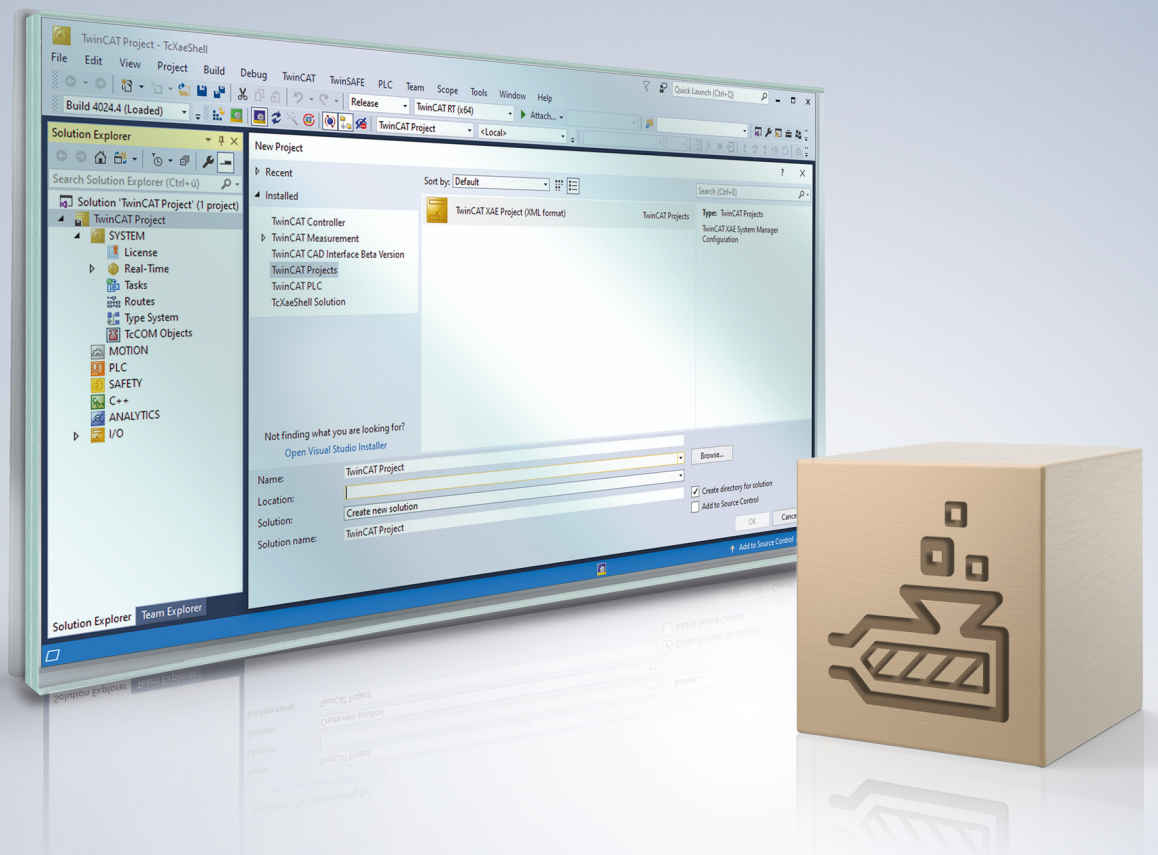
BECKHOFF New Automation Technology

Manual | EN

TF8560

TwinCAT 3 | Plastic Technology Functions



2024-01-16 | Version: 1.0.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

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

Disclaimer

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

Trademarks

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

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

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

Copyright

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

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

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Concept of Libraries

TF8560 TC3 Plastic Technology Functions is a solution that allows the plastics processing industry to rapidly implement TwinCAT 3-based motion tasks.

In plastics processing machines, two different drive types, electric and hydraulic, can be used individually or in combination. The underlying Motion Control libraries for these two drive types are different. Therefore, there is the TwinCAT 3 NC PTP (TF5000) for electric axes and the TwinCAT 3 Hydraulic Positioning (TF5810) for hydraulic axes on the TwinCAT 3 platform. If the control program is developed directly on the basis of these two Motion Control libraries, the customer must re-implement all interfaces that call different libraries when the drive technology is changed.

TC3 Plastic Technology Functions provides a unified interface for the common functions of the hydraulic and electric Motion Control libraries. When developing the control program based on TC3 Plastic Technology Functions, only minimal modification is necessary for a different drive technology.

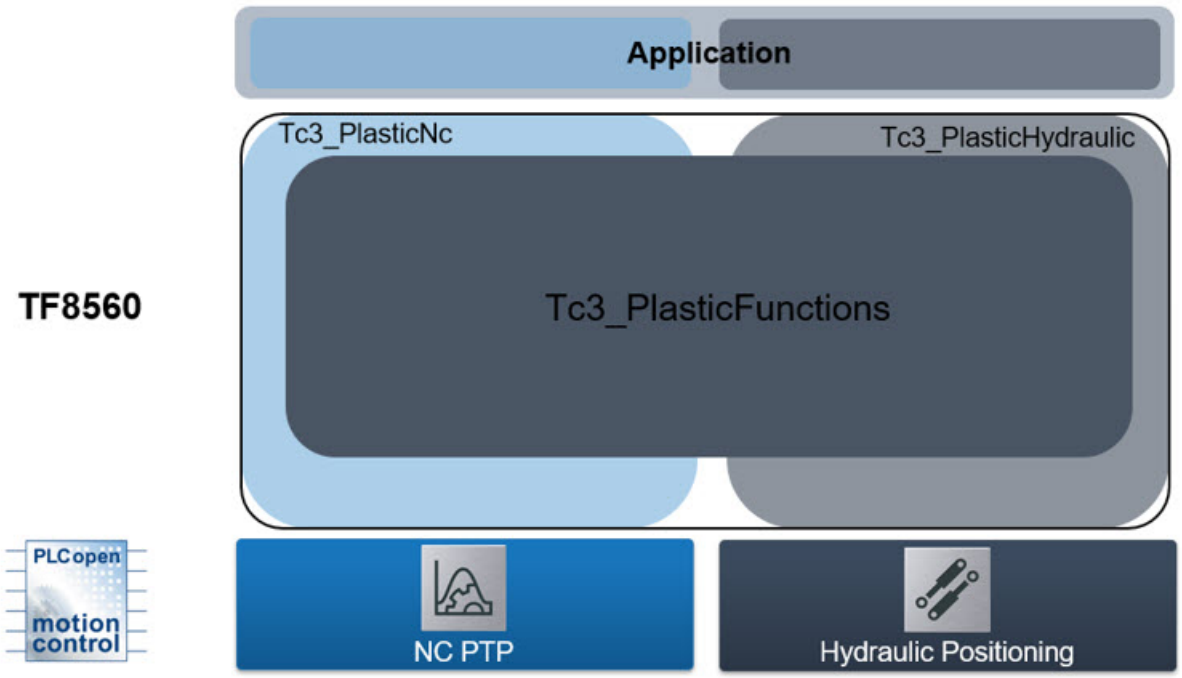
In addition, the elementary motion tasks commonly used in plastics processing, such as cam plates for wall thickness control, multi-segment PTP motion, and pressure control, have been implemented and fully tested in TC3 Plastic Technology Functions, encapsulated as core functions. Customers are freed from building them from scratch and can directly use the provided components to achieve complex functions with little engineering effort.

The basic concept of TC3 Plastic Technology Functions:

- Uniformity: A uniform interface for calling common Motion Control functions for hydraulic and electric axes.
- Ready-to-use: Implementation and encapsulation of essential motion tasks commonly used in plastics processing processes.
- Expandability: Extension or modification of the functions through inheritance.
- Flexibility: Choice of language, object/process-oriented programming approach, multitasking/multi-core capability (to be tested).

TC3 Plastic Technology Functions libraries and license

TC3 Plastic Technology Functions can be considered as an interface between the customer application and the TwinCAT 3 platform. TC3 Plastic Technology Functions consists of three libraries, namely Tc3_PlasticFunctions, Tc3_PlasticNc, and Tc3_PlasticHydraulic. Their dependencies are as shown in the figure below. Tc3_PlasticFunctions implements the common functions for both drive types. The functions specific to the electric or hydraulic axes are implemented respectively in Tc3_PlasticNc and Tc3_PlasticHydraulic and will correspondingly call TF5000 TwinCAT 3 NC PTP or TF5810 TwinCAT 3 Hydraulic Positioning.



Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

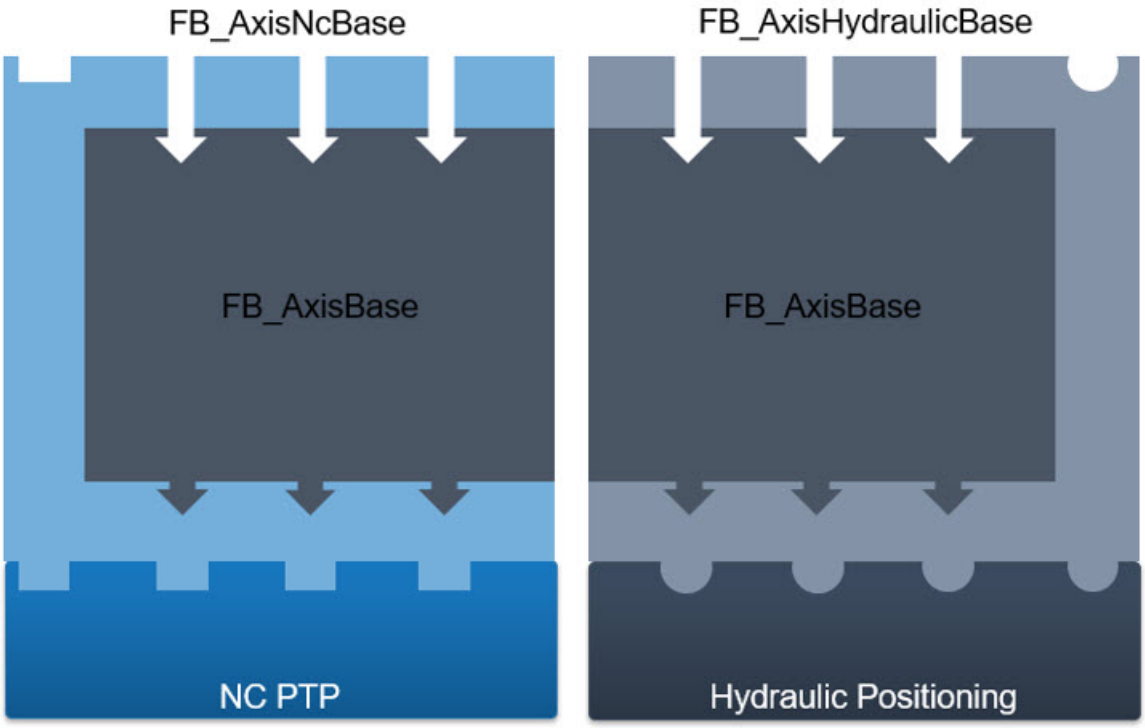
3 Concept of Axes

When setting up the functions of an axis, the methods and properties for implementing an elementary motion task are bundled in special function blocks. These are so-called core functions, which can be an active command, such as executing a multi-segment PTP movement, or a passive task, such as displaying the motion state and parameters of an axis, and are described in more detail in the chapter "Core functions concept".

3.1 FB_AxisBase and derived Axes

TC3 Plastic Technology Functions defines FB_AxisBase as a virtual base axis. This provides the core functions required for both electric and hydraulic drives. FB_AxisBase cannot be instantiated because it is defined as ABSTRACT and the link to specific Motion Control libraries is missing.

FB_AxisNcBase for electrical axes or FB_AxisHydraulicBase for hydraulic axes are derived by inheriting FB_AxisBase. On the one hand, the connection to the Motion Control libraries is established. On the other hand FB_AxisNcBase and FB_AxisHydraulicBase add core functions for their specific scene. The following figure shows how the axis FBs are structured in TC3_Plastic Technology Functions.



3.2 Axes Instantiation

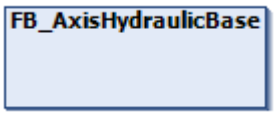
FB_AxisNcBase, FB_AxisHydraulicBase and FB_AxisInvBase can be instantiated. Following you will find sample code of how to create instances of these axis types. The input variables of these FBs are all defined in the method FB_init(). FB_init() is always called implicitly when initializing an instance of a FB. For a detailed description see FB_init().

Some input variables of FB_init() must be assigned mandatory, while others are optional and can be 0. The following table shows the requirements of FB_AxisNcBase, FB_AxisHydraulicBase and FB_AxisInvBase when assigning their input variables. For a detailed explanation of each input variable, see METHOD FB_init().

The open source Tc3_PlasticBaseApplication project, based on TC3_Plastic Technology Functions, has designed common machine axes and their motion tasks in blow molding or injection molding machines, such as clamp, carriage, blow pin, wall thickness control, etc. Customers can use it according to their needs, either to create their own applications or to use this Tc3_PlasticBaseApplication project as a tutorial for TC3_Plastic Technology Functions. The code for Tc3_PlasticBaseApplication is available here.

Name	Description
FB_AxisHydraulicBase	Hydraulic axes, operated with the Tc2_Hydraulics library.
FB_AxisNcBase	Servo axes, operated with the Tc2_MC2 library.
FB_AxisInvBase	Inverter axes.

3.2.1 FB_AxisHydraulicBase



This FB creates an axis operated with the Tc2_Hydraulics library.

Syntax:

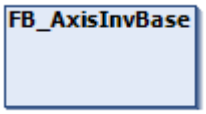
```
fbHydClampAxis1 : FB_AxisHydraulicBase
            (
                AxisName        := 'HydClampAxis1',
                nPtpPoints      := 10,
                iProcessHandler := 0,
                iPosCamLookup   := 0,
                iVeloCamLookup  := 0,
                iEncoder        := fbHydClampAxisEncIn01,
                iDrive          := fbHydClampAxisDriveOut01,
                iPressureP      := fbHydClampAxis1PrsInP,
                iPressureM      := fbHydClampAxis1PrsInM,
                iPosFilter      := 0,
                iVeloFilter     := 0
            );
```

Name	Type	Obligatory	
AxisName	STRING	Yes	Used for messages and file names, among other things.
nPtpPoints	INT	Yes	Number of available segments in PTP tables.
iProcessHandler	I_ProcessHandler	No	Optional: A function block for handling pressures and other process variables.
iPosCamLookup	I_CammingLookup	No	Optional: A lookup function block with a PvsP camming table.
iVeloCamLookup	I_CammingLookup	No	Optional: A lookup function block with a VvsP camming table.
iEncoder	I_InputBase	Yes	A function block for determining the actual position.
iDrive	I_OutputBase	Yes	A function block for the connection of a drive.
iPressureP	I_InputBase	No	Optional: A function block for determining the pressure on the positive acting cylinder surface.
iPressureM	I_InputBase	No	Optional: A function block for determining the pressure on the negative acting cylinder surface.
iPosFilter	I_Filter	No	Optional: A function block for filtering the actual position.
iVeloFilter	I_Filter	No	Optional: A function block for filtering the actual velocity.

Required libraries

Hydraulic axes require the Tc2_Hydraulics library.

3.2.2 FB_AxisInvBase



This FB creates an axis operated with a simple inverter. The axis does not require position feedback and does not support position-bound functions.

Syntax:

```
fbInvExtruder : FB_AxisInvBase
            (
                AxisName        := 'InvExtruder',
                nPtpPoints      := 10,
                iProcessHandler := 0,
                iPosCamLookup   := 0,
                iVeloCamLookup  := 0
            );
```

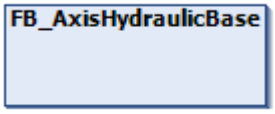
Name	Type	Obligatory	
AxisName	STRING	Yes	Used for messages and file names, among other things.
nPtpPoints	INT	Yes	Number of available segments in PTP tables.
iProcessHandler	I_ProcessHandler	No	Optional: A function block for handling pressures and other process variables.
iPosCamLookup	I_CammingLookup	No	Optional: A lookup function block with a PvsP camming table.
iVeloCamLookup	I_CammingLookup	No	Optional: A lookup function block with a VvsP camming table.

Required libraries

Inverter axes are fully implemented in Tc3_PlasticFunctions and do not require the libraries mentioned above.

This type of axis is prepared for the use of drive components whose definition is below the possibilities of a servo axis. The biggest problem might be the lack of position feedback. But even very simple drives like frequency inverters may be used to support some tasks in a machine, such as turning an extruder or moving a conveyor belt.

3.2.3 FB_AxisNcBase



This FB creates an axis operated with the Tc2_MC2 library.

Syntax:

```
fbNcClampAxis1 : FB_AxisNcBase
            (
                AxisName        := 'NcClampAxis1',
                nPtpPoints      := 10,
                iProcessHandler := 0,
                iPosCamLookup   := 0,
                iVeloCamLookup  := 0
            );
```

Name	Type	Obligatory	
AxisName	STRING	Yes	Used for messages and file names, among other things.
nPtpPoints	INT	Yes	Number of available segments in PTP tables.
iProcessHandler	I_ProcessHandler	No	Optional: A function block for handling pressures and other process variables.
iPosCamLookup	I_CammingLookup	No	Optional: A lookup function block with a PvsP camming table.
iVeloCamLookup	I_CammingLookup	No	Optional: A lookup function block with a VvsP camming table.

Required libraries

NC-based axes require the Tc2_MC2 library and a TwinCAT NC license.

3.3 Access to axes

3.3.1 Access to the axes via interfaces

The TC3_Plastic Technology Functions creates the corresponding INTERFACEs for each function block (FB). For a detailed description see INTERFACE concept. In this document the following naming rules are applied:

FB_Xyz: Declaration of a function block

fbXyz: an instance of FB_Xyz

I_Xyz: the INTERFACE declaration corresponding to FB_Xyz

iXyz: a variable of I_Xyz that is instantiated with fbXyz

NOTICE

Don't access the fbXyz of an axis directly

We suggest that you don't access the fbXyz of an axis directly, but instead access the properties and methods of the FB_Xyz via iXyz, as is shown below. This is because the development of I_Xyz in TC3_Plastic Technology Functions only considers the properties and methods that are needed externally to operate FB_Xyz. In contrast to that, the properties and methods that are only to be called by other methods inside FB_Xyz do not appear in I_Xyz. As a result, iXyz is clearer than fbXyz and customers can find the information they need faster than with fbXyz, while the risk of misuse is prevented.

```
sName := fbNcClampAxis1.AxisName;  // not prefered
sName := iNcClampAxis1_B.AxisName; // prefered
```

It should be noted that TC3_Plastic Technology Functions creates two INTERFACEs with different information content for each function block of the axes. E.g. FB_AxisBase has I_AxisBase and I_AxisBaseDev. I_AxisBase contains the necessary functions of the axes for most common cases. In contrast, I_AxisBaseDev provides more information access and enables customers to implement more complex functions. For reasons of simplicity and security, it is recommended to use I_AxisBase.

```
iNcClampAxis1_D:              I_AxisNcBaseDev:=fbNcClampAxis1;
iNcClampAxis1_B:              I_AxisNcBase:=fbNcClampAxis1;

iHydClampAxis1_D:             I_AxisHydraulicBaseDev:=fbHydClampAxis1;
iHydClampAxis1_B:             I_AxisHydraulicBase:=fbHydClampAxis1;
```

The following table shows the properties and methods that can be accessed via the interfaces of the individual axes. For detailed definitions, see Calling methods of an axis [▶ 19][link].

I_AxisBase

Name	Description
Actuals [▶ 33]	An interface to a local function block that provides current actual values (positions, velocities, etc.).
AutoTorqueLimitSelect	A TRUE allows the axis to select an internal function block for torque limitation if the type of a servo drive (CoE, SoE) has been detected.
AxisName	The text-based name of the axis.
Camming [▶ 38]	An interface to a local function block that provides functions for processing cam plates.
CycleTime	The call cycle time of the most important axis functionalities.
CycleTimeValid	TRUE, if the call cycle time was determined to be valid.
Cyclic	This method is called cyclically and organizes the calculation of the runtime values of the axis.
DisableSoftEnd [▶ 42]	An interface to a local function block that provides the ability to temporarily disable the software limit switches of the axis and restore them to their original state.
DoReset	This method triggers the clearing of error states in the axis and its local functions.
Estop [▶ 44]	An interface to a local function block that cancels an active travel command and brings the axis to a stop. If available, increased dynamic parameters are applied.
GetProcessHandler	Reserved for future extension.
Homing [▶ 45]	An interface to a local function block that provides a choice of homing procedures.
Jog [▶ 68]	An interface to a local function block that commands motion without a specified destination (driving on revocation).
MotionParams [▶ 72]	An interface to a local function block that provides a set of axis parameters.
MotionSetpoints [▶ 74]	An interface to a local function block that provides current setpoints (position, velocity, etc.).
Power [▶ 74]	An interface to a local function block that controls the general enable of the axis and its directional feed enables.
Ptp [▶ 86]	An interface to a local function block that triggers active motion. A table with a section-by-section definition is used.
SetPosition [▶ 96]	An interface to a local function block that is used to change the actual position of the axis.
SetProcessHandler	Reserved for future extension.
State	The current state of the axis.
Stop [▶ 98]	An interface to a local function block that cancels an active travel command and brings the axis to a stop. The dynamics parameters of the travel command are used.
SwitchOver	Reserved for future extension.
TeachPosition	An interface to a local function block that supports the commissioning of axes with analog position measuring systems.
TeachUpdate	An interface to a local function block that supports the commissioning of axes with analog position measuring systems.
ToolAdaption [▶ 100]	An interface to a local function block that performs the conversion of positions and velocities between an axis and a tool.
TorqueLimiting	An interface to a function block used for torque limitation of the axis.
VelocityFeed [▶ 101]	An interface to a local function block that triggers a motion without specifying a destination.

I_AxisBaseDev

Name	Description
Actuals [▶ 33]	See I_AxisBase
AppendCorefunction	Reserved, for internal use only.
AutoTorqueLimitSelect	See I_AxisBase

Name	Description
AxisIsInverter	TRUE, if the axis is implemented with a frequency inverter.
AxisIsNc	TRUE, if the axis is based on TwinCAT NC.
AxisName	See I_AxisBase
Camming [▶ 38]	See I_AxisBase
CmdCurrent	Reserved, a counter that assigns an identification to each activated command.
CmdNext	Reserved, the next value to be used as CmdCurrent.
ConvertCountToPos	Reserved, for internal use only.
ConvertPosToCount	Reserved, for internal use only.
CoreDebug	Reserved, an interface to a local function block that supports debugging.
CycleTime	See I_AxisBase
CycleTimeValid	See I_AxisBase
Cyclic	See I_AxisBase
DisableSoftEnd [▶ 42]	See I_AxisBase
DoReset	See I_AxisBase
EnterCriticalSection	Reserved, for internal use only.
Estop [▶ 44]	See I_AxisBase
GetProcessHandler	See I_AxisBase
ExtGenerated	TRUE, if the setpoint generation is done via a special method.
ForceState	This method changes the state of the axis with high priority.
GoErrorBase	This method can be used to set the axis to an error state.
Homing [▶ 45]	See I_AxisBase
Jog [▶ 68]	See I_AxisBase
LeaveCriticalSection	Reserved, for internal use only.
MotionParams [▶ 72]	See I_AxisBase
MotionSetpoints [▶ 74]	See I_AxisBase
Power [▶ 74]	See I_AxisBase
Ptp [▶ 86]	See I_AxisBase
PtpPoints	The number of supported segments in the table supported by Ptp.
ReadCycleTime	Reserved, used to determine the cycle time.
SetPosition [▶ 96]	See I_AxisBase
SetProcessHandler	See I_AxisBase
SetTorqueLimiting	This method connects a function block for torque limitation with the axis.
State	See I_AxisBase
Stop [▶ 98]	See I_AxisBase
SwitchOver	See I_AxisBase
TeachPosition	See I_AxisBase
TeachUpdate	See I_AxisBase
ToolAdaption [▶ 100]	See I_AxisBase
TorqueLimiting	See I_AxisBase
VelocityFeed [▶ 101]	See I_AxisBase

I_AxisNcBase EXTENDS I_AxisBase

Name	Description
ActualsNc [▶ 34]	An interface to a local function block that provides current actual values (positions, velocities, torque, etc.). It is an extended version of Actuals.

I_AxisNcBaseDev EXTENDS I_AxisBaseDev

Name	Description
ActualsNc [▶ 34]	See I_AxisNcBase.
GetNcAxisRef	The address to the mapping interface between NC and PLC.

I_AxisHydraulicBase EXTENDS I_AxisBase

Name	Description
ActualsHydraulics [▶ 33]	An interface to a local function block that provides current actual values (positions, velocities, pressures, etc.). It is an extended version of Actuals.
AutoIdent [▶ 34]	An interface to a function block that performs the automatic characteristic measurement of the axis.
DirectOutput [▶ 41]	An interface to a function block that allows direct output via the drive interface of the axis.
UseDatFile	A TRUE here signals that the axis loads its parameters with function blocks of the hydraulics library from a file during startup.

I_AxisHydraulicBaseDev EXTENDS I_AxisBaseDev

Name	Description
ActualsHydraulics [▶ 33]	See I_AxisHydraulicsBase.
AutoIdent [▶ 34]	See I_AxisHydraulicsBase.
DirectOutput [▶ 41]	See I_AxisHydraulicsBase.
GetHydAxisRef	The address to the mapping interface between hydraulics library and PLC.
NextHydAxisRef	The hydraulic axes are organized in a simple linked list. This property gives an interface to the successor.
NextHydAxisChainlength	The number of successors of the axis.
PosFilter	An interface to the function block used for filtering the actual position of the axis.
PressureInputM	An interface to the function block used to determine the actual pressure of the axis acting in the negative direction of movement.
PressureInputP	An interface to the function block used to determine the actual pressure of the axis acting in the positive direction of movement.
SetDrive	Here, an interface is transferred to a function block that executes the communication with the hardware for the drive of the axis.
SetEncoder	Here, an interface is transferred to a function block that executes the communication with the hardware for the actual value acquisition of the axis.
UseDatFile	See I_AxisHydraulicsBase.
VeloFilter	An interface to the function block used for filtering the actual velocity of the axis.

I_AxisInvBase EXTENDS I_AxisBase

Name	Description
no additional functionality	

I_AxisInvBaseDev EXTENDS I_AxisBaseDev

Name	Description
RefDevice	An interface to a local function block of the axis, which is used as an interface between the general inverter axis and the special adaptation to the used or simulated device.
RefMotionSetpoints	An interface to a local function block of the axis, which allows the inverter axis to provide its setpoints.

Name	Description
RefReadParameter	An interface to a local function block of the axis, which allows the inverter axis to provide parameters.
RefReset	An interface to a local function block of the axis, which allows the inverter axis to clear errors that have occurred.
RefWriteParameter	An interface to a local function block of the axis, which allows the inverter axis to write parameters.

3.3.2 Accessing the properties of an axis

Some of the properties of axes are standard variables (e.g. iNcSampleAxis.AxisName of type STRING) and the required information can be processed directly. Others are so-called core functions and of type INTERFACE.

For example, the axis has a property of type I_Power called Power, which is the INTERFACE of FB_Power. This interface can be used to access properties (e.g. iNcSampleAxis.Power.Status) and methods (e.g. iNcSampleAxis.Power.DoPower()) provided there.

3.3.3 Calling methods of an axis

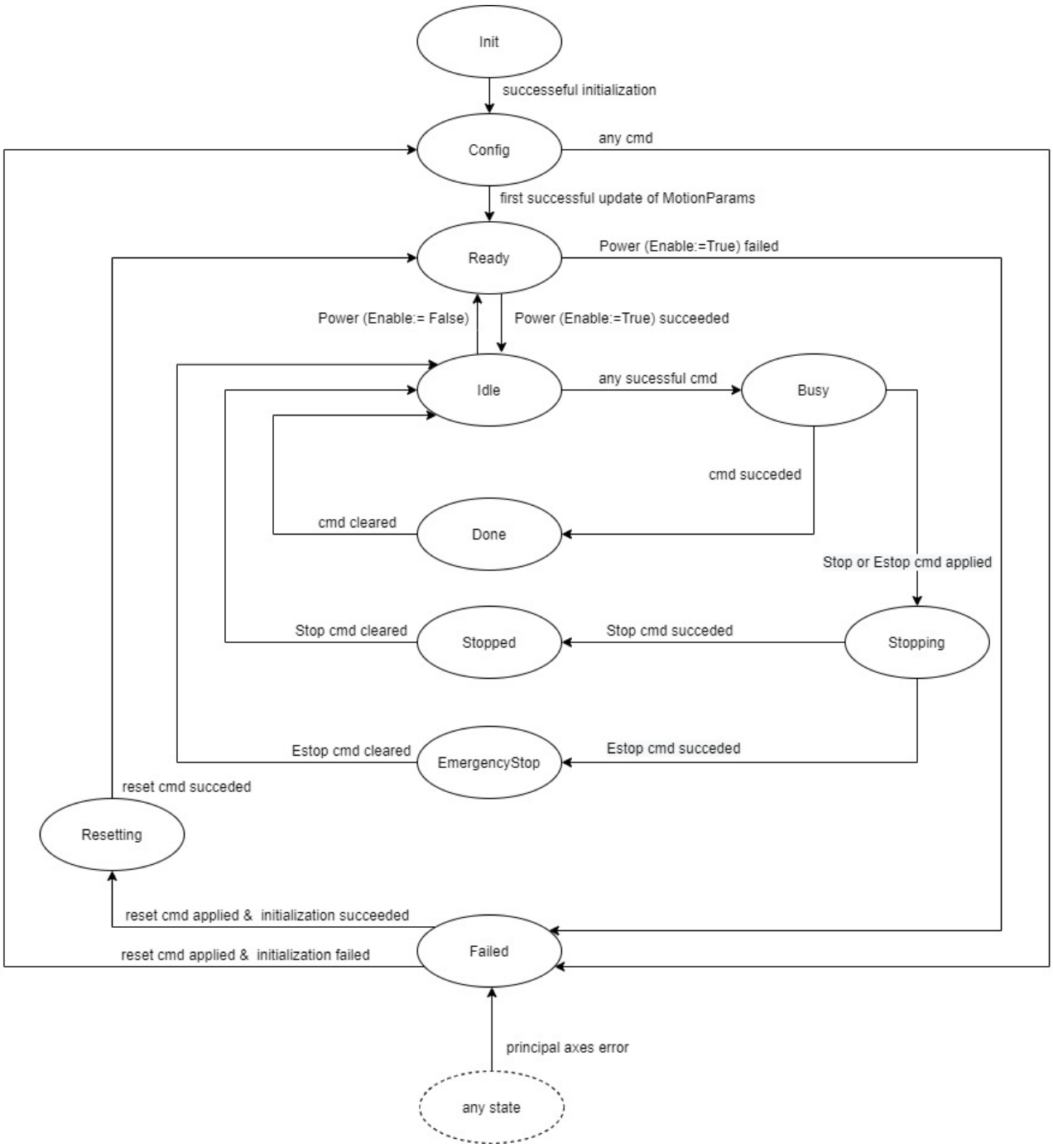
In TC3 Plastic Technology Functions, none of the code implementation is in the body of FBs, but in their methods. The code in the methods is executed only when the methods are called. In TC3 Plastic Technology Functions, the code that must be executed in every PLC cycle is placed in the Cyclic() method. Customers should call this Cyclic() method in every PLC cycle when developing on the basis of TC3 Plastic Technology Functions. For more details on how to call the methods of the axes, you can refer to Tc3 PlasticBaseApplication. See [Link] for more details on methods.

