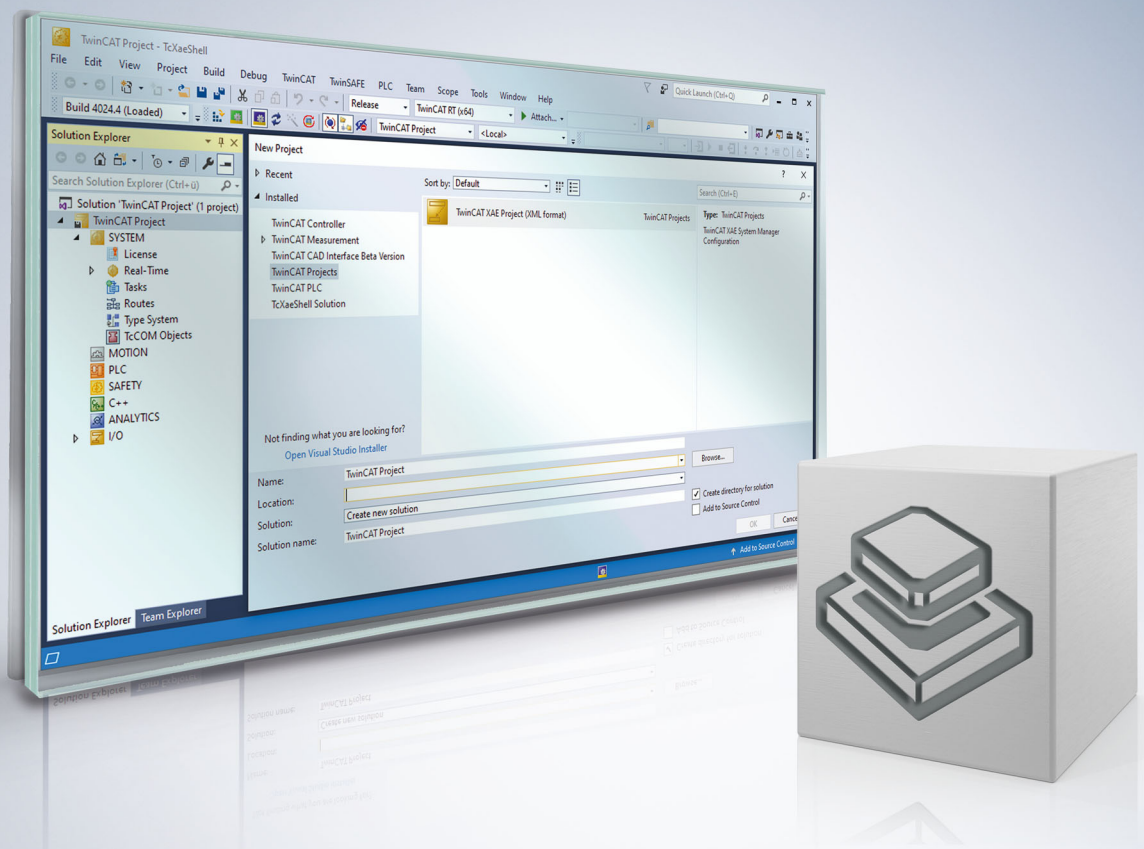
BECKHOFF New Automation Technology

Manual | EN

TF5890

TwinCAT 3 XPlanar



2026-05-20 | Version 1.2.1

Table of contents

Table of contents

1 Documentation notes

1.1 Disclaimer

Beckhoff products are subject to continuous further development. We reserve the right to revise the documentation at any time and without notice. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams, and descriptions in this documentation.

This product contains software.
The relevant legal information and licenses can be found in the internal memory of this product or in the open source software declarations in the download area of the product on the Beckhoff website at

www.beckhoff.com/en-en/products/

You can also send an e-mail to the address below for further information.

osscompliance@beckhoff.com

By using the product, you agree to the relevant licenses for the software included in the product.

1.1.1 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.

The use by third parties of other brand names or trademarks contained in this documentation may lead to an infringement of the rights of the respective trademark owner.

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

1.1.2 Limitation of liability

All components of this product described in the original operating instructions are delivered in a hardware and software configuration, depending on the application requirements. Modifications and changes to the hardware or software configuration that go beyond the documented options are prohibited and nullify the liability of Beckhoff Automation GmbH & Co. KG.

The following is excluded from the liability:

- Failure to comply with this documentation
- Improper use
- Use of untrained personnel
- Use of unauthorized spare parts

1.1.3 Copyright

1.1.4 Third-party brands

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

www.beckhoff.com/trademarks

1.2 Version numbers

On request we can send you a list of revision levels for changes to the documentation. Please send your request to:

support@beckhoff.com

Origin of the document

This documentation was originally written in German. All other languages are derived from the German original.

Product features

The valid product features are always those specified in the current documentation. Further information given on the product pages of the Beckhoff homepage, in emails or in other publications is not authoritative.

1.3 Scope of the documentation

In addition to this documentation, the following documents are part of the complete documentation:

Manual | TF5430

Direct link to the documentation TF5430 | TwinCAT 3 Planar Motion

Operating instructions | XPlanar APS4322

Direct link to the XPlanar APS4322-0000 original operating instructions

Operating instructions | XPlanar APS42xx

Direct link to the XPlanar APS42xx-1x00 original operating instructions

1.4 Safety and instruction

Read the contents that are related to the activities you will perform with the product. Always read the For your safety chapter in the documentation. Observe the warning notes in the chapters so that you can handle the product and work with it properly and safely.

1.4.1 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.

1.5 Explanation of symbols

Various symbols are used for a clear arrangement:

► The triangle indicates instructions that you should execute

• The bullet point indicates an enumeration

[…] The square parentheses indicate cross-references to other text passages in the document

[+] The plus sign in square brackets indicates ordering options and accessories

1.5.1 Pictograms

NOTICE

Notes
Notes are used for important information on the product. The possible consequences of failure to observe these include:

- product malfunctions
- damage to the product
- damage to the environment

Information
This symbol indicates information, tips, and notes for handling the product or the software.

Examples
This symbol shows examples of how to use the product or software.

1.6 Beckhoff Services

Beckhoff and its international partner companies offer comprehensive support and service.

www.beckhoff.com/en-en/support/global-availability/

1.6.1 Support services

The Beckhoff Support offers technical advice on the use of individual Beckhoff products and system planning. The support engineers offer you competent assistance, for comprehension questions as well as for commissioning.

+49 5246 963-157
support@beckhoff.com
www.beckhoff.com/en-en/support/our-support-services/

1.6.2 Training offerings

Training in Germany takes place at the Beckhoff branches or, after consultation, at the customer's premises. Beckhoff offers both face-to-face and online training courses.

+49 5246 963-5000
training@beckhoff.com
www.beckhoff.com/en-en/support/training-offerings/

1.6.3 Service offerings

The Beckhoff service experts support you worldwide in all areas of after-sales service.

+49 5246 963-460
service@beckhoff.com
www.beckhoff.com/en-en/support/our-service-offerings/

1.6.4 Headquarters Germany

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl, Germany

+49 5246 963-0
info@beckhoff.com
www.beckhoff.com/en-en/

A detailed overview of the Beckhoff locations worldwide can be found at:

www.beckhoff.com/en-en/company/global-presence/

1.6.5 Downloadfinder

In the Download finder you will find configuration files, technical documentation and application reports to download.

www.beckhoff.com/documentations

2 Notes on information security

The products from Beckhoff Automation GmbH & Co. KG – if they are reachable 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.

Beckhoff recommends the following protective measures for information security and industrial security:

www.infosys.beckhoff.com

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. Subscribe to the RSS feed to stay informed about information security for Beckhoff products. For more information, see:

www.infosys.beckhoff.com

3 General Information

These operating instructions will help you to commission and program an XPlanar system.

The following chapters will guide you through the basic configuration of the XPlanar hardware components in *TwinCAT 3*, so that you can move the XPlanar Movers via the basic MC3 UI interface and program them in the PLC. The aim is to move the XPlanar Mover for the first time and to verify the system concept. You will also receive an initial overview of all XPlanar functionalities of the *TF5890 | TwinCAT 3 XPlanar*.

The *TF5890 | TwinCAT 3 XPlanar* documentation is currently still under construction and will be continuously expanded. The current version of these operating instructions can be found at:

www.beckhoff.com/en-en/support/download-finder/

3.1 Compatibility

This documentation is not suitable for commissioning XPlanar systems with older software versions. If you have any questions, please contact the product specialist responsible for your region.

Some of the functions described are only available from *TwinCAT 3 | Build 4026*.

This documentation refers to the following software versions:

Software	Version
TF5890 | TwinCAT 3 XPlanar	3.20.700.0 or higher
TF5430 | TwinCAT 3 Planar Motion	3.1.6.57 or higher
TwinCAT 3	3.1.4024.0 or higher

3.2 Presentation of the screenshots

The language setting English in TwinCAT was used for this documentation. All screenshots in this documentation are shown as examples in English. The terms from the screenshots are used in the corresponding instructions.

All functions and settings can be found in the same place in the software, regardless of the language set on your PC.

4 Installation and software updates

Software version
It cannot be guaranteed that all the functions explained here are implemented in TwinCAT 3 | 4024. The development of the *TF5890 | TwinCAT 3 XPlanar* for *TwinCAT 3 | 4024* was completed with version **4.2.xxx**. If you want to use all functions, install *TwinCAT 3 | Build 4026* and the latest version of *TF5890 | TwinCAT 3 XPlanar*.

For the installation of the *TF5890 | TwinCAT 3 XPlanar*, it is crucial which version of TwinCAT 3 you want to use or which version is already installed on your PC. You have the option of using *TwinCAT 3 Build 4024* or *TwinCAT 3 Build 4026*.

Please note the different procedures for installing the software:

TwinCAT 3 | Build 4026
Further information can be found in chapter "TwinCAT 3 | Build 4026", [Page 15].

TwinCAT 3 | Build 4024
Further information can be found in chapter "TwinCAT 3 | 4024", [Page 17].

4.1 TwinCAT 3 | Build 4026

To install the workload *TF5890 | TwinCAT 3 XPlanar* you need:

- TwinCAT Package Manager
- TwinCAT 3 | Build 4026

You must also install the latest version of the *TF5430 | Planar Motion* workload.

All information on installation can be found in the Beckhoff Information System under *TwinCAT 3 > Installation*

https://infosys.beckhoff.com

4.1.1 Additional licensing requirements

TF5890 | TwinCAT 3 XPlanar requires the TC1250 license.

4.1.2 Installing the TwinCAT Package Manager

Further information on installing the *TwinCAT Package Manager* can be found in the Beckhoff Information System:

Direct link Infosys - Installing the TwinCAT Package Manager

4.1.3 Installing TwinCAT 3 | Build 4026

Further information on the installation of *TwinCAT 3 | Build 4026* can be found in the Beckhoff Information System:

Direct link Infosys - Installing TwinCAT 3 | Build 4026

4.1.4 Installing workloads

Further information on installing the *TF5890* | *TwinCAT 3 XPlanar* and *TF5430* | *Planar Motion* workloads can be found in the Beckhoff Information System:

Direct link Infosys - Installing workloads

The order in which the workloads are installed is arbitrary.

4.2 TwinCAT 3 | 4024

Software version
It cannot be guaranteed that all the functions explained here are implemented in TwinCAT 3 | 4024. The development of the *TF5890 | TwinCAT 3 XPlanar* for *TwinCAT 3 | 4024* was completed with version **4.2.xxx**. If you want to use all functions, install *TwinCAT 3 | Build 4026* and the latest version of *TF5890 | TwinCAT 3 XPlanar*.

Before you can commission an XPlanar system, you must first install the *TF5890 | TwinCAT 3 XPlanar* software. The following steps are also required when updating to the latest version of *TF5890 | TwinCAT 3 XPlanar*.

If you have already installed a software version, check if you have installed the latest version. Further information can be found in chapter "Check version", [Page 18].

As a prerequisite for the installation, a version of TwinCAT 3 XAE (TcXaeShell) or a compatible Microsoft Visual Studio that is integrated with TwinCAT 3 XAE must be available on your system.

You can find the latest version of the software in the download finder:

www.beckhoff.com/de-de/support/downloadfinder/

In addition to the *TF5890 | TwinCAT 3 XPlanar* software, the *TF5430 | TwinCAT 3 Planar Motion* software must also be installed on your system. Further information can be found in the *TF5430 TwinCAT 3 | Planar Motion* documentation:

Direct link to the TF5430 | TwinCAT 3 Planar Motion documentation

Whether you first install the *TF5430 | TwinCAT 3 Planar Motion* or the *TF5890 | TwinCAT 3 XPlanar* software is not relevant.

NOTICE

Run as administrator
Run the installation as administrator on your system to avoid problems during installation.

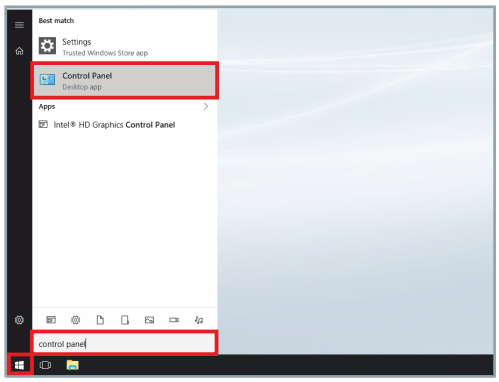
► Download and save the **TF5890 | TwinCAT 3 XPlanar** software

4.2.1 Additional licensing requirements

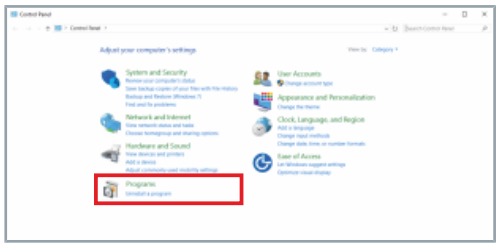
TF5890 | TwinCAT 3 XPlanar requires the *TC1250* license.

4.2.2 Check version

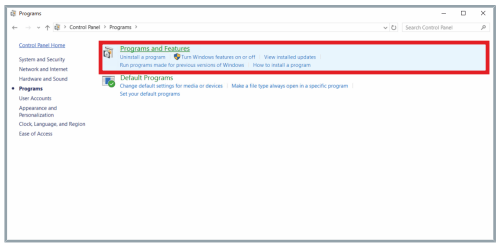
Observe operating system
Calling up the version depends on the operating system installed on your device and may differ from the variant described here.



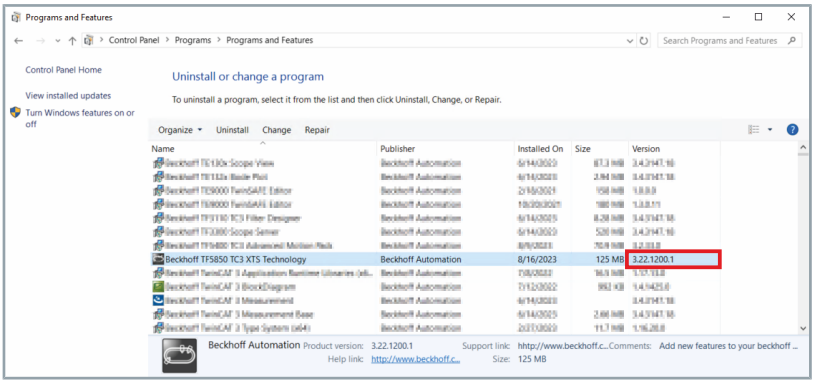
► Click **Start**
► Call up the Control Panel



The window *Control Panel* opens.
► Click on **Programs**



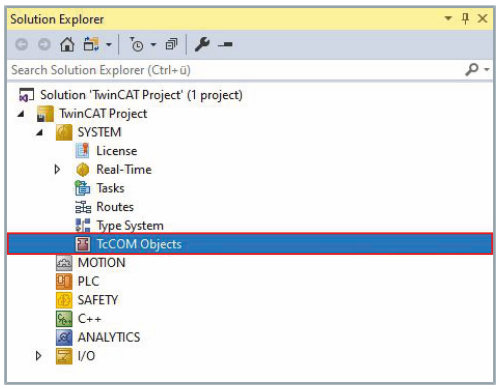
The window *Programs* opens.
► Click on **Programs and Features**



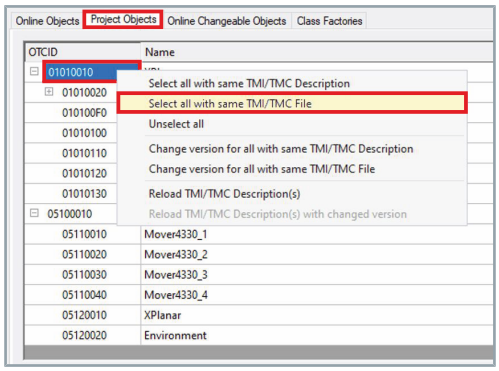
The *Programs and Features* window opens.
► Check the version of Beckhoff **TF5890 TC3 XPlanar Technology**

4.3 Reload TcCOM objects

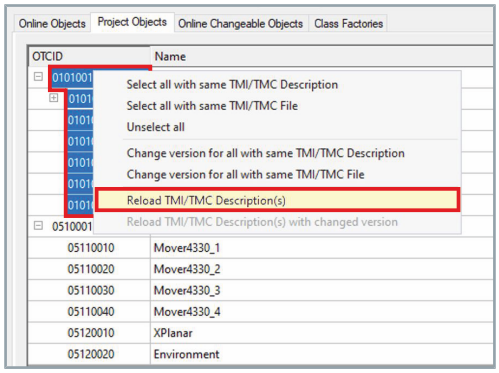
If you have updated the *TF5890 | TwinCAT 3 XPlanar*, you must reload the TcCOM objects.



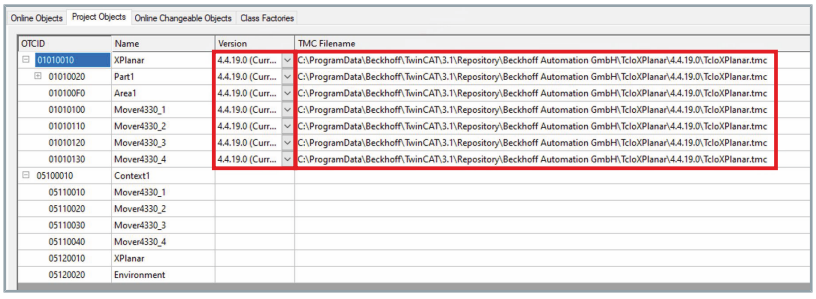
- ► Expand *Solution Explorer* > *TwinCAT Project* > *SYSTEM*
- ► Double click on **TcCOM Objects**



- ► Click on the **Project Objects** tab in the project window
- ► Select an object
- ► Right-click the selected object to open the context menu
- ► Click on **Select all with same TMI/TMC file** in the context menu



- ► Right-click the selected objects to open the context menu
- ► Click on **Reload TMI/TMC Description(s)** in the context menu



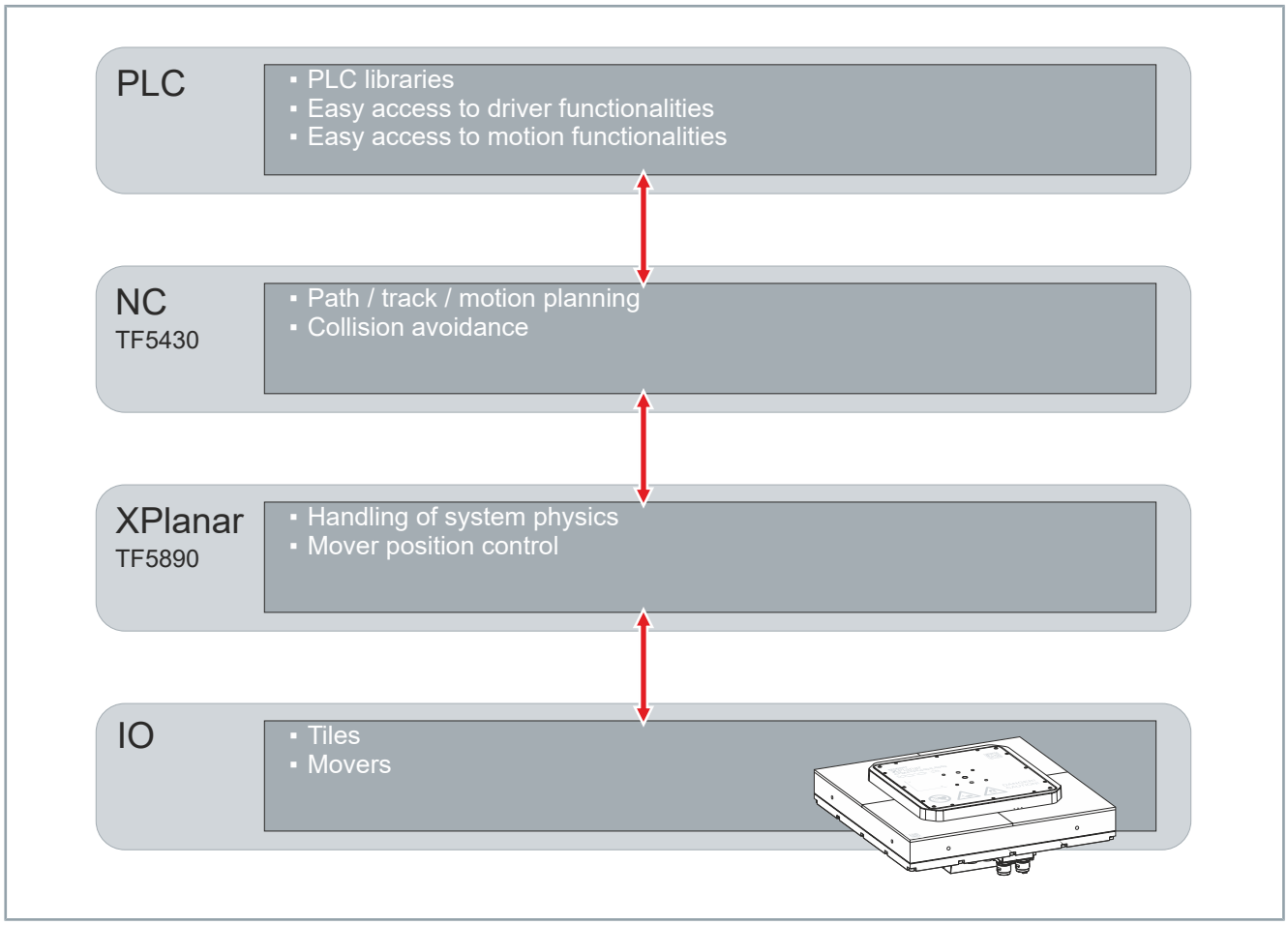
After the TMI/TMC descriptions have been reloaded, you must check that all XPlanar objects are using the correct version of the driver and the correct TMC file.

4.4 High level XPlanar components

The XPlanar system consists of:

- Tiles
  - *APS4322-0000* or *APS42xx-1x00*. The tiles contain all the necessary power electronics, measurement electronics and communication electronics
- Movers
  - *APM4220-0000, APM4221-0000, APM4330-000x* and *APM4550-0000*
- Scalable IPC

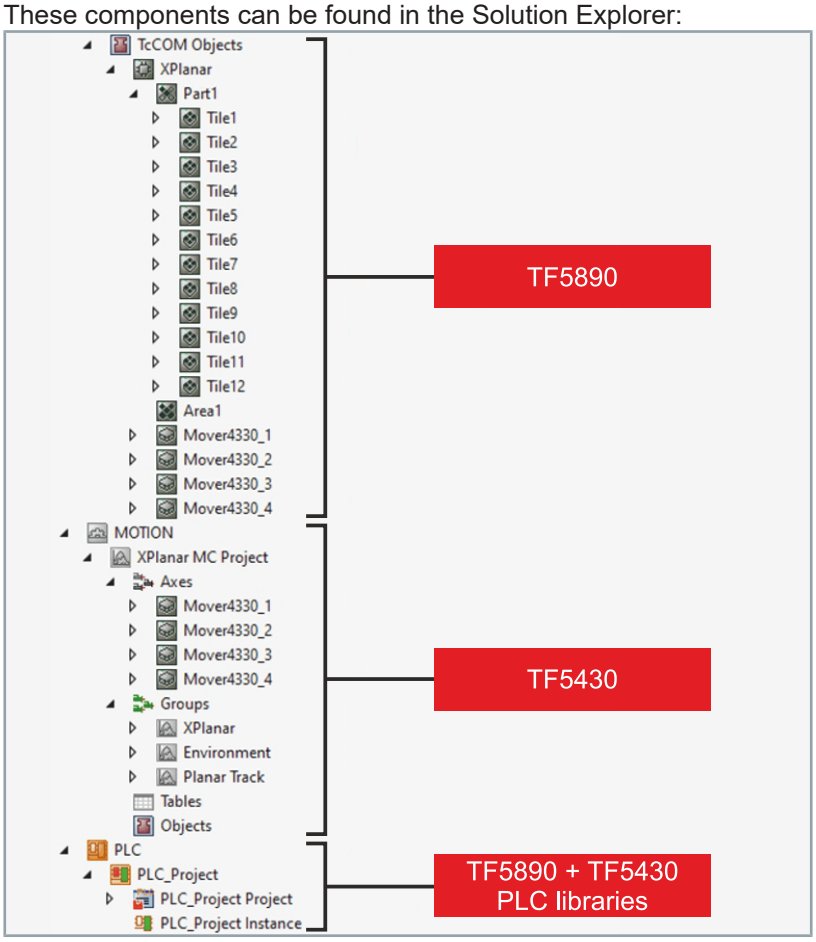
The *TF5890* | *TwinCAT 3 XPlanar* software enables the creation of corresponding objects in the TwinCAT configuration tree, which act as interfaces between the hardware components and the automation software.



The XPlanar software requires coordination between several components:

- TcCOM objects
  - *TF5890* | *TwinCAT 3 XPlanar*
  - *TF5430* | *TwinCAT 3 Planar Motion*
- PLC
  - Planar Motion and Physics libraries, which are provided by the *TF5430*, allowing access to high-level motion commands of the XPlanar system
  - The XPlanar Utility library, which is provided by the *TF5890*, and allows access to the driver objects and their parameters

These components can be found in the Solution Explorer:



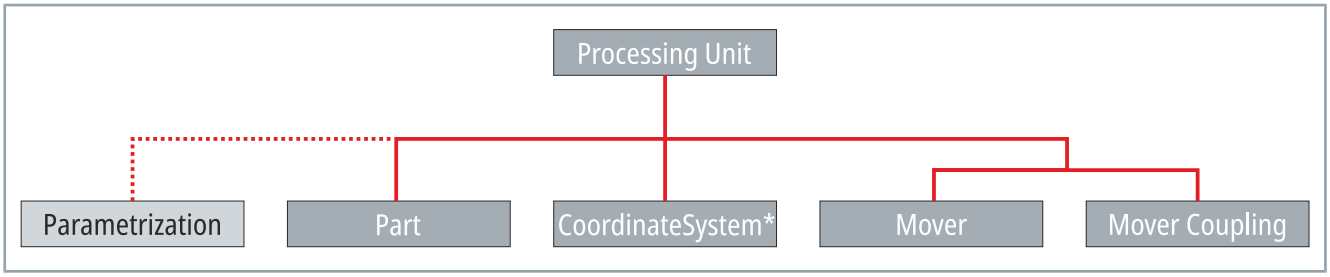
5 XPlanar Configuration Procedure

Before you can use an XPlanar system, it must first be fully configured with all components and all *Real Time* and *Distributed Clock* settings.

The main steps in configuring an XPlanar system are as follows:

5.1 XPlanar Processing Units

The *XPlanar Processing Unit* is the center of the XPlanar system. All the necessary objects converge in the *XPlanar Processing Unit* and are logically linked to each other. An XPlanar system can contain several *XPlanar Processing Units* that act as independent machines.



* The coordinate system replaces the MovementArea.

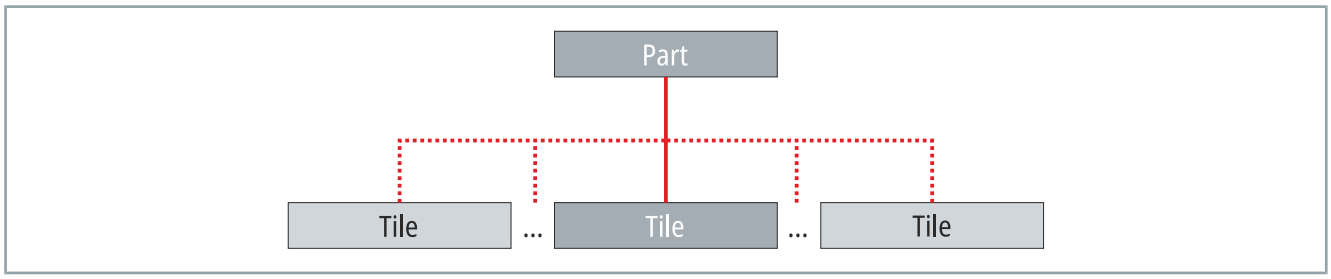
Tiles, movers or mover couplings, parts and coordinate systems always belong to a *processing unit*. Parametrization sets can be added as required.

You can create one or more *XPlanar Processing Units*, depending on the number of individual XPlanar systems used in the machine. An *XPlanar Processing Unit* organizes various XPlanar objects that need to be created for tiles and movers.

For more information about the *XPlanar Tasks* required for your XPlanar system, contact the product specialist responsible for your region.

5.2 XPlanar Parts

An *XPlanarPart* is the minimum required component of an XPlanar system. An *XPlanarPart* consists of one or more subordinate *XPlanarTiles*.



If several parts are used, an *XPlanarCoordinateSystem* must be added.

5.3 XPlanar Tiles

An *XPlanar Tile* is the smallest possible *XPlanar Part.* An *XPlanar Tile* corresponds to a physical XPlanar Tile *APS4xxx-xx00*. An *XPlanar Tile* is always a subordinate object of an *XPlanar Part*.

Simulation mode or normal operation
No hardware is required for the simulation mode of the driver.
For normal operation, a physical *APS4xxx* XPlanar Tile must be assigned to each *XPlanar Tile*.

A task must be assigned to each *XPlanar Tile* in the Context menu in order to be able to perform active calculations. All *XPlanar Tiles* that belong to an EtherCAT master must run on the same task. Only *XPlanar Tiles* have EtherCAT communication interfaces.

An *XPlanar Tile* can be arranged in a fixed grid:

- APS42xx-1x00: 40 mm grid
- APS4322-0000: 48 mm grid

Further information on this can be found in the XPlanar original operating instructions:

Direct link to the XPlanar APS4322-0000 original operating instructions

Direct link to the XPlanar APS42xx-1x00 original operating instructions

The functionality of some driver features is not available for offset tiles.

5.4 XPlanar Movers

Together with the tiles, the *XPlanar Movers* are the main components of the XPlanar system. The *XPlanar Movers* move on the tiles.

An *XPlanar Mover* TcCOM object corresponds to a physical mover.

Simulation mode or normal operation
No hardware is required for the simulation mode of the driver.
For normal operation, a physical APM4xx0 XPlanar Mover must be assigned to each *XPlanar Mover*.

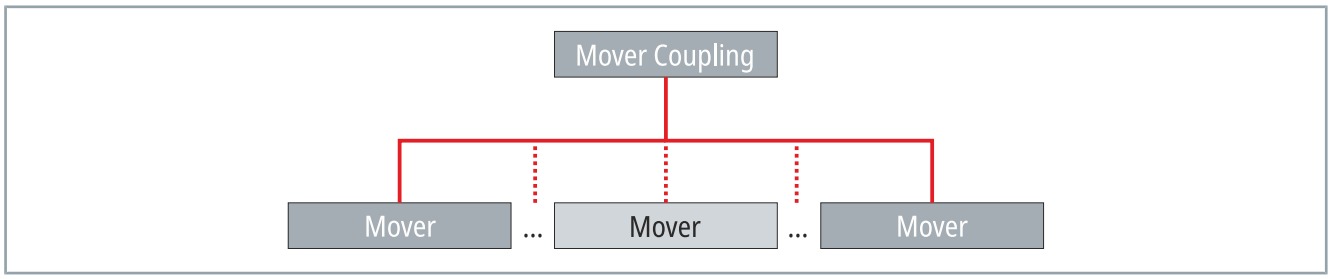
A task must be assigned to each *XPlanar Mover* in order to be able to perform active calculations.

Depending on the configuration and task, up to two movers can be assigned to one CPU core, which calculates the current position and the set position.

The *XPlanar Movers* have an interface to external position setpoints. Movement commands are sent to the *XPlanar Movers* via the interface.

5.5 XPlanar Mover Coupling

An *XPlanarRigidMoverCoupling* object realizes the physical, rigid coupling of several XPlanar Movers. At least two movers are required for an *XPlanarRigidMoverCoupling* object. Two movers of a coupling can share one core. When using *APS4322-0000* tiles, Beckhoff recommends arranging the submovers at a distance of 240 mm from center to center. The movers used are regarded by the *TF5430 | TwinCAT 3 Planar Motion* software and the PLC as one mover object *XPlanarRigidMoverCoupling*. The configured setpoint origin is used to control the mover coupling. *Further information on the mover couplings can be found in chapter "Mover coupling", [Page 68].



5.6 XPlanar Coordinate System

Coordinate system replaces the MovementArea
The XPlanarCoordinateSystem replaces the previous XPlanarMovementArea.

A part can be positioned at one or more positions in one or more coordinate systems.

As soon as the driver is started, each part registers with its defined position in the corresponding coordinate system. MovementAreas with all possible arrangements of the parts are then automatically generated for each coordinate system.

If no coordinate system position can be created for the part, the part uses its own MovementArea. Therefore, no coordinate system object needs to be created if only one part is used or if no connection between different parts is required.

If a part is never to be removed from a coordinate system, the *IsFixed* parameter must be set to TRUE. This procedure reduces the start-up time and requires less storage capacity.

