# Table of contents

1 Foreword .................................................................................................................................................. 5
   1.1 Notes on the documentation ........................................................................................................ 5
   1.2 Safety instructions ....................................................................................................................... 6

2 Dynamic constraints ................................................................................................................................ 7
   2.1 Example ........................................................................................................................................ 8

3 Function Blocks ....................................................................................................................................... 9
   3.1 Dynamics ....................................................................................................................................... 9
      3.1.1 DynamicConstraint_CONTAINER ....................................................................................... 9
      3.1.2 DynamicConstraint_Coordinates ....................................................................................... 10
      3.1.3 DynamicConstraint_Path ..................................................................................................... 14
      3.1.4 DynamicConstraint_PathXY ............................................................................................... 14
   3.2 Spatial ........................................................................................................................................... 15
      3.2.1 Positions .............................................................................................................................. 15

4 Functions ............................................................................................................................................... 29
   4.1 Coordinates ..................................................................................................................................... 29
      4.1.1 GetMcsCoordinateType ........................................................................................................ 29

5 Data Types ............................................................................................................................................... 30
   5.1 Enums .......................................................................................................................................... 30
      5.1.1 E_CoordCategory ................................................................................................................. 30
      5.1.2 E_McsCoordType .................................................................................................................. 30
      5.1.3 E_RotationCoordinates ......................................................................................................... 31
   5.2 Structs .......................................................................................................................................... 32
      5.2.1 CoordinateType ..................................................................................................................... 32
1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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

Disclaimer

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

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
with corresponding applications or registrations in various other countries.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

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

Personnel qualification

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

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="DANGER" /></td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="image" alt="WARNING" /></td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="image" alt="CAUTION" /></td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><img src="image" alt="NOTE" /></td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer

This symbol indicates information that contributes to better understanding.
Dynamic constraints

The IPicDynamicConstraint 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. In many cases combinations of different types are accepted.

If several constraints are in effect at the same time, all are adhered to. Therefore, there is no need for the user to calculate which constraint will have the most limiting effect on the dynamics during the motion. Constraints that are not applicable, for example a constraint on the z-coordinate during a movement that only takes place in the xy-plane, 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 if, for example, you want to limit the acceleration and jerk of a coordinate, but not the velocity.

**DynamicConstraint_Container**

The FB 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.

**DynamicConstraint_PathXY**

The FB DynamicConstraint_PathXY constrains the tangential dynamic values of a movement in the xy-plane. V is the maximum value for the velocity within the xy-plane. A, D and J are the maximum values for acceleration, deceleration and jerk 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.

**DynamicConstraint_Coordinates**

The FB DynamicConstraint_Coordinates 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.
2.1 Example

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 := MCIGNORE, 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);
3 Function Blocks

3.1 Dynamics

3.1.1 DynamicConstraint_Container

A container for dynamic constraints.
Do not call the main FB directly. Only use the available methods.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>Removes all dynamic constraints from the container.</td>
</tr>
<tr>
<td>AddConstraint</td>
<td>Adds a dynamic constraint to the container.</td>
</tr>
</tbody>
</table>

Required License

TC3 Physics Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.1.1 Clear

Clear

Removes all dynamic constraints from the container.

Syntax

Definition:

METHOD Clear

3.1.1.2 AddConstraint

AddConstraint Constraint : IPlcDynamicConstraint

Adds a dynamic constraint to the container.

Syntax

Definition:

METHOD AddConstraint
VAR_INPUT
  Constraint : IPlcDynamicConstraint;
END_VAR
Function Blocks

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint</td>
<td>IPlcDynamicConstraint</td>
<td>A reference to the constraint to be added.</td>
</tr>
</tbody>
</table>

3.1.2 DynamicConstraint_Coordinates

Dynamic constraints for individual coordinates.

