAR_INPUT
    Execute            : BOOL;
    Load               : LREAL;
    LoadRamp           : LREAL;
    LoadRampDerivative : LREAL;
    BufferMode         : EBufferMode;
END_VAR
VAR_OUTPUT
    InLoad             : BOOL;
    Busy               : BOOL;
    Active             : BOOL;
    CommandAborted     : BOOL;
    Error              : BOOL;
    ErrorId            : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Start the motion at rising edge.
Load	LREAL	#Invalid	Commanded load value ([LoadUnit]). It is approached using a defined ramp (LoadRamp).
LoadRamp	LREAL	#Invalid	Maximum load derivative ([LoadUnit/s], positive)
LoadRampDerivative	LREAL	#Ignore	Maximum load ramp derivative ([LoadUnit/s^2], positive). It can be left as 'Ignore' if no ramp derivative limit is desired.
BufferMode	EBufferMode	Aborting	Chronological sequence of the command

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF [▶ 287]	Reference to the axis

Outputs

Name	Type	Description
InLoad	BOOL	Commanded load value reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.8 Motion Objects

5.3.5.8.1 AXIS_REF

An MC3 axis used with PLCopen motion control function blocks.

Syntax

Definition:

```
FUNCTION_BLOCK AXIS_REF
VAR_INPUT
    PlcToMc : AXIS_REF_TOMC;
    McToPlc : AXIS_REF_TOPLC;
END_VAR
```

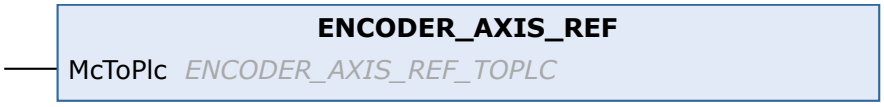
Inputs

Name	Type	Description
PlcToMc	AXIS_REF_TOMC [▶ 239]	Data area for cyclic PLC-to-MC data exchange (mapped to MC axis TcCOM object input process image).
McToPlc	AXIS_REF_TOPLC [▶ 240]	Data area for cyclic MC-to-PLC data exchange (mapped to MC axis TcCOM object output process image).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.8.2 ENCODER_AXIS_REF



An MC3 encoder axis used with PLCopen motion control function blocks.

Syntax

Definition:

```
FUNCTION_BLOCK ENCODER_AXIS_REF
VAR_INPUT
    McToPlc : ENCODER_AXIS_REF_TOPLC;
END_VAR
```

Inputs

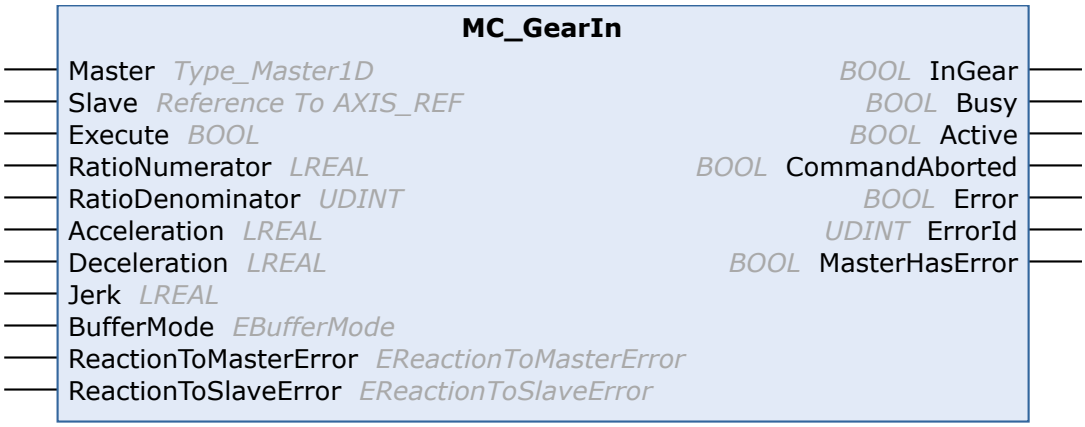
Name	Type	Description
McToPlc	ENCODER_AXIS_REF_TOPLC [▶ 241]	Data area for cyclic MC-to-PLC data exchange (mapped to MC encoder axis TcCOM object output process image).

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.9 Synchronized Motion

5.3.5.9.1 MC_GearIn



This function block commands a ratio between the velocity of the slave and master axis.

Syntax

Definition:

```
FUNCTION_BLOCK MC_GearIn
VAR_INPUT
    Master                : Type_Master1D;
    Slave                 : Reference To AXIS_REF;
    Execute               : BOOL;
    RatioNumerator        : LREAL;
    RatioDenominator      : UDINT;
    Acceleration          : LREAL;
    Deceleration          : LREAL;
    Jerk                  : LREAL;
    BufferMode            : EBufferMode;
    ReactionToMasterError : EReactionToMasterError;
    ReactionToSlaveError  : EReactionToSlaveError;
END_VAR
VAR_OUTPUT
    InGear                : BOOL;
    Busy                  : BOOL;
    Active                : BOOL;
    CommandAborted        : BOOL;
    Error                 : BOOL;
    ErrorId               : UDINT;
    MasterHasError        : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Reference To AXIS_REF [▶ 287]		Reference to the slave axis
Execute	BOOL		Trigger the command with rising edge.
RatioNumerator	LREAL	1.0	Gear ratio numerator
RatioDenominator	UDINT	1	Gear ratio denominator
Acceleration	LREAL	#Default	Maximum slave acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum slave deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum slave jerk ([PosUnit]/s^3, positive)

Name	Type	Default	Description
BufferMode	EBufferMode	Aborting	Chronological sequence of the command
ReactionToMasterError	EReactionToMasterError	TriggerSlaveError	Reaction if an error occurs in the master while this command is active.
ReactionToSlaveError	EReactionToSlaveError	TriggerMasterError	Reaction if an error occurs in the slave while this command is active.

Outputs

Name	Type	Description
InGear	BOOL	Synchronization established
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
MasterHasError	BOOL	Master is in error state.

Further information

The Acceleration and Deceleration inputs must be set to the same value. Furthermore, coupling is only possible when the slave axis is at a standstill.

Additional information on ReactionToSlaveError and ReactionToMasterError

As a general rule, if there is an error on the slave axis:

- Slave axis: The coupling is released, and the response depends on the error type.
- Master Axis: The master axis reports an error and either follows the MC's stop ramp or continues its current motion without being affected.

If there is an error on the master axis:

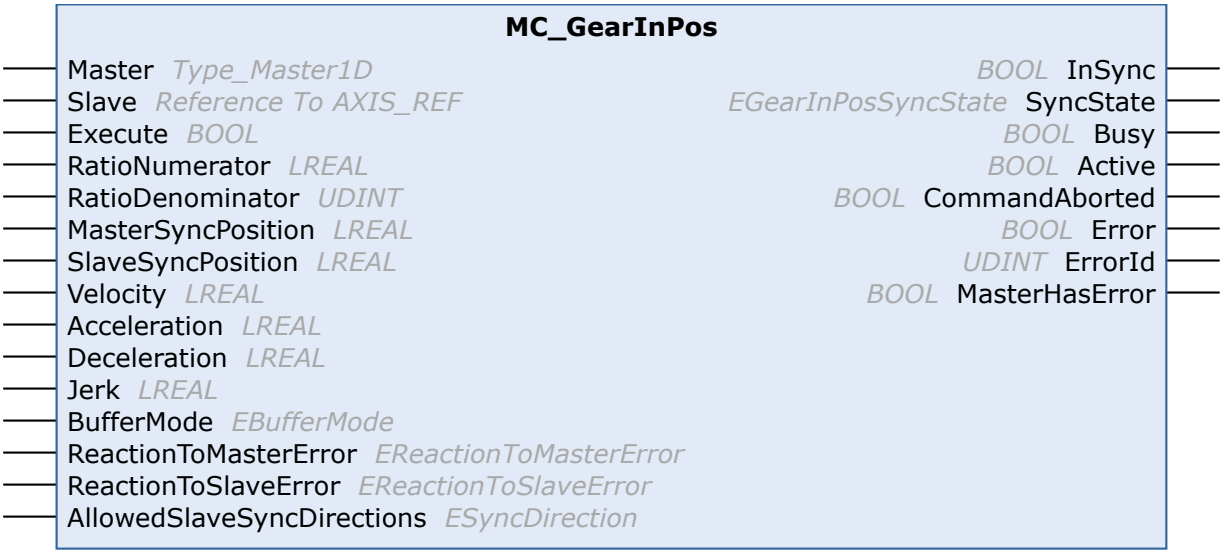
- Master axis: The response depends on the type of error.
- Slave axis: The slave axis reports an error and follows either the MC's stop ramp or the master axis's setpoints.

Detailed information is provided in the Error behavior [▶ 306] chapter.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.9.2 MC_GearInPos



This function block commands a gear ratio between the position of the slave and master axes from the synchronization point onwards.

Syntax

Definition:

```
FUNCTION_BLOCK MC_GearInPos
VAR_INPUT
    Master                    : Type_Master1D;
    Slave                     : Reference To AXIS_REF;
    Execute                   : BOOL;
    RatioNumerator            : LREAL;
    RatioDenominator          : UDINT;
    MasterSyncPosition        : LREAL;
    SlaveSyncPosition         : LREAL;
    Velocity                  : LREAL;
    Acceleration              : LREAL;
    Deceleration              : LREAL;
    Jerk                      : LREAL;
    BufferMode                : EBufferMode;
    ReactionToMasterError     : EReactionToMasterError;
    ReactionToSlaveError      : EReactionToSlaveError;
    AllowedSlaveSyncDirections : ESyncDirection;
END_VAR
VAR_OUTPUT
    InSync                    : BOOL;
    SyncState                 : EGearInPosSyncState;
    Busy                      : BOOL;
    Active                    : BOOL;
    CommandAborted            : BOOL;
    Error                     : BOOL;
    ErrorId                   : UDINT;
    MasterHasError            : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Reference To AXIS_REF [▶ 287]		Reference to the slave axis
Execute	BOOL		Trigger the command with rising edge.
RatioNumerator	LREAL	1.0	Gear ratio numerator
RatioDenominator	UDINT	1	Gear ratio denominator

Name	Type	Default	Description
MasterSyncPosition	LREAL	#Invalid	Position of master axis where master and slave axis are synchronized.
SlaveSyncPosition	LREAL	#Invalid	Position of slave axis where master and slave axis are synchronized.
Velocity	LREAL	#Maximum	Maximum velocity during synchronization ([PosUnit]/s, positive)
Acceleration	LREAL	#Default	Maximum acceleration during synchronization ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Default	Maximum deceleration during synchronization ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk during synchronization ([PosUnit]/s^3, positive)
BufferMode	EBufferMode	Aborting	Chronological sequence of the command
ReactionToMasterError	EReactionToMasterError	TriggerSlaveError	Reaction if an error occurs in the master while this command is active.
ReactionToSlaveError	EReactionToSlaveError	TriggerMasterError	Reaction if an error occurs in the slave while this command is active.
AllowedSlaveSyncDirections	ESyncDirection	TowardsSlaveSyncPosition	Allowed directions for establishing initial synchronization

Outputs

Name	Type	Description
InSync	BOOL	True if SyncState == InSync (slave coupled to the master AND master has passed its master sync position).
SyncState	EGearInPosSyncState [▶ 236]	Synchronization state. See EGearInPosSyncState for details.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
MasterHasError	BOOL	Master is in error state.

Additional information on ReactionToSlaveError and ReactionToMasterError

As a general rule, if there is an error on the slave axis:

- Slave axis: The coupling is released, and the response depends on the error type.
- Master Axis: The master axis reports an error and either follows the MC's stop ramp or continues its current motion without being affected.

If there is an error on the master axis:

- Master axis: The response depends on the type of error.
- Slave axis: The slave axis reports an error and follows either the MC's stop ramp or the master axis's setpoints.

Detailed information is provided in the Error behavior [▶ 306] chapter.

Version information

- TwinCAT Standard >= v3.1.4026.23.1

- TF5500 MC3 Base >= v4.0.6

5.3.5.9.3 MC_HaltPhasing

This function block commands a phase shift stop in the master position of a synchronized slave axis. It is only valid for camming synchronization. The master axis is not aware of the phase shift.

Syntax

Definition:

```
FUNCTION_BLOCK MC_HaltPhasing
VAR_INPUT
    Master          : Type_Master1D;
    Slave           : Type_Phasing1D;
    Execute         : BOOL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
END_VAR
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorId         : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Type_Phasing1D		Reference to the slave axis
Execute	BOOL		Trigger the command with rising edge.
Deceleration	LREAL	#Invalid	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Invalid	Maximum jerk ([PosUnit]/s^3, positive)

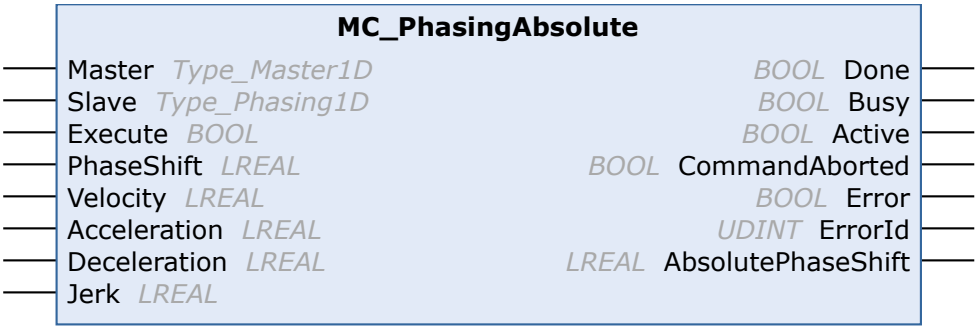
Outputs

Name	Type	Description
Done	BOOL	Zero dynamics has been reached, axis state is transferred to Standstill.
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.9.4 MC_PhasingAbsolute



This function block commands an absolute phase shift in the master position of a synchronized slave axis. It is only valid for camming synchronization. The master axis is not aware of the phase shift.

Syntax

Definition:

```
FUNCTION_BLOCK MC_PhasingAbsolute
VAR_INPUT
    Master              : Type_Master1D;
    Slave               : Type_Phasing1D;
    Execute             : BOOL;
    PhaseShift          : LREAL;
    Velocity            : LREAL;
    Acceleration        : LREAL;
    Deceleration        : LREAL;
    Jerk                : LREAL;
END_VAR
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Active              : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
    AbsolutePhaseShift  : LREAL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Type_Phasing1D		Reference to the slave axis
Execute	BOOL		Trigger the command with rising edge.
PhaseShift	LREAL	#Invalid	Absolute phase difference in master position of the slave axis ([PosUnit])
Velocity	LREAL	#Invalid	Maximum velocity ([PosUnit]/s, positive)
Acceleration	LREAL	#Invalid	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Invalid	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Invalid	Maximum jerk ([PosUnit]/s^3, positive)

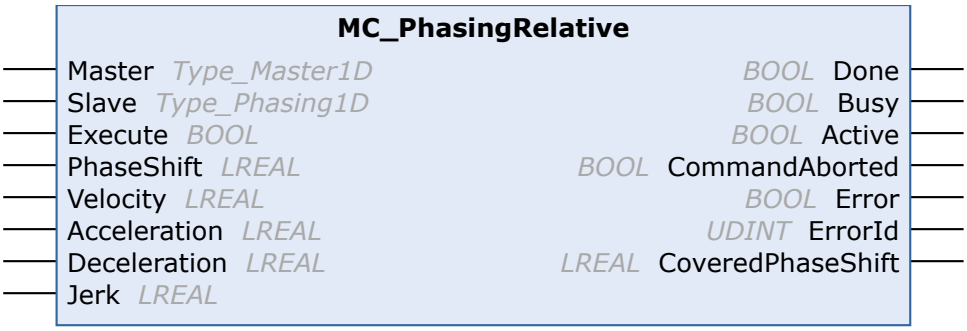
Outputs

Name	Type	Description
Done	BOOL	Commanded phasing reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
AbsolutePhaseShift	LREAL	Current absolute phase shift between master and slave.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.9.5 MC_PhasingRelative



This function block commands a relative phase shift in the master position of a synchronized slave axis. It is only valid for camming synchronization. The master axis is not aware of the phase shift.