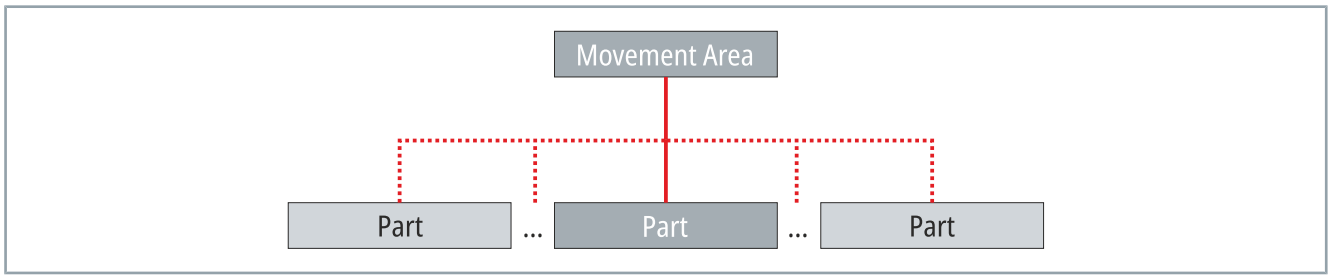
The *ActiveIndex* parameter indicates the position at which the part is assigned in the XPlanarCoordinateSystem:

- The Part *ActiveIndex* is set to 0 by default. This means that the part uses its own MovementArea.
- If the Part *ActiveIndex* >0, the MovementArea is selected from the list. The corresponding coordinate system starts with index 1.

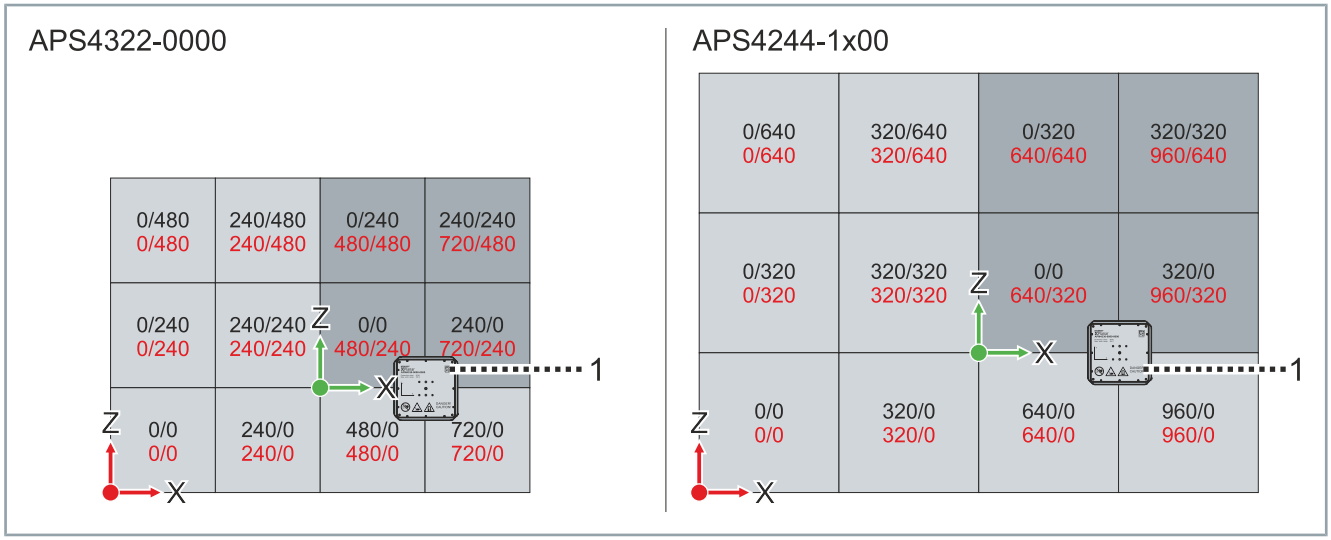
5.7 XPlanarMovementArea

XPlanarCoordinateSystem will replace XPlanarMovementArea
The *XPlanarMovementArea* is gradually being replaced by the *XPlanarCoordinateSystem*.

An *XPlanarMovementArea* consists of one or more *XPlanarParts*. Each *XPlanarPart* has its origin relative to the origin of the *XPlanarMovementArea*.



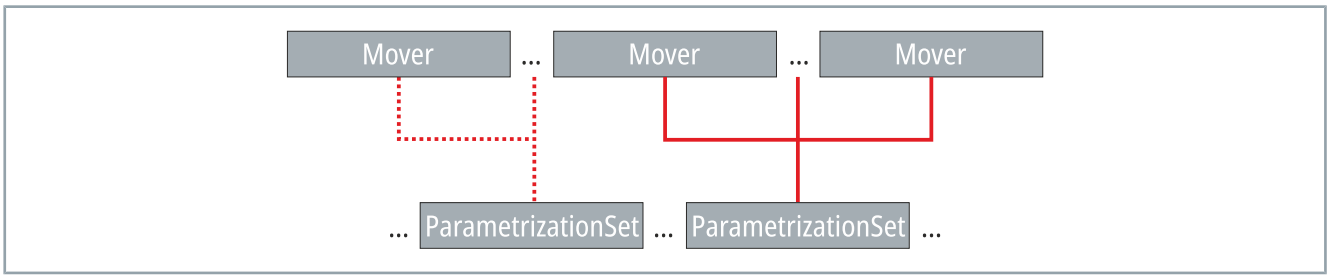
An *XPlanarPart* has its coordinate origin relative to the coordinate origin of the *XPlanarMovement Area*. The following graphic shows an *XPlanarMovementArea* consisting of two parts with the corresponding coordinates of the *XPlanarParts* and in relation to the position in the *XPlanarMovementArea*:



Position	Explanation
1	Mover
	XPlanar Part 1
	XPlanar Part 2
+	XPlanar Movement Area
	Coordinate origin • Part 1 • Movement Area
	Coordinate origin part 2

5.8 XPlanar Mover parametrization sets

The movers and mover couplings can access *XPlanar Mover parametrization sets* if the default parametrizations available in the software are not to be used. General parameters as well as specific observer parameters, control parameters, filter parameters and inertia parameters can be set in the parametrizations.



One or more movers and mover couplings can access an *XPlanar Mover parametrization set* simultaneously. However, a mover can only access one parametrization set at a time. A dynamic change of the parametrization set is possible during operation.

If changes are made to a parametrization set, these changes are applied to all movers and mover couplings that use this parametrization set.

6 Add TcCOM objects

Manual creation or use of TwinCAT Tools
You have the option of creating the hardware components and the TcCOM objects manually or quickly and easily via the TwinCAT tool XPlanar Configurator. Further information can be found in chapter .

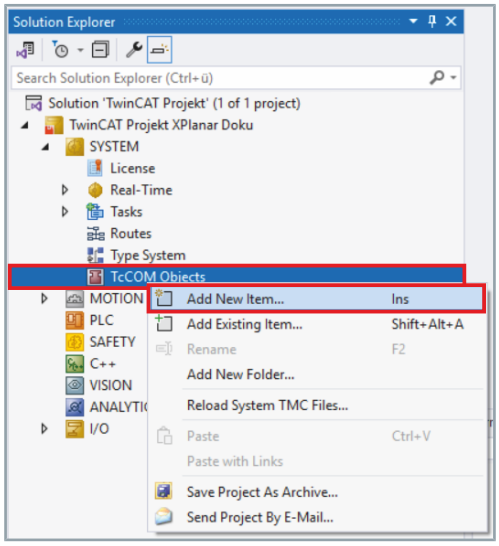
This chapter describes how to create hardware components and TcCOM objects manually and without the help of XPlanar Tools

The TcCOM objects must be added in the following order for a manual configuration:

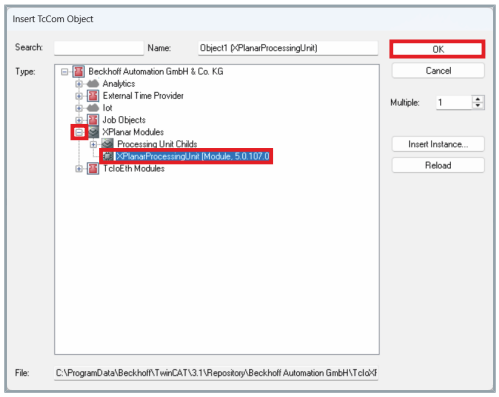
- Processing Unit
- Part
- Tile
- Movement Area | CoordinateSystem
- Mover | Mover coupling
- Mover parametrization

The creation of the TcCOM objects is shown using an example configuration with an XPlanar system consisting of 2 x 3 tiles and four APS4330-0000 movers. The figures show the corresponding settings for this example configuration.

6.1 Processing Unit



- ► Expand *Solution-Explorer* > *System*
- ► Right-click on **TcCOM Objects** to open the context menu
- ► Click **Add New Item** in the context menu

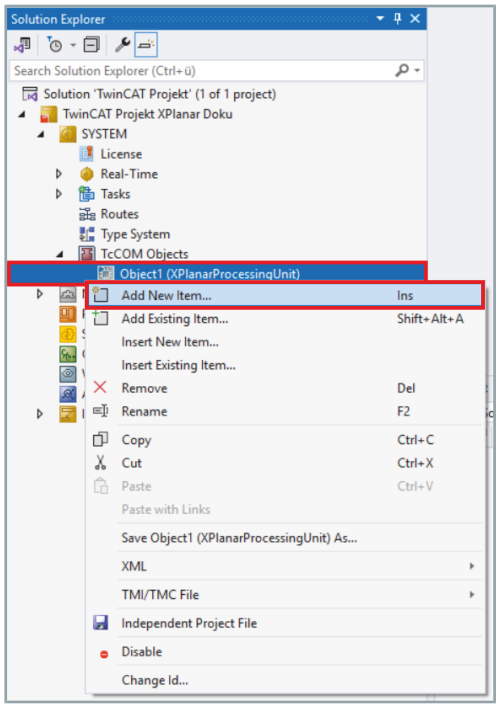


The *Insert TcCOM Object* dialog box opens.

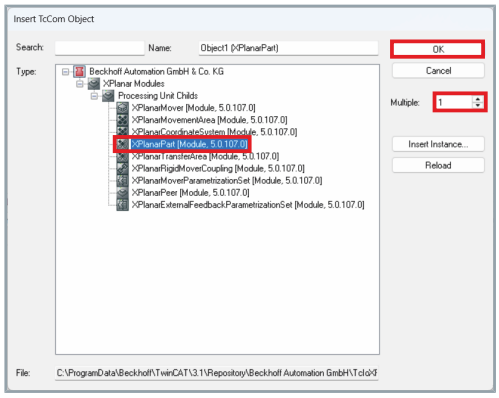
- ► Click on the **+** button of the *XPlanar Modules* to see the available TcCOM objects
- ► Select **XPlanarProcessingUnit [Module]**
- ► Double-click on the selection or confirm with **OK**

A Processing Unit is added to the TcCOM objects.

6.2 Part



- ► Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ► Right-click on **Object (XPlanarProcessingUnit)** to open the context menu
- ► Click **Add New Item** in the context menu

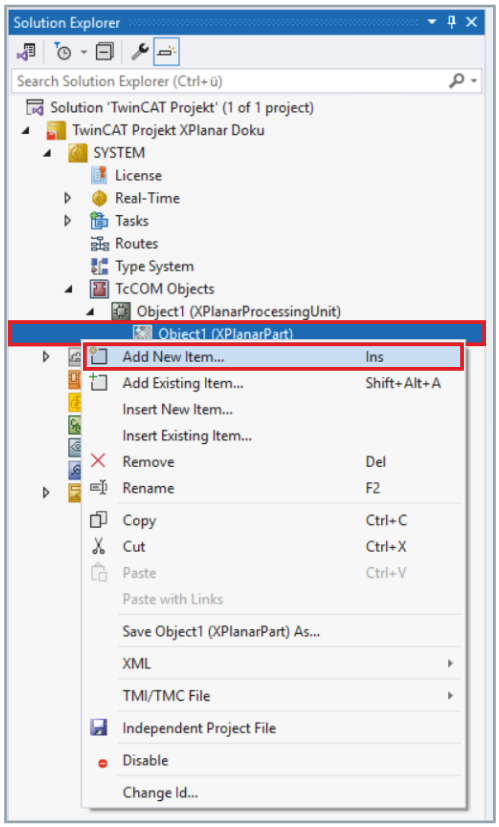


The *Insert TcCOM Object* dialog box opens.

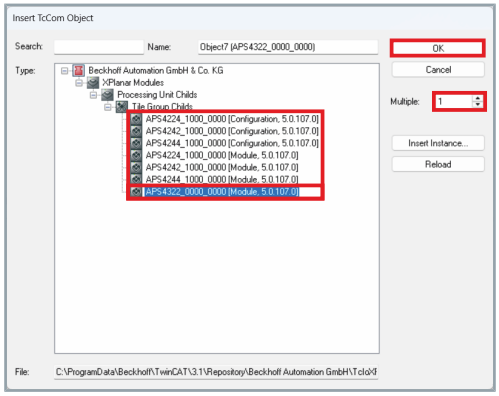
- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- ► Select **XPlanarPart [Module]**
- ► Enter the number of required Parts
- ► Double-click on the selection or confirm with **OK**

One or more parts are added to the Processing Unit.

6.3 Tile



- Expand *Solution Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)*
- Right-click on **Object (XPlanarPart)** to open the context menu
- Click **Add New Item** in the context menu



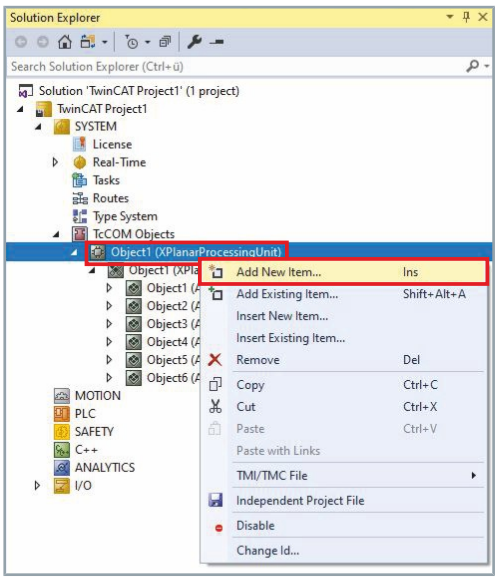
The *Insert TcCOM Object* dialog box opens.

- Click on the **+** button of the *XPlanar Modules*
- Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- Select **APS4xxx-x000-0000**
- Enter the number of tiles required
- Double-click on the selection or confirm with **OK**

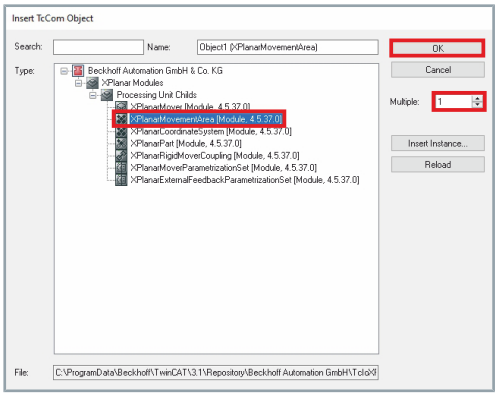
One or more tiles are added to the XPlanar Part.

6.4 Movement Area

Coordinate system instead of MovementArea
The MovementArea will be replaced by the coordinate system in the long term. Beckhoff recommends not creating new MovementAreas and using coordinate systems instead. Further information can be found in chapter "Coordinate system", [Page 33].



- ► Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ► Right-click on **Object (XPlanarProcessingUnit)** to open the context menu
- ► Click **Add New Item** in the context menu

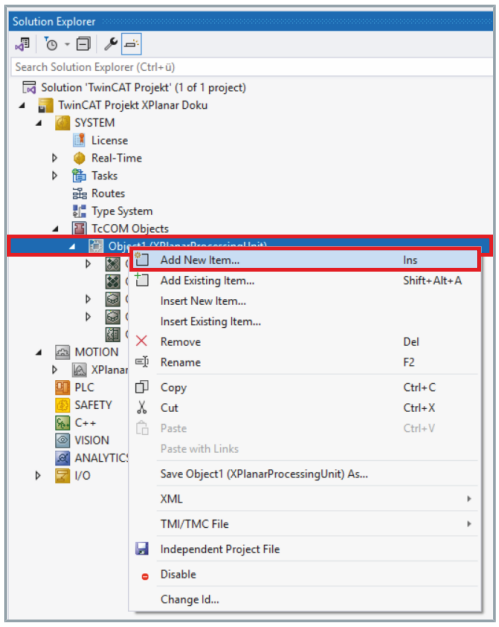


The *Insert TcCOM Object* dialog box opens.

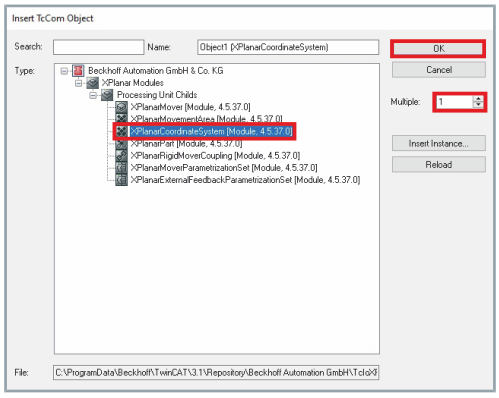
- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- ► Select **XPlanarMovementArea [Module]**
- ► Enter the number of movement areas required
- ► Double-click on the selection or confirm with **OK**

One or more movement areas are added to the Processing Unit.

6.5 Coordinate system



- ► Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ► Right-click on **Object (XPlanarProcessingUnit)** to open the context menu
- ► Click **Add New Item** in the context menu

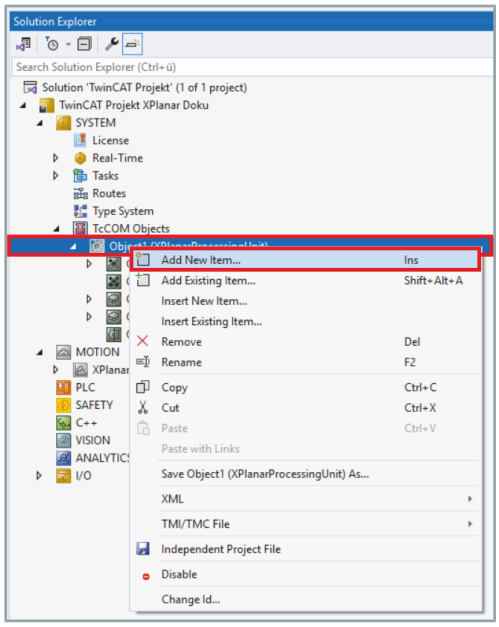


The *Insert TcCOM Object* dialog box opens.

- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- ► Select **XPlanarCoordinateSystem**
- ► Enter the number of coordinate systems required
- ► Double-click on the selection or confirm with **OK**

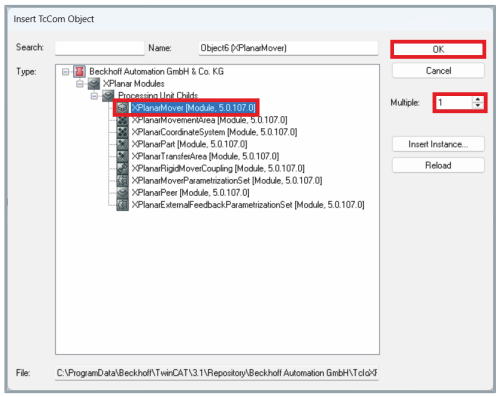
One or more coordinate systems are added to the Processing Unit.

6.6 Mover



- ► Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ► Right-click on **Object (XPlanarProcessingUnit)** to open the context menu
- ► Click **Add New Item** in the context menu

Determination of the mover type
A universal mover object is added to the TcCOM objects at the beginning. The mover type is defined later in the Parameters (Init) tab. You can find more information on defining the mover type in chapter "Mover", [Page 39].



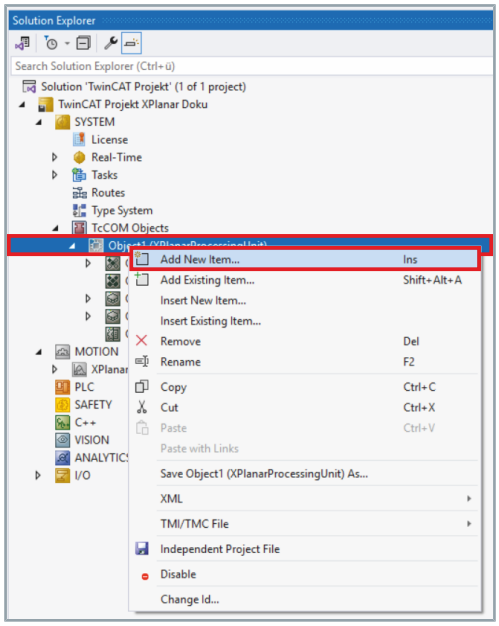
The *Insert TcCOM Object* dialog box opens.

- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- ► Select **XPlanarMover [Module]**
- ► Enter the number of movers required
- ► Double-click on the selection or confirm with **OK**

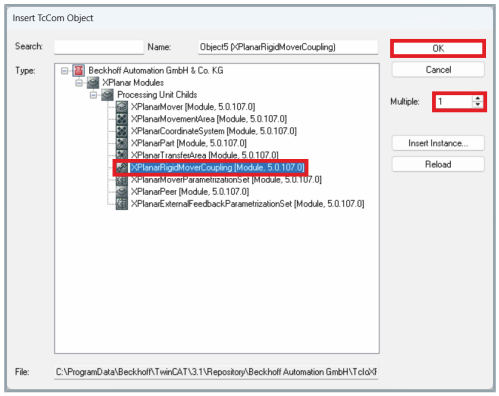
One or more movers are added to the Processing Unit.

6.7 Mover coupling

Add mover couplings only if required
You only need to create mover couplings if you want to use mover couplings on your system. Further information on the mover couplings can be found in chapter "Mover coupling", [Page 68].



- ► Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ► Right-click on **Object (XPlanarProcessingUnit)** to open the context menu
- ► Click **Add New Item** in the context menu



The *Insert TcCOM Object* dialog box opens.

- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- ► Select **XPlanarRigidMoverCoupling [Module]**
- ► Enter the number of mover couplings required
- ► Double-click on the selection or confirm with **OK**

One or more mover couplings are added to the Processing Unit.

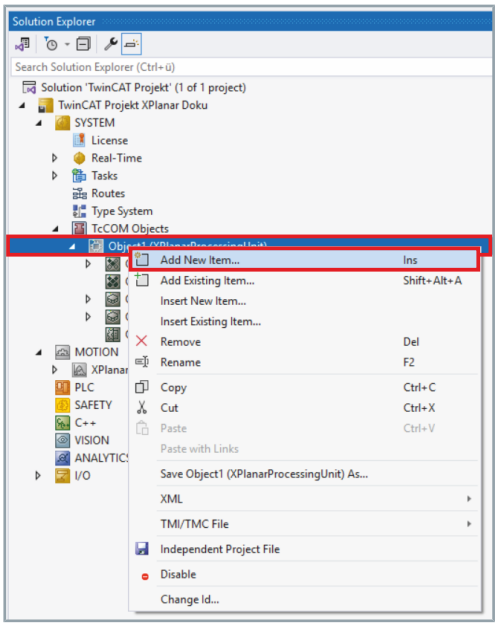
6.8 Mover parametrization

Add parametrizations only if required
You only need to create parametrizations and parametrization sets if you do not want to use the pre-set default parameters of the movers. You can find a list of the parametrizations available in chapter Parameterization sets

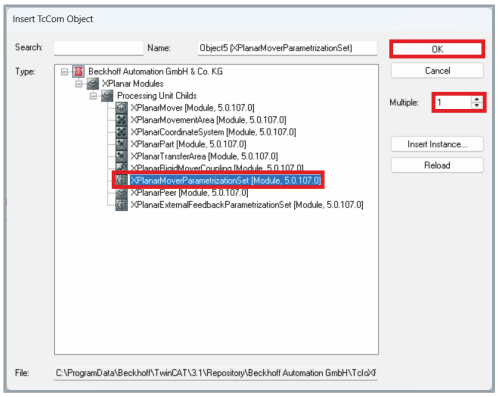
In addition to the hardware components, you can also add one or more parametrization sets to the *Processing Unit*.

In a parametrization set, you can change and assign the parameters for individual or multiple movers on the system without the other movers losing their assigned parameters. It is also possible to switch between different parametrization sets during operation. Different parametrizations can be combined in one parametrization set.

Parametrizations only in parametrization sets
To be able to use parametrizations, they must be created in a parametrization set. An individual parametrization must also be assigned to a parametrization set.



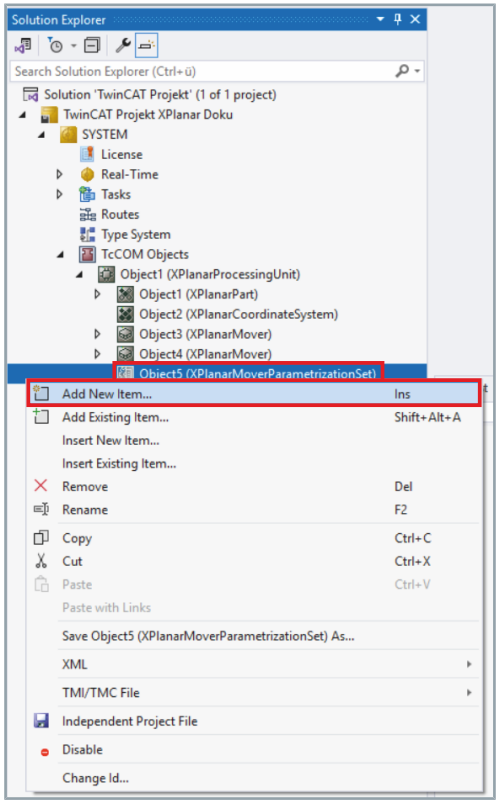
- ► Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ► Right-click on **Object (XPlanarProcessingUnit)** to open the context menu
- ► Click **Add New Item** in the context menu



The *Insert TcCOM Object* dialog box opens.

- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs* to see the available TcCOM objects
- ► Select **XPlanarMoverParametrizationSet [Module]**
- ► Enter the number of parametrization sets required
- ► Double-click on the selection or confirm with **OK**

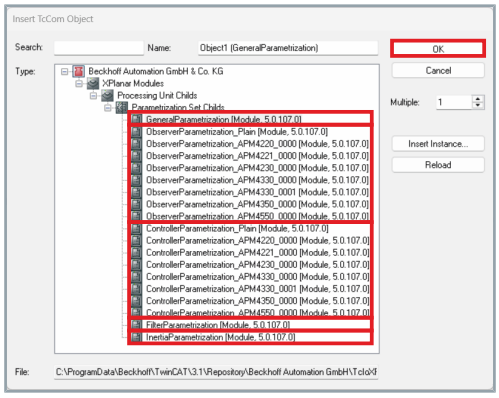
One or more parametrization sets are added to the Processing Unit.



- ► Expand *Solution-Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)*
- ► Right-click on **Object (XPlanarMoverParametrizationSet)** to open the context menu
- ► Click **Add New Item** in the context menu

NOTICE

Each parametrization type can only be contained once in a parametrization set.
It is not possible to use a *ControllerParametrization_Plain* and a *ControllerParametrization_APM4330* in a parametrization set, for example, as both parametrizations are of the same type.



The *Insert TcCOM Object* dialog box opens.

- ► Click on the **+** button of the *XPlanar Modules*
- ► Click on the **+** button of the *Processing Unit Childs*
- ► Click on the **+** button of the *Parametrization Set Childs* to see the available parametrizations

Different parametrizations are available:

- GeneralParametrization
- ObserverParametrization_xxxx
- ControllerParametrization_xxxx
- FilterParametrization
- InertiaParametrization

- ► Select parametrization
- ► Double-click on the selection or confirm with **OK**

The selected parametrization is added to the parametrization set.

- ► Add further parametrizations to the parametrization set in the same way

7 TcCOM objects basic settings

To be able to start the XPlanar system, some parameter settings must be made for the tiles, movers and MovementAreas.

For the parameter settings of the tiles and movers, you need the *BTN | Beckhoff Traceability Number* of the corresponding hardware components, which is part of the *BIC | Beckhoff Identification Code*. Further information can be found in the *Product overview* chapter in the operating instructions and on the Beckhoff website:

XPlanar | Operating instructions
Description of the mechanical and electrical parameters as well as all necessary information for the assembly of the XPlanar system.

Direct link to the XPlanar documentation | Operating instructions

XPlanar | APS42xx-1x00
Description of the mechanical and electrical parameters as well as all necessary information for the assembly of the XPlanar system.

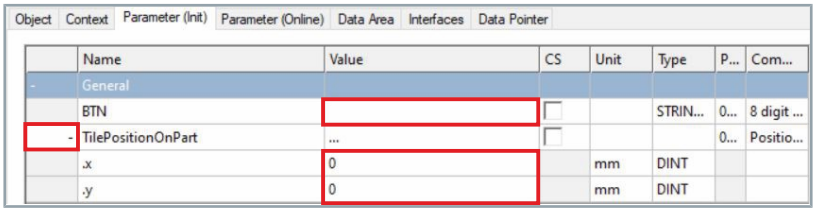
Direct link to the XPlanar | APS42xx-1x00 documentation

BIC | Beckhoff Identification Code

Direct link to further information on the BIC | Beckhoff Identification Code

7.1 Tiles

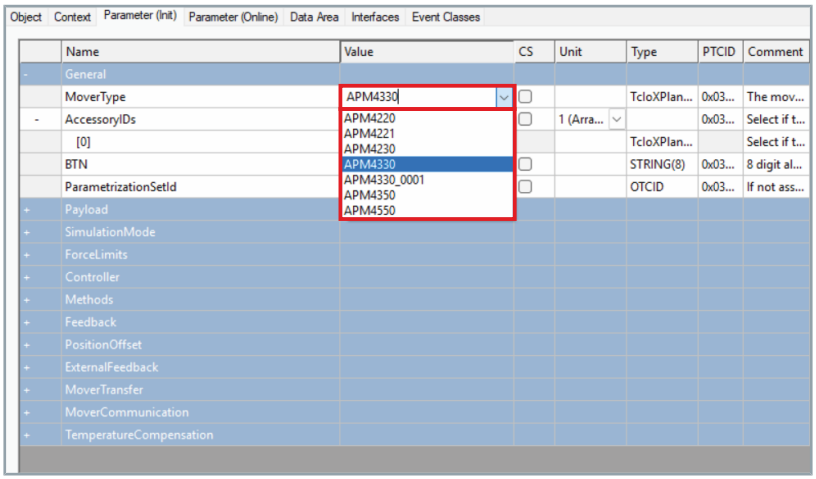
- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (XPlanarPart)*
- ► Double-click on **Object (APS4322)**
- ► Click the *Parameter (Init)* tab in the project window



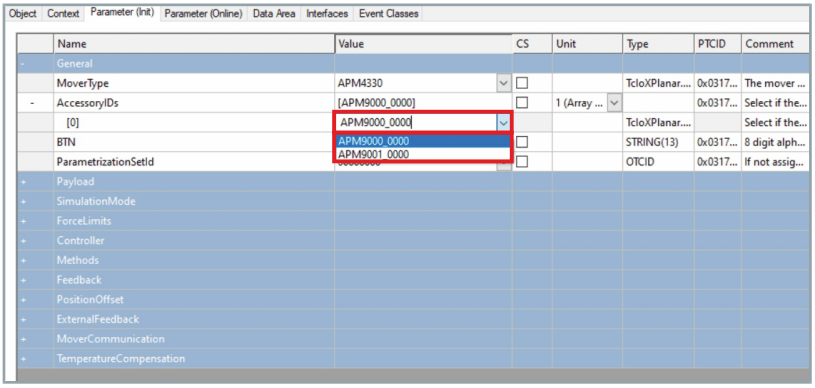
- ► Enter **BTN** of the tile
- ► Click on **+** of *TilePositionOnPart* to call up the input fields for the X-position and Y-position of the tile
- ► Enter values for **x** and **y** that correspond to the position of the tile in the XPlanar Part

7.2 Mover

- ► Expand *Solution Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit)*
- ► Double-click on **Object (XPlanarMover)**
- ► Click the *Parameter (Init)* tab in the project window



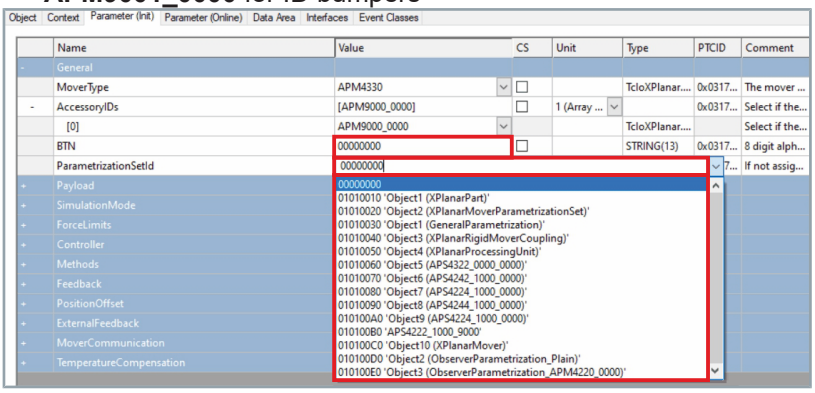
- ► Select **Movertype** from the drop-down menu



► Expand *AccessoryIDs*

► Select the bumper type in drop-down menu *[0]*:

- **APM9000_0000** for passive bumpers
- **APM9001_0000** for ID bumpers



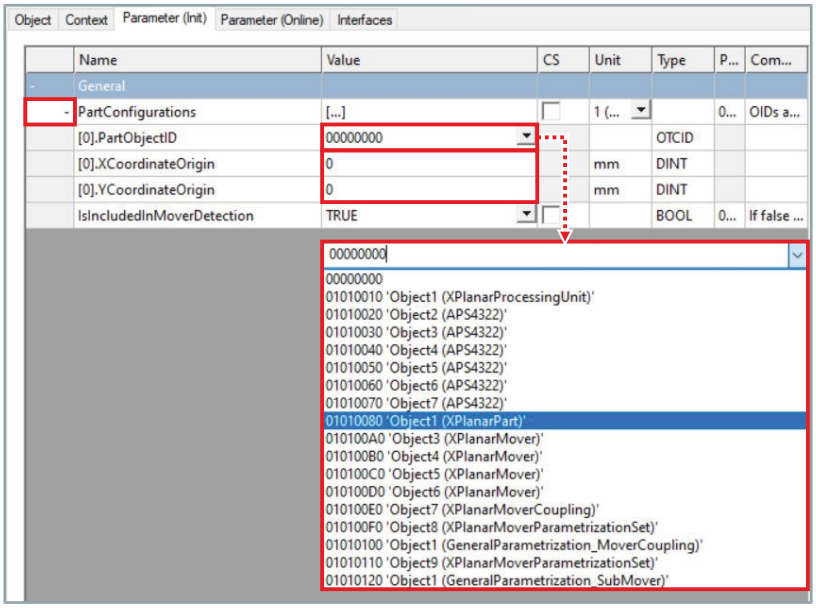
► Enter the mover's **BTN**

► If required, select a parametrization set from the *ParametrizationSetId* drop-down menu

If Mover Identification is not required in the process, the BTN *00000000* can be assigned for all movers. The mover IDs are assigned in the order in which they are detected. Further information can be found in chapter BIC | Beckhoff Identification Code.