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

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddLimit [10]</td>
<td>Adds limits for a specific coordinate. If the coordinate is already limited, limits are merged.</td>
</tr>
<tr>
<td>SetLimit [11]</td>
<td>Sets limits for a specific coordinate. If the coordinate is already limited, existing limits are overwritten.</td>
</tr>
<tr>
<td>GetV [12]</td>
<td>Gets the contained velocity limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.</td>
</tr>
<tr>
<td>GetA [12]</td>
<td>Gets the contained acceleration limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.</td>
</tr>
<tr>
<td>GetD [13]</td>
<td>Gets the contained deceleration limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.</td>
</tr>
<tr>
<td>GetJ [13]</td>
<td>Gets the contained jerk limit for a specific coordinate. If no limits for the coordinate are set, returns MC_INVALID.</td>
</tr>
</tbody>
</table>

Required License

TC3 Physics Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.2.1 AddLimit

AddLimit

Coordinate CoordinateType

V MC_LREAL
A MC_LREAL
D MC_LREAL
J MC_LREAL

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...</td>
</tr>
<tr>
<td>V</td>
<td>MC_LREAL</td>
<td>Maximum velocity.</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
<td>Maximum acceleration.</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
<td>Maximum deceleration.</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
<td>Maximum jerk.</td>
</tr>
</tbody>
</table>

3.1.2.2 SetLimit

SetLimit

<table>
<thead>
<tr>
<th>Coordinate</th>
<th>CoordinateType</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>MC_LREAL</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...</td>
</tr>
<tr>
<td>V</td>
<td>MC_LREAL</td>
<td>Maximum velocity.</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
<td>Maximum acceleration.</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
<td>Maximum deceleration.</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
<td>Maximum jerk.</td>
</tr>
</tbody>
</table>

3.1.2.3 RemoveLimit

RemoveLimit

| Coordinate | CoordinateType |

Removes limits for a specific coordinate.
Function Blocks

Syntax

Definition:
METHOD RemoveLimit
VAR_INPUT
  Coordinate : CoordinateType;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType [32]</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y,...</td>
</tr>
</tbody>
</table>

3.1.2.4 Clear

Clear

Removes limits for all coordinates.

Syntax

Definition:
METHOD Clear

3.1.2.5 GetV

GetV

Coordinate CoordinateType MC_LREAL 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType [32]</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y,...</td>
</tr>
</tbody>
</table>

Return value

MC_LREAL

3.1.2.6 GetA

GetA

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

**Syntax**

**Definition:**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType [32]</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...</td>
</tr>
</tbody>
</table>

**Return value**

MC_LREAL

### 3.1.2.7 GetD

```
GetD
Coordinate CoordinateType MC_LREAL GetD
```

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

**Syntax**

**Definition:**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType [32]</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...</td>
</tr>
</tbody>
</table>

**Return value**

MC_LREAL

### 3.1.2.8 GetJ

```
GetJ
Coordinate CoordinateType MC_LREAL 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordinate</td>
<td>CoordinateType ] 32</td>
<td>Coordinate type. E.g. Coord_Mcs_X, Coord_Mcs_Y, ...</td>
</tr>
</tbody>
</table>

Return value

MC_LREAL

3.1.3 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.

Required License

TC3 Physics Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.4 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetValuesVADJ [15]</td>
<td>Set the dynamic limits of this instance.</td>
</tr>
</tbody>
</table>

Required License

TC3 Physics Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.1.4.1 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>MC_LREAL</td>
<td>Maximum velocity.</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
<td>Maximum acceleration.</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
<td>Maximum deceleration.</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
<td>Maximum jerk.</td>
</tr>
</tbody>
</table>

3.2 Spatial

3.2.1 Positions

3.2.1.1 PositionXY

Position in the 2D Cartesian space.

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

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetZero [16]</td>
<td>Set the coordinates of this position to '0.0'.</td>
</tr>
<tr>
<td>SetValues [16]</td>
<td>Set the coordinates of this position.</td>
</tr>
<tr>
<td>SetValuesXY [17]</td>
<td>Set the xy-coordinates of this position.</td>
</tr>
<tr>
<td>SetValuesRP [18]</td>
<td>Set the components of this position in polar coordinates.</td>
</tr>
<tr>
<td>GetRadius [18]</td>
<td>Get the distance from the origin.</td>
</tr>
<tr>
<td>GetPhi [19]</td>
<td>Get the polar angle scaled in degrees [°].</td>
</tr>
<tr>
<td>Invert [19]</td>
<td>Invert this position.</td>
</tr>
<tr>
<td>ConcatenateWith [20]</td>
<td>Concatenate with a 2nd position.</td>
</tr>
<tr>
<td>ShiftByXY [20]</td>
<td>Displace the components of this position.</td>
</tr>
</tbody>
</table>
www.beckhoff.com

Function Blocks

Required License
TC3 Physics Base

System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2.1.1.1 SetZero

Set the coordinates of this position to '0.0'.

Syntax

Definition:
METHOD SetZero

Example: PositionXY.SetZero

PLC declaration

VAR
  position : PositionXY;
VAR_END

PLC implementation

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

Expected result

position.x = 0
position.y = 0

3.2.1.1.2 SetValues

Set the coordinates of this position.

Syntax

Definition:
METHOD SetValues
VAR_IN_OUT
  positionXY : PositionXY;
END_VAR

In/Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
Example: PositionXY.SetValues

**PLC declaration**

