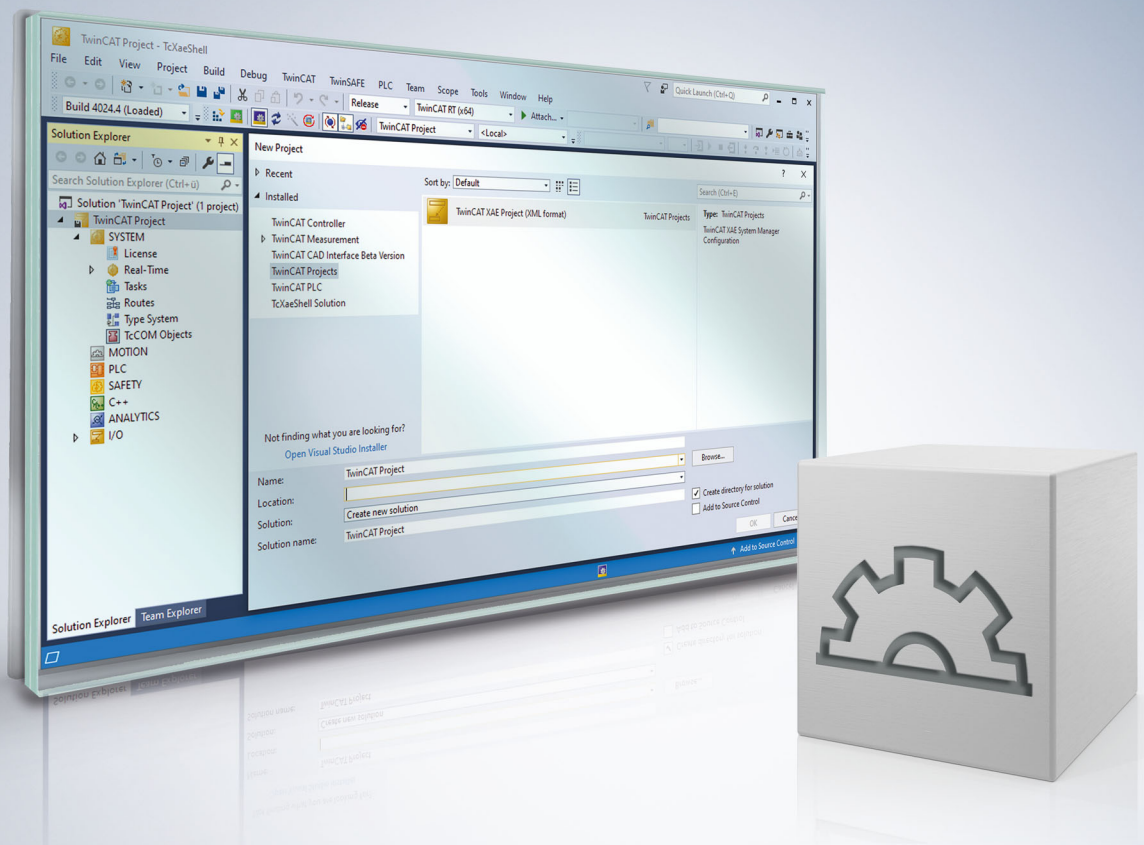
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc3_Physics



2024-03-11 | Version: 1.3.1

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®

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 Data Types

2.1 Enums

2.1.1 E_CoordCategory

Enumeration coordinate categories.

Syntax

Definition:

```
TYPE E_CoordCategory :
(
    Invalid       := 0x0,
    MCS           := 0x1,
    ACS           := 0x2,
    Uninterpreted := 0x3
)UDINT;
END_TYPE
```

Values

Name	Description
Invalid	Invalid
MCS	MCS
ACS	ACS
Uninterpreted	Uninterpreted

2.1.2 E_McsCoordType

Enumeration of MCS coordinate types.

Syntax

Definition:

```
TYPE E_McsCoordType :
(
    Invalid := 0x0,
    X       := 0x11,
    Y       := 0x21,
    Z       := 0x31,
    A1      := 0x41,
    A2      := 0x42,
    A3      := 0x43,
    B1      := 0x51,
    B2      := 0x52,
    B3      := 0x53,
    C1      := 0x61,
    C2      := 0x62,
    C3      := 0x63
)UDINT;
END_TYPE
```

Values

Name	Description
Invalid	Invalid
X	X
Y	Y
Z	Z
A1	A1

Name	Description
A2	A2
A3	A3
B1	B1
B2	B2
B3	B3
C1	C1
C2	C2
C3	C3

2.1.3 E_RotationCoordinates

Enumeration of rotation conventions. The order of letters defines the order of rotation around local axes. This also defines order in which user should program rotation values.

Syntax

Definition:

```
TYPE E_RotationCoordinates :
(
    ABC := 0,
    ACB := 1,
    BCA := 2,
    BAC := 3,
    CAB := 4,
    CBA := 5,
    ABA := 6,
    ACA := 7,
    BAB := 8,
    BCB := 9,
    CAC := 10,
    CBC := 11
)UDINT;
END_TYPE
```

Values

Name	Description
ABC	ABC
ACB	ACB
BCA	BCA
BAC	BAC
CAB	CAB
CBA	CBA
ABA	ABA
ACA	ACA
BAB	BAB
BCB	BCB
CAC	CAC
CBC	CBC

2.2 Structs

2.2.1 CoordinateType

Coordinate type. Use provided constants such as Coord_Mcs_X or factory functions such as GetMcsCoordinateType for creation.