3.3.4 States & state machine of an axis

The state of an axis is declared via an enumeration of the type E_AxisState. This information is provided by each axis as a property with the name State (e.g. iNcSampleAxis.State). The following states are defined:

State	Description
eInit	The axis is in the initialization phase and must be initialized according to the application requirements. In this state the axis is not ready for operation.
eConfig	The axis applies a series of parameters from the subordinate drive technology (NC, hydraulics library). Settings that are important for correct operation are checked for correspondence to the motion technology. ADS and mapping connections are also tested at the same time.
eReady	The axis has been successfully initialized and configured. It is ready to accept an enable, given via iAxis.Power.
eIdle	The axis is enabled and ready to accept motion commands (e.g. JogP()).
eBusy	The axis is processing a command (e.g. JogP()).
eDone	The axis has successfully completed a command (e.g.TableMove()).
eStopping	The axis is in the state of processing a stop or Estop command.
eStopped	The Stop command was successful.
eEmergencyStop	The Estop command was successful.
eResetting	The axis was instructed by Reset() to initiate the change from the state eFailed to the state eReady.
eFailed	The axis is in the error state.

The state machine of an axis is shown in the figure below:



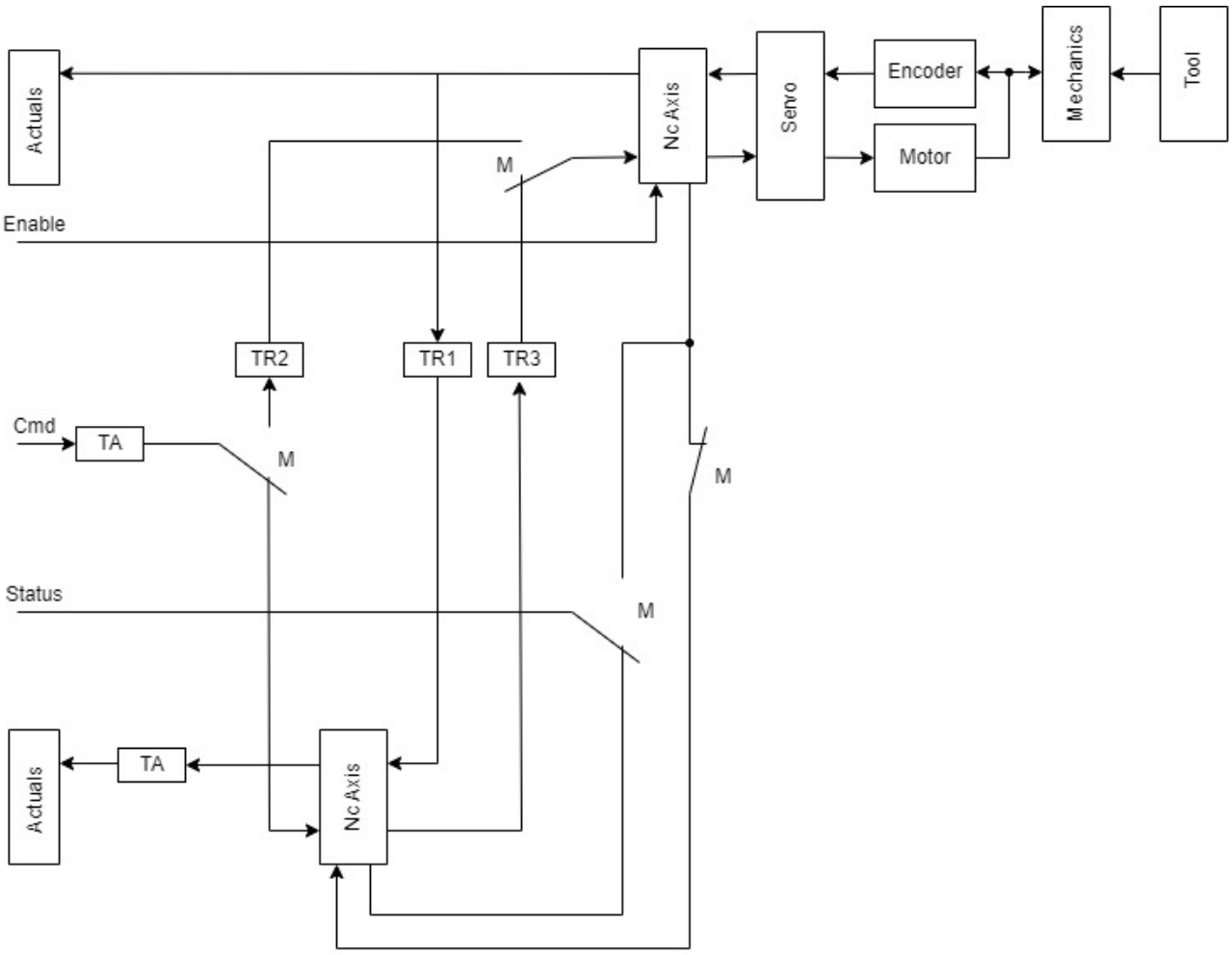
3.4 Transformation axes

The mechanical solution for some axes is created by implementing a non-linear transmission. Here, the tool travel is not following the actuator movement by a constant ratio. This enables the optimization of axis properties for varying requirements in individual areas of the travel. A differing behavior like this must be represented by a matching software concept.

Examples: Toggles, Cranks, Scotch Yokes, Customized concepts.

3.4.1 Construction of a transforming axis

A transformation axis is a container object that implements the same interface as a standard Nc axis. Inside this object there are two local standard axis objects, named Load Side and Drive Side. These internal objects are used to handle the specific requirements for the effective tool and the actuator part. There is almost no exchange between the application project and these internal objects because the usual interactions use the interfaces of the container object.



Used symbols

Symbol	Description
Actuals	A common core function that contains information about the current situation of the axis. There are alternative sub-versions for the different axis types.
TA: Tool Adaptation	A common core function that is used to handle the differences between the axis and the effective tool.
TRx: Transformation	A core function specific to transforming axes. It is used to convert actual values of the drive side axis into load side actual values (TR1) and load side setpoints (TR2) or setpoints (TR3) into drive side values.
MM: Operation mode select	In this figure the "Full transformation mode" is selected.

3.4.2 Semi-transformation mode

In this operation mode all commands are forwarded to the Drive Side axis. To avoid unexpected position lag errors, the Load Side axis is not enabled.

The actual position and velocity values of the Load Side are updated using translated values from the Drive Side.

Any commanded motion will be executed by the Drive Side using translated target position values. A commanded move to 100.0 will make the tool travel to 100.0 mm, no matter what Drive Side motor angle is required.

Velocity cannot be translated

The commanded velocity cannot be translated because the result would depend on the position.

No constant velocity

Because the profile generation is executed by the Drive Side axis, the tool will not travel with constant velocity.

No position or velocity camming

A position or velocity camming is not supported.

3.4.3 Full transformation mode

In this operation mode, almost all commands are forwarded to the Load Side axis. Enabling the container object will enable both internal axes.

The actual position and velocity values of the Load Side are updated using translated values from the Drive Side.

The transformation will be performed by converting the output of the Load Side profile calculation. Again, a commanded move to 100.0 will make the tool travel to 100.0 mm, no matter what Drive Side motor angle is required.

Constant velocity

Because the profile generation is executed by the Load Side axis, the tool will travel with constant velocity.

Excessive Drive Side velocity values required

In some areas of the travel, even low tool velocities may require excessive Drive Side velocity values.

Full transformation temporarily paused

For Jog or Homing commands, the full transformation is temporarily paused. All position and velocity values are used following Drive Side definitions without any translation.

4 Core functions concept

In plastics machines, an axis must perform a variety of elementary motion tasks, such as multi-stage PTP movements, pressure control, homing, and so on. Usually, these motion commands are independent of each other and the axis performs only one motion command at a time. TC3 Plastic Technology Functions implements and encapsulates each of these motion tasks in a separate FB called a core function.

The core functions are well tested and have been defined in a consistent format. The base axis defined in TC3_Plastic Technology Functions provides the common core functions. By deriving a specific axis, customers can also replace or append the function of a core function without affecting the behavior of other core functions. This makes the TC3 Plastic Technology Functions flexible and easy to customize.

4.1 Embedding core functions in an axis

A core function cannot operate independently because it contains only the code implementation of the motion task, but is not connected to the Motion Control library. When an axis is initialized, the core function receives the interface of the axis, while the axis contains an interface of the core function. At the same time, the core function receives interfaces to Motion Control library functions.

The axis creates a concatenated list to call its core functions. This list is used during the operation of the axis for the following tasks:

- Signaling an Online Change
- Passing on information about the cycle time
- Cyclic calling of methods
- Command for resetting errors

The core functions have access to the information of the axis. In addition, it is possible to call the interfaces of other core functions of the axis if this is necessary for the coordinated execution of tasks.

4.2 The basics of core functions

All core functions have a number of common features that are implemented in an ABSTRACT FB_Corefunction.

A number of core functions are derived from FB_CorefunctionFeedback [▶ 24] and provide further properties and methods.

Through the inheritance of FB_Corefunction, each core function receives the following properties:

Name	Type	Description
AxisState	E_AxisState	The current state of the axis state machine.
CycleTime	LREAL	The cycle time of the PLC task from which the Cyclic() method of the core function is executed.
CycleTimeValid	BOOL	A TRUE signals that the CycleTime of the core function is defined.
NextCore	I_Corefunction	This property is part of the execution chain and must not be affected by the application task!
OnlineChangeMark	BOOL	The axis uses this property to signal an Online Change to the core function.

Methods

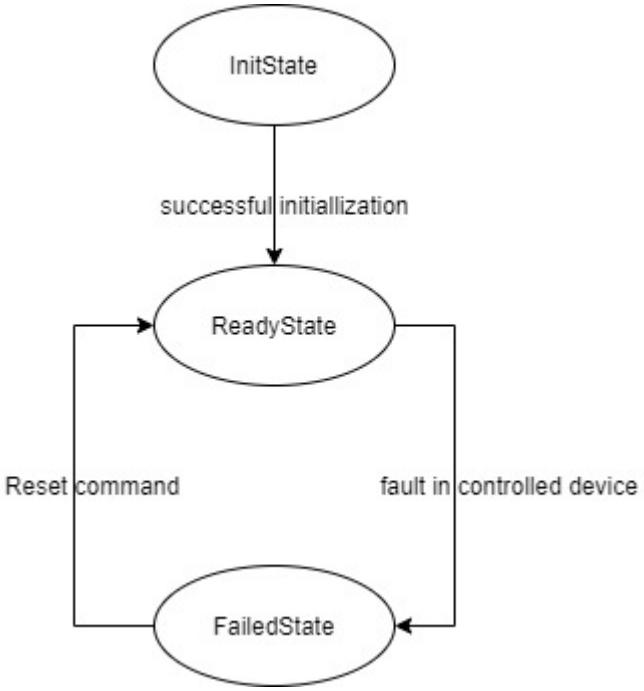
Name	Description
Cyclic	This method is called cyclically by the axis.

4.3 Classification of core functions

Core functions can be classified according to their different characteristics.

4.3.1 Permanently active core functions

A core function from this group does not accept any commands and remains active once it has established and maintained a connection with the axis, i.e. it remains in ReadyState. The state transitions and conditions of the constantly active core functions are shown in the figure below.

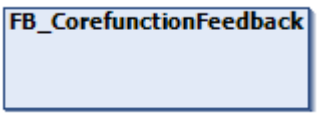


The following core functions are of this type:

Permanently active core functions	Description
Actuals [▶ 33]	Contains the state of the axis.
ActualsHydraulics [▶ 33]	Is inherited from MotionActuals and extends it by hydraulic-specific elements.
ActualsNc [▶ 34]	Is inherited from MotionActuals and extends it by NC-specific elements.
MotionParams [▶ 72]	Provides access to a range of axis parameters.
MotionSetpoints [▶ 74]	Contains the current setpoints of the axis.
PtpLookUp [▶ 92]	Contains the multi-segment PTP move command.
ToolAdaptation [▶ 100]	Contains the parameters of a tool adaptation. It provides methods for the conversion between axis and tool positions.

4.3.2 Commanded core functions

These core functions are activated by a command. The functions of this group are derived from an ABSTRACT FB_CorefunctionFeedback.



Thus all core functions of this group have the following properties:

Name	Type	Description
AbortedState	BOOL	Signals the abort of a command by another core function.
BusyState	BOOL	Signals the active execution of a command.
DoneState	BOOL	Signals the successful execution of a command.
FailedState	BOOL	If IsActivated is TRUE at the same time: Signals the failure of an accepted command. If IsActivated is FALSE at the same time: Signals the rejection of a command.
HasFeedback	BOOL	The core function has responded to a pending command.
IdleState	BOOL	The core function is ready for operation and commandable.
InitState	BOOL	The core function is not completely and successfully initialized.
IsActivated	BOOL	The core function has an accepted command pending.
IsCommanded	BOOL	Signals the pending of a command
IsLocalCmd	BOOL	Signals that the axis is assigned with a command of this core function.
ReadyState	BOOL	The core function is ready for operation, but is not commandable at this time. Possible reasons are: The axis is not released. Another core function is active.

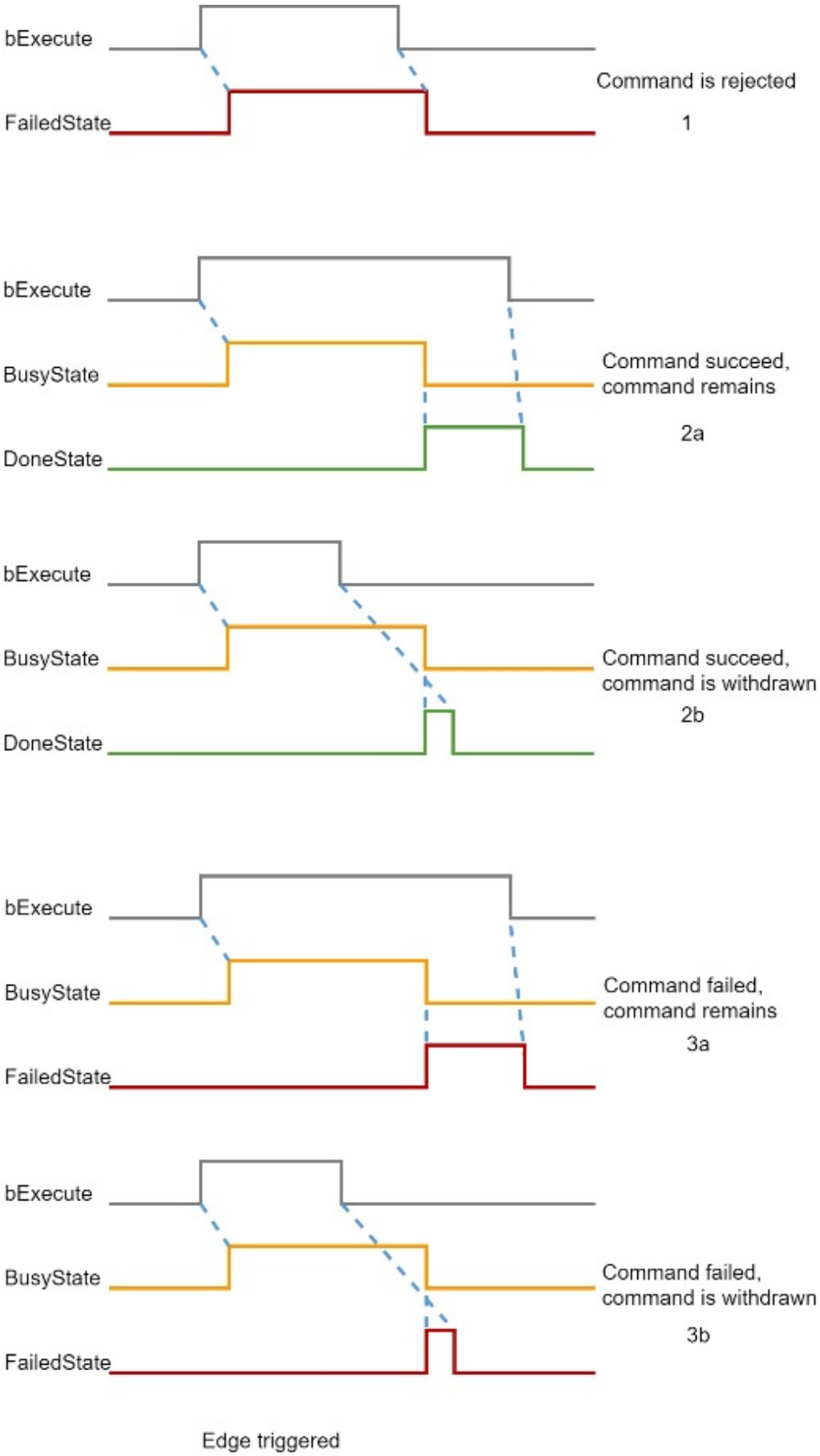
Methods

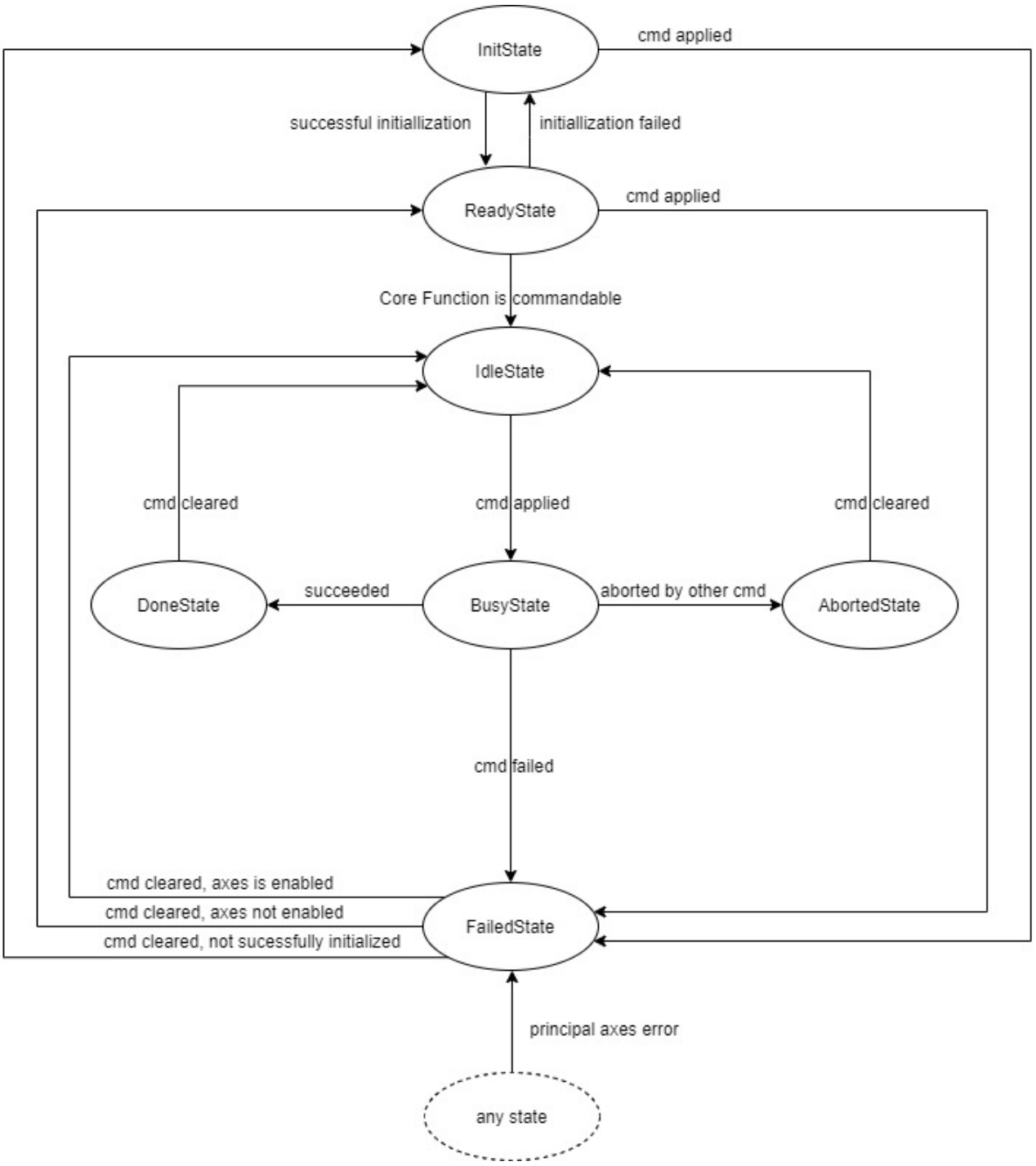
Name	Description
ApplyCommand	This method is used internally by a number of core functions when they start an active access to an axis.
DoReset	This method resets the error message and the FailedState signal of the core function.
RemoveCmd	This method is used internally by a number of core functions when active access to the axis is complete. A call to this method will cause IsLocalCmd to report FALSE.

4.3.2.1 Edge-triggered core functions

The command accepted by these core functions is usually called bExcute (with the exception of Power, which is described below).

The rising edge of bExecute triggers a series of checks to see if the execution conditions are met. If the execution conditions are not met, the command is rejected and the core function is set to FailedState (case 1 in the figure below). If the execution conditions are met, the command is accepted and the core function enters BusyState. The falling edge of bExecute does not directly trigger a response in BusyState (case b below). If the command was executed successfully, the core function enters the DoneState (case 2 below), otherwise it enters the FailedState (case 3 below). At this time it is checked if bExcute is still TRUE and the state of the core function is changed in the next PLC cycle. The state transition is shown in the figure below.


bExecute
FailedState
Command is rejected
1
bExecute
BusyState
DoneState
Command succeed,
command remains
2a
bExecute
BusyState
DoneState
Command succeed,
command is withdrawn
2b
bExecute
BusyState
FailedState
Command failed,
command remains
3a
bExecute
BusyState
FailedState
Command failed,
command is withdrawn
3b
Edge triggered



The following functions are edge-triggered core functions:

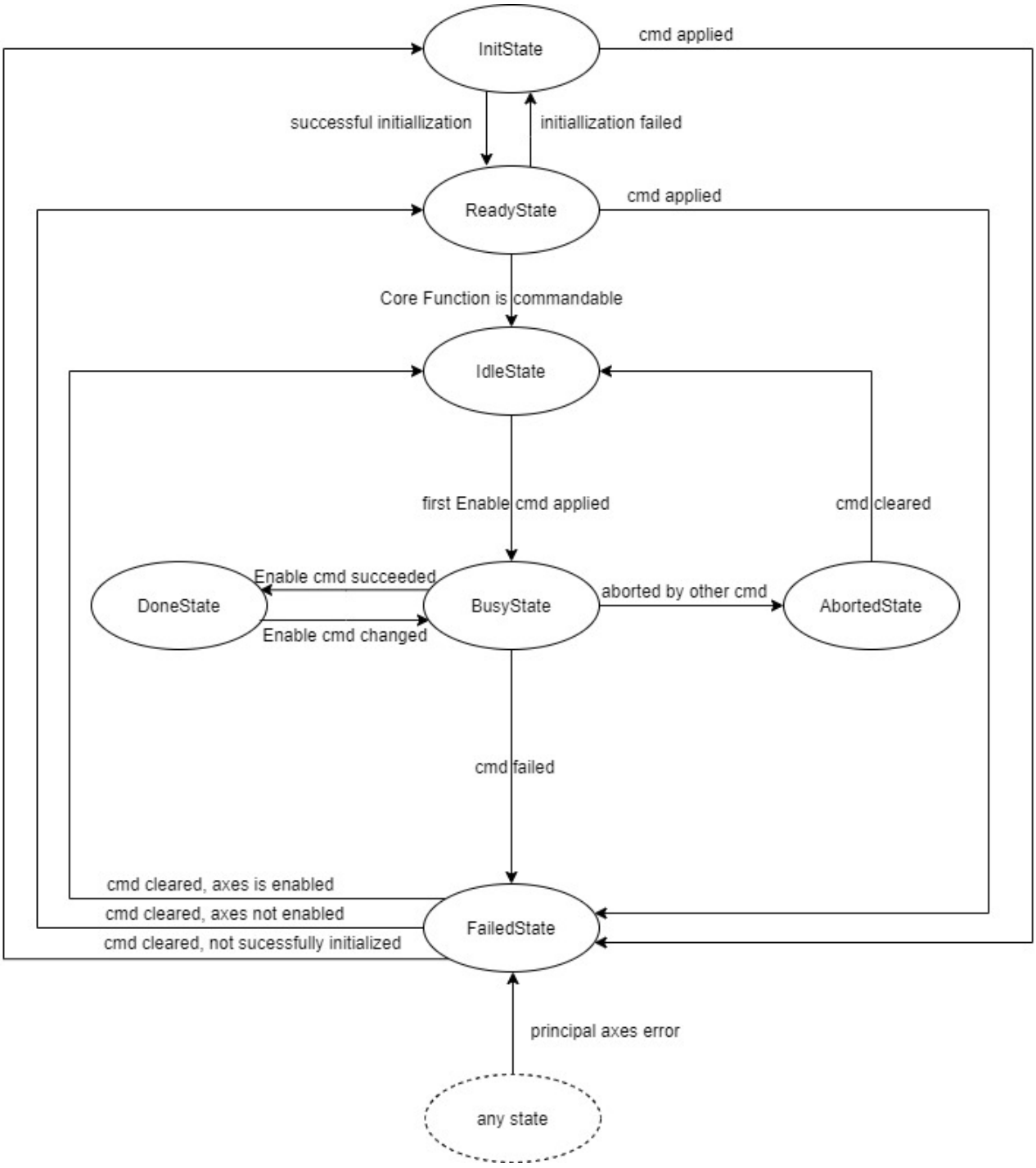
Core function	Description
DisableSoftEnd [▶ 42]	Is used to temporarily disable and re-enable the soft limit switches of the axis.
Homing [▶ 45]	Provides a range of homing methods.
Power [▶ 74]	Used to activate and deactivate the axis.
Ptp [▶ 86]	Is used to perform multi-segment PTP movements.
Reset	Is used to reset the error state of axis and devices.
SetPosition [▶ 96]	Is used to change the actual position of the axis.
Estop [▶ 44]	Command for emergency shutdown using the maximum dynamic parameters to stop the axis.
Stop [▶ 98]	Command for stop with the standard dynamic parameters for stopping the axis.

Core function	Description
TeachPosition	Reserved for future extension.
TeachUpdate	Reserved for future extension.
AutoIdent [▶ 34]	A special core function for hydraulic axes. It is used to identify the characteristic velocity behavior of the axis.
DirectOutput [▶ 41]	A special core function for hydraulic axes. It is used to send output signals directly to the control device.

NOTICE

Power is different

Although power is an edge-triggered core function, the command for power is called bEnable. Power has a different state machine than other edge-triggered core functions: with DoneState it returns to BusyState when the bEnable signal changes.