7.3 MovementArea

- ► Expand *Solution Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit)*
- ► Double-click on **Object (XPlanarMovementArea)**
- ► Click the *Parameter (Init)* tab in the project window



- ► Click on **+** of *PartConfigurations* to access the input fields
- ► Select **PartObjectID** from the drop-down menu
- ► Enter values for **XCoordinateOrigin** and **YCoordinateOrigin** that correspond to the position origin of the part in the MovementArea

8 Link TcCOM objects

Once all TcCOM objects have been created, some links still need to be created in order to be able to perform calculations.

8.1 Mover

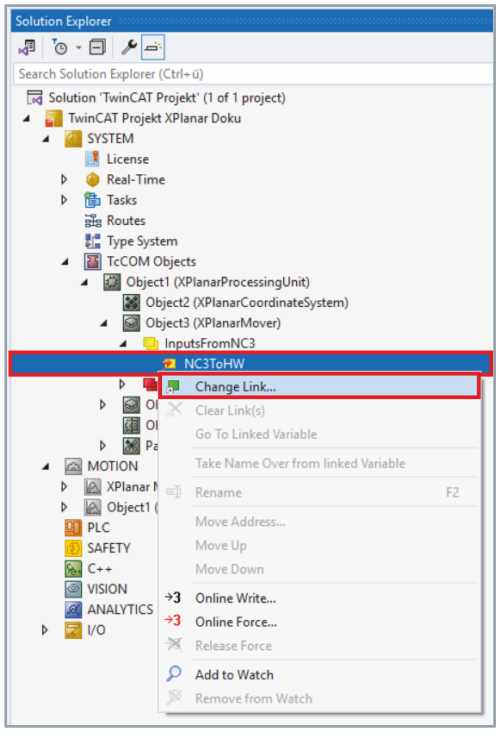
To be able to control the movers, the setpoints of the movers must be calculated. To do this, each XPlanar Mover interface must be linked to external setpoints with a corresponding NC3 counterpart.

Create corresponding mover objects with TF5430
To link the TcCOM objects with the NC3-axes, corresponding XPlanar Mover software objects must be created with the TF5430 TC3 Planar Motion software package. For more information, see the TF5430 documentation:

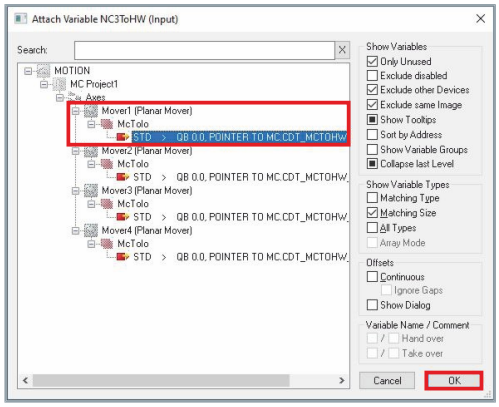
Direct link to the TF5430 | TwinCAT 3 Planar Motion documentation

8.1.1 NC3 To HW

The position setpoints and dynamics setpoints for the mover are provided for the calculation.

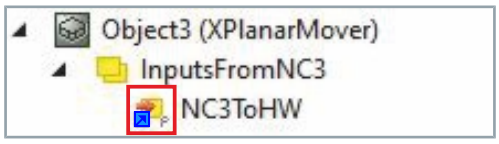


- ► Expand *Solution-Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (XPlanarMover)* > *InputsFromNC3*
- ► Right-click on **NC3ToHW** to open the context menu
- ► Click on **Change Link …** in the context menu



The *Attach Variable NC3ToHW (Input)* dialog box opens.

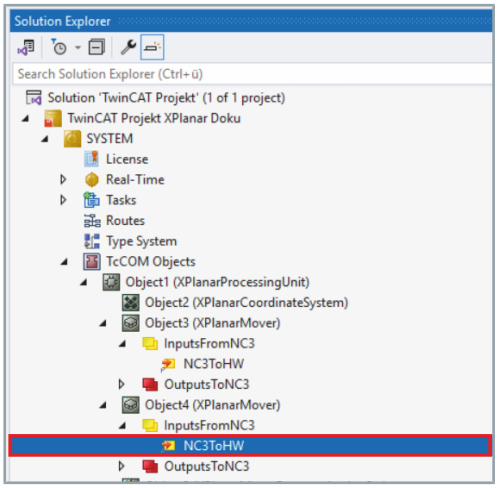
- ► Select the corresponding *STD* variable of a mover
- ► Confirm with **OK**



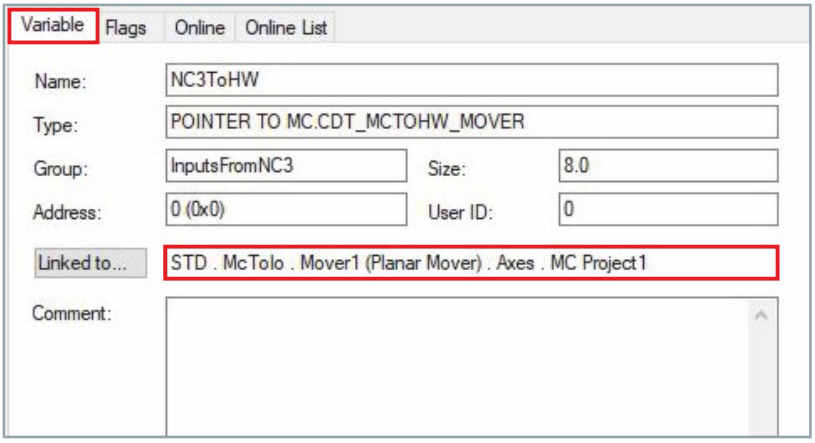
A blue arrow on the variable indicates successful linking.

► Create NC3ToHW links for all movers in the same way

8.1.1.1 NC3ToHW Linking Details



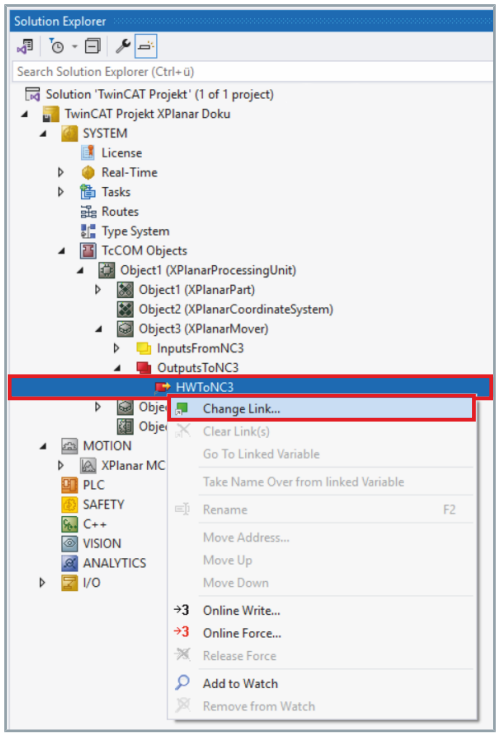
- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (XPlanarMover) > InputsFromNC3*
- ► Click on **NC3ToHW**
- ► Click on the **Variable** tab in the project window



Details of the linking are displayed at **Linked to …**.

8.1.2 HW To NC3

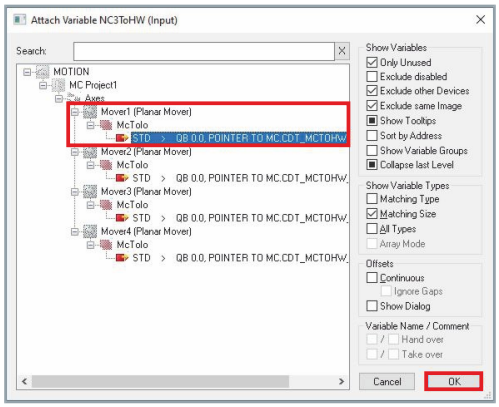
The actual values for the position and dynamics of the mover are provided for the application.



- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (XPlanarMover) > OutputsFromNC3*
- ► Right-click on **HWToNC3** to open the context menu
- ► Click on **Change Link …** in the context menu

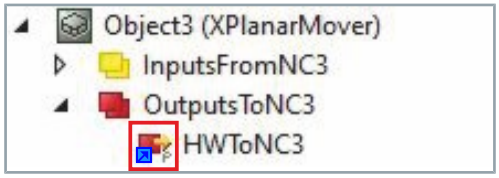
NOTICE

For HWToNC3 linking, make sure that you select the same mover that you used for NC3ToHW linking.



The *Attach Variable HWToNC3 (Output)* dialog box opens.

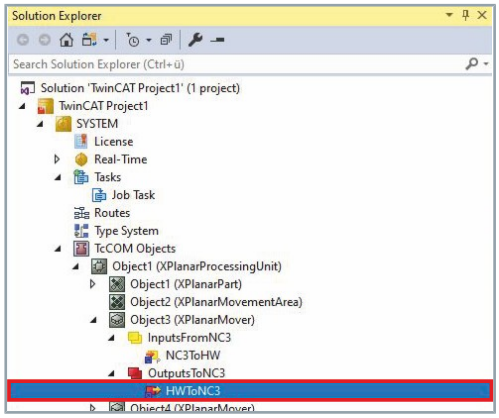
- ► Select the corresponding *STD* variable of a mover
- ► Confirm with **OK**



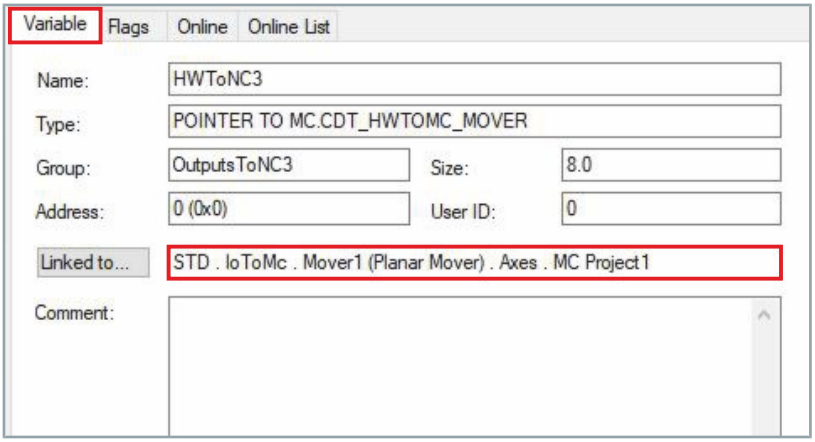
A blue arrow on the variable indicates successful linking.

- ► Create HWToNC3 links for all movers in the same way

8.1.2.1 HWToNC3 Linking Details



- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (XPlanarMover) > OutputsFromNC3*
- ► Click on **HWToNC3**
- ► Click on the **Variable** tab in the project window



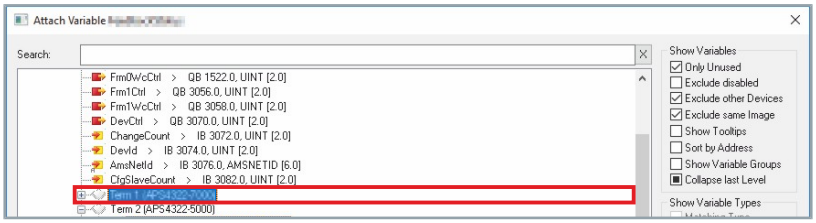
Details of the linking are displayed at **Linked to …**.

8.2 Tiles

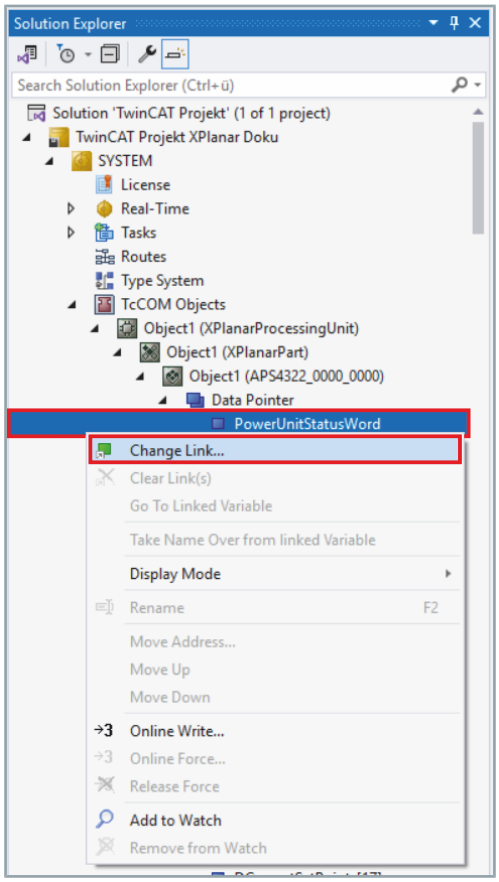
Simulation mode or normal operation
No hardware is required for the simulation operation of the driver and the linking of an *XPlanar Tile* with a physical *APS4xxx-xx00* XPlanar Tile is not necessary.
For normal operation, a physical *APS4xxx-xx00* XPlanar Tile must be assigned to each *XPlanar Tile*.

8.2.1 Drive

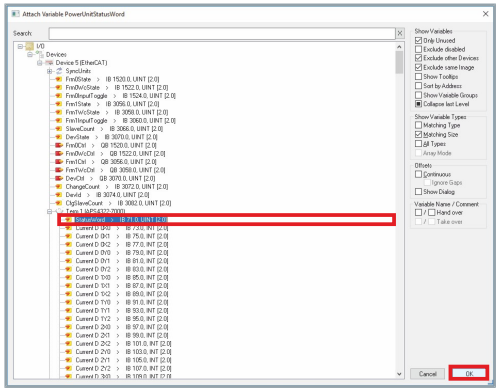
The variables for the drive part of the tiles can be found in the *Term 1 (APS4322-7000)* folder:



8.2.1.1 PowerUnitStatusWord



- ► Expand *Solution-Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (APS4322_0000_0000)* > *Data Pointer*
- ► Right-click on **PowerUnitStatusWord** to open the context menu
- ► Click on **Change Link …** in the context menu

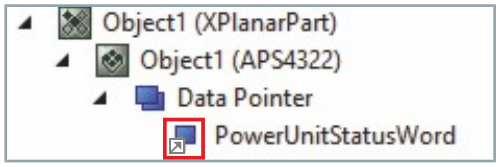


The *Attach Variable PowerUnitStatusWord* dialog box opens.

► Expand *I/O > Devices > Term 1 (APS4322-7000)*

► Select **StatusWord**

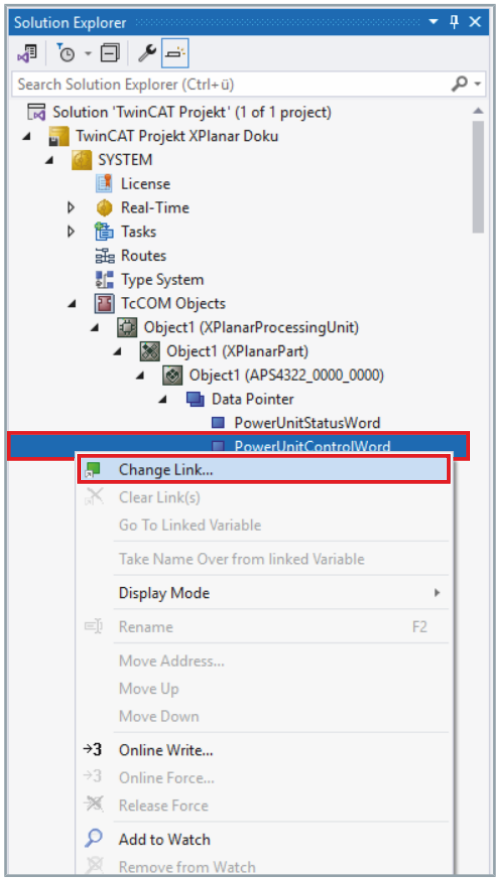
► Confirm with **OK**



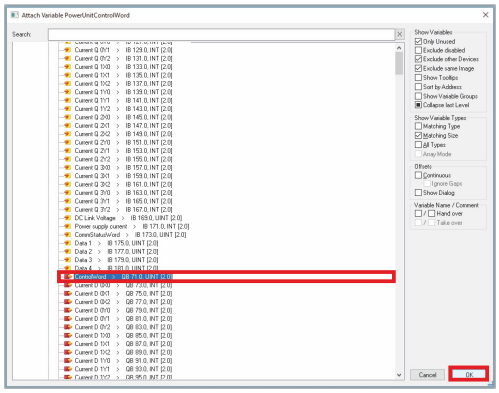
A gray arrow on the variable indicates successful linking.

Further information can be found in chapter "Linking details", [Page 54].

8.2.1.2 PowerUnitControlWord

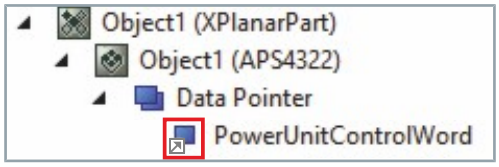


- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (APS4322_0000_0000) > Data Pointer*
- ► Right-click on **PowerUnitControlWord** to open the context menu
- ► Click on **Change Link …** in the context menu



The *Attach Variable PowerUnitControlWord* dialog box opens.

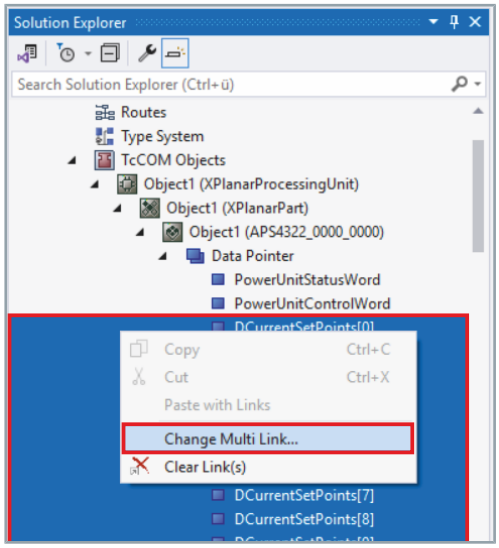
- ► Expand *I/O > Devices > Term 1 (APS4322-7000)*
- ► Select **ControlWord**
- ► Confirm with **OK**



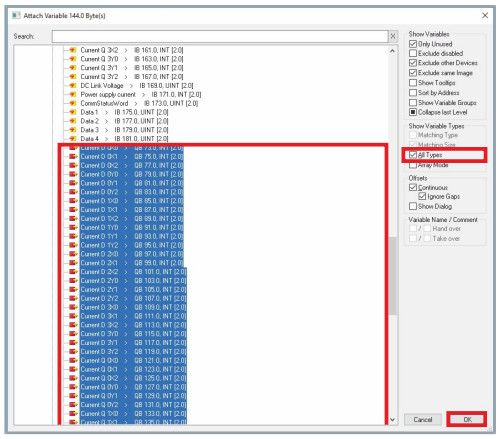
A gray arrow on the variable indicates successful linking.

Further information can be found in chapter "Linking details", [Page 54].

8.2.1.3 CurrentSetpoints

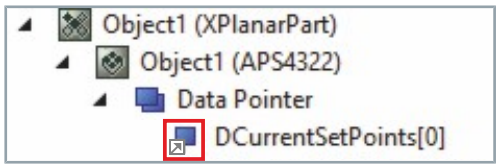


- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (APS4322_0000_0000) > Data Pointer*
- ► Select all **DCurrentSetPoints**, **QCurrentSetPoints** and **PhiSet-Points** from top to bottom
- ► Right-click on the top variable of the selection to open the context menu
- ► Click on **Change Multi Link …** in the context menu



The *Attach Variable 144.0 Byte(s)* dialog box opens.

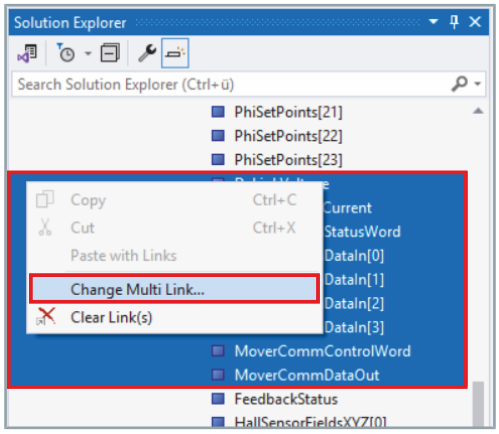
- ► Expand *I/O > Devices > Term 1 (APS4322-7000)*
- ► Select the **All Types** checkbox
- ► Select the following variables:
- **Current D 0X0** to **Current D 3Y2**
- **Current Q 0X0** to **Current Q 3Y2**
- **PhE 0X0** to **PhE 3Y2**
- ► Confirm with **OK**



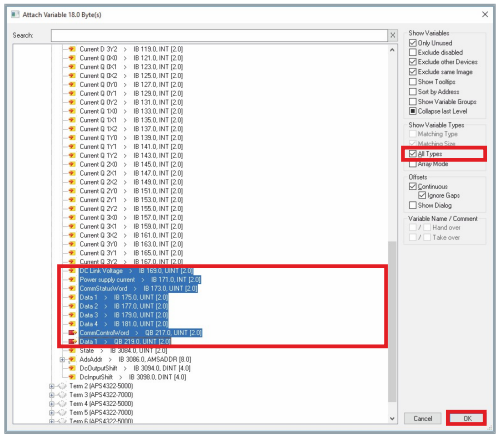
A gray arrow on the variable indicates successful linking.

Further information can be found in chapter "Linking details", [Page 54].

8.2.1.4 Diag Data

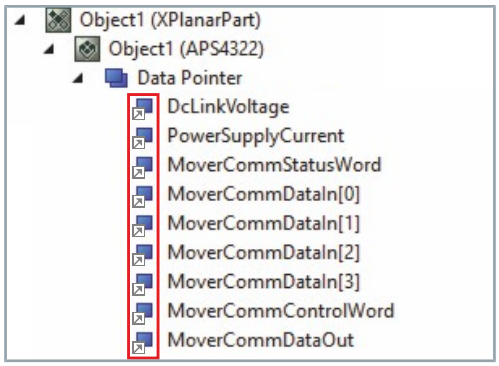


- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (APS4322_0000_0000) > Data Pointer*
- ► Select the following variables from top to bottom:
  - **DcLinkVoltage**
  - **PowerSupplyCurrent**
  - **MoverCommStatusWord**
  - **MoverCommDataIn[0]** to **MoverCommDataIn[3]**
  - **MoverCommControlWord**
  - **MoverCommDataOut**
- ► Right-click on the top variable of the selection to open the context menu
- ► Click on **Change Multi Link …** in the context menu



The *Attach Variable 18.0 Byte(s)* dialog box opens.

- ► Expand *I/O > Devices > Term 1 (APS4322-7000)*
- ► Select the **All Types** checkbox
- ► Select the following variables:
  - **DC Link Voltage**
  - **Power supply current**
  - **CommStatusWord**
  - **Data 1** to **Data 4**
  - **CommControlWord**
  - **Data 1**
- ► Confirm with **OK**

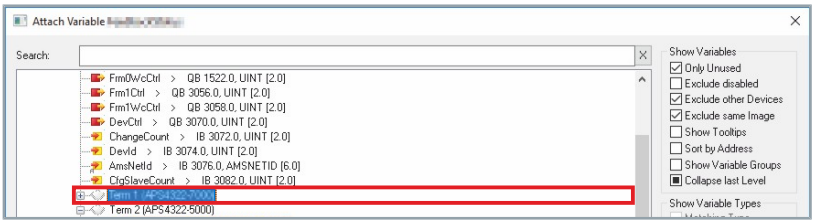


A gray arrow on the variable indicates successful linking.

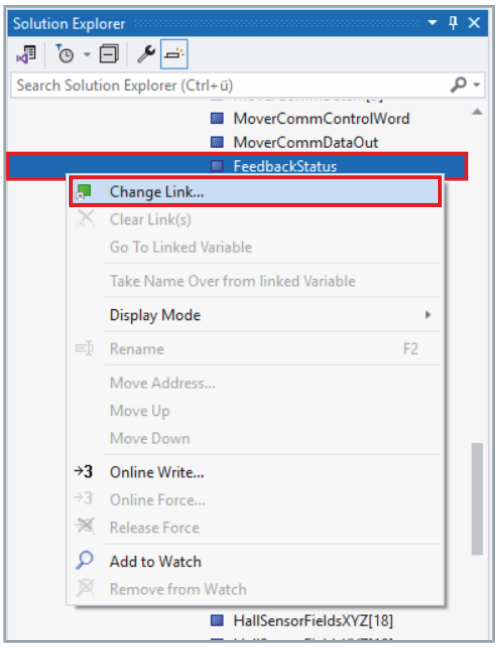
Further information can be found in chapter "Linking details", [Page 54].

8.2.2 Feedback

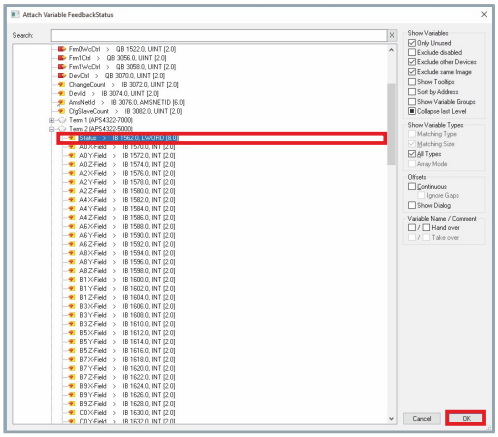
The variables for the feedback part of the tiles can be found in the *Term 2 (APS4322-5000)* folder:



8.2.2.1 FeedbackStatus

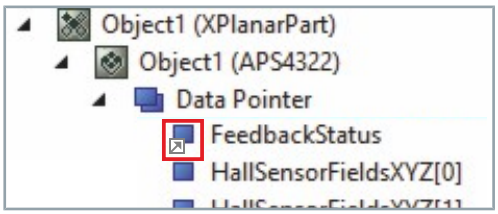


- ► Expand *Solution-Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (APS4322_0000_0000)* > *Data Pointer*
- ► Right-click on **FeedbackStatus** to open the context menu
- ► Click on **Change Link …** in the context menu



The *Attach Variable FeedbackStatus* dialog box opens.

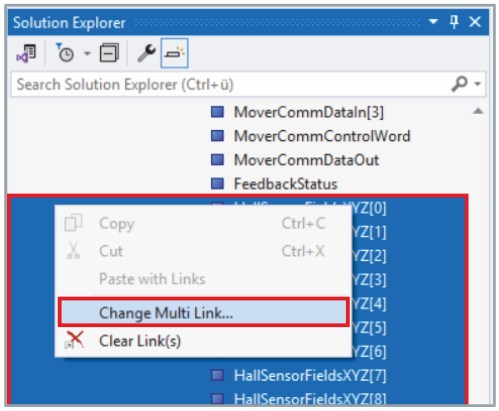
- ► Expand *I/O* > *Devices* > *Term 2 (APS4322-5000)*
- ► Select **Status**
- ► Confirm with **OK**



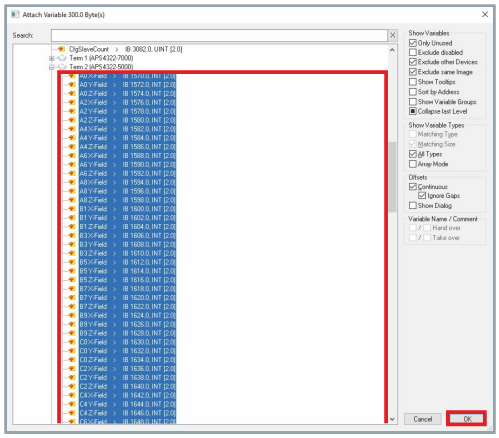
A gray arrow on the variable indicates successful linking.

Further information can be found in chapter "Linking details", [Page 54].

8.2.2.2 HallSensors

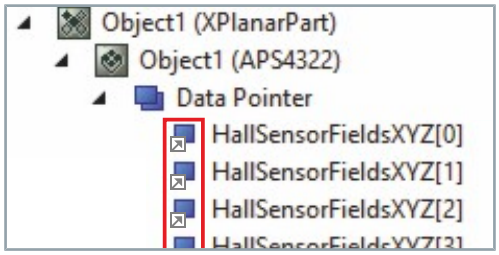


- ► Expand *Solution-Explorer > System > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (APS4322_0000_0000) > Data Pointer*
- ► Select all **HallSensorFieldsXYZ[0]** to **HallSensorFieldsXYZ[149]** from top to bottom
- ► Right-click on the top variable of the selection to open the context menu
- ► Click on **Change Multi Link ...** in the context menu



The *Attach Variable 300.0 Byte(s)* dialog box opens.

- ► Expand *I/O > Devices > Term 2 (APS4322-5000)*
- ► Select the **All Types** checkbox
- ► Select all **A0 X-Field** to **J9 Z-Field**
- ► Confirm with **OK**



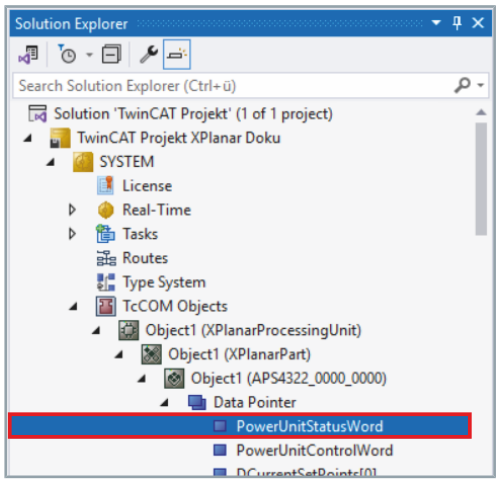
A gray arrow on the variable indicates successful linking.

Further information can be found in chapter "Linking details", [Page 54].

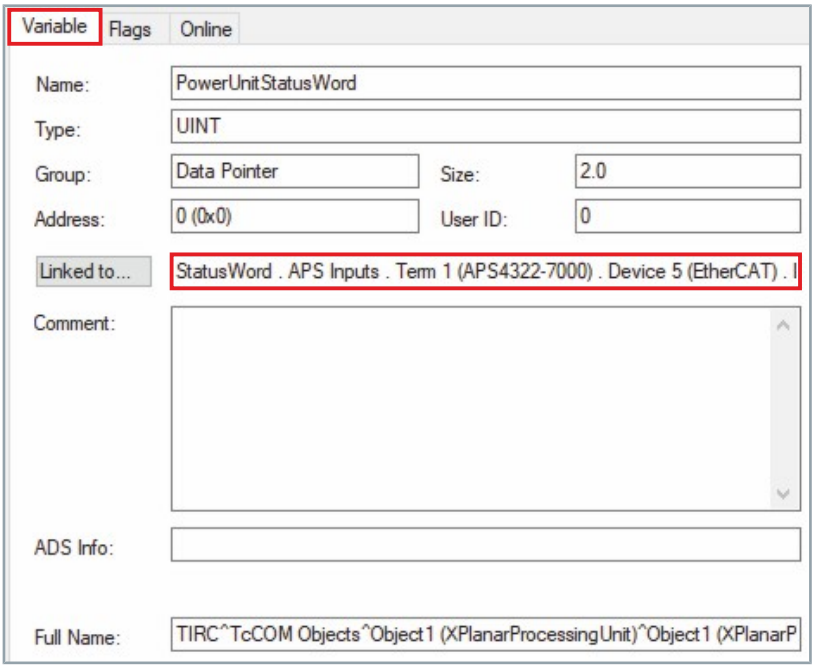
8.2.3 Linking details

After successful linking of the *Tiles* TcCOM objects with the physical tiles of the I/O, the linking is indicated by gray arrows at the individual Data Pointer variables of the TcCOM objects. You now have the option to view more details about the linking.

Linking example
This chapter uses a successful *PowerUnitStatusWord* linking as an example to show you how to display the details of the linking.



- ► Expand *Solution-Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (APS4322_0000_0000)* > *Data Pointer*
- ► Click on **PowerUnitStatusWord**
- ► Click on the **Variable** tab in the project window

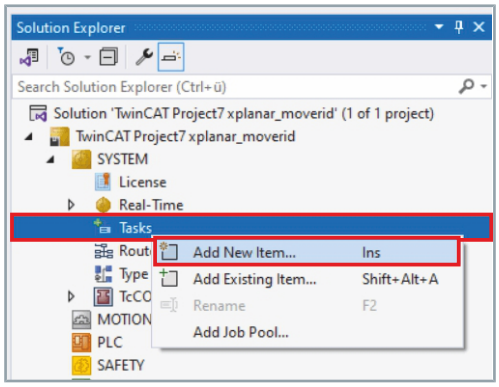


Details of the linking are displayed at **Linked to …**.

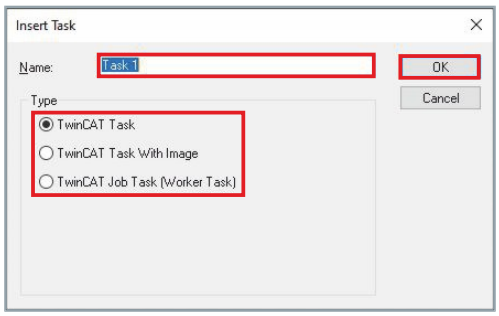
9 Add task

To be able to perform active calculations, each XPlanar Tile and each XPlanar Mover requires a task. All XPlanar Tiles that belong to an EtherCAT master must run on the same task.

Each task may only run on one core and must be set to 250 µs.



- ► Expand *Solution-Explorer* > *System*
- ► Right-click on **Tasks** to open the context menu
- ► Click **Add New Item** in the context menu



The *Insert Task* dialog box opens.

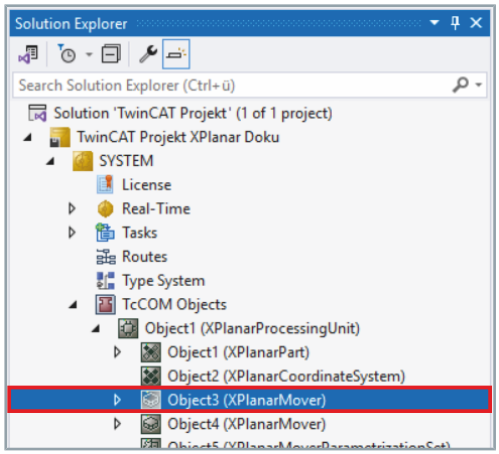
- ► Enter the **name** for the task
- ► Select task **type**
- ► Confirm your entry and selection with **OK**

A correspondingly named task is added to the tasks.