Syntax

Definition:

```
FUNCTION_BLOCK MC_PhasingRelative
VAR_INPUT
    Master            : Type_Master1D;
    Slave             : Type_Phasing1D;
    Execute           : BOOL;
    PhaseShift        : LREAL;
    Velocity          : LREAL;
    Acceleration      : LREAL;
    Deceleration      : LREAL;
    Jerk              : LREAL;
END_VAR
VAR_OUTPUT
    Done              : BOOL;
    Busy              : BOOL;
    Active            : BOOL;
    CommandAborted    : BOOL;
    Error             : BOOL;
    ErrorId           : UDINT;
    CoveredPhaseShift : LREAL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Type_Phasing1D		Reference to the slave axis
Execute	BOOL		Trigger the command with rising edge.
PhaseShift	LREAL	#Invalid	Additional phase difference in master position of the slave axis ([PosUnit])
Velocity	LREAL	#Invalid	Maximum velocity ([PosUnit]/s, positive)
Acceleration	LREAL	#Invalid	Maximum acceleration ([PosUnit]/s^2, positive)
Deceleration	LREAL	#Invalid	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Invalid	Maximum jerk ([PosUnit]/s^3, positive)

Outputs

Name	Type	Description
Done	BOOL	Commanded phasing reached
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
CoveredPhaseShift	LREAL	Accumulated relative phase shift applied so far.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.10 TouchProbe

5.3.5.10.1 BinaryTouchProbeRef

TouchProbeRef for Binary Touch Probes.

Do not call the main FB directly. Only use the available methods.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.10.2 DriveTouchProbeRef

TouchProbeRef for Drive Touch Probe.

Do not call the main FB directly. Only use the available methods.

Version information

- TwinCAT Standard >= v3.1.4026.23.1

- TF5500 MC3 Base >= v4.0.6

5.3.5.10.3 EncoderTouchProbeRef

TouchProbeRef for Encoder Touch Probe.

Do not call the main FB directly. Only use the available methods.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.10.4 FixedStopTouchProbeRef

TouchProbeRef for Fixed Stop Detection.

Do not call the main FB directly. Only use the available methods.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.10.5 MC_AbortTrigger

This Function Block is used to abort function blocks which are used to record trigger events (e.g. MC_TouchProbe).

Syntax

Definition:

```
FUNCTION_BLOCK MC_AbortTrigger
VAR_INPUT
    Axis         : Type_PtpTouchProbe;
END_VAR
VAR_IN_OUT
    TriggerInput : Reference To TRIGGER_REF;
END_VAR
VAR_INPUT
    Execute      : BOOL;
END_VAR
VAR_OUTPUT
    Done         : BOOL;
    Busy         : BOOL;
    Error        : BOOL;
    ErrorId      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
Axis	Type_PtpTouchProbe [▶ 247]	Reference to the axis
Execute	BOOL	Aborts the monitoring of trigger event at rising edge.

In/Outputs

Name	Type	Description
TriggerInput	Reference To TRIGGER_REF [▶ 244]	Reference to the trigger signal source like DriveTouchProbeRef or BinaryTouchProbeRef

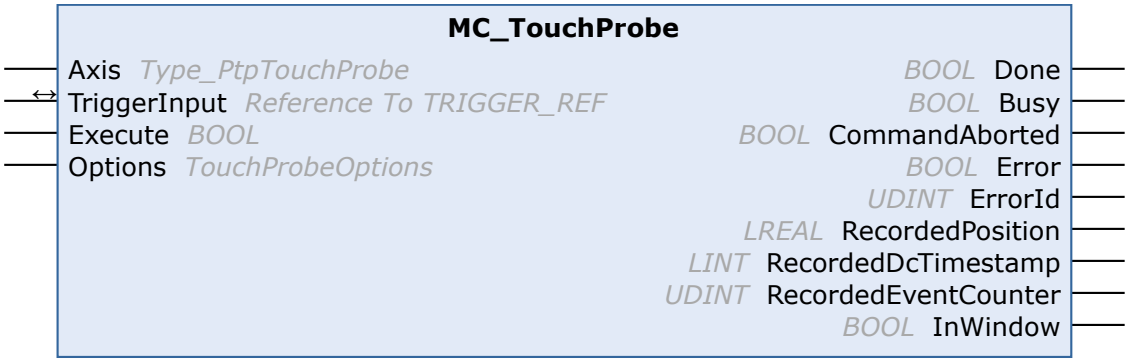
Outputs

Name	Type	Description
Done	BOOL	Abort Trigger succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.10.6 MC_TouchProbe



This function block controls the recording of an axis position at a trigger event.
One of WindowFirstPosition or WindowLastPosition can be left as 'Ignore' if no corresponding limit is desired.

Syntax

Definition:

```
FUNCTION_BLOCK MC_TouchProbe
VAR_INPUT
    Axis                : Type_PtpTouchProbe;
END_VAR
VAR_IN_OUT
    TriggerInput        : Reference To TRIGGER_REF;
END_VAR
VAR_INPUT
    Execute             : BOOL;
    Options             : TouchProbeOptions;
END_VAR
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
    RecordedPosition    : LREAL;
    RecordedDcTimestamp : LINT;
    RecordedEventCounter : UDINT;
    InWindow            : BOOL;
END_VAR
```

Inputs

Name	Type	Description
Axis	Type_PtpTouchProbe [▶ 247]	Reference to the axis
Execute	BOOL	Trigger the command with rising edge.
Options	TouchProbeOptions [▶ 243]	Options for the touch probe

In/Outputs

Name	Type	Description
TriggerInput	Reference To TRIGGER_REF [▶ 244]	Reference to the trigger signal source like DriveTouchProbeRef or BinaryTouchProbeRef

Outputs

Name	Type	Description
Done	BOOL	Touch probe process succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
CommandAborted	BOOL	Command is aborted by another command (MC_AbortTrigger).
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
RecordedPosition	LREAL	Position where trigger event occurred ([PosUnit]).
RecordedDcTimestamp	LINT	DcTime at trigger event
RecordedEventCounter	UDINT	Number of recorded trigger events
InWindow	BOOL	TRUE if the recorded position is within the configured window.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.5.10.7 TimestampTouchProbeRef

TouchProbeRef for Timestamp Touch Probes.

Do not call the main FB directly. Only use the available methods.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5500 MC3 Base >= v4.0.6

5.3.6 Appendix

5.3.6.1 Upgrade from NC2 to MC3

Function blocks

NC2 function block	MC3 function block	Note
MC_Power.Enable_Positive MC_Power.Enable_Negative	MC_Power [▶ 248].EnablePositive MC_Power [▶ 248].EnableNegative	Underscore removed
MC_ExtSetPointGenEnable	MC_ActivateExternalSetpointGeneration [▶ 278]	Further details on External setpoint generation [▶ 302]
MC_ExtSetPointGenDisable	Cancel MC_ActivateExternalSetpointGeneration [▶ 278] with an abort command	Example: MC_Halt
MC_ExtSetPointGenFeed	See ITcNcExternalSetpointGeneration [▶ 278]	
MC_ExtSetPointGenFeedWithTorque	-	
MC_GearInDyn	In the Future via MC_GearIn [▶ 288]	
MC_GearOut		Issuing a motion or hold/stop command to a slave axis disconnects it from the master.
MC_CamOut		Issuing a motion or hold/stop command to a slave axis disconnects it from the master.
MC_Power.Override (Value range in %)	MC_SetOverride [▶ 256].VelocityFactor (Value range normalized to 1)	On the MC3, the override is set to 1 (100%) by default. In Motion UI, values are entered as percentages.
MC_SetPosition.Mode	MC_SetPosition [▶ 258].Relative	Renaming the input for better readability in the code.
MC_Reset	MC_Reset [▶ 255]	MC3: Takes effect when the axis is in the Error state. An additional asynchronous reset command for Sercos devices is also triggered and evaluated. `MC_Reset.Done = TRUE`, if the hardware has been reset (including an additional reset for Sercos) and the axis status has changed to “Disabled” or “Standstill.” The position lag is not reset and persists despite the reset command.
MC_Home.bCalibrationCam	See the MC_Home [▶ 283] and Homing Modules [▶ 106] chapters	Centralized configuration, parameterization, and linking of I/O components in Homing modules [▶ 106]
MC_Home.HomingMode := MC_DefaultHoming	MC_Home [▶ 283]	MC_Home performs the standard homing procedure configured in the Homing object [▶ 106].

NC2 function block	MC3 function block	Note
MC_Home.HomingMode := MC_Direct	MC_SetHomingState [▶ 284].Mode := EHomingMode.DirectSetPosition	Additional function block in addition to MC_Home [▶ 283].
MC_Home.HomingMode := MC_ForceCalibration	MC_SetHomingState [▶ 284].Mode := EHomingMode [▶ 237].ForceIsPositionValid	Additional function block in addition to MC_Home [▶ 283].
MC_Home.HomingMode := MC_ResetCalibration	MC_SetHomingState [▶ 284].MOD := EHomingMode [▶ 237].ResetIsPositionValid	Additional function block in addition to MC_Home [▶ 283].
MC_MoveVelocity.InVelocity MC_MoveVelocity.Busy MC_MoveVelocity.Active	MC_MoveVelocity [▶ 267].InVelocity MC_MoveVelocity [▶ 267].Busy MC_MoveVelocity [▶ 267].Active	In the initial situation, the command was sent using `Execute := TRUE`, and `Execute := FALSE` is set again. **NC2**: `Busy = TRUE, Active = TRUE` and `InVelocity = TRUE` are reset by a falling edge at the Execute input. However, the outputs (especially `InVelocity`) signal TRUE for at least one cycle as soon as the target velocity is reached. **MC3**: `Busy = TRUE, Active = TRUE` and `InVelocity = TRUE` remain set until the motion command is replaced by another command, even if `Execute` is already set to `FALSE` (PLCopen-compliant).
MC_GearIn.InGear MC_GearIn.Busy MC_GearIn.Active (same as MC_GearInPos)	MC_GearIn [▶ 288].InGear MC_GearIn [▶ 288].Busy MC_GearIn [▶ 288].Active (same as MC_GearInPos [▶ 290])	In the initial situation, the command was sent using `Execute := TRUE`, and `Execute := FALSE` is set again. **NC2**: `Busy = TRUE, Active = TRUE` and `InGear = TRUE` are reset by a falling edge at the Execute input. However, the outputs (in particular `InGear`) signal TRUE for at least one cycle as soon as the master and slave axes move in sync with each other. **MC3**: `Busy = TRUE, Active = TRUE` and `InGear = TRUE` remain set until the motion command is replaced by another command, even if `Execute` is already set to `FALSE` (PLCopen-compliant).

Re-triggering an MC function block using MC_MoveAbsolute as an example

```
VAR
    axis_nc2 : TC2_MC2.AXIS_REF;
    axis_mc3 : Tc3_Mc3Ptp.AXIS_REF;

    move_absolute_mc3 : Tc3_Mc3Ptp.MC_MoveAbsolute;
    move_absolute_nc2_1 : Tc2_MC2.MC_MoveAbsolute;
    move_absolute_nc2_2 : Tc2_MC2.MC_MoveAbsolute;
END_VAR
```

```
CASE step OF
    0:
        // MC3 init movement
        move_absolute_mc3(Axis := axis_mc3, Position := 111, Execute := TRUE);

        // NC2 init movement
        move_absolute_nc2_1(Axis := axis_nc2, Position := 111, Velocity := 100, Execute := TRUE);

        step := step + 1;

    1:
        // MC3 retrigger same FB instance during one cycle
        move_absolute_mc3 (Axis := axis_mc3, Position := 111, Velocity := 100, Execute := FALSE);
        move_absolute_mc3 (Axis := axis_mc3, Position := 222, Velocity := 100, Execute := TRUE);

        // NC2 use additional FB instance
        move_absolute_nc2_1(Axis := axis_nc2, Position := 111, Velocity := 100,Execute := FALSE);
        move_absolute_nc2_2(Axis := axis_nc2, Position := 222, Velocity := 100,Execute := TRUE);
        step := step + 1;
END_CASE
```

Re-triggering an MC function block using MC_MoveAbsolute as an example

In NC2, if the `Execute` input is triggered again while the command is being executed, the function block does not provide any feedback, nor does it execute any further commands.

This restriction does not apply to MC3. If the `Execute` input is triggered again while the command is running, the new parameters are applied and the function block is re-executed according to the buffer mode. This can be done within a single PLC cycle. For example, the first call to `Execute := FALSE` (changing from TRUE to FALSE) is `fbMyMoveAbsolute(Axis := ax, Execute := FALSE, Position := 111)`, and immediately following that is the second call to the same block instance with a new position: `fbMyMoveAbsolute(Axis := ax, Execute := TRUE, Position := 222)`.

Status

NC2	MC3	Note
myAxisRef.Status.IoDataInvalid	myAxisRef.McToPlc.Std.AxisState	PLCopen state diagram expanded (states: Invalid, NotReady, Enabling, Disabling). Example: The axis remains in the NotReady state if the drive hardware is not ready (see the State Diagram [▶ 229] chapter)
myAxisRef.Status.Homed	AxisRef.McToPlc.Std.IsPositionValid	`IsPositionValid` also takes into account, for example, the state of the axis's driven object (CoE drive input: `Position actual value invalid`). If the axis starts up using, among other things, an absolute encoder, the axis has a unique reference position and reports `IsPositionValid := TRUE`.
myAxisRef.ReadStatus()	myAxisRef.McToPlc.Std	Status information can be retrieved directly from the CDT `Std`.

Error at the drive hardware level or missing communication

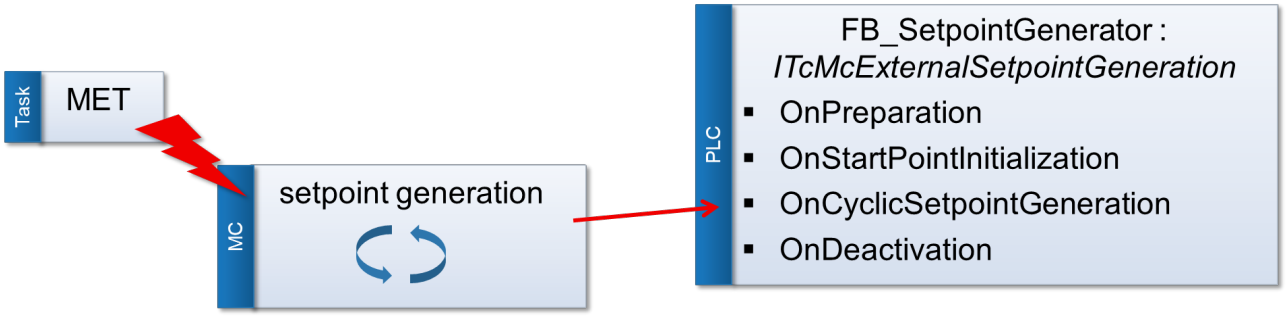
The starting point is the transition of the TwinCAT system from Config to Run. This occurs when the drive hardware is in an error state or communication via the EtherCAT fieldbus is disrupted.

- **NC2**: An axis error is displayed when `MC_Power.Enable = TRUE`. In addition, you can check for communication issues, for example, via `myAxisRef.Status.IoDataInvalid .`
- **MC3**: Once communication between the axis and the drive is established, the drive status and any errors are evaluated immediately and, if necessary, displayed as axis errors. The state diagram for the MC3 axis has been expanded compared to the NC2 and PLCopen (states: Invalid, NotReady, Enabling, Disabling). The MC3's axis status can be used to centrally monitor its state.
  - Example 1: If `MC_Power.Enable := FALSE` and communication with the drive is not possible, the axis status remains in the state `myAxisRef.McToPlc.Std.AxisState = EAxisState.NotReady.`
  - Example 2: If `MC_Power.Enable := FALSE` and the drive is in an error state, then the MC3 axis reports `myAxisRef.McToPlc.Std.AxisState = EAxisState.ErrorStop.`

5.3.6.2 External setpoint generation

External setpoint generation can be used to implement a custom setpoint profile that is not available by default in TwinCAT MC3. This profile can be mathematically very simple or as complex as necessary. In external setpoint generation, the setpoints are generated in function blocks or TcCOM modules that implement the ITcMcExternalSetpointGenerator [▶ 211] interface. This is called from the MC context.

The main input is ExternalSetpointGenerator of type ITcMcExternalSetpointGenerator [▶ 211]. This interface defines methods that must be implemented (e.g., by a PLC FB). These methods are called by the MC task context (MET-Task) via a Callback mechanism. This mechanism forms the basis for decoupling the MC task and the PLC task during external setpoint generation. This basic operating principle is represented in the figure below.



With the MC3, it is possible to execute PLC and MC3 axes with different cycle times in different contexts when using external setpoint generation. Be aware of the update rate of the data used in the Callback methods when AllowOtherContext is enabled.

MC3 can check the calculated setpoints for consistency (from the previous cycle to the current cycle, taking into account the maximum dynamic parameters of the axis). To do so, set `EnableDynamicChecks := TRUE`.

The interpretation of the input variable `InvokeId`, which is of type XWORD, is customer-specific. For example, any numeric value or the address (pointer) to a customer-specific data structure can be passed. This InvokeId is passed as an input to every method included in the interface.