Syntax

Definition:

```
TYPE CoordinateType :
STRUCT
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description

3 Functions

3.1 Coordinates

3.1.1 GetMcsCoordinateType

Creates a CoordinateType FB from E_McsCoordType.

Syntax

Definition:

```
FUNCTION GetMcsCoordinateType : CoordinateType
VAR_INPUT
    mcsType : E_McsCoordType;
END_VAR
```

Inputs

Name	Type	Description
mcsType	E_McsCoordType [▶ 8]	Mcs coordinate type.

Return value

CoordinateType [▶ 9]

Required License

TC3 Physics Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Physics

3.1.2 GetUninterpretedCoordinateType

Creates a CoordinateType FB from an index.

Syntax

Definition:

```
FUNCTION GetUninterpretedCoordinateType : CoordinateType
VAR_INPUT
    mcsType : UDINT;
END_VAR
```

Inputs

Name	Type	Description
mcsType	UDINT	Mcs coordinate type.

Return value

CoordinateType [▶ 9]

Required License

TC3 Physics Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.35 Advanced Motion Pack V3.2.27	PC or CX (x64)	Tc3_Physics

4 Function Blocks

4.1 Dynamics

The IPlcDynamicConstraint interface is accepted by many positioning commands as an optional input to constrain the permitted values for velocity, acceleration, deceleration or jerk during motion. There are different types of constraints, often combinations of different types are accepted, which are combined in a DynamicConstraint_Container [▶ 16]:

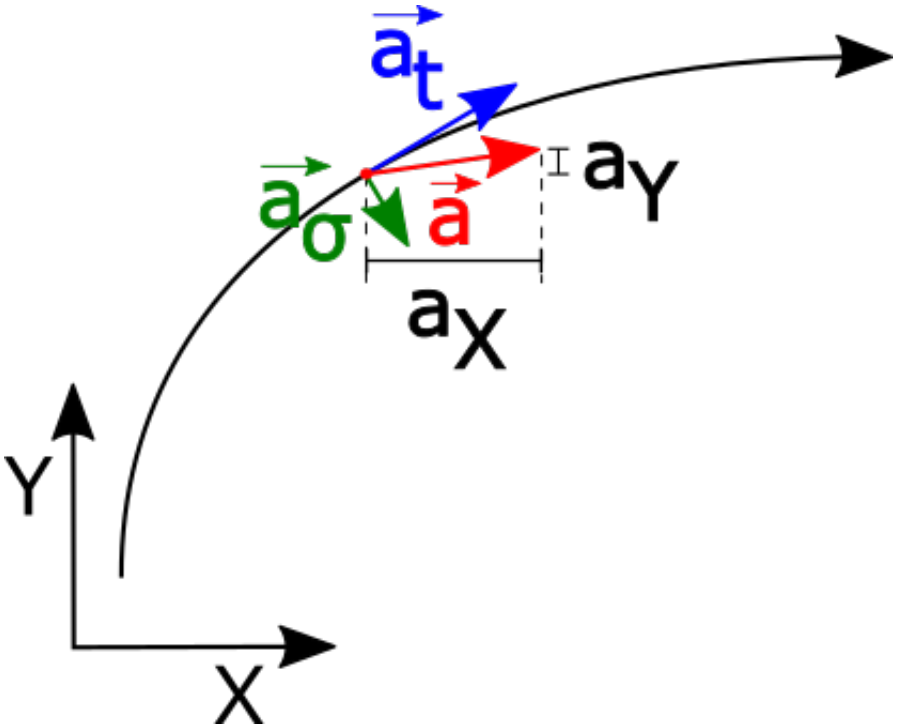
- Constraints of individual coordinates
  - DynamicConstraint_Coordinates [▶ 18]
- Constraints in the direction of travel
  - DynamicConstraint_PathXY [▶ 22]
  - DynamicConstraint_PathXYZ [▶ 23]
- Constraints symmetrical in all directions
  - DynamicConstraint_CartesianXY [▶ 14]
  - DynamicConstraint_CartesianXYZ [▶ 15]

If several constraints act at the same time, all of them will be respected. Therefore, there is no need for the user to calculate which constraint will have the most limiting effect on the dynamics during the motion. Ineffective constraints, for example a constraint to the Z coordinate during a pure XY movement, are ignored.

The following special values are supported:

- MC_DEFAULT: Is replaced by the recipient of the positioning command with the corresponding default value, if available.
- MC_IGNORE: Does not lead to any constraint. This can be useful, for example, if you want to limit acceleration and jerk of a coordinate, but not the velocity.

Example plot for acceleration in 2D



2D velocity (not plotted, in direction of travel):

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

2D acceleration:

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \end{pmatrix} = \vec{a}_{tang} - \vec{a}_{orth}$$

Tangential acceleration in the direction of travel:

$$a_{tang} = \frac{(\vec{v} \cdot \vec{a})}{|\vec{v}|}$$

Possible constraints for acceleration:

DynamicConstraint_PathXY [▶ 22]:

$$a_{tang} \leq A$$

$$-a_{tang} \leq D$$

DynamicConstraint_CartesianXY [▶ 14]:

$$|\vec{a}| \leq A$$

DynamicConstraint_Coordinates [▶ 18]:

$$a_x \leq A_x$$

$$-a_x \leq D_x$$

$$a_y \leq A_y$$

$$-a_y \leq D_y$$

Example PLC

```
PROGRAM MAIN
VAR
    ConstraintPath : DynamicConstraint_PathXY;
    ConstraintCoords : DynamicConstraint_Coordinates;
    ConstraintCombined : DynamicConstraint_Container;
END_VAR
```

```
// Velocity in XY is limited to 1000, the derivative of this velocity with respect to time is
limited to 5000.
// No restriction on the jerk.