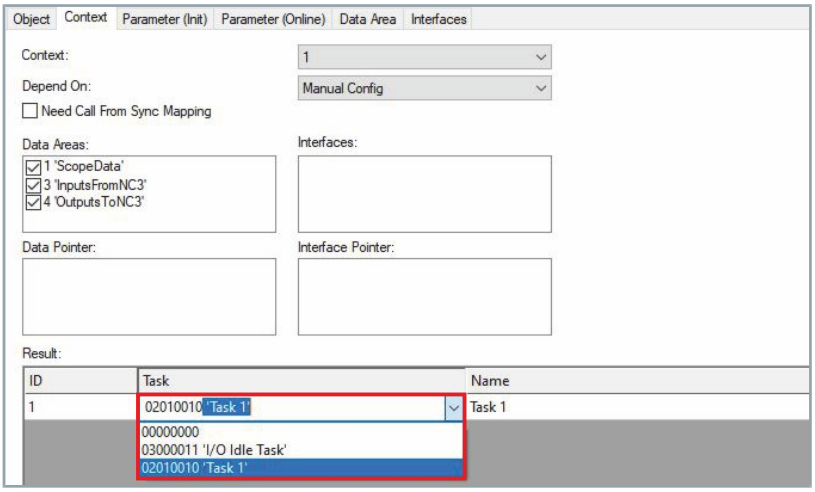
10 Link task

After one or more tasks have been added, all XPlanar Tiles and XPlanar Movers must be linked to a task.

10.1 Mover

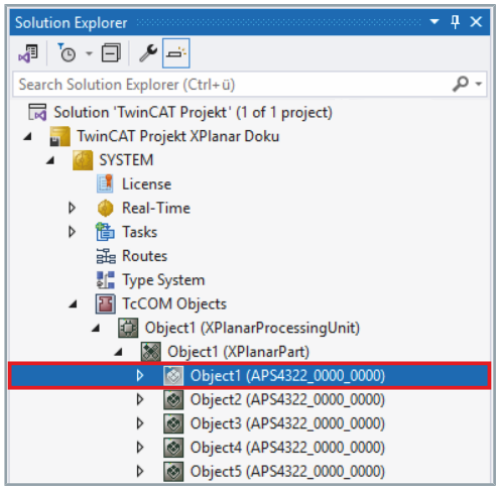


- ► Expand *Solution-Explorer > TcCOM Objects > Object (XPlanarProcessingUnit)*
- ► Double-click on **Object (XPlanar Mover)**
- ► Click on the **Context** tab in the project window

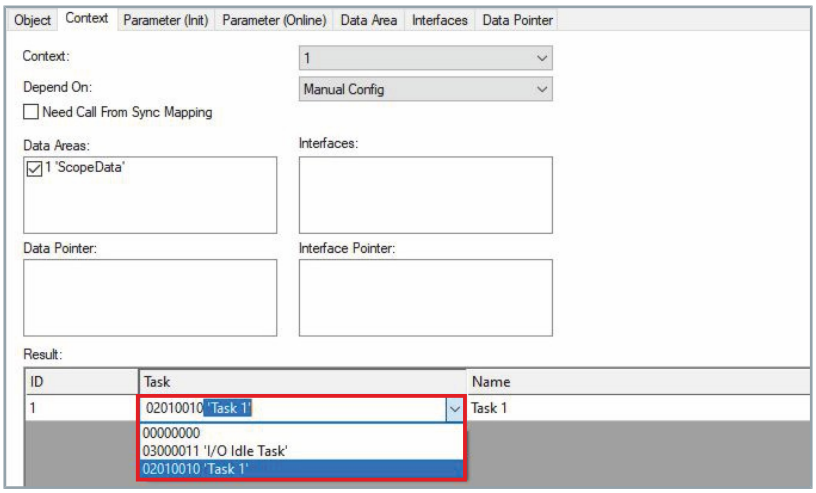


- ► Select the task in the **Task** drop-down menu

10.2 Tile



- ► Expand *Solution-Explorer > TcCOM Objects > Object (XPlanarProcessingUnit) > Object (XPlanarPart)*
- ► Double-click on **Object (APS4322_0000_0000)**
- ► Click on the **Context** tab in the project window

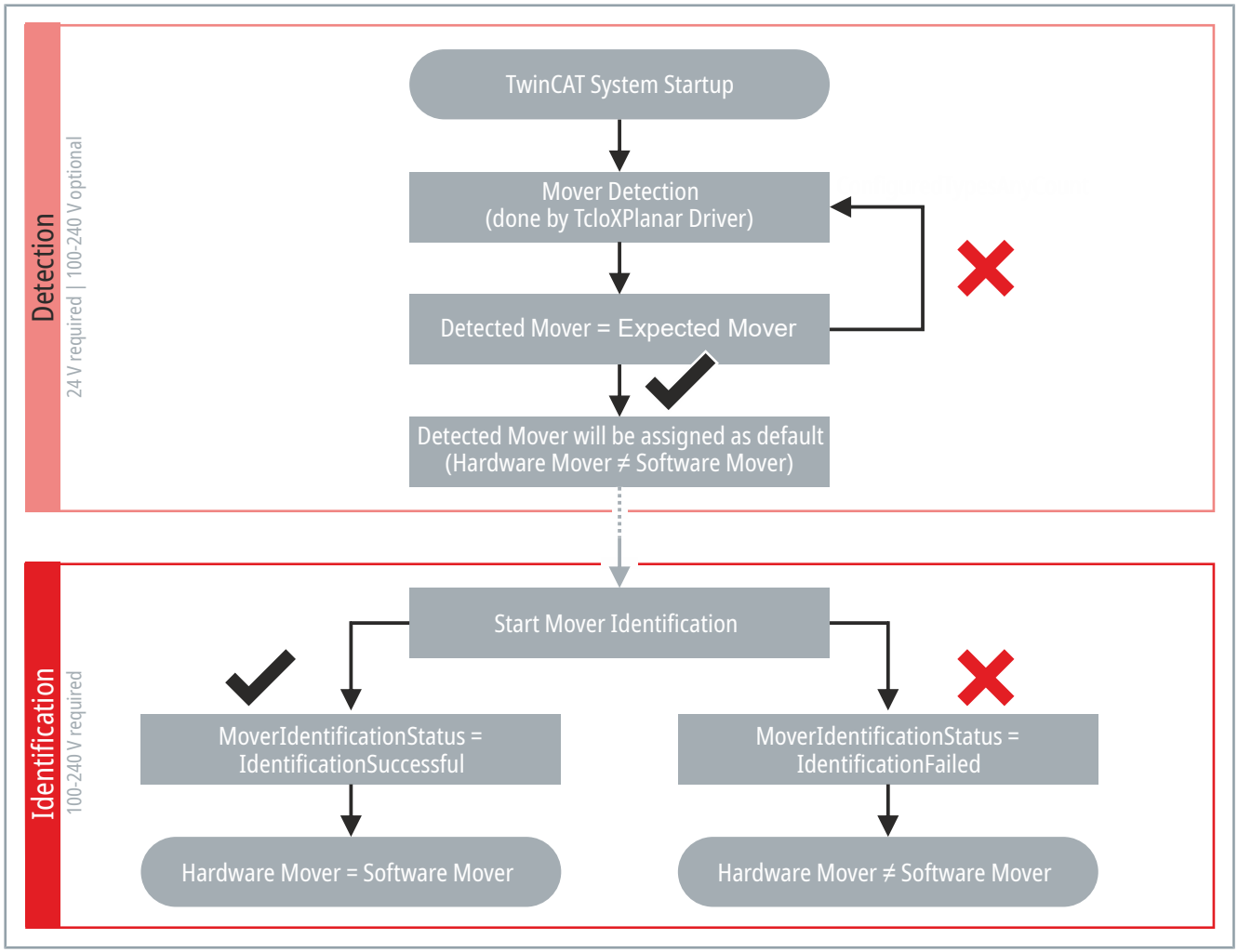


► Select the task in the **Task** drop-down menu

11 Mover Detection and Mover Identification

Movers that have an ID bumper automatically transmit their ID during Mover Identification so that the mover can be accurately assigned to the XPlanar system. Mover Detection depends on the selection of the *DetectionMode* parameter. Movers with ID bumpers are required for Mover Identification.

11.1 Detection Mode: AsConfigured



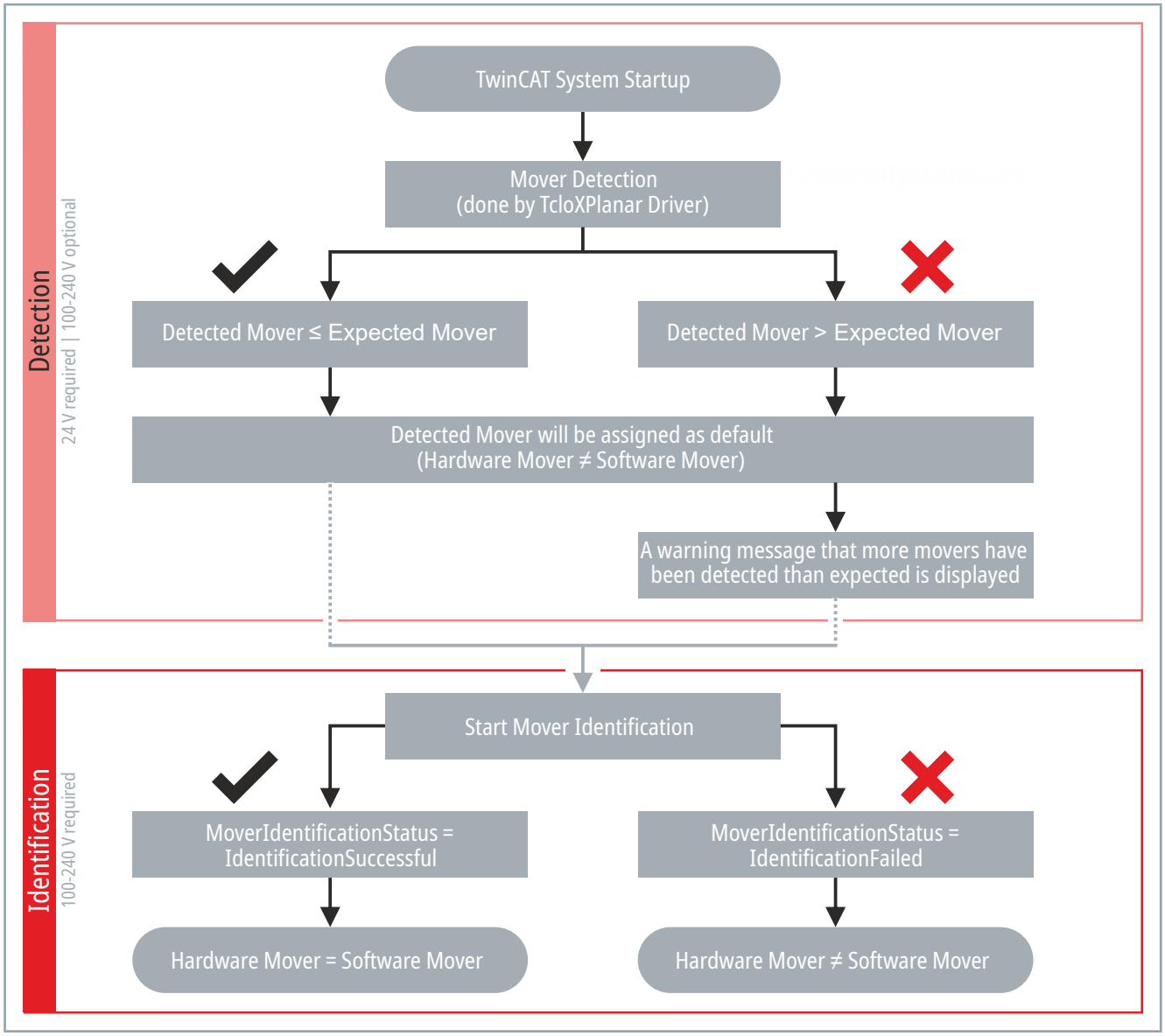
After TwinCAT system startup, a Mover Detection is performed and a check is made to see whether the number of detected movers matches the number of expected movers. During detection, only the absolute positions of the movers are detected. The movers are assigned to the software axes using ascending Y-positions. Mover Detection is terminated when the number of detected movers matches the expected number of movers.

If the number matches and the load voltage is applied to the XPlanar system, Mover Identification can be started. As soon as Mover Identification has been started, the system automatically communicates with the movers with ID bumpers to establish a unique assignment between the XPlanar Movers and the software axes.

The current status of the communication process is displayed via the *CoordinatedMoverCommStatus*. After successful identification, the hardware movers are assigned to the software movers.

11.2 Detection Mode: ConfiguredTypesAnyCount

More configured movers than existing movers possible
The DetectionMode *ConfiguredTypesAnyCount* allows you to create more movers in the configuration of your system than are available on the system. Movers that are not available can be added at a later date.



After TwinCAT system startup, a Mover Detection is performed and a check is made to see whether the number of detected movers matches the number of expected movers. During detection, only the absolute positions of the movers are detected. The movers are assigned to the software axes using ascending Y-positions. The Mover Detection runs through once.

If the number of detected movers is greater than the number of expected movers, a warning message is issued. Too many movers must be removed before Mover Identification.

⚠ WARNING

Correct the number of movers after receiving a warning message

If more movers have been detected on the system than expected, you must adjust the number of movers on the system according to your configuration or the configuration according to the number of movers on the system.

If the number of detected and expected movers does not match, this can result in damage to the movers and injuries from splinters in the eyes.

- Remove the excess mover.
- Adjust the configuration according to the detected excess movers.

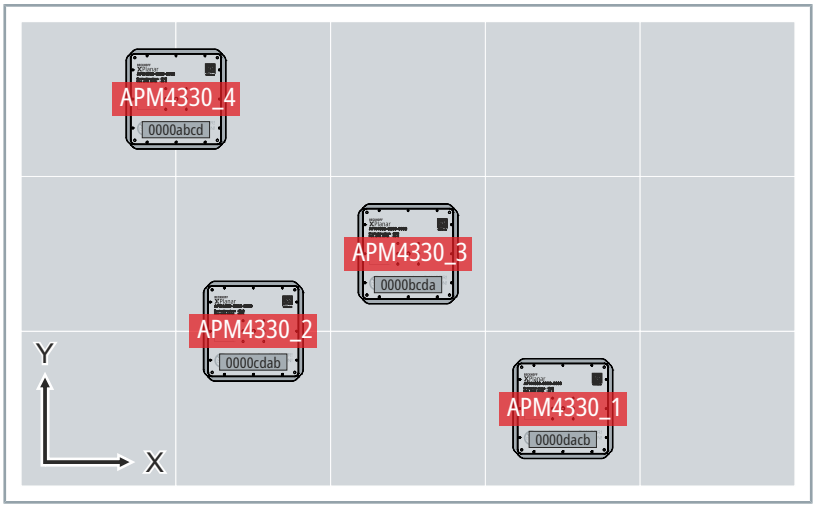
After successful detection, removal of excess movers and when the load voltage is applied to the XPlanar system, Mover Identification can be started. As soon as Mover Identification has been started, the system automatically communicates with the movers with ID bumpers to establish a unique assignment between the XPlanar Movers and the software axes.

The current status of the communication process is displayed via the *CoordinatedMoverCommStatus*. After successful identification, the hardware movers are assigned to the software movers.

11.3 Mover Assignment

After system startup and before Mover Identification has been performed, the software Mover ID and the hardware Mover ID do not match:

Mover assignment after system startup

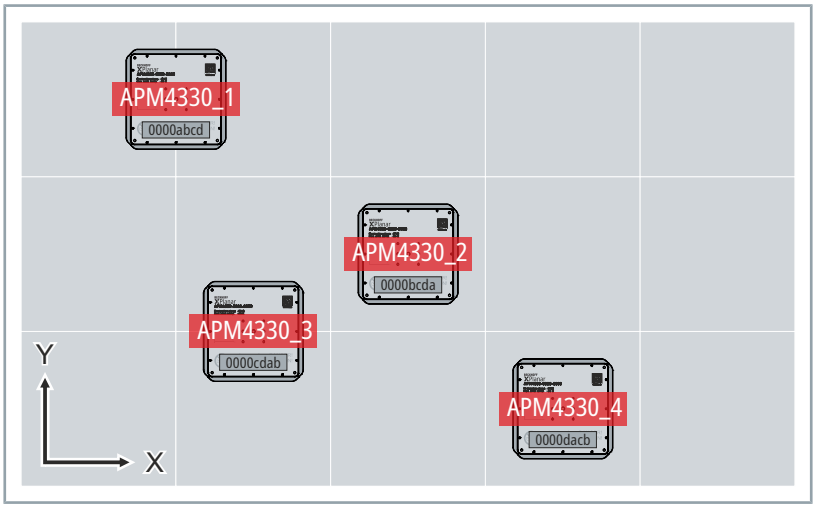


Software mover	Hardware mover (BTN)		Position (X/Y)
APM4330_1 (0000abcd)[1)]	≠	0000dacb	840/120
APM4330_2 (0000bcda)[1)]	≠	0000cdba	360/240
APM4330_3 (0000cdab)[1)]	≠	0000bcda	600/360
APM4330_4 (0000dabc)[1)]	≠	0000abcd	240/600

1) 0000xxxx is the configured BTN in the software object

Mover assignment after Mover Identification

After a Mover Identification has been carried out, the assignment of the software Mover ID to the hardware Mover ID is correct:



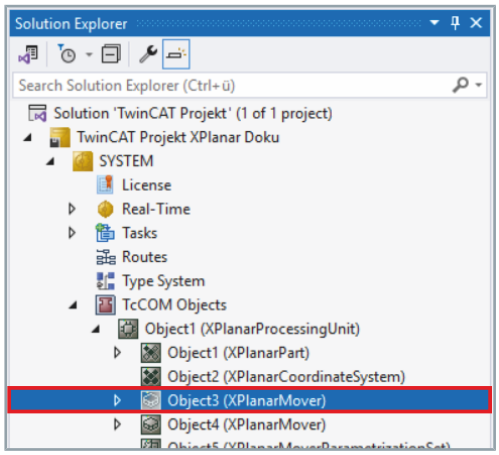
Software mover	Hardware mover (BTN)		Position (X/Y)
APM4330_1 (0000abcd)[1)]	=	0000abcd	240/600
APM4330_2 (0000bcda)[1)]	=	0000bcda	600/360
APM4330_3 (0000cdab)[1)]	=	0000cdab	360/240
APM4330_4 (0000dabc)[1)]	=	0000dabc	840/120

1) 0000xxxx is the configured BTN in the software object

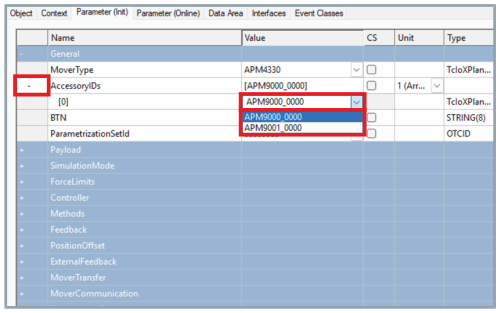
11.4 Required Settings

Before Mover Detection can be started, some settings must be made in TwinCAT.

Example presentation
The following example shows the settings for an *APM4330-0000* mover and an *AMP9001-0000-4330* ID bumper. The figures show the corresponding settings for this example configuration.



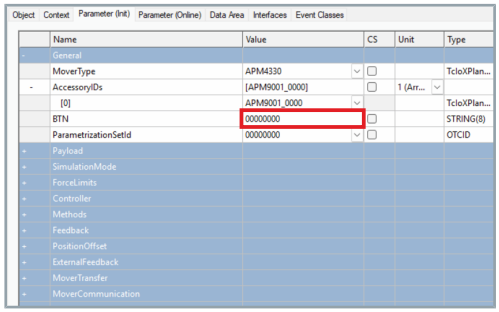
► Expand *Solution Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)*
► Click on **Object (XPlanarMover)**



► Click the **Parameter (Init)** tab in the project window
► Expand *General*
► Expand *AccessoryIDs*
► Select ID bumper **APM9001-0000** in the *[0]* drop-down menu

BTN of the mover
Each mover requires its own *BTN | Beckhoff Traceability Number*. The mover's BTN can be found on every mover under the BIC | Beckhoff Identification Code. Multiple assignment of a mover BTN leads to errors in Mover Identification.
Further information on the BTN can be found in the XPlanar original operating instructions or at:

www.beckhoff.com/bic

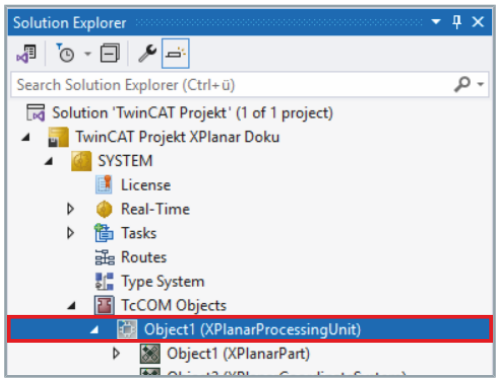


► Enter the BTN of a hardware mover in the *BTN* input field

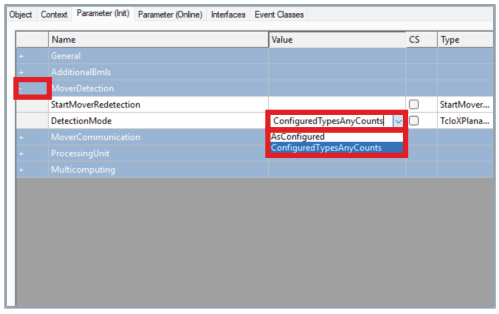
► Configure ID bumpers and BTN for all *objects (XPlanarMover)* in the same way

To apply the settings, the configuration must be reactivated and the TwinCAT system must be restarted.

11.5 Starting Mover Detection



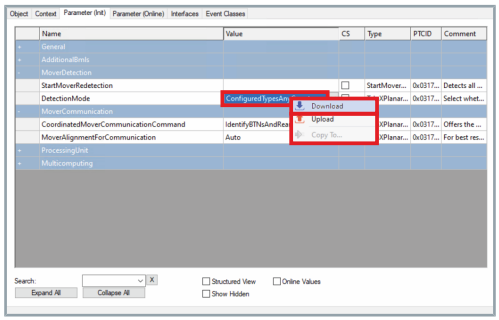
- ▶ Expand *Solution Explorer* > *System* > *TcCOM Objects*
- ▶ Click on **Object (XPlanarProcessingUnit)**



- ▶ Click the **Parameter (Init)** tab in the project window
- ▶ Expand *MoverDetection*
- ▶ Select **AsConfigured** in the *DetectionMode* drop-down menu if Mover Detection is to be executed until the number of detected and expected movers matches

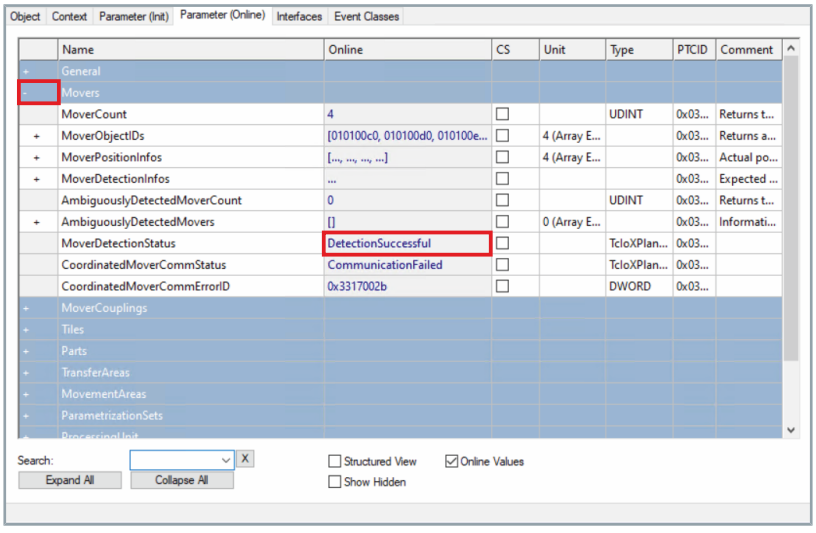
OR

- ▶ Select **ConfiguredTypesAnyCounts** in the *DetectionMode* drop-down menu if the Mover Detection is to be run through once



- ▶ Right-click in the *DetectionMode* input field to open the context menu
- ▶ In the context menu click **Download**

11.5.1 Control of Mover Detection



- ▶ Click on the **Parameter (Online)** tab in the project window
- ▶ Expand *Movers*
- ▶ Check status of *MoverDetectionStatus*

11.6 Starting Mover Identification

Once the ID bumpers and BTN have been configured for all movers, Mover Identification can be started. You have two options for starting Mover Identification:

- via the *XPlanarProcessingUnit object* in TwinCAT
- via the PLC.

ID bumper required
ID bumpers are required to identify the movers. Passive bumpers *APM9000-0000* can be replaced with ID bumpers *APM9001-0000*. Further information on this can be found in the XPlanar original operating instructions:

Direct link to the XPlanar APS4322-0000 original operating instructions

Direct link to the XPlanar APS42xx-1x00 original operating instructions

ID bumpers *APM9001-0000* are available for the following movers:

- APM4220-0000
- APM4221-0000
- APM4330-0000
- APM4550-0000

The *APM4330-0001* mover has an integrated ID function.

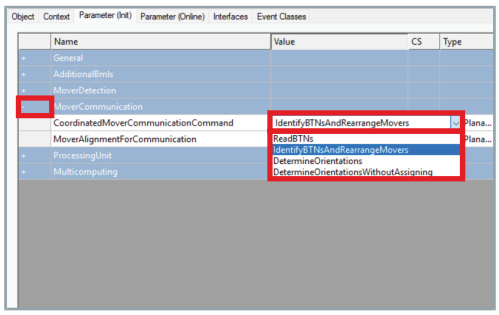
11.6.1 Supported Combinations

So far, not all combinations of tiles and movers are supported in Mover Identification.

Mover	Tile	
	APS4322-0000	APS42xx-0000
APM4220-0000	✓	✓
APM4221-0000	✓	–
APM4330-0000	✓	✓
APM4330-0001	✓	–
APM4550-0000	✓	✓
Mover coupling	–	–

11.6.2 Object XPlanarProcessingUnit

To start Mover Identification via the *object (XPlanarProcessingUnit)* in TwinCAT, the XPlanar system must be in *Run Mode* and the load voltage must be applied to the system.



- ► Expand *MoverCommunication*
- ► Select **ReadBTNs** in the *CoordinatedMoverCommunicationCommand* drop-down menu if the BTNs of the mover objects are to be saved for reading into the PLC under *XPlanarMover/Parameter(Online)/MoverCommunicationData/IdentifiedMoverBTN*
- ► Select **IdentifyBTNsAndRearrangeMovers** in the *CoordinatedMoverCommunicationCommand* drop-down menu if mover objects are to be assigned to the BTNs found
- ► Select **DetectOrientationsWithoutAssigning** in the *CoordinatedMoverCommunicationCommand* drop-down menu if the mover's orientation is to be detected and not applied

OR

- ► Select **DetectOrientations** in the *CoordinatedMoverCommunicationCommand* drop-down menu if the mover's orientation is to be detected and assigned

⚠ WARNING

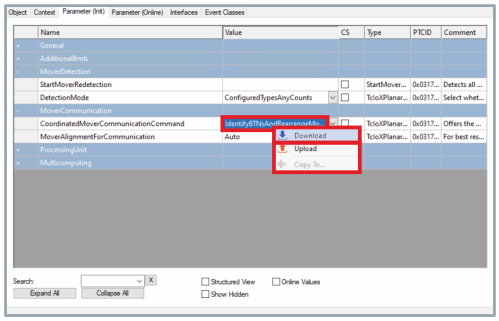
Correct the number of movers after receiving a warning message

If more movers have been detected on the system than expected, you must adjust the number of movers on the system according to your configuration or the configuration according to the number of movers on the system.

If the number of detected and expected movers does not match, this can result in damage to the movers and injuries from splinters in the eyes.

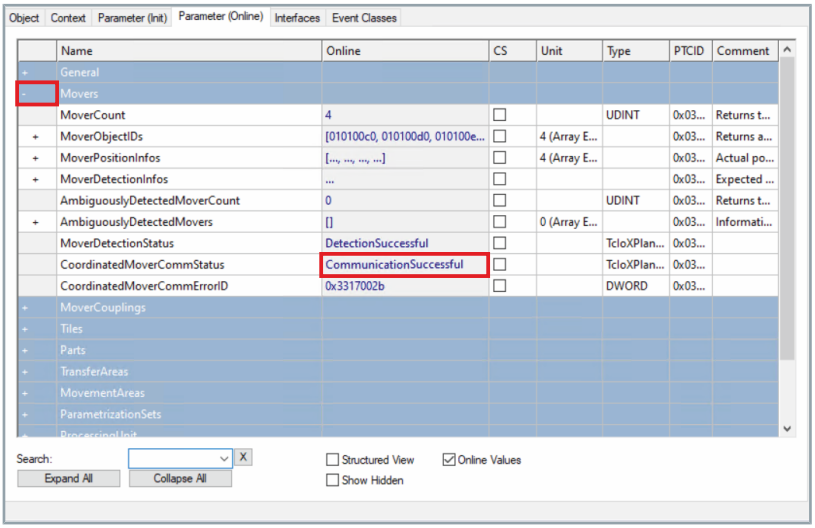
- Remove the excess mover.
- Adjust the configuration according to the detected excess movers.

Only start Mover Identification when the number of detected movers is less than or equal to the number of expected movers.



- ► Right-click in the *CoordinatedMoverCommunicationCommand* input field to open the context menu
- ► In the context menu click **Download**

11.6.2.1 Control of Mover Identification



- ► Click on the **Parameter (Online)** tab in the project window
- ► Expand *Movers*
- ► Check status of *CoordinateMoverCommStatus*

Mover Identification error
If the Mover Identification was not successful, check the entries for the **AccessoryIDs** and **BTN** of all movers and restart the Mover Identification. This error often occurs when there is no power supply and the system is in emergency shutdown.

11.6.3 PLC

This chapter describes how to start Mover Identification via your PLC.

TF5890 | TwinCAT 3 XPlanar required
The XPlanar Utility library is a prerequisite for starting Mover Identification, which is carried out with the setup of the *TF5890 | TwinCAT 3 XPlanar*.

11.6.3.1 Variable Declaration

```
VAR
   fbEnviroment               :FB_TcIoXPlanarEnviroment;
   ipXpu                      :I_TcIoXPlanarProcessingUnit;
   bStartMoverIdentification  :BOOL;
   nStateXPU                  :INT;
END_VAR
```

11.6.3.2 Example of a program sequence

```
CASE nStateXPU OF
0: // init environment
          IF fbEnvironment.Init(TRUE) THEN
                ipXpu:=fbEnvironment.XpuTcIo(1);
                nStateXPU:=1;
          END_IF

1: //check if MoverDetection is done and the Identification process should be started
          IF bStartMoverIdentification
                AND (ipXpu.GetMoverDetectionInfos().TotalDetected = ipXpu.GetMoverDetectionInfos().To-
talExpected)
                THEN
                   ipXpu.SetStartMoverIdentification():
                   nStateXpu:=2;
          END_IF
2://check and wait if the Identification process has succeded successfully
          IF ipXpu.GetMoverDetectionInfos = MoverIdentificationStatus.IdentificationSuccessful THEN
                //MoverIdentification Successful
                nStateXPU:=3
          END IF
END_CASE
```

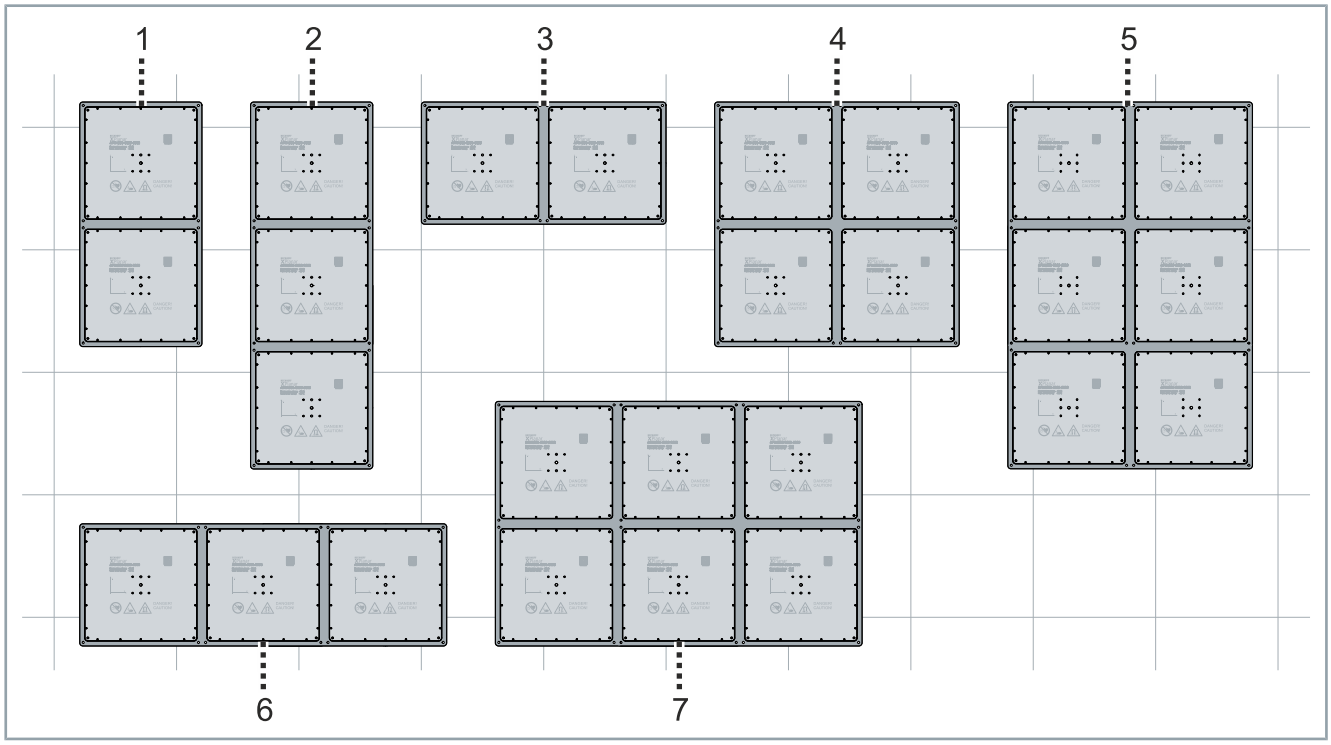
12 Mover coupling

A frame can be used to connect two or more movers to form a mover coupling. Some parameters must be defined for the mover coupling and the submovers so that the mover coupling can be controlled in *TwinCAT 3*.