4.3.2.2 Statically controlled core functions

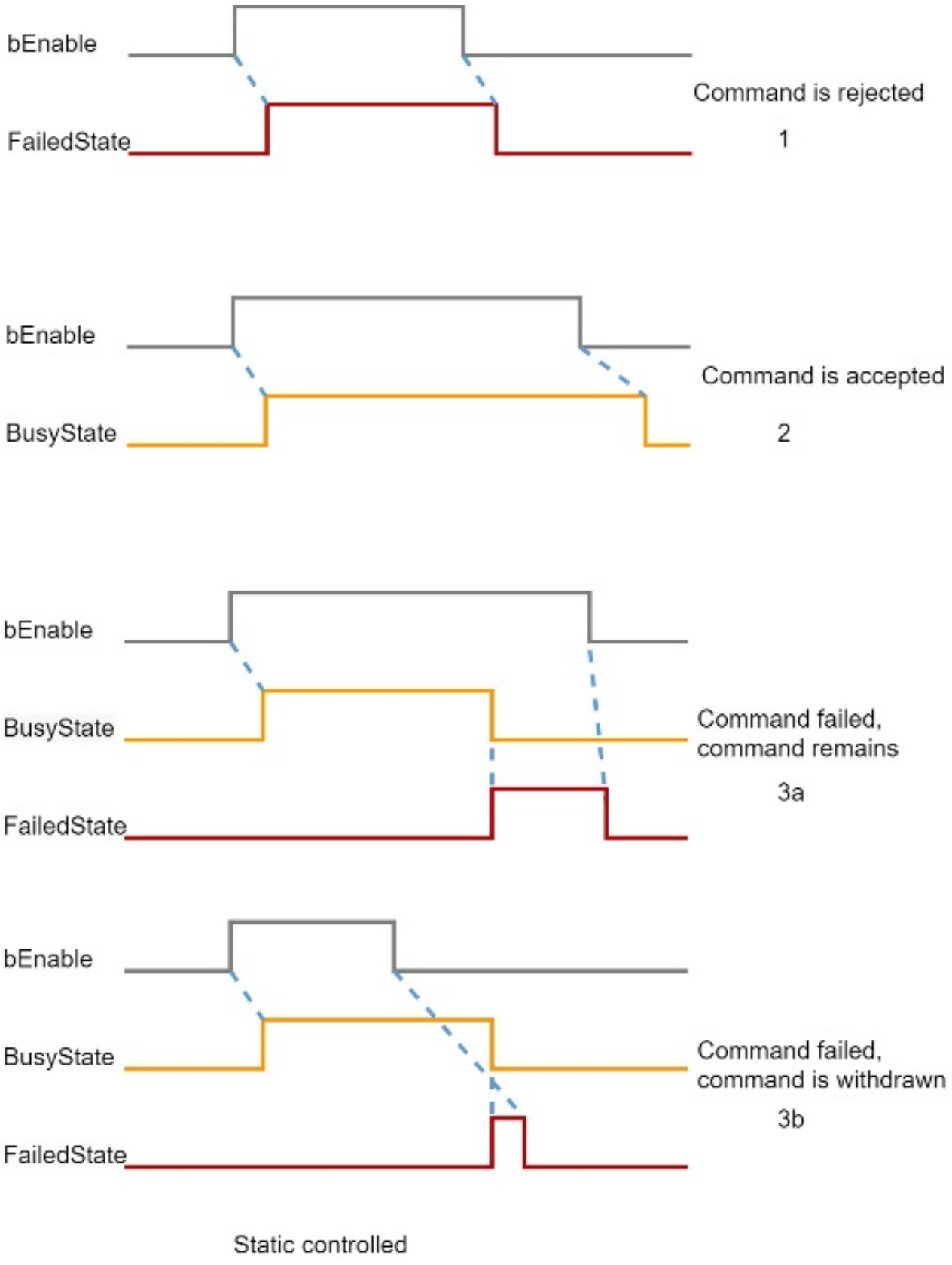
The command that is accepted by these core functions is usually called bEnable.

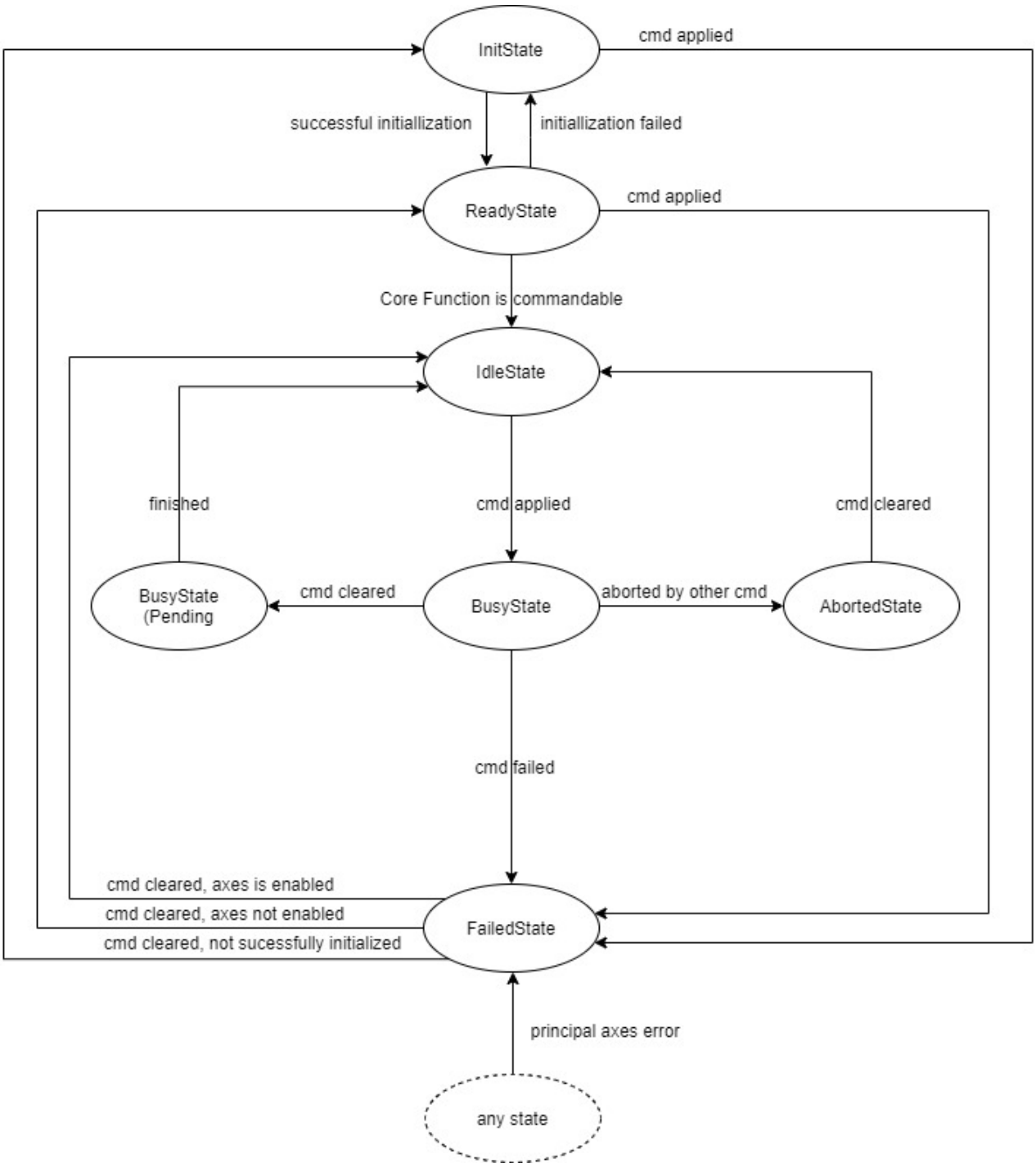
The rising edge of bEnable triggers a series of checks to determine whether the execution conditions are met. If the execution conditions are not met, the command is rejected and the core function enters the FailedState (case 1 in the figure below). If the execution conditions are met, the command is accepted and the core function enters BusyState. The falling edge of bEnable triggers a response to terminate execution (case 2 below).

If an error occurs during the execution of the command, the core function enters the FailedState (case 3 below). At this time it is checked if bEnable is still TRUE and the state of the core function is changed in the next PLC cycle. The state machine is shown in the following figure.

The following core functions are statically controlled:

Core function	Description
Camming [▶ 38]	Is used to activate setpoint generation that is controlled by a guide value.
Jog [▶ 68]	Is used to command a movement without a specified target.
PressureControl	Is used to activate a pressure control circuit.
VelocityFeed [▶ 101]	Is used to activate a movement without a specified target and a velocity that is controlled by a guide value.


bEnable
FailedState
Command is rejected
1
bEnable
BusyState
Command is accepted
2
bEnable
BusyState
FailedState
Command failed,
command remains
3a
bEnable
BusyState
FailedState
Command failed,
command is withdrawn
3b
Static controlled


InitState
cmd applied
successful initiallization
initiallization failed
ReadyState
cmd applied
Core Function is commandable
IdleState
finished
cmd applied
cmd cleared
BusyState (Pending
cmd cleared
BusyState
aborted by other cmd
AbortedState
cmd failed
cmd cleared, axes is enabled
cmd cleared, axes not enabled
cmd cleared, not sucessfully initialized
FailedState
principal axes error
any state

5 Core functions of the axis

A wide range of axis commands is implemented as core function. Each core function is accessible via an axis property and provides an interface. This implementation enables the following use:

```
Axis.CoreFunction.Property := assigned_value;
```

```
hrvar : HRESULT;

hrvar := Axis.CoreFunction.Methode(input:=…);
IF SUCCEEDED( hrvar ) THEN …
IF FAILED( hrvar ) THEN …
```

Core functions are instantiated within axes as local elements. At startup, the core function receives an interface to its host axis and, if necessary, to a drive-related adaptation.

Terms used

Non-functional situation

A situation that does not allow the use of the core function. This may be caused by a failed or missing initialization or any other problem that causes damage to the axis or its sub-components. In this case the axis is permanently in InitState.

Idling situation

In this situation, the axis has no active, failed or completed command pending. As a rule, there are some requirements that must be considered when activating the core function. If it can be activated, it reports IdleState, otherwise ReadyState. Refer to the core function documentation for details.

No command

The core functions of the always active group do not support a command (and do not need one). Consequently, they will not report IdleState at any time.

Active situation

While the core function is actively performing its task, it reports BusyState. This situation ends when a fault is detected in the controlled component or device, or when another function has taken over control. For edge-triggered core functions and some statically controlled core functions, this situation ends when the task is successfully completed. In these cases, the core function is changed to the final situation.

Performed task is aborted

Statically controlled core functions leave this situation by terminating the executed task and falling back into idle mode as soon as the command of the core function is cleared.

Final situation

In this situation, the signals are determined by the result of the previous active situation. A DoneState reports a successfully completed task. A FailedState or AbortedState signals a bad result or the abort by another function.

Check the command input

In the next cycle, after the result of the above active situation has been signaled, the core function starts checking the command input. If the input is FALSE, the core function falls back to idle.

5.1 Actuals

This core function is not instantiated directly. It is used as a common part of type-specific core functions like ActualsHydraulics or ActualsNc.

All derivatives of Actuals are members of the group of permanently active core functions.

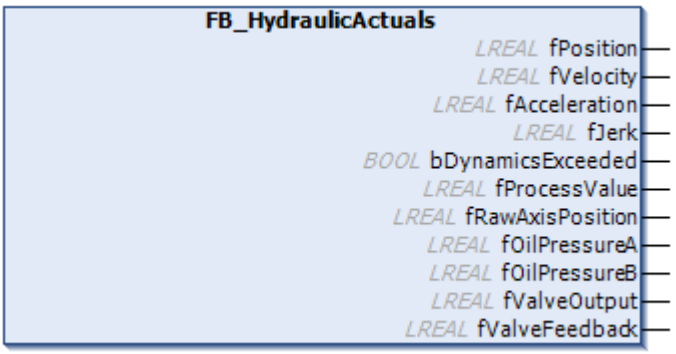
Properties

Name	Type	Description
Acceleration	LREAL	The current actual acceleration.
DynamicsExceeded	BOOL	TRUE if the dynamic limits of the axis were exceeded during a currently active camming. This signal is only deleted when camming is stopped.
Jerk	LREAL	The current actual jerk.
Position	LREAL	The current actual position.
ProcessValue	LREAL	The current actual process value.
RawAxisPosition	LREAL	The unconverted current actual position.
SetEvent	BOOL	Reserved.
Velocity	LREAL	The current actual velocity.

New status interface

In a non-functional situation, the core function reports InitState. Otherwise ReadyState is reported.

5.2 ActualsHydraulics (hydraulics axes only)



This core function is used to display a compilation of actual values of the axis. It is an extension of Actuals and extends the range of its parent element by adding specific values for hydraulic axes.

Like all derivatives of Actuals, this core function belongs to the group of permanently active core functions.

Core function supports all Actuals properties and adds the following elements.

Properties

Name	Type	Description
OilPressureA	LREAL	The current actual pressure on the A-side of the cylinder.
OilPressureB	LREAL	The current actual pressure on the B-side of the cylinder.
ValveFeedback	LREAL	The current feedback value (slider actual position) of the valve.
ValveOutput	LREAL	The current output value (slider set position) for the valve.

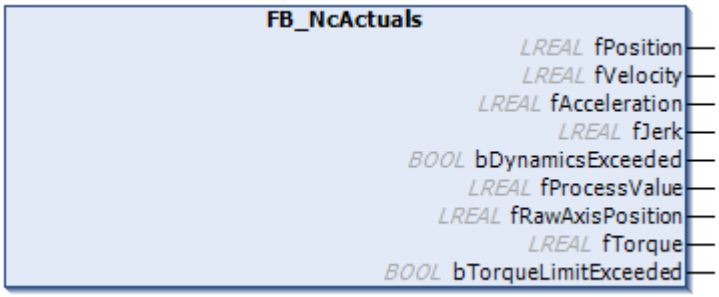
The terms A and B

The terms **A** and **B** are used in the definition of the hydraulics library, i.e. the side of the cylinder that makes the axis move in positive direction is regarded as the **A** side.

Sometimes no feedback

Not all types of proportional valves support this kind of feedback.

5.3 ActualsNc (Nc axes only)



This core function is used to display a compilation of actual values of the axis. It is an extension of Actuals and extends the range of its parent element by adding specific values for NC axes.

Like all derivatives of Actuals, this core function belongs to the group of permanently active core functions.

Core function supports all Actuals properties and adds the following elements.

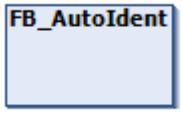
Properties

Name	Type	Description
Torque	LREAL	The current torque actual value.
TorqueLimitExceeded	BOOL	TRUE if the torque limit has been reached.

Clear this property

TorqueLimitExceeded is set by the library, but not cleared. The application must be sure to clear this property at the start of a function that has to be monitored.

5.4 AutoIdent (hydraulic axes only)

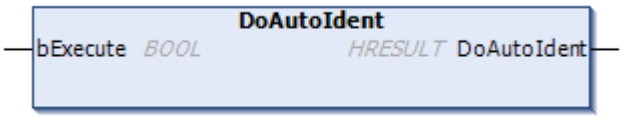


This core function is used to analyze the non-linear transfer characteristic of hydraulic axes. It belongs to the group of edge-triggered core functions.

Methods

Name	Description
DoAutoIdent [▶ 35]	Activates and terminates the measurement procedure.
SetParameter [▶ 36]	Sets the parameters for the measurement procedure.

5.4.1 DoAutoIdent



This method is used to activate the core function.

Syntax:

```
METHOD DoAutoIdent : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoAutoIdent	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

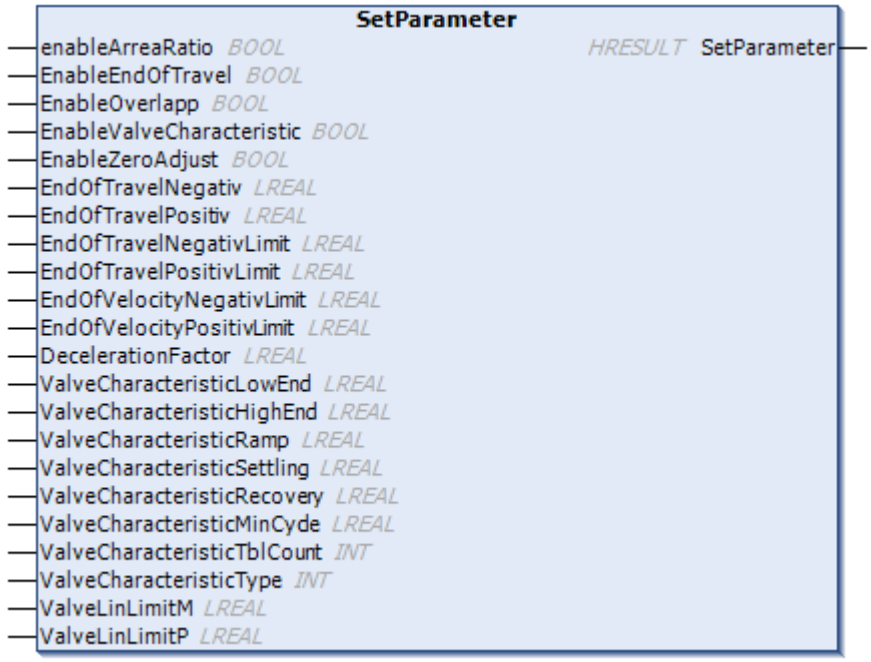
The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge starts the identification. A falling edge cancels a still active identification with an invalid table.

5.4.2 SetParameter



This method can be used to set the parameters for identification.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    EnableArreaRatio           : BOOL;
    EnableEndOfTravel          : BOOL;
    EnableOverlapp             : BOOL;
    EnableValveCharacteristic  : BOOL;
    EnableZeroAdjust           : BOOL;
    EndOfTravelNegativ         : LREAL;
    EndOfTravelPositiv         : LREAL;
    EndOfTravelNegativLimit    : LREAL;
    EndOfTravelPositivLimit    : LREAL;
    EndOfVelocityNegativLimit  : LREAL;
    EndOfVelocityPositivLimit  : LREAL;
    DecelerationFactor         : LREAL;
    ValveCharacteristicLowEnd  : LREAL;
    ValveCharacteristicHighEnd : LREAL;
    ValveCharacteristicRamp    : LREAL;
    ValveCharacteristicSettling: LREAL;
    ValveCharacteristicRecovery: LREAL;
    ValveCharacteristicMinCycle: LREAL;
    ValveCharacteristicTblCount: INT;
    ValveCharacteristicType    : INT;
    ValveLinLimitM             : LREAL;
    ValveLinLimitP             : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

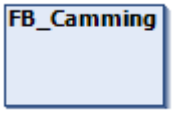
Inputs

Name	Type	Description
EnableArreaRatio	BOOL	A TRUE will request a separate identification of effects caused by cylinder asymmetry.
EnableEndOfTravel	BOOL	A TRUE will request a separate identification of mechanical limits of traveling.
EnableOverlapp	BOOL	A TRUE will request a separate identification of effects caused by valve overlap.
EnableValveCharacteristic	BOOL	A TRUE will request the identification of the valves transfer characteristic.
EnableZeroAdjust	BOOL	A TRUE will request a separate identification of effects caused by valve offset.
EndOfTravelNegativ	LREAL	A mechanical travel limit. This value may be found by identification or entered by using an HMI.
EndOfTravelPositiv	LREAL	A mechanical travel limit. This value may be found by identification or entered by using an HMI.
EndOfTravelNegativLimit	LREAL	This value defines a control value limit. The identification will be concluded for that direction if the output to the control device has reached the limit.
EndOfTravelPositivLimit	LREAL	This value defines a control value limit. The identification will be concluded for that direction if the output to the control device has reached the limit.
EndOfVelocityNegativLimit	LREAL	This value defines an actual velocity limit. The identification will be concluded for that direction if the actual velocity exceeds the limit.
EndOfVelocityPositivLimit	LREAL	This value defines an actual velocity limit. The identification will be concluded for that direction if the actual velocity exceeds the limit.
DecelerationFactor	LREAL	This value defines the limits of travel that will be used for the identification.
ValveCharacteristicLowEnd	LREAL	This value defines the limits of travel that will be used for the identification.
ValveCharacteristicHighEnd	LREAL	This value defines the limits of travel that will be used for the identification.
ValveCharacteristicRamp	LREAL	This parameter defines the ramping to the output value currently under investigation.
ValveCharacteristicSettling	LREAL	This parameter defines the delay for starting the investigation after ramping to the output value.
ValveCharacteristicRecovery	LREAL	This parameter defines a recovery time before continuing the identification in the opposite direction.
ValveCharacteristicMinCycle	LREAL	This value specifies a minimum limit for identification.

Name	Type	Description
ValveCharacteristicTblCount	INT	This parameter defines the number of points in the linearization table. **Notice This value must be an odd number. It must be in the range of 5 to 1001. Recommended values are 101, 201 or 401.**
ValveCharacteristicType	INT	This parameter is reserved to indicate the use of valves with special behavior details. **For more details, see the hydraulics library documentation.**
ValveLinLimitM	LREAL	This value limits the use of the linearization table.
ValveLinLimitP	LREAL	This value limits the use of the linearization table.

5.5 Camming

The property provides two options: Camming.Pos for position camming and Camming.Velo for velocity camming. To do this, two elements of the same type are instantiated, but with different rule settings.



All these functions belong to the group of statically controlled core functions.

Not supported by all axis types

The core function position camming is not supported by inverter axes. Any use will report DEVICE_NOTINIT and trigger an error message.

Properties

Name	Type	Description
Overrun	BOOL	TRUE if the current default value is not within the range of the camming lookup table.
Synchronize	BOOL	With Synchronize=FALSE the axis should follow every setpoint change immediately. A TRUE requires the axis to follow setpoint changes with respect to the dynamic limit parameters.
Synchronized	BOOL	TRUE if Synchronize=TRUE and the setpoint changes are within the limits of the dynamic parameters of the axis.
UseAsPosition	BOOL	This property defines the camming rule. A TRUE causes the core function to act as a position camming. A FALSE will cause it to act as a velocity camming. This property is set during startup. The application must not change its setting.

Methods

Name	Description
DoCamming	This method is used to enable and disable the core function.
SetGuidingValue	This method is used to update the default value.

Name	Description
SetLookupInterface	This method must be used to connect a camming table.

Both types of camming use a default value to identify a control value within a camming lookup table. For details see CammingLookUp.

Any useful information can serve as a default value. Common options are a time (LREAL variable that starts with zero and is cyclically updated by adding the cycle time of the PLC tasks) or the position of another axis.

⚠ WARNING

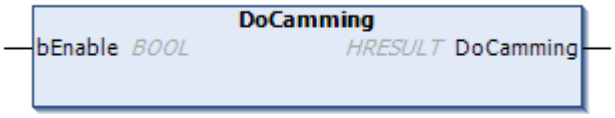
Unexpected responses of the controlled axis

Unsuitable default values or table points can lead to unexpected reactions of the controlled axis. This may result in risk of accident or damage.

Using a camming requires several steps:

- A camming lookup table must be instantiated. There is no rule for the number of these tables. A table used once can be used again at a later time by repeating the following steps.
- The table must be defined by loading the table points. Make sure that the point data corresponds to the camming type (position, velocity) for which it is to be used.
- The properties of the table must be updated.
- The table must be assigned to the camming with the SetLookupinterface() method. At this time, camming must not be commanded.
- An initial default value must be specified with the SetGuidingValue() method.
- At the right time use DoCamming(bEnable:=TRUE) to activate the function. Make sure that the axis is in a situation (position, velocity) that matches the situation of the camming table.
- Make sure that the default value is updated cyclically.
- Use DoCamming(bEnable:=FALSE) at the right time to disable the function. Pay attention to the situation (position, velocity) of the axis.

5.5.1 DoCamming



This method is used to enable and disable the core function.

Syntax:

```
METHOD DoCamming : HRESULT
VAR_INPUT
    bEnable: BOOL;
END_VAR
```

Return value

Name	Type	Description
DoCamming	HRESULT	See below

Inputs

Name	Type	Description
bEnable	BOOL	A TRUE activates the camming. A FALSE ends the camming and slows down a still existing movement.

5.5.2 SetGuidingValue

This method must be used cyclically to update the default value.

Syntax:

```
METHOD SetGuidingValue : HRESULT
VAR_INPUT
    fGuidingValue: LREAL;
END_VAR
```

Return value

Name	Type	Description
SetGuidingValue	HRESULT	See below

Inputs

Name	Type	Description
fGuidingValue	LREAL	The default value valid in the next cycle.

5.5.3 SetLookupInterface

This method must be used to connect a camming table.

Syntax:

```
METHOD SetLookupInterface: HRESULT
VAR_INPUT
    iLookup: I_CammingLookUp;
END_VAR
```

Return value

Name	Type	Description
SetLookupInterface	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

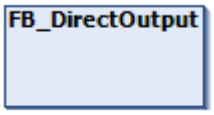
Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
iLookUp	I_CammingLookUp	The camming table to use.

5.6 DirectOutput (hydraulics axes only)



This core function is used to directly control the output of a hydraulic axis. There will be no monitoring of position limits.

This core function is a member of the group of statically-controlled core functions.

Properties

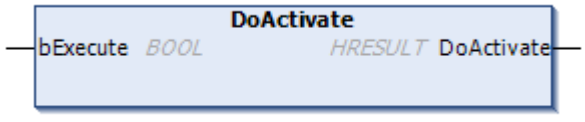
Name	Type	Description
OutputReference	LREAL	This property defines the value that must be specified as OutPutValue to cause a full-scale output to the controlled device.
OutputValue	LREAL	This property is used to define the output to the device. OutputReference as a scaling here.
Ramptime	LREAL	RampTime is used to define the time for ramping from zero to the full-scale value.

Any change in the output whose amount is less than the full-scale value requires a proportional part of RampTime.

Methods

Name	Description
DoActivate	This method enables and disables the direct output.

5.6.1 DoActivate



This method enables and disables the direct output.

Syntax:

```
METHOD DoActivate : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoActivate	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bExecute	BOOL	A TRUE enables the output. A FALSE ramps the output to zero and disables the core function.

5.7 DisableSoftEnd

This core function is used to temporarily disable and re-enable the software position limits of the axis.

DisableSoftEnd is a member of the group of edge-triggered core functions.

Not supported by all axis types

This core function is not supported by inverter axes. Any use will report DEVICE_NOTINIT and trigger an error message.

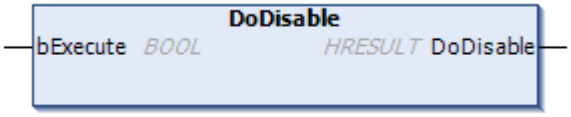
Properties

Name	Type	Description
Disabled	BOOL	A TRUE signal reports active disabling.

Methods

Name	Description
DoDisable [▶ 43]	This method disables the set software position limits.
ReEnable [▶ 43]	This method re-enables the set software position limits.

5.7.1 DoDisable



This method disables the set software position limits.

Syntax:

```
METHOD DoDisable:
HRESULT
VAR_INPUT
    bExecute: BOOL;
END_VAR
```

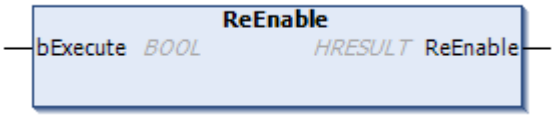
Return value

Name	Type	Description
DoDisable	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers the disabling.

5.7.2 ReEnable



This method re-enables the set software position limits.

Syntax:

```
METHOD ReEnable: HRESULT
VAR_INPUT
    bExecute   : BOOL;
END_VAR
```

Return value

Name	Type	Description
ReEnable	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers the re-enabling.

5.8 Estop

This core function is used to trigger an emergency shutdown operation. It will use the maximum dynamic parameters that are allowed for this axis by the underlying motion technology.

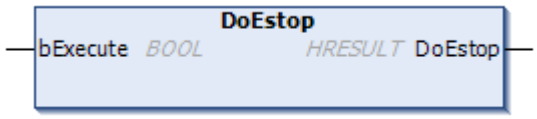
Properties

Name	Type	Description
NoCreeping	BOOL	A TRUE in this property avoids the creep phase at the end of the stopping process of the hydraulics library.

Methods

Name	Description
DoEstop [▶ 45]	A rising edge triggers the stop.

5.8.1 DoEstop



This method is used to trigger the stop.

Syntax:

```
METHOD DoEstop : HRESULT
VAR_INPUT
    bExecute: BOOL;
END_VAR
```

Return value

Name	Type	Description
DoEstop	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge at this input triggers the stop.

5.9 Homing

The property offers access to a range of homing functions.

All these functions belong to the group of edge-triggered core functions.

Not supported by all axis types

This core function is not supported by inverter axes. Any use will report DEVICE_NOTINIT and trigger an error message.

There are two groups of homing functions: Initiating or continuing functions (AbsoluteSwitch, AbsoluteSwitchDetect, Block, BlockDetect, LimitSwitch, LimitSwitchDetect) and terminating functions (Abort, Finish).

Triggering a function of the first group changes the behavior of the axis by activating the homing mode. If this change has already been made by another function in this group, this has no effect. In this mode, the underlying motion technology disables a number of mechanisms such as lag monitoring, velocity pre-control, software position limit switch, etc.

NOTICE

Unexpected behavior

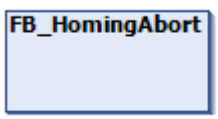
Axes in homing mode may respond to motion commands in an unexpected manner.

As the last step of a homing procedure, the functions of the second group are used to terminate the homing mode and set the axis to a normal behavior.

Properties

Name	Type	Description
Abort [▶ 46]	I_HomingAbort	An active homing is aborted without success.
AbsoluteSwitch [▶ 48]	I_HomingAbsoluteSwitch	Homing is performed at a position reported by a binary sensor.
AbsoluteSwitchDetect [▶ 51]	I_HomingAbsoluteSwitchDetection	The actual position is recorded at a position reported by a binary sensor.
Block [▶ 54]	I_HomingBlock	Homing is performed on a mechanical stop.
BlockDetect [▶ 57]	I_HomingBlockDetection	The actual position is recorded at a mechanical stop.
Finish [▶ 61]	I_HomingFinish	An active homing is completed successfully.
LimitSwitch [▶ 63]	I_HomingLimitSwitch	Homing is performed at a position reported by a hardware limit switch.
LimitSwitchDetect [▶ 66]	I_HomingLimitSwitchDetection	The actual position is recorded at a position reported by a hardware limit switch.

5.9.1 Abort



This core function can be used to abort a homing in case of a problem.

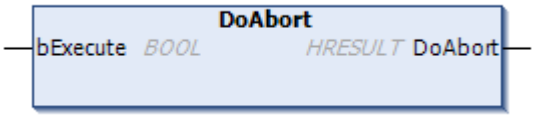
Abort required

Abort is also required if a homing function fails.

Methods

Name	Description
DoAbort [▶ 47]	A rising edge triggers the abort.
SetParameter [▶ 47]	The procedure for returning to normal operation can be specified.

5.9.1.1 DoAbort



This method triggers the abort in case of a rising edge.