    ConstraintPath.SetValuesVADJ(V := 1000, A := 5000, D:= 5000, J := MC_IGNORE);

// Acceleration, deceleration and jerk of the X-coordinate are limited to their default values.

    ConstraintCoords.SetLimit(Coordinate := Coord_Mcs_X, V := MC_IGNORE, A := MC_DEFAULT, D :=
MC_DEFAULT, J := MC_DEFAULT);

// The velocity of the C-coordinate is limited to its default value. Acceleration, deceleration and
jerk are limited to specific values.

    ConstraintCoords.SetLimit(Coordinate := Coord_Mcs_C1, V := MC_DEFAULT, A := 1000, D := 1000,
J := 10000);

// The constraints on path and coordinates are combined into a single object.

    ConstraintCombined.AddConstraint(ConstraintPath);
    ConstraintCombined.AddConstraint(ConstraintCoords);
```

4.1.1 DynamicConstraint_CartesianXY

Dynamic constraint for movement in the XY-plane, including non-tangential effects.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
SetValuesVAJ [▶ 15]	Set the dynamic limits of this instance.

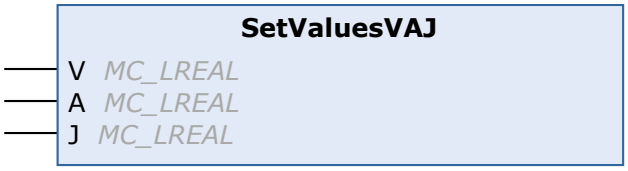
Further information

The function block `DynamicConstraint_CartesianXY` constrains the dynamic values of a movement in the XY plane. V, A and J are the maximum values for velocity (V), acceleration (A) and jerk (J) within the XY plane. Unlike DynamicConstraint_PathXY [▶ 22], these constraints apply symmetrically in all XY directions, not just in the direction of travel.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.35 Advanced Motion Pack V3.2.27	PC or CX (x64)	Tc3_Physics

4.1.1.1 SetValuesVAJ



Set the dynamic limits of this instance.

Syntax

Definition:

```
METHOD SetValuesVAJ
VAR_INPUT
    V : MC_LREAL;
    A : MC_LREAL;
    J : MC_LREAL;
END_VAR
```

Inputs

Name	Type	Description
V	MC_LREAL	Maximum velocity.
A	MC_LREAL	Maximum acceleration.
J	MC_LREAL	Maximum jerk.

4.1.2 DynamicConstraint_CartesianXYZ

Dynamic constraint for movement in the XYZ-space, including non-tangential effects.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
SetValuesVAJ [▶ 16]	Set the dynamic limits of this instance.

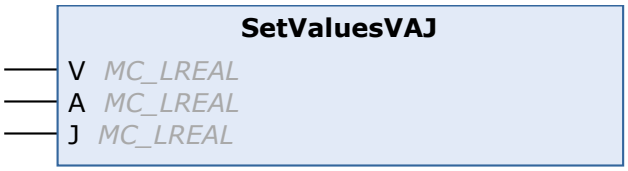
Further information

The DynamicConstraint_CartesianXYZ function block constrains the dynamic values of a motion in XYZ space. V, A and J are the maximum values for velocity (V), acceleration (A) and jerk (J) within the XYZ space, respectively. Unlike DynamicConstraint_PathXYZ [▶ 23], these constraints apply symmetrically in all XYZ directions, not just in the direction of travel.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.35 Advanced Motion Pack V3.2.27	PC or CX (x64)	Tc3_Physics

4.1.2.1 SetValuesVAJ



Set the dynamic limits of this instance.

Syntax

Definition:

```
METHOD SetValuesVAJ
VAR_INPUT
    V : MC_LREAL;
    A : MC_LREAL;
    J : MC_LREAL;
END_VAR
```

Inputs

Name	Type	Description
V	MC_LREAL	Maximum velocity.
A	MC_LREAL	Maximum acceleration.
J	MC_LREAL	Maximum jerk.

4.1.3 DynamicConstraint_Container

A container for dynamic constraints.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Clear [▶ 17]	Removes all dynamic constraints from the container.
AddConstraint [▶ 17]	Adds a copy of a dynamic constraint to the container. If the dynamic constraint changes afterwards, the value in the container will not reflect that change.
AddConstraintByReference [▶ 18]	Adds a reference to a dynamic constraint to the container. If the dynamic constraint changes afterwards, the value in the container will reflect that change.

Further information

The function block `DynamicConstraint_Container` does not define its own constraints. Its purpose is to combine several other dynamics constraints into one object. This may be necessary, for example, if both the dynamic values on the path and the dynamic values of a single coordinate are to be constrained for a movement.

Both copies and references of constraints can be added to an instance of the `DynamicConstraint_Container`:

AddConstraint [▶ 17]

A copy of a dynamics constraint is added to the instance of the `DynamicConstraint_Container`. Changes to the origin constraint do not affect the copy of the constraint in the instance of the `DynamicConstraint_Container`.

AddConstraintByReference [▶ 18]

A reference to a dynamics constraint is added to the instance of the `DynamicConstraint_Container`. Changes in the constraint are taken into account by the instance of `DynamicConstraint_Container`.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.30	PC or CX (x64)	Tc3_Physics

4.1.3.1 Clear

Clear

Removes all dynamic constraints from the container.

Syntax

Definition:

```
METHOD Clear
```

4.1.3.2 AddConstraint

Adds a copy of a dynamic constraint to the container. If the dynamic constraint changes afterwards, the value in the container will not reflect that change.

Syntax

Definition:

```
METHOD AddConstraint
VAR_INPUT
    Constraint : Reference To IPlcDynamicConstraint;
END_VAR
```

Inputs

Name	Type	Description
Constraint	Reference To IPlcDynamicConstraint	A reference to the constraint to be added.

4.1.3.3 AddConstraintByReference

Adds a reference to a dynamic constraint to the container. If the dynamic constraint changes afterwards, the value in the container will reflect that change.

Syntax

Definition:

```
METHOD AddConstraintByReference
VAR_INPUT
    Constraint : Reference To IPlcDynamicConstraint;
END_VAR
```

Inputs

Name	Type	Description
Constraint	Reference To IPlcDynamicConstraint	A reference to the constraint to be added.

4.1.4 DynamicConstraint_Coordinates

Dynamic constraints for individual coordinates.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
AddLimit [▶ 19]	Adds limits for a specific coordinate. If the coordinate is already limited, limits are merged.
SetLimit [▶ 19]	Sets limits for a specific coordinate. If the coordinate is already limited, existing limits are overwritten.
RemoveLimit [▶ 20]	Removes limits for a specific coordinate.
Clear [▶ 20]	Removes limits for all coordinates.
GetV [▶ 20]	Gets the contained velocity limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.
GetA [▶ 21]	Gets the contained acceleration limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.
GetD [▶ 21]	Gets the contained deceleration limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.
GetJ [▶ 22]	Gets the contained jerk limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.

Further information

The DynamicConstraint_Coordinates function block constrains the dynamic values of individual coordinates, for example `Coord_Mcs_X`. Velocity (V), acceleration (A), deceleration (D) and jerk (J) can be limited for each coordinate.

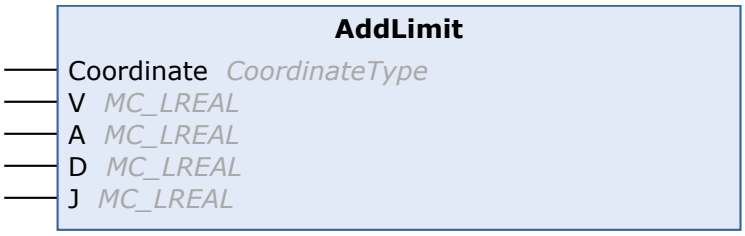
Required License

TC3 Physics Base

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.30	PC or CX (x64)	Tc3_Physics

4.1.4.1 AddLimit



Adds limits for a specific coordinate. If the coordinate is already limited, limits are merged.

Syntax

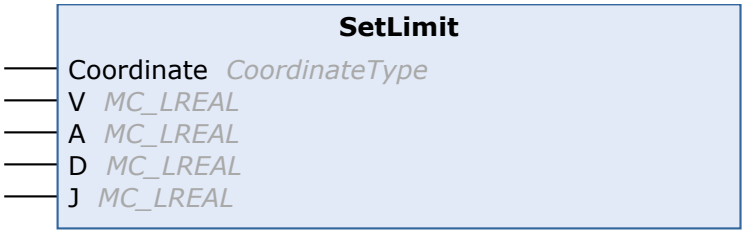
Definition:

```
METHOD AddLimit
VAR_INPUT
    Coordinate : CoordinateType;
    V          : MC_LREAL;
    A          : MC_LREAL;
    D          : MC_LREAL;
    J          : MC_LREAL;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...
V	MC_LREAL	Maximum velocity.
A	MC_LREAL	Maximum acceleration.
D	MC_LREAL	Maximum deceleration.
J	MC_LREAL	Maximum jerk.

4.1.4.2 SetLimit



Sets limits for a specific coordinate. If the coordinate is already limited, existing limits are overwritten.

Syntax

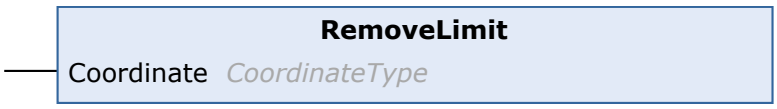
Definition:

```
METHOD SetLimit
VAR_INPUT
    Coordinate : CoordinateType;
    V          : MC_LREAL;
    A          : MC_LREAL;
    D          : MC_LREAL;
    J          : MC_LREAL;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...
V	MC_LREAL	Maximum velocity.
A	MC_LREAL	Maximum acceleration.
D	MC_LREAL	Maximum deceleration.
J	MC_LREAL	Maximum jerk.

4.1.4.3 RemoveLimit



Removes limits for a specific coordinate.

Syntax

Definition:

```
METHOD RemoveLimit
VAR_INPUT
    Coordinate : CoordinateType;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...

4.1.4.4 Clear

Removes limits for all coordinates.

Syntax

Definition:

```
METHOD Clear
```

4.1.4.5 GetV

Gets the contained velocity limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.

