# Inhaltsverzeichnis

1 Vorwort .................................................................................................................................................. 5
   1.1 Hinweise zur Dokumentation ........................................................................................................ 5
   1.2 Sicherheitshinweise .......................................................................................................................... 6
2 Dynamikbeschränkungen ....................................................................................................................... 7
   2.1 Beispiel ........................................................................................................................................... 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 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.

Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentlichte Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits geliefelter Produkte geltend gemacht werden.

Marken

Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.
Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.
1.2 Sicherheitshinweise

Sicherheitsbestimmungen
Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

Haftungsausschluss
Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals
Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Erklärung der Symbole
In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠️ GEFAHR

Akute Verletzungsgefahr!
Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠️ WARNUNG

Verletzungsgefahr!
Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠️ VORSICHT

Schädigung von Personen!
Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

⚠️ HINWEIS

Schädigung von Umwelt oder Geräten
Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

⚠️ Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.
2 Dynamikbeschränkungen


Wenn mehrere Beschränkungen gleichzeitig wirken, werden alle eingehalten. Für den Nutzer besteht daher keine Notwendigkeit, selbst zu berechnen, welche Beschränkung die Dynamik während der Bewegung am stärksten limitieren wird. Unwirksame Beschränkungen, beispielsweise eine Beschränkung an die z-Koordinate während einer reinen xy-Bewegung, werden ignoriert.

Folgende spezielle Werte werden unterstützt:

- **MC_DEFAULT**: Wird vom Empfänger des Verfahrbefehls mit dem entsprechenden Default-Wert ersetzt, falls dieser vorhanden ist.
- **MC_IGNORE**: Führt zu keiner Beschränkung. Dies kann sinnvoll sein, wenn man beispielsweise Beschleunigung und Ruck einer Koordinate limitieren möchte, die Geschwindigkeit aber nicht.

DynamicConstraint_Container


DynamicConstraint_PathXY


Zu beachten ist, dass die Gesamtbeschleunigung (-Ruck) in xy die tangentielle Beschleunigung (Ruck) übersteigen kann, wenn der Pfad in der xy-Ebene keine Gerade ist.

DynamicConstraint_Coordinates

Der FB DynamicConstraint_Coordinates beschränkt die Dynamikwerte einzelner Koordinaten, zum Beispiel Coord_Mcs_X. Pro Koordinate können Geschwindigkeit (V), Beschleunigung (A), Verzögerung (D) und Ruck (J) limitiert werden.
2.1 Beispiel

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

// Geschwindigkeit in XY wird auf 1000 beschränkt, die Änderungsrate dieser Geschwindigkeit auf 5000.
// Keine Beschränkung an den Ruck.
  ConstraintPath.SetValuesVADJ(V := 1000, A := 5000, D:= 5000, J := MC_IGNORE);

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

// Die Geschwindigkeit der C-Koordinate wird auf ihren Default-Wert limitiert, Beschleunigung, Verzögerung und Ruck auf konkrete Werte.
  ConstraintCoords.SetLimit(Coordinate := Coord_Mcs_C1, V := MC_DEFAULT, A := 1000, D := 1000,
                           J := 10000);

// Pfad- und Koordinatenbeschränkungen werden in einem Objekt kombiniert
  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

Removes all dynamic constraints from the container.

Syntax

Definition:

```
METHOD Clear
```

3.1.1.2 AddConstraint

Add a dynamic constraint to the container.

Syntax

Definition:

```
METHOD AddConstraint
VAR_INPUT
    Constraint : IPlcDynamicConstraint