```plaintext
VAR
    position0 : PositionXY;
    position1 : PositionXY;
VAR_END
```

**PLC implementation**

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

**Expected result**

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

3.2.1.1.3 **SetValuesXY**

Set the xy-coordinates of this position.

**Syntax**

**Definition:**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>LREAL</td>
<td>x-component of the position.</td>
</tr>
<tr>
<td>y</td>
<td>LREAL</td>
<td>y-component of the position.</td>
</tr>
</tbody>
</table>

**Example: PositionXY.SetValuesXY**

**PLC declaration**

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

**PLC implementation**

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

**Expected result**

```plaintext
position.x = 4.3
position.y = 2.5
```
3.2.1.1.4 SetValuesRP

Set the components of this position in polar coordinates.

**Syntax**

**Definition:**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>radius</td>
<td>LREAL</td>
<td>Distance from the origin.</td>
</tr>
<tr>
<td>phi</td>
<td>LREAL</td>
<td>Polar angle of this position in degrees [°].</td>
</tr>
</tbody>
</table>

**Example: PositionXY.SetValuesRP**

**PLC declaration**

VAR
  position : PositionXY;
  r : LREAL := 1;
  p : LREAL := 15;
VAR_END

**PLC implementation**

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

**Expected result**

position.x = 0.966
position.y = 0.259

3.2.1.1.5 GetRadius

Get the distance from the origin.

**Syntax**

**Definition:**

METHOD GetRadius : LREAL

**Return value**

LREAL
Example: PositionXY.GetRadius

PLC declaration

```plaintext
VAR
    position : PositionXY;
    distance : LREAL;
VAR_END
```

PLC implementation

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

Expected result

```plaintext
distance = 5.604
```

3.2.1.1.6 GetPhi

Get the polar angle scaled in degrees [°].

**Syntax**

Definition:

```plaintext
METHOD GetPhi : LREAL
```

Return value

LREAL

**Example: PositionXY.GetPhi**

PLC declaration

```plaintext
VAR
    position : PositionXY;
    polarAngle : LREAL;
VAR_END
```

PLC implementation

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

Expected result

```plaintext
polarAngle = 2.045
```

3.2.1.1.7 Invert

Invert this position.

**Syntax**

Definition:

```plaintext
METHOD Invert
```
Example: PositionXY.Invert

PLC declaration

```
VAR
    position : PositionXY;
VAR_END
```

PLC implementation

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

Expected result

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

3.2.1.1.8 ConcatenateWith

Concatenate with a 2nd position.

Syntax

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

In/Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>PositionXY</td>
<td>The 2nd position to concatenate with.</td>
</tr>
</tbody>
</table>

Example: PositionXY.ConcatenateWith

PLC declaration

```
VAR
    position0 : PositionXY;
    position1 : PositionXY;
VAR_END
```

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
```

3.2.1.1.9 ShiftByXY

Displace the components of this position.
Syntax

Definition:

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

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dx</td>
<td>LREAL</td>
<td>Shift of the x-component of this position.</td>
</tr>
<tr>
<td>dy</td>
<td>LREAL</td>
<td>Shift of the y-component of this position.</td>
</tr>
</tbody>
</table>

Example: PositionXY.ShiftByXY

PLC declaration

VAR
    position : PositionXY;
    dx : LREAL := 2.0;
    dy : LREAL := 0.0;
VAR_END