Syntax:

```
METHOD DoAbort : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoAbort	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers the abort.

5.9.1.2 SetParameter

The procedure for returning to normal operation can be specified.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    bOptionsDisableDriveAccess : BOOL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

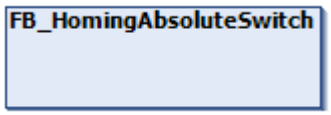
Inputs

Name	Type	Description
bOptionsDisableDriveAccess	BOOL	A TRUE prevents ADS communication via the fieldbus from being used to change parameters of the controlled device in order to return to normal operation.

Non-Beckhoff servo drives

This option must be used for most non-Beckhoff servo drives.

5.9.2 AbsoluteSwitch



This core function performs homing by searching for a fixed edge of a signal. A specified position is set at this location.

Properties

Name	Type	Description
AbsoluteSwitch	BOOL	This signal indicates the homing position.
NegativeLimitSwitch	BOOL	The hardware limit switch at the lower end of the available travel path.
PositiveLimitSwitch	BOOL	The hardware limit switch at the upper end of the available travel path.

The action of the limit switches is defined by PLCopen standards.

Methods

Name	Description
DoHoming [▶ 49]	This method triggers the homing.
SetParameter [▶ 49]	A set of parameters specific to this homing procedure is set.
SetParameterGeneral [▶ 50]	A set of travel parameters for the homing procedure is set.

5.9.2.1 DoHoming

This method triggers the homing.

Syntax:

```
METHOD DoHoming : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoHoming	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers homing.

5.9.2.2 SetParameter

A set of parameters specific to this homing procedure is set.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    fSetPosition                  : LREAL;
    eSwitchMode                   : E_AdaptableSwitchMode;
    bOptionsDisableDriveAccess    : BOOL;
    bOptionsEnableLagErrorDetection : BOOL;
END_VAR
```

Return value

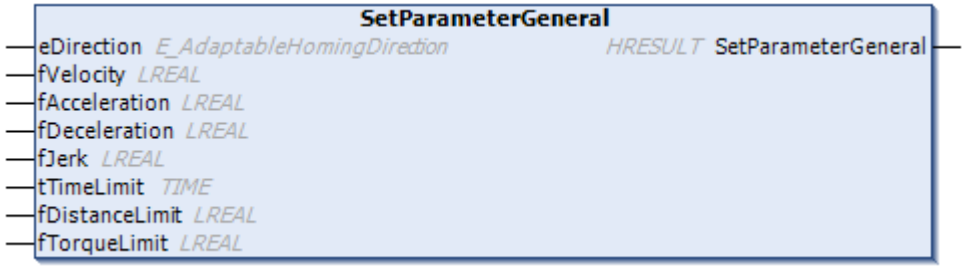
Name	Type	Description
SetParameter	HRESULT	See below

Inputs

Name	Description
fSetPosition	Specifies the value that is applied to the actual position at the homing event.
eSwitchMode	Defines how the AbsoluteSwitch is interpreted as a homing event.

Name	Description
bOptionsDisableDriveAccess	A TRUE prevents the underlying motion technology from using ADS communication via the fieldbus to change parameters of the controlled device and return to normal operation. **Notice This option must be used for most non-Beckhoff servo drives.**
bOptionsEnableLagErrorDetection	A TRUE will enable the lag error detection while the function is being executed.

5.9.2.3 SetParameterGeneral



A set of travel parameters for the homing procedure is set.

Syntax:

```
METHOD SetParameterGeneral : HRESULT
VAR_INPUT
    eDirection     : E_AdaptableHomingDirection;
    fVelocity      : LREAL;
    fAcceleration  : LREAL;
    fDeceleration  : LREAL;
    fJerk          : LREAL;
    tTimeLimit     : TIME;
    fDistanceLimit : LREAL;
    fTorqueLimit   : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameterGeneral	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.

Return value	Cause
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

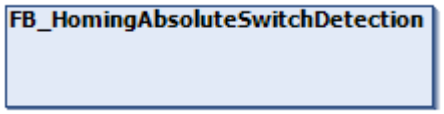
Inputs

Name	Type	Description
eDirection	E_AdaptableHomingDirection	The movement with which the homing event is found.
fVelocity	LREAL	The velocity of the movement used in the search of the homing event.
fAcceleration	LREAL	The acceleration of the movement used in the search of the homing event.
fDeceleration	LREAL	The deceleration of the movement used in the search of the homing event.
fJerk	LREAL	The jerk of the movement used in the search of the homing event.
tTimeLimit	TIME	The timeout limit of the core function.
fDistanceLimit	LREAL	The maximum distance that may be traveled in the search of the homing event.
fTorqueLimit	LREAL	The limit of the torque applied in the search of the homing event.

Torque limitation in the event of a mechanical blockage

The torque limitation is used to prevent damage if the axis encounters a mechanical blockage without having detected the homing event.

5.9.3 AbsoluteSwitchDetect



This core function performs homing by searching for a fixed edge of a signal. At this location the actual position is latched and reported.

Properties

Name	Type	Description
AbsoluteSwitch	BOOL	This signal indicates the homing position.
NegativeLimitSwitch	BOOL	The hardware limit switch at the lower end of the available travel path.
PositiveLimitSwitch	BOOL	The hardware limit switch at the upper end of the available travel path.
RecordedPosition	LREAL	The position latched at the location of the signal.

The action of the limit switches is defined by PLCopen standards.

Methods

Name	Description
DoHoming [▶ 52]	This method triggers the homing.
SetParameter [▶ 52]	A set of parameters specific to this homing procedure is set.
SetParameterGeneral [▶ 53]	A set of travel parameters for the homing procedure is set.

5.9.3.1 DoHoming

This method triggers the homing.

Syntax:

```
METHOD DoHoming : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoHoming	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers homing.

5.9.3.2 SetParameter

A set of parameters specific to this homing procedure is set.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    eSwitchMode                   : E_AdaptableSwitchMode;
    bOptionsDisableDriveAccess    : BOOL;
    bOptionsEnableLagErrorDetection : BOOL;
END_VAR
```

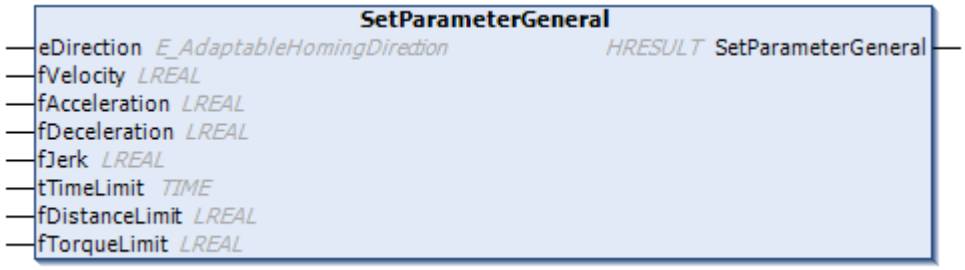
Return value

Name	Type	Description
SetParameter	HRESULT	See below

Inputs

Name	Description
eSwitchMode	Defines how the AbsoluteSwitch is interpreted as a homing event.
bOptionsDisableDriveAccess	A TRUE prevents the underlying motion technology from using ADS communication via the fieldbus to change parameters of the controlled device and return to normal operation. **Notice This option must be used for most non-Beckhoff servo drives.**
bOptionsEnableLagErrorDetection	A TRUE will enable the lag error detection while the function is being executed.

5.9.3.3 SetParameterGeneral



A set of travel parameters for the homing procedure is set.

Syntax:

```
METHOD SetParameterGeneral : HRESULT
VAR_INPUT
    eDirection     : E_AdaptableHomingDirection;
    fVelocity      : LREAL;
    fAcceleration  : LREAL;
    fDeceleration  : LREAL;
    fJerk          : LREAL;
    tTimeLimit     : TIME;
    fDistanceLimit : LREAL;
    fTorqueLimit   : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameterGeneral	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
eDirection	E_AdaptableHomingDirection	The movement with which the homing event is found.
fVelocity	LREAL	The velocity of the movement used in the search of the homing event.
fAcceleration	LREAL	The acceleration of the movement used in the search of the homing event.
fDeceleration	LREAL	The deceleration of the movement used in the search of the homing event.
fJerk	LREAL	The jerk of the movement used in the search of the homing event.
tTimeLimit	TIME	The timeout limit of the core function.
fDistanceLimit	LREAL	The maximum distance that may be traveled in the search of the homing event.
fTorqueLimit	LREAL	The limit of the torque applied in the search of the homing event.

Torque limitation in the event of a mechanical blockage

The torque limitation is used to prevent damage if the axis encounters a mechanical blockage without having detected the homing event.

5.9.4 Block

This core function performs homing by searching for a mechanical stop. At this location the actual position is latched and reported.

Methods

Name	Description
DoHoming [▶ 55]	This method triggers the homing.
SetParameter [▶ 55]	A set of parameters specific to this homing procedure is set.
SetParameterGeneral [▶ 56]	A set of travel parameters for the homing procedure is set.

5.9.4.1 DoHoming

This method triggers the homing.

Syntax:

```
METHOD DoHoming : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

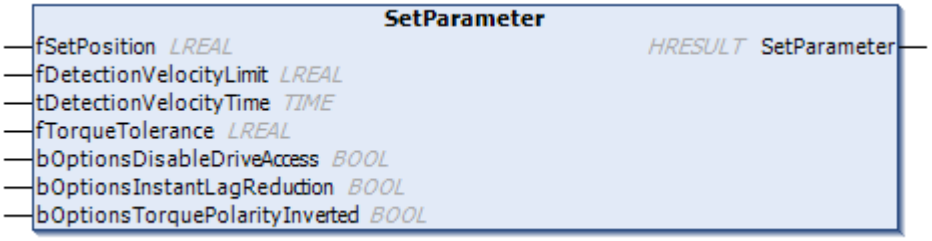
Return value

Name	Type	Description
DoHoming	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers homing.

5.9.4.2 SetParameter



A set of parameters specific to this homing procedure is set.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    fSetPosition                : LREAL;
    fDetectionVelocityLimit     : LREAL;
    tDetectionVelocityTime      : TIME;
    fTorqueTolerance            : LREAL;
    bOptionsDisableDriveAccess  : BOOL;
    bOptionsInstantLagReduction : BOOL;
    bOptionsTorquePolarityInverted : BOOL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Inputs

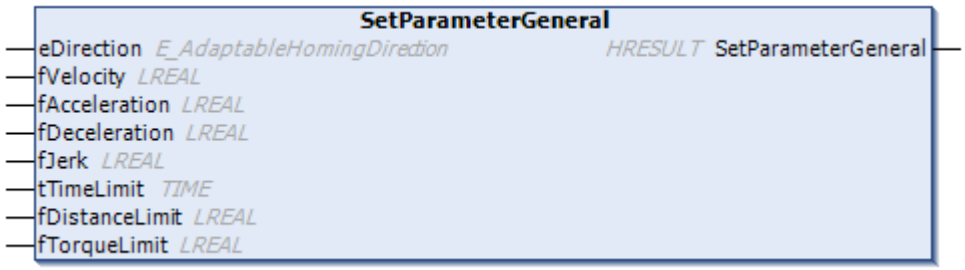
Name	Description
fSetPosition	The position to be assigned to the homing event.

Name	Description
fDetectionVelocityLimit	A velocity threshold for the detection of the homing event. For details see below.
tDetectionVelocityTime	A filter time for the detection of the homing event. For details see below.
fTorqueTolerance	A torque threshold for the detection of the homing event. For details see below.
bOptionsDisableDriveAccess	A TRUE prevents the underlying motion technology from using ADS communication via the fieldbus to change parameters of the controlled device and return to normal operation. **Notice This option must be used for most non-Beckhoff servo drives.**
bOptionsInstantLagReduction	A TRUE causes the lag error to be cleared (set position := actual position) when the homing event is detected.
bOptionsTorquePolarityInverted	A TRUE causes an inverted evaluation of the torque. **Notice This inversion must be used if the signs of the torque and the direction of movement do not match. This may be caused by direction-reversing mechanics (gears, etc.).**

The homing event

A mechanical block as a homing event is detected if at the same time the torque is below the torque limit by less than fTorqueTolerance and the actual velocity has been continuously below fDetectionVelocityLimit since tDetectionVelocityTime.

5.9.4.3 SetParameterGeneral



A set of travel parameters for the homing procedure is set.

Syntax:

```
METHOD SetParameterGeneral : HRESULT
VAR_INPUT
    eDirection     : E_AdaptableHomingDirection;
    fVelocity      : LREAL;
    fAcceleration  : LREAL;
    fDeceleration  : LREAL;
    fJerk          : LREAL;
    tTimeLimit     : TIME;
    fDistanceLimit : LREAL;
    fTorqueLimit   : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameterGeneral	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
eDirection	E_AdaptableHomingDirection	The movement with which the homing event is found.
fVelocity	LREAL	The velocity of the movement used in the search of the homing event.
fAcceleration	LREAL	The acceleration of the movement used in the search of the homing event.
fDeceleration	LREAL	The deceleration of the movement used in the search of the homing event.
fJerk	LREAL	The jerk of the movement used in the search of the homing event.
tTimeLimit	TIME	The timeout limit of the core function.
fDistanceLimit	LREAL	The maximum distance that may be traveled in the search of the homing event.
fTorqueLimit	LREAL	The limit of the torque applied in the search of the homing event.

Torque limitation in the event of a mechanical blockage

The torque limitation is used to prevent damage if the axis encounters a mechanical blockage without having detected the homing event.

5.9.5 BlockDetect

This core function performs homing by searching for a mechanical stop. At this location the actual position is latched and reported.

Properties

Name	Type	Description
RecordedPosition	LREAL	The position latched at the location of the signal.

Methods

Name	Description
DoHoming [▶ 58]	This method triggers the homing.
SetParameter [▶ 58]	A set of parameters specific to this homing procedure is set.
SetParameterGeneral [▶ 59]	A set of travel parameters for the homing procedure is set.

5.9.5.1 DoHoming

This method triggers the homing.

Syntax:

```
METHOD DoHoming : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

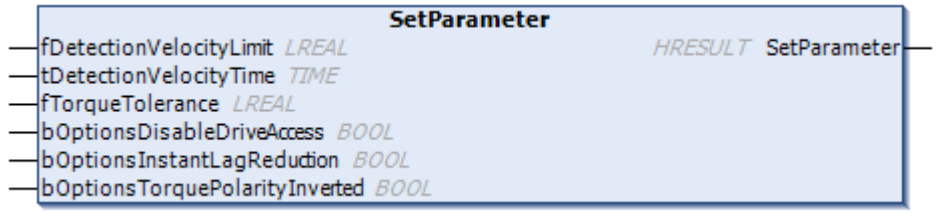
Return value

Name	Type	Description
DoHoming	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers homing.

5.9.5.2 SetParameter



A set of parameters specific to this homing procedure is set.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    fDetectionVelocityLimit        : LREAL;
    tDetectionVelocityTime         : TIME;
    fTorqueTolerance               : LREAL;
    bOptionsDisableDriveAccess     : BOOL;
```

```
    bOptionsInstantLagReduction    : BOOL;
    bOptionsTorquePolarityInverted : BOOL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

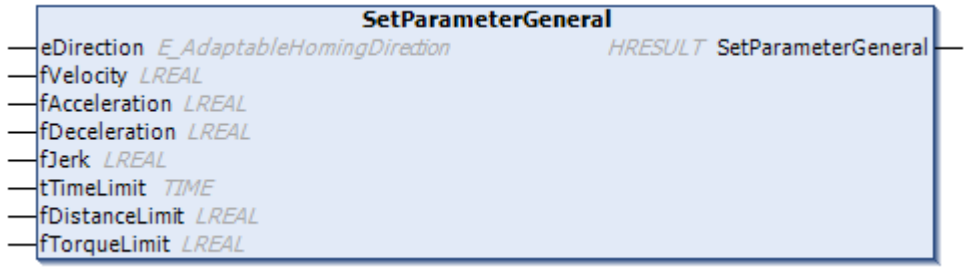
Inputs

Name	Description
fDetectionVelocityLimit	A velocity threshold for the detection of the homing event. For details see below.
tDetectionVelocityTime	A filter time for the detection of the homing event. For details see below.
fTorqueTolerance	A torque threshold for the detection of the homing event. For details see below.
bOptionsDisableDriveAccess	A TRUE prevents the underlying motion technology from using ADS communication via the fieldbus to change parameters of the controlled device and return to normal operation. **Notice This option must be used for most non-Beckhoff servo drives.**
bOptionsInstantLagReduction	A TRUE causes the lag error to be cleared (set position := actual position) when the homing event is detected.
bOptionsTorquePolarityInverted	A TRUE causes an inverted evaluation of the torque. **Notice This inversion must be used if the signs of the torque and the direction of movement do not match. This may be caused by direction-reversing mechanics (gears, etc.).**

The homing event

A mechanical block as a homing event is detected if at the same time the torque is below the torque limit by less than fTorqueTolerance and the actual velocity has been continuously below fDetectionVelocityLimit since tDetectionVelocityTime.

5.9.5.3 SetParameterGeneral



A set of travel parameters for the homing procedure is set.

Syntax:

```
METHOD SetParameterGeneral : HRESULT
VAR_INPUT
    eDirection     : E_AdaptableHomingDirection;
    fVelocity      : LREAL;
    fAcceleration  : LREAL;
    fDeceleration  : LREAL;
    fJerk          : LREAL;
```

```
    tTimeLimit     : TIME;
    fDistanceLimit : LREAL;
    fTorqueLimit   : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameterGeneral	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

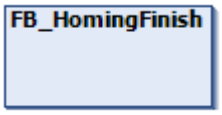
Inputs

Name	Type	Description
eDirection	E_AdaptableHomingDirection	The movement with which the homing event is found.
fVelocity	LREAL	The velocity of the movement used in the search of the homing event.
fAcceleration	LREAL	The acceleration of the movement used in the search of the homing event.
fDeceleration	LREAL	The deceleration of the movement used in the search of the homing event.
fJerk	LREAL	The jerk of the movement used in the search of the homing event.
tTimeLimit	TIME	The timeout limit of the core function.
fDistanceLimit	LREAL	The maximum distance that may be traveled in the search of the homing event.
fTorqueLimit	LREAL	The limit of the torque applied in the search of the homing event.

Torque limitation in the event of a mechanical blockage

The torque limitation is used to prevent damage if the axis encounters a mechanical blockage without having detected the homing event.

5.9.6 Finish



This core function must be used to successfully complete a homing.

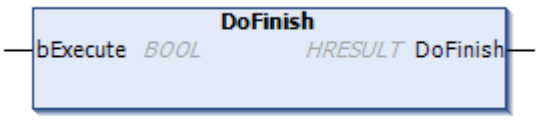
Abort required

Abort is required if a homing function fails.

Methods

Name	Description
DoFinish [▶ 61]	A rising edge triggers the termination.
SetParameter [▶ 61]	The procedure for returning to normal operation can be specified.

5.9.6.1 DoFinish



This method triggers the termination on a rising edge.

Syntax:

```
METHOD DoFinish : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

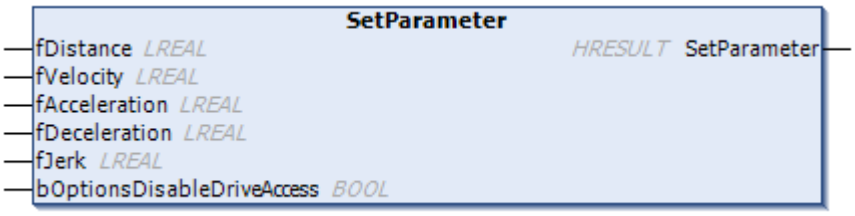
Return value

Name	Type	Description
DoFinish	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers the termination.

5.9.6.2 SetParameter



The procedure for returning to normal operation can be specified.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    fDistance : LREAL;
    fVelocity : LREAL;
    fAcceleration : LREAL;
    fDeceleration : LREAL;
    fJerk : LREAL;
    bOptionsDisableDriveAccess : BOOL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

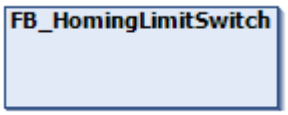
Inputs

Name	Type	Description
fDistance	LREAL	Here you can define a distance by which the axis should move away from the homing position. ***Notice*** **In many cases, there is a torque at the end of a homing procedure. The mechanics of the axis can be relieved by a movement suitably selected in amount and direction.**
fVelocity	LREAL	The velocity to be commanded for this.
fAcceleration	LREAL	The acceleration to be commanded for this.
fDeceleration	LREAL	The deceleration to be commanded for this.
fJerk	LREAL	The jerk to be commanded for this.
bOptionsDisableDriveAccess	BOOL	A TRUE prevents ADS communication via the fieldbus from being used to change parameters of the controlled device in order to return to normal operation.

Non-Beckhoff servo drives

The bOptionsDisableDriveAccess option must be used for most non-Beckhoff servo drives.

5.9.7 LimitSwitch



This core function performs homing by searching for a fixed edge of a signal. A specified position is set at this location.

Properties

Name	Type	Description
NegativeLimitSwitch	BOOL	The hardware limit switch at the lower end of the available travel path.
PositiveLimitSwitch	BOOL	The hardware limit switch at the upper end of the available travel path.

The action of the limit switches is defined by PLCopen standards.

Methods

Name	Description
DoHoming [▶ 63]	This method triggers the homing.
SetParameter [▶ 64]	A set of parameters specific to this homing procedure is set.
SetParameterGeneral [▶ 64]	A set of travel parameters for the homing procedure is set.

5.9.7.1 DoHoming

This method triggers the homing.

Syntax:

```
METHOD DoHoming : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoHoming	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers homing.

5.9.7.2 SetParameter

A set of parameters specific to this homing procedure is set.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    fSetPosition                    : LREAL;
    eSwitchMode                     : E_AdaptableSwitchMode;
    bOptionsDisableDriveAccess      : BOOL;
    bOptionsEnableLagErrorDetection : BOOL;
END_VAR
```

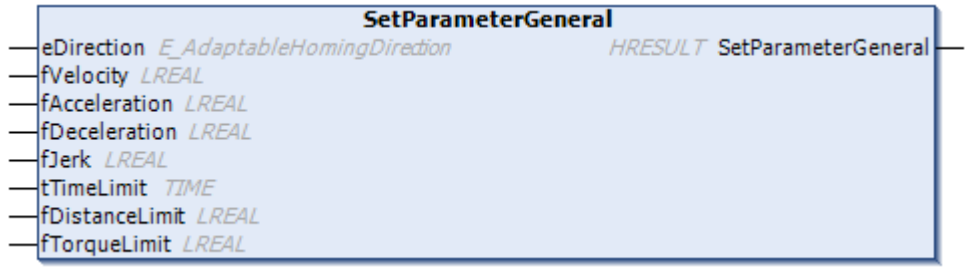
Return value

Name	Type	Description
SetParameter	HRESULT	See below

Inputs

Name	Description
fSetPosition	Specifies the value that is applied to the actual position at the homing event.
eSwitchMode	Defines how the AbsoluteSwitch is interpreted as a homing event.
bOptionsDisableDriveAccess	A TRUE prevents the underlying motion technology from using ADS communication via the fieldbus to change parameters of the controlled device and return to normal operation. **This option must be used for most non-Beckhoff servo drives.**
bOptionsEnableLagErrorDetection	A TRUE will enable the lag error detection while the function is being executed.

5.9.7.3 SetParameterGeneral



A set of travel parameters for the homing procedure is set.

Syntax:

```
METHOD SetParameterGeneral : HRESULT
VAR_INPUT
    eDirection     : E_AdaptableHomingDirection;
    fVelocity      : LREAL;
    fAcceleration  : LREAL;
    fDeceleration  : LREAL;
```

```
    fJerk          : LREAL;
    tTimeLimit     : TIME;
    fDistanceLimit : LREAL;
    fTorqueLimit   : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameterGeneral	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

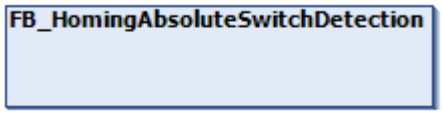
Inputs

Name	Type	Description
eDirection	E_AdaptableHomingDirection	The movement with which the homing event is found.
fVelocity	LREAL	The velocity of the movement used in the search of the homing event.
fAcceleration	LREAL	The acceleration of the movement used in the search of the homing event.
fDeceleration	LREAL	The deceleration of the movement used in the search of the homing event.
fJerk	LREAL	The jerk of the movement used in the search of the homing event.
tTimeLimit	TIME	The timeout limit of the core function.
fDistanceLimit	LREAL	The maximum distance that may be traveled in the search of the homing event.
fTorqueLimit	LREAL	The limit of the torque applied in the search of the homing event.

Torque limitation in the event of a mechanical blockage

The torque limitation is used to prevent damage if the axis encounters a mechanical blockage without having detected the homing event.

5.9.8 LimitSwitchDetect



This core function performs homing by searching for a fixed edge of a signal. At this location the actual position is latched and reported.

Properties

Name	Type	Description
NegativeLimitSwitch	BOOL	The hardware limit switch at the lower end of the available travel path.
PositiveLimitSwitch	BOOL	The hardware limit switch at the upper end of the available travel path.
RecordedPosition	LREAL	The position latched at the location of the signal.

The action of the limit switches is defined by PLCopen standards.

Methods

Name	Description
DoHoming [▶ 66]	This method triggers the homing.
SetParameter [▶ 67]	A set of parameters specific to this homing procedure is set.
SetParameterGeneral [▶ 67]	A set of travel parameters for the homing procedure is set.

5.9.8.1 DoHoming

This method triggers the homing.

Syntax:

```
METHOD DoHoming : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoHoming	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers homing.

5.9.8.2 SetParameter

A set of parameters specific to this homing procedure is set.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    eSwitchMode                    : E_AdaptableSwitchMode;
    bOptionsDisableDriveAccess     : BOOL;
    bOptionsEnableLagErrorDetection : BOOL;
END_VAR
```

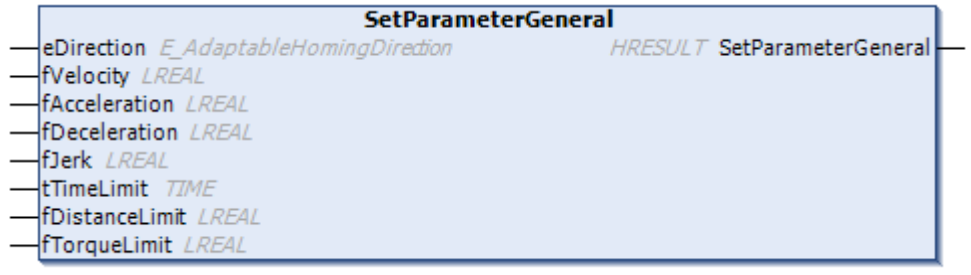
Return value

Name	Type	Description
SetParameter	HRESULT	See below

Inputs

Name	Description
eSwitchMode	Defines how the AbsoluteSwitch is interpreted as a homing event.
bOptionsDisableDriveAccess	A TRUE prevents the underlying motion technology from using ADS communication via the fieldbus to change parameters of the controlled device and return to normal operation. **This option must be used for most non-Beckhoff servo drives.**
bOptionsEnableLagErrorDetection	A TRUE will enable the lag error detection while the function is being executed.

5.9.8.3 SetParameterGeneral



A set of travel parameters for the homing procedure is set.

Syntax:

```
METHOD SetParameterGeneral : HRESULT
VAR_INPUT
    eDirection     : E_AdaptableHomingDirection;
    fVelocity      : LREAL;
    fAcceleration  : LREAL;
    fDeceleration  : LREAL;
    fJerk          : LREAL;
    tTimeLimit     : TIME;
    fDistanceLimit : LREAL;
    fTorqueLimit   : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameterGeneral	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
eDirection	E_AdaptableHomingDirection	The movement with which the homing event is found.
fVelocity	LREAL	The velocity of the movement used in the search of the homing event.
fAcceleration	LREAL	The acceleration of the movement used in the search of the homing event.
fDeceleration	LREAL	The deceleration of the movement used in the search of the homing event.
fJerk	LREAL	The jerk of the movement used in the search of the homing event.
tTimeLimit	TIME	The timeout limit of the core function.
fDistanceLimit	LREAL	The maximum distance that may be traveled in the search of the homing event.
fTorqueLimit	LREAL	The limit of the torque applied in the search of the homing event.

Torque limitation in the event of a mechanical blockage

The torque limitation is used to prevent damage if the axis encounters a mechanical blockage without having detected the homing event.

5.10 Jog

This core function is used to start and stop the axis with a certain velocity but without a defined target position.

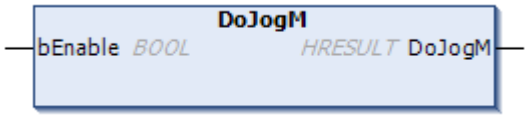
Properties

Name	Type	Description
IsJoggingM	BOOL	A TRUE signals an active jogging in negative direction.
IsJoggingP	BOOL	A TRUE signals an active jogging in positive direction.
IsStopping	BOOL	A TRUE signals the stopping of the axis after an active jogging.
TorqueLimiting	LREAL	This sets the torque limit during jogging.

Methods

Name	Description
DoJogM [▶ 69]	This method performs the start and stop of the axis in the negative direction.
DoJogP [▶ 70]	This method performs the start and stop of the axis in the positive direction.
SetParameter [▶ 71]	This method is used to set the parameters of a movement in jogging mode.

5.10.1 DoJogM



This method performs the start and stop of the axis in the negative direction.