Syntax

Definition:

```
METHOD GetV : MC_LREAL
VAR_INPUT
    Coordinate : CoordinateType;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...

Return value

MC_LREAL

4.1.4.6 GetA

Gets the contained acceleration limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.

Syntax

Definition:

```
METHOD GetA : MC_LREAL
VAR_INPUT
    Coordinate : CoordinateType;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...

Return value

MC_LREAL

4.1.4.7 GetD

Gets the contained deceleration limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.

Syntax

Definition:

```
METHOD GetD : MC_LREAL
VAR_INPUT
    Coordinate : CoordinateType;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...

Return value

MC_LREAL

4.1.4.8 GetJ

Gets the contained jerk limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.

Syntax

Definition:

```
METHOD GetJ : MC_LREAL
VAR_INPUT
    Coordinate : CoordinateType;
END_VAR
```

Inputs

Name	Type	Description
Coordinate	CoordinateType [▶ 9]	Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...

Return value

MC_LREAL

4.1.5 DynamicConstraint_Path

DEPRECATED. Please replace with either DynamicConstraint_PathXY or DynamicConstraint_Coordinates, depending on use case.

Do not call the main FB directly. Only use the available methods.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.30	PC or CX (x64)	Tc3_Physics

4.1.6 DynamicConstraint_PathXY

One dimensional dynamic constraint along the XY-components of a path, ignoring non-tangential effects.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
SetValuesVADJ [▶ 23]	Set the dynamic limits of this instance.

Further information

The DynamicConstraint_PathXY function block constrains the tangential dynamic values of a motion in the XY plane. V is the maximum value for the velocity within the XY plane. A, D and J are respectively the maximum values for acceleration (A), deceleration (D) and jerk (J) in the direction of the current XY velocity.

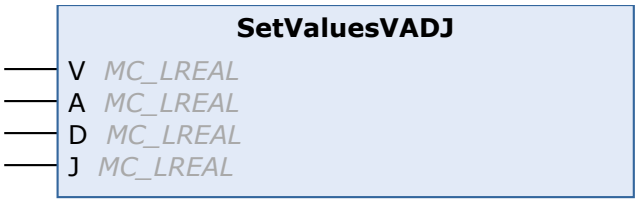
Note that the total acceleration (or jerk) in XY may exceed the tangential acceleration (or jerk) if the path in the XY plane is not a straight line.

If the constraints are to act not only in the direction of travel, but symmetrically in all XY directions, the function block DynamicConstraint_CartesianXY [▶ 14] must be used.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.30	PC or CX (x64)	Tc3_Physics

4.1.6.1 SetValuesVADJ



Set the dynamic limits of this instance.

Syntax

Definition:

```
METHOD SetValuesVADJ
VAR_INPUT
    V : MC_LREAL;
    A : MC_LREAL;
    D : MC_LREAL;
    J : MC_LREAL;
END_VAR
```

Inputs

Name	Type	Description
V	MC_LREAL	Maximum velocity.
A	MC_LREAL	Maximum acceleration.
D	MC_LREAL	Maximum deceleration.
J	MC_LREAL	Maximum jerk.

4.1.7 DynamicConstraint_PathXYZ

One dimensional dynamic constraint along the XYZ-components of a path, ignoring non-tangential effects.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
SetValuesVADJ [▶ 24]	Set the dynamic limits of this instance.

Further information

The DynamicConstraint_PathXYZ function block limits the tangential dynamic values of a motion in the XYZ area. V is the maximum value for the velocity within the XYZ area. A, D and J are the maximum values for acceleration, deceleration, and jerk (respectively) in the direction of the current XYZ speed.

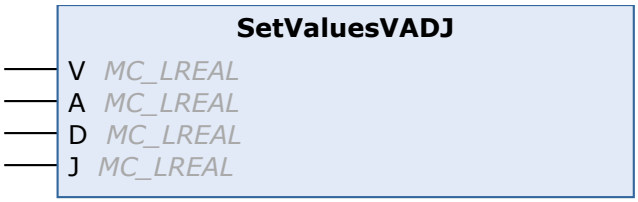
Note that the total acceleration (-jerk) in XYZ may exceed the tangential acceleration (jerk) if the path in XYZ is not a straight line.

If the constraints should apply not only in the direction of travel but also symmetrically in all XYZ directions, the DynamicConstraint_CartesianXYZ [▶ 15] function block must be used.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.35 Advanced Motion Pack V3.2.27	PC or CX (x64)	Tc3_Physics

4.1.7.1 SetValuesVADJ



Set the dynamic limits of this instance.

Syntax

Definition:

```
METHOD SetValuesVADJ
VAR_INPUT
    V : MC_LREAL;
    A : MC_LREAL;
    D : MC_LREAL;
    J : MC_LREAL;
END_VAR
```

Inputs

Name	Type	Description
V	MC_LREAL	Maximum velocity.
A	MC_LREAL	Maximum acceleration.
D	MC_LREAL	Maximum deceleration.
J	MC_LREAL	Maximum jerk.

4.2 Spatial

4.2.1 Positions

4.2.1.1 PositionXY

Position in the 2D Cartesian space.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
SetZero [▶ 25]	Set the coordinates of this position to '0.0'.
SetValues [▶ 26]	Set the coordinates of this position.
SetValuesXY [▶ 26]	Set the xy-coordinates of this position.
SetValuesRP [▶ 27]	Set the components of this position in polar coordinates.
GetRadius [▶ 28]	Get the distance from the origin.
GetPhi [▶ 28]	Get the polar angle scaled in degrees [°].
Invert [▶ 29]	Invert this position.
ConcatenateWith [▶ 29]	Concatenate with a 2nd position.
ShiftByXY [▶ 30]	Displace the components of this position.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Physics

4.2.1.1.1 SetZero

SetZero

Set the coordinates of this position to '0.0'.

Syntax

Definition:

```
METHOD SetZero
```

Example: PositionXY.SetZero

PLC declaration

```
VAR
   position : PositionXY;
END_VAR
```

PLC implementation

```
position.x := 0.3;
position.y := 0.5;
position.SetZero();
```

Expected result

```
position.x = 0
position.y = 0
```

4.2.1.1.2 SetValues

Set the coordinates of this position.

Syntax

Definition:

```
METHOD SetValues
VAR_IN_OUT
    positionXY : PositionXY;
END_VAR
```

In/Outputs

Name	Type	Description
positionXY	PositionXY [▶ 25]	The coordinate values to set.

Example: PositionXY.SetValues

PLC declaration

```
VAR
   position0 : PositionXY;
   position1 : PositionXY;
END_VAR
```

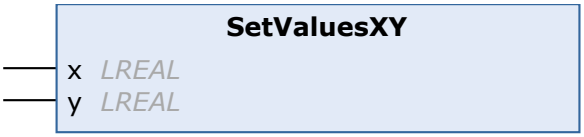
PLC implementation

```
position0.x := 5.6;
position0.y := 0.2;
position1.SetValues(position0);
```

Expected result

```
position1.x = 5.6
position1.y = 0.2
```

4.2.1.1.3 SetValuesXY



Set the xy-coordinates of this position.

Syntax

Definition:

```
METHOD SetValuesXY
VAR_INPUT
    x : LREAL;
    y : LREAL;
END_VAR
```

Inputs

Name	Type	Description
x	LREAL	x-component of the position.
y	LREAL	y-component of the position.

Example: PositionXY.SetValuesXY

PLC declaration

```
VAR
   position  : PositionXY;
   x : LREAL := 4.3;
   y : LREAL := 2.5;
END_VAR
```

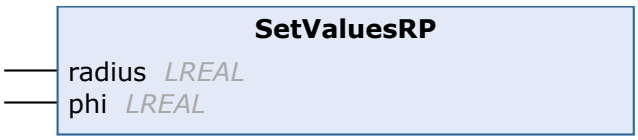
PLC implementation

```
position.x := 0;
position.y := 0;
position.SetValuesXY(x, y);
```

Expected result

```
position.x = 4.3
position.y = 2.5
```

4.2.1.1.4 SetValuesRP



Set the components of this position in polar coordinates.

Syntax

Definition:

```
METHOD SetValuesRP
VAR_INPUT
    radius : LREAL;
    phi    : LREAL;
END_VAR
```

Inputs

Name	Type	Description
radius	LREAL	Distance from the origin.
phi	LREAL	Polar angle of this position in degrees [°].

Example: PositionXY.SetValuesRP

PLC declaration

```
VAR
   position  : PositionXY;
   r : LREAL := 1;
   p : LREAL := 15;
END_VAR
```

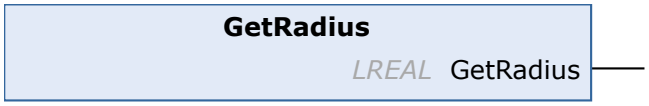
PLC implementation

```
position.x := 0;
position.y := 0;
position.SetValuesRP(r, p);
```

Expected result

```
position.x = 0.966
position.y = 0.259
```

4.2.1.1.5 GetRadius



Get the distance from the origin.

Syntax

Definition:

```
METHOD GetRadius : LREAL
```

Return value

LREAL

Example: PositionXY.GetRadius

PLC declaration

```
VAR
   position : PositionXY;
   distance : LREAL;
END_VAR
```

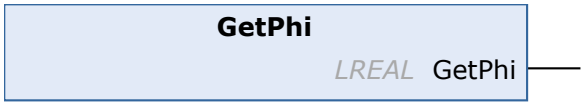
PLC implementation

```
position.SetValues(5.6, 0.2);
distance := position.GetRadius();
```

Expected result

```
distance = 5.604
```

4.2.1.1.6 GetPhi



Get the polar angle scaled in degrees [°].

Syntax

Definition:

```
METHOD GetPhi : LREAL
```

Return value

LREAL

Example: PositionXY.GetPhi

PLC declaration

```
VAR
   position   : PositionXY;
   polarAngle : LREAL;
END_VAR
```

PLC implementation

```
position.SetValues(5.6, 0.2);
polarAngle := position.GetPhi();
```

Expected result

```
polarAngle = 2.045
```

4.2.1.1.7 Invert

Invert

Invert this position.

Syntax

Definition:

```
METHOD Invert
```

Example: PositionXY.Invert

PLC declaration

```
VAR
   position : PositionXY;
END_VAR
```

PLC implementation

```
position.SetValuesXY(1, 3);
position.Invert();
```

Expected result

```
position.x = -1
position.y = -3
```

4.2.1.1.8 ConcatenateWith

ConcatenateWith

↔ position *PositionXY*

Concatenate with a 2nd position.

Syntax

Definition:

```
METHOD ConcatenateWith
VAR_IN_OUT
    position : PositionXY;
END_VAR
```

In/Outputs

Name	Type	Description
position	PositionXY [▶ 25]	The 2nd position to concatenate with.

Example: PositionXY.ConcatenateWith

PLC declaration