Mover Identification only for individual movers
Mover Identification is only possible for individual movers with ID bumpers. Mover couplings cannot be identified as the movers are mounted in the mover coupling without ID bumpers. Mover couplings can only be detected via Mover Detection.

12.1 Recommended arrangements for tiles APS4322-0000

Any number of movers can be coupled with one frame as long as they can be controlled by the IPC. When using *APS4322-0000* tiles, a center-to-center distance of 240 mm between movers must be maintained to ensure an almost linear increase in the payload. The following figure shows examples of mover couplings with two, four or six movers.

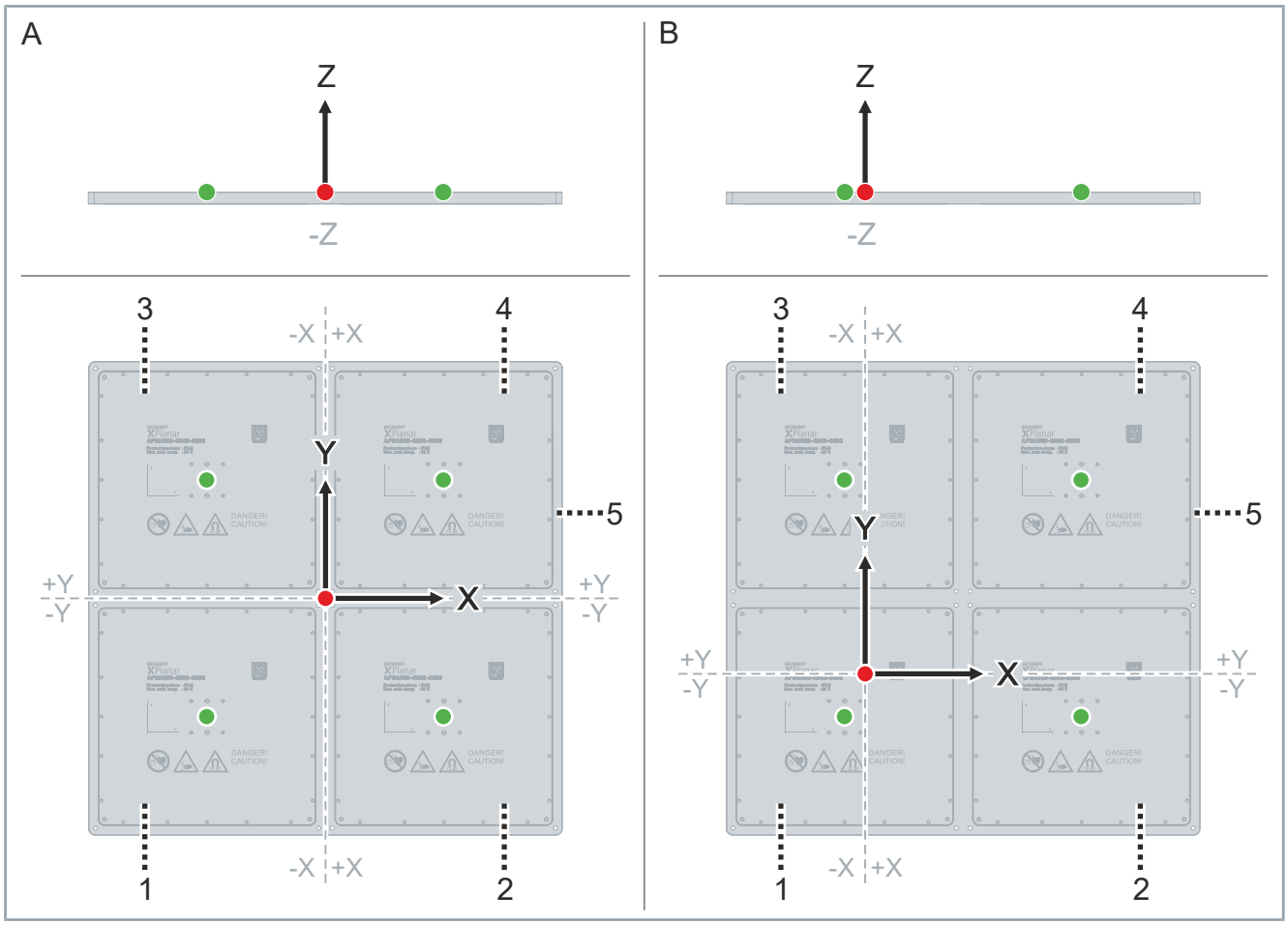


Position	Explanation
1	1 x 2 submovers
2	1 x 3 submovers
3	2 x 1 submovers
4	2 x 2 submovers
5	2 x 3 submovers
6	3 x 1 submovers
7	3 x 2 submovers

12.2 Setpoint origin

Starting from the setpoint origin O, the values of the setpoint origins OX of the submovers in the X-axis, Y-axis and Z-axis must be calculated.

Examples of the positioning of the setpoint origin
The positioning of the setpoint origin is shown using two 2 x 2 mover couplings as an example.



Position	Explanation
1	Submover 1
2	Submover 2
3	Submover 3
4	Submover 4
5	Framework of the mover coupling
●	Setpoint origin O of the mover coupling
●	Setpoint origin O_x of the submover, O_1 for submover 1, …

Example A
The setpoint origin O is located in the center of the surface of the mover coupling. The setpoint origins O_X of the submovers have the same distance to the setpoint origin O.

Example B
The setpoint origin O is not in the center of the mover coupling. The setpoint origins O_X of the movers have different distances to the setpoint origin O.

Starting from the setpoint origin O, the values of the setpoint origins O_X of the submovers in the X-axis, Y-axis and Z-axis must be calculated and entered under the parameters (Init) of *XPlanarRigidMoverCoupling*. The following tables show values for examples A and B on the previous page with *APS4550-0000* movers.

X-axis

Setpoint origin	**O**	**O_1**	**O_2**	**O_3**	**O_4**
Distance example A [mm]	0	-120	120	-120	120
Distance example B [mm]	0	-20	220	-20	220

Y-axis

Setpoint origin	**O**	**O_1**	**O_2**	**O_3**	**O_4**
Distance example A [mm]	0	-120	-120	120	120
Distance example B [mm]	0	-45	-45	195	195

Z-axis

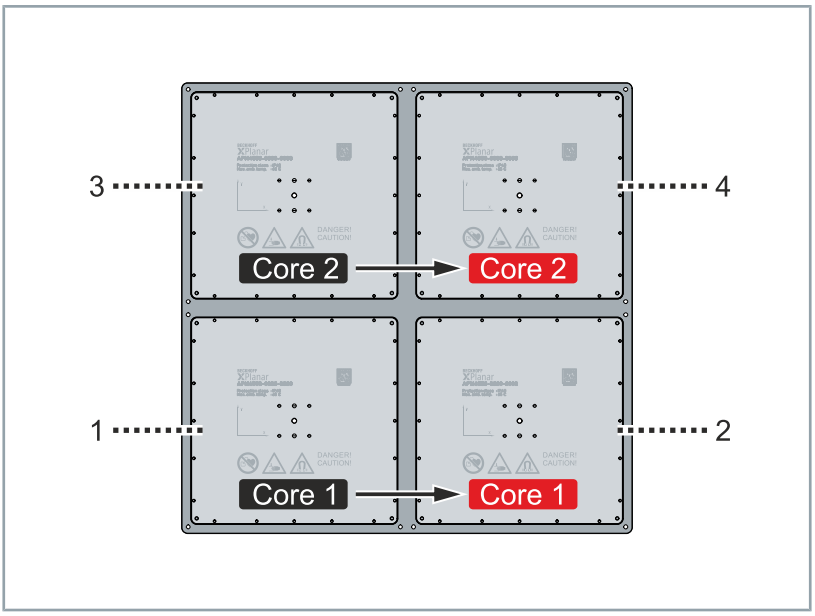
Setpoint origin	**O**	**O_1**	**O_2**	**O_3**	**O_4**
Distance example A [mm]	0	0	0	0	0
Distance example B [mm]	0	0	0	0	0

12.3 CPU core

12.3.1 Mover coupling 2 x 2

Each mover of a 2 x 2 mover coupling requires its own CPU core to be able to perform calculations at sufficient speed.

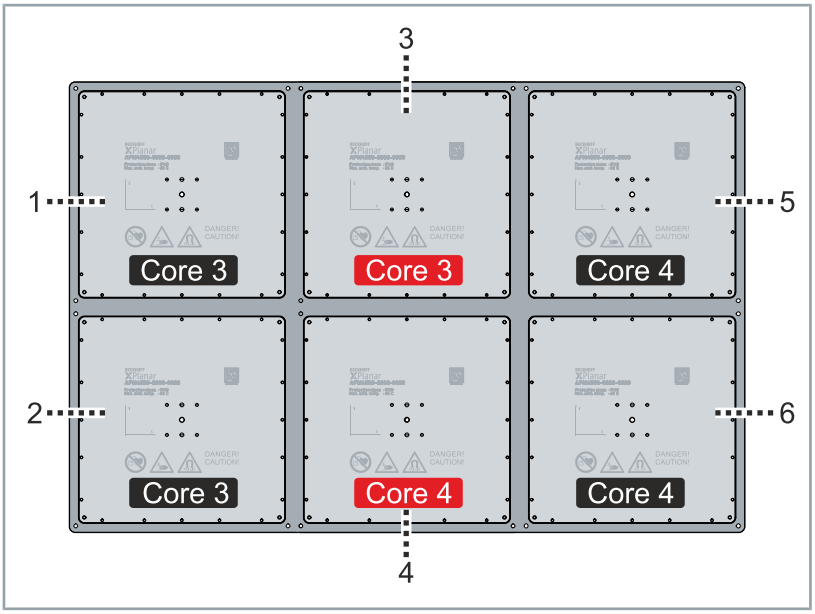
The task required for a mover coupling can share a CPU core with other tasks.



Position	Name	CPU core
1	Submover 1	1
2	Submover 2	1
3	Submover 3	2
4	Submover 4	2

12.3.2 Mover coupling 3 x 2

For larger mover couplings of 3 x 2 or more movers, two movers share a CPU core at each corner. As the middle movers perform fewer calculations, they can be added to one of the two CPU cores of the corner movers.

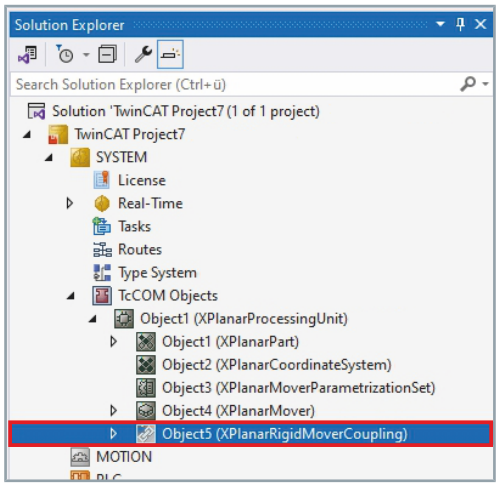


Position	Name	CPU core
1	Submover 1	3
2	Submover 2	3
3	Submover 3	3
4	Submover 4	4
5	Submover 5	4
6	Submover 6	4

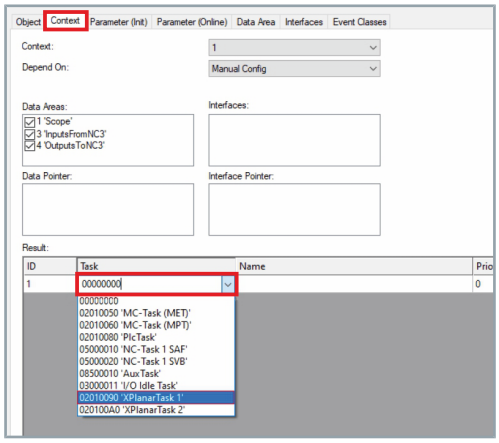
12.4 Selecting a Submover

A TcCOM object *XPlanarMoverCoupling* must be added to the processing unit before the linking of the submovers can begin. Further information can be found in chapter "Mover coupling", [Page 35].

The submovers are created automatically when the *XPlanarRigidMoverCoupling* is created.

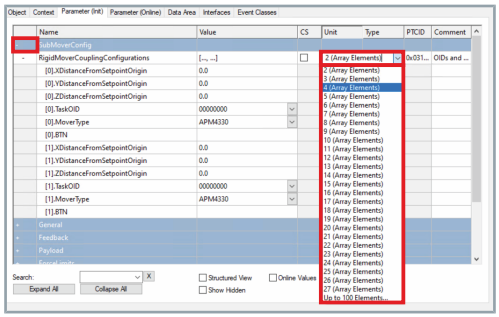


- ► Expand *Solution Explorer* > *System* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)*
- ► Double-click on *Object (XPlanarRigidMoverCoupling)*



- ► Click on the **Context** tab in the project window
- ► In the drop-down menu, select a task to be used as the primary task for all objects

The selected task must be used for at least one submover.

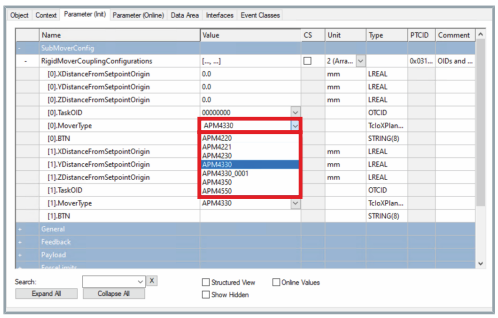


- ► Click the **Parameter (Init)** tab in the project window
- ► Expand *SubMoverConfig*
- ► Select the number of submovers in the *RigidMoverCouplingConfigurations* drop-down menu

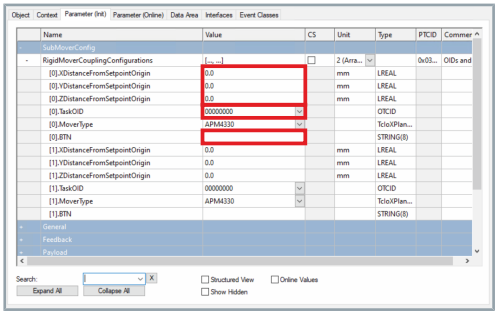
The following input fields are created for each submover of a mover coupling:

- XDistanceFromSetpointOrigin
- YDistanceFromSetpointOrigin
- ZDistanceFromSetpointOrigin
- TaskOID
- MoverType
- BTN

Only one mover type possible
Only one mover type can be chosen for each mover coupling. The choice of different mover types in a mover coupling is not permitted.



► Select MoverType from the *MoverType* drop-down menu



► Enter values for the position of the submover to the setpoint origin in the *XDistanceFromSetpointOrigin*, *YDistanceFromSetpointOrigin* and *ZDistanceFromSetpointOrigin* input fields

Further information can be found in chapter "Setpoint origin", [Page 69].

► Select a task from the *TaskOID* drop-down menu

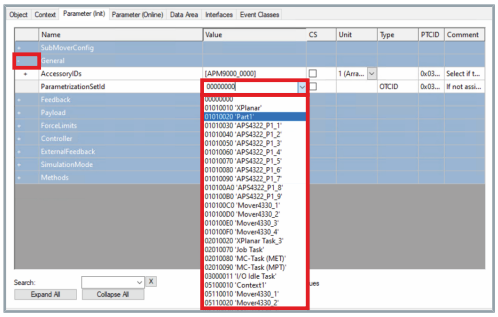
At least one submover must use the same task that was chosen for the coupling.

► If required, enter the submover's BTN in the *BTN* input field

Further information on the BTN can be found in the XPlanar original operating instructions or at:

www.beckhoff.com/bic

► Enter values for all submovers in the same way



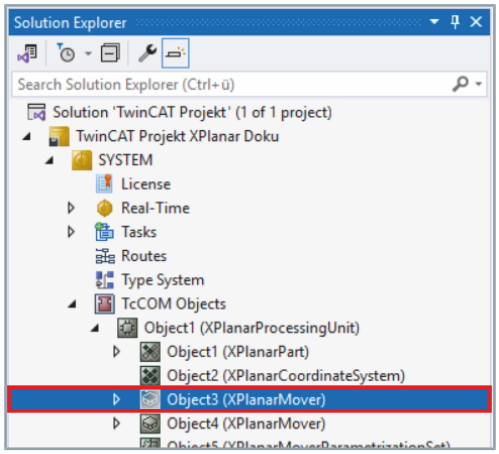
► Expand *General*

► If required, select a parametrization set from the *ParametrizationSetId* drop-down menu

The mover coupling is active as soon as all movers are found at a defined distance from each other during Mover Detection.

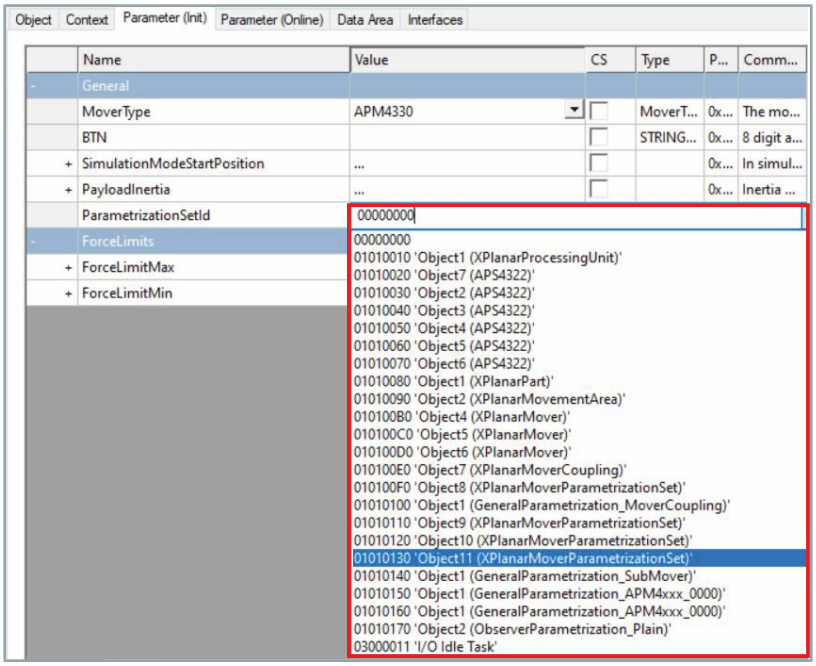
13 Selecting a Parametrization Set

Parametrization sets can be used for single or multiple movers as well as for mover couplings. Further information on the parametrization sets can be found in chapter "Mover Parametrizations", [Page 76].



► Expand *Solution-Explorer* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)*

► Double-click on **Object (XPlanarMover)** or on **Object (XPlanar-RigidMoverCoupling)**



► Click the **Parameter (Init)** tab in the project window

► Select the parametrization set in the **ParametrizationSetId** drop-down menu

14 Mover Parametrizations

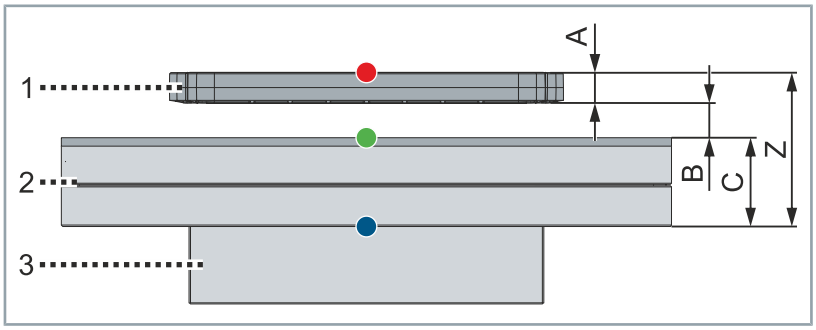
This chapter explains the functions of the different parametrizations. The parametrizations are defined in parametrization sets and can be applied to movers and mover couplings.

Further information can be found in chapters "Mover parametrization", [Page 36], "Selecting a Parametrization Set", [Page 75] and Parameter.

14.1 General Parametrization

General parametrization is used to define the flight characteristics of the movers and mover couplings.

14.1.1 Flight height



Position	Explanation
1	Mover
2	Tile
3	Cover
●	Mover surface: feedback and setpoint origin of the mover
●	Top of the tile
●	Bottom of the tile

The flight height is the sum of the height of the mover, height at which the mover is flying, and the height of the tile:

$$A + B + C = Z$$

Variable	Explanation
A	Height of the mover
B	Flight height of the mover
C	Tile height: base profile and stator
Z	Setpoint Z-position of the mover: distance from the bottom of the tile to the surface of the mover

Different calculations are made depending on the reference type. The decisive factor is whether a fixed flight height is to be defined for all movers or whether a common surface is to be achieved for all movers, regardless of the height of the mover.

14.1.2 Without common reference surface

With this parameter setting, the movers have the same flight height, regardless of the height of the movers. The setpoint origin, which is located on the mover surface, is not at the same height for movers of different heights. The following formula can be used to calculate the Z-position setpoints for the movers:

$$Z_x = A_x + B + C$$

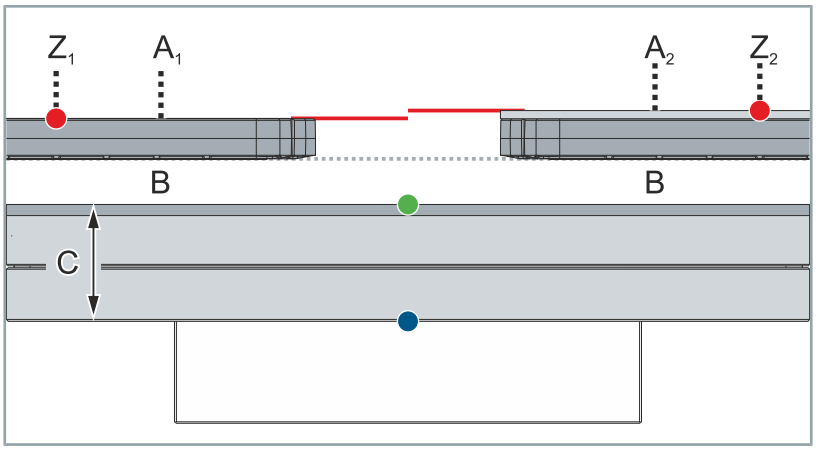
In the following example, the setpoint Z-positions are calculated for two movers of different heights. Mover A_1 is 12.1 mm high and mover A_2 is 13.8 mm high. Tile C is 35.2 mm high and the flight height B is 2 mm.

	A_x	B	C_x	Z_x
Height [mm]	12.1	2	35.2*	49.3*
Height [mm]	12.1	2	48.8**	62.9**
Height [mm]	13.8	2	35.2*	51*
Height [mm]	13.8	2	48.8**	64.6**

* for APM4322-0000

** for APM42xx-1x00

If no common reference surface is specified for the reference Z-position, the mover surfaces are at different heights. The height difference between the mover surfaces and the setpoints for the Z-position of the two movers is 1.7 mm.



14.1.3 With common reference surface

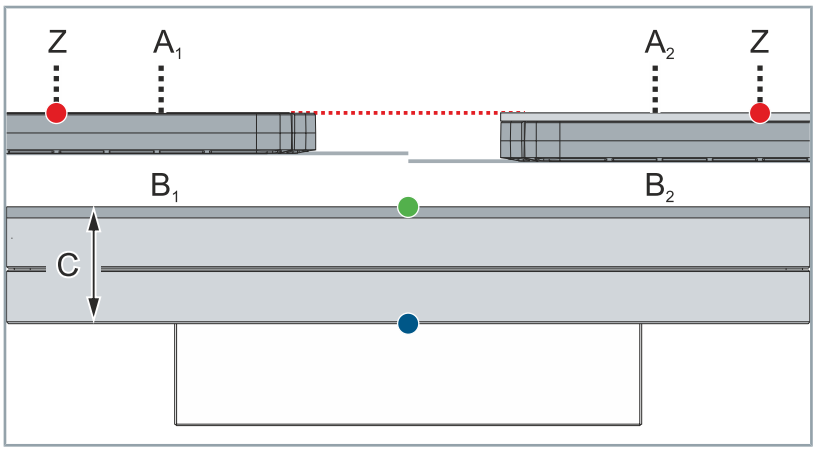
With this parameter setting, the movers have a common setpoint for the Z-position, regardless of the height of the movers. The setpoint origin, which is located on the mover surface, is at the same height for movers of different heights. The reference surface for the setpoint of the Z-position is the bottom of the tile. The following formula can be used to calculate the flight heights for the movers:

$$Z = A_x + B_x + C \longrightarrow B_x = Z - A_x - C$$

In the following example, the flight heights are calculated for two movers of different heights. Mover A_1 is 12.1 mm high and mover A_2 is 13.8 mm high. Tile C is 35.2 mm high and the setpoint Z-position is 50 mm.

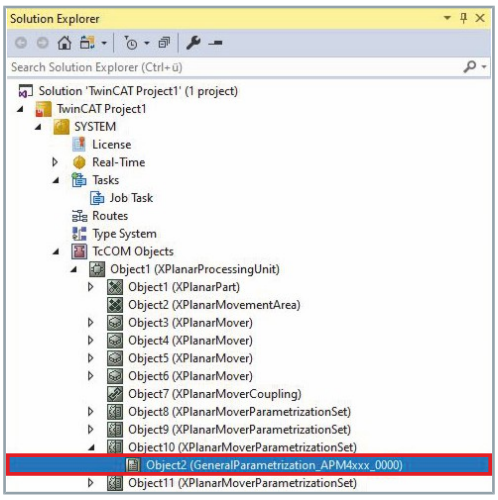
	A_x	C	Z	B_x
Height [mm]	12.1	35.2	50	2.7
Height [mm]	13.8	35.2	50	1

If a common reference surface is specified for the reference Z-position, the mover surfaces are at the same height. The flight heights of the two movers differ by 1.7 mm in order to reach the common reference Z-position.

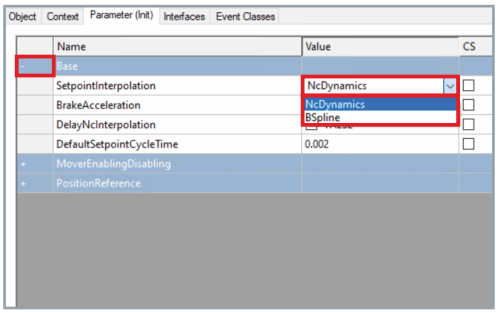


14.1.4 Defining parameters

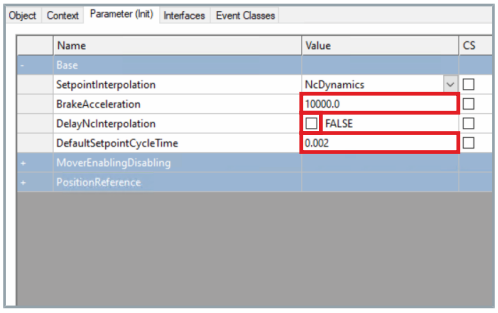
14.1.4.1 APM4xxx-0000



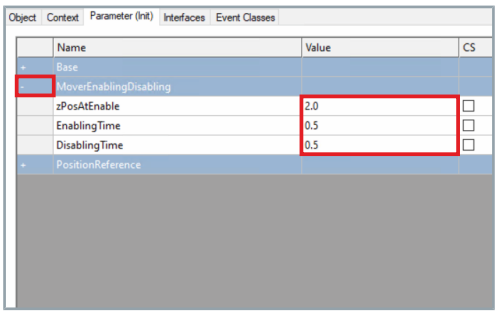
- ▶ Expand *Solution Explorer* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (XPlanarMoverParametrizationSet)*
- ▶ Double-click on **Object (GeneralParametrization)**



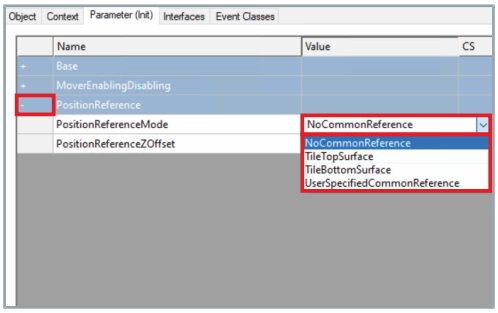
- ▶ Click the **Parameter (Init)** tab in the project window
- ▶ Expand *Base*
- ▶ Select the type of setpoint interpolation in the **SetpointInterpolation** drop-down menu



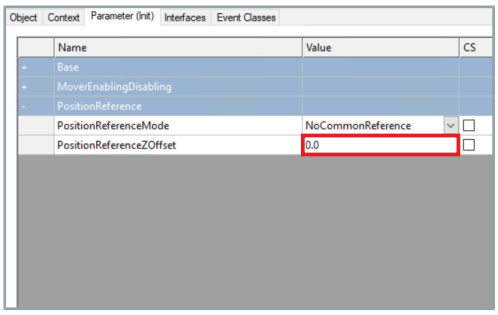
- ▶ Change values for **BrakeAcceleration** and **DefaultSetpointCycleTime** if required
- ▶ Select the **DelayNcInterpolation** checkbox to choose *FALSE* if the NC interpolation is to be delayed by one NC cycle



- ▶ Change values for **zPosAtEnable**, **EnablingTime** and **DisablingTime** if required

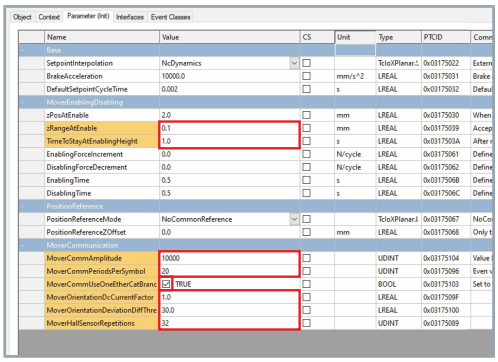


- ▶ Select the Z reference position in the **PositionReferenceMode** drop-down menu



► Enter value for **PositionReferenceZOffset**

► Activate the **Show Hidden Parameter** checkbox to display the *Hidden Parameters*



► Change value for **zRangeAtenable** and **TimeToStayAtEnablingHeight** if required

► Change *value for* **MoverCommAmplitude** and **MoverCommPeriodsPerSymbol** if required

► *Select the* **MoverCommUseOneEtherCatBranch** checkbox to choose *TRUE* if no synchronized EtherCAT ports are available

OR

► Select the **MoverCommUseOneEtherCatBranch** checkbox to choose *FALSE* if synchronized EtherCAT ports are available

► Change value for **MoverOrientationDcCurrentFactor, MoverOrientationDeviationDiffThreshold** and **MoverHallSensorRepetition** if required

14.2 Observer Parametrization

The Observer parametrizations are entered for the X-axis and Y-axis for movement, for the Z-axis for lifting and lowering, for the A-axis and B-axis for tilting and for the C-axis for rotation.

Four different observer parametrizations are available:

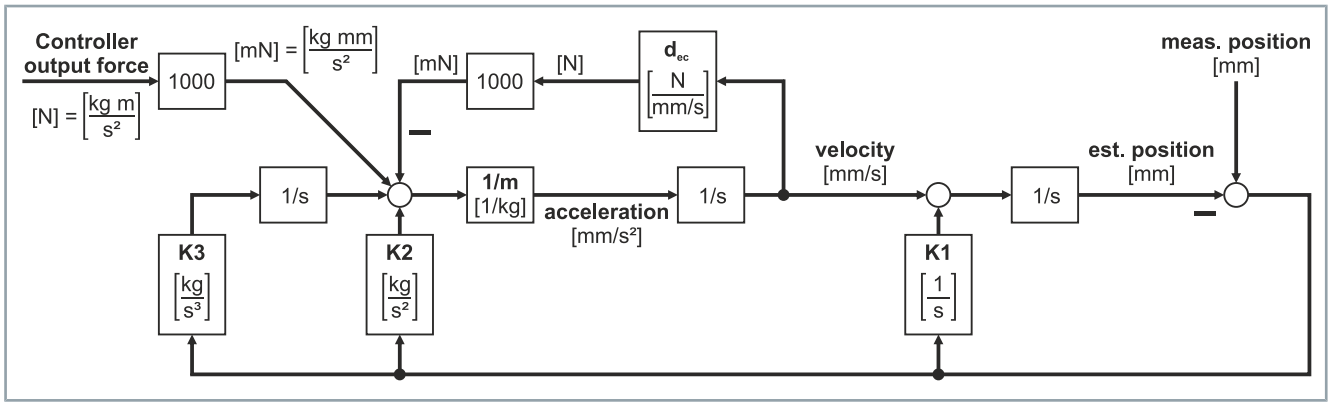
- Plain
- APM4220
- APM4330
- APM4550

All Observer parametrizations have the same functions and the same parameters. Plain parametrization can be used for tuning for a mover coupling. The *APM4xxx* parametrizations have predefined values that are pre-set specifically for the mover type and should be used according to the mover type. The pre-set values serve as a guide and can be adjusted according to your application.

14.2.1 Linear axes observer

The parameters K1, K2, K3 and d_{ec} in the X-axis, Y-axis and Z-axis can be adjusted using an Observer parametrization set.