Syntax:

```
METHOD DoJogM : HRESULT
VAR_INPUT
    bEnable : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoJogM	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

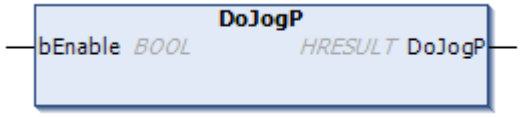
Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.

Return value	Cause
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bEnable	BOOL	A rising edge starts a jogging in negative direction. A falling edge triggers a stop of the axis.

5.10.2 DoJogP



This method performs the start and stop of the axis in the positive direction.

Syntax:

```
METHOD DoJogP : HRESULT
VAR_INPUT
    bEnable : BOOL;
END_VAR
```

Return value

Name	Type	Description
DoJogP	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bEnable	BOOL	A rising edge starts a jogging in positive direction. A falling edge triggers a stop of the axis.

5.10.3 SetParameter

This method is used to set the parameters of a movement in jogging mode.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
    fAcceleration : LREAL;
    fDeceleration : LREAL;
    fJerk         : LREAL;
    fVelocity     : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

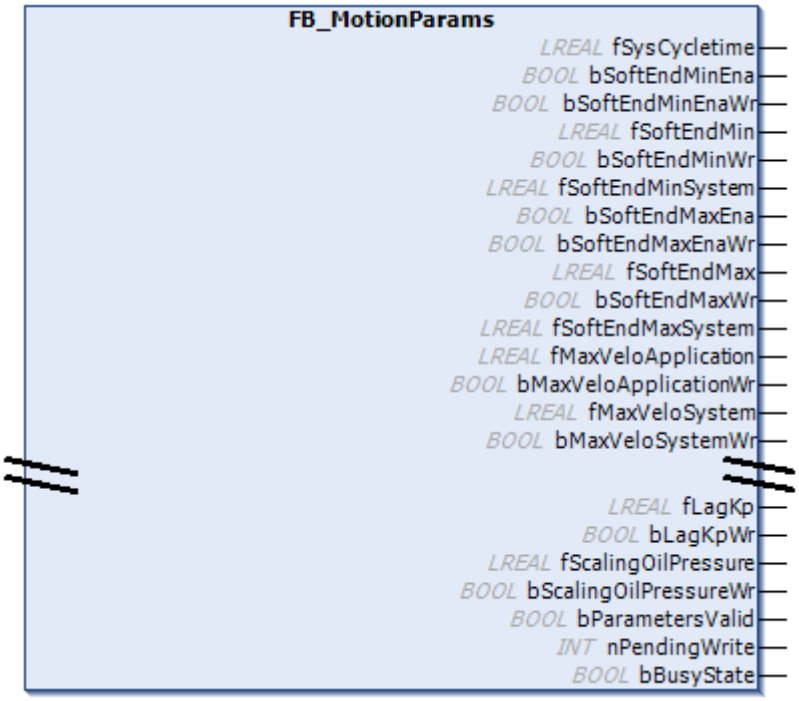
The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
fAcceleration	LREAL	The commanded acceleration.
fDeceleration	LREAL	The commanded deceleration.
fJerk	LREAL	The commanded jerk.
fVelocity	LREAL	The commanded velocity.

5.11 MotionParams



This core function is used to represent a range of parameter values of the axis.

Not all parameters are writeable.

Some parameters are read only (r/o) while others are readable and writeable (r/w).

Axis must be disabled

Some parameters cannot be written while the axis is enabled.

Signal pattern deviating from the standards

This core function implements a signal pattern that is different from the standards. See below for more information.

Syntax:

```
FUNCTION_BLOCK FB_MotionParams EXTENDS FB_Corefunction IMPLEMENTS I_MotionParams
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

Name	Type		Description
AsymTargeting	BOOL	Get, Set	TRUE if the parameters for the target approach are direction-dependent. Otherwise, the parameters for the negative direction are ignored and the parameters for the positive direction are used regardless of the direction. **Only active with hydraulic axes.**
AutoBrakeCalculation	BOOL	Get, Set	TRUE if the braking distance is determined automatically. **Only active with hydraulic axes.**

Name	Type		Description
AxisIsNc	BOOL	Get	TRUE, if the axis is operated with TwinCAT NC.
BrakeDistanceM	LREAL	Get, Set	The braking distance time in negative direction. **Only active with hydraulic axes.** See also AsymTargeting.
BrakeDistanceP	LREAL	Get, Set	The braking distance time in negative direction. **Only active with hydraulic axes.**
BusyState	BOOL	Get	TRUE, if the function block writes changed parameters.
CreepDistanceM	LREAL	Get, Set	The creep distance in negative direction. **Only active with hydraulic axes.** See also AsymTargeting.
CreepDistanceP	LREAL	Get, Set	The creep distance in positive direction. **Only active with hydraulic axes.**
CreepSpeedM	LREAL	Get, Set	The creep speed in negative direction. **Only active with hydraulic axes.**
CreepSpeedP	LREAL	Get, Set	The creep speed in negative direction. **Only active with hydraulic axes.**
CycleTime	LREAL	Get	The cycle time of the task in which the real-time functions of the axis are executed.
DriveIsCoE	BOOL	Get	TRUE, if the axis has a CoE interface.
DriveIsServo	BOOL	Get	TRUE if the axis is a servo drive.
DriveIsSimulated	BOOL	Get	TRUE if the drive of the axis is simulated.
DriveIsSoE	BOOL	Get	TRUE, if the axis has a SoE interface.
DriveReversed	BOOL	Get, Set	TRUE if the drive of the axis is inverted.
DriveType	UDINT	Get	A numeric identifier for the type of the connected drive. **The meaning of the constants is defined in the Tc2 NC or the Tc2_Hydraulics library.**
EncoderInterpolation	LREAL	Get, Set	The divider for the encoder evaluation. It specifies the number of increments corresponding to the distance specified by EncoderWeighting.
EncoderIsAnalog	BOOL	Get	TRUE if the encoder of the axis uses an analog input.
EncoderIsSimulated	BOOL	Get	TRUE if the encoder of the axis is simulated.
EncoderReversed	BOOL	Get, Set	TRUE if the encoder of the axis is inverted.
EncoderType	UDINT	Get	A numeric identifier for the type of the connected encoder. **The meaning of the constants is defined in the Tc2 NC or the Tc2_Hydraulics library.**
EncoderWeighting	LREAL	Get, Set	The factor for encoder evaluation. It specifies the distance corresponding to a number of increments specified by EncoderInterpolation.
EncoderZeroShift	LREAL	Get, Set	The zero offset shift of the encoder.
HasTorqueLimiting	BOOL	Get	TRUE if the axis has a torque limit.
LagControlled	BOOL	Get	TRUE if the axis has a position controller.
LagCtrlKp	LREAL	Get, Set	The kP factor of the position controller.
LagFilter	LREAL	Get, Set	The filter time of the lag error monitoring.
LagLimit	LREAL	Get, Set	The threshold value of the lag error monitoring.
LagMonitored	BOOL	Get, Set	TRUE, if the lag error monitoring of the axis is active.
MaxAcceleration	LREAL	Get, Set	The maximum permissible acceleration.
MaxDeceleration	LREAL	Get, Set	The maximum permissible deceleration.
MaxJerk	LREAL	Get, Set	The maximum permissible jerk.

Name	Type		Description
MaxVeloApplication	LREAL	Get, Set	The maximum permissible velocity of the axis.
MaxVeloSystem	LREAL	Get, Set	The maximum velocity of the axis.
MinVeloJog	LREAL	Get, Set	The minimum jogging velocity. **Only relevant for hydraulic axes. This parameter is 0.0 for NC axes.**
ScalingOilPressure	LREAL	Get, Set	The scaling factor for the actual pressure acquisition. **Only active with hydraulic axes.**
SoftEndMax	LREAL	Get, Set	The maximum software limit switch.
SoftEndMaxEna	BOOL	Get, Set	Enable for the maximum software limit switch.
SoftEndMaxSystem	LREAL	Get, Set	For the load side of transforming axes: The converted maximum software limit switch of the drive side. For the drive side of transforming axes: The converted maximum software limit switch of the load side. For non-transforming axes: A copy of the maximum software limit switch.
SoftEndMin	LREAL	Get, Set	The minimum software limit switch.
SoftEndMinEna	BOOL	Get, Set	Enable for the minimum software limit switch.
SoftEndMinSystem	LREAL	Get, Set	For the load side of transforming axes: The converted minimum software limit switch of the drive side. For the drive side of transforming axes: The converted minimum software limit switch of the load side. For non-transforming axes: A copy of the minimum software limit switch.
Valid	BOOL	Get	TRUE if all parameters are valid.

5.12 MotionSetpoints

This core function offers a range of current setpoint values.

Properties

Name	Type	Description
Acceleration	LREAL	The current acceleration setpoint.
Jerk	LREAL	The current jerk setpoint.
Position	LREAL	The current position setpoint.
TorqueLimiting	LREAL	The current setpoint for torque limitation.
Velocity	LREAL	The current velocity setpoint.

5.13 Power

This core function is used to enable the operation of the controlled device.

Syntax:

```
FUNCTION_BLOCK FB_Power EXTENDS FB_CorefunctionFeedback IMPLEMENTS I_Power
VAR_INPUT
END_VAR
```

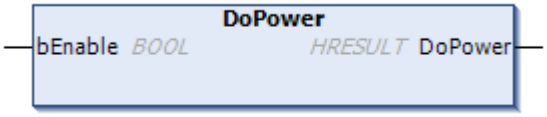
Properties

Name	Type	Access	Description
IsFeedEnabledM	BOOL	Get	TRUE if the axis is enabled for active movement in negative direction.
IsFeedEnabledP	BOOL	Get	TRUE if the axis is enabled for active movement in positive direction.
Override	LREAL	Get, Set	A factor for scaling commanded velocities. **The effect is significantly influenced by the type of axis and its parameterization.**
Status	BOOL	Get	TRUE, if the axis is enabled for active operation. **For an active movement, the direction-related enable is also required.**

Methods

Name	Description
DoPower [▶ 75]	Enable for active operation of the axis.
FeedEnable [▶ 76]	Directional enables for commanding active axis movements.

5.13.1 DoPower



This method is used to enable or disable the axis for active operation. If this requires an exchange of signals with a device, this exchange is performed and monitored.

Syntax:

```
METHOD DoPower: HRESULT
VAR_INPUT
    bEnable: BOOL;
END_VAR
```

Return value

Name	Type	Description
DoPower	HRESULT	See below

Inputs

Name	Type	Description
bEnable	BOOL	A rising edge starts the enable process. A falling edge starts the disable process.

5.13.2 FeedEnable

This method is used to define directional enables for active movements of the axis.

Syntax:

```
METHOD FeedEnable:   HRESULT
VAR_INPUT
    bFeedEnaPositive: BOOL;
    bFeedEnaNegative: BOOL;
END_VAR
```

Return value

Name	Type	Description
FeedEnable	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

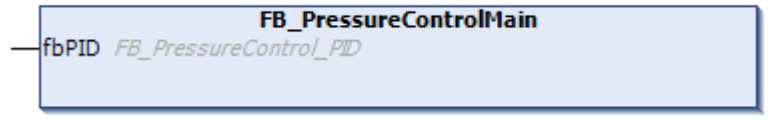
The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

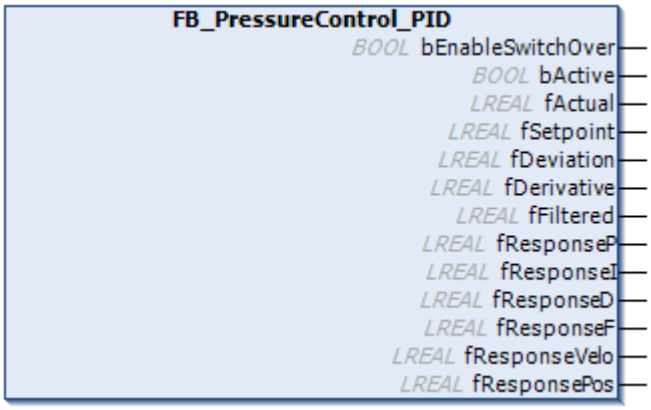
Name	Type	Description
bFeedEnaPositive	BOOL	A TRUE enables active movements in positive direction.
bFeedEnaNegative	BOOL	A TRUE enables active movements in negative direction.

5.14 PressureControl



This core function is provided to allow access to a number of controller types. Currently, there is an extended PID controller [▶ 77].

5.14.1 PressureControl.PID



This core function implements an extended PID controller functionality that can be used for a number of tasks:

- Holding pressure control in injection molding machines
- Back pressure control in injection molding machines
- Others

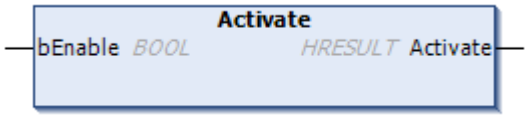
Properties

Name	Type	Description
IsEnabled	BOOL	A TRUE signals the active state of the controller.

Methods

Name	Description
Activate [▶ 77]	Activate / deactivate the controller
EnableSwitchOver [▶ 78]	Enable for automatic activation by a PressureHandler.
GetActual [▶ 78]	The actual value of the controller is determined.
GetParams [▶ 79]	An interface to the connected parameter set is determined.
SetParams [▶ 79]	A parameter set is connected to the controller.
Setpoint [▶ 80]	The setpoint of the controller is set.
SwitchOver [▶ 80]	Automatic activation by a PressureHandler.

5.14.1.1 Activate



This method is used to activate and deactivate the controller.

Syntax:

```
METHOD Activate: HRESULT
VAR_INPUT
    bEnable: BOOL;
END_VAR
```

Return value

Name	Type	Description
Activate	HRESULT	See below

Inputs

Name	Type	Description
bEnable	BOOL	Enable for the controller.

5.14.1.2 EnableSwitchOver

This method can be used to enable or disable automatic activation by a pressure handler.

Syntax:

```
METHOD EnableSwitchOver: HRESULT
VAR_INPUT
    bEnable: BOOL;
END_VAR
```

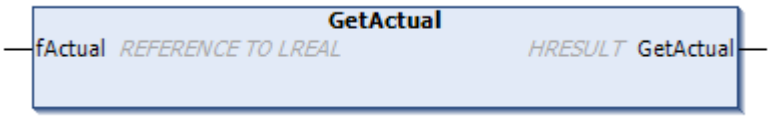
Return value

Name	Type	Description
EnableSwitchOver	HRESULT	See below

Inputs

Name	Type	Description
bEnable	BOOL	The enable for automatic activation.

5.14.1.3 GetActual



The actual value of the controller is determined.

Syntax:

```
METHOD GetActual : HRESULT
VAR_INPUT
fActual: REFERENCE TO LREAL;
END_VAR
```

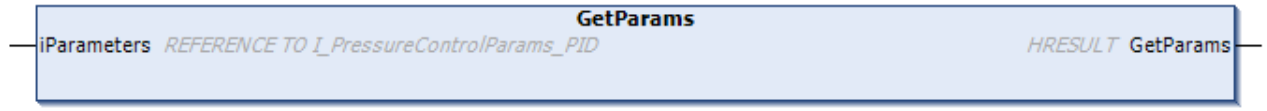
Return value

Name	Type	Description
GetActual	HRESULT	See below

Inputs

Name	Type	Description
fActual	REFERENCE TO LREAL	A reference to the variable to be updated with the actual value.

5.14.1.4 GetParams



An interface to the parameters of the controller is determined.

Syntax:

```
METHOD GetParams: HRESULT
VAR_INPUT
    iParameters: REFERENCE TO I_PressureControlParams_PID;
END_VAR
```

Return value

Name	Type	Description
GetParams	HRESULT	See below

Inputs

Name	Type	Description
iParameters	REFERENCE TO I_PressureControlParams_PID	A reference to the variable to be updated with the interface.

5.14.1.5 SetParams

A parameter set is connected to the controller.

Syntax:

```
METHOD SetParams: HRESULT
VAR_INPUT
    iParams: I_PressureControlParams_PID [▶ 81];
END_VAR
```

Return value

Name	Type	Description
SetParams	HRESULT	See below

Inputs

Name	Type	Description
iParameters	I_PressureControlParams_PID	An interface to the parameter set.

5.14.1.6 Setpoint

The setpoint of the controller is defined.

Syntax:

```
METHOD Setpoint: HRESULT
VAR_INPUT
    fValue: LREAL;
END_VAR
```

Return value

Name	Type	Description
Setpoint	HRESULT	See below

Inputs

Name	Type	Description
fValue	LREAL	The setpoint.

5.14.1.7 SwitchOver

This method can be used by a pressure handler function block.

A TRUE at bSwitchover will activate the controller if EnableSwitchOver(TRUE) has been called before.

Syntax:

```
METHOD SwitchOver: HRESULT
VAR_INPUT
    bSwitchOver: BOOL;
END_VAR
```

Return value

Name	Type	Description
SwitchOver	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

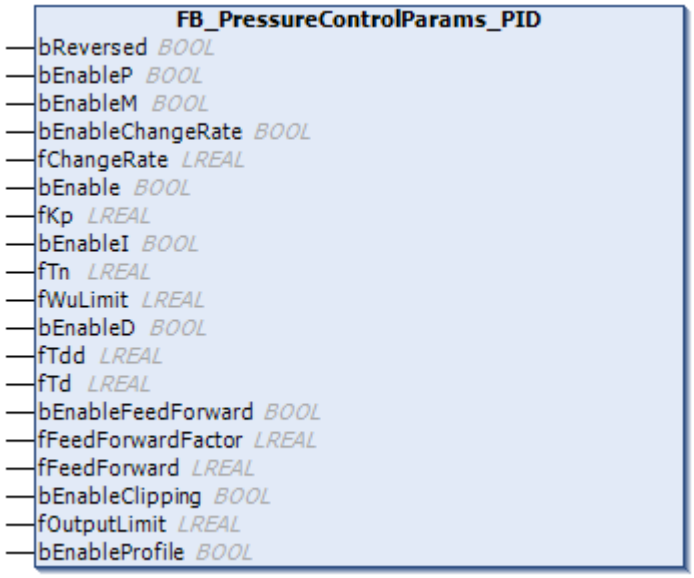
Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bSwitchOver	BOOL	The activation signal of the pressure handler.

5.14.2 FB_PressureControlParams_PID



This function block contains a parameter set to be used by a FB_PressureControl_PID [▶ 77] function block.

Syntax:

```
FUNCTION_BLOCK FB_PressureControlParams_PID IMPLEMENTS I_PressureControlParams_PID
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

Name	Type	Access	Properties
ChangeRate	LREAL	Get, Set	The ramp rate to use. **See EnableChangeRate for details.**
Enable	BOOL	Get, Set	This property is used to activate and deactivate the proportional component of the PID controller. **For details see Kp.**
EnableChangeRate	BOOL	Get, Set	A setpoint ramp function can be activated and deactivated.

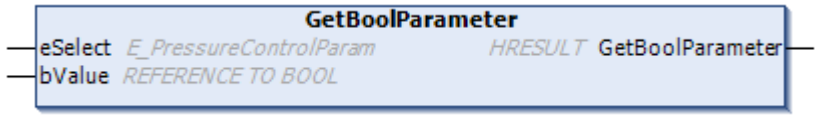
Name	Type	Access	Properties
			The internally used setpoint for the controller may be updated to the given setpoint by a limited rate, given as ChangeRate. This parameter is given in pressure units per second. **A setting of ChangeRate:=0.0 or EnableChangeRate:=FALSE disables the ramp function and causes the internal setpoint to immediately follow the specified setpoint.**
EnableClipping	BOOL	Get, Set	A limiting function for the output can be activated and deactivated. **For details see OutputLimit.**
EnableD	BOOL	Get, Set	The differential component of the PID controller can be activated and deactivated. **For details see Td.**
EnableFeedForward	BOOL	Get, Set	A velocity pre-control can be activated and deactivated. **For details see FeedForward.**
EnableI	BOOL	Get, Set	This property is used to enable and disable the integrating component of the PID controller. For details see Tn below.
EnableM	BOOL	Get, Set	This property is used to enable and disable a negative output of the controller.
EnableP	BOOL	Get, Set	This property is used to enable or disable a positive output of the controller.
FeedForward	LREAL	Get, Set	A velocity feed forward component. **When EnableFeedForward is set to TRUE, the value of FeedForward is multiplied by FeedForwardFactor and added to the PID response output.** **An active back pressure controller is used to match the injector backward velocity to the effect of the dosing axis. This function can be used to achieve a more dynamic adjustment to changes in turn rate.**
FeedForwardFactor	LREAL	Get, Set	This property is a parameter of the velocity feed forward function. **For details see FeedForward.**
Kp	LREAL	Get, Set	The proportional gain of the PID controller. Enable must be TRUE to allow the calculation. **The unit is velocity unit per pressure unit.**
OutputLimit	LREAL	Get, Set	A limit for the response of the controller.
Reversed	BOOL	Get, Set	This property is used to invert the output of the PID controller. **In a number of applications, the axis must move in the positive direction to relieve excess pressure. Typical examples are holding pressure and back pressure controllers in injection molding machines.**
Td	LREAL	Get, Set	The differential component of the PID controller. **The response is calculated if EnableD is TRUE and Td and Tdd are >=cycle time, otherwise it is zero.** **The unit is velocity units * second per pressure unit.**
Tdd	LREAL	Get, Set	A parameter of the differential component of the PID controller. For details see Td above.
Tn	LREAL	Get, Set	The integrating component of the PID controller.

Name	Type	Access	Properties
			The response is calculated if EnableI is TRUE and Tn >= cycle time, otherwise it is zero. The output is limited to WuLimit. **The unit is velocity units per (pressure unit * second).**
WuLimit	LREAL	Get, Set	A parameter of the integrating component of the PID controller. **For details see Tn above.**

Methods

Name	Description
GetBoolParameter [▶ 83]	This method is used to read BOOL parameters of the controller. For details see E_PressureControlParam [▶ 85].
GetFloatParameter [▶ 84]	This method is used to read LREAL parameters of the controller. For details see E_PressureControlParam [▶ 85].
SetBoolParameter [▶ 84]	This method is used to define BOOL parameters of the controller. For details see E_PressureControlParam [▶ 85].
SetFloatParameter [▶ 85]	This method is used to define LREAL parameters of the controller. For details see E_PressureControlParam [▶ 85].

5.14.2.1 GetBoolParameter



This method is used to read BOOL parameters of the controller. For details see E_PressureControlParam [▶ 85].

Syntax:

```
METHOD GetBoolParameter : HRESULT
VAR_INPUT
    eSelect: E_PressureControlParam;
    bValue : REFERENCE TO BOOL;
END_VAR
```

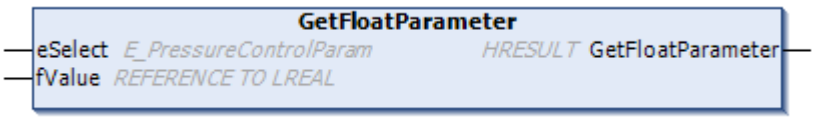
Return value

Name	Type	Description
GetBoolParameter	HRESULT	See below

Inputs

Name	Type	Description
eSelect	E_PressureControlParam	The selection of the parameter.
bValue	REFERENCE TO BOOL	A reference to the variable to be updated with the parameter.

5.14.2.2 GetFloatParameter



This method is used to read LREAL parameters of the controller. For details see E_PressureControlParam [▶ 85].

Syntax:

```
METHOD GetFloatParameter : HRESULT
VAR_INPUT
    eSelect: E_PressureControlParam;
    fValue : REFERENCE TO LREAL;
END_VAR
```

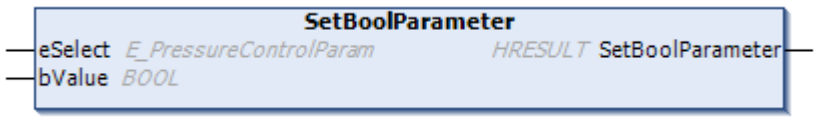
Return value

Name	Type	Description
GetFloatParameter	HRESULT	See below

Inputs

Name	Type	Description
eSelect	E_PressureControlParam	The selection of the parameter.
fValue	REFERENCE TO LREAL	A reference to the variable to be updated with the parameter.

5.14.2.3 SetBoolParameter



This method is used to define BOOL parameters of the controller. For details see E_PressureControlParam [▶ 85].

Syntax:

```
METHOD SetBoolParameter : HRESULT
VAR_INPUT
    eSelect: E_PressureControlParam;
    bValue : BOOL;
END_VAR
```

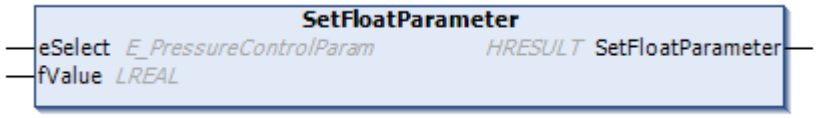
Return value

Name	Type	Description
SetBoolParameter	HRESULT	See below

Inputs

Name	Type	Description
eSelect	E_PressureControlParam	The selection of the parameter.
bValue	BOOL	The value with which the parameter is to be defined.

5.14.2.4 SetFloatParameter



This method is used to define LREAL parameters of the controller. For details see E_PressureControlParam [▶ 85].

Syntax:

```
METHOD SetFloatParameter : HRESULT
VAR_INPUT
    eSelect: E_PressureControlParam;
    fValue : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetFloatParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
eSelect	E_PressureControlParam	The selection of the parameter.
fValue	LREAL	The value with which the parameter is to be defined.

5.14.3 E_PressureControlParam

The values of this enumeration are used by GetBoolParameter(), GetFloatParameter(), SetBoolParameter() and SetFloatParameter() of FB_PressureControlParams_PID [▶ 77].

```
TYPE E_PressureControlParam :
(
eKp := 1,
eTn,
eTd,
eTdd,
```

```
 //
eWuLimit,
eOutLimit,
//
eChangeRate,
eFeedForward,
eFeedForwardFactor,

eEnable,
eEnableP,
eEnableM,
//
eReversed,
//
eEnableFF,
eEnableI,
eEnableD,
eEnableClipping,
eEnableChangeRate
);
END_TYPE
```

5.15 Ptp

This core function is used to perform multi-segment movements with the possibility to switch to torque or constant output clamping at the end.

Connection during startup

The internal PtpLookUp core function of the axis is connected to the Ptp core function at startup.

Not supported by all axis types

This core function is not supported by inverter axes. Any use will report DEVICE_NOTINIT and trigger an error message.

Properties

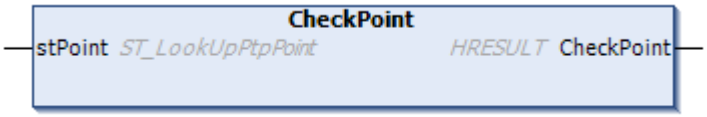
Name	Type	Description
ActiveSegment	INT	The number of the segment that is currently being executed.
IsClamping	BOOL	TRUE, if the axis has switched to Clamping.
MovingNegative	BOOL	TRUE if the axis is actively moving in the negative direction.
MovingPositive	BOOL	TRUE if the axis is actively moving in the positive direction.
NumberOfPoints	INT	The number of motion segments that the core function can store.

Methods

Name	Description
CheckPoint [▶ 87]	The specified segment is compared with the axis parameters.
DoMove [▶ 87]	The execution is triggered.
GetClampPoint [▶ 88]	A segment of the clamping table is read back.
GetPoint [▶ 88]	A segment of the lookup table is read back.
GetUpdatedPoint [▶ 89]	A segment of the lookup table is read back.
InvalidateClampPoint [▶ 89]	All segments in the clamping table are marked as invalid.
InvalidateTable [▶ 90]	All segments in the lookup table are marked as invalid.

Name	Description
SetClampPoint [▶ 90]	A segment of the clamping table is defined.
SetPoint [▶ 90]	A segment of the lookup table is defined.
UpdatePosition [▶ 91]	The target position of the segment is changed after it is defined.

5.15.1 CheckPoint



The specified segment is compared with the axis parameters. The result is only tested as SUCCEEDED() if the target does not exceed any of the enabled software position limits and does not fall below the minimum velocity.

Syntax:

```
METHOD CheckPoint: HRESULT
VAR_INPUT
    stPoint: ST_LookUpPtpPoint;
END_VAR
```

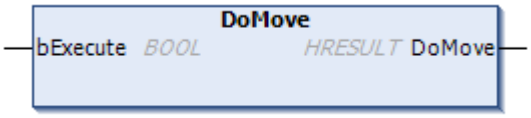
Return value

Name	Type	Description
CheckPoint	HRESULT	See below

Inputs

Name	Type	Description
stPoint	LookUpPtpPoint	The segment to be checked.

5.15.2 DoMove



The execution is triggered.

Syntax:

```
METHOD DoMove: HRESULT
VAR_INPUT
    bExecute: BOOL;
END_VAR
```

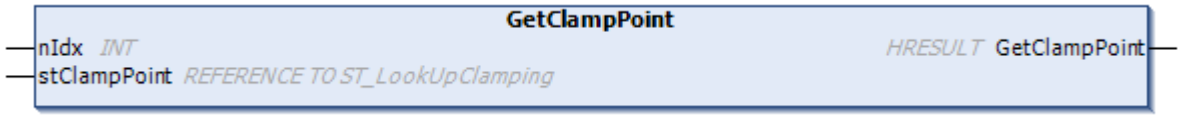
Return value

Name	Type	Description
DoMove	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge triggers the start.

5.15.3 GetClampPoint



A segment of the clamping table is read back.

Syntax:

```
METHOD GetClampPoint: HRESULT
VAR_INPUT
    nIdx: INT;
END_VAR
```

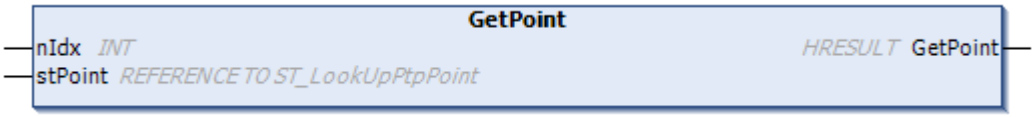
Return value

Name	Type	Description
GetClampPoint	HRESULT	See below

Inputs

Name	Type	Description
nIdx	INT	The index of the segment.

5.15.4 GetPoint



A segment of the lookup table is read back.

Syntax:

```
METHOD GetPoint: HRESULT
VAR_INPUT
    nIdx   : INT;
    stPoint: REFERENCE TO ST_LookUpPtpPoint;
END_VAR
```

Return value

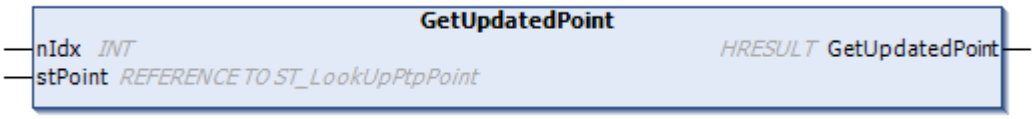
Name	Type	Description
GetPoint	HRESULT	See below

Inputs

Name	Type	Meeting
nIdx	INT	The index of the segment.