```
VAR
   position0 : PositionXY;
   position1 : PositionXY;
END_VAR
```

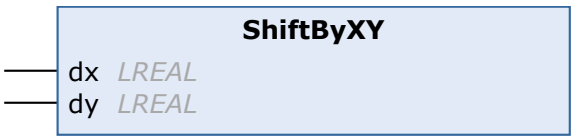
PLC implementation

```
position0.SetValuesXY(0.5, 1.2);
position1.SetValuesXY(0.3, 0.5);
position0.ConcatenateWith(position1);
```

Expected result

```
position0.x = 0.8
position0.y = 1.7
```

4.2.1.1.9 ShiftByXY



Displace the components of this position.

Syntax

Definition:

```
METHOD ShiftByXY
VAR_INPUT
    dx : LREAL;
    dy : LREAL;
END_VAR
```

Inputs

Name	Type	Description
dx	LREAL	Shift of the x-component of this position.
dy	LREAL	Shift of the y-component of this position.

Example: PositionXY.ShiftByXY

PLC declaration

```
VAR
   position   : PositionXY;
   dx : LREAL := 2.0;
   dy : LREAL := 0.0;
END_VAR
```

PLC implementation

```
position.SetValuesXY(0, 1);
position.ShiftByXY(dx, dy);
```

Expected result

```
position.x = 2
position.y = 1
```

4.2.1.2 PositionXYC

A position in the xy-plane with an in-plane direction c.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
SetZero [▶ 31]	Set the components of this vector to '0.0'.
SetValues [▶ 31]	Set the components of this position.
SetValuesXY [▶ 32]	Set the spatial components of this position.
SetValuesXYC [▶ 33]	Set the components of this position.
Invert [▶ 33]	Invert this position.

Name	Description
ConcatenateWith [▶ 34]	Multiply this position by a 2nd from the right.
StepByLocalXY [▶ 34]	Apply a local step in respect to the local coordinate frame without changing the orientation.
ShiftByXY [▶ 35]	Displace the components of this position.
TurnByC [▶ 36]	Rotate the attached orientation without changing the Cartesian coodinates.
Transform [▶ 36]	Place a PositionXY into the coordinate system referenced by this position.

System Requirements

Development environment	Target system type	PLC libraries to include
TwinCAT V3.1.4024.12 Advanced Motion Pack V3.1.10.11	PC or CX (x64)	Tc3_Physics

4.2.1.2.1 SetZero

Set the components of this vector to '0.0'.

Syntax

Definition:

```
METHOD SetZero
```

Example: PositionXYC.SetZero

PLC declaration

```
VAR
   position : PositionXYC;
END_VAR
```

PLC implementation

```
position.x := 0.3;
position.y := 0.5;
position.c := 20.0;
position.SetZero();
```

Expected result

```
position.x = 0
position.y = 0
position.c = 0
```

4.2.1.2.2 SetValues

Set the components of this position.

Syntax

Definition:

```
METHOD SetValues
VAR_IN_OUT
    positionXYC : PositionXYC;
END_VAR
```

In/Outputs

Name	Type	Description
positionXYC	PositionXYC [▶ 30]	The values to set.

Example: PositionXYC.SetValues

PLC declaration

```
VAR
   position0 : PositionXYC;
   position1 : PositionXYC;
END_VAR
```

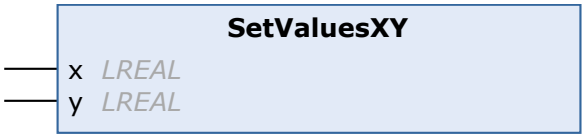
PLC implementation

```
position0.x := 5.6;
position0.y := 0.2;
position0.c := 4.2;
position1.SetValues(position0);
```

Expected result

```
position1.x = 5.6
position1.y = 0.2
position1.c = 4.2
```

4.2.1.2.3 SetValuesXY



Set the spatial components of this position.

Syntax

Definition:

```
METHOD SetValuesXY
VAR_INPUT
    x : LREAL;
    y : LREAL;
END_VAR
```

Inputs

Name	Type	Description
x	LREAL	x-component of the position.
y	LREAL	y-component of the position.

Example: PositionXYC.SetValuesXY

PLC declaration

```
VAR
   position  : PositionXYC;
   x : LREAL := 4.3;
   y : LREAL := 2.5;
END_VAR
```

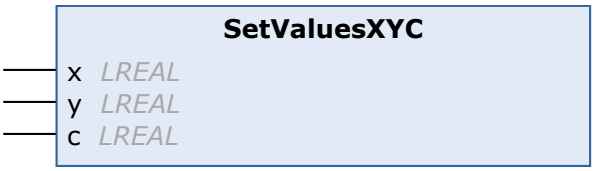
PLC implementation

```
position.x := 0;
position.y := 0;
position.c := 3;
position.SetValuesXY(x, y);
```

Expected result

```
position.x = 4.3
position.y = 2.5
position.c = 3
```

4.2.1.2.4 SetValuesXYC



Set the components of this position.

Syntax

Definition:

```
METHOD SetValuesXYC
VAR_INPUT
    x : LREAL;
    y : LREAL;
    c : LREAL;
END_VAR
```

Inputs

Name	Type	Description
x	LREAL	x-component of the position.
y	LREAL	y-component of the position.
c	LREAL	c-component of the position scaled in degrees [°].

Example: PositionXYC.SetValuesXYC

PLC declaration