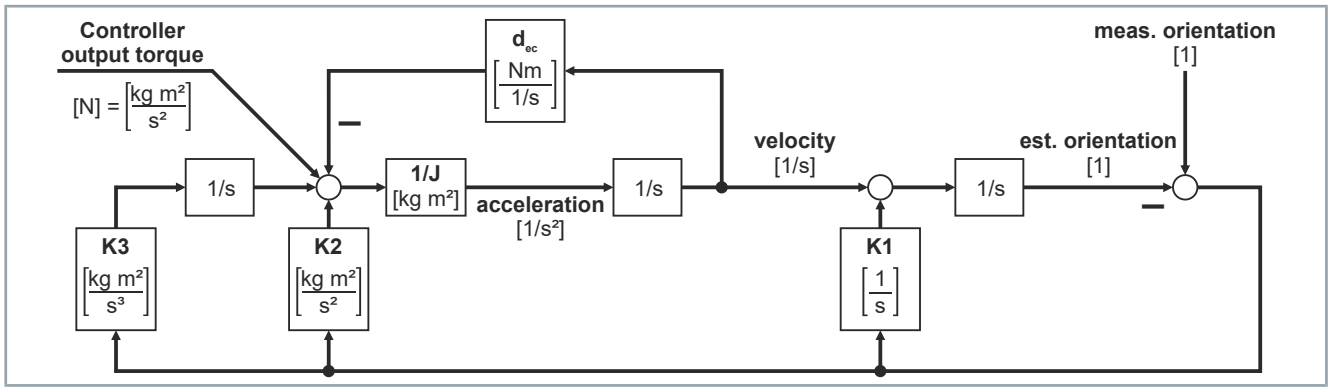
The mass (m) is adjusted depending on the PayloadInertia parameter of a mover object or by the BasicInertia parameter of an Inertia parametrization set.



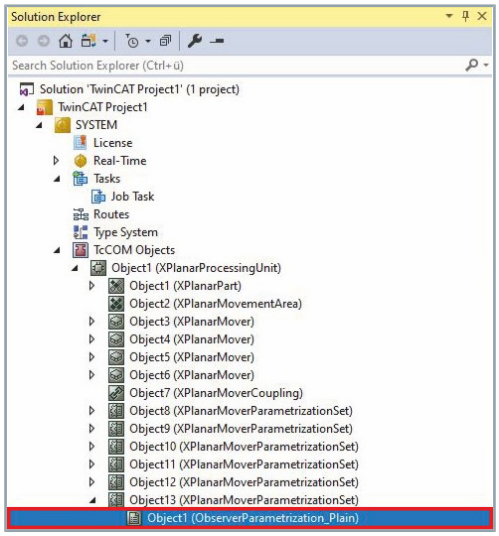
14.2.2 Rotational axes observer

The parameters K1, K2, K3 and d_{ec} in the A-axis, B-axis and C-axis can be adjusted using an Observer parametrization set.

The inertia (J) is adjusted depending on the PayloadInertia parameter of a mover object or by the BasicInertia parameter of an inertia parametrization set.

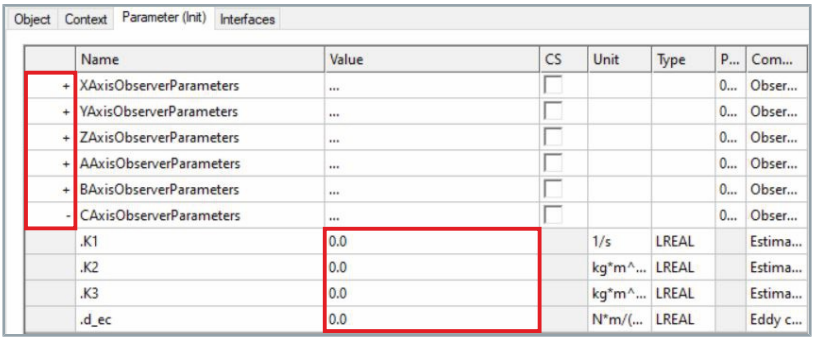


14.2.3 Defining parameters



- ► Expand *Solution Explorer* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (XPlanarMoverParametrizationSet)*
- ► Double-click on **Object (ObserverParametrization_...)**

Pre-set values
The Observer parametrizations for the *APM4220-0000*, *APM4330-0000* and *APM4550-0000* movers have pre-set values for the parameters *K1*, *K2*, *K3* and *d_ec*. If necessary, you can adjust the mover-specific values; they serve as a guide when determining your values. The *Plain* parametrization set has no pre-set values and can be used for a mover coupling.



- ► Click the **Parameter (Init)** tab in the project window
- ► Click on **+** of *XAxisObserverParameters*, *YAxisObserverParameters*, *ZAxisObserverParameters*, *AAxisObserverParameters*, *BAxisObserverParameters* and *CAxisObserverParameters* to call up the input fields for the filter parameters of the axes
- ► Enter or change values for **K1**, **K2**, **K3** and **d_ec**

14.3 Controller Parametrization

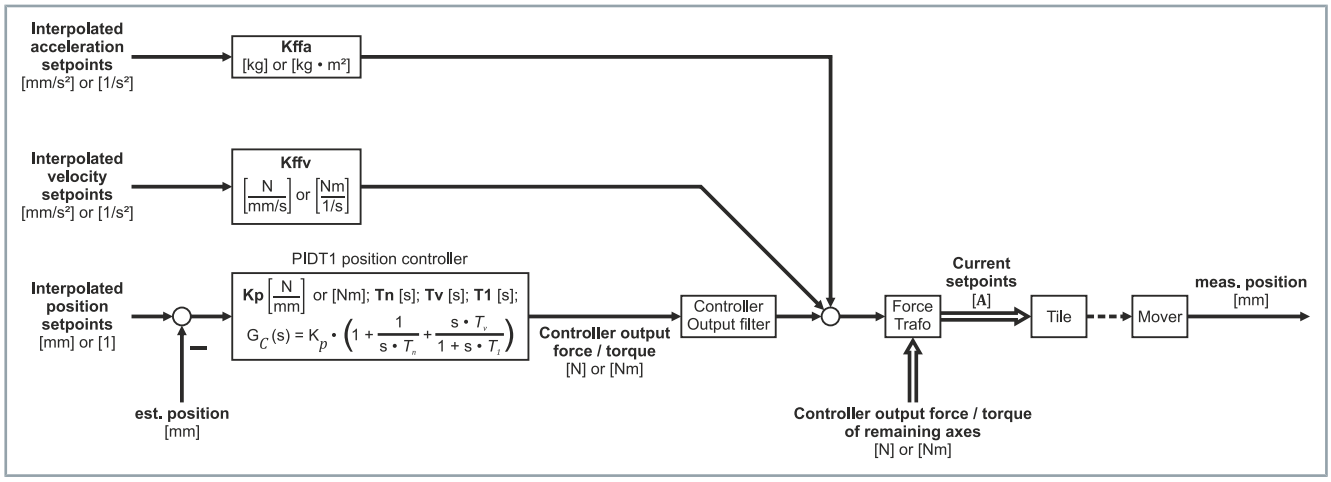
Four different control parametrizations are available:

- Plain
- APM4220
- APM4330
- APM4550

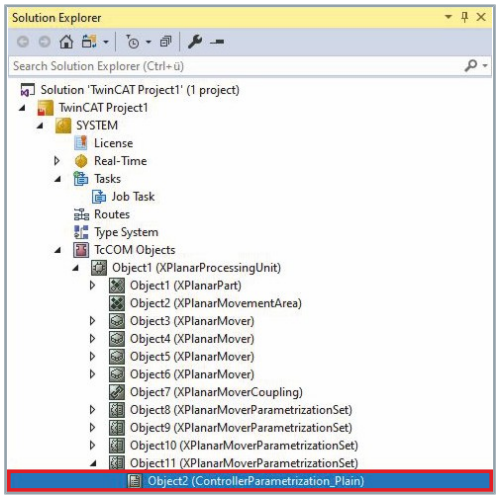
All control parametrizations have the same parameters. Plain parametrization can be used for tuning for a mover coupling. The *APM4xxx-000x* parametrizations have predefined values that are pre-set specifically for the mover type and should be used according to the mover type. The pre-set values serve as a guide and can be adjusted according to your application.

You can define the parameters for the controllers *Kp*, *Tn*, *Tv* and *T1* as well as the FeedForward parameters *Kffa* and *Kffv*.

If the *AutoAdjust* controller is activated, the *Kffa* parameter is adjusted depending on the defined payload. If the *AutoAdjust* controller is activated, the *Kp* parameter is adjusted depending on the defined payload.

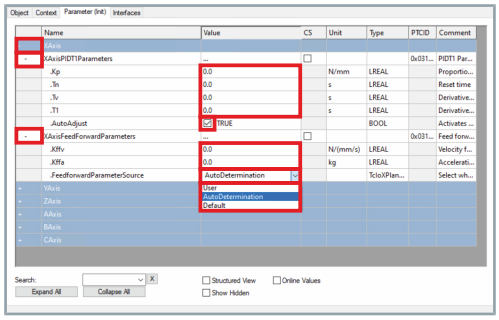


14.3.1 Defining parameters



- ▶ Expand *Solution Explorer* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (XPlanarMoverParametrizationSet)*
- ▶ Double-click on **Object (ControllerParametrization_...)**
- ▶ Click the **Parameter (Init)** tab in the project window

Pre-set values
The control parametrizations for the *APM4220-0000*, *APM4330-0000* and *APM4550-0000* movers have pre-set values for the parameters *Kp*, *Tn*, *Tv*, *T1*, *Kffa* and *Kffv*. If necessary, you can adjust the mover-specific values; they serve as a guide when determining your values. The *Plain* parametrization set has no pre-set values and can be used for a mover coupling.



- ▶ Click on **+** of XAxis, YAxis, ZAxis, AAxis, BAxis and CAxis respectively
- ▶ Expand *XAxisPIDT1Parameters*, *YAxisPIDT1Parameters*, *ZAxisPIDT1Parameters*, *AAxisPIDT1Parameters*, *BAxisPIDT1Parameters* and *CAxisPIDT1Parameters* by clicking the **+** icons
- ▶ Enter values for **Kp**, **Tn**, **Tv** and **T1** or change them if necessary
- ▶ Select the **FALSE** checkbox of *AutoAdjust* to activate AutoAdjust
- ▶ Expand *XAxisFeedForwardParameters*, *YAxisFeedForwardParameters*, *ZAxisFeedForwardParameters*, *AAxisFeedForwardParameters*, *BAxisFeedForwardParameters* and *CAxisFeedForwardParameters* by clicking on the **+** buttons
- ▶ Enter values for **Kffv** and **Kffa** or change them if necessary
- ▶ Select **AutoDetermination** in the *FeedForwardParameterSource* drop-down menu

14.4 Filter Parametrization

The frequency *fd* must be entered in the unit Hz.

The *FilterParametrization_APM4xxx_0000* can be used for every mover type and mover coupling. There are four filter types to choose from:

14.4.1 Low Pass 1

A value must be entered for the *fd* parameter for the *1st order low-pass filter*. The other parameters do not require any input. The *1st order low-pass filter* is calculated using the following formula:

$$G_f(s) = \frac{1}{1 + s \cdot \tau_d} \quad \text{with:} \quad \tau = \frac{1}{2\pi \cdot f}$$

14.4.2 Low Pass 2

Values for the *fd* and *Dd* parameters must be entered for the *2nd order low-pass filter*. The other parameters do not require any input. The *2nd order low-pass filter* is calculated using the following formula:

$$G_f(s) = \frac{1}{\tau_d^2 \cdot s^2 + 2D_d \tau_d \cdot s + 1} \quad \text{with:} \quad \tau = \frac{1}{2\pi \cdot f}$$

14.4.3 Notch

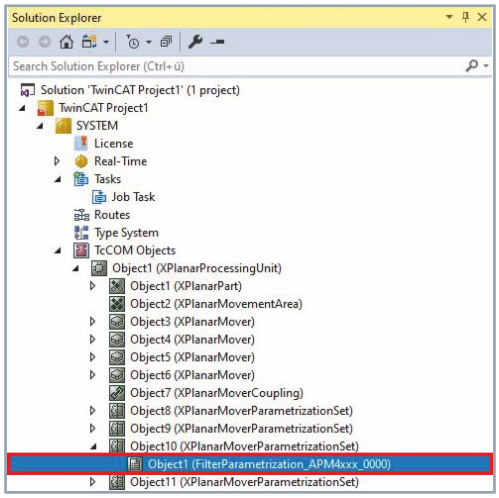
Values must be entered for all parameters for the notch filter. The notch filter is calculated using the following formula:

$$G_f(s) = \frac{s^2\tau_n^2 + 2D_n\tau_n \cdot s + 1}{s^2\tau_d^2 + 2D_d\tau_d \cdot s + 1} \quad \text{with:} \quad \tau = \frac{1}{2\pi \cdot f}$$

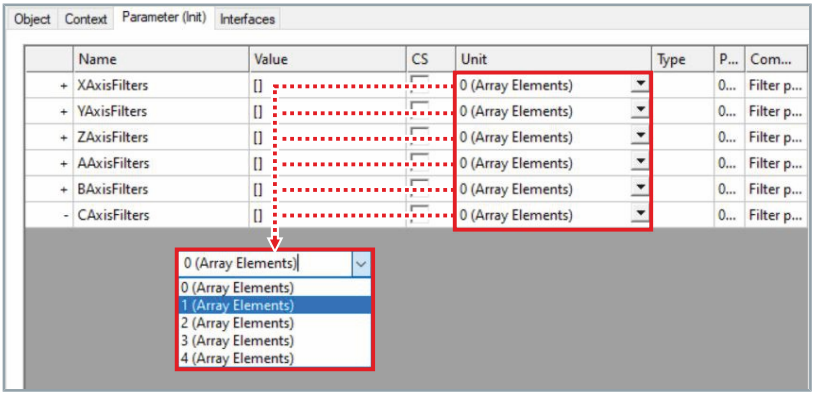
14.4.4 Bypassed

No values need to be entered for the bypassed filter. If this filter type is selected, the filter is inactive.

14.4.5 Defining parameters



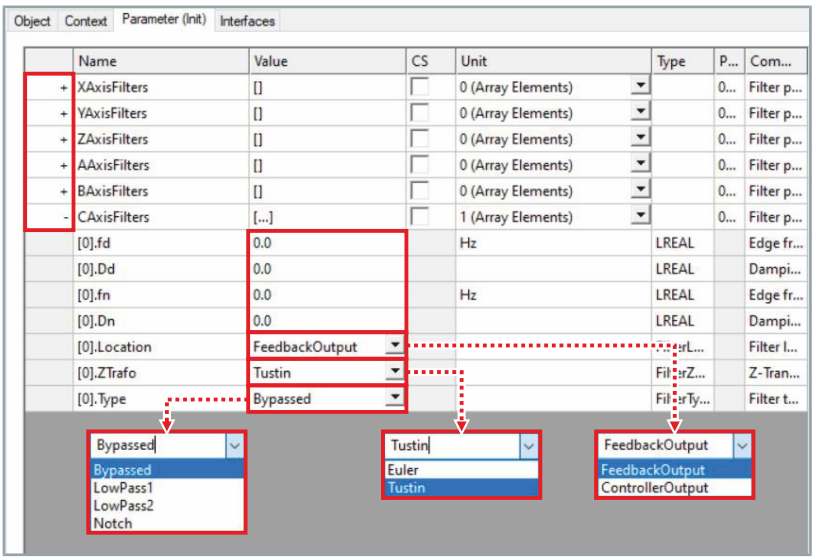
- ▶ Expand *Solution Explorer* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (XPlanarMoverParametrizationSet)*
- ▶ Double-click on **Object (FilterParametrization_APM-4xxx_0000)**
- ▶ Click the **Parameter (Init)** tab in the project window



► Select the number of **Array Elements** for the axes in the drop-down menu

For each array element, four input fields and three drop-down menus are added to an axis.

At least one array element required
At least one array element must have been selected from the axis drop-down menu before values can be entered into an input field or the filter type selected from the drop-down menu. Without an array element, no input fields and drop-down menus are available for input.



► Click on **+** of an *AxisFilter* to call up the input fields for the filter

► Select **Location**, **ZTrafo** and **Type** from the drop-down menus

► Enter the required values for **fd**, **Dd**, **fn** and **Dn**, depending on the selected filter type

14.5 Inertia Parametrization

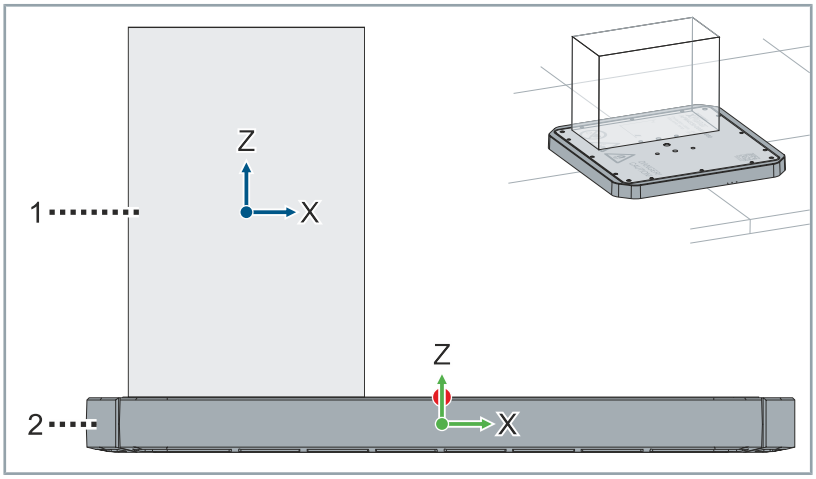
The inertia parametrization is defined as the total inertia for the following objects:

- Mover with mounted tool
- Mover coupling, all submovers with frame and any tools mounted
- Mover with a constant payload

If no inertia parametrization is assigned or this parametrization has a value of zero, the default inertia parameters of the respective mover type are used.

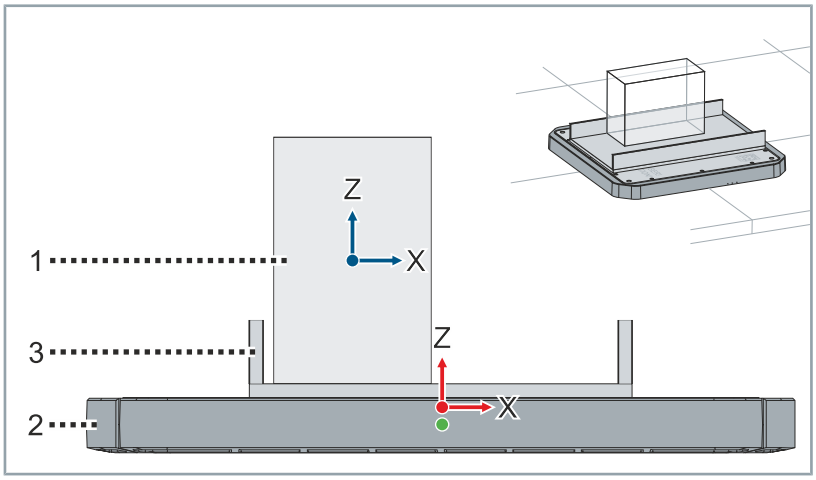
The inertia of the payload is combined with the inertia of the mover. The center of mass of the payload is specified in relation to the setpoint origin of the mover. The setpoint origin of a standard mover is located in the center of the mover's surface. The axes used to define the inertia must be parallel to the coordinate axes of the mover.

14.5.1 Mover



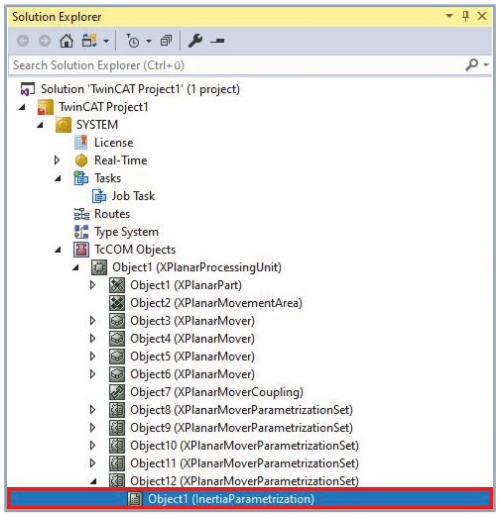
Position	Explanation
1	Payload
2	Mover
●	Setpoint origin of the mover
●	Center of mass of the mover
●	Center of mass of the payload

14.5.2 Mover and tool mounted

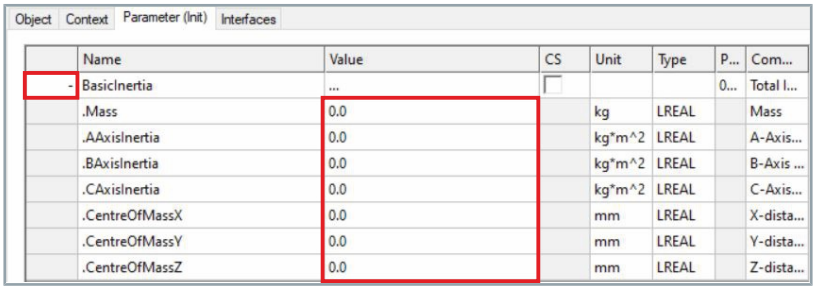


Position	Explanation
1	Payload
2	Mover
3	Tool mounted
●	Center of mass of the basic inertia
●	Center of mass of the mover
●	Center of mass of the payload

14.5.3 Defining parameters



- ▶ Expand *Solution Explorer* > *TcCOM Objects* > *Object (XPlanarProcessingUnit)* > *Object (XPlanarMoverParametrizationSet)*
- ▶ Double-click on **Object (InertiaParametrization)**



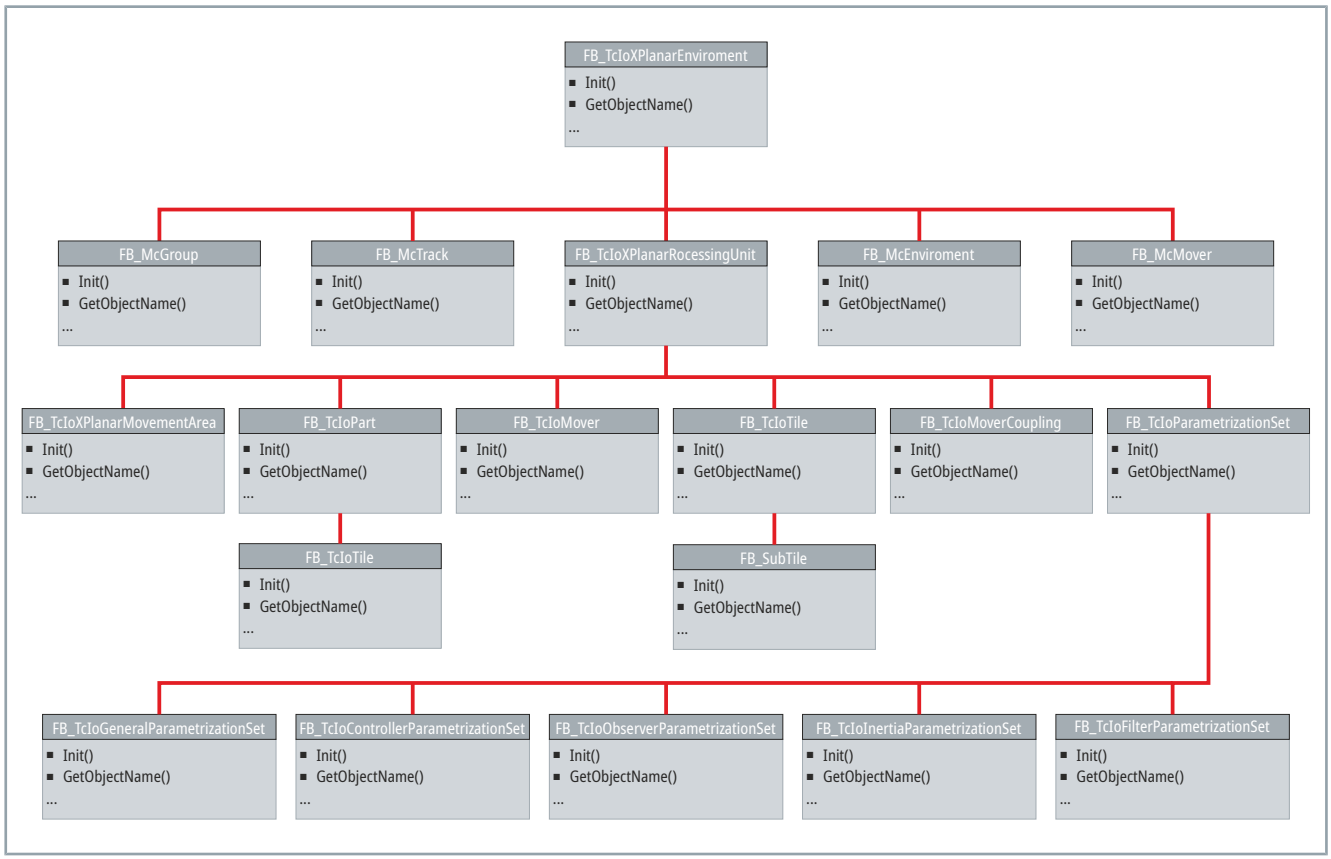
- ▶ Click the **Parameter (Init)** tab in the project window
- ▶ Click on **+** of *BasicInertia* to call up the input fields for the payload parameters
- ▶ Enter the weight of the payload in the **Dimensions** field
- ▶ Enter the inertia parameters of the payload via the **AAxisInertia**, **BAxisInertia** and **CAxisInertia** fields
- ▶ Enter the position of the payload's center of mass using the **CentreOfMassX**, **CentreOfMassY** and **CentreOfMassZ** fields

15 PLC libraries

15.1 TwinCAT 3 XPlanar Utility

The TwinCAT 3 XPlanar Utility is installed together with the TF5890 software package. The TwinCAT 3 XPlanar Utility is a PLC library and is used to read or set parameters of the XPlanar system.

The following diagram shows the structure of the TwinCAT 3 XPlanar Utility:



15.1.1 Check version

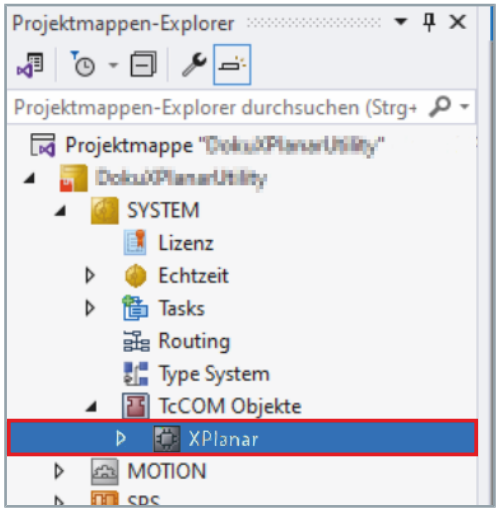
To ensure correct functioning, make sure that the major version and minor version of the Tc3_XPlanarUtility and the XPlanar driver match. The table shows an example of a compatible configuration of Tc3_XPlanarUtility and XPlanar driver:

Product	Major	Minor	Revision	Build
Tc3_XPlanarUtility	4	5	13	0
XPlanar TcCOM Objects XPlanar driver	4	5	31	0

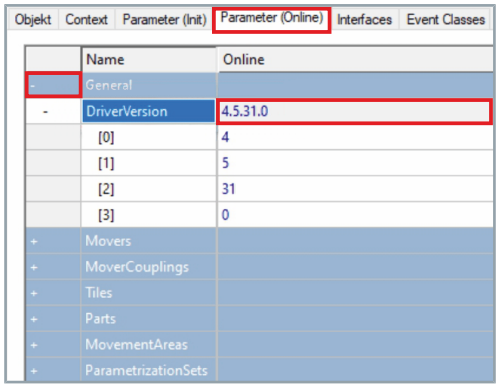
15.1.1.1 TcCOM objects

You have two options for checking the version of the TcCOM objects:

Parameter Online

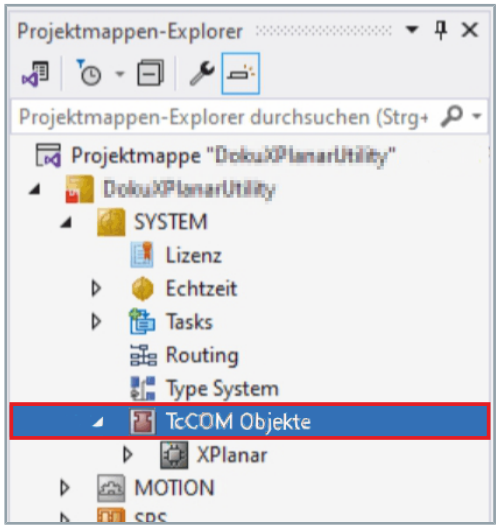


- ► Expand *Solution Explorer* > *TwinCAT Project* > *SYSTEM* > *TcCOM Objects*
- ► Double-click on **XPlanar**

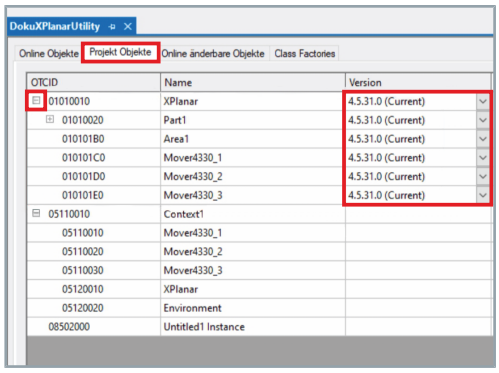


- ► Click on the **Parameter (Online)** tab in the project window
- ► Expand *General*
- ► Check the *Driver Version*

Project Objects

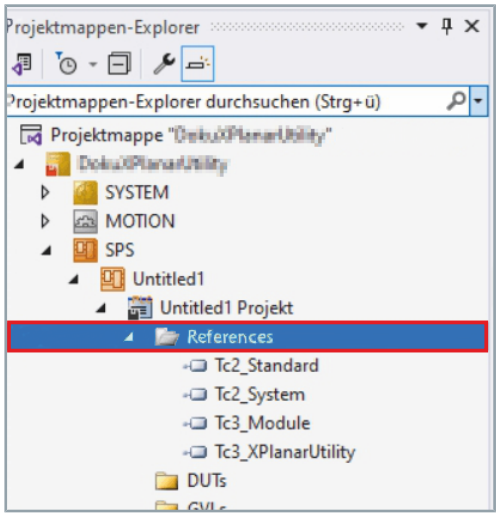


- ► Expand *Solution Explorer* > *TwinCAT Project* > *SYSTEM*
- ► Double click on **TcCOM Objects**

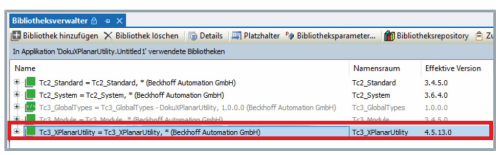


- ► Click on the **Projects (Online)** tab in the project window
- ► Expand *01010010*
- ► Check version

15.1.1.2 Tc3_XPlanarUtility



- ► Expand *Solution Explorer* > *TwinCAT Project* > *SPS* > *Untitled* > *Untitled Project*
- ► Double-click on **References**



- ► Check in the Library Manager which version of Tc3_XPlanarUtility is being used

15.1.2 Initialization

The *FB_TcIoXPlanarEnvironment* function block must be instantiated once for the XPlanar project so that all parameters of the Processing Unit can be accessed. If visualization is to be used in the project, the *FB_XPlanarUtilityVisu* function block must also be configured.

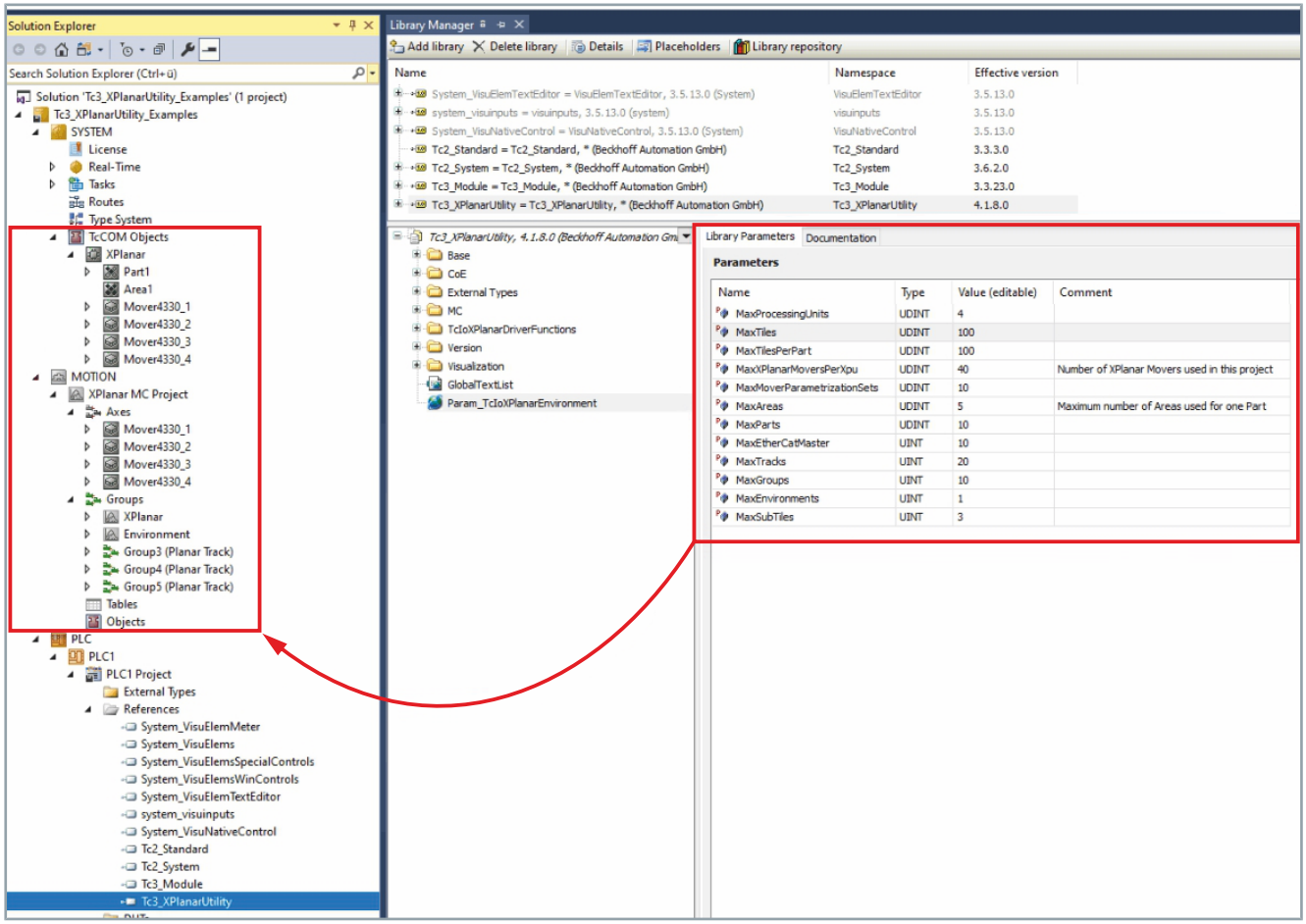
The *FB_TcIoXPlanarEnvironment* function block must be initialized once at startup to collect information about all objects. Optionally, the status bits can be checked to obtain feedback from the function block. After initialization, all objects subordinate to the *XPlanarEnvironment* can be accessed by calling the corresponding methods. There is no cyclic updating of the data.