Name	Type	Meeting
stPoint	REFERENCE TO ST_LookUpPtpPoint	A reference to the variable that is to be updated.

5.15.5 GetUpdatedPoint



A segment of the lookup table is read back.

Syntax:

```
METHOD GetUpdatedPoint: HRESULT
VAR_INPUT
    nIdx   : INT;
    stPoint: REFERENCE TO ST_LookUpPtpPoint;
END_VAR
```

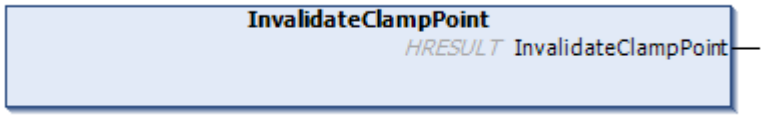
Return value

Name	Type	Description
GetUpdatedPoint	HRESULT	See below

Inputs

Name	Type	Meeting
nIdx	INT	The index of the segment.
stPoint	REFERENCE TO ST_LookUpPtpPoint	A reference to the variable that is to be updated.

5.15.6 InvalidateClampPoint



All segments in the clamping table are marked as invalid.

Syntax:

```
METHOD InvalidateClampPoint: HRESULT
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
InvalidateClampPoint	HRESULT	See below

5.15.7 InvalidateTable

All segments in the lookup table are marked as invalid.

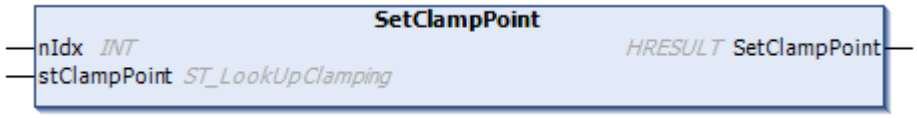
Syntax:

```
METHOD InvalidateTable: HRESULT
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
InvalidateTable	HRESULT	See below

5.15.8 SetClampPoint



A segment of the clamping table is defined.

Syntax:

```
METHOD SetClampPoint: HRESULT
VAR_INPUT
    Idx: INT;
END_VAR
```

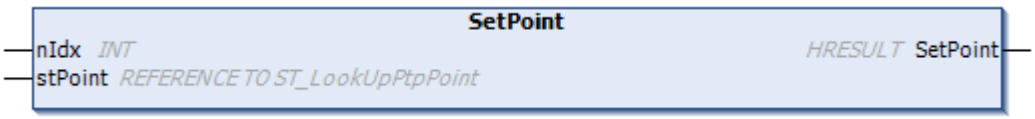
Return value

Name	Type	Description
SetClampPoint	HRESULT	See below

Inputs

Name	Type	Description
nIdx	INT	The index of the clamping segment.
stClampPoint	ST_LookUpClamping	The clamping segment to be used.

5.15.9 SetPoint



A segment of the lookup table is defined.

Syntax:

```
METHOD SetPoint: HRESULT
VAR_INPUT
    nIdx   : INT;
    stPoint: REFERENCE TO ST_LookUpPtpPoint;
END_VAR
```

Return value

Name	Type	Description
SetPoint	BOOL	See below

Inputs

Name	Type	Description
nIdx	INT	The index of the segment.
stPoint	REFERENCE TO ST_LookUpPtpPoint	A reference to the variable that is to be updated.

5.15.10 UpdatePosition

The target position of the segment is changed after it is defined.

Syntax:

```
METHOD UpdatePosition:  HRESULT
VAR_INPUT
    nIdx     : INT;
    fPosition: LREAL;
    bSwap    : BOOL;
    bForce   : BOOL;
END_VAR
```

Return value

Name	Type	Description
UpdatePosition	BOOL	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

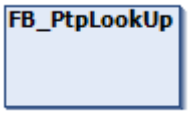
Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
nIdx	INT	The index of the segment to be updated.
fPosition	LREAL	The new target position.
bSwap	BOOL	With bSwap=TRUE the direction related enables (PositiveOnly, NegativeOnly) are swapped.
bForce	BOOL	If TRUE, the segment will be updated even if the axis executes a command. **Make sure that the axis does not execute a Ptp command.**

5.16 PtpLookUp



This core function is used to store the definition of a complex multi-segment movement.

Connection during startup

The internal PtpLookUp core function of the axis is connected to the Ptp core function at startup.

Not available via the axis interfaces

This core function is not available via the axis interfaces. Direct access by the application is not required. Nevertheless, this is documented here for a better understanding of the Ptp function.

Properties

Name	Type	Description
NumberOfPoints	INT	The number of motion segments that the core function can store.

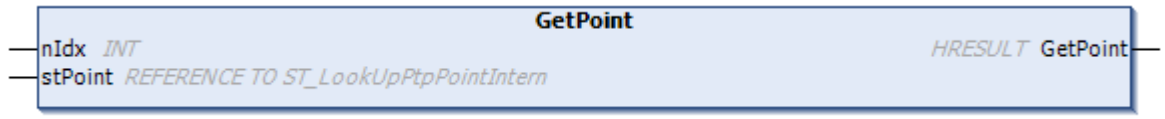
Definition at the instantiation

The capacity of the core function is determined at instantiation.

Methods

Name	Description
GetPoint [▶ 93]	A segment from the lookup table is read back.
Invalidate [▶ 93]	All segments in the lookup table are marked as invalid.
ReadMaster [▶ 93]	A master value is determined that corresponds to the specified slave value.
SetPoint [▶ 94]	A segment of the lookup table is defined.
UpdatePosition [▶ 95]	In some applications, the target position of the segments must be changed after definition. **This method is used, for example, with transforming axes.**

5.16.1 GetPoint



A segment from the lookup table is read back.

Syntax:

```
METHOD GetPoint: HRESULT
VAR_INPUT
    nIdx    : INT;
    stPoint: REFERENCE TO ST_LookUpPtpPointIntern;
END_VAR
```

Return value

Name	Type	Description
GetPoint	HRESULT	See below

Inputs

Name	Type	Description
nIdx	INT	The index of the segment. Permissible range 1 ... NumberOfPoints.
stPoint	REFERENCE TO ST_LookUpPtpPointIntern [▶ 96]	A reference to the variable to be updated with the segment data.

5.16.2 Invalidate

This method invalidates all segments in the lookup table.

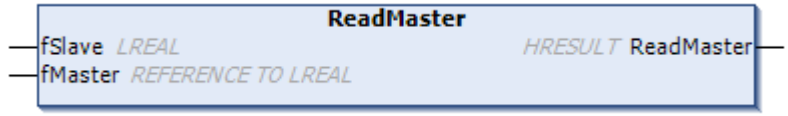
Syntax:

```
METHOD Invalidate: HRESULT
VAR_INPUT
END_VAR
```

Return value

Name	Type	Description
Invalidate	HRESULT	See below

5.16.3 ReadMaster



This method finds a master value that corresponds to the specified slave value.

Syntax:

```
METHOD ReadMaster : HRESULT
VAR_INPUT
fSlave: LREAL;
fMaster: REFERENCE TO LREAL;
END_VAR
```

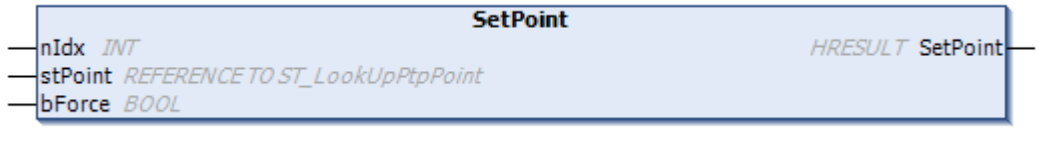
Return value

Name	Type	Description
ReadMaster	HRESULT	See below

Inputs

Name	Type	Description
fSlave	LREAL	The specified slave value.
fMaster	REFERENCE TO LREAL	A reference to the variable to be updated with the master position.

5.16.4 SetPoint



A segment of the lookup table is defined.

Syntax:

```
METHOD SetPoint: HRESULT
VAR_INPUT
    nIdx   : INT;
    stPoint: REFERENCE TO ST_LookUpPtpPoint;
    bForce : BOOL;
END_VAR
```

Return value

Name	Type	Description
SetPoint	HRESULT	See below

Inputs

Name	Type	Description
nIdx	INT	The index of the segment. Permissible range 1 ... NumberOfPoints.
stPoint	REFERENCE TO ST_LookUoPtpPoint [▶ 96]	A reference to the variable with which the segment is to be updated.
bForce	BOOL	If TRUE, the segment will be updated even if the axis executes a command. **Make sure that the axis does not execute a Ptp command.**

5.16.5 UpdatePosition

In some use cases, the target position of the segments must be modified after definition. This method is used for transforming axes, for example.

Syntax:

```
METHOD UpdatePosition: HRESULT
VAR_INPUT
    nIdx     : INT;
    fPosition: LREAL;
    bSwap    : BOOL;
    bForce   : BOOL;
END_VAR
```

Return value

Name	Type	Description
UpdatePosition	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
nIdx	INT	The index of the segment. Permissible range 1 ... NumberOfPoints.
fPosition	LREAL	The value to be used for the update.
bSwap	BOOL	With bSwap=TRUE the direction related enables (PositiveOnly, NegativeOnly) are swapped.
bForce	BOOL	If TRUE, the segment will be updated even if the axis executes a command. **Make sure that the axis does not execute a Ptp command.**

5.16.6 ST_LookUpPtpPoint

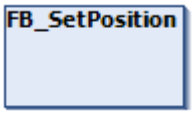
This data type is used to hold segments of a complex movement.

Syntax:

```
TYPE ST_LookUpPtpPoint :
STRUCT
    Position    : LREAL;
    Velocity    : LREAL;
    Acceleration: LREAL;
    Deceleration: LREAL;
    Jerk        : LREAL;
    Limiting    : LREAL;
    Valid       : BOOL;
    PositiveOnly: BOOL;
    NegativeOnly: BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
Position	LREAL	The target of the segment. If there is another segment in the motion profile, it will be blended with the BlendingLow rule.
Velocity	LREAL	The velocity in the segment. **Due to circumstances, the specified velocity may be unattainable.**
Acceleration	LREAL	The acceleration in the segment.
Deceleration	LREAL	The deceleration in the segment.
Jerk	LREAL	The jerk in the segment.
Limiting	LREAL	The torque limiting in the segment.
Valid	BOOL	Must be set to TRUE to validate the segment.
PositiveOnly	BOOL	Will prevent movement in the negative direction due to the segment.
NegativeOnly	BOOL	Will prevent movement in the positive direction due to the segment.

5.17 SetPosition



This core function is used to change the actual position without physically moving the axis. It updates the offset of the position encoder function.

NOTICE

Not supported by all axis types

This core function is not supported by inverter axes. Any use will report DEVICE_NOTINIT and trigger an error message.

Properties

Name	Type	Description
Mode	BOOL	Mode := TRUE causes the actual position to be changed by an amount specified as Target. Mode := FALSE causes the actual position to be set to the value specified as the target. **The Mode property has the same effect as the bRelative input of the SetParameter() method.**

Name	Type	Description
Target	LREAL	This sets the new position value. **The Target property has the same effect as the fPosition input of the SetParameter() method.**

Methods

Name	Description
DoSetPosition [▶ 97]	A rising edge at the bExecute input triggers the core function.
SetParameter [▶ 97]	Here the new position and the operation mode of the core function are defined.

5.17.1 DoSetPosition

This method triggers the core function.

Syntax:

```
METHOD DoSetPosition : HRESULT
VAR_INPUT
bExecute: BOOL;
END_VAR
```

Return value

Name	Type	Description
DoSetPosition	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge starts the core function.

5.17.2 SetParameter

Here the new position and the operation mode of the core function are defined.

Syntax:

```
METHOD SetParameter : HRESULT
VAR_INPUT
fPosition: LREAL;
bRelative: BOOL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
fPosition	LREAL	The position value to be used is defined here.
bRelative	BOOL	bRelative := TRUE causes the actual position to be changed by an amount specified as fPosition. bRelative:= FALSE causes the actual position to be set to the value specified as fPosition.

5.18 Stop

This core function is used to perform a stop operation using certain dynamic parameters.

Properties

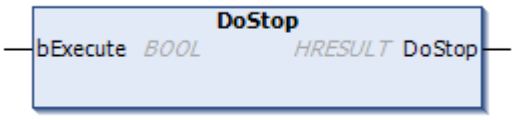
Name	Type	Description
NoCreeping	BOOL	A TRUE in this property avoids the creep phase at the end of the stopping process of the hydraulics library.

Methods

Name	Description
DoStop [▶ 99]	A rising edge triggers the stop.

Name	Description
SetParameter [▶ 99]	This method is used to define the dynamic parameters of the operation.

5.18.1 DoStop



This method is used to trigger the stop.

Syntax:

```
METHOD DoStop : HRESULT
VAR_INPUT
    bExecute: BOOL;
END_VAR
```

Return value

Name	Type	Description
DoStop	HRESULT	See below

Inputs

Name	Type	Description
bExecute	BOOL	A rising edge at this input triggers the stop.

5.18.2 SetParameter

This method is used to define the dynamic parameters of the operation.

Syntax:

```
METHOD SetParameter: HRESULT
VAR_INPUT
    fDeceleration: LREAL;
    fJerk        : LREAL;
END_VAR
```

Return value

Name	Type	Description
SetParameter	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

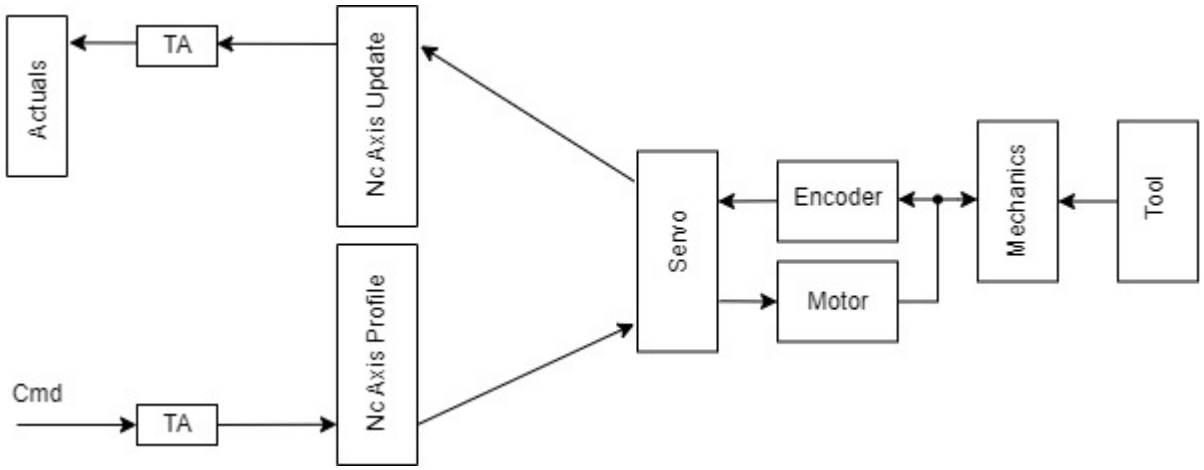
Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
fDeceleration	LREAL	The deceleration to be used.
fJerk	LREAL	The jerk to be used.

5.19 ToolAdaptation

This core function is used to convert between axis coordinates and tool working position. An offset and an optional inversion are taken into account.



Properties

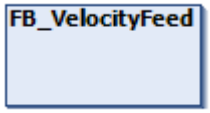
Name	Type	Description
Inverting	BOOL	A TRUE here signals a reversal of direction between axis and tool movement.
Offset	LREAL	The difference between axis and tool position.

Methods

Name	Description
AxisPosition	The axis position is determined for a given tool position.
AxisVelocity	The axis velocity is determined for a given tool velocity.

Name	Description
ToolPosition	The tool position is determined for a given axis position.
ToolVelocity	The tool velocity is determined for a given axis velocity.

5.20 VelocityFeed



This core function is used to start the axis with a commanded velocity but without a defined target position.

Properties

Name	Type	Description
GearFactor	LREAL	The factor with which to respond to GuidingValue.
GuidingValue	LREAL	The commanded velocity.

Decouple the axis

A GearFactor of 0.0 will effectively decouple the axis from the GuidingValue.

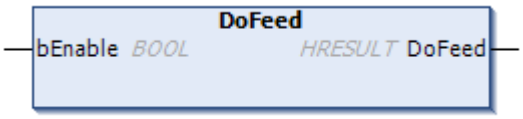
Move in the opposite direction

A negative GearFactor causes the axis to move in the opposite direction.

Methods

Name	Description
DoFeed	A rising edge triggers the core function.

5.20.1 DoFeed



A rising edge at bEnable triggers the core function, while a falling edge causes it to stop and return to idle.

Syntax:

```
METHOD DoFeed : HRESULT
VAR_INPUT
bEnable: BOOL;
END_VAR
```

Return value

Name	Type	Description
DoFeed	HRESULT	See below

Return values of core function methods

The return values of the methods are of type HRESULT. This 32-bit data type encodes various other information in the upper 16 bits besides success / failure. In the lower 16 bits an indication of a cause is transported.

To evaluate a HRESULT the FUNCTIONs SUCCEEDED(hr) and FAILED(hr) can be used.

The following values are to be expected here:

Return value	Cause
F_HresultFailure(E_AdsErr.DEVICE_BUSY)	The axis is busy performing another core function.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDDATA)	Not all data and parameters for the core function are available and valid.
F_HresultFailure(E_AdsErr.DEVICE_INVALIDSTATE)	The state of the axis does not allow the execution.
F_HresultFailure(E_AdsErr.DEVICE_NOTINIT)	The core function is not or not completely initialized.
F_HresultSuccess(E_AdsErr.NOERR)	The core function has accepted the command.

Inputs

Name	Type	Description
bEnable	BOOL	A rising edge triggers the core function, while a falling edge causes it to stop and return to idle.

6 Axis properties and methods

6.1 Axis properties

A range of data elements (i.e. parameters, states etc.) of the axis are implemented as properties. Some of these elements are readable and writable (i.e. Get, Set), while others are read-only (Get).

Interfaces to core functions

There is a range of properties that return interfaces to core functions. Details about these functions are given in a separate part of this documentation.

Name	Type	Access	Description
AutoTorqueLimitSelect	BOOL	Get, Set	An NC-based axis provides integrated function blocks for torque-limiting communication with SoE (i.e. Sercos over EtherCAT) and CoE (i.e. CAN over EtherCAT). If AutoTorqueLimitSelect is TRUE (default setting), the axis uses the appropriate internal function block as soon as it can detect the type of the connected servo drive. **This property must be set to FALSE if there is an application function block used for torque-limiting communication.**
AxisIsInverter	BOOL	Get	TRUE, if the drive of the axis is a frequency converter.
AxisIsNc	BOOL	Get	TRUE, if the axis is based on TwinCAT NC.
AxisName	STRING	Get, Set	The textual designation of the axis. In most cases the name is given in the instantiation of the axis as in fbSomeAxis: FB_AxisNcBase(AxisName:='NcClampAxis1', … The name can be defined at the latest as the first action in the first cyclic call of the axis. The axis name must be unique. This is of particular importance for hydraulic axes. The library used here uses the axis name as file name for loading and saving the axis parameters.
CycleTime	LREAL	Get	In the startup phase, the axis will determine the cycle time of the PLC task that runs the axis function. The axis will not execute any function if it could not update this value.
CycleTimeValid	BOOL	Get	TRUE, if CycleTimeValid was determined.
State	E_AxisState	Get, Set	The state [▶ 19] of the axis. **This property is writeable for technical reasons. The application must not use this to modify the situation or behavior of the axis.**
SwitchOver	BOOL	Get	TRUE if a pressure handling has triggered a pressure controller activation.
UseDatFile	BOOL		Axis parameters are loaded from a DAT file if TRUE. **Only effective with hydraulic axes.** **A TRUE must be set before the first Cyclic call.**

6.2 Axis methods

METHOD Cyclic

This method implements all axis mechanisms that must be called cyclically.

No implementation of real-time requirements

The library does not implement real-time requirements. The application is responsible for calling this method for all axes.

```
METHOD DoReset : HRESULT
VAR_INPUT
    bExecute : BOOL;
END_VAR
```

This method calls the DoReset() method of all core functions and all torque-limiting functions associated with or contained within the axis. The error state of the axis is deleted.

METHOD FB_init : BOOL (NC based axis)

```
VAR_INPUT
    bInitRetains    : BOOL;
    bInCopyCode     : BOOL;
    AxisName        : STRING;
    nPtpPoints      : INT;
    iProcessHandler : I_ProcessHandler;
    iPosCamLookup   : I_CammingLookUp;
    iVeloCamLookup  : I_CammingLookUp;
END_VAR
```

This method is used by TwinCAT PLC to initialize, move or enable NC-based axes. The specified parameters are used to initialize parameters, assign buffers and connect optional components.

This method is different for hydraulics library axes. Details can be found below.

NOTICE

This method must not be called by the application.

```
bInitRetains : BOOL;
```

```
bInCopyCode : BOOL;
```

These inputs are used to coordinate the commissioning of objects (i.e. FBs), the start, the stop or the Online Change of PLC projects. They are defined by the TwinCAT PLC runtime. Details can be found in the TwinCAT documentation.

```
AxisName : STRING;
```

This string is used to label the axis.

Hydraulic axes use this string to create the path to the parameter file.

```
nPtpPoints : INT;
```

This parameter is used to allocate memory for a buffer for multi-segmented motion definitions. The allocation is done at PLC start. The memory will be released automatically at PLC stop.

This parameter can be 0 for axes that do not use a Ptp command at any time.

```
iProcessHandler : I_ProcessHandler;
```

Reserved for future extension.

```
iPosCamLookup : I_CammingLookUp;
```

This parameter is an INTERFACE to a Camming Lookup FB. This can be set during commissioning or later.

```
iVeloCamLookup : I_CammingLookUp;
```

This parameter is an INTERFACE to a Camming Lookup FB. This can be set during commissioning or later.

METHOD FB_init : BOOL (Hydraulics library based axis)

```
VAR_INPUT
    bInitRetains    : BOOL;
    bInCopyCode     : BOOL;
    AxisName        : STRING;
    nPtpPoints      : INT;
    iProcessHandler : I_ProcessHandler;
    iPosCamLookup   : I_CammingLookUp;
    iVeloCamLookup  : I_CammingLookUp;
    iEncoder        : I_InputBase;
    iDrive          : I_OutputBase;
    iPressureP      : I_InputBase;
    iPressureM      : I_InputBase;
    iPosFilter      : I_Filter;
    iVeloFilter     : I_Filter;
END_VAR
```

This method is used by TwinCAT PLC to initialize, move or release axes based on the hydraulics library. The specified parameters are used to initialize parameters, to assign buffers and to connect external or optional components.

Different for NC axes

This method is different for NC axes. Details can be found above.

NOTICE

Must not be called by the application

This method must not be called by the application.

```
bInitRetains    : BOOL;
bInCopyCode     : BOOL;
AxisName        : STRING;
nPtpPoints      : INT;
iProcessHandler : I_ProcessHandler;
iPosCamLookup   : I_CammingLookUp;
iVeloCamLookup  : I_CammingLookUp;
```

Same as above.

```
iEncoder : I_InputBase;
```

In the case of axes based on a hydraulics library, a variety of devices can be used as a source of position information. This is done by assigning an INTERFACE to a standardized FB corresponding to the input hardware used.

```
iDrive : I_OutputBase;
```

In the case of axes based on a hydraulics library, a variety of devices can be used as targets for the control output. This is done by assigning an INTERFACE to a standardized FB that corresponds to the output hardware used.

```
iPressureP : I_InputBase;
```

```
iPressureM : I_InputBase;
```

In the case of axes based on a hydraulics library, a wide range of devices can be used as a source of actual pressure information. This is done by assigning an INTERFACE to a standardized FB corresponding to the input hardware used.

```
iPosFilter : I_Filter;
```

In some cases, the actual position information may have a high noise level. This parameter allows a standardized filter FB to be applied.

Strong filtering

Strong filtering may have a negative effect on the performance of the axis.

```
iVeloFilter : I_Filter;
```

In some cases, the actual velocity information may have a high noise level. This parameter allows a standardized filter FB to be applied.

Strong filtering

Strong filtering may have a negative effect on the performance of the axis.

METHOD ABSTRACT SetProcessHandler : HRESULT

```
VAR_INPUT
    iProcessHandler : I_ProcessHandler;
END_VAR
```

Reserved for future extension.

6.2.1 FB_init

The FB_init method is automatically called by the TwinCAT runtime environment during a system start or stop of the PLC. Must not be called by the application.

FB_Init for FBs of the types FB_AxisNcBase and FB_AxisInvBase

FBs of the types FB_AxisNcBase and FB_AxisInvBase do not overwrite the FB_init method, but use a constructor that is specified by FB_AxisBase. This constructor defines the syntax used to create an axis of these types.

The inputs bInitRetains and bInCopyCode are specified by TwinCAT PLC and are present in the FB_init() method of each function block. They are used by the TwinCAT runtime environment to inform the method about the reason for the call.

Syntax:

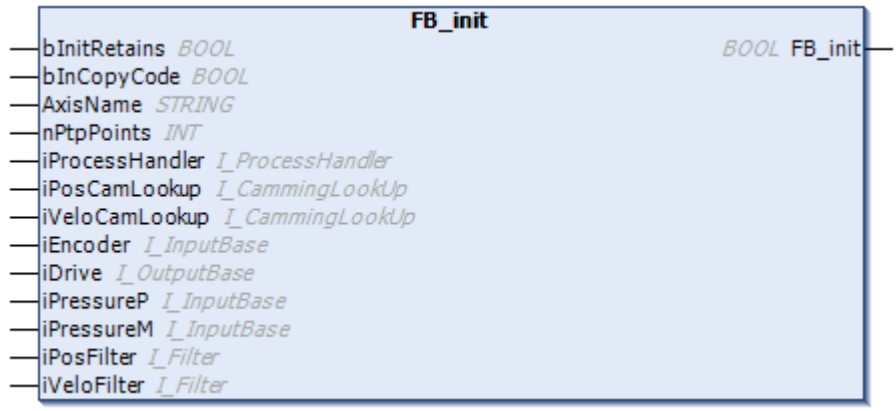
Defined in FB_AxisBase:

```
METHOD FB_init : BOOL
VAR_INPUT
bInitRetains : BOOL;
bInCopyCode : BOOL;
AxisName : STRING;
nPtpPoints : INT;
iProcessHandler : I_ProcessHandler;
iPosCamLookup : I_CammingLookUp;
iVeloCamLookup : I_CammingLookUp;
END_VAR
```

Inputs

Name	Type	Description
bInitRetains	BOOL	Reserved for TwinCAT PLC, see above
bInCopyCode	BOOL	Reserved for TwinCAT PLC, see above
AxisName	STRING	The text-based name of the axis.
nPtpPoints	INT	The number of supported segments in Ptp commands. **This input can be initially set to zero. In this case, the number must be specified at a later time.**
iProcessHandler	I_ProcessHandler	An interface to a FB for automatic pressure controller activation. **Reserved for extension: This input is currently always to be assigned with null.**
iPosCamLookUp	I_CammingLookUp	An interface to a FB with a position cam plate. **This input is usually assigned null, since the cam plate is not used or is defined at a later time.**
iVeloCamLookUp	I_CammingLookUp	An interface to a FB with a velocity cam plate. **This input is usually assigned null, since the cam plate is not used or is defined at a later time.**

FB_Init for FBs of the type FB_AxisHydraulicsBase



FBs of the type FB_AxisHydraulicsBase extend the constructor of FB_AxisBase. The extended constructor defines the syntax used to create an axis of this type.

Syntax:

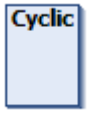
Extended in FB_AxisHydraulicBase:

```
METHOD FB_init : BOOL
VAR_INPUT
bInitRetains : BOOL;
bInCopyCode : BOOL;
AxisName : STRING;
nPtpPoints : INT;
iProcessHandler : I_ProcessHandler;
iPosCamLookup : I_CammingLookUp;
iVeloCamLookup : I_CammingLookUp;
iEncoder : I_InputBase;
iDrive : I_OutputBase;
iPressureP : I_InputBase;
iPressureM : I_InputBase;
iPosFilter : I_Filter;
iVeloFilter : I_Filter;
END_VAR
```

Inputs

Name	Type	Description
bInitRetains	BOOL	Reserved for TwinCAT PLC, see above
bInCopyCode	BOOL	Reserved for TwinCAT PLC, see above
AxisName	STRING	The text-based name of the axis.
nPtpPoints	INT	The number of supported segments in Ptp commands. **This input can be initially set to zero. In this case, the number must be specified at a later time.**
iProcessHandler	I_ProcessHandler	An interface to a FB for automatic pressure controller activation. **Reserved for extension: This input is currently always to be assigned with null.**
iPosCamLookUp	I_CammingLookUp	An interface to a FB with a position cam plate. **This input is usually assigned null, since the cam plate is not used or is defined at a later time.**
iVeloCamLookUp	I_CammingLookUp	An interface to a FB with a velocity cam plate. **This input is usually assigned null, since the cam plate is not used or is defined at a later time.**
iEncoder	I_InputBase	An interface to a function block that establishes the connection to a hardware (terminal, fieldbus device, etc.). This function block is used to determine the actual position of the axis. **A selection of typical FBs is available.**
iDrive	I_OutputBase	An interface to a function block that establishes the connection to a hardware (terminal, fieldbus device, etc.). This function block is used to determine the set velocity of the axis. **A selection of typical FBs is available.**
iPressureP	I_InputBase	An interface to a function block that establishes the connection to a hardware (terminal, fieldbus device, etc.). This function block is used to determine the actual pressure of a cylinder. **A selection of typical FBs is available.**
iPressureM	I_InputBase	An interface to a function block that establishes the connection to a hardware (terminal, fieldbus device, etc.). This function block is used to determine the actual pressure of a cylinder. **A selection of typical FBs is available.**
iPosFilter	I_Filter	An interface to a function block used for filtering the actual position of a cylinder. **A selection of typical FBs is available.**
iVeloFilter	I_Filter	An interface to a function block used for filtering the actual velocity of a cylinder. **A selection of typical FBs is available.**

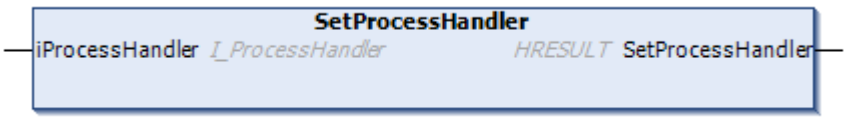
6.2.2 Cyclic



This method performs all cyclic calculations and decisions for the operation of the axis.