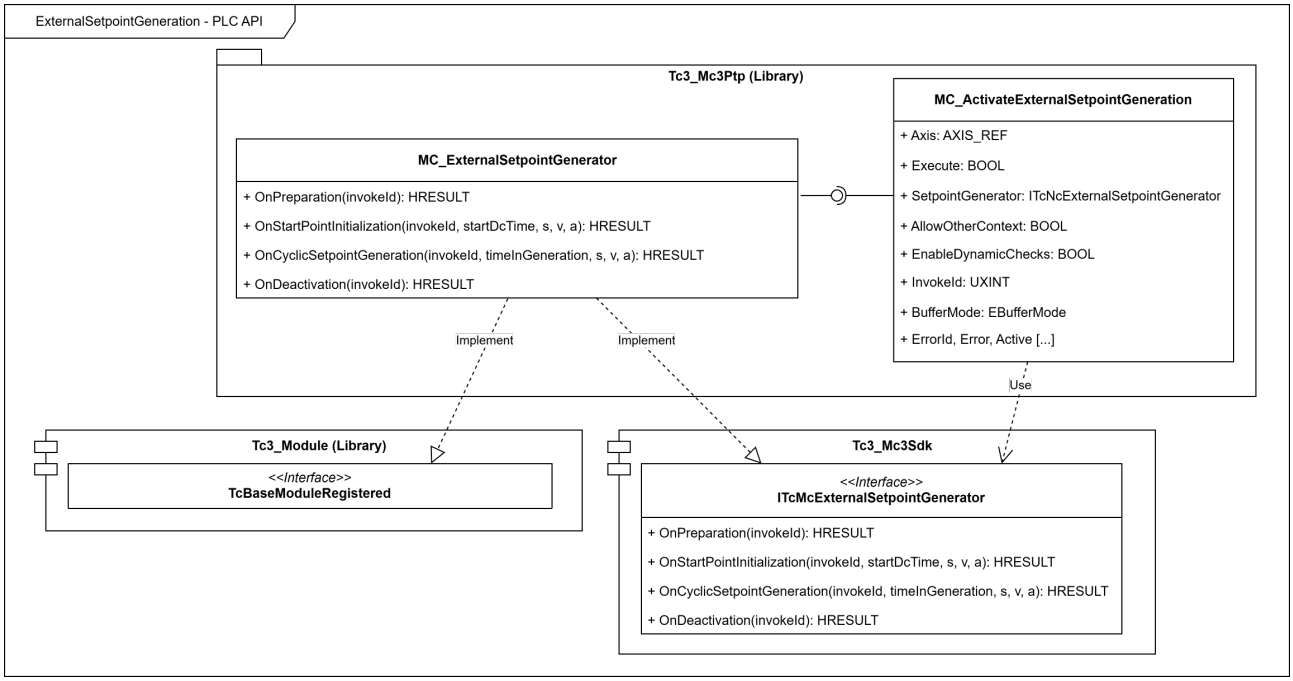
The Callback methods contained in the ITcMcExternalSetpointGenerator [▶ 211] interface are called by the MC task context (MET task) in the order listed:

1. `OnPreparation (invokeId : XWORD)`
   - Is called directly when MC_ActivateExternalSetpointGeneration [▶ 278].Execute = TRUE
   - For example, this method can be used to initialize setpoint generation.
2. `OnStartPointInitialization (invokeId : XWORD, startDcTime : LINT, position : LREAL, velocity : LREAL, acceleration : LREAL)`
   - Called shortly before the external setpoint generator becomes active and the next setpoint is generated by that generator for the first time.
   - For example, this method can be used to initialize setpoint generation.

- The last setpoint (position and dynamics) and the corresponding timestamp of the previous command (which has completed by this point) are passed to this method. The next setpoint/cycle is calculated by the external setpoint generator.

3. `ABSTRACT OnCyclicSetpointGeneration (invokeId : XWORD, timeInGeneration : LREAL, position : REFERENCE TO LREAL, velocity : REFERENCE TO LREAL, acceleration : REFERENCE TO LREAL)`
   - This method must be implemented.
   - The method is called cyclically when MC_ActivateExternalSetpointGeneration [▶ 278].`Active = TRUE`.
   - This method is used to calculate the setpoint for the specified timestamp.
   - The input variable `timeInGeneration` represents the elapsed time while external setpoint generation is active.
   - Position, velocity, and acceleration must be set to the values for the current setpoint generation cycle.
4. `OnDeactivation (invokeId : XWORD)`
   - This method is intended for the final steps when external setpoint generation is terminated by another motion command.
   - For example, it can be used to free up storage space that was allocated during external setpoint generation.

The FB MC_ExternalSetpointGenerator [▶ 280] provides the API for the implementation of external setpoint generation from the PLC environment. The component diagram below represents the relationships between the modules. The generator itself is a TcCOM object that is created at runtime and registered in the TwinCAT system (TcBaseModuleRegistered). This Generator FB already implements methods for handling online change processes. As a core feature, it implements the ITcMcExternalSetpointGenerator [▶ 211] (an extension of ITcUnknown).



5.3.6.3 Modulo positioning

The modulo positioning can be applied to closed linear axes as well as to rotary axes. TwinCAT does not distinguish between these types. A modulo axis has a consecutive absolute position in the range ±∞. The modulo position of the axis provides additional information about the absolute axis position. Modulo positioning represents the required target position in a different way.

Unlike absolute positioning, in which the user specifies the target unambiguously, in MC3 modulo positioning, the absolute target position is determined based on the following parameters:

- MC3 axis parameters [▶ 28]:
  - Position Modulus
  - Position Modulo Tolerance Window
- Tc3_Mc3Ptp.MC_MoveModulo [▶ 272] input parameters
  - MC_MoveModulo.Position
  - MC_MoveModulo.AdditionalTurns
  - MC_MoveModulo.Direction

Axis parameterization

Modulo support must be enabled on an MC3 axis using the “Is Modulo Axis” parameter. If this parameter is set, the other modulo parameters can be set.

	Name	Value	CS	Unit	Type
-	General				
	Simulation mode	☑ TRUE	☐		BOOL
	Position Unit	mm	☐	[PosUnit]	MC.EPositionUnit
	Is Modulo Axis	☑ TRUE	☐		BOOL
	Position Modulus	360.0	☐	[PosUnit]	LREAL
	Position Modulo Tolerance Window	0.0	☐	[PosUnit]	LREAL

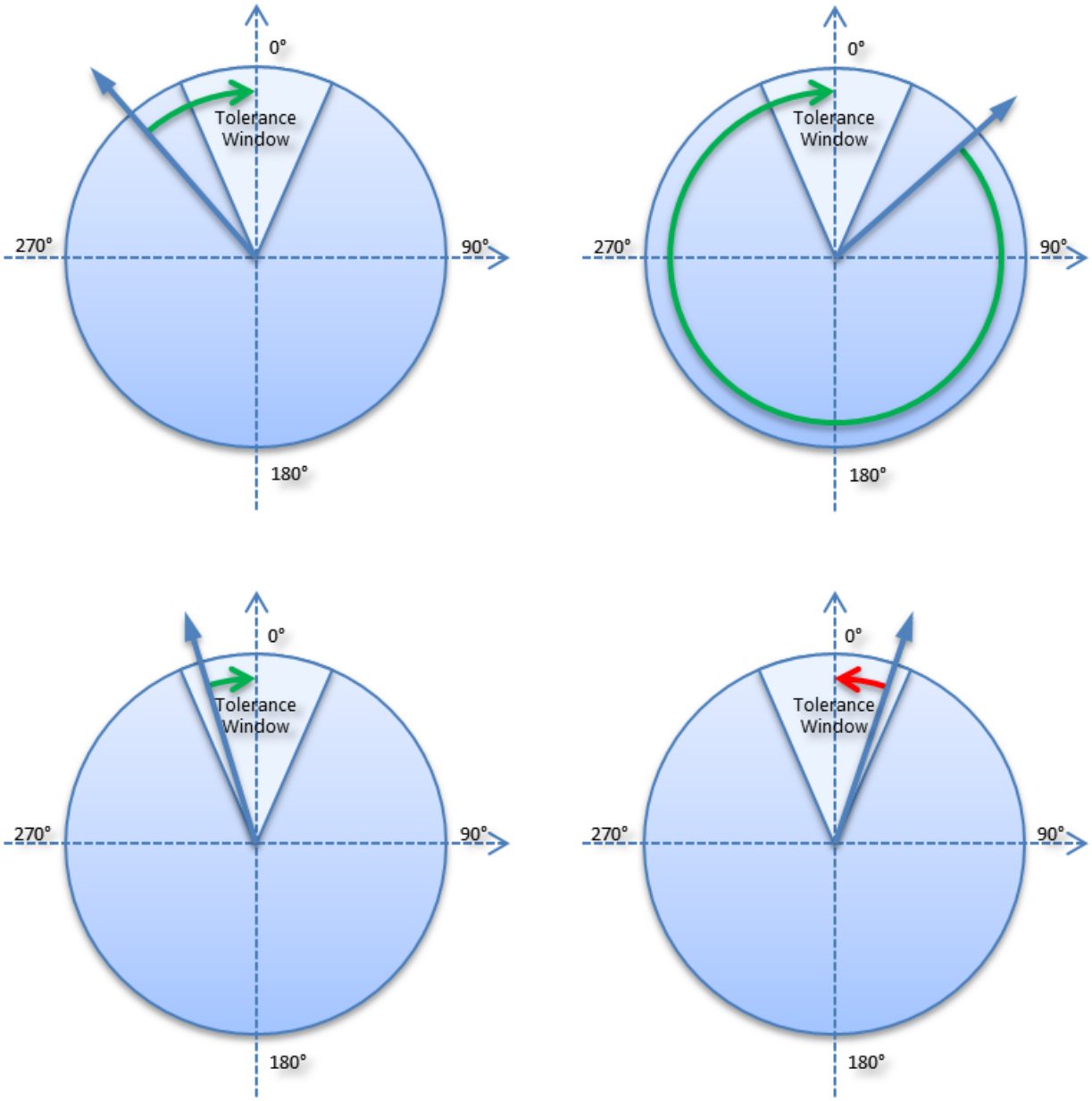
Position Modulus

The Position Modulus parameter defines the modulo factor used to calculate the modulo rotation and modulo position.

Position Modulo Tolerance Window

The Position Modulo Tolerance Window parameter defines a position window around the current modulo target position of the axis. The position window is equal to twice the specified value (target position ± position window).

The positioning of an axis is always referenced to its current actual position. Unintentional revolutions may be performed if the actual position and the target position are very close to each other. For example, if the actual position is slightly greater than the target position and `MC_MoveModulo.Direction= EModuloDirection.PositiveDirection` has been selected. This can occur, in particular, when the actual position is determined inaccurately. In order to avoid this, a tolerance window for modulo positioning can be set. If the distance between the start and target positions is less than or equal to the tolerance window, the target position is approached via the shortest path (as with `MC_ MoveModulo.Direction = EModuloDirection.ShortestDistance`), i.e. even if this is opposite to the specified direction.

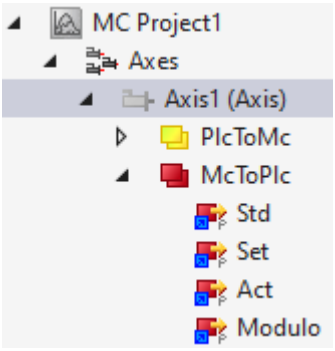


Cyclical interface

The target and actual modulo positions can be included in the cyclic interface.

Object | Settings | Init Parameter | Online Parameter | Data Area | Motion UI

	Area No	Enable	Name	Type	Size
-	0 (0)	☑	PlcToMc	InputDst	8
		☑	Std	POINTER TO MC.CDT_PLCTOMC_AXIS_STD	8.0 (Offs: 0.0)
-	1 (0)	☑	McToPlc	OutputSrc	32
		☑	Std	POINTER TO MC.CDT_MCTOPLC_AXIS_STD	8.0 (Offs: 0.0)
		☑	Set	POINTER TO MC.CDT_MCTOPLC_AXIS_SET	8.0 (Offs: 8.0)
		☑	Act	POINTER TO MC.CDT_MCTOPLC_AXIS_ACT	8.0 (Offs: 16.0)
		☑	Modulo	POINTER TO MC.CDT_MCTOPLC_AXIS_MODULO	8.0 (Offs: 24.0)
		☐	LoadControl	POINTER TO MC.CDT_MCTOPLC_AXIS_LOADCONTROL	8.0
		☐	FluidPower	POINTER TO MC.CDT_MCTOPLC_AXIS_FLUIDPOWER	8.0



PLC programming

An MC3 axis is moved to a modulo position using the MC_MoveModulo [▶ 272] function block from the Tc3_Mc3Ptp library.

MC_MoveModulo.Position

With modulo positioning, the target position must be less than or equal to the axis's configured position modulus.

MC_MoveModulo.AdditionalTurns

If a defined number of revolutions is to be performed before reaching the target position, this number must be specified via the `AdditionalTurns` input.

MC_MoveModulo.Direction

The direction of motion is specified via the `Direction` input of the `EModuloDirection` type from the Tc3_Mc3Ptp library. The possible values are:

- `ShortestDistance`
- `PositiveDirection`
- `NegativeDirection`

5.3.6.4 Error behavior

Error type

TwinCAT MC3 distinguishes between *execution errors* and *control errors*.

In the event of an *execution error*, the MC3 retains control of the axis. Communication with the hardware and the drive itself are functioning properly. Setpoint generation is still possible. In this case, the default response to an error is a controlled deceleration ramp using the maximum dynamics configured for the axis. This stop can be canceled using MC_Reset. After the reset, a new command can be initiated in the same cycle.

In the event of a *control error*, the MC3 loses control of the axis, typically due to a hardware or drive-related error. In this state, the response is defined by the parameters of the axis. Currently, a hard stop with zero dynamics is possible (HardStop).

Error propagation

The function blocks for couplings offer special settings for controlling error handling between the axes. The mechanism is identical for MC_GearIn, MC_GearInPos, and MC_CamIn.

ReactionToMasterError defines the behavior of the slave axis when the master axis enters an error state. When the `TriggerSlaveError` setting is enabled, any master error immediately triggers an error in the slave axis as well. In addition, the coupling is released. The slave axis then follows the standard MC3 response for that specific error type (see above). When the `KeepCouplingWithSetValues` setting is enabled, the coupling remains active even if the master axis has an error. The slave axis continues to generate setpoints based on the setpoints from the master axis. This mode allows for coordinated stopping while maintaining the synchronous relationship.

The reverse direction of propagation is configured using *ReactionToSlaveError*. When the `TriggerMasterError` setting is enabled, the slave axis forwards its error to its master axis. This is also how the master axis transitions into its error response. When the `NoReaction` setting is enabled, the master axis remains unaffected. The slave axis and the coupling module signal the error condition. In both cases, the coupling for this slave axis is disabled.

With the exception of the specific case described above (`KeepCouplingWithSetValues`), the master and slave axes are automatically decoupled if an error occurs on either side. After the underlying error has been acknowledged with MC_Reset, the application must re-execute the desired coupling command to resume coupled operation. Based on the diagnostic information available at the function block (`Error`, `ErrorID`, and `MasterHasError`), it is possible to determine whether the error occurred in the master or slave axis.

A propagated error is always an *execution error* on the receiver side.

Error propagation – Examples

As a general rule, if there is an error on the slave axis:

- Slave axis: The coupling is released, and the response depends on the error type.
- Master Axis: The master axis reports an error and either follows the MC's stop ramp or continues its current motion without being affected.

If there is an error on the master axis:

- Master axis: The response depends on the type of error.
- Slave axis: The slave axis reports an error and follows either the MC's stop ramp or the master axis's setpoints.

Scenario 1

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
TriggerSlaveError	TriggerMasterError	Control Error

Reaction to errors:

Master error	Slave error
• Zero-Dynamic Master • Controlled deceleration ramp of the slave axis • Coupling released	• Zero-Dynamic Slave • Controlled deceleration ramp for the master axis • Coupling released

Scenario 2

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
TriggerSlaveError	NoReaction	Control Error

Reaction to errors:

Master error	Slave error
• Zero-Dynamic Master • Controlled deceleration ramp of the slave axis • Coupling released	• Zero-Dynamic Slave • Master axis not affected • Coupling released

Scenario 3

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
KeepCouplingWithSetValues	TriggerMasterError	Control Error

Reaction to errors:

Master error	Slave error
• Zero-Dynamic Master • The slave axis attempts to follow the setpoints of the master axis • Maintain the coupling	• Zero-Dynamic Slave • Controlled deceleration ramp for the master axis • Coupling released

Scenario 4

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
KeepCouplingWithSetValues	NoReaction	Control Error

Reaction to errors:

Master error	Slave error
• Zero-Dynamic Master • The slave axis attempts to follow the setpoints of the master axis • Maintain the coupling	• Zero-Dynamic Slave • Master axis not affected • Coupling released

Scenario 5

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
TriggerSlaveError	TriggerMasterError	Execution Error

Reaction to errors:

Master error	Slave error
• Controlled deceleration ramp for the master axis • Controlled deceleration ramp of the slave axis • Coupling released	• Controlled deceleration ramp of the slave axis • Controlled deceleration ramp for the master axis • Coupling released

Scenario 6

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
TriggerSlaveError	NoReaction	Execution Error

Reaction to errors:

Master error	Slave error
• Controlled deceleration ramp for the master axis • Controlled deceleration ramp of the slave axis • Coupling released	• Controlled deceleration ramp of the slave axis • Master axis not affected • Coupling released

Scenario 7

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
KeepCouplingWithSetValues	TriggerMasterError	Execution Error

Reaction to errors:

Master error	Slave error
• Controlled deceleration ramp for the master axis • The slave axis attempts to follow the setpoints of the master axis • Maintain the coupling	• Controlled deceleration ramp of the slave axis • Controlled deceleration ramp for the master axis • Coupling released

Scenario 8

Settings for the function block and the error type in question:

ReactionToMasterError	ReactionToSlaveError	Error Type
KeepCouplingWithSetValues	NoReaction	Execution Error

Reaction to errors:

Master error	Slave error
• Controlled deceleration ramp for the master axis • The slave axis attempts to follow the setpoints of the master axis • Maintain the coupling	• Controlled deceleration ramp of the slave axis • Master axis not affected • Coupling released