PLC implementation

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

Expected result

position.x = 2
position.y = 1

3.2.1.2 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetZero</td>
<td>Set the components of this vector to '0.0'.</td>
</tr>
<tr>
<td>SetValues</td>
<td>Set the components of this position.</td>
</tr>
<tr>
<td>SetValuesXY</td>
<td>Set the spatial components of this position.</td>
</tr>
<tr>
<td>SetValuesXYC</td>
<td>Set the components of this position.</td>
</tr>
<tr>
<td>Invert</td>
<td>Invert this position.</td>
</tr>
<tr>
<td>ConcatenateWith</td>
<td>Multiply this position by a 2nd from the right.</td>
</tr>
<tr>
<td>StepByLocalXY</td>
<td>Apply a local step in respect to the local coordinate frame without changing the orientation.</td>
</tr>
<tr>
<td>ShiftByXY</td>
<td>Displace the components of this position.</td>
</tr>
<tr>
<td>TurnByC</td>
<td>Rotate the attached orientation without changing the Cartesian coordinates.</td>
</tr>
<tr>
<td>Transform</td>
<td>Place a PositionXY into the coordinate system referenced by this position.</td>
</tr>
</tbody>
</table>

Required License

TC3 Physics Base
System Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2.1.2.1 SetZero

Set the components of this vector to '0.0'.

Syntax

Definition:
METHOD SetZero

Example: PositionXYC.SetZero

PLC declaration

VAR
  position : PositionXYC;
VAR_END

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

3.2.1.2.2 SetValues

Set the components of this position.

Syntax

Definition:
METHOD SetValues
VAR_IN_OUT
  positionXYC : PositionXYC;
END_VAR

In/Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>positionXYC</td>
<td>PositionXYC</td>
<td>The values to set.</td>
</tr>
</tbody>
</table>
Example: PositionXYC.SetValues

PLC declaration

```plaintext
VAR
    position0 : PositionXYC;
    position1 : PositionXYC;
VAR_END
```

PLC implementation

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

Expected result

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

3.2.1.2.3 SetValuesXY

SetValuesXY

```plaintext
x  LREAL  
|---
|   y  LREAL
```

Set the spatial components of this position.

Syntax

Definition:

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

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>LREAL</td>
<td>x-component of the position.</td>
</tr>
<tr>
<td>y</td>
<td>LREAL</td>
<td>y-component of the position.</td>
</tr>
</tbody>
</table>

Example: PositionXYC.SetValuesXY

PLC declaration

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

PLC implementation

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

Expected result

```plaintext
position.x = 4.3
position.y = 2.5
position.c = 3
```
### 3.2.1.2.4 SetValuesXYC

SetValuesXYC

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>LREAL</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>LREAL</td>
<td></td>
</tr>
</tbody>
</table>

Set the components of this position.

**Syntax**

**Definition:**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>LREAL</td>
<td>x-component of the position.</td>
</tr>
<tr>
<td>y</td>
<td>LREAL</td>
<td>y-component of the position.</td>
</tr>
<tr>
<td>c</td>
<td>LREAL</td>
<td>c-component of the position scaled in degrees [°].</td>
</tr>
</tbody>
</table>

**Example: PositionXYC.SetValuesXYC**

**PLC declaration**

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

**PLC implementation**

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

**Expected result**

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

### 3.2.1.2.5 Invert

**Syntax**

**Definition:**

```plaintext
METHOD Invert
```
Example: PositionXYC.Invert

PLC declaration

```
VAR
  position : PositionXYC;
VAR_END
```

PLC implementation

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

Expected result

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

3.2.1.2.6 ConcatenateWith

```
ConcatenateWith
  position  : PositionXYC
```

Multiply this position by a 2nd from the right.

Syntax

Definition:

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

In/Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>PositionXYC</td>
<td>The position to concatenate with.</td>
</tr>
</tbody>
</table>

Example: PositionXYC.ConcatenateWith

PLC declaration

```
VAR
  position0 : PositionXYC;
  position1 : PositionXYC;
VAR_END
```

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
```

3.2.1.2.7 StepByLocalXY

```
StepByLocalXY
  dx  : LREAL
  dy  : LREAL
```

Function Blocks

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dx</td>
<td>LREAL</td>
<td>The displacement within the local x-direction.</td>
</tr>
<tr>
<td>dy</td>
<td>LREAL</td>
<td>The displacement within the local y-direction.</td>
</tr>
</tbody>
</table>

Example: PositionXYC.StepByLocalXY

PLC declaration

VAR
   position : PositionXYC;
   dx : LREAL := 2.0;
   dy : LREAL := 0.0;
VAR_END

PLC implementation

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

Expected result

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

3.2.1.2.8 ShiftByXY

Syntax

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

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dx</td>
<td>LREAL</td>
<td>Shift of the x-component of this position.</td>
</tr>
<tr>
<td>dy</td>
<td>LREAL</td>
<td>Shift of the y-component of this position.</td>
</tr>
</tbody>
</table>
Example: PositionXYC.ShiftByXY

**PLC declaration**

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

**PLC implementation**

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

**Expected result**

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

### 3.2.1.2.9 TurnByC

**Syntax**

**Definition:**

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

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc</td>
<td>LREAL</td>
<td>Angular displacement given in degrees [°].</td>
</tr>
</tbody>
</table>

**Example: PositionXYC.TurnByC**

**PLC declaration**