The application must ensure that this method is called with an appropriate cycle time for each axis.

6.2.3 SetProcessHandler



Reserved for extension: An interface to a FB for automatic pressure controller activation is entered.

7 Utilities

7.1 Functions for customizing enumerations

Tc3_MC2 and Tc2_Hydraulics libraries are using incompatible definitions for various enumerations. This is solved by the Tc3_Plastic library family by using independent definitions like E_AdaptableDirection. Tc3_PlasticMc and Tc3_PlasticHydraulic provide and use adapting functions to convert ENUM values as required.

- FUN_AdaptBufferModusNc / FUN_AdaptBufferModusHyd
- FUN_AdaptDirectionNc / FUN_AdaptDirectionHyd
- FUN_AdaptHomingDirectionNc / FUN_AdaptHomingDirectionHyd
- FUN_AdaptSwitchModeNc / FUN_AdaptSwitchModeHyd

7.2 Filter

Filters are FBs that may be applied to reduce noise in actual values like axis position or velocity, pressures, or forces. To be compatible with the intended use they must implement at least the pre-defined INTERFACE I_Filter. Tc3 Plastic libraries supply a range of basic filter FBs. Find details below.

7.2.1 FB_FilterBase

This FB is defined ABSTRACT and cannot be instantiated. It is intended to be used to derive specific filters.

Syntax:

```
FUNCTION_BLOCK ABSTRACT FB_FilterBase EXTENDS FB_MessageBase IMPLEMENTS I_Filter
```

Outputs

Name	Type	Access	Initial value	Description
Output	LREAL	Get	0.0	The current filtered value

Methods

Name	Description
Cyclic	Cyclically called by TC3 Plastic
CyclicUpdate [▶ 111]	Cyclic transfer of a new input value.

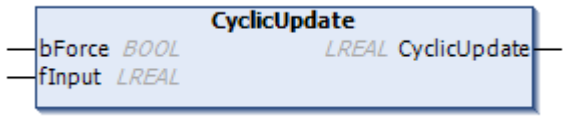
Interfaces

Type	Description
I_Filter	Basic interface for filter function blocks

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.2.1.1 CyclicUpdate



Cyclic transfer of a new value.

Syntax:

```
METHOD CyclicUpdate: LREAL
VAR_INPUT
bForce: BOOL;
fInput: LREAL;
END_VAR
```

Inputs

Name	Type	Description
bForce	BOOL	The filtered value is updated with the input value regardless of filter parameters if this input is TRUE. **Recommended standard: FALSE**
fInput	LREAL	New input value

Outputs

Name	Type	Description
CyclicUpdate	LREAL	The current filtered value

7.2.2 FB_FilterPt1

This FB derived from FB_FilterBase implements a filter of type PT1.

Syntax:

```
FUNCTION_BLOCK FB_FilterPt1 EXTENDS FB_FilterBase
```

Properties

Name	Type	Access	Initial value	Description
Output	LREAL	Get	0.0	The current filtered value
Tau	LREAL	Get, Set	1.0	The filter time constant

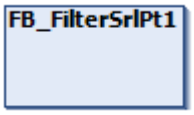
Methods

Name	Description
Cyclic	Cyclically called by TC3 Plastic
CyclicUpdate [▶ 111]	Cyclic transfer of a new input value.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.2.3 FB_FilterSlewRateLimit



This FB, derived from FB_FilterBase, implements a filter that limits the rate of rise (i.e. the ramp rate).

Syntax:

```
FUNCTION_BLOCK FB_FilterSlewRateLimit EXTENDS FB_FilterBase
```

Properties

Name	Type	Access	Initial value	Description
ChangeRateLimit				The maximum rate of change of the filtered value
Output	LREAL	Get	0.0	The current filtered value

Methods

Name	Description
Cyclic	Cyclically called by TC3 Plastic
CyclicUpdate [▶ 111]	Cyclic transfer of a new input value.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3 Simulation

A simulation requires the emulation of a process that is influenced by a controller. This is realized by sending control values to output devices that control actuators. The state of the process is detected by sensors, which respond by sending signals to input devices.

Typical examples for these devices are:

- servo drives and valves with direct fieldbus interface
- rotary encoders and sensors with direct fieldbus interface
- rotary encoders and sensors via analog input terminals
- servo drives and valves via analog output terminals

The most efficient way to connect a process simulation with a control implementation such as an application based on TC3 Plastic Functions is to replace the real I/O devices with compatible container objects.

In such a concept, the simulation must perform the following tasks:

- emulate the functionality of the output device
- simulate the behavior of the actuator

- simulate the effect on the process
- simulate the behavior of the sensor
- emulate the functionality of the input device

7.3.1 Simulation of an EtherCAT based servo drive axis

There are two types of EtherCAT-based servo drives: SoE and CoE devices. Both profiles can be successfully used in real machines, but simulating them is another matter. In CoE drives, a device reset command is triggered via a ControlWord/StatusWord mechanism, while SoE uses acyclic ADS via EtherCAT communication. Since the simulation of the SoE mechanism is much more complicated, only a CoE-based simulation is provided.

Implementing a simulated servo axis

A simulation requires an implementation that can run in one of two places:

- in the same context as the control implementation
- in a separate runtime or even on a separate CPU

Implementation in the same context is recommended in the form given below:

Name	Type	Description
{attribute 'TcContextName':='FastTask'}		Instructs TwinCAT RT to update the I/O of the FB_SimCoE402_Servo function block in a specific task. In this case it must be the calling task that executes the Cyclic() method of the simulation block.
fbNcAxisSim	FB_SimCoE402_Servo	Creates an instance of the simulation FB.
bNcAxisSimGoError	BOOL	Creates a signal with which the simulation triggers an error state and requests a reset procedure.
fbNcAxisSimLimiter	FB_SimTorqueLimitingCoE402_Servo	Creates an instance of a FB that is needed to provide the simulation with the torque limitation capability. This FB will exchange data with the control functionality via an interface access.

Implementation in a separate context is recommended as described below:

Name	Type	Description
{attribute 'TcContextName':='FastTask'}		Instructs TwinCAT RT to update the I/O of the FB_SimCoE402_Servo function block in a specific task. In this case it must be the calling task that executes the Cyclic() method of the simulation block.
fbNcAxisSim	FB_SimCoE402_Servo	Creates an instance of the simulation block.
bNcAxisSimGoError	BOOL	Creates a signal with which the simulation triggers an error state and requests a reset procedure.
{attribute 'TcContextName':='FastTask'}		Instructs the FB in line 5 to be updated in a specific task. In this case it must be the calling task that executes the Cyclic() method of the simulation block.
fbNcAxisSimLimiter	FB_SimTorqueLimitingCoE402_Servo	Creates an instance of a FB that is needed to provide the simulation with the torque limitation capability. This FB exchanges data with the control functionality via mapping.

Initialization of a simulated servo axis

In the startup phase, it is necessary to implement an initialization sequence as described below.

```
IF NOT bInitialised AND bSimulation THEN
fbNcAxisSim.EncoderWeighting:=5.0;
fbNcAxisSim.EncoderZeroShift:=100.0;
fbNcAxisSim.EncoderNoiseLevel:=0.001;
fbNcAxisSim.HighSideBlock:=700.0;
fbNcAxisSim.HighSideSpringLength:=1.0;
fbNcAxisSim.LowSideBlock:=100.0;
fbNcAxisSim.LowSideSpringLength:=1.0;
fbNcAxisSim.PositionLagKp:=1.0;
fbNcAxisSim.HighSideEndswitch:=699.0;
fbNcAxisSim.LowSideEndswitch:=101.0;
fbNcAxisSim.AbsSwitchHighEnd:=679.0;
fbNcAxisSim.AbsSwitchLowEnd:=21.0;
// the next line must be used if the simulation
// is implemented in a separate context
fbNcAxisSimLimiterMapped.ConnectToSim(ipSim:=fbNcAxis2Sim);
// the next two lines must be used if the simulation
// is implemented in the same context
fbNcAxisSimLimiter.ConnectToSim(ipSim:=fbNcAxis2Sim);
iNcAxis_D.SetTorqueLimiting(fbNcAxisSimLimiter);
END_IF
```

The last line defines the FB used by the axis for torque limitation when the simulation is performed in the same context. In this case the penultimate line is required to connect this FB to the drive simulation.

Above these lines is an example of implementation in a separate context.

For details on the specified parameters, refer to the documentation below.

Mapping of a simulated servo axis

The simulation FB provides a local instance of a mapping interface defined as follows:

```
NcAdapt : FB_SimCoE402_ServoNcAdapt;
```

Details of this FB can be found in the documentation below.

The supplied mapping structures for encoder and drive inputs and outputs are compatible with the TwinCAT NC interfaces and can thus be used.

Functionality of the CoE402 simulation

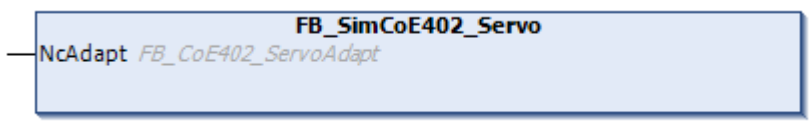
All mechanisms that use mapped interfaces are supported in a way that is expected by TwinCAT NC. This also includes torque limitation.

Since there is no way to emulate ADS communication, some mechanisms cannot be supported.

- Simulated axes cannot be supported by DriveManager. There is no servo drive and no motor. The commissioning requires just parameters and must be done hard coded in the application project.
- The Advanced Homing library of TwinCAT NC uses the ADS communication to disable some functions of the AX servo drives while the homing procedures are executed. The core functions of the TC3 Plastic Functions provide methods to avoid the use of this communication. The homing mechanisms are also ready for use with simulated servo axes.

7.3.1.1 NC simulation components

7.3.1.1.1 FB_SimCoE402_Servo



With such a FB a simulation of a CoE402 servo drive is realized. The FB offers mapping elements for CoE interfaces of servo drives and the mapping is done in the same way as for the real drive unit.

Syntax:

```
FUNCTION_BLOCK FB_SimCoE402_Servo EXTENDS FB_SimAxCommon IMPLEMENTS I_SimCoE402_Servo
VAR_INPUT
    NcAdapt: FB_CoE402_ServoAdapt;
END_VAR
VAR_OUTPUT
END_VAR
```

Inputs

Name	Type	Description
NcAdapt	FB_CoE402_ServoAdapt	The mapping interface for the NC axis.

Properties

Name	Type	Access	Description
AbsolutSwitch	BOOL	Get	The signal of the simulated absolute switch.
AbsSwitchHighEnd	LREAL	Get, Set	The position at which an absolute switch is simulated at the upper end of the travel path.
AbsSwitchHighSelect	BOOL	Get, Set	A TRUE selects the absolute switch at the upper end of the travel path. A FALSE selects the absolute switch at the lower end of the travel path.
AbsSwitchLowEnd	LREAL	Get, Set	The position at which an absolute switch is simulated at the lower end of the travel path.
ActualPosition	LREAL	Get, Set	The actual position of the simulated servo.
ActualTorque	LREAL	Get	The simulated actual torque.
CycleTime	LREAL	Get, Set	The cycle time with which the Cyclic method is called.
EncoderInterpolation	LREAL	Get, Set	The simulated encoder interpolation. **This value must match the parameter of the NC axis.**
EncoderNoiseLevel	LREAL	Get, Set	A noise component for the signaled actual position can be specified here.
EncoderWeighting	LREAL	Get, Set	The simulated encoder weighting. **This value must match the parameter of the NC axis.**
EncoderZeroShift	LREAL	Get, Set	The simulated zero offset shift. **Depending on homing methods applied, this value may differ from that in the NC axis.**
ExternalTorque	LREAL	Get, Set	Here, an external torque can be specified from the simulation of a simulated process.
HighSideBlock	LREAL	Get, Set	The simulated mechanical block at the upper end of the travel path.
HighSideEndswitch	LREAL	Get, Set	The position at which a hardware limit switch is simulated at the upper end of the travel path.
HighSideSpringLength	LREAL	Get, Set	A spring travel before the upper mechanical block. **This simulates a torque increase.**
LowerEndSwitch	BOOL	Get	The signal of the limit switch at the lower end of the travel path.
LowSideBlock	LREAL	Get, Set	The simulated mechanical block at the lower end of the travel path.

Name	Type	Access	Description
LowSideEndswitch	LREAL	Get, Set	The position at which a hardware limit switch is simulated at the lower end of the travel path.
LowSideSpringLength	LREAL	Get, Set	A spring travel before the lower mechanical block. **This simulates a torque increase.**
MovingMass	LREAL	Get, Set	A mass can be specified here. It is used to simulate a torque proportional to the acceleration.
NoBlock	BOOL	Get, Set	A TRUE suppresses the mechanical stops at the upper and lower end of the travel path. The axis can travel endlessly.
PositionLagKp	LREAL	Get, Set	The position control gain of the simulated servo.
Reversed	BOOL	Get, Set	TRUE if the direction of rotation of the simulated servo is inverted.
StatusWord	WORD	Get	The status word of the servo
TorqueLimitNegative	LREAL	Get, Set	The negative torque limit.
TorqueLimitPositive	LREAL	Get, Set	The positive torque limit.
UpperEndSwitch	BOOL	Get	The signal of the limit switch at the upper end of the travel path.

NOTICE

When using a simulated servo drive, a FB of type FB_SimTorqueLimitingCoE402_Servo or FB_SimTorqueLimitingCoE402_Mapped must be used to perform the data exchange between TC3 Plastic Functions and the simulation.

Methods

Name	Description
Cyclic [▶ 116]	This method must be called cyclically.
GoFaultState [▶ 116]	This method can be used to simulate an error situation on the simulated axis.
Process [▶ 116]	This method is called once per cycle to calculate the actual values for position and torque.

7.3.1.1.1.1 Cyclic

```
METHOD Cyclic : BOOL
```

This method must be called cyclically. It performs all the calculations of the simulation.

7.3.1.1.1.2 GoFaultState

```
METHOD GoFaultState : BOOL
```

This method can be used to simulate an error situation on the simulated axis.

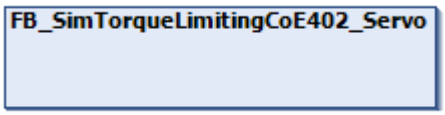
The simulated axis does not respond to setpoints, updates the simulated drive state machine and reports the problem via the status word.

7.3.1.1.1.3 Process

```
METHOD Process : BOOL
```

This method is called once per cycle to calculate the actual values for position and torque.

7.3.1.1.2 FB_SimTorqueLimitingCoE402_Servo



If the simulation is performed in the same task in which the process software of the axis is running: A FB of this type must be used to provide a simulated servo drive with a torque limitation.

If the simulation is not performed in the same task in which the process software of the axis is running: A FB of this type must be used to provide a simulated servo drive with a torque limitation and to connect it by a mapping.

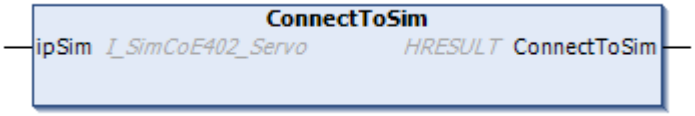
Properties

Name	Type	Access	Description
Activate	BOOL	Get, Set	A TRUE signals that a torque limitation is active.

Methods

Name	Description
ConnectToSim [▶ 117]	This method establishes a connection between the torque-limiting FB and the simulated servo drive.
Cyclic [▶ 118]	This method, to be called cyclically, performs all calculations and decisions.

7.3.1.1.2.1 ConnectToSim



This method establishes a connection between the torque-limiting FB and the simulated servo drive. This connection replaces the EtherCAT communication with a real CoE402-based servo drive.

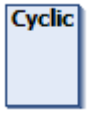
Syntax:

```
METHOD ConnectToSim : HRESULT
VAR_INPUT
ipSim: I_SimCoE402_Servo;
END_VAR
```

Inputs

Name	Type	Description
ipSim	I_SimCoE402_Servo	The simulation block for the simulation of a CoE402 servo.

7.3.1.1.2.2 Cyclic



This method is used to perform the necessary calculations. It must be called cyclically.

Syntax:

```
METHOD Cyclic
```

7.3.2 Simulation of an inverter drive axis

This simulation corresponds to an inverter axis and can be used to replace a real inverter drive. For this purpose, compatible mapping elements are provided and a behavior very similar to that of a real axis is emulated.

Implementation of a simulated inverter drive axis

A simulation requires an implementation that can run in one of two places:

- in the same context as the control implementation
- in a separate runtime or even on a separate CPU

Implementation in both contexts is recommended as described below:

Name	Type	Description
{attribute 'TcContextName':='FastTask'}		Instructs FB_SimCoE402_Servo to be updated in a specific task. In this case it must be the calling task that executes the Cyclic() method of the simulation block.
fbInvAxisSim	FB_SimCoE402_Inverter_Xyz	Creates an instance of the simulation FB.
bInvAxisSimGoError	BOOL	Creates a signal for the simulation to report an error state and request a reset procedure.

Range of supported inverters

The term FB_SimCoE402_Inverter_Xyz above must be replaced by one of the following options:

FB_SimCoE402_Inverter_CoE402FI

FB_SimCoE402_Inverter_CoE402SI

Initialization of a simulated inverter drive axis

In the startup phase, it is necessary to implement an initialization sequence as described below.

```
IF NOT bInitialised AND bSimulation THEN
fbInverterSimDS402.CycleTime := 0.002;
fbInverterSimDS402.MaxTurnRate := 1380.0;
fbInverterSimDS402.MinTurnrate := 45.0;
fbInverterSimDS402.RampTime := 2.0;
fbInverterSimDS402.OutputFactor := 1.0;
fbInverterSimDS402.ReferenceLoad := 100.0;
END_IF
```

The parameters must correspond to the behavior of the simulated device, not to the intended use.

7.3.2.1 Adaptation of an inverter

7.3.2.1.1 FB_SimCoE402_Inverter_CoE402FI

This simulation corresponds to an inverter axis and can be used to replace a real inverter drive by providing compatible mapping elements and emulating behavior very similar to that of a real axis.

The FB provides mapping elements for CoE interfaces of basic frequency inverters (FI) that match the interfaces implemented by the adapter FB for this drive type. Mapping is done in the same way as for the real drive unit.

Syntax:

```
FUNCTION_BLOCK FB_InverterAdaption_CoE402FI
VAR_INPUT
CoE402FI_Outputs AT %Q* : ST_CoE402FI_Outputs;
CoE402FI_Inputs AT %I*  : ST_CoE402FI_Inputs;
EtC_Device AT %I*       : FB_EtC_Device;
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

Name	Type	Access	Description
InverterType	E_Tc3pInverterType	Get	The type of the simulated inverter.
Load	LREAL	Get, Set	Here you can specify a load that is claimed by a simulated process.
MaxTurnRate	LREAL	Get, Set	The maximum turn rate of the drive.
MinTurnrate	LREAL	Get, Set	A minimum turn rate can be specified here. If a value > 0.0 is specified, the drive ignores set turn rates below this threshold and a dead band is created. If the default is less than 10 RPM, it is assumed that the inverter supports vector control.
RampTime	LREAL	Get, Set	Here the time for the ramp from zero to MaxTurnRate or vice versa is defined.
ReferenceLoad	LREAL	Get, Set	A reference value for Load.
VectorControlled	BOOL	Get	A TRUE specifies that the actual turn rate is load independent. If FALSE, the inverter responds to a load torque with a proportional slip. The simulation assumes a slip of 0 to 50 % for a load of 0 to 100 % of the reference load.

Methods

Name	Description
Cyclic	This method must be called cyclically by the application.
TriggerError	A call to this method places the simulated drive into the error state.

7.3.2.1.2 FB_SimCoE402_Inverter_CoE402SI

This simulation corresponds to an inverter axis and can be used to replace a real inverter drive by providing compatible mapping elements and emulating behavior very similar to that of a real axis.

The FB provides mapping elements for CoE interfaces of frequency inverters that implement a servo-like architecture and that match the interfaces implemented by the adapter FB for this type of drives. Mapping is done in the same way as for the real drive unit.

Syntax:

```
FUNCTION_BLOCK FB_InverterAdaption_CoE402SI
VAR_INPUT
CoE402SI_Outputs AT %Q* : ST_CoE402SI_Outputs;
CoE402SI_Inputs AT %I*  : ST_CoE402SI_Inputs;
EtC_Device AT %I*       : FB_EtC_Device;
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

Name	Type	Access	Description
InverterType	E_Tc3pInverterType	Get	The type of the simulated inverter.
Load	LREAL	Get, Set	Here you can specify a load that is claimed by a simulated process.
MaxTurnRate	LREAL	Get, Set	The maximum turn rate of the drive.
MinTurnrate	LREAL	Get, Set	A minimum turn rate can be specified here. If a value > 0.0 is specified, the drive ignores set turn rates below this threshold and a dead band is created. If the default is less than 10 RPM, it is assumed that the inverter supports vector control.
OutputFactor	LREAL	Get, Set	The factor used to exchange turn rates with the device.
RampTime	LREAL	Get, Set	Here the time for the ramp from zero to MaxTurnRate or vice versa is defined.
ReferenceLoad	LREAL	Get, Set	A reference value for Load.
VectorControlled	BOOL	Get	A TRUE specifies that the actual turn rate is load independent. If FALSE, the inverter responds to a load torque with a proportional slip. The simulation assumes a slip of 0 to 50 % for a load of 0 to 100 % of the reference load.

Methods

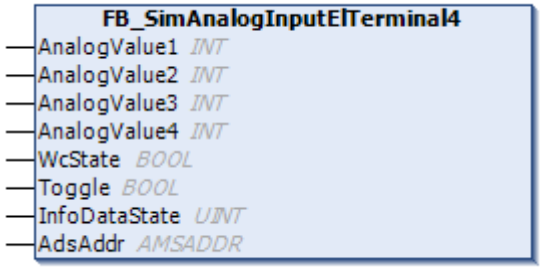
Name	Description
Cyclic	This method must be called cyclically by the application.
TriggerError	A call to this method places the simulated drive into the error state.

7.3.3 I/O Simulation containers

To create a simulation, function blocks are needed that can replace an I/O device in a compatible way.

Name	Channels	Description
FB_SimAnalogInputElTerminal4 [▶ 121]	4	Simulation of an analog EtherCAT input terminal. Example EL3134
FB_SimAnalogOutputElTerminal4 [▶ 122]	4	Simulation of an analog EtherCAT output terminal. Example EL4134
FB_SimSsiInputElTerminal1 [▶ 123]	1	Simulation of an EtherCAT SSI input terminal. Example EL5001
FB_CoE402_ServoAdapt [▶ 122]	1	AX8000

7.3.3.1 Simulation of an analog input terminal



Simulation of a 4-channel analog input terminal. This FB only provides a mapping interface and does not contain an implementation.

Syntax:

```
FUNCTION_BLOCK FB_SimAnalogInputElTerminal4
```

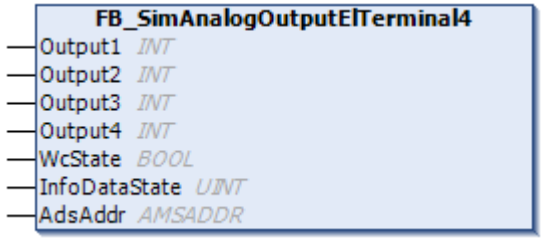
Inputs

Name	Type	Access	Initial value	Description
AnalogValue1	INT	IN	0	The simulated input value of the 1st channel.
AnalogValue2	INT	IN	0	The simulated input value of the 2nd channel.
AnalogValue3	INT	IN	0	The simulated input value of the 3rd channel.
AnalogValue4	INT	IN	0	The simulated input value of the 4th channel.
WcState	BOOL	OUT	FALSE	The simulated Working Counter State.
Toggle	BOOL	OUT	FALSE	The simulated toggle bit.
InfoDataState	UINT	OUT	8	The simulated Device State.
AdsAddr	AMSADDR	OUT		The simulated ADS address.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3.3.2 Simulation of an analog output terminal



Simulation of a 4-channel analog output terminal. This FB only provides a mapping interface and does not contain an implementation.

Syntax:

```
FUNCTION_BLOCK FB_SimAnalogOutputElTerminal4
```

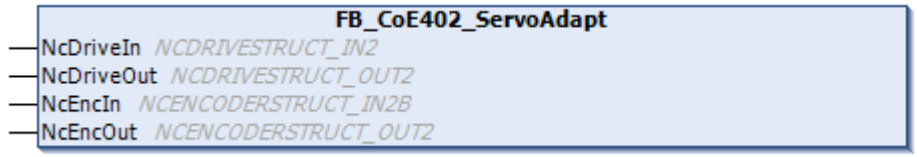
Inputs

Name	Type	Access	Initial value	Description
Output1	INT	IN	0	The simulated output value of the 1st channel.
Output2	INT	IN	0	The simulated output value of the 2nd channel.
Output3	INT	IN	0	The simulated output value of the 3rd channel.
Output4	INT	IN	0	The simulated output value of the 4th channel.
WcState	BOOL	OUT	FALSE	The simulated Working Counter State.
InfoDataState	UINT	OUT	8	The simulated Device State.
AdsAddr	AMSADDR	OUT		The simulated ADS address.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3.3.3 Simulation of a CoE402 servo drive



Simulation of a CoE408 servo drive. This FB only provides a mapping interface and does not contain an implementation.

Syntax:

```
FUNCTION_BLOCK FB_CoE402_ServoAdapt
```

Inputs

Name	Type	Access	Description
NcDriveIn	NCDRIVESTRUCT_IN2	OUT	For the DriveIn interface of the NC.

Name	Type	Access	Description
NcDriveOut	NCDRIVESTRUCT_OUT2	IN	For the DriveOut interface of the NC.
NcEncIn	NCENCODERSTRUCT_IN2B	OUT	For the EncoderIn interface of the NC.
NcEncOut	NCENCODERSTRUCT_OUT2	IN	For the EncoderOut interface of the NC.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3.3.4 Simulation of an SSI input terminal

Simulation of a 1-channel SSI input terminal. This FB only provides a mapping interface and does not contain an implementation.

Syntax:

```
FUNCTION_BLOCK FB_SimSsiInputElTerminal1
```

Inputs

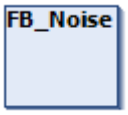
Name	Type	Access	Initial value	Description
AnalogValue1	INT	IN	0	The simulated input value of the 1st channel.
AnalogValue2	INT	IN	0	The simulated input value of the 2nd channel.
AnalogValue3	INT	IN	0	The simulated input value of the 3rd channel.
AnalogValue4	INT	IN	0	The simulated input value of the 4th channel.
WcState	BOOL	OUT	FALSE	The simulated Working Counter State.
Toggle	BOOL	OUT	FALSE	The simulated toggle bit.
InfoDataState	UINT	OUT	8	The simulated Device State.
AdsAddr	AMSADDR	OUT		The simulated ADS address.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3.4 Common simulation components

7.3.4.1 FB_Noise



This FB is used to generate a pseudo-random signal that resembles a white noise disturbance. Noise signals may be used to simulate a common problem of analog sensors and interfaces.

Objects of this type are typically used as local elements in an implementation of an axis simulation.

Syntax:

```
FUNCTION_BLOCK FB_Noise
```

Methods

Name	Description
Cyclic [▶ 124]	Cyclic call to generate a new value.
SetLineNoiseLevel [▶ 124]	Determination of the level of an influence by a supply network.
SetSparkNoiseLevel [▶ 125]	Determination of the level of an influence by static discharges.
SetWhiteNoiseLevel [▶ 125]	Setting the level of the noise signal.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3.4.1.1 Cyclic

Method to be called cyclically to generate a new value.

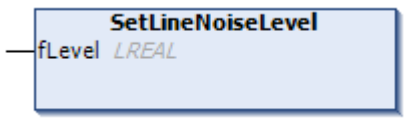
Syntax:

```
METHOD Cyclic : LREAL
```

Outputs

Name	Type	Description
Cyclic	LREAL	The new value of the noise signal

7.3.4.1.2 SetLineNoiseLevel



Determination of the level of an influence by a supply network.

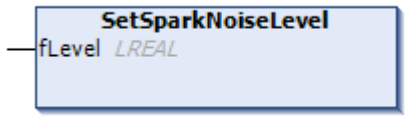
Syntax:

```
METHOD SetLineNoiseLevel
VAR_INPUT
fLevel: LREAL;
END_VAR
```

Inputs

Name	Type	Description
fLevel	LREAL	Level of an influence by a supply network. **Recommended standard: depending on the use of the signal.**

7.3.4.1.3 SetSparkNoiseLevel



Determination of the level of an influence by static discharges.

Syntax:

```
METHOD SetSparkNoiseLevel
VAR_INPUT
fLevel: LREAL;
END_VAR
```

Inputs

Name	Type	Description
fLevel	LREAL	Level of an influence by static discharges. **Recommended standard: depending on the use of the signal.**

7.3.4.1.4 SetWhiteNoiseLevel

Setting the level of the noise signal.

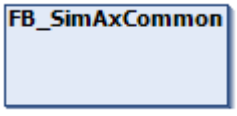
Syntax:

```
METHOD SetWhiteNoiseLevel
VAR_INPUT
fLevel: LREAL;
END_VAR
```

Inputs

Name	Type	Description
fLevel	LREAL	Level of the noise signal. **Recommended standard: depending on the use of the signal.**

7.3.4.2 FB_SimAxCommon



Such a FB is used as a common platform for performing axle simulations of various types. It is not intended to instantiate objects.