5.4 Tc3_Mc3Camming

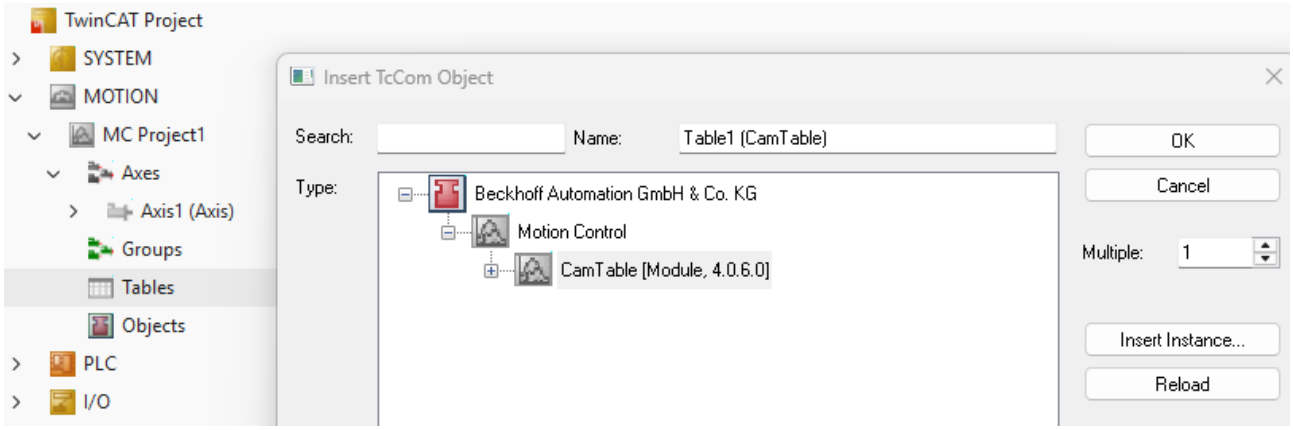
5.4.1 Tc3_Mc3Camming

5.4.1.1 Create a cam plate

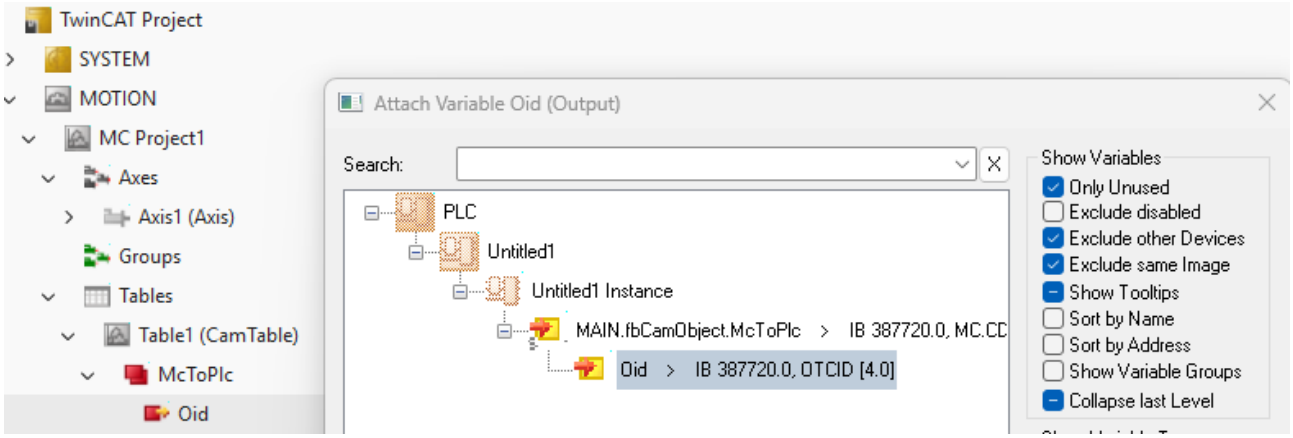
In MC3, the functions and operations related to cam plates are centralized in the MC_CamObject [▶ 333] function block. The Tc3_Mc3Camming library provides access to familiar PLCopen function blocks. In addition, the MC_CamObject class provides methods for creating and modifying cam plates, as well as readable properties of the active cam plate.

✓ A cam plate can be preconfigured in the project during configuration.

1. Open **Axes > Tables**.
2. Right-click the table in the structure tree and open the **Insert TcCOM Object** dialog.



⇨ In this case, the object ID of the module or TcCOM object must be linked to an MC_CamObject in the PLC.



⇨ At runtime, this connection is then initialized using the MC_CamObject.Connect() [▶ 334] method.

⇨ Alternatively, a cam plate can also be created at runtime. A TcCOM module is created by calling the MC_CamObject.Create() [▶ 335] method from the PLC. In this case, linking and further initialization are not necessary.

Fill a cam plate with MotionFunctionPoint as interpolation points

After successfully creating a cam plate object, it is then populated with the desired data points and interpolation functions. The definition of an individual data point is provided by the MotionFunctionPoint structure. Using the MC_CamObject.WriteData() [▶ 335] method, a series of data points (an array of MotionFunctionPoints) can be loaded synchronously into the cam plate object. Once this method has completed, the basic configuration of the cam plate is complete, and axes can be coupled using the defined relationship.

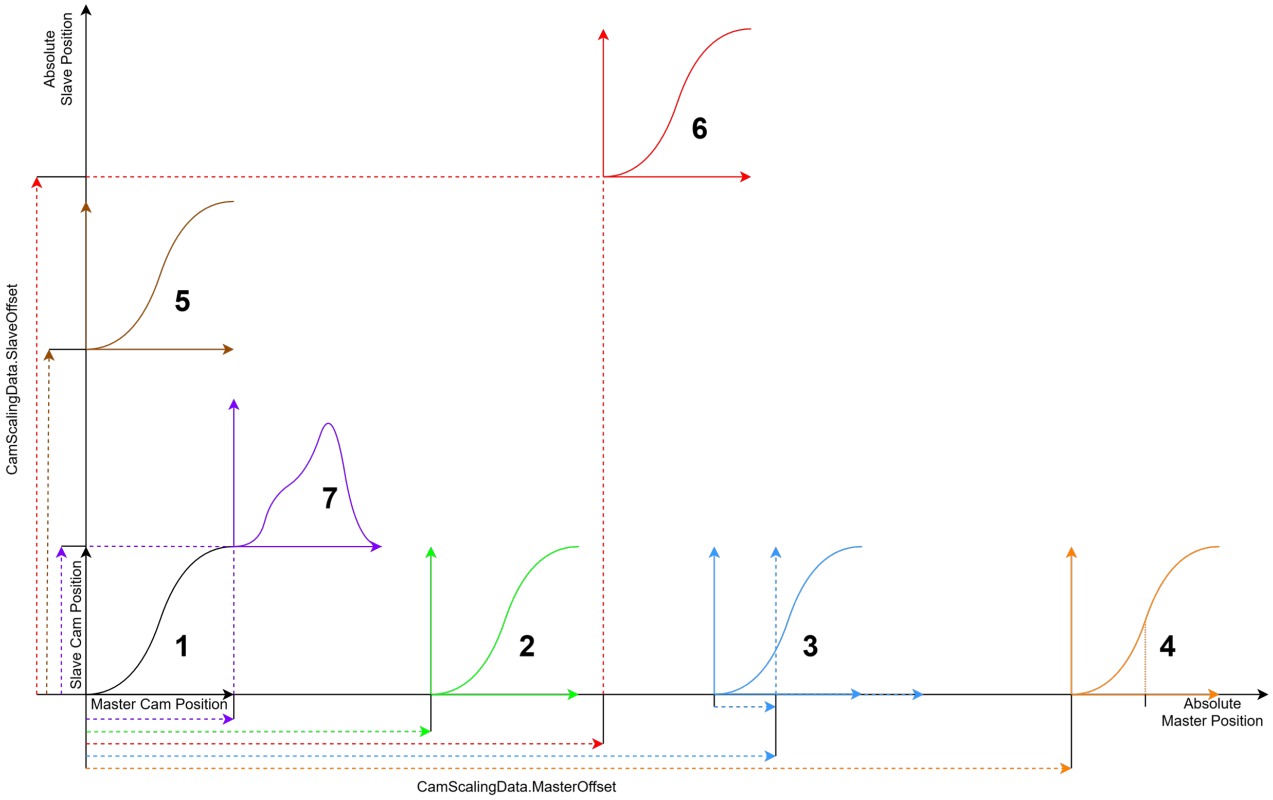
```
PROGRAM MAIN
VAR
    fbCamObject          : MC_CamObject;
    fbCamObject2         : MC_CamObject;
    motionFunctionPoints : ARRAY[0..360] OF MotionFunctionPoint;
    bInit                : BOOL;
    i                    : INT;
END_VAR
```

```
IF NOT bInit THEN

    fbCamObject.Connect();

    FOR i:=0 TO 360 DO
        motionFunctionPoints[i].Ignored           := FALSE;
        motionFunctionPoints[i].FunctionType      := EMotionFunctionType.CubicSpline;
        motionFunctionPoints[i].MasterPosition    := i;
        motionFunctionPoints[i].SlavePosition := -(COS(INT_TO_LREAL(i) / 360.0 * 2 * PI) - 1) * 100;
        motionFunctionPoints[i].SlaveVelocity     := STRING_TO_LREAL('#Ignore');
        motionFunctionPoints[i].SlaveAcceleration := STRING_TO_LREAL('#Ignore');
        motionFunctionPoints[i].SlaveJerk         := STRING_TO_LREAL('#Ignore');
    END_FOR
```

```
    fbCamObject.WriteData(SourceData := ADR(motionFunctionPoints), NumberOfPoints := 361,
Periodic:=FALSE);
    fbCamObject2.Create();
    fbCamObject2.CopyFrom(fbCamObject);

    bInit:=TRUE;
END_IF
```

5.4.1.2 CamScalingData: Using Master/SlaveOffsetMode

Examples of using ECamOffsetMode.Absolute/Relative in combination with CamActivationMode



1. No offset, instant activation/switching
   - MasterOffsetMode = None, SlaveOffsetMode = None, ActivationMode = Instantaneous
   - MasterOffsetMode = None, SlaveOffsetMode = None, ActivationMode = AtMasterCamPosition, ActivationPosition = 0.0
   - MasterOffsetMode = None, SlaveOffsetMode = None, ActivationMode = AtMasterAbsolutePosition, ActivationPosition = 0.0
2. Master offset (SlaveOffsetMode = None)
   - MasterOffsetMode = Absolute, MasterOffset = 100, ActivationMode = Instantaneous
   - MasterOffsetMode = Absolute, MasterOffset = 100, ActivationMode = AtMasterAbsolutePosition, ActivationPosition = 100
   - MasterOffsetMode = Absolute, MasterOffset = 100, ActivationMode = AtMasterCamPosition, ActivationPosition = 0
   - MasterOffsetMode = Relative, MasterOffset = 0, ActivationMode = AtMasterAbsolutePosition, ActivationPosition = 100
3. Relative master offset
   - MasterOffsetMode = Relative, MasterOffset = -10, ActivationMode = AtMasterAbsolutePosition, ActivationPosition = 200
   - The slave position jumps at the dashed line. This is the point at which the cam plate becomes active.
4. Master offset with ActivationPosition (recommended if the first section of a cam plate is not to be considered)

- MasterOffsetMode = Absolute, MasterOffset = 300, ActivationMode = AtMasterCamPosition, ActivationPosition = 10
- MasterOffsetMode = Absolute, MasterOffset = 300, ActivationMode = AtMasterAbsolutePosition, ActivationPosition = 310
- The slave position jumps at the dashed line. This is the point at which the cam plate becomes active.

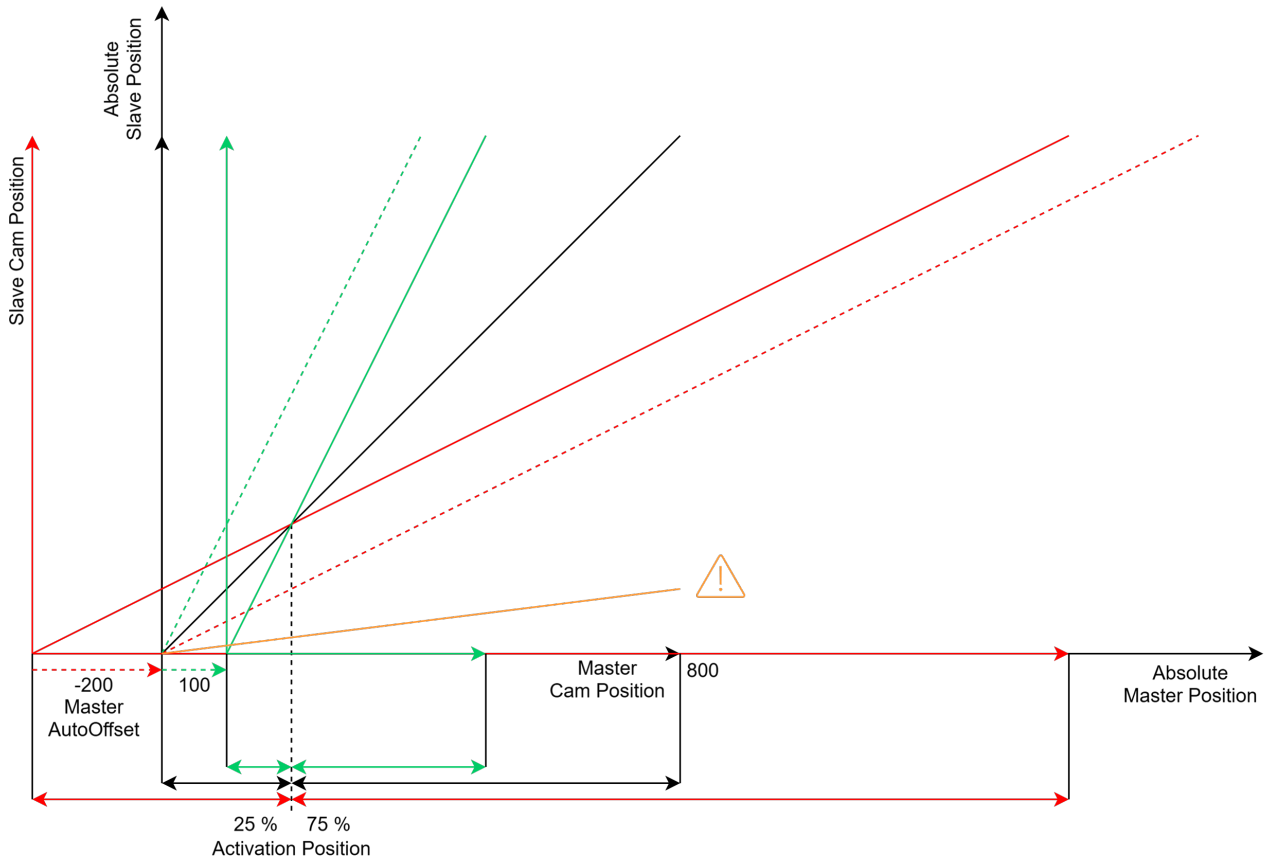
5. Slave Offset (MasterOffsetMode = None)
   - SlaveOffsetMode = Absolute, SlaveOffset = 100, ActivationMode = Instantaneous
   - SlaveOffsetMode = Relative, SlaveOffset = 100, ActivationMode = Instantaneous
6. Master and slave offset
   - MasterOffsetMode = Absolute, MasterOffset = 150, SlaveOffsetMode = Absolute, SlaveOffset = 150, ActivationMode = Instantaneous
   - MasterOffsetMode = Relative, MasterOffset = 150, SlaveOffsetMode = Relative, SlaveOffset = 150, ActivationMode = Instantaneous
7. No offset; activation/switching at the end/start of a cam plate
   - MasterOffsetMode = None, SlaveOffsetMode = None, ActivationMode = AtBoundary

Using ECamOffsetMode.AutoOffset

The ECamOffsetMode.AutoOffset mode ensures automatic adjustment of the cam plate offset. ECamOffsetMode.AutoOffset can be applied independently to the master and slave axes of a cam plate. To use AutoOffset, a cam plate must already be active. This means that the AutoOffset mode can be used when switching between or scaling cam plates.

Compared to TF5050 TwinCAT 3 NC Camming with MC_CamScalingMode and MC_StartMode, these settings are grouped under the enumeration type ECamOffsetMode in TF5550 TwinCAT 3 MC3 Camming. The ECamOffsetMode.AutoOffset setting can be used to prevent a jump in the position of the master and/or slave axis.

Examples of using ECamOffsetMode.AutoOffset for the master axis



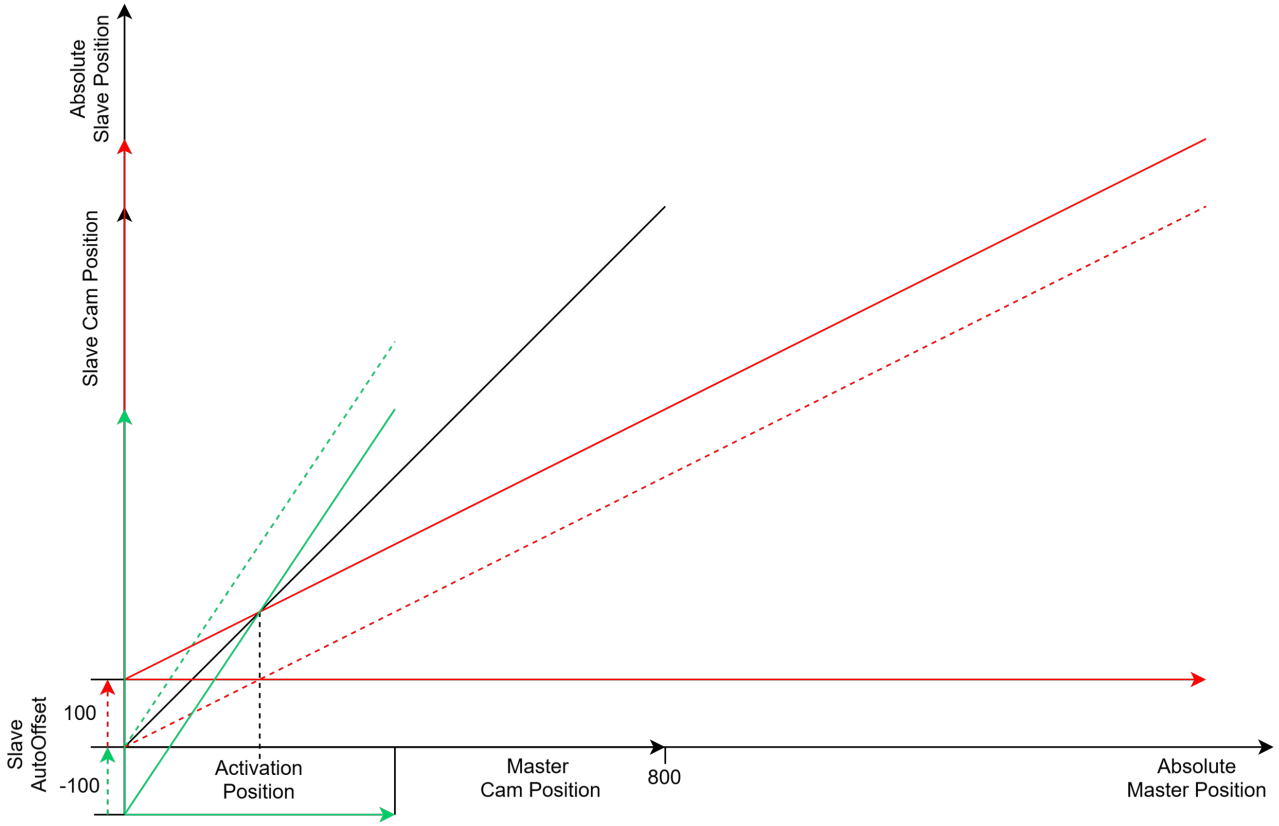
In the diagram shown, the cam plate is depicted in black color. The goal is to switch to the red, green, or orange cam plate. The original position of each cam plate (without offset) is shown with a dashed line, and the resulting position is shown with a solid line in the corresponding color. The switch occurs at a defined activation position (at Master Cam position 200 in the diagram).

ECamOffsetMode.AutoOffset for the master axis determines the master offset of the cam plate such that the master position within the cam plate is maintained. For master scaling or switchover to a cam plate with a different master cycle, this means that the relative (percentage) position before and after the switchover is identical. In this case, the red cam plate is automatically assigned a master offset of -200, and the green cam plate is assigned an offset of 100.

The red and green cam plates allow for adjustment, ensuring that no discontinuities occur even in the slave position.

In the example of the orange cam plate, there is no shift or automatic adjustment of the offset. The master scale, or master period, is the same for both the black and orange cam plates. In this case, the master position is maintained by the AutoOffset mode. Please note that the slave position jumps at the moment of switching.

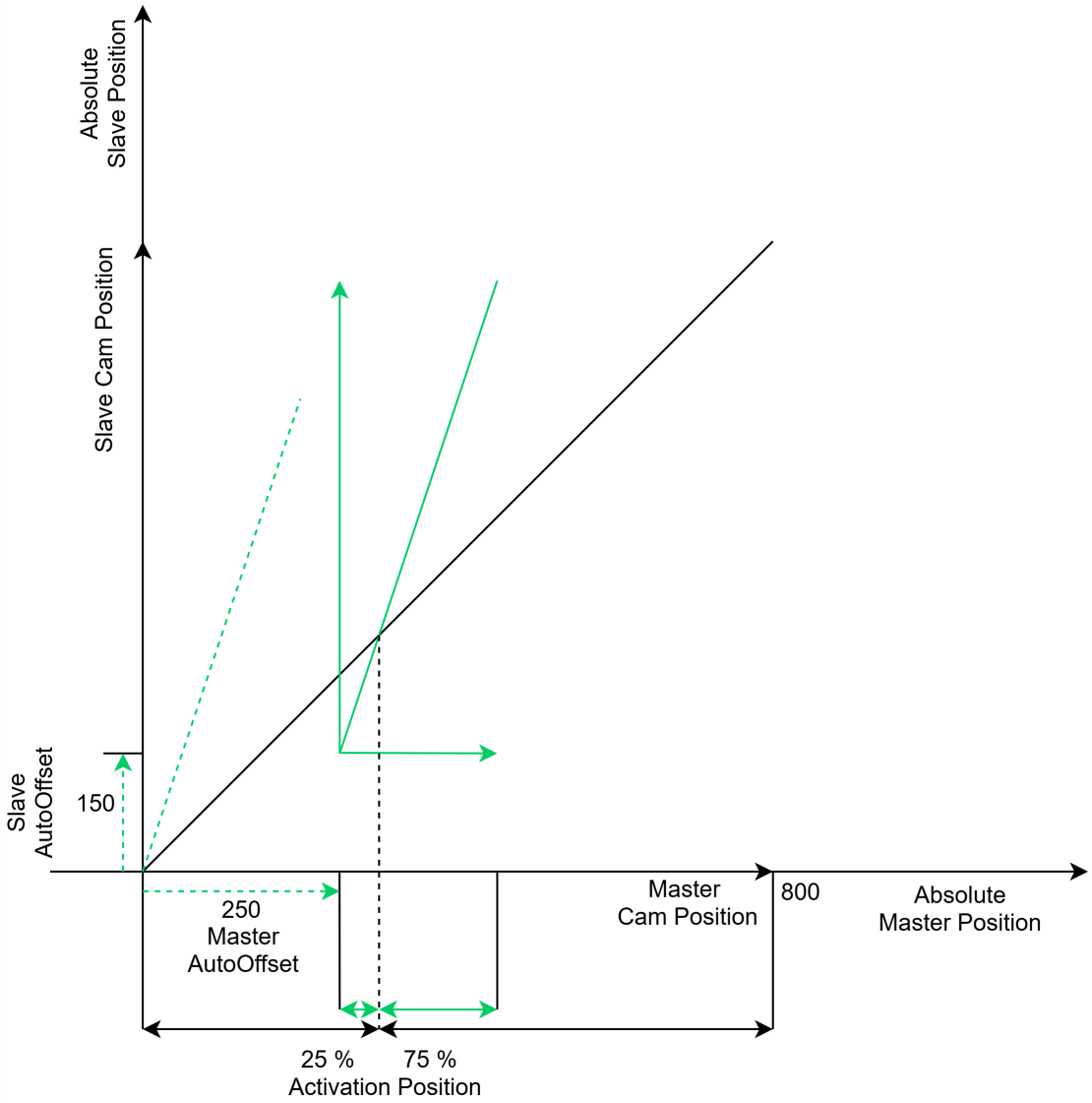
Examples of using ECamOffsetMode.AutoOffset for the slave axis



ECamOffsetMode.AutoOffset for the slave axis calculates the offset such that no discontinuities in the slave position occur during cam plate switching or scaling. The cam plate offset is adjusted to ensure that the slave position is identical before and after the action.

Examples of using ECamOffsetMode.AutoOffset for the master and slave axes

If ECamOffsetMode.AutoOffset is used for both the master axis and the slave axis during cam plate switching or scaling, the master offset is calculated first, followed by the slave offset.



5.4.2 Data Types

5.4.2.1 Enums

5.4.2.1.1 ECamActivationMode

Defines when a cam table activation or exchange takes effect.

Syntax

Definition:

```
TYPE ECamActivationMode :
(
    Instantaneous            := 0,
    AtBoundary               := 1,
    AtMasterCamPosition      := 2,
    AtMasterAbsolutePosition := 3
)UINT;
END_TYPE
```

Values

Name	Description
Instantaneous	Activation occurs during the next MET cycle.
AtBoundary	Activation occurs when either the start or end of the cam table is crossed.
AtMasterCamPosition	Activation occurs when the master cam position crosses the activation position. Activation position must be within the limits of the cam table.
AtMasterAbsolutePosition	Activation occurs when master axis position crosses the activation position.

Further information

`ECamActivationMode` determines how the corresponding command is processed or activated. Either the activation time is specified, or activation occurs at a position specified by another input.

ECamActivationMode.AtMasterCamPos

- This can be combined with CamScalingData.MasterOffsetMode = ECamOffsetMode.Relative if the ActivationPosition is determined from an already active cam plate coupling.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.1.2 ECamOffsetMode

Defines how master and slave offsets are applied during cam activation.

Syntax

Definition:

```
TYPE ECamOffsetMode :
(
    None       := 0,
    Absolute   := 1,
    Relative   := 2,
    AutoOffset := 3
)UINT;
END_TYPE
```

Values

Name	Description
None	Ignores input factors and takes default ones (Offset = 0, Scale = 1).
Absolute	Cam table and axis coordinate systems are deemed to be aligned. Offset and Scale are not affected.
Relative	Offset is calculated such the origin of the cam table coordinate system is aligned to the axis position at the activation time. If the input offset is not 0, it is added. Scale is not affected.
AutoOffset	Input offset is ignored and a new offset is calculated so that no position jump occurs during activation. Scale is not affected.

Further information

This enumeration type can be used to specify how a position offset should be interpreted for scaling a cam plate.

ECamOffsetMode.Relative:

- The orientation of the cam plate's coordinate system relative to the absolute master and slave positions must be known. It can be combined with ECamActivationMode.AtMasterCamPos if the ActivationPosition is determined from an already active cam plate coupling.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.1.3 EMotionFunctionType

Interpolation function type for cam table segments.

Syntax

Definition:

```
TYPE EMotionFunctionType :
(
    Undefined             := 0,
    Linear                := 1,
    Polynomial3           := 2,
    Polynomial5           := 3,
    Polynomial7           := 4,
    Sinusoid              := 5,
    SinusoidCombination   := 6,
    InclinedSinusoid      := 7,
    ModifiedSinusoid      := 8,
    HarmonicCombination   := 9,
    AccelerationTrapezoid := 10,
    CubicSpline           := 11
)UINT;
END_TYPE
```

Values

Name	Description
Undefined	No function type defined. Segment is not valid.
Linear	Linear interpolation between segment points.
Polynomial3	3rd-order polynomial interpolation.
Polynomial5	5th-order polynomial interpolation.
Polynomial7	7th-order polynomial interpolation.
Sinusoid	Simple sinusoidal motion law.
SinusoidCombination	Combined sinusoidal motion law.
InclinedSinusoid	Bestehorn sinusoidal motion law for minimal peak dynamic loads.
ModifiedSinusoid	Modified sinusoidal motion law.
HarmonicCombination	Harmonic combination motion law.
AccelerationTrapezoid	Trapezoidal acceleration profile motion law.
CubicSpline	Cubic spline interpolation through the segment points.

Further information

`EMotionFunctionType` specifies the mathematical method used to interpolate between two adjacent interpolation points. Based on the selected dynamics and boundary conditions, the system automatically determines whether the motion is a rest-to-rest (RR), velocity-to-velocity (VV), or other type of motion.

Transferring the MotionFunctionType from NC2 to MC3

MC_MotionFunctionType.[…]	EMotionFunctionType.[…]

MOTIONFUNCTYPE_NOTDEF	Undefined
MOTIONFUNCTYPE_POLYNOM1	Linear
MOTIONFUNCTYPE_POLYNOM3	Polynom3
MOTIONFUNCTYPE_POLYNOM5	Polynom5
MOTIONFUNCTYPE_POLYNOM7	Polynom7
MOTIONFUNCTYPE_POLYNOM8	On request
MOTIONFUNCTYPE_POLYNOM9	On request
MOTIONFUNCTYPE_SINUS LINE	Sinusoid
MOTIONFUNCTYPE_MODSINUSLINE	ModSinusoid
MOTIONFUNCTYPE_BESTEHORN	InclinedSinusoid
MOTIONFUNCTYPE_BESCHLTRAPEZ	AccTrapezoid
MOTIONFUNCTYPE_POLYNOM5_MM	Polynom5
MOTIONFUNCTYPE_HARMONIC_KOMBI_RT	HarmonicCombi
MOTIONFUNCTYPE_HARMONIC_KOMBI_TR	HarmonicCombi
MOTIONFUNCTYPE_HARMONIC_COMBI_VT	HarmonicCombi
MOTIONFUNCTYPE_HARMONIC_COMBO_TV	HarmonicCombi
MOTIONFUNCTYPE_BESCHLTRAPEZ_RT	AccTrapezoid
MOTIONFUNCTYPE_BESCHLTRAPEZ_TR	AccTrapezoid
MOTIONFUNCTYPE_MODSINUSLINIE_VV	ModSinusoid
MOTIONFUNCTYPE_POLYNOM7_MM	Polynom7
MOTIONFUNCTYPE_BESCHLTRAPEZ_RV	AccTrapezoid
MOTIONFUNCTYPE_BESCHLTRAPEZ_VR	AccTrapezoid
MOTIONFUNCTYPE_POLYNOM3_VV	Polynom3
MOTIONFUNCTYPE_POLYNOM11	OnRequest
MOTIONFUNCTYPE_PEISEKAH11	OnRequest
MOTIONFUNCTYPE_SPLINE	CubicSpline
MOTIONFUNCTYPE_SPLINE_NATURAL	Cubic Spline (transferable via boundary conditions)
MOTIONFUNCTYPE_SPLINE_TANGENTIAL	Cubic Spline (transferable via boundary conditions)
MOTIONFUNCTYPE_SPLINE_PERIODIC	Cubic Spline (transferable via boundary conditions)
MOTIONFUNCTYPE_SPLNE_LINEAR	Cubic Spline (transferable via boundary conditions)

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.2 Structs

5.4.2.2.1 CamActivationOptions

Options for cam table activation timing and reference.

Syntax

Definition:

```
TYPE CamActivationOptions :
STRUCT
    Mode            : ECamActivationMode;
    Position        : LREAL;
    ReferenceObject : Type_CouplingFunction1D;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
Mode	ECamActivationMode [▶ 314]	Instantaneous	Activation mode defining when the cam operation takes effect.
Position	LREAL	#Ignore	Master absolute position or master cam position at which the activation occurs. Only relevant for position-based activation modes.
ReferenceObject	Type_CouplingFunction1D [▶ 323]		Reference cam object for configured the activation options.

Further information

This structure contains the most important properties of a cam plate and can be retrieved using the MC_CamObject.ReadCharacteristics() method.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.2.2 CamCharacteristics

Computed characteristics of a cam table describing its motion profile properties.

Syntax

Definition:

```
TYPE CamCharacteristics :
STRUCT
    NominalMasterVelocity       : LREAL;
    StartPoint                  : CamProfilePoint;
    EndPoint                    : CamProfilePoint;
    PointAtMinSlavePosition     : CamProfilePoint;
    PointAtMinSlaveVelocity     : CamProfilePoint;
    PointAtMinSlaveAcceleration : CamProfilePoint;
    PointAtMinSlaveJerk         : CamProfilePoint;
    PointAtMaxSlavePosition     : CamProfilePoint;
    PointAtMaxSlaveVelocity     : CamProfilePoint;
    PointAtMaxSlaveAcceleration : CamProfilePoint;
    PointAtMaxSlaveJerk         : CamProfilePoint;
    SlaveMeanAbsoluteVelocity   : LREAL;
    SlaveEffectiveAcceleration  : LREAL;
    Oid                         : OTCID;
    NumberOfPoints              : UDINT;
    NumberOfActivePoints        : UDINT;
    NumberOfActiveSegments      : UDINT;
    IsValid                     : BOOL;
    IsPeriodic                  : BOOL;
    MinContinuityRequirement    : EContinuity;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
NominalMasterVelocity	LREAL	0.0	Nominal master velocity used to convert dimensionless slave values to time-based values.

Name	Type	Default	Description
StartPoint	CamProfilePoint [▶ 320]		Cam profile point at the start of the table.
EndPoint	CamProfilePoint [▶ 320]		Cam profile point at the end of the table.
PointAtMinSlavePosition	CamProfilePoint [▶ 320]		Cam profile point where the slave position reaches its minimum.
PointAtMinSlaveVelocity	CamProfilePoint [▶ 320]		Cam profile point where the slave velocity reaches its minimum.
PointAtMinSlaveAcceleration	CamProfilePoint [▶ 320]		Cam profile point where the slave acceleration reaches its minimum.
PointAtMinSlaveJerk	CamProfilePoint [▶ 320]		Cam profile point where the slave jerk reaches its minimum.
PointAtMaxSlavePosition	CamProfilePoint [▶ 320]		Cam profile point where the slave position reaches its maximum.
PointAtMaxSlaveVelocity	CamProfilePoint [▶ 320]		Cam profile point where the slave velocity reaches its maximum.
PointAtMaxSlaveAcceleration	CamProfilePoint [▶ 320]		Cam profile point where the slave acceleration reaches its maximum.
PointAtMaxSlaveJerk	CamProfilePoint [▶ 320]		Cam profile point where the slave jerk reaches its maximum.
SlaveMeanAbsoluteVelocity	LREAL	0.0	Mean absolute velocity of the slave over the entire cam table range.
SlaveEffectiveAcceleration	LREAL	0.0	RMS (effective) acceleration of the slave over the entire cam table range.
Oid	OTCID	0	Object ID of the cam table these characteristics belong to.
NumberOfPoints	UDINT	0	Total number of profile points in the cam table.
NumberOfActivePoints	UDINT	0	Number of non-ignored profile points in the cam table.
NumberOfActiveSegments	UDINT	0	Number of active interpolation segments between the profile points.
IsValid	BOOL	false	TRUE if the characteristics have been computed successfully.
IsPeriodic	BOOL	false	TRUE if the cam table is periodic (start and end boundary conditions match).

Name	Type	Default	Description
MinContinuityRequirement	EContinuity	PositionVelocityAcceleration	Minimum continuity level required for smooth operation of the cam table.

Further information

This structure contains the most important properties of a cam plate and can be retrieved using the MC_CamObject.ReadCharacteristics() method.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.2.3 CamProfilePoint

Describes a point on a cam profile with master position and slave motion state.