In addition to using the *FB_TcIoXPlanarEnvironment* function block, you also have the option of using the *Tc3_XPlanarUtility* function blocks as a standalone variant. Each function block has its own initialization method. You can use this method to initialize the function block and use the corresponding function block as a standalone variant.

If *FB_TcIoXPlanarEnvironment* has been assigned the *Environment* property of *FB_XPlanarUtilityVisu*, the *Cycle()* method must be called cyclically.

15.1.2.1 Sample code

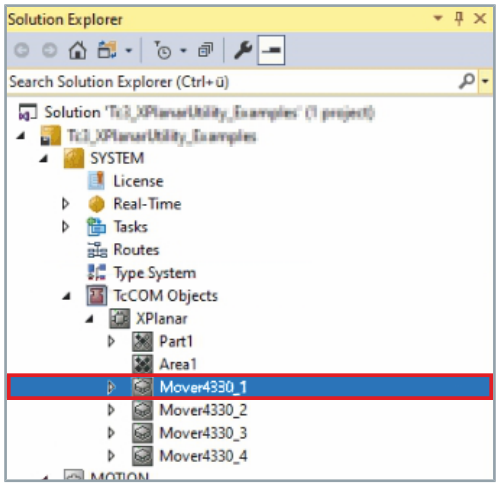
```
VAR
     nStateInitEnvironment     : INT;
     fbXplanarEnvironment      : FB_TcIoXPlanarEnvironment;
     fbVisuXPlanarDiag         : FB_XPlanarUtilityVisu;
     ipXpu                     : I_TcIoXplanarProcessingUnit;
END_VAR
CASE nStateInitEnvironment OF
0: // Init Xplanar Environment
     IF fbXplanarEnvironment.Init(TRUE) THEN
          fbXplanarEnvironment.Init(FALSE);
          ipXpu := fbXplanarEnvironment.XpuTcIo(1);
          nStateInitEnvironment := nStateInitEnvironment + 1;
     END_IF

1: // Init Visu
     fbVisuXPlanarDiag.Environment := fbXplanarEnvironment;
     nStateInitEnvironment := nStateInitEnvironment + 1;

ELSE
     fbVisuXPlanarDiag.Cycle();
END_CASE
```



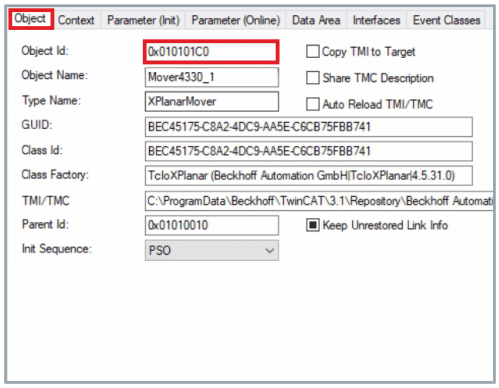
The *Param_TcIoXPlanarEnvironment* parameter list must be adapted to the configuration. Each individual parameter must be greater than or at least equal to the configuration to ensure that the initialization of *FB_TcIoXPlanarEnvironment* can be completed successfully.

15.1.2.2 Initialization of standalone function blocks

If *FB_TcIoXPlanarEnvironment* is not to be used, each subordinate function block can be initialized individually. The OTCID of the corresponding TcCOM object is required for this.



- ► Expand *Solution Explorer* > *SYSTEM* > *TcCOM Objects* > *XPlanar*
- ► Double-click on a **Mover4330**



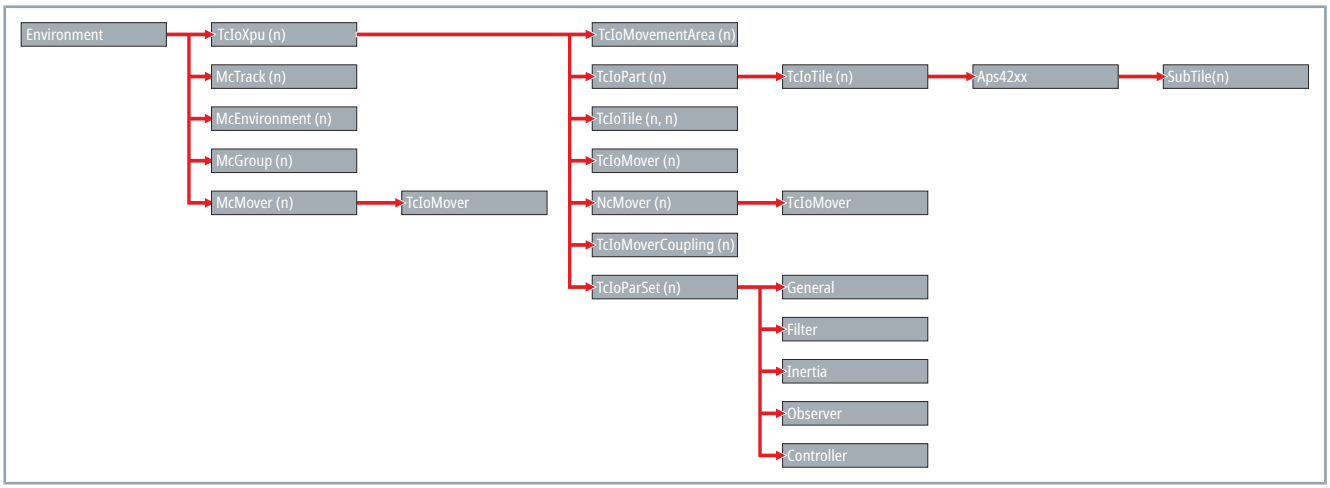
- ► Click on the **Object** tab in the project window
- ► Read value at *Object Id*

```
VAR
nState                      : UDINT;
fbTcIoMoverStandalone       : FB_TcIoXPlanarMover;
END_VAR

CASE nState OF

130: // Init stadalone fuction block
     IF fbTcIoMoverStandalone.Init(TRUE,16#010101C0, 1) THEN
          nState := nState + 1;
     END_IF

END_CASE
```

15.1.3 PLC access

15.1.3.1 Call Chain

The following call chain allows you to access the essential parameters of the XPlanar objects. An overview of the parameters can be found in chapter "Parameters", [Page 106].



15.1.3.2 Access to parameters in the PLC

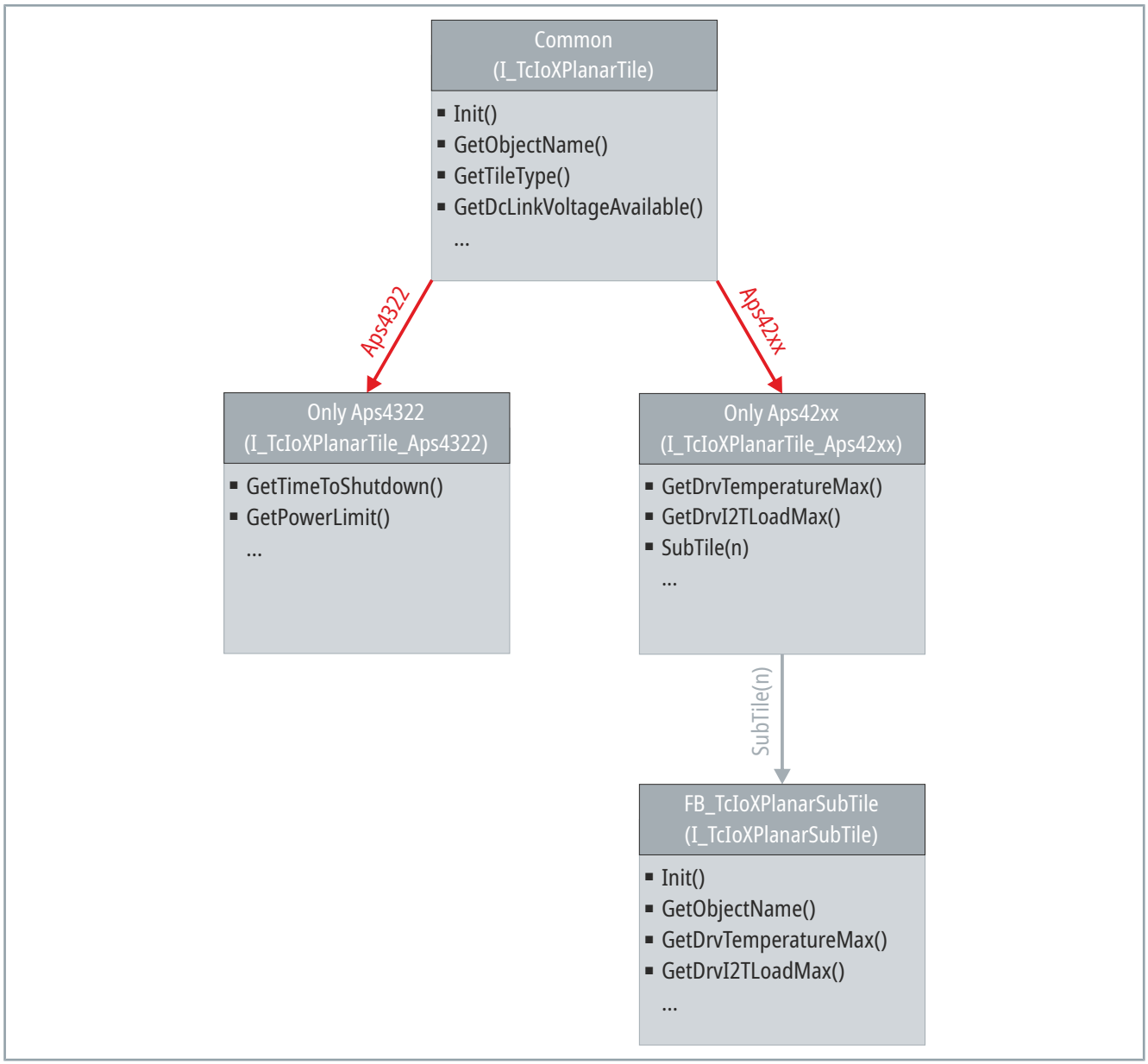
Similar to the Call Chain, you can use the *subitem* methods to navigate to the desired object and parameter. The method input of the *subitem* methods can be interpreted as an index, a consecutive number from 1 to X, or as the OTCID of the object. The following sample shows how to access the TcCOM object of a mover:

```
120: // Read object name of mover 1 by index
     sObjectNameIdx := ipXpu.TcIoMover(1).GetObjectName();
     nState := nState + 1;

121: // Read object name of mover 1 by OTCID
     sObjectNameOid := ipXpu.TcIoMover(16#010101C0).GetObjectName();
     nState := nState + 1;
```

Both calls return *Mover4330_1*, as they access the same TcCOM object.

Additional notes for FB-TcIoXPlanarTile

The different structure of the AP4322 and APS42xx tiles requires different access to the respective TcCOM parameters. The *FB_TcIoXPlanarTile* function block is divided into a section for APS4322 and a section for APS42xx, in which the corresponding interfaces are implemented.



Symbol	Explanation	Symbol	Explanation
↓	Access via properties	↓	Access via methods

A significant difference between the *APS4322* and *APS42xx* tiles is that the *APS42xx* tile can have up to three sub-tiles: *FB_TcIoXPlanarSubTile*.

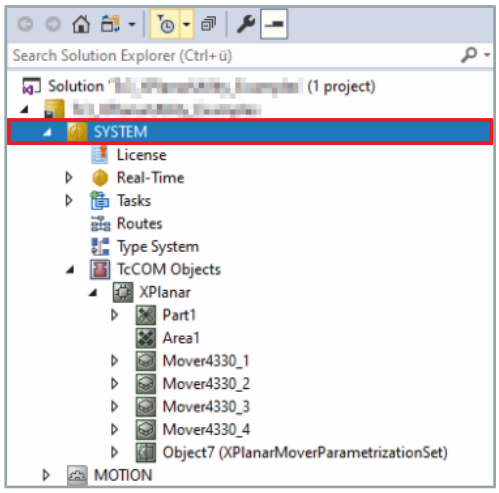
Access to TcIo parameters

In general, all TcIo parameters on the TcCOM object server can be accessed using the PLC. The following sample shows how to read/ write the *MoverParametrizationSet* OTCID of a mover. The PLC code shown represents the read access and write access to this TcIo parameter.

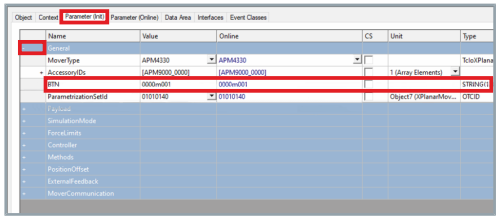
This principle works in the same way for all methods and parameters, according to the Call Chain of the *Tc3_XPlanarUtility*.

```
10: // Read object id of the parametrization set of mover 1
     nParaSetOid := ipXpu.TcIoMover(1).GetParameterizationSetId();
     nState := nState + 1;

11: // Write object id of a parametrization set to mover 2
     IF ipXpu.TcIoMover(2).SetParameterizationSetId(nParaSetOid) THEN
          nState := nState + 1;
     END_IF
```

Example Mover Parameter Set
Access to TcIo parameters is shown as an example by reading the MoverParametrizationSet OTCID of a mover. Calling the Init parameters is done in the same way for all TcIo parameters.



► Expand *Solution Explorer*
► Double-click on **SYSTEM**



► Click on the **Parameter (Init)** tab
► Expand *General*
► Read the value for *ParametrizationSetId*

Access to scope parameters

Using *Tc3_XPLanarUtility* allows access to all scope parameters of the XPlanar. All scope parameters are called in the same way.

```
VAR
     stOutputForce:MoverVector;
END_VAR
```

```
20: // Read output force of mover 1 from scope data
     stOutputForce := ipXpu.TcIoMover(1).ScopeAdr.GetOutputForce()^;
     nState := nState + 1;
```

15.1.3.3 Samples of frequently used methods

You need these calls to access various parameters in the PLC:

Start Mover Redetection (Mover Detection)

```
0: //Start mover redetection
     ipXpu.SetStartMoverRedetection();
     nState := nState +1;
```

Get Mover Detection Status (Mover Detection)

```
VAR
     stMoverDetectionStatus:MoverDetectionStatus;
END_VAR
```

```
50: // Read mover detection status
     stMoverDetectionStatus := ipXpu.GetMoverDetectionStatus();
     IF stMoverDetectionStatus = MoverDetectionStatus.DetectionSuccessful THEN
          nState := nState + 1;
     END_IF
```

Get Mover Detection Infos (Mover Detection)

```
VAR
     stMoverDetectionInfo:ST_TcIoMoverDetectionInfos;
END_VAR
```

```
60: // Read mover detection infos
     stMoverDetectionInfo := ipXpu.GetMoverDetectionInfos();
     IF stMoverDetectionInfo.TotalDetected = stMoverDetectionInfo.TotalDetected
          AND stMoverDetectionInfo.APM4330Detected = stMoverDetectionInfo.APM4330Expected THEN
          nState := nState + 1;
     END_IF
```

Mover Detection Setup (Mover Detection)

```
VAR
     arMoverDetectionSetup:ARRAY[0..4] OF ST_TcIoMoverDetectionSetup;
END_VAR
```

```
2: // Adapt mover detection setup
     // Set number of APM4330
     arMoverDetectionSetup[0].MoverType := XPlanarMoverType.APM4330;
     arMoverDetectionSetup[0].ExpectedCount := 4;
     // Set number of APM4220
     arMoverDetectionSetup[1].MoverType := XPlanarMoverType.APM4220;
     arMoverDetectionSetup[1].ExpectedCount := 0;
     // Set number of APM4221
     arMoverDetectionSetup[2].MoverType := XPlanarMoverType.APM4221;
     arMoverDetectionSetup[2].ExpectedCount := 0;
     // Set number of APM4550
     arMoverDetectionSetup[3].MoverType := XPlanarMoverType.APM4550;
     arMoverDetectionSetup[3].ExpectedCount := 0;

     ipXpu.SetMoverDetectionSetup(4,ADR(arMoverDetectionSetup));
     nState := nState - 1;
```

Set CoordMoverCommunicationCmd (Mover Identification)

```
10: // Start coordianted mover communication command
     ipXpu.SetCoordMoverCommunicationCmd(CoordinatedMoverCommCommand.IdentifyAndAssignPossible);
     nState := nState + 1;
```

Get CoordMoverCommunicationStatus (Mover Identification)

```
VAR
     stCoordMoverCommStatus:CoordinatedMoverCommStatus;
END_VAR
```

```
11: // Check for identification in process
     stCoordMoverCommStatus := ipXpu.GetCoordMoverCommStatus();
     IF stCoordMoverCommStatus = CoordinatedMoverCommStatus.CommunicationInProcess THEN
          nState := nState + 1;
     END_IF

12: // Check result of mover identification
     stCoordMoverCommStatus := ipXpu.GetCoordMoverCommStatus();
     IF stCoordMoverCommStatus = CoordinatedMoverCommStatus.CommunicationSuccessful THEN
          nState := 20;
     ELSIF stCoordMoverCommStatus = CoordinatedMoverCommStatus.CommunicationFailed THEN
          // Error: Identification failed
          nState := 9990;
     END_IF
```

Get IdentifiedMoverBtn (Mover Identification)

```
72: // Check IdentifiedMoverBtn
     sBtn := ipXpu.TcIoMover(1).GetIdentifiedMoverBtn();
     nState := nState + 1;
```

Reading out a single ID bumper (Mover Identification)

The code shows an example of how the bumper ID *APM9001-0000-4xxx* of a single mover can be read:

```
70: // Set mover communication command - start reading id-bumper
     IF ipXpu.TcIoMover(1).SetMoverCommunicationCommand(MoverCommunicationCommand.GetMoverBtn) THEN
          nState := nState + 1;
     END_IF

71: // check status
     stMoverCommunicationStatus := ipXpu.TcIoMover(1).GetCommunicationStatus();
     IF stMoverCommunicationStatus = MoverCommunicationStatus.CommunicationSuccessful THEN
          nState := nState + 1;
     END_IF

72: // Check IdentifiedMoverBtn
     sBtn := ipXpu.TcIoMover(1).GetIdentifiedMoverBtn();
     nState := nState + 1;
```

Switch Controllers on/off

```
80: // Switch contoller on/off
     stTcIoControllersOn.x := TRUE;
     stTcIoControllersOn.y := TRUE;
     stTcIoControllersOn.z := FALSE;
     stTcIoControllersOn.a := TRUE;
     stTcIoControllersOn.b := TRUE;
     stTcIoControllersOn.c := TRUE;

     IF ipXpu.TcIoMover(1).SetControllersOn(stTcIoControllersOn) THEN
          nState := nState + 1;
     END_IF
```

Leaving Movement Area

```
90: // Leave MovementArea
     stEnterOrLeaveMovementareaParameters_XYC.MovementAreaOID := 0; // Set to 0 to remove a mover
     stEnterOrLeaveMovementareaParameters_XYC.Positions.x := 0;
     stEnterOrLeaveMovementareaParameters_XYC.Positions.y := 0;
     stEnterOrLeaveMovementareaParameters_XYC.Positions.C := 0;

     IF ipXpu.TcIoMover(1).SetMovementArea_XYC(stEnterOrLeaveMovementareaParameters_XYC) THEN
          nState := nState + 1;
     END_IF
```

Enter Movement Area

```
91: // Enter MovementArea
     stEnterOrLeaveMovementareaParameters_XYC.MovementAreaOID := 16#01010020; // Set to OTCID of Part-
Object
     stEnterOrLeaveMovementareaParameters_XYC.Positions.x := 120; // approximate x-position
     stEnterOrLeaveMovementareaParameters_XYC.Positions.y := 120; // approximate y-position
     stEnterOrLeaveMovementareaParameters_XYC.Positions.C := 0; // approximate c-position

     IF ipXpu.TcIoMover(1).SetMovementArea_XYC(stEnterOrLeaveMovementareaParameters_XYC) THEN
          nState := nState + 1;
     END_IF
```

Get IsDcLinkVoltageAvailable (Aps4322/Aps42xx)

```
100: // Get "IsDcLinkVoltageAvailable"
     bDcLinkVoltageAvailable := ipXpu.TcIoTile(1,1).GetDcLinkAvailable();
     nState := nState + 1;
```

Get TileType (Aps4322/Aps42xx)

```
110: // Get Tiletype
     stTileType := ipXpu.TcIoTile(1,1).GetTileType();
     nState := nState + 1;
```

Get GetDrvTemperatureMax (only APS42xx)

```
110: // Get highest temperature of this tile - only available for Aps42xx tiles!
     stAps42xxDrvInfo := ipXpu.TcIoTile(1,1).Aps42xx.GetDrvTemperatureMax();
     stAps42xxDrvInfo.fValue; // Temperature in Deg
     stAps42xxDrvInfo.nSubTileIndex; // Index of the sub-tile with the highest temperature
     stAps42xxDrvInfo.eSensorLocation; // Sensor location on the sub-tile --> north, south, ...
     nState := nState + 1;
```

Read CoE data "General"

```
140: // Read CoE-Data "General" from Aps4322/Aps42xx tiles
    sCoE_Btn         := ipXpu.TcIoTile(1,1).CoE_General.GetCoeBtn();
    sCoE_DeviceName  := ipXpu.TcIoTile(1,1).CoE_General.GetDeviceName();
    // ...
    // ...
    // Go on like this with all relevant CoE-data ...
    nState := nState + 1;
```

Read CoE data "Diag Data"

```
150: // Read CoE-Data "Diag Data" from Aps4322 tiles
    fTileTemperatureNorth    := ipXpu.TcIoTile(1,1).Aps4322.CoE_DiagData.GetTemperatureNorth();
    fTileTemperatureEast     := ipXpu.TcIoTile(1,1).Aps4322.CoE_DiagData.GetTemperatureEast();
    fTileTemperatureSouth    := ipXpu.TcIoTile(1,1).Aps4322.CoE_DiagData.GetTemperatureSouth();
    fTileTemperatureWest     := ipXpu.TcIoTile(1,1).Aps4322.CoE_DiagData.GetTemperatureWest();
    fTileTemperatureCenter   := ipXpu.TcIoTile(1,1).Aps4322.CoE_DiagData.GetTemperatureCenter();
    // ...
    // ...
    // Go on like this with all relevant CoE-data ...
    nState := nState + 1;
```

15.1.3.4 Obsolete functions

Reading the CoE parameters

This function is obsolete and should not be used.

Updating a CoE object
The *Tc3_XPlanarUtility* coordinates access to CoE data to avoid communication errors. You can only update the data of one CoE object at a time. Parallel access to modules within an EtherCAT master is not permitted. These types of function blocks and methods must not be used in FOR-loops or similar.

```
30: // Update CoE Data - DONT USE IN FOR-LOOP !!!
     IF ipXpu.TcIoTile(1,1).Aps4322.CoE.DiagData.Update() THEN
          fTemp1 :=ipXpu.TcIoTile(1,1).Aps4322.CoE.DiagData.TemperatureCenter;
          nState := nState + 1;
     END_IF
```

Access to all data of a CoE object

This function is obsolete and should not be used.

The entire CoE object is always read with the *Update* method. If, for example, all temperatures are to be read, the entire object can be made available with the *All* property.

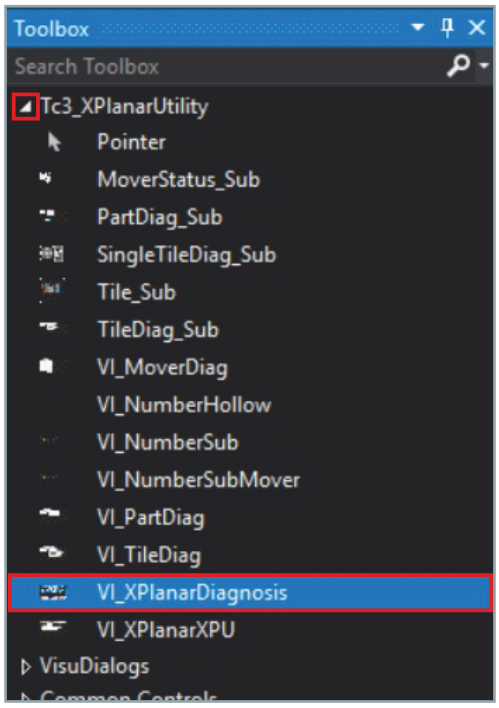
```
31: // Update CoE Data - DO NOT USE IN FOR-LOOP !!!
     IF ipXpu.TcIoTile(1,1).Aps4322.CoE.DiagData.Update() THEN
          stCoeDriveDiagData := ipXpu.TcIoTile(1,1).Aps4322.CoE.DiagData.All;
          nState := nState + 1;
     END_IF
```

15.1.4 Visualization

NOTICE

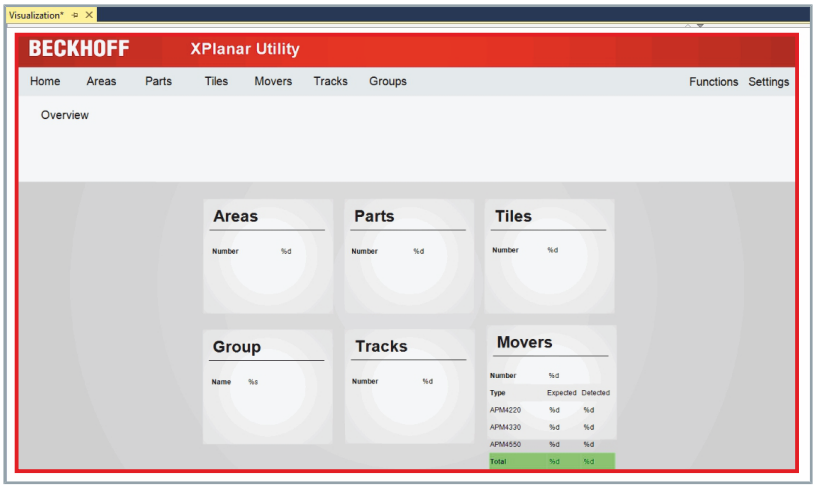
Visualization only serves as support during programming
The visualization is only for support during programming of an XPlanar system and cannot be used for visualization in the plant operator interface.

15.1.4.1 Adding a visualization to a project



► Expand *Toolbox* > *Tc3_XPlanarUtility*

► Press and hold the left mouse button and drag **VI_XPlanar_Diagnostics** into the project window *Visualization*



The visualization is displayed in the project window *Visualization*.

► If required, enlarge the visualization for better operability

► If necessary, position the visualization in the project window

15.1.5 Parameters

15.1.5.1 FB_TcIoXPlanarEnvironment

15.1.5.1.1 Method

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetXpuOids	–	BOOL	–	Updates the object ID list and the number of Planar Groups that can be retrieved via *P_XpuOids* or *P_XpuCount*.
GetEnvironmentOids	–	BOOL	–	Updates the object ID list and the number of Planar Groups that can be retrieved via *P_EnvironmentOids* or *P_EnvironmentCount*.
GetGroupOids	–	BOOL	–	Updates the object ID list and the number of Planar Groups that can be retrieved via the *P_GroupOids* or *P_GroupCount*.
GetTrackOids	–	BOOL	–	Updates the object ID list and the number of Planar Tracks that can be retrieved via the *P_TrackOids* or *P_TrackCount* property.
Subitem				
TcIoXpu	–	I_TcIoXPlanarProcessingUnit	UDINT	Selects an Xpu for further operation by entering the number or OTCID of this Xpu.
McEvironment	–	I_McPlanarEnvironement	UDINT	Selects a planar environment for further operation by entering the number or OTCID of this environment.
McGroup	–	I_McPlanarGroup	UDINT	Selects a Planar Group for further operations by entering the number or OTCID of this group.
McTrack	–	I_McPlanarTrack	UDINT	Selects a Planar Track for further operations by entering the number or OTCID of the track.
McMover	–	I_McPlanarMover	UDINT	Selects a Planar Mover for further operations by entering the number or OTCID of the mover.

15.1.5.1.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_XpuCount	–	UDINT	–	Returns the number of Xpus.
P_XpuOids	–	REFERENCE TO ARRAY [1..Param_TcIoX-PlanarEnviron-ment.MaxProcessin-gUnits] OF OTCID	–	Returns the object ID list of the Xpus.
P_Environment-Count	–	UDINT	–	Returns the number of Pla-nar Environments.
P_EnvironmentOids	–	REFERENCE TO ARRAY [1..Param_TcIoX-PlanarEnviron-ment.MaxEnviron-ments] OF OTCID	–	Returns the object ID list of the Planar Environment.
P_GroupCount	–	UDINT	–	Returns the number of Pla-nar Groups.
P_GroupOids	–	REFERENCE TO ARRAY [1..Param_TcIoX-PlanarEnviron-ment.MaxGroups] OF OTCID	–	Returns the object ID list of the Planar Group.
P_TrackCount	–	UDINT	–	Returns the number of Pla-nar Tracks.
P_TrackOids	–	REFERENCE TO ARRAY [1..Param_TcIoX-PlanarEnviron-ment.MaxTracks] OF OTCID	–	Returns the object ID list of the Planar Tracks.
P_MoverCount	–	UDINT	–	Returns the number of Pla-nar Movers.
P_MoverOids	–	REFERENCE TO ARRAY [1..Param_TcIoX-PlanarEnviron-ment.MaxMcPlanar-Movers] OF OTCID	–	Returns the object ID list of the Planar Mover.

15.1.5.2 FB_TcIoXPlanarProcessingUnit - I_TcIoXPlanarProcessingUnit

15.1.5.2.1 Method - parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
GetOperationMode	General	XPlanarOperation-Mode	–	Indicates the selected operation mode.
GetEtherCatMaster-SyncTaskObjectId	General	OTCID	–	Returns the ID of the EtherCAT Master Sync Task.
GetDetectionMode	MoverDetection	DetectionMode	–	Returns the Detection Mode.
SetDetectionMode	MoverDetection	BOOL	DetectionMode	Sets the Detection Mode for the next Mover Detection.
SetStartMoverRedetection	MoverDetection	BOOL	–	Starts Mover Detection.
SetMoverDetection-Setup	MoverDetection	BOOL	UNIT, PVOID	Sets the Mover Detection setup based on the number and address of a series of Mover Detection setups.
SetCoordMoverCommunicationCmd	MoverDetection	BOOL	Coordinated-MoverComm-Command	Starts Mover Detection based on the selected communication command for detecting the movers.

15.1.5.2.2 Method - parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetDriverVersion	ProcessingUnit	STRING	–	Returns the XPlanar driver version as a STRING.
GetTaskCount	ProcessingUnit	UDINT	–	Returns the number of configured tasks.
GetTaskOid	ProcessingUnit	BOOL	–	Updates the object ID list of all configured tasks. The list can be accessed via the P_TaskOids property.
GetPartCount	Parts	UDINT	–	Returns the number of configured parts.
GetPartOid	Parts	BOOL	–	Updates the object ID list of all configured tasks. The list can be accessed via the P_PartOids property.
GetTileCount	Tiles	UDINT	–	Returns the number of configured tiles.
GetTileOid	Tiles	BOOL	–	Updates the object ID list of all configured tasks. The list can be accessed via the P_TileOids property.
GetMovementArea-Count	MovementAreas	UDINT	–	Returns the number of configured movement areas.

Parameter	Group	Return Type	Input Type	Explanation
GetMovementArea-Oid	MovementAreas	BOOL	–	Updates the object ID list of all configured tasks. The list can be accessed via the P_MovementAreaOids property.
GetMoverCount	Mover	UDINT	–	Returns the number of all movers.
GetMoverOids	Mover	BOOL	–	Updates the object ID of all movers.
GetAmbiguouslyDe-tectedMoverCount	Mover	UDINT	–	Returns the number of movers that were not un-ambiguously detected.
GetAmbiguouslyDe-tectedMover	Mover	REFERENCE TO DetectedMoverInfo	–	Returns the information of the movers that were not unambiguously detected.
GetMoverDetection-Infos	Mover	REFERENCE TO ST_TcIoMoverDetec-tionInfos	–	Returns the information of detected movers.
GetMoverDetection-Status	Mover	REFERENCE TO MoverDetectionSta-tus	–	Returns the current status of the Mover Detection.
GetCoordMover-CommStatus	Mover	REFERENCE TO CoordinatedMover-CommStatus	–	Returns the current status of the Mover Detection.
GetCoordMoverCom-municationErrorId	Mover	DWORD	–	Returns the error ID of the last incorrect Mover Detec-tion.
GetMoverCoupling-Count	MoverCouplings	UDINT	–	Returns the number of all mover couplings. (Rigidly coupled movers)
GetMoverCoupling-Oids	MoverCouplings	BOOL	–	Updates the object ID of all mover couplings. (Rigidly coupled movers)
GetProcessingLi-nesCount	ProcessingLines	UINT	–	Returns the number of pro-cessing lines.
GetProcessingLine-Oids	ProcessingLines	BOOL	–	Updates the object ID list of all processing lines. The list can be accessed via the P_ProcessingLineOids property.
GetParametrization-SetCount	ParametrizationSets	UINT	–	Returns the number of Pa-rameter Sets.
GetParametrization-SetOids	ParametrizationSets	BOOL	–	Updates the object ID list of all Parameter Sets. The list can be accessed via the P_ParametrizationSe-tOids property.

Parameter	Group	Return Type	Input Type	Explanation
Init	Initializing	BOOL	BOOL	Initializes Xpu.
InitNc3	Initializing	BOOL	BOOL	Initializes the Mc_Mover objects. This method is called once in the Init() method. The method should only be repeated separately if the order of the movers has changed. The link between TcIo_XPlanarMover and Mc_XPlanarMover can be changed after a Mover Identification.
Subitems				
TcIoMover	–	I_TcIoXPlanarMover	UDINT	Selects a TcIo mover for further operation by entering the number or OTCID of the selected TcIo mover.
TcIoMoverCoupling	–	I_TcIoXPlanarMover-Coupling	UDINT	Selects a TcIo mover coupling for further operation by entering the number or OTCID of the selected TcIo mover coupling.
McMover	–	I_MCPlanarMover	UDINT	Selects a Mc mover for further operation by entering the number or OTCID of the selected Mc mover.
TcIoMovementArea	–	I_TcIoXPlanarMovementArea	UDINT	Selects a MovementArea for further operation by entering the number or OTCID of the selected MovementArea.
TcIoPart	–	I_TcIoXPlanarPart	UDINT	Selects a part for further operation by entering the number or OTCID of the selected part.
TcIoParaSet	–	I_TcIoMover-ParametrizationSet	UDINT	Selects a Parameter Set for further operation by entering the number or OTCID of the selected Parameter Set.
TcIoTile	–	I_TcIoXPlanarTile	UDINT, UDINT	Selects a tile for further operation by entering the number or OTCID of the selected tile.

Parameter	Group	Return Type	Input Type	Explanation
CoETile	–	I_Aps4322Drive	UDINT, UDINT	Selects a tile for further operation (access to the CoE parameter of the APS4322) by entering the numbers of the part and the tile or the OTCID of the selected tile.

15.1.5.2.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_MoverCoupling-Oids	–	REFERENCE TO ARRAY [0..Param_TcIoX-PlanarEnviron-ment.MaxMover-sPerXpu] of OTCID	–	Returns the object ID of all mover couplings (rigidly coupled movers).
P_MoverOids	–	REFERENCE TO ARRAY [0..Param_TcIoX-PlanarEnviron-ment.MaxMover-sPerXpu] of OTCID	–	Returns the object ID of all movers.
P_PartOids	–	REFERENCE TO ARRAY [0..Param_TcIoX-PlanarEnviron-ment.MaxParts] of OTCID	–	Returns the object ID of all parts.
P_TaskOids	–	REFERENCE TO ARRAY [1..12] of OTCID	–	Returns the object ID of all tasks.
P_TileOids	–	REFERENCE TO ARRAY [0..Param_TcIoX-PlanarEnviron-ment.MaxTilesPer-Part] of OTCID	–	Returns the object ID of all tiles per part.
P_MovementArea-Oids	–	REFERENCE TO ARRAY [0..Param_TcIoX-PlanarEnviron-ment.MaxAreas]	–	Returns the object ID of all MovementAreas.
P_Parametrization-SetOids	–	REFERENCE TO ARRAY [0..Param_TcIoX-PlanarEnviron-ment.MaxMover-ParametrizationSets]	–	Returns the object ID of all Parameter Sets.

15.1.5.3 FB_TcIoXPlanarMovementArea – I_TcIoXPlanarMovementArea

15.1.5.3.1 Method - parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(31)	–	Returns the name of this TcCOM object.
GetIsIncludedIn-MoverDetection	–	BOOL	–	Returns the information that this area is included in Mover Detection.
GetPartConfigura-tions	–	PartConfiguration	UINT	Returns the configuration of the part based on an in-dex.

15.1.5.3.2 Method – parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetDetectedMover-Count	–	UDINT	–	Returns the number of de-tected movers in this area.
GetDetectedMover-Info	–	ST_TcIoDetected-MoverInfo	Int	Returns information about a mover in this area based on an index.
GetMoverCount	–	UDINT	–	Returns the current num-ber of movers in this area.
GetPartCount	–	UDINT	–	Returns the number of parts in this area.

15.1.5.4 FB_TcIoXPlanarMover – I_TcIoXPlanarMover

15.1.5.4.1 Method - parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetIpMcMover	–	I_McPlanarMover	–	Returns the interface of the linked NC mover object.
GetAccessoryId	General	MoverAccessoryID	–	Returns the configured accessory ID of this mover.
GetAccessoryIdCount	General	UDINT	–	Returns the number of configured accessory IDs that are configured for this mover.
GetBTN	General	STRING(13)	–	Returns the BTN of this mover.
GetMoverType	General	XPlanarMoverType	–	Returns the mover type.
GetMoverParametrizationSetId	General	OTCID	–	Returns the Parameter Set that is configured for this mover.
GetTaskOid	General	OTCID	–	Returns the task object ID of this mover.
SetMoverParametrizationSetOid	General	BOOL	OTCID	Sets a new Parameter Set object ID for this mover.
SetTaskOid	General	BOOL	OTCID	Sets a new task object ID for this mover.
GetPayloadInertia	Payload	ST_TcIoPayload	–	Returns the payload inertia of this mover.
GetPayloadProductsOfInertia	Payload	ST_TcIoPayloadProductsOfInertia	–	Returns the inertia products of the payload of this mover.
SetPayloadInertia	Payload	BOOL	ST_TcIoPayload	Sets the payload inertia of this mover.
SetPayloadProductsOfInertia	Payload	BOOL	ST_TcIoPayloadProductsOfInertia	Sets the inertia products of the payload of this mover.
GetSimulationModeStartPosition	SimulationMode	REFERENCE TO SimulationModeStartPosition	–	Returns the start position of the mover in simulation mode.
GetForceLimitMax	ForceLimits	REFERENCE TO Force6D	–	Returns the maximum force limit of this mover.
GetForceLimitMin	ForceLimits	REFERENCE TO Force6D	–	Returns the minimum force limit of this mover.
SetForceLimitMax	ForceLimits	BOOL	Force6D	Sets the maximum force limit for this mover.
SetForceLimitMin	ForceLimits	BOOL	Force6D	Sets the minimum force limit for this mover.
GetControllersOn	Controller	ST_TcIoControllersOn	–	Returns the status of the controller of this mover.
SetControllersOn	Controller	BOOL	ST_TcIoControllersOn	Sets the status of the controller for this mover.

Parameter	Group	Return Type	Input Type	Explanation
SetMovementArea	Methods	BOOL	EnterOrLeave-MovementArea-Parameters	Sets the MovementArea parameters of this mover.
SetMoverCommuni-cationCommand	Communication	BOOL	MoverCommu-nicationCom-mand	Sets the mover communi-cation command for this mover.
GetPositionOffset	Position Offset	MoverVector	–	Returns the current posi-tion offset of this mover.
SetPositionOffset	Position Offset	BOOL	MoverVector	Sets a position offset for this mover.

15.1.5.4.2 Method - parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetActualPositions	–	MoverPositionInfo	–	Returns the position infor-mation of this mover.
GetNcMoverOid	–	OTCID	–	Returns the object ID of the linked NC mover ob-ject.
GetNcPartOid	–	OTCID	–	Returns the object ID of the NC part object ID.
GetBumperHwVer-sion	Communication	STRING(13)	–	Returns the hardware ver-sion of the mounted ID bumper. The ID com-munication must be trig-gered first.
GetCommErrorId	Communication	DWORD	–	Returns the ID of the error of the last ID communica-tion triggered.
GetSignalToNoiseR-ation	Communication	LREAL	–	Returns the signal-to-noise ratio of the last ID commu-nication triggered.
GetCommunication-Status	Communication	MoverCommunica-tionStatus	–	Returns the communica-tion status of the mover.
GetCurrentCommuni-cationAttempt	Communication	UDINT	–	Returns the number of communication attempts of the last ID communication triggered.
GetFileTransfer-Progress	Communication	UDINT	–	Returns the progress of the current file transfer - only for firmware updates of the mover.
GetFirmwareVersion	Communication	MoverFirmwareVer-sion	–	Returns the firmware ver-sion of the installed bumper. The ID communi-cation must be triggered first.
GetIdentified-BumperBtn	Communication	STRING(13)	–	Returns the last recog-nized BTN of the mounted bumper. The ID communi-cation must be triggered first.
GetIdentified-MoverBtn	Communication	STRING(13)	–	Returns the last recog-nized BTN of the mover. The ID communication must be triggered first.

15.1.5.4.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
ScopeAdr	–	I_XPlanarMoverScopeAdr	–	Returns the access to the scope interface of this mover.

15.1.5.4.4 I_TcIoXPlanarMoverScopeAdr

15.1.5.4.4.1 Method

Parameter	Group	Return Type	Input Type	Explanation
GetActualPosition	–	POINTER TO MoverVector	–	Returns the current position from the scope data.
GetActualVelocity	–	POINTER TO MoverVector	–	Returns the current velocity from the scope data.
GetActualAcceleration	–	POINTER TO MoverVector	–	Returns the current acceleration from the scope data.
GetInternalPositionSetpoints	–	POINTER TO MoverVector	–	Returns the interpolated position setpoints from the scope data.
GetInternalVelocitySetpoints	–	POINTER TO MoverVector	–	Returns the interpolated velocity setpoints from the scope data.
GetInternalAccelerationSetpoints	–	POINTER TO MoverVector	–	Returns the interpolated acceleration setpoints from the scope data.
GetExternalPositionSetpoints	–	POINTER TO MoverVector	–	Returns the external position setpoints from the scope data.
GetExternalVelocitySetpoints	–	POINTER TO MoverVector	–	Returns the external velocity setpoints from the scope data.
GetExternalAccelerationSetpoints	–	POINTER TO MoverVector	–	Returns the external acceleration setpoints from the scope data.
GetForceLimit	–	POINTER TO Force6D	–	Returns the current force limit from the scope data.
GetForceLimitAdaptionFactor	–	POINTER TO LREAL	–	Returns the current adjustment factor for the force limit from the scope data.
GetLimitedOutputForces	–	POINTER TO MoverVector	–	Returns the limited output force from the scope data.
GetMovementareaOID	–	POINTER TO OTCID	–	Returns the MovementArea object ID from the scope data.
GetNcDcTimeStamp	–	POINTER TO ULINT	–	Returns the NC-DC timestamp from the scope data.
GetOutputForces	–	POINTER TO MoverVector	–	Returns the output force from the scope data.
GetPartOID	–	POINTER TO OTCID	–	Returns the object ID of the part from the scope data.
GetPositionError	–	POINTER TO MoverVector	–	Returns the position error from the scope data.

Parameter	Group	Return Type	Input Type	Explanation
GetPositionOnCurrentPart	–	POINTER TO Position2D_LREAL	–	Returns the part based on the XY-position of the scope data.

15.1.5.5 FB_TcIoXPlanarMoverCoupling – I_TcIoXPlanarMoverCoupling

15.1.5.5.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetMover-ParametrizationSetId	General	OTCID	–	Returns the Parameter Set that is configured for this mover.
SetMover-ParametrizationSe-tOid	General	BOOL	OTCID	Sets a new Parameter Set object ID for this mover.
GetPayloadInertia	Payload	ST_TcIoPayload	–	Returns the payload inertia of this mover.
GetPayloadPro-ductsOfInertia	Payload	ST_TcIoPayload-ProductsOfInertia	–	Returns the inertia prod-ucts of the payload of this mover.
SetPayloadInertia	Payload	BOOL	ST_TcIoPay-load	Sets the payload inertia of this mover.
SetPayloadPro-ductsOfInertia	Payload	BOOL	ST_TcIoPay-loadProduct-sOfInertia	Sets the inertia products of the payload of this mover.
GetForceLimitMax	Force Limits	REFERENCE TO Force6D	–	Returns the maximum force limit of this mover.
GetForceLimitMin	Force Limits	REFERENCE TO Force6D	–	Returns the minimum force limit of this mover.
SetForceLimitMax	Force Limits	BOOL	Force6D	Sets the maximum force limit for this mover.
SetForceLimitMin	Force Limits	BOOL	Force6D	Sets the minimum force limit for this mover.
GetControllersOn	Controller	ST_TcIoController-sOn	–	Returns the status of the controller of this mover.
SetControllersOn	Controller	BOOL	ST_TcIoCon-trollersOn	Sets the status of the con-troller for this mover.
GetPositionOffset	PositionOffset	MoverVector	–	Returns the current posi-tion offset of this mover.
SetPositionOffset	PositionOffset	BOOL	MoverVector	Sets a position offset for this mover.

15.1.5.5.2 Method – parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetActualPositions	–	MoverPositionInfo	–	Returns the position infor-mation of this mover.
GetNcMoverOid	–	OTCID	–	Returns the object ID of the linked NC mover ob-ject.
GetSubMoverCount	–	UDINT	–	Returns the number of submovers.
GetSubMoverOids	–	BOOL	–	Reads all submover object IDs.

15.1.5.5.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_SubMoverOids	–	REFERENCE TO Array [0.. Param_TcIoXPlanarEnvironment.MaxMoversPerXpu] OF OTCID	–	Returns the object ID of all submovers.
ScopeAdr	–	I_XPlanarMoverCouplingScopeAdr	–	Returns the access to the scope interface of this mover coupling.

15.1.5.5.4 I_TcIoXPlanarMoverCouplingScopeAdr

Parameter	Group	Return Type	Input Type	Explanation
GetActualPosition	–	POINTER TO MoverVector	–	Returns the current position from the scope data.
GetActualVelocity	–	POINTER TO MoverVector	–	Returns the current velocity from the scope data.
GetActualAcceleration	–	POINTER TO MoverVector	–	Returns the current acceleration from the scope data.
GetInternalPositionSetpoints	–	POINTER TO MoverVector	–	Returns the interpolated setpoints from the scope data.
GetInternalVelocitySetpoints	–	POINTER TO MoverVector	–	Returns the interpolated velocity setpoints from the scope data.
GetInternalAccelerationSetpoints	–	POINTER TO MoverVector	–	Returns the interpolated acceleration setpoints from the scope data.
GetExternalPositionSetpoints	–	POINTER TO MoverVector	–	Returns the external position setpoints from the scope data.
GetExternalVelocitySetpoints	–	POINTER TO MoverVector	–	Returns the external velocity setpoints from the scope data.
GetExternalAccelerationSetpoints	–	POINTER TO MoverVector	–	Returns the external acceleration setpoints from the scope data.
GetForceLimit	–	POINTER TO Force6D	–	Returns the current force limit from the scope data.
GetForceLimitAdaptionFactor	–	POINTER TO LREAL	–	Returns the current adjustment factor for the force limit from the scope data.
GetLimitedOutputForces	–	POINTER TO MoverVector	–	Returns the limited output force from the scope data.
GetMovementareaOID	–	POINTER TO OTCID	–	Returns the MovementArea object ID from the scope data.
GetNcDcTimeStamp	–	POINTER TO ULINT	–	Returns the NC DC timestamp from the scope data
GetOutputForces	–	POINTER TO MoverVector	–	Returns the output force from the scope data.
GetPartOID	–	POINTER TO OTCID	–	Returns the part object ID from the scope data.
GetPositionError	–	POINTER TO MoverVector	–	Returns the position error from the scope data.

Parameter	Group	Return Type	Input Type	Explanation
GetPositionOnCurrentPart	–	POINTER TO Position2D_LREAL	–	Returns the part based on the XY-position of the scope data.

15.1.5.6 FB_TcIoXPlanarPart – I_TcIoXPlanarPart

15.1.5.6.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(31)	–	Returns the name of this TcCOM object.
GetActiveIndex	–	UINT	–	Returns the active index of this part.
GetCoeControlWord	–	UINT	–	Returns the CoE control word of the part.
GetPositions	–	PartConfiguration	–	Returns the XY-position based on the part.
SetStartMoverRedetection	–	BOOL	–	Triggers the Mover Detection for this part.

15.1.5.6.2 Method – parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetCoeStatusWord	–	UINT	–	Returns the CoE status word of this part.
GetMoverDetectionInfos	–	REFERENCE TO PartMoverDetectionInfos	–	Returns the information for part-based Mover Detection, based on the part.
GetMoverDetectionStatus	–	MoverDetectionStatus	–	Returns the Mover Detection status based on the part.
GetTileCount	–	UDINT	–	Returns the number of tiles of the part.
GetTileObjectIds	–	OTCID	UINT	Returns the object ID of a specific tile of the part. Use the input as the index of the requested object ID.
Subitems				
TileCoe	–	I_Aps4322Drive	UINT	Returns the CoE interface of the tile using the input parameter as an index.
TileTcIo	–	I_TcIoXPlanarTile	UINT	Returns the interface of the tile using the input parameter as an index.

15.1.5.6.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_TileOids	–	REFERENCE TO Array [0...Param_TcIoXPlanarEnvironment.MaxTilesPerPart] OF OTCID	–	Returns the object ID of all tiles of the part.

15.1.5.7 FB_TcIoXPlanarTile – I_TcIoXPlanarTile

15.1.5.7.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(31)	–	Returns the name of this TcCOM object.
GetTaskOid	–	OTCID	–	Returns the object ID of the task for this tile.
GetTilePositionOn-Part	–	ST_TcIoTilePosi-tionOnPart	-	Returns the position of the tile based on the part.

15.1.5.7.2 Method – parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetActDcLinkVoltage	–	UINT	–	Returns the current DC link voltage of the tile.
GetPowerSupplyCur-rent	–	INT	–	Returns the current of the tile's power supply.
GetCoeTimeToShut-down	–	UDINT	–	Returns the time-to-shut-down parameter of the tile.
GetDriveOid	–	OTCID	–	Returns the object ID of the tile's drive.
GetDriveStatusWord	–	UINT	–	Returns the status word of the tile's drive.
GetFeedbackOid	–	OTCID	–	Returns the object ID of the feedback of the tile.
GetNetId	–	AMSNETID	–	Returns the AMSNETID of the tile.
GetObservedTile-Power	–	LREAL	–	Returns the observed power of the tile.
GetPowerLimit	–	LREAL	–	Returns the power limit of the tile.
GetPowerOverLimit-Factor	–	LREAL	–	Returns the power-over-limit factor of the tile.
GetSlaveAdr	–	UINT	–	Returns the slave address of the tile.
GetTileType	–	TileType	–	Returns the type of the tile.

15.1.5.8 FB_TcIoXPlanarMoverParametrizationSet – I_TcIoXPlanarMoverParametrizationSet

15.1.5.8.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetControllerParam-Oid	–	OTCID	–	Returns the object ID of the control parameter of the mover parameter set.
GetFilterParamOid	–	OTCID	–	Returns the object ID of the filter parameters of the mover parameter set.
GetGeneralParam-Oid	–	OTCID	–	Returns the object ID of the general parametriza-tion of the mover parame-ter set.
GetInertiaParamOid	–	OTCID	–	Returns the object ID of the inertia parametrization of the mover parameter set.
GetObserverParam-Oid	–	OTCID	–	Returns the object ID of the observer parameters of the mover parameter set.

15.1.5.8.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
Controller	–	I_TcIoController-Parametrization	–	Returns the interface of the control parameters.
Filter	–	I_TcIoFilter-Parametrization	–	Returns the interface of the filter parameters.
General	–	I_TcIoGeneral-Parametrization	–	Returns the interface of the general parameters.
Inertia	–	I_TcIoInertia-Parametrization	–	Returns the interface of the inertia parametrization.
Observer	–	I_TcIoObserver-Parametrization	–	Returns the interface of the observer parameters.

15.1.5.9 FB_TcIoControllerParametrization – I-TcIoControllerParametrization

15.1.5.9.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetAAxisFeedForwardParameters	–	REFERENCE TO RotationalFeedForwardParameters	–	Returns the feedforward parameters of the A-axis of the parameter set.
GetBAxisFeedForwardParameters	–	REFERENCE TO RotationalFeedForwardParameters	–	Returns the feedforward parameters of the B-axis of the parameter set.
GetCAxisFeedForwardParameters	–	REFERENCE TO RotationalFeedForwardParameters	–	Returns the feedforward parameters of the C-axis of the parameter set.
SetAAxisFeedForwardParameters	–	BOOL	REFERENCE TO RotationalFeedForwardParameters	Sets the feedforward parameters of the A-axis of the parameter set.
SetBAxisFeedForwardParameters	–	BOOL	REFERENCE TO RotationalFeedForwardParameters	Sets the feedforward parameters of the B-axis of the parameter set.
SetCAxisFeedForwardParameters	–	BOOL	REFERENCE TO RotationalFeedForwardParameters	Sets the feedforward parameters of the C-axis of the parameter set.
GetAAxisPIDT1Parameters	–	REFERENCE TO RotationalAxisPIDT1Parameters	–	Returns the PIDT1 parameters of the A-axis of the parameter set.
GetBAxisPIDT1Parameters	–	REFERENCE TO RotationalAxisPIDT1Parameters	–	Returns the PIDT1 parameters of the B-axis of the parameter set.
GetCAxisPIDT1Parameters	–	REFERENCE TO RotationalAxisPIDT1Parameters	–	Returns the PIDT1 parameters of the C-axis of the parameter set.
SetAAxisPIDT1Parameters	–	BOOL	REFERENCE TO RotationalAxisPIDT1Parameters	Sets the PIDT1 parameters of the A-axis of the parameter set.
SetBAxisPIDT1Parameters	–	BOOL	REFERENCE TO RotationalAxisPIDT1Parameters	Sets the PIDT1 parameters of the B-axis of the parameter set.
SetCAxisPIDT1Parameters	–	BOOL	REFERENCE TO RotationalAxisPIDT1Parameters	Sets the PIDT1 parameters of the C-axis of the parameter set.
GetXAxisFeedForwardParameters	–	REFERENCE TO LinearAxisFeedForwardParameters	–	Returns the feedforward parameters of the X-axis of the parameter set.
GetYAxisFeedForwardParameters	–	REFERENCE TO LinearAxisFeedForwardParameters	–	Returns the feedforward parameters of the Y-axis of the parameter set.

Parameter	Group	Return Type	Input Type	Explanation
GetZAxisFeedForwardParameters	–	REFERENCE TO LinearAxisFeedForwardParameters	–	Returns the feedforward parameters of the Z-axis of the parameter set.
SetXAxisFeedForwardParameters	–	BOOL	REFERENCE TO LinearAxisFeedForwardParameters	Sets the feedforward parameters of the X-axis of the parameter set.
SetYAxisFeedForwardParameters	–	BOOL	REFERENCE TO LinearAxisFeedForwardParameters	Sets the feedforward parameters of the Y-axis of the parameter set.
SetZAxisFeedForwardParameters	–	BOOL	REFERENCE TO LinearAxisFeedForwardParameters	Sets the feedforward parameters of the Z-axis of the parameter set.
SetXAxisPIDT1Parameters	–	BOOL	REFERENCE TO LinearAxisPIDT1Parameters	Sets the PIDT1 parameters of the X-axis of the parameter set.
SetYAxisPIDT1Parameters	–	BOOL	REFERENCE TO LinearAxisPIDT1Parameters	Sets the PIDT1 parameters of the Y-axis of the parameter set.
SetZAxisPIDT1Parameters	–	BOOL	REFERENCE TO LinearAxisPIDT1Parameters	Sets the PIDT1 parameters of the Z-axis of the parameter set.

15.1.5.9.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.10 FB_TcIoFilterParametrization – I-TcIoFilterParametrization

15.1.5.10.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
SetAAxisFilters	–	BOOL	UINT; PVOID	Sets the filter parameters for the A-axis of the parameter set. Use the UINT as the index and the PVOID as the address of a local variable that is to be written to the filter parameter set.
SetBAxisFilters	–	BOOL	UINT; PVOID	Sets the filter parameters for the B-axis of the parameter set. Use the UINT as the index and the PVOID as the address of a local variable that is to be written to the filter parameter set.
SetCAxisFilters	–	BOOL	UINT; PVOID	Sets the filter parameters for the C-axis of the parameter set. Use the UINT as the index and the PVOID as the address of a local variable that is to be written to the filter parameter set.
SetXAxisFilters	–	BOOL	UINT; PVOID	Sets the filter parameters for the X-axis of the parameter set. Use the UINT as the index and the PVOID as the address of a local variable that is to be written to the filter parameter set.

Parameter	Group	Return Type	Input Type	Explanation
SetYAxisFilters	–	BOOL	UINT; PVOID	Sets the filter parameters for the Y-axis of the parameter set. Use the UINT as the index and the PVOID as the address of a local variable that is to be written to the filter parameter set.
SetZAxisFilters	–	BOOL	UINT; PVOID	Sets the filter parameters for the Z-axis of the parameter set. Use the UINT as the index and the PVOID as the address of a local variable that is to be written to the filter parameter set.

15.1.5.10.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.11 FB_TcIoGeneralParametrization – I-TcIoGeneralParametrization

15.1.5.11.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetBrakeAcceleration	–	LREAL	–	Returns the braking acceleration.
GetDefaultSetpointCycleTime	–	LREAL	–	Returns the default setpoint cycle time.
GetDisablingForceDecrement	–	LREAL	–	Returns the decrements of the disabling force per cycle.
GetDisablingTime	–	LREAL	–	Returns the time the mover has for disabling.
GetEnablingForceIncrement	–	LREAL	–	Returns the increments of the enabling force per cycle.
GetEnablingTime	–	LREAL	–	Returns the time the mover has for enabling.
SetDisablingForceDecrement	–	BOOL	LREAL	Sets the decrements of the disabling force per cycle.
SetDisablingTime	–	BOOL	LREAL	Sets the time that the mover has for disabling.
SetEnablingForceIncrement	–	BOOL	LREAL	Sets the increments of the enabling force per cycle.
SetEnablingTime	–	BOOL	LREAL	Sets the time that the mover has for enabling.
GetPositionReferenceMode	–	PositionReferenceMode	–	Returns the current position reference mode.
GetPositionReferenceZOffset	–	LREAL	–	Returns the current position offset reference for the Z-axis.
GetSetpointInterpolation	–	SetpointInterpolation	–	Returns the current setpoint interpolation mode.
SetPositionReferenceMode	–	BOOL	PositionReferenceMode	Sets the current position reference mode.
SetPositionReferenceZOffset	–	BOOL	LREAL	Sets the current position offset reference for the Z-axis.
SetSetpointInterpolation	–	BOOL	SetpointInterpolation	Sets the current setpoint interpolation mode.
SetBrakeAcceleration	–	BOOL	LREAL	Sets the braking acceleration.
SetDefaultSetpointCycleTime	–	BOOL	LREAL	Sets the default setpoint cycle time.
GetZPosAtEnable	–	LREAL	–	Returns the Z-position that the mover attempts to reach when enabled.
SetZPosAtEnable	–	BOOL	LREAL	Sets the Z-position that the mover attempts to reach when enabled.

15.1.5.11.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.12 FB_TcIoInertiaParametrization – I_TcIoInertiaParametrization

15.1.5.12.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetBasicInertia	–	ST_TcIoPayload	–	Returns the basic inertia
GetProductsOfInertiaParameters	–	ST_TcIoPayload-ProductsOfInertia	–	Returns the product of inertia.
SetBasicInertia	–	BOOL	ST_TcIoPayload	Sets the basic inertia.
SetProductsOfInertiaParameters	–	BOOL	ST_TcIoPayloadProductsOfInertia	Sets the inertia products of the payload.

15.1.5.12.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.13 FB_TcIoObserverParametrization – I_TcIoObserverParametrization

15.1.5.13.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetAAxisObserver-Parameters	–	REFERENCE TO RotationalAxisObserverParameters	–	Returns the observer parameters of the A-axis.
GetBAxisObserver-Parameters	–	REFERENCE TO RotationalAxisObserverParameters	–	Returns the observer parameters of the B-axis.
GetCAxisObserver-Parameters	–	REFERENCE TO RotationalAxisObserverParameters	–	Returns the observer parameters of the C-axis.
GetXAxisObserver-Parameters	–	REFERENCE TO LinearAxisObserver-Parameters	–	Returns the observer parameters of the X-axis.
GetYAxisObserver-Parameters	–	REFERENCE TO LinearAxisObserver-Parameters	–	Returns the observer parameters of the Y-axis.
GetZAxisObserver-Parameters	–	REFERENCE TO LinearAxisObserver-Parameters	–	Returns the observer parameters of the Z-axis.
SetAAxisObserver-Parameters	–	BOOL	REFERENCE TO RotationalAxisObserver-Parameters	Sets the observer parameters of the A-axis.
SetBAxisObserver-Parameters	–	BOOL	REFERENCE TO RotationalAxisObserver-Parameters	Sets the observer parameters of the B-axis.
SetCAxisObserver-Parameters	–	BOOL	REFERENCE TO RotationalAxisObserver-Parameters	Sets the observer parameters of the C-axis.
SetXAxisObserver-Parameters	–	BOOL	REFERENCE TO LinearAxisObserverParameters	Sets the observer parameters of the X-axis.
SetYAxisObserver-Parameters	–	BOOL	REFERENCE TO LinearAxisObserverParameters	Sets the observer parameters of the Y-axis.
SetZAxisObserver-Parameters	–	BOOL	REFERENCE TO LinearAxisObserverParameters	Sets the observer parameters of the Z-axis.

15.1.5.13.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.14 FB_McPlanarGroup – I_McPlanarGroup

15.1.5.14.1 Method

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetGroupState	–	MC_PLA-NAR_STATE	–	Returns the status of this group.
GetObjectCount	–	UDINT	–	Returns the number of objects in this group.

15.1.5.14.2 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.15 FB_McPlanarMover – I_McPlanarMover

15.1.5.15.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	General	BOOL	BOOL	Initialization
GetObjectName	General	STRING(20)	–	Returns the name of this TcCOM object.
GetCCoodinateModuloTolerance	General	LREAL	–	Returns the modulo tolerance of the C-axis.
GetCCoodinateModulus	General	LREAL	–	Returns the modulus for the C-axis.
GetMoverHeight	General	LREAL	–	Returns the height of the mover.
GetMoverWidth	General	LREAL	–	Returns the width of the mover.
GetInitialPosition	Simulation	MoverVector	–	Returns the mover's starting position.
GetSimulationMode	Simulation	BOOL	–	Returns the simulation mode of the mover.
GetMaximalAcceleration	Maximum Dynamics	MoverVector	–	Returns the maximum acceleration of the mover.
GetMaximalDeceleration	Maximum Dynamics	MoverVector	–	Returns the maximum deceleration of the mover.
GetMaximalVelocity	Maximum Dynamics	MoverVector	–	Returns the maximum velocity of the mover.
GetMaximalJerk	Maximum Dynamics	MoverVector	–	Returns the maximum jerk of the mover.
GetMaximumDynamicA	Maximum Dynamics	DynamicLimits1D	–	Returns the maximum dynamics for the A-axis.
GetMaximumDynamicB	Maximum Dynamics	DynamicLimits1D	–	Returns the maximum dynamics for the B-axis.
GetMaximumDynamicC	Maximum Dynamics	DynamicLimits1D	–	Returns the maximum dynamics for the C-axis.
GetMaximumDynamicXY	Maximum Dynamics	DynamicLimits1D	–	Returns the maximum dynamics for the X-axis and Y-axis.
GetMaximumDynamicZ	Maximum Dynamics	DynamicLimits1D	–	Returns the maximum dynamics for the Z-axis.
GetDefaultAcceleration	Default Dynamics	MoverVector	–	Returns the default acceleration of the mover.
GetDefaultDeceleration	Default Dynamics	MoverVector	–	Returns the default deceleration of the mover.
GetDefaultVelocity	Default Dynamics	MoverVector	–	Returns the default velocity of the mover.
GetDefaultJerk	Default Dynamics	MoverVector	–	Returns the default jerk of the mover.
GetDefaultDynamicA	Default Dynamics	DynamicLimits1D	–	Returns the default dynamics for the A-axis.
GetDefaultDynamicB	Default Dynamics	DynamicLimits1D	–	Returns the default dynamics for the B-axis.
GetDefaultDynamicC	Default Dynamics	DynamicLimits1D	–	Returns the default dynamics for the C-axis.

Parameter	Group	Return Type	Input Type	Explanation
GetDefaultDynamicXY	Default Dynamics	DynamicLimits1D	–	Returns the default dynamics for the X-axis and Y-axis.
GetDefaultDynamicZ	Default Dynamics	DynamicLimits1D	–	Returns the default dynamics for the Z-axis.
GetMaximumLagFilterTime	Monitoring	MoverVector	–	Returns the maximum lag filter time.
GetMaximumPositionLagValue	Monitoring	MoverVector	–	Returns the maximum value of the position lag.
GetPositionLagMonitoringEnabled	Monitoring	BOOL	–	Returns the activated monitoring of the position lag.

15.1.5.15.2 Method – parameter (Online)

Parameter	Group	Return Type	Input Type	Explanation
GetCommandMode	–	MC_PLANAR_MOVER_COMMAND_MODE	–	Returns the command mode of the mover.
GetExternalSetpositionMode	–	MC_EXTERNAL_SET_POSITION_MODE	–	Returns the mode of the external set position of the mover.
GetGroupOid	–	OTCID	–	Returns the group to which the mover belongs.
GetMoverState	–	MC_PLANAR_STATE	–	Returns current state of the mover.
GetPosition	–	MoverVector	–	Returns the current position of the mover.
GetPositionOnTrack	–	ST_MCMoverSetPosOnTrack	–	Returns the mover-on-track information of the mover.
GetTcIoMoverOid	–	OTCID	–	Returns the object ID of the linked TcIo mover.

15.1.5.15.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.16 FB_McPlanarTrack – I_McPlanarTrack

15.1.5.16.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetCheckCollision-AgainstStaticObjects	–	BOOL	–	Returns the collision-with-a-static-object parameter.
GetCollisionRangeA-tEnd	–	LREAL	–	Returns the collision distance at the end of the track.
GetCollision-RangeAtStart	–	LREAL	–	Returns the collision distance at the start of the track.
GetCollisionRange-Mode	–	MC_PLA-NAR_TRACK_COL-LI-SION_RANGE_MOD E	–	Returns the collision distance mode of the track.
GetMaximalMover-Height	–	LREAL	–	Returns the maximum height of the track's movers.
GetMaximalMover-Width	–	LREAL	–	Returns the maximum width of the track's movers.

15.1.5.16.2 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
GetGroupOid	–	OTCID	–	Returns the object ID of the group to which the track belongs.
GetMoverCountOn-Track	–	UDINT	–	Returns the number of the track's movers.
GetMovingMover-Count	–	UDINT	–	Returns the number of movers moving on the track or located along the track.
GetOperationMode	–	MC_PLA-NAR_TRACK_OP-ERATION_MODE	–	Returns the current operation mode of the track.
GetTrackLength	–	LREAL	–	Returns the total length of the track.
GetTrackState	–	MC_PLA-NAR_STATE	–	Returns the state of the track.

15.1.5.16.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of this TcCOM object.

15.1.5.17 FB_McPlanarEnvironment – I_McPlanarEnvironment

15.1.5.17.1 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
Init	–	BOOL	BOOL	Initialization
GetObjectName	–	STRING(20)	–	Returns the name of this TcCOM object.
GetXPlanarXpuOid	–	OTCID	–	Returns the object ID of the XPlanar Processing Unit that is linked to this environment.

15.1.5.17.2 Method – parameter (Init)

Parameter	Group	Return Type	Input Type	Explanation
GetBoundaryElementCount	–	UDINT	–	Returns the number of borders in this environment.
GetGroupOid	–	OTCID	–	Returns the object ID of the group added to this environment.
GetStatorCount	–	UDINT	–	Returns the number of stators in this environment.

15.1.5.17.3 Properties

Parameter	Group	Return Type	Input Type	Explanation
P_GetObjectId	–	OTCID	–	Returns the object ID of the TcCOM object.

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.

More Information:
www.beckhoff.com/xplanar

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