Syntax:

```
PROPERTY AbsolutSwitch       : BOOL (r/o)
PROPERTY AbsSwitchHighEnd    : LREAL (r/w)
PROPERTY AbsSwitchHighSelect : BOOL (r/w)
PROPERTY AbsSwitchLowEnd     : LREAL
PROPERTY ActualPosition      : LREAL
PROPERTY CycleTime           : LREAL
PROPERTY EncoderInterpolation: LREAL
PROPERTY EncoderNoiseLevel   : LREAL
PROPERTY EncoderWeighting    : LREAL
PROPERTY EncoderZeroShift    : LREAL
METHOD GoFaultState          : BOOL
PROPERTY HighSideBlock       : LREAL
PROPERTY HighSideEndswitch   : LREAL
PROPERTY HighSideSpringLength: LREAL
PROPERTY LowerEndSwitch      : BOOL
PROPERTY LowSideBlock        : LREAL
PROPERTY LowSideEndswitch    : LREAL
PROPERTY LowSideSpringLength : LREAL
PROPERTY MovingMass          : LREAL
PROPERTY NoBlock             : BOOL
PROPERTY UpperEndSwitch      : BOOL
```

Properties

Name	Type	Access	Description
AbsolutSwitch	BOOL	Get	This runtime value provides information about the simulated feedback signal as defined by AbsSwitchHighSelect.
AbsSwitchHighEnd	LREAL	Get, Set	This parameter defines the threshold in case AbsSwitchHighSelect=TRUE: AbsolutSwitch will be TRUE if the simulated actuator position is >= AbsSwitchHighEnd. **The simulated position of the actuator can deviate from the reported axis position.**
AbsSwitchHighSelect	BOOL	Get, Set	This parameter defines the active AbsolutSwitch. A TRUE will select AbsSwitchHighEnd while a FALSE makes the AbsSwitchLowEnd the active AbsolutSwitch.
AbsSwitchLowEnd	LREAL	Get, Set	This parameter defines the threshold in case AbsSwitchHighSelect=FALSE: AbsolutSwitch will be TRUE if the simulated actuator position is <= AbsSwitchLowEnd. **The simulated position of the actuator can deviate from the reported axis position.**
ActualPosition	LREAL	Get, Set	This runtime value provides information about the simulated axis position. **The simulated position of the actuator can deviate from the reported axis position.**
CycleTime	LREAL	Get, Set	This parameter must be initialized with the update calling cycle of the Cyclic() method and will be forwarded to any sub-component if required. It is used to define any time behavior.
EncoderInterpolation	LREAL	Get, Set	This parameter must be initialized with the same value as the corresponding parameter in the motion technology: • NC: encoder, parameter, scaling factor denominator • Hydraulics library: encoder, inc. Interpolation

Name	Type	Access	Description
EncoderNoiseLevel	LREAL	Get, Set	The simulated axis offers the possibility to disturb the reported position with a pseudo-random white noise. This is realized by a FB_Noise() function block. The parameter EncoderNoiseLevel is forwarded as SetWhiteNoiseLevel.
EncoderWeighting	LREAL	Get, Set	This parameter must be initialized with the same value as the corresponding parameter in the motion technology: • NC: encoder, parameter, scaling factor numerator • Hydraulics library: encoder, weighting factor
EncoderZeroShift	LREAL	Get, Set	This parameter must be initialized with the same value as the corresponding parameter in the motion technology: • NC: rotary encoder, parameter, position bias • Hydraulics library: encoder, zero offset
HighSideBlock	LREAL	Get, Set	The simulated position of the actuator is limited to a value less than or equal to this parameter.
HighSideEndswitch	LREAL	Get, Set	This parameter defines the UpperEndSwitch threshold that becomes TRUE when the simulated position of the actuator is >= this parameter.
HighSideSpringLengt	LREAL	Get, Set	This parameter defines the length of a simulated spring-like effect at the upper side of the actuator stroke.
LowerEndSwitch	BOOL	Get	This runtime value provides information about the state of a simulated sensor. Becomes TRUE if the position of the simulated actuator is >= HighSideEndswitch.
LowSideBlock	LREAL	Get, Set	The simulated position of the actuator is limited to a value above or equal to this parameter.
LowSideEndswitch	LREAL	Get, Set	This parameter defines the LowerEndSwitch threshold that becomes TRUE when the simulated position of the actuator is <= this parameter.
LowSideSpringLength	LREAL	Get, Set	This parameter defines the length of a simulated spring-like effect at the lower side of the actuator stroke.
MovingMass	LREAL	Get, Set	This parameter can be used to define a moving mass. It is used to calculate dynamic acceleration and deceleration forces, torques or pressures.
NoBlock	BOOL	Get, Set	If this parameter is set to TRUE, the use of the springs and blocks on the upper and lower sides will be disabled.
UpperEndSwitch	BOOL	Get	This runtime value provides information about the state of a simulated sensor. It becomes TRUE if the position of the simulated actuator is <= LowSideEndswitch.

Methods

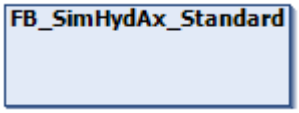
Name	Description
GoFaultState	This method can be used to simulate an error situation on the simulated axis.

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT v3.1.4024.35	PC or CX (x64, x86)	Tc3_PlasticFunction V3.12.4.26 or higher

7.3.5 Components of the hydraulic simulation

7.3.5.1 FB_SimHydAx_Standard



This FB is a simplified simulation of a universal hydraulic axis. A number of typical effects are supported.

The implementation of this object is not a full-fledged and realistic simulation.

Syntax:

```
FUNCTION_BLOCK FB_SimHydAx_Standard EXTENDS FB_SimAxCommon IMPLEMENTS I_SimHydAx
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

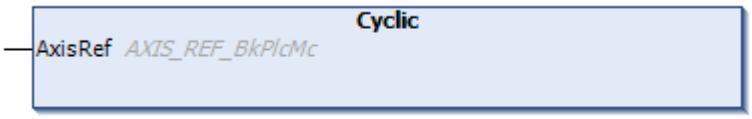
Name	Type	Access	Description
AbsolutSwitch	BOOL	Get	The signal of the simulated absolute switch.
AbsSwitchHighEnd	LREAL	Get, Set	The position at which an absolute switch is simulated at the upper end of the travel path.
AbsSwitchHighSelect	BOOL	Get, Set	A TRUE selects the absolute switch at the upper end of the travel path. A FALSE selects the absolute switch at the lower end of the travel path.
AbsSwitchLowEnd	LREAL	Get, Set	The position at which an absolute switch is simulated at the lower end of the travel path.
ActualPosition	LREAL	Get, Set	The actual position of the simulated axis.
CycleTime	LREAL	Get, Set	The cycle time with which the Cyclic method is called.
Cylinder	I_SimCylinder	Get	An interface to the local FB_SimCylinder object within the simulated axis.
Encoder	I_SimUniversalEncoder	Get	An interface to the local FB_SimUniversalEncoder object within the simulated axis.
EncoderInterpolation	LREAL	Get, Set	The simulated encoder interpolation. **This value must match the parameter of the NC axis.**
EncoderNoiseLevel	LREAL	Get, Set	A noise component for the signaled actual position can be specified here.
EncoderWeighting	LREAL	Get, Set	The simulated encoder weighting. **This value must match the parameter of the NC axis.**

Name	Type	Access	Description
EncoderZeroShift	LREAL	Get, Set	The simulated zero offset shift. **Depending on homing methods applied, this value may differ from that in the NC axis.**
ExternalForce	LREAL	Get, Set	This runtime value can be updated with pressure values required for process simulation. It is used to calculate the total pressure on the simulated cylinder and is part of the reported actual pressure.
HighSideBlock	LREAL	Get, Set	The simulated mechanical block at the upper end of the travel path.
HighSideEndswitch	LREAL	Get, Set	The position at which a hardware limit switch is simulated at the upper end of the travel path.
HighSideSpringLength	LREAL	Get, Set	A spring travel before the upper mechanical block. **This simulates a pressure increase.**
LowSideBlock	LREAL	Get, Set	The simulated mechanical block at the lower end of the travel path.
LowSideEndswitch	LREAL	Get	The signal of the limit switch at the lower end of the travel path.
LowSideSpringLength	LREAL	Get, Set	A spring travel before the lower mechanical block. **This simulates a pressure increase.**
MovingMass	LREAL	Get, Set	A mass can be specified here. It is used to simulate a torque proportional to the acceleration.
NoBlock	BOOL	Get, Set	A TRUE suppresses the mechanical stops at the upper and lower end of the travel path. The axis can travel endlessly.
PressureTransducerA	I_PressureTransducer		An interface to the local FB_PressureTransducer object on the A-side of the FB_SimCylinder object within the simulated axis.
PressureTransducerB	I_PressureTransducer		An interface to the local FB_PressureTransducer object on the B-side of the FB_SimCylinder object within the simulated axis.
Valve	I_SimValve		An interface to the local FB_PressureTransducer object on the B-side of the FB_SimCylinder object within the simulated axis.

Methods

Name	Description
Cyclic [▶ 130]	This method performs all calculations of the simulation.
GoFaultState [▶ 130]	This method can be used to simulate an error situation on the simulated axis.

7.3.5.1.1 Cyclic



This method must be called cyclically. It performs all the calculations of the simulation.

Syntax:

```
METHOD Cyclic
VAR_IN_OUT
AxisRef: AXIS_REF_BkPlcMc;
END_VAR
```

Inputs

Name	Type	Description
AxisRef	AXIS_REF_BkPlcMc	The AXIS-REF of the hydraulic axis.

7.3.5.1.2 GoFaultState

This method can be used to simulate an error situation on the simulated axis. The axis reports a subsequent error and is set to the fault state.

7.3.5.2 FB_SimValveAnalog

This simplified simulation of an analog proportional or servo valve is used as part of a simulated hydraulic axis. It supports a range of typical variants of non-linearities. There is no dynamic behavior.

The implementation of this object is not a full-fledged and realistic simulation.

Objects of this type are typically used as local elements in an implementation of an axis simulation.

Syntax:

```
FUNCTION_BLOCK ABSTRACT FB_SimValve IMPLEMENTS I_SimValve
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

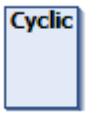
Name	Type	Access	Description
AdsAddr	AMSADDR	Get, Set	This variable is used to implement a mapping interface of an analog valve.
ControlValue	INT	Get, Set	This runtime value must be updated cyclically with an output property of a FB_SimAnalogOutputElTerminal4 instance. **This assignment can be understood as the cable connecting an output terminal to an input pin of the valve connector.**
ControlValueNorm	LREAL	Get	This runtime value provides information about the current control value in abstract units (e.g. %).
ControlValueType	E_SimControlValueType	Get, Set	This parameter must be initialized with a value that specifies the type of the control signal. It must match the type of the simulated output terminal.
InfoDataState	UINT	Get, Set	This variable is used to implement a mapping interface of an analog valve.
Orifice_PA	LREAL	Get	This runtime value provides information about the current opening between the supply port and the A-side port of the valve.
Orifice_PB	LREAL	Get	This runtime value provides information about the current opening between the supply port and the B-side port of the valve.
Overlapp_PA	LREAL	Get, Set	This parameter must be initialized with the overlap on the A-side of the valve. **Typical values can be found in the manufacturer's data sheets.**
Overlapp_PB	LREAL	Get, Set	This parameter must be initialized with the overlap on the B-side of the valve. **Typical values can be found in the manufacturer's data sheets.**
P_A	LREAL	Get, Set	This runtime value reports about the pressure at the A-side port of the valve.
P_B	LREAL	Get, Set	This runtime value reports about the pressure at the B-side port of the valve.
Q_PA	LREAL	Get	This runtime value reports about the oil flow between the supply port and the A-side port of the valve.
Q_PB	LREAL	Get	This runtime value reports about the oil flow between the supply port and the B-side port of the valve.
Qnominal_PA	LREAL	Get, Set	This parameter must be initialized with the capacity of the supply to the A-side opening of the simulated valve. **Typical values can be found in the manufacturer's data sheets.**
Qnominal_PB	LREAL	Get, Set	This parameter must be initialized with the capacity of the supply to the B-side opening of the simulated valve. **Typical values can be found in the manufacturer's data sheets.**
SpoolFeedback	INT	Get	This runtime value reports about the actual position of the valve piston. It is used to implement a mapping interface of an analog valve.
SupplyPressure	LREAL	Get, Set	This runtime value must be updated with the supply pressure. If the value is assumed to be constant, it can be updated once during initialization.

Name	Type	Access	Description
Toggle	BOOL	Get	This variable is used to implement a mapping interface of an analog valve.
ValveSpoolType	E_SimValveSpoolType	Get, Set	This parameter sets the transfer characteristic of the valve.
WcState	BOOL	Get, Set	This variable is used to implement a mapping interface of an analog valve.

Methods

Name	Description
Cyclic [▶ 132]	This method must be called cyclically. It performs all the calculations of the simulation.

7.3.5.2.1 Cyclic

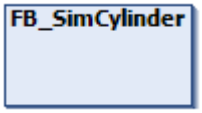


This method must be called cyclically. It performs all the calculations of the simulation.

Syntax:

```
METHOD Cyclic
VAR_INPUT
END_VAR
```

7.3.5.3 FB_SimCylinder



This simplified simulation of a universal cylinder is used as part of a simulated hydraulic axis. A number of typical effects are supported.

The implementation of this object is not a full-fledged and realistic simulation.

Objects of this type are typically used as local elements in an implementation of an axis simulation.

Properties

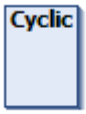
Name	Type	Access	Description
CycleTime	LREAL	Get, Set	This parameter must be initialized with the call cycle time.
ExternalForce	LREAL	Get, Set	This runtime value can be used to update an external force in the simulation calculation. It is used for conversion to pressure values.
HighSideBlock	LREAL	Get, Set	The simulated upper mechanical stop.
HighSideSpringLength	LREAL	Get, Set	The simulated spring action before the upper mechanical stop.

Name	Type	Access	Description
LowSideBlock	LREAL	Get, Set	The simulated lower mechanical stop.
LowSideSpringLength	LREAL	Get, Set	The simulated spring action before the lower mechanical stop.
P_A	LREAL	Get, Set	The simulated pressure at the A-side cylinder port.
P_B	LREAL	Get, Set	The simulated pressure at the B-side cylinder port.
PistonDiameter	LREAL	Get, Set	The diameter of the cylinder piston. It is used to calculate the effective areas of the cylinder.
Position	LREAL	Get, Set	The current position of the cylinder.
Q_PA	LREAL	Get, Set	This runtime value provides information about the oil flow flowing into or out of the A-side cylinder port. **This value is usually determined by a valve simulation and passed on by a hydraulic axis simulation.**
Q_PB	LREAL	Get, Set	This runtime value provides information about the oil flow flowing into or out of the A-side cylinder port. **This value is usually determined by a valve simulation and passed on by a hydraulic axis simulation.**
RodDiameter	LREAL	Get, Set	The diameter of the rod of the cylinder. It is used to calculate the effective areas of the cylinder.
Stroke	LREAL	Get, Set	The stroke length (i.e. the distance between the lower and upper mechanical end positions) of the cylinder. It is used to limit the actual position.

Methods

Name	Description
Cyclic [▶ 133]	This method must be called cyclically. It performs all the calculations of the simulation.

7.3.5.3.1 Cyclic

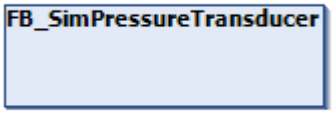


This method must be called cyclically. It performs all the calculations of the simulation.

Syntax:

```
METHOD Cyclic
VAR_INPUT
END_VAR
```

7.3.5.4 FB_SimPressureTransducer



This simplified simulation of a universal pressure transducer is used as component of a simulated hydraulic axis. A number of typical effects are supported.

The implementation of this object is not a full-fledged and realistic simulation.

Objects of this type are typically used as local elements in an implementation of an axis simulation.

Syntax:

```
FUNCTION_BLOCK FB_SimPressureTransducer IMPLEMENTS I_PressureTransducer
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
```

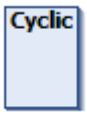
Properties

Name	Type	Access	Description
AdcValue	INT	Get	This runtime value contains the input value of the simulated terminal.
CycleTime	LREA	Get, Set	This parameter must be initialized with the call cycle time.
NoiseLevel	LREAL	Get, Set	This parameter can be used to set the amplitude of a simulated noise signal that is added to the pressure at the sensor.
NominalPressure	LREAL	Get, Set	This parameter must be initialized with the rated pressure (i.e. full scale) of the simulated transducer.
Pressure	LREAL	Get, Set	This runtime value must be updated cyclically with the pressure of a specific point in a simulated hydraulic system.
Signal	LREAL	Get	This runtime value contains the electrical signal of a simulated 0 to 10 V pressure transducer.

Methods

Name	Description
Cyclic [▶ 134]	This method must be called cyclically. It performs all the calculations of the simulation.

7.3.5.4.1 Cyclic



This method must be called cyclically. It performs all the calculations of the simulation.

Syntax:

```
METHOD Cyclic
VAR_INPUT
END_VAR
```

7.3.5.5 FB_SimUniversalEncoder

This simplified simulation of a universal encoder is used as part of a simulated hydraulic axis. A number of typical effects are supported.

The implementation of this object is not a full-fledged and realistic simulation.

Objects of this type are typically used as local elements in an implementation of an axis simulation.

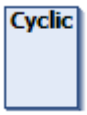
Properties

Name	Type	Access	Description
AdsAddr	AMSADDR	Get	This variable is used to implement a mapping interface of an analog valve.
Count_UDINT	UDINT	Get	This variable is used to implement a mapping interface of an analog valve.
Count_UINT	UINT	Get	This variable is used to implement a mapping interface of an analog valve.
InfoDataState	UINT	Get, Set	This variable is used to implement a mapping interface of an analog valve.
NoiseLevel	LREAL	Get, Set	This parameter must be initialized with the white noise amplitude to be simulated.
Offset	LREAL	Get, Set	This parameter must be initialized with the displacement between the cylinder zero point (e.g. the lower stop) and the encoder zero point.
Position	LREAL	Get, Set	This runtime value provides information about the actual position of the encoder.
Resolution	LREAL	Get, Set	This parameter must be initialized with the resolution (i.e. the path per increment) of the encoder.
Toggle	BOOL	Get	This variable is used to implement a mapping interface of an analog valve.
WcState	BOOL	Get, Set	This variable is used to implement a mapping interface of an analog valve.

Methods

Name	Description
Cyclic [▶ 135]	This method performs all calculations of the simulation.

7.3.5.5.1 Cyclic



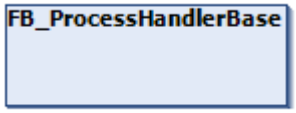
This method must be called cyclically. It performs all the calculations of the simulation.

Syntax:

```
METHOD Cyclic
VAR_INPUT
END_VAR
```

7.4 Pressure handling

7.4.1 FB_ProcessHandlerBase



This function block is used in injection molding machines to switch from injection pressure to holding pressure.

Syntax:

```
FUNCTION_BLOCK FB_ProcessHandlerBase IMPLEMENTS I_ProcessHandler
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
```

Properties

Name	Type	Description
Axis	I_AxisBase	An interface to the axis to be monitored.
BadSwitchOver	BOOL	TRUE if an undesired switchover occurred.
ProcessValue	LREAL	The process value to be monitored.
SwitchOver	BOOL	TRUE if an expected switchover occurred.

Methods

Name	Description
GetControlParameter [▶ 137]	This method can be used to get access to the parameter container of the axis pressure controller.
GetProcessValues [▶ 137]	This method is used to update the FB with a range of actual values (pressure, position, state) of the axis. **This method is called cyclically by the axis. The application is not intended to call this method directly.**
GetSwitchEnable [▶ 137]	This method is used to read back the rules for the switchover.
GetSwitchParameter [▶ 138]	This method may be used to read back parameters of the switchover rules.
SetControlParameter [▶ 138]	This method can be used to assign a parameter container to the axis pressure controller.
SetSwitchEnable [▶ 139]	This method is used to define the rules for the switchover. See below for more information.
SetSwitchParameter [▶ 139]	This method may be used to define parameters of the switchover rules.

If the monitored axis is in an inactive state (Init, Ready, Idle), the FB is inactive.

An active command of the axis also activates the monitoring.

The rules set with SetSwitchEnable determine the response:

If no faulty switchover has been detected yet:

- If bSwitchOnPressure is TRUE and the process value exceeds ePressureThreshold and the axis was moved longer than eGardingTravel from the start point, an expected switchover is detected.
- If bSwitchOnTravel is TRUE and the axis has traveled longer than eTravelThreshold from the start point, an expected switchover is detected.
- If bSwitchOnTime is TRUE and the time since the start of monitoring exceeds the eTimeThreshold, an expected switchover is detected.
- If there is more than one active rule the first match will cause the expected switchover.

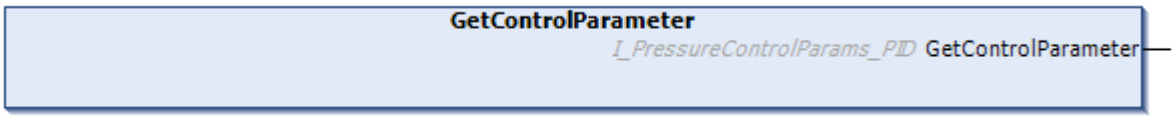
If there was still no switchover:

- If eTimeout is set to more than 0.0 and the time since the start of monitoring exceeds eTimeout, a faulty switchover is detected.
- If the axis position falls below eAlarmPositionLimit, a faulty switchover is detected.
- If eAlarmPressureLimit is set to more than 0.0 and the process value exceeds eAlarmPressureLimit, a faulty switchover is detected.

Also see about this

E_SwitchoverParameter [▶ 139]

7.4.1.1 GetControlParameter



This method can be used to get access to the parameter container of the axis pressure controller.

Syntax:

```
METHOD GetControlParameter : I_PressureControlParams_PID
VAR_INPUT
END_VAR
```

7.4.1.2 GetProcessValue

This method is used to update the FB with a range of actual values (pressure, position, state) of the axis.

This method is called cyclically by the axis. The application is not intended to call this method directly.

Syntax:

```
METHOD GetProcessValues : HRESULT
VAR
END_VAR
```

7.4.1.3 GetSwitchEnable

This method is used to read back the rules for the switchover.

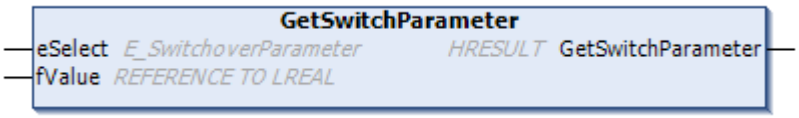
Syntax:

```
METHOD GetSwitchEnable : HRESULT
VAR_INPUT
bSwitchOnPressure: REFERENCE TO BOOL;
bSwitchOnTravel: REFERENCE TO BOOL;
bSwitchOnTime: REFERENCE TO BOOL;
END_VAR
```

Inputs

Name	Type	Description
bSwitchOnPressure	REFERENCE TO BOOL	A reference to the variable to be updated with the parameter.
bSwitchOnTravel	REFERENCE TO BOOL	A reference to the variable to be updated with the parameter.
bSwitchOnTime	REFERENCE TO BOOL	A reference to the variable to be updated with the parameter.

7.4.1.4 GetSwitchParameter



This method may be used to read back parameters of the switchover rules.

Syntax:

```
METHOD SetSwitchParameter : HRESULT
VAR_INPUT
eSelect: E_SwitchoverParameter;
fValue: REFERENCE TO LREAL;
END_VAR
```

Inputs

Name	Type	Description
eSelect	E_SwitchoverParameter [▶ 139]	The selection of the parameter.
fValue	REFERENCE TO LREAL	A reference to the variable to be updated with the parameter.

7.4.1.5 SetControlParameter

This method can be used to assign a parameter container to the axis pressure controller.

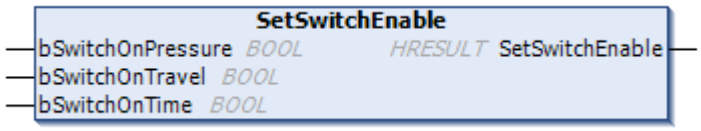
Syntax:

```
METHOD SetControlParameter : HRESULT
VAR_INPUT
iParams: I_PressureControlParams_PID;
END_VAR
```

Inputs

Name	Type	Description
iParams	I_PressureControlParams_PID	An interface to a FB with a parameter set for a pressure controller.

7.4.1.6 SetSwitchEnable



This method is used to define the rules for the switchover. See below for more information.

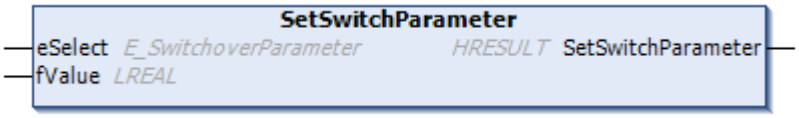
Syntax:

```
METHOD SetSwitchEnable : HRESULT
VAR_INPUT
bSwitchOnPressure: BOOL;
bSwitchOnTravel: BOOL;
bSwitchOnTime: BOOL;
END_VAR
```

Inputs

Name	Type	Description
bSwitchOnPressure	BOOL	A TRUE enables the switchover by exceeding the pressure threshold.
bSwitchOnTravel	BOOL	A TRUE enables the switchover by falling below a position threshold.
bSwitchOnTime	BOOL	A TRUE enables the switchover by reaching a time threshold.

7.4.1.7 SetSwitchParameter



This method may be used to define parameters of the switchover rules.

Syntax:

```
METHOD SetSwitchParameter : HRESULT
VAR_INPUT
eSelect: E_SwitchoverParameter;
fValue: LREAL;
END_VAR
```

Inputs

Name	Type	Description
eSelect	E_SwitchoverParameter [▶ 139]	The selection of the parameter.
fValue	LREAL	The value with which the parameter should be updated.

7.4.2 E_SwitchoverParameter

Values of this enumeration are use by GetSwitchParameter() and SetSwitchParameter() of FB_ProcessHandlerBase [▶ 136].

```
TYPE E_SwitchoverParameter :
(
ePressureThreshold := 0,
eGardingTravel,
eTravelThreshold,
eTimeThreshold,

eAlarmPressureLimit,
eAlarmPositionLimit,
eTimeout
);
END_TYPE
```

7.4.3 FB_ReadProcessValue

A function block of this type is used to supply a function block of type FB_ProcessHandlerBase [▶ 136] with actual values.

Syntax:

```
FUNCTION_BLOCK FB_ReadProcessValue EXTENDS FB_MessageBase IMPLEMENTS I_ReadProcessValue
VAR_INPUT
END_VAR
VAR_OUTPUT
fActualValue: LREAL;
END_VAR
```

Outputs

Name	Type	Description
fActualValue	LREAL	The current actual value.

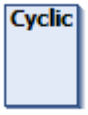
Properties

Name	Type	Access	Description
Offset	LREAL	Get, Set	The offset taken into account when calculating the actual value.
ProcessValue	LREAL	Get	The current process value calculated by the last call to Cyclic().
Weighting	LREAL	Get, Set	The scaling factor.

Methods

Name	Description
Cyclic [▶ 140]	This method must be called by the application once per cycle. It checks the EtherCAT connection and updates the ProcessValue with a filtered actual value.
DoReset [▶ 141]	A call of this method with bExecute=TRUE resets the connected sensor interface and clears all local errors.

7.4.3.1 Cyclic



This method must be called by the application once per cycle. It checks the EtherCAT connection and updates the ProcessValue with a filtered actual value.

Syntax:

```
METHOD Cyclic
VAR_INPUT
END_VAR
```

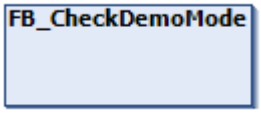
7.4.3.2 DoReset

A call of this method with bExecute=TRUE resets the connected sensor interface and clears all local errors.

Syntax:

```
METHOD DoReset : HRESULT
VAR_INPUT
bExecute: BOOL;
END_VAR
```

7.5 FB_CheckDemoMode



A FB of this type can be instantiated and used by the PLC application to scan the fieldbus configuration. It reports on fieldbuses with special features such as real-time performance or USB support.

Syntax:

FUNCTION_BLOCK FB_CheckDemoMode
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR

Properties

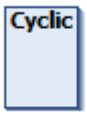
Name	Type	Description
DemoMode	BOOL	A TRUE after scanning the fieldbus configuration has one of the following causes: There is no fieldbus configured with real-time performance. ForceDemoMode=TRUE and ForceNonDemoMode=FALSE.
EtC_detected	BOOL	A TRUE after scanning the fieldbus configuration signals that the configuration provides for an EtherCAT fieldbus.
ForceDemoMode	BOOL	With a TRUE the DemoMode can be forced.
ForceNonDemoMode	BOOL	With a TRUE the NonDemoMode can be forced.
NonDemoMode	BOOL	A TRUE after scanning the fieldbus configuration has one of the following causes: At least one fieldbus with real-time performance is configured. ForceDemoMode=FALSE and ForceNonDemoMode=TRUE.

Name	Type	Description
ReScan	BOOL	A TRUE on this property triggers a rescan of the fieldbus configuration.
RtBus_detected	BOOL	A TRUE after scanning the fieldbus configuration signals that the configuration provides for a fieldbus with real-time performance.
USB_detected	BOOL	A TRUE after scanning the fieldbus configuration signals that the configuration provides for a USB connection. This typically indicates a control panel.

Methods

Name	Description
Cyclic [▶ 142]	This method is to be called cyclically by the application.

7.5.1 Cyclic



This method must be called cyclically by the application.

Syntax:

```
METHOD Cyclic
```

More Information:
www.beckhoff.com/tf8560.html

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