Syntax

Definition:

```
TYPE CamProfilePoint :
STRUCT
    MasterPosition    : LREAL;
    SlavePosition     : LREAL;
    SlaveVelocity     : LREAL;
    SlaveAcceleration : LREAL;
    SlaveJerk         : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
MasterPosition	LREAL	#Invalid	Master cam position at this profile point.
SlavePosition	LREAL	#Invalid	Slave cam position at this profile point.
SlaveVelocity	LREAL	#Invalid	Slave velocity at this profile point.
SlaveAcceleration	LREAL	#Invalid	Slave acceleration at this profile point.
SlaveJerk	LREAL	#Invalid	Slave jerk at this profile point.

Further information

A CamProfilePoint contains information about the interpolation points of a cam plate. The structure is used for the characteristics of a cam plate (CamCharacteristics), to, for example, retrieve extreme values and unambiguously assign them to an interpolation point on the cam plate.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.2.4 CamScalingData

Contains scaling and offset factors to be applied in cam table couplings.
MasterCamPos = (MasterPos - MasterOffset) / MasterScaling
SlaveCamPos = CamTable(MasterCamPos)
SlavePos = (SlaveCamPos * SlaveScaling) + SlaveOffset.

Syntax

Definition:

```
TYPE CamScalingData :
STRUCT
    MasterOffsetMode : ECamOffsetMode;
    MasterOffset     : LREAL;
    MasterScaling    : LREAL;
    SlaveOffsetMode  : ECamOffsetMode;
    SlaveOffset      : LREAL;
    SlaveScaling     : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
MasterOffsetMode	ECamOffsetMode [▶ 315]	None	Defines how the master offset is determined during cam activation.
MasterOffset	LREAL	0.0	Position offset applied to the master input before cam table lookup.
MasterScaling	LREAL	1.0	Scaling factor applied to the master input before cam table lookup.
SlaveOffsetMode	ECamOffsetMode [▶ 315]	None	Defines how the slave offset is determined during cam activation.
SlaveOffset	LREAL	0.0	Position offset added to the slave output after cam table evaluation.
SlaveScaling	LREAL	1.0	Scaling factor applied to the slave output after cam table evaluation.

Further information

A cam plate can be scaled using the MC_CamScaling function block for active cam plate couplings. Even when the coupling state is changed using MC_CamAdd, MC_CamIn, and MC_CamExchange, it is possible to scale the new cam plate coupling.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.2.5 MC_CAM_REF

Reference to a cam table point array for cam table definition.

Syntax

Definition:

```
TYPE MC_CAM_REF :
STRUCT
    PointsAddress  : Pointer To MotionFunctionPoint;
    NumberOfPoints : UDINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
PointsAddress	Pointer To MotionFunctionPoint [▶ 322]	Pointer to the source array of data points.
NumberOfPoints	UDINT	Number of data points in the source array.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.2.2.6 MotionFunctionPoint

Defines a cam table point with its interpolation function type and boundary conditions.

Syntax

Definition:

```
TYPE MotionFunctionPoint :
STRUCT
    FunctionType      : EMotionFunctionType;
    MasterPosition    : LREAL;
    SlavePosition     : LREAL;
    SlaveVelocity     : LREAL;
    SlaveAcceleration : LREAL;
    SlaveJerk         : LREAL;
    SymmetryRatio     : LREAL;
    Ignored           : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
FunctionType	EMotionFunctionType [▶ 316]	Undefined	Interpolation function used for the segment starting at this point.
MasterPosition	LREAL	#Invalid	Master axis position at this point.
SlavePosition	LREAL	#Invalid	Slave axis position at this point.
SlaveVelocity	LREAL	#Ignore	Slave velocity boundary condition at this point. Set to #Ignore for automatic calculation (given by interpolation).
SlaveAcceleration	LREAL	#Ignore	Slave acceleration boundary condition at this point. Set to #Ignore for automatic calculation (given by interpolation).

Name	Type	Default	Description
SlaveJerk	LREAL	#Ignore	Slave jerk boundary condition at this point. Set to #Ignore for automatic calculation (given by interpolation).
SymmetryRatio	LREAL	#Ignore	Normalized inflection point is valid only for rest-to-rest motion functions. This parameter is ignored if it is out of range or set for non-rest-to-rest motion functions.
Ignored	BOOL	false	If TRUE, this point is skipped during cam table compilation.

Further information

The MotionFunctionPoint data structure describes an interpolation point of a cam plate.

- FunctionType
  - Specifies the mathematical function that determines the interpolation between two interpolation points. The starting point for the interpolation is the current interpolation point.
- SlaveVelo
  - Velocity of the slave axes at this interpolation point. If the slave velocity is determined implicitly by the selected FunctionType, select the value STRING_TO_LREAL('#Ignore') (default). If a different value is specified, however, it will be rejected with an error during MC_CamObject.Construct(), and an error description will be displayed in the Error List.
- SlaveAcc
  - Acceleration of the slave axes at this interpolation point. If the slave acceleration is determined implicitly by the selected FunctionType, STRING_TO_LREAL('#Ignore') (default) must be used. If a different value is specified, however, it will be rejected with an error during MC_CamObject.Construct(), and an error description will be displayed in the Error List.
- SlaveJerk
  - Jerk of the slave axes at this interpolation point. If the slave jerk is determined implicitly by the selected FunctionType, STRING_TO_LREAL('#Ignore') (default) must be used. If a different value is specified, however, it will be rejected with an error during MC_CamObject.Construct(), and an error description will be displayed in the Error List.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.3 Interfaces

5.4.3.1 Types

5.4.3.1.1 Type_CouplingFunction1D

Type for a motion control object that provides a 1D coupling function (e. g. supported by MC_CamObject).

Inheritance Hierarchy

Type_CouplingFunction1D

Methods

Name	Origin	Description
GetCouplingFunc1DOid [▶ 324]	Type_CouplingFunction1D	

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.4.3.1.1.1 GetCouplingFunc1DOid

Syntax

Definition:

```
METHOD GetCouplingFunc1DOid : OTCID
```

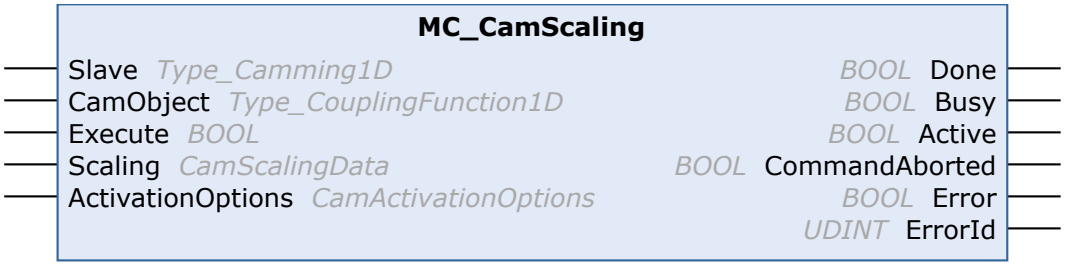
Return value

OTCID

5.4.4 Function Blocks

5.4.4.1 Administrative

5.4.4.1.1 MC_CamScaling



This function block applies the scaling factor of an existing cam table coupling on a slave. InSync of the existing cam table coupling is not affected by this function block.

Syntax

Definition:

```
FUNCTION_BLOCK MC_CamScaling
VAR_INPUT
    Slave             : Type_Camming1D;
    CamObject         : Type_CouplingFunction1D;
    Execute           : BOOL;
    Scaling           : CamScalingData;
    ActivationOptions : CamActivationOptions;
END_VAR
VAR_OUTPUT
    Done              : BOOL;
    Busy              : BOOL;
    Active            : BOOL;
    CommandAborted    : BOOL;
```

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

Inputs

Name	Type	Description
Slave	Type_Camming1D	Reference to the slave axis
CamObject	Type_CouplingFunction1D [▶ 323]	Reference to the cam object
Execute	BOOL	Apply cam scaling at rising edge
Scaling	CamScalingData [▶ 321]	New cam scaling
ActivationOptions	CamActivationOptions [▶ 317]	Options controlling how the cam coupling is activated (mode, position, reference object).

Outputs

Name	Type	Description
Done	BOOL	Cam scaling succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Further information

A cam plate coupling can be scaled with the function block MC_CamScaling. The raw table data of the cam plate are not affected; instead, the scaling refers to an active master/slave coupling. The following parameters can be modified: scaling factors for master and slave, and offsets for the cam plate within the coordinate system.

Optionally, the modification will only take effect from a certain master position, enabling precise scaling during the motion.

NOTICE

Property damage due to modifying an active cam plate coupling

This function block can also be used for an active cam plate. Valid changes are applied to the cam plate and may result in setpoint jumps. To activate cam plate scaling at a specific time or at a specific position, the following procedure is recommended:

- Use ECamActivationMode.AtBoundary, ECamActivationMode.AtMasterCamPosition, or ECamActivationMode.AtMasterAbsolutePosition
- Alternatively, create a copy of the active cam plate (using the CopyFrom() method) and make the changes to the copy. Next, switch between the cam plates (MC_CamExchange [▶ 329]).

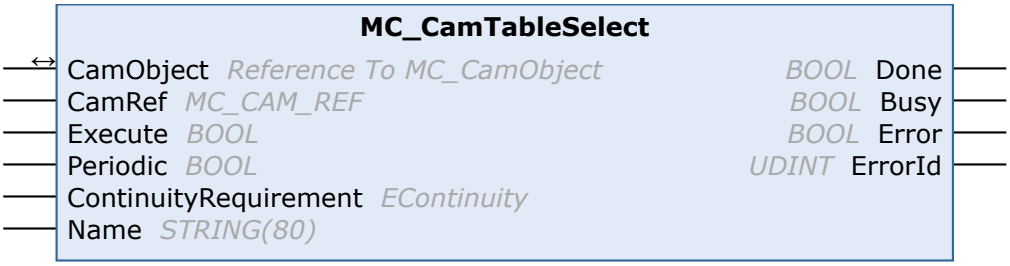
Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.1.2 MC_CamTableSelect



This function block selects the cam table. Internally the function block calls CamObject.Connect() if the cam object is linked to an existing cam table object, otherwise CamObject.Create(), and CamObject.Construct() are called based on CamRef data.

Syntax

Definition:

```
FUNCTION_BLOCK MC_CamTableSelect
VAR_IN_OUT
    CamObject             : Reference To MC_CamObject;
END_VAR
VAR_INPUT
    CamRef                : MC_CAM_REF;
    Execute               : BOOL;
    Periodic              : BOOL;
    ContinuityRequirement : EContinuity;
    Name                  : STRING(80);
END_VAR
VAR_OUTPUT
    Done                  : BOOL;
    Busy                  : BOOL;
    Error                 : BOOL;
    ErrorId               : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
CamRef	MC_CAM_REF [▶ 321]		Data point information.
Execute	BOOL		Start cam select at rising edge.
Periodic	BOOL		Set to TRUE for a periodic cam table.
ContinuityRequirement	EContinuity		Minimum continuity requirement. The higher the requirement, the stricter the point validation.
Name	STRING(80)	""	Only applicable if the cam object is not linked and creation of a new cam table object is required.

In/Outputs

Name	Type	Description
CamObject	Reference To MC_CamObject [▶ 333]	Reference to the cam object

Outputs

Name	Type	Description
Done	BOOL	Selection succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.

Name	Type	Description
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

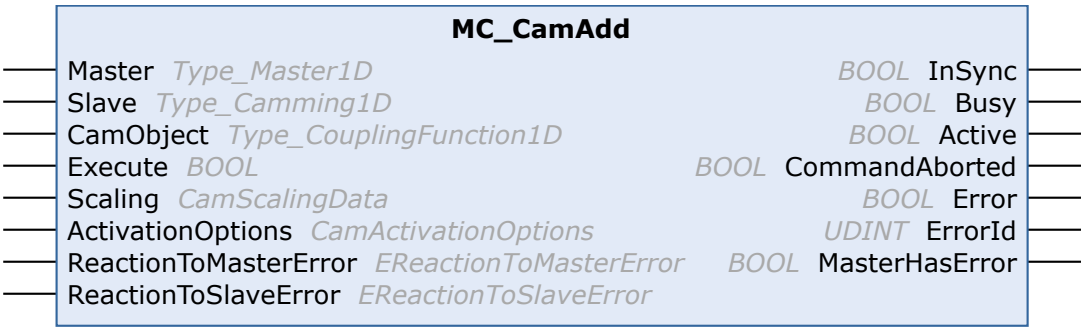
- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.2 Motion

5.4.4.2.1 MC_CamAdd



This function block adds a cam table coupling on a slave. Generated setpoints from single cam table couplings are internally added.
As MC_CamIn, this function block can also be used to initiate camming-based synchronization.
As this function block is used for multi-camming, some special rules apply:
The same master can be used in multiple cam table couplings. A cam table can only be used once in a slave.

Syntax

Definition:

```
FUNCTION_BLOCK MC_CamAdd
VAR_INPUT
    Master                : Type_Master1D;
    Slave                 : Type_Camming1D;
    CamObject             : Type_CouplingFunction1D;
    Execute               : BOOL;
    Scaling               : CamScalingData;
    ActivationOptions     : CamActivationOptions;
    ReactionToMasterError : EReactionToMasterError;
    ReactionToSlaveError  : EReactionToSlaveError;
END_VAR
VAR_OUTPUT
    InSync                : BOOL;
    Busy                  : BOOL;
    Active                : BOOL;
    CommandAborted        : BOOL;
    Error                 : BOOL;
    ErrorId               : UDINT;
    MasterHasError        : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis

Name	Type	Default	Description
Slave	Type_Camming1D		Reference to the slave axis
CamObject	Type_CouplingFunction1D [▶ 323]		Reference to the cam object
Execute	BOOL		Start cam add at rising edge.
Scaling	CamScalingData [▶ 321]		Scaling of the new coupling
ActivationOptions	CamActivationOptions [▶ 317]		Options controlling how the cam coupling is activated (mode, position, reference object).
ReactionToMasterError	EReactionToMasterError	TriggerSlaveError	Reaction if an error occurs in the master while this command is active.
ReactionToSlaveError	EReactionToSlaveError	TriggerMasterError	Reaction if an error occurs in the slave while this command is active.

Outputs

Name	Type	Description
InSync	BOOL	Synchronization succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
MasterHasError	BOOL	Master is in error state.

Further information:

The same rules apply to ActivationOptions.ReferenceObject as to MC_CamIn. ActivationOptions.ReferenceObject:

- ReferenceObject is not required when curve synchronization is initiated (the master curve table coupling of this function block is used for activation).
- By default, the slave's active master curve table coupling is used, provided that only one exists (ReferenceObject = 0). If there are multiple slaves, set ReferenceObject = CamObject to ensure proper synchronization.
- ReferenceObject must not be 0 if there is more than one active master curve table coupling.

MC_CamAdd allows you to program cam plate couplings for a slave axis to multiple different master axes. It should be noted that a master axis can be used in multiple cam plate couplings, i.e., for multiple slave axes. An MC_CamObject can be used only once for the same slave axis.

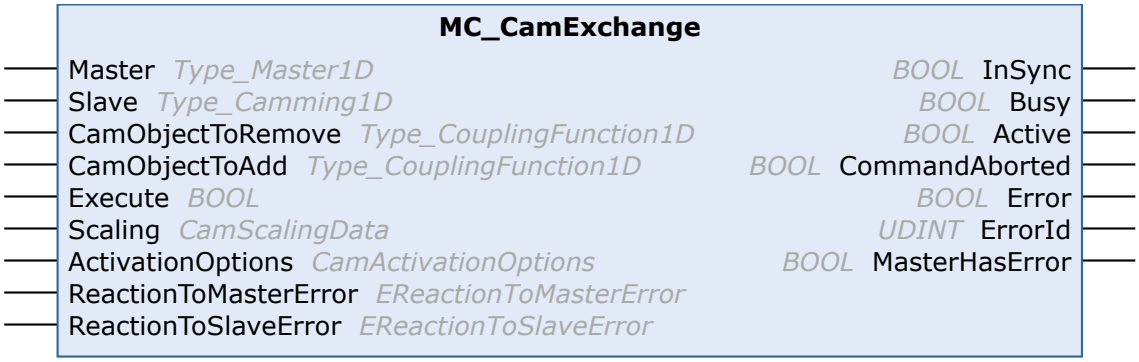
Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.2.2 MC_CamExchange



This function block exchanges cam table couplings on a slave. The cam table coupling that should be removed has to be active when this function block is executed and the activation condition is met, otherwise an error is triggered. Activation might depend either on the new entered cam table coupling (ActivationOptions.ReferenceObject = ECamActivationRef.NewlyEnteredCamTable) or on the one to be removed (ActivationOptions.ReferenceObject = ECamActivationRef.CamTableAlreadyInUse)

Syntax

Definition:

```
FUNCTION_BLOCK MC_CamExchange
VAR_INPUT
    Master                : Type_Master1D;
    Slave                 : Type_Camming1D;
    CamObjectToRemove     : Type_CouplingFunction1D;
    CamObjectToAdd        : Type_CouplingFunction1D;
    Execute               : BOOL;
    Scaling               : CamScalingData;
    ActivationOptions     : CamActivationOptions;
    ReactionToMasterError : EReactionToMasterError;
    ReactionToSlaveError  : EReactionToSlaveError;
END_VAR
VAR_OUTPUT
    InSync                : BOOL;
    Busy                  : BOOL;
    Active                : BOOL;
    CommandAborted        : BOOL;
    Error                 : BOOL;
    ErrorId               : UDINT;
    MasterHasError        : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Type_Camming1D		Reference to the slave axis
CamObjectToRemove	Type_CouplingFunction1D [▶ 323]		Cam object to be removed
CamObjectToAdd	Type_CouplingFunction1D [▶ 323]		Cam object to be added
Execute	BOOL		Start cam exchange at rising edge.
Scaling	CamScalingData [▶ 321]		Scaling of the new coupling
ActivationOptions	CamActivationOptions [▶ 317]		Options controlling how the cam coupling is activated (mode, position, reference object).
ReactionToMasterError	EReactionToMasterError	TriggerSlaveError	Reaction if an error occurs in the master while this command is active.
ReactionToSlaveError	EReactionToSlaveError	TriggerMasterError	Reaction if an error occurs in the slave while this command is active.

Outputs

Name	Type	Description
InSync	BOOL	Synchronization succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
MasterHasError	BOOL	Master is in error state.

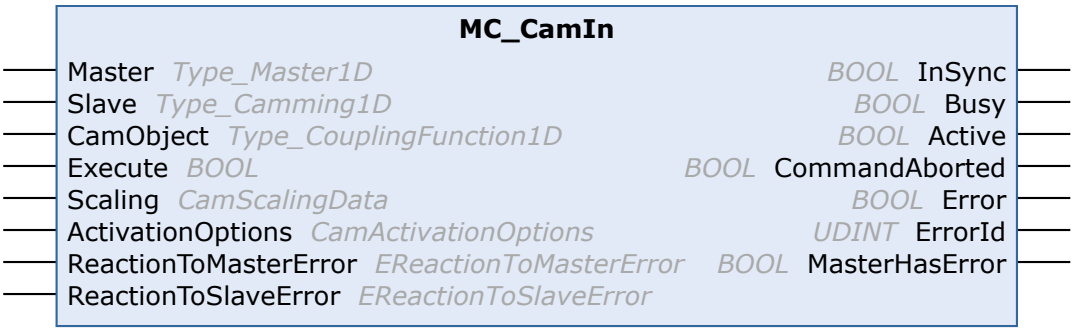
Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.2.3 MC_CamIn



This function block initiates camming-based synchronization on a slave.
If the slave is already coupled to one or more cam table couplings, it switches to the cam table coupling that is entered in this function block.
Note: Use MC_CamAdd to switch from single-camming to multi-camming. Switch back and forth between single-camming and multi-camming is possible and not limited.

Syntax

Definition:

```
FUNCTION_BLOCK MC_CamIn
VAR_INPUT
    Master                : Type_Master1D;
    Slave                 : Type_Camming1D;
    CamObject             : Type_CouplingFunction1D;
    Execute               : BOOL;
    Scaling               : CamScalingData;
    ActivationOptions     : CamActivationOptions;
    ReactionToMasterError : EReactionToMasterError;
    ReactionToSlaveError  : EReactionToSlaveError;
END_VAR
VAR_OUTPUT
    InSync                : BOOL;
    Busy                  : BOOL;
    Active                : BOOL;
    CommandAborted        : BOOL;
    Error                 : BOOL;
```

```
    ErrorId               : UDINT;
    MasterHasError        : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Master	Type_Master1D		Reference to the master axis
Slave	Type_Camming1D		Reference to the slave axis
CamObject	Type_CouplingFunction1D [▶ 323]		Reference to the cam object
Execute	BOOL		Start cam in at rising edge.
Scaling	CamScalingData [▶ 321]		Scaling of the new coupling
ActivationOptions	CamActivationOptions [▶ 317]		Options controlling how the cam coupling is activated (mode, position, reference object).
ReactionToMasterError	EReactionToMasterError	TriggerSlaveError	Reaction if an error occurs in the master while this command is active.
ReactionToSlaveError	EReactionToSlaveError	TriggerMasterError	Reaction if an error occurs in the slave while this command is active.

Outputs

Name	Type	Description
InSync	BOOL	Synchronization succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier
MasterHasError	BOOL	Master is in error state.

Further information

The following rules apply to ActivationOptions.ReferenceObject:

- ReferenceObject is not required when curve synchronization is initiated (the master curve table coupling of this function block is used for activation).
- By default, the slave's active master curve table coupling is used, provided that only one exists (ReferenceObject = 0). If there are multiple slaves, set ReferenceObject = CamObject to ensure proper synchronization.
- ReferenceObject must not be 0 if there is more than one active master curve table coupling.

Additional information on ReactionToSlaveError and ReactionToMasterError

As a general rule, if there is an error on the slave axis:

- Slave axis: The coupling is released, and the response depends on the error type.
- Master Axis: The master axis reports an error and either follows the MC's stop ramp or continues its current motion without being affected.

If there is an error on the master axis:

- Master axis: The response depends on the type of error.

- Slave axis: The slave axis reports an error and follows either the MC's stop ramp or the master axis's setpoints.

Detailed information is provided in the Error behavior [▶ 306] chapter.

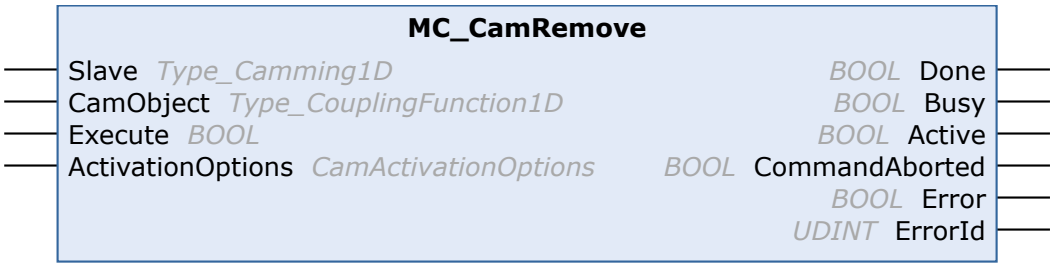
Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.2.4 MC_CamRemove



This function block removes a cam table coupling from a slave. As a cam table can only be used once in a coupled slave, the master information is not needed. However, the cam table must be in use when this function block is executed and when the activation condition is met, otherwise an error is triggered.
If there is no active coupling left and there are no pending commands to add couplings, the camming synchronization ends by itself and axis state changes to Standstill.

Syntax

Definition:

```
FUNCTION_BLOCK MC_CamRemove
VAR_INPUT
    Slave             : Type_Camming1D;
    CamObject         : Type_CouplingFunction1D;
    Execute           : BOOL;
    ActivationOptions : CamActivationOptions;
END_VAR
VAR_OUTPUT
    Done              : BOOL;
    Busy              : BOOL;
    Active            : BOOL;
    CommandAborted    : BOOL;
    Error             : BOOL;
    ErrorId           : UDINT;
END_VAR
```

Inputs

Name	Type	Description
Slave	Type_Camming1D	Reference to the slave axis
CamObject	Type_CouplingFunction1D [▶ 323]	Cam object to be removed (reference to the object)
Execute	BOOL	Start cam remove at rising edge.
ActivationOptions	CamActivationOptions [▶ 317]	Options controlling how the cam coupling is activated (mode, position, reference object).

Outputs

Name	Type	Description
Done	BOOL	Removal succeeded
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.3 Motion Object

5.4.4.3.1 MC_CamObject

Wrapper that holds a reference to a cam table object and provides methods for setting, accessing and modifying its data.
To use this wrapper, the cam table must be mapped and the method Connect() must be called.
Once a reference is successfully stored, it cannot be modified. Subsequent calls to Connect() or Create() will return an error.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Connect [▶ 334]	Stores a reference to an existing cam table object. The cam table must have been previously linked to McToPlc, or the object ID must be passed as parameter.
Create [▶ 335]	Creates a new cam table object and stores a reference to it.
WriteData [▶ 335]	Calculates the entire cam table from a new set of data points.
ReadData [▶ 336]	Reads data points from index StartIndex to StartIndex + NumberOfPoints - 1. The first valid index is 1.
OverwriteData [▶ 336]	Overwrites data points from index StartIndex to StartIndex + NumberOfPoints - 1 and recalculates the modified cam table segments. The first valid index is 1.
ReadSlaveDynamics [▶ 337]	Evaluates slave position, velocity and acceleration at a specific master position within the cam table profile.
ComputeMasterPositions [▶ 338]	Finds the master position(s) for a given slave position.
ReadMasterRange [▶ 339]	Returns leftmost and rightmost master position of the cam table profile.
ComputeCharacteristics [▶ 339]	Computes cam table characteristics such as extreme slave dynamics for a given nominal master velocity.
ClearData [▶ 340]	Removes all stored data. The existing reference remains intact.

Name	Description
CopyFrom [▶ 340]	Copies data points and cam table segments from another valid MC_CamObject.

Further information

The MC_CamObject is the central object for handling cam plates. This includes, among other things, methods for setting, reading, and modifying data points. To use this wrapper, you must first obtain a reference to a curve table using one of the following options:

1. Linking to an existing curve table at configuration time; see Create a cam plate [▶ 309]:
   - Link the “McToPlc.Oid” input of the Cam object in the PLC to the “McToPlc.Oid” output of an existing curve table in the MC3 Motion project.
   - Call the Connect() [▶ 334] method in the PLC to initialize the linking. Once the link has been successfully created, it cannot be changed. Calling “Connect()” again will result in an error.
2. Creating a new curve table at runtime in the PLC: Call Create() [▶ 335] to internally request the instantiation of a new curve table object and store a reference to this newly created object. Optionally, a name can be specified as a function block input. After that, an MC_CamObject is ready for use. Calling “Create()” again will result in an error.
3. The cam plate is calculated based on the given data points using the WriteData() [▶ 335] method.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5550 MC3 Camming >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Camming

5.4.4.3.1.1 Connect

Stores a reference to an existing cam table object. The cam table must have been previously linked to McToPlc, or the object ID must be passed as parameter.

Syntax

Definition:

```
METHOD Connect : UDINT
VAR_INPUT
    OptionalOid : OTCID;
END_VAR
```

Inputs

Name	Type	Description
OptionalOid	OTCID	Cam table object ID, used when McToPlc is not linked. If both are set, McToPlc takes priority.

Return value

UDINT

5.4.4.3.1.2 Create

Creates a new cam table object and stores a reference to it.

Syntax

Definition:

```
METHOD Create : UDINT
VAR_INPUT
    Name : STRING(80);
END_VAR
```

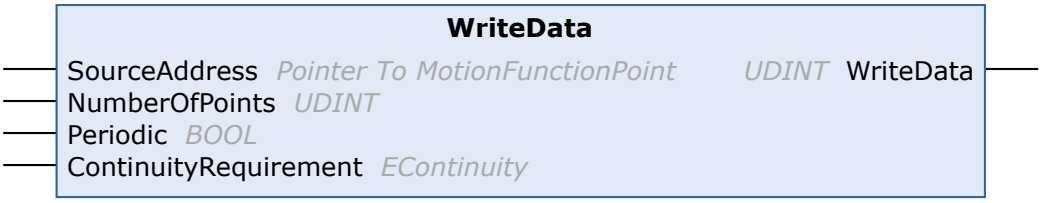
Inputs

Name	Type	Description
Name	STRING(80)	Optional name for the new cam table.

Return value

UDINT

5.4.4.3.1.3 WriteData



Calculates the entire cam table from a new set of data points.

Syntax

Definition:

```
METHOD WriteData : UDINT
VAR_INPUT
    SourceAddress         : Pointer To MotionFunctionPoint;
    NumberOfPoints        : UDINT;
    Periodic              : BOOL;
    ContinuityRequirement : EContinuity;
END_VAR
```

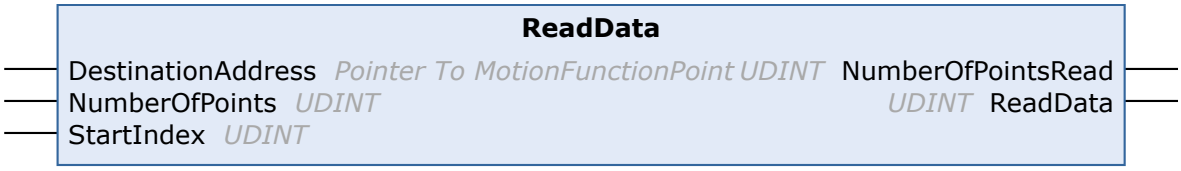
Inputs

Name	Type	Description
SourceAddress	Pointer To MotionFunctionPoint [▶ 322]	Pointer to the new set of data points.
NumberOfPoints	UDINT	Number of data points in the source array.
Periodic	BOOL	TRUE for a periodic cam table.
ContinuityRequirement	EContinuity	Minimum continuity requirement. The higher the requirement, the stricter the point validation.

Return value

UDINT

5.4.4.3.1.4 ReadData



Reads data points from index StartIndex to StartIndex + NumberOfPoints - 1. The first valid index is 1.

Syntax

Definition:

```
METHOD ReadData : UDINT
VAR_INPUT
    DestinationAddress : Pointer To MotionFunctionPoint;
    NumberOfPoints     : UDINT;
    StartIndex         : UDINT;
END_VAR
VAR_OUTPUT
    NumberOfPointsRead : UDINT;
END_VAR
```

Inputs

Name	Type	Description
DestinationAddress	Pointer To MotionFunctionPoint [▶ 322]	Pointer to the destination buffer for the requested data points.
NumberOfPoints	UDINT	Number of points to read.
StartIndex	UDINT	Index of the first point to read.

Outputs

Name	Type	Description
NumberOfPointsRead	UDINT	The actual number of points read, which can be less than or equal to the specified NumberOfPoints.

Return value

UDINT

5.4.4.3.1.5 OverwriteData

Overwrites data points from index StartIndex to StartIndex + NumberOfPoints - 1 and recalculates the modified cam table segments. The first valid index is 1.

Syntax

Definition:

```
METHOD OverwriteData : UDINT
VAR_INPUT
    SourceAddress  : Pointer To MotionFunctionPoint;
    NumberOfPoints : UDINT;
    StartIndex     : UDINT;
END_VAR
```

Inputs

Name	Type	Description
SourceAddress	Pointer To MotionFunctionPoint [▶ 322]	Pointer to the new data points to write.
NumberOfPoints	UDINT	Number of points to overwrite.
StartIndex	UDINT	Index of the first point to overwrite.

Return value

UDINT

Further information

The property allows writing or overwriting a defined data range of the cam plate.

NOTICE

Property damage due to modifying an active cam plate coupling

This method can also be called while a cam plate is active. Valid changes are applied synchronously to the cam plate and may result in setpoint jumps. To switch cam plates at a specific time or position, the following steps are recommended:

- Create a copy of the active cam plate (using the CopyFrom() method).
- Make the changes to the copy (using the OverwriteData() method).
- Switch between the cam plates (MC_CamExchange [▶ 329]).

5.4.4.3.1.6 ReadSlaveDynamics

Evaluates slave position, velocity and acceleration at a specific master position within the cam table profile.

Syntax

Definition:

```
METHOD ReadSlaveDynamics : UDINT
VAR_INPUT
    MasterPosition    : LREAL;
END_VAR
VAR_OUTPUT
    SlavePosition     : LREAL;
    SlaveVelocity     : LREAL;
    SlaveAcceleration : LREAL;
END_VAR
```

Inputs

Name	Type	Description
MasterPosition	LREAL	Master cam position at which to evaluate the slave dynamics.

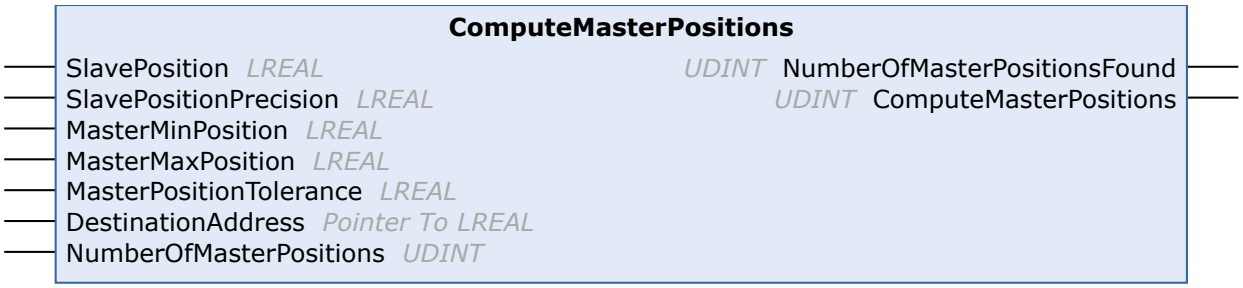
Outputs

Name	Type	Description
SlavePosition	LREAL	Computed slave position at the given master cam position.
SlaveVelocity	LREAL	Computed slave velocity at the given master cam position.
SlaveAcceleration	LREAL	Computed slave acceleration at the given master cam position.

Return value

UDINT

5.4.4.3.1.7 ComputeMasterPositions



Finds the master position(s) for a given slave position.

Syntax

Definition:

```
METHOD ComputeMasterPositions : UDINT
VAR_INPUT
    SlavePosition               : LREAL;
    SlavePositionPrecision      : LREAL;
    MasterMinPosition           : LREAL;
    MasterMaxPosition           : LREAL;
    MasterPositionTolerance     : LREAL;
    DestinationAddress          : Pointer To LREAL;
    NumberOfMasterPositions     : UDINT;
END_VAR
VAR_OUTPUT
    NumberOfMasterPositionsFound : UDINT;
END_VAR
```

Inputs

Name	Type	Description
SlavePosition	LREAL	Position of the slave axis for which the master positions are to be determined.
SlavePositionPrecision	LREAL	Absolute precision for which a slave position at a certain master position is considered equal to the SlavePosition input (must be larger than 0, e.g. 1E-3).
MasterMinPosition	LREAL	Leftmost limit of the cam profile range to be evaluated.
MasterMaxPosition	LREAL	Rightmost limit of the cam profile range to be evaluated.

Name	Type	Description
MasterPositionTolerance	LREAL	Minimum distance between distinct master positions found (must be larger than 0, e.g. 1E-3).
DestinationAddress	Pointer To LREAL	Pointer to the destination array for found master positions.
NumberOfMasterPositions	UDINT	Capacity of the DestinationAddress array.

Outputs

Name	Type	Description
NumberOfMasterPositionsFound	UDINT	Number of master positions found matching the given slave position and master position range.

Return value

UDINT

5.4.4.3.1.8 ReadMasterRange

Returns leftmost and rightmost master position of the cam table profile.

Syntax

Definition:

```
METHOD ReadMasterRange : UDINT
VAR_OUTPUT
    MasterMinPosition : LREAL;
    MasterMaxPosition : LREAL;
END_VAR
```

Outputs

Name	Type	Description
MasterMinPosition	LREAL	Leftmost master position of the cam table profile.
MasterMaxPosition	LREAL	Rightmost master position of the cam table profile.

Return value

UDINT

5.4.4.3.1.9 ComputeCharacteristics

Computes cam table characteristics such as extreme slave dynamics for a given nominal master velocity.

Syntax

Definition:

```
METHOD ComputeCharacteristics : UDINT
VAR_INPUT
    NominalMasterVelocity : LREAL;
    Destination           : Reference To CamCharacteristics;
END_VAR
```

Inputs

Name	Type	Description
NominalMaster Velocity	LREAL	Nominal master velocity used for computing absolute slave dynamics.
Destination	Reference To CamCharacteristics [▶ 318]	Reference to a CamCharacteristics structure to receive the computed results.

Return value

UDINT

5.4.4.3.1.10 ClearData

Removes all stored data. The existing reference remains intact.

Syntax

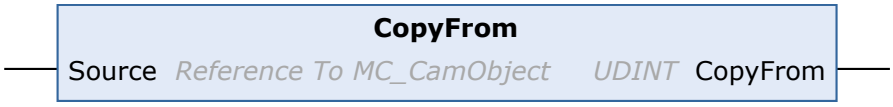
Definition:

```
METHOD ClearData : UDINT
```

Return value

UDINT

5.4.4.3.1.11 CopyFrom



Copies data points and cam table segments from another valid MC_CamObject.

Syntax

Definition:

```
METHOD CopyFrom : UDINT
VAR_INPUT
    Source : Reference To MC_CamObject;
END_VAR
```

Inputs

Name	Type	Description
Source	Reference To MC_CamObject [▶ 333]	Source MC_CamObject to copy from.

Return value

UDINT

5.4.5 Appendix

Upgrade from NC2 to MC3

With the MC3, the cam plate functionality takes a new, more object-oriented approach. The central element is the MC_CamObject [▶ 333].

Function blocks

NC2 function block	MC3 function block	Note
MC_CamIn.InSync MC_CamIn.Busy MC_CamIn.Active	MC_CamIn [▶ 330].InSync MC_CamIn [▶ 330].Busy MC_CamIn [▶ 330].Active	In the initial situation, the command was sent using `Execute := TRUE`, and `Execute := FALSE` is set again. **NC2**: `Busy = TRUE, Active = TRUE` and `InSync = TRUE` are reset by a falling edge at the Execute input. However, the outputs (in particular `InGear`) signal TRUE for at least one cycle as soon as the master and slave axes move in sync with each other. **MC3**: `Busy = TRUE, Active = TRUE` and `InSync = TRUE` remain set until the motion command is replaced by another command, even if `Execute` is already set to `FALSE` (PLCopen-compliant).

5.5 Tc3_Mc3FluidPower

5.5.1 Data Types

5.5.1.1 Structs

5.5.1.1.1 ValveCharacteristicPoint

A single point on the valve characteristic curve mapping velocity to spool position.

Syntax

Definition:

```
TYPE ValveCharacteristicPoint :
STRUCT
    Velocity      : LREAL;
    SpoolPosition : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
Velocity	LREAL	#Invalid	Axis velocity corresponding to the given valve spool position.
SpoolPosition	LREAL	#Invalid	Valve spool position corresponding to the given velocity.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5560 MC3 FluidPower >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.5.1.1.2 ValveCharacterizationOptions

Contains additional options for MC_ValveCharacterization

Syntax

Definition:

```
TYPE ValveCharacterizationOptions :
STRUCT
    WaitTimeAfterVelocityStep   : LREAL;
    WaitTimeAtPositionLimit     : LREAL;
    MinimumTravelPerMeasurement : LREAL;
    Acceleration                : LREAL;
    Deceleration                : LREAL;
    Jerk                        : LREAL;
    OutputLimit                 : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
WaitTimeAfterVelocityStep	LREAL	0.2	Wait time after velocity step ([sec]). Wait time should be large enough for velocity oscillations to settle. Must have value >= 0.0 s. Default: 0.2.
WaitTimeAtPositionLimit	LREAL	0.5	Wait time at a position limit ([sec]). Wait time should be large enough for the axis to come to a complete stop. Must have value >= 0.0 s. Wait time can be extended using the Wait input if it depends on external processes (e.g. switching valves). Default: 0.5.
MinimumTravelPerMeasurement	LREAL	10.0	Minimum position distance per velocity measurement ([PosUnit]). Must have value >= 0.0. Default 10.0.
Acceleration	LREAL	#Default	Maximum acceleration ([PosUnit]/s^2, positive)

Name	Type	Default	Description
Deceleration	LREAL	#Default	Maximum deceleration ([PosUnit]/s^2, positive)
Jerk	LREAL	#Default	Maximum jerk ([PosUnit]/s^3, positive)
OutputLimit	LREAL	1.0	Maximum valve output during valve characterization. Valid values: [0..1]. Valve characterization may exceed this value by up to OutputStepSize value.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5560 MC3 FluidPower >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

5.5.2 Function Blocks

5.5.2.1 Valve Characteristic Curve

5.5.2.1.1 MC_ValveCharacteristicCurve

Wrapper that holds a reference to a valve characteristic curve object and provides methods for setting, accessing and modifying its data.
To use this wrapper, the valve characteristic curve must be mapped and the method Connect() must be called.
Once a reference is successfully stored, it cannot be modified. Subsequent calls to Connect() will return an error.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Connect [▶ 344]	Stores a reference to an existing valve characteristic curve object. The valve characteristic curve must have been previously linked to McToPlc.
ReadData [▶ 344]	Reads all stored data points into the given array. Returns the number of data points read.
OverwriteData [▶ 344]	Overwrites all stored data points with data points from the given array.
ClearData [▶ 345]	Removes all stored data points. The existing reference remains intact.
CopyFrom [▶ 345]	Copies all stored data points from another MC_ValveCharacteristicCurve.
Enable [▶ 346]	Enables valve linearization.
Disable [▶ 346]	Disables valve linearization.
ComputeVelocityValue [▶ 346]	Computes velocity value at the given SpoolPosition.
ComputeSpoolPositionValue [▶ 347]	Computes spool position value at the given Velocity.

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5560 MC3 FluidPower >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Fluid Power

5.5.2.1.1.1 Connect

Stores a reference to an existing valve characteristic curve object. The valve characteristic curve must have been previously linked to McToPlc.

Syntax

Definition:

```
METHOD Connect : UDINT
```

Return value

UDINT

5.5.2.1.1.2 ReadData

Reads all stored data points into the given array. Returns the number of data points read.

Syntax

Definition:

```
METHOD ReadData : UDINT
VAR_INPUT
    DestinationAddress : Pointer To ValveCharacteristicPoint;
    NumberOfPoints     : UDINT;
END_VAR
```

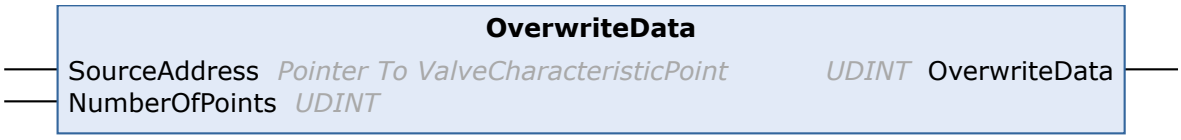
Inputs

Name	Type	Description
DestinationAddress	Pointer To ValveCharacteristicPoint [▶ 341]	Pointer to ARRAY[1..NumberOfPoints] of ValveCharacteristicPoint
NumberOfPoints	UDINT	Capacity of the destination array (maximum number of data points to read).

Return value

UDINT

5.5.2.1.1.3 OverwriteData



Overwrites all stored data points with data points from the given array.

Syntax

Definition:

```
METHOD OverwriteData : UDINT
VAR_INPUT
    SourceAddress  : Pointer To ValveCharacteristicPoint;
    NumberOfPoints : UDINT;
END_VAR
```

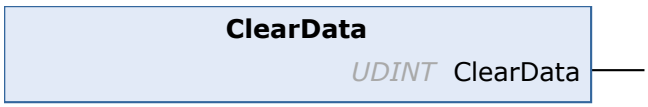
Inputs

Name	Type	Description
SourceAddress	Pointer To ValveCharacteristicPoint [▶ 341]	Pointer to ARRAY[1..NumberOfPoints] of ValveCharacteristicPoint
NumberOfPoints	UDINT	Number of points to be written.

Return value

UDINT

5.5.2.1.1.4 ClearData



Removes all stored data points. The existing reference remains intact.

Syntax

Definition:

```
METHOD ClearData : UDINT
```

Return value

UDINT

5.5.2.1.1.5 CopyFrom

Copies all stored data points from another MC_ValveCharacteristicCurve.

Syntax

Definition:

```
METHOD CopyFrom : UDINT
VAR_INPUT
    Source : Reference To MC_ValveCharacteristicCurve;
END_VAR
```

Inputs

Name	Type	Description
Source	Reference To MC_ValveCharacteristicCurve [▶ 343]	Source MC_ValveCharacteristicCurve to copy from.

Return value

UDINT

5.5.2.1.1.6 Enable

Enables valve linearization.

Syntax

Definition:

```
METHOD Enable : UDINT
```

Return value

UDINT

5.5.2.1.1.7 Disable

Disables valve linearization.

Syntax

Definition:

```
METHOD Disable : UDINT
```

Return value

UDINT

5.5.2.1.1.8 ComputeVelocityValue

Computes velocity value at the given SpoolPosition.

Syntax

Definition:

```
METHOD ComputeVelocityValue : LREAL
VAR_INPUT
    SpoolPosition : LREAL;
END_VAR
```

Inputs

Name	Type	Description
SpoolPosition	LREAL	Spool position at which to evaluate the velocity.

Return value

LREAL

5.5.2.1.1.9 ComputeSpoolPositionValue

Computes spool position value at the given Velocity.

Syntax

Definition:

```
METHOD ComputeSpoolPositionValue : LREAL
VAR_INPUT
    Velocity : LREAL;
END_VAR
```

Inputs

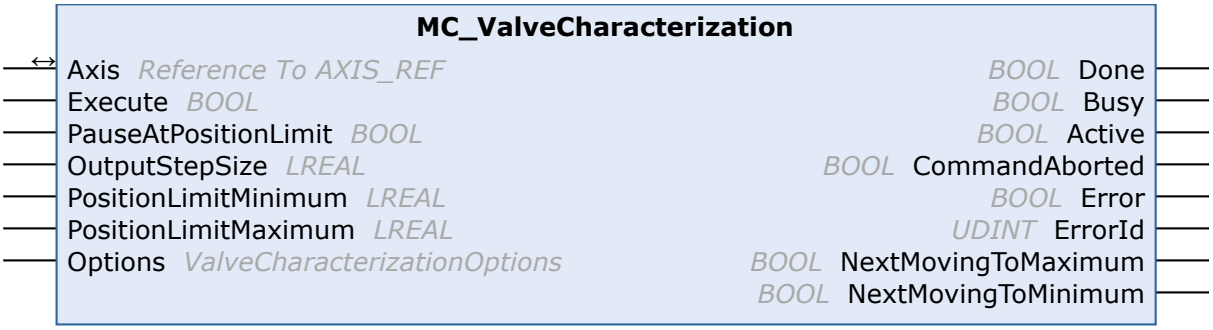
Name	Type	Description
Velocity	LREAL	Velocity at which to evaluate the spool position.

Return value

LREAL

5.5.2.2 Valve Characterization

5.5.2.2.1 MC_ValveCharacterization



This function block characterizes the valve characteristic curve of the axis through a series of velocity measurements.
For each measurement the function block generates an output value for the drive based on the OutputStepSize input and the drive's actuating value range and measures the resulting actual velocity.

Syntax

Definition:

```
FUNCTION_BLOCK MC_ValveCharacterization
VAR_IN_OUT
    Axis                : Reference To AXIS_REF;
END_VAR
VAR_INPUT
    Execute             : BOOL;
    PauseAtPositionLimit : BOOL;
    OutputStepSize      : LREAL;
    PositionLimitMinimum : LREAL;
    PositionLimitMaximum : LREAL;
    Options             : ValveCharacterizationOptions;
END_VAR
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Active              : BOOL;
    CommandAborted      : BOOL;
    Error               : BOOL;
    ErrorId             : UDINT;
    NextMovingToMaximum : BOOL;
    NextMovingToMinimum : BOOL;
END_VAR
```

Inputs

Name	Type	Default	Description
Execute	BOOL		Start the valve characterization process at rising edge.
PauseAtPositionLimit	BOOL	false	The upcoming move is delayed while PauseAtPositionLimit = TRUE.
OutputStepSize	LREAL	#Invalid	Size of output steps. Must have value [0..1], where 1 = output value changes by 100% between two measurements (0 = no change)
PositionLimitMinimum	LREAL	#Invalid	Minimum position ([PosUnit]). Must have value < PositionLimitMaximum.
PositionLimitMaximum	LREAL	#Invalid	Maximum position ([PosUnit]). Must have value > PositionLimitMinimum.
Options	ValveCharacterizationOptions [▶ 342]		Additional function block options

In/Outputs

Name	Type	Description
Axis	Reference To AXIS_REF	Reference to the axis

Outputs

Name	Type	Description
Done	BOOL	Valve characterization finished successfully
Busy	BOOL	Function block is not finished and new output values are to be expected.
Active	BOOL	Function block has active control on the axis.
CommandAborted	BOOL	Command is aborted by another command.
Error	BOOL	Error occurred within function block.
ErrorId	UDINT	Error identifier

Name	Type	Description
NextMovingToMaximum	BOOL	Positive move upcoming
NextMovingToMinimum	BOOL	Negative move upcoming

Version information

- TwinCAT Standard >= v3.1.4026.23.1
- TF5560 MC3 FluidPower >= v4.0.6 (includes TF5500 MC3 Base >= v4.0.6)

Required License

TC3 MC3 Fluid Power

5.6 Sample programs

Sample code and configurations for the individual product areas can be obtained from the corresponding repository on GitHub (see links below).

For example, you have the option to clone the repository or to download a ZIP file containing the sample by clicking on **Download ZIP**.

TF5500 | TwinCAT 3 MC3 Base

The repository demonstrates basic MC3 scenarios such as gearing, PTP movements, and error management in coupled axes. It is suitable for understanding typical motion sequences, state machines, and the practical application of core MC3 modules.

- MC3 Version 4.0.6
- https://github.com/Beckhoff/TF5500_Samples

TF5550 | TwinCAT 3 MC3 Camming

This repository focuses on axis couplings using cam plates, with examples of MC_CamIn, cam plate changes, and various activation strategies during motion. This shows how nonlinear couplings between master and slave axes are configured, tested, and switched in MC3.

- MC3 Version 4.0.6
- https://github.com/Beckhoff/TF5550_Samples

TF5560 | TwinCAT 3 MC3 Fluid Power

The focus here is on FluidPower applications, particularly valve characterization and reading/writing characteristic curves to CSV files. The samples illustrate the workflow for the automatic measurement of valve characteristic curves.

- MC3 Version 4.0.6
- https://github.com/Beckhoff/TF5560_Samples

6 Support and Service

Beckhoff Support and Service provides assistance with questions regarding Beckhoff products and system solutions, as well as with their planning, commissioning, and operation.

Beckhoff subsidiaries and representative offices

Please contact your Beckhoff subsidiary or your representative office for local support and service for Beckhoff products. You can find the addresses of Beckhoff subsidiaries and representative offices at: www.beckhoff.com/worldwide.

For Beckhoff products and technologies, we offer Training programs, tutorials, and webinars.

Beckhoff Support

Beckhoff Support provides assistance with technical questions regarding the use, planning, programming, and commissioning of Beckhoff products.

☎ Hotline:	+49 5246 963-157
✉ Email:	support@beckhoff.com, Support form
Web page:	www.beckhoff.com/support

Beckhoff Service

Our service specialists are here to assist you with any questions you may have regarding after-sales service processes, such as replacement parts, repairs, etc.

☎ Hotline:	+49 5246 963-460
✉ Email:	service@beckhoff.com, Service form
Web page:	www.beckhoff.com/service

Download finder

The Download finder serves as a central source for downloadable documents and files related to Beckhoff products, such as application reports, technical documentation, drawings, and configuration files.

Beckhoff Information System

The Beckhoff Information System is the central online documentation resource for Beckhoff products and contains documentation on products and automation topics, as well as sample code.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl, Germany

☎ Phone:	+49 5246 963-0
✉ Email:	info@beckhoff.com
Web page:	www.beckhoff.com

Trademark statements

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

Third-party trademark statements

Excel, IntelliSense, Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

MAX®, Stratix®, Cyclone®, Altera®, Agilex™, Arria®, Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Sercos is a registered trademark of Sercos International e.V.

More Information:
www.beckhoff.com/TF5500

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