```plaintext
VAR
    position : PositionXYC;
    angle : LREAL := -5;
VAR_END
```

**PLC implementation**

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

**Expected result**

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

### 3.2.1.2.10 Transform

```plaintext
positionXY : PositionXY
```
Place a PositionXY into the coordinate system referenced by this position.

**Syntax**

**Definition:**

METHOD Transform
VAR_IN_OUT
    positionXY : PositionXY;
END_VAR

**In/Outputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>positionXY</td>
<td>PositionXY</td>
<td>Position to transform.</td>
</tr>
</tbody>
</table>

**Example: PositionXYC.Transform**

**PLC declaration**

VAR
    position0 : PositionXYC;
    position1 : PositionXY;
VAR_END

**PLC implementation**

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

**Expected result**

position1.x = 0.3160
position1.y = 3.8794
4 Functions

4.1 Coordinates

4.1.1 GetMcsCoordinateType

GetMcsCoordinateType

<table>
<thead>
<tr>
<th>mcsType</th>
<th>E_McsCoordType</th>
<th>CoordinateType</th>
<th>GetMcsCoordinateType</th>
</tr>
</thead>
</table>

Creates a CoordinateType FB from E_McsCoordType.

Syntax

Definition:

FUNCTION GetMcsCoordinateType : CoordinateType
VAR_INPUT
  mcsType : E_McsCoordType;
END_VAR

_inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcsType</td>
<td>E_McsCoordType</td>
<td>Mcs coordinate type.</td>
</tr>
</tbody>
</table>

_Return value_

CoordinateType

_Required License_

TC3 Physics Base

_System Requirements_

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT V3.1.4024.12</td>
<td>PC or CX (x64)</td>
<td>Tc3_Mc3PlanarMotion, Tc3_Physics</td>
</tr>
<tr>
<td>Advanced Motion Pack V3.1.10.11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5  Data Types

5.1  Enums

5.1.1  E_CoordCategory

Enumeration coordinate categories.

Syntax

Definition:

```plaintext
TYPE E_CoordCategory :
{
   Invalid := 0x0,
   MCS    := 0x1,
   ACS    := 0x2,
   Uninterpreted := 0x3
} UDINT;
END_TYPE
```

Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid</td>
<td>Invalid</td>
</tr>
<tr>
<td>MCS</td>
<td>MCS</td>
</tr>
<tr>
<td>ACS</td>
<td>ACS</td>
</tr>
<tr>
<td>Uninterpreted</td>
<td>Uninterpreted</td>
</tr>
</tbody>
</table>

5.1.2  E_McsCoordType

Enumeration of MCS coordinate types.

Syntax

Definition:

```plaintext
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid</td>
<td>Invalid</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
### 5.1.3 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:**

```plaintext
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>ABC</td>
</tr>
<tr>
<td>ACB</td>
<td>ACB</td>
</tr>
<tr>
<td>BCA</td>
<td>BCA</td>
</tr>
<tr>
<td>BAC</td>
<td>BAC</td>
</tr>
<tr>
<td>CAB</td>
<td>CAB</td>
</tr>
<tr>
<td>CBA</td>
<td>CBA</td>
</tr>
<tr>
<td>ABA</td>
<td>ABA</td>
</tr>
<tr>
<td>ACA</td>
<td>ACA</td>
</tr>
<tr>
<td>BAB</td>
<td>BAB</td>
</tr>
<tr>
<td>BCB</td>
<td>BCB</td>
</tr>
<tr>
<td>CAC</td>
<td>CAC</td>
</tr>
<tr>
<td>CBC</td>
<td>CBC</td>
</tr>
</tbody>
</table>
5.2 Structs

5.2.1 CoordinateType

Coordinate type.

Syntax

Definition:

```
TYPE CoordinateType :
  STRUCT
    _reserved0 : ULI NT;
  END_STRUCT
END_TYPE
```

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_reserved0</td>
<td>ULI NT</td>
<td>0</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>
More Information:
www.beckhoff.com/TE1000