```
VAR
   position  : PositionXYC;
   x : LREAL := 4.3;
   y : LREAL := 2.5;
   c : LREAL := 20;
END_VAR
```

PLC implementation

```
position.x := 0;
position.y := 0;
position.c := 0;
position.SetValuesXYC(x, y, c);
```

Expected result

```
position.x = 4.3
position.y = 2.5
position.c = 20
```

4.2.1.2.5 Invert

Invert this position.

Syntax

Definition:

```
METHOD Invert
```

Example: PositionXYC.Invert

PLC declaration

```
VAR
   position : PositionXYC;
END_VAR
```

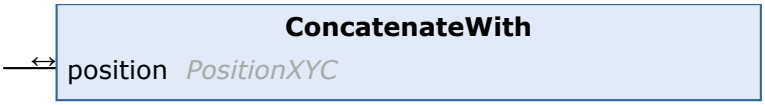
PLC implementation

```
position.SetValuesXYC(1, 3, 10);
position.Invert();
```

Expected result

```
position.x = -1
position.y = -3
position.c = -10
```

4.2.1.2.6 ConcatenateWith



Multiply this position by a 2nd from the right.

Syntax

Definition:

```
METHOD ConcatenateWith
VAR_IN_OUT
    position : PositionXYC;
END_VAR
```

In/Outputs

Name	Type	Description
position	PositionXYC [▶ 30]	The position to concatenate with.

Example: PositionXYC.ConcatenateWith

PLC declaration

```
VAR
   position0 : PositionXYC;
   position1 : PositionXYC;
END_VAR
```

PLC implementation

```
position0.SetValuesXYC(0.5, 1.2, 10);
position1.SetValuesXYC(0.3, 0.5, 15);
position0.ConcatenateWith(position1);
```

Expected result

```
position0.x = 0.7086
position0.y = 1.7450
position0.c = 25
```

4.2.1.2.7 StepByLocalXY

Apply a local step in respect to the local coordinate frame without changing the orientation.

Syntax

Definition:

```
METHOD StepByLocalXY
VAR_INPUT
    dx : LREAL;
    dy : LREAL;
END_VAR
```

Inputs

Name	Type	Description
dx	LREAL	The displacement within the local x-direction.
dy	LREAL	The displacement within the local y-direction.

Example: PositionXYC.StepByLocalXY

PLC declaration

```
VAR
   position   : PositionXYC;
   dx : LREAL := 2.0;
   dy : LREAL := 0.0;
END_VAR
```

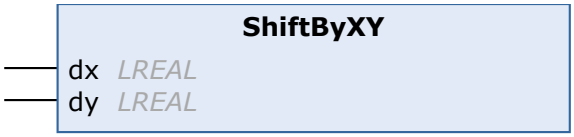
PLC implementation

```
position.SetValuesXYC(0, 3, 10);
position.StepByLocalXY(dx, dy);
```

Expected result

```
position.x = 1.9696
position.y = 3.3473
position.c = 10
```

4.2.1.2.8 ShiftByXY



Displace the components of this position.

Syntax

Definition:

```
METHOD ShiftByXY
VAR_INPUT
    dx : LREAL;
    dy : LREAL;
END_VAR
```

Inputs

Name	Type	Description
dx	LREAL	Shift of the x-component of this position.
dy	LREAL	Shift of the y-component of this position.

Example: PositionXYC.ShiftByXY

PLC declaration

```
VAR
   position   : PositionXYC;
   dx : LREAL := 2.0;
   dy : LREAL := 0.0;
END_VAR
```

PLC implementation

```
position.SetValuesXYC(0, 3, 10);
position.ShiftByXY(dx, dy);
```

Expected result

```
position.x = 2
position.y = 3
position.c = 10
```

4.2.1.2.9 TurnByC

Rotate the attached orientation without changing the Cartesian coodinates.

Syntax

Definition:

```
METHOD TurnByC
VAR_INPUT
    dc : LREAL;
END_VAR
```

Inputs

Name	Type	Description
dc	LREAL	Angular displacement given in degrees [°].

Example: PositionXYC.TurnByC

PLC declaration

```
VAR
   position   : PositionXYC;
   angle      : LREAL := -5;
END_VAR
```

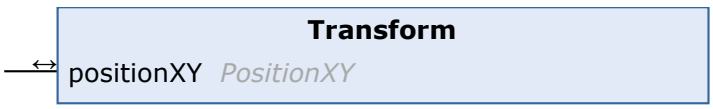
PLC implementation

```
position.SetValuesXYC(1.5, 2.5, 20);
position.TurnByC(angle);
```

Expected result

```
position.x = 1.5
position.y = 2.5
position.c = 15
```

4.2.1.2.10 Transform



Place a PositionXY into the coordinate system referenced by this position.

Syntax

Definition:

```
METHOD Transform
VAR_IN_OUT
    positionXY : PositionXY;
END_VAR
```

In/Outputs

Name	Type	Description
positionXY	PositionXY [▶ 25]	Position to transform.

Example: PositionXYC.Transform

PLC declaration

```
VAR
   position0 : PositionXYC;
   position1 : PositionXY;
END_VAR
```

PLC implementation

```
position0.SetValuesXYC(1, 2, 20);
position1.SetValuesXY(0, 2);
position0.Transform(position1);
```

Expected result

```
position1.x = 0.3160
position1.y = 3.8794
```

5 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

More Information:
www.beckhoff.com/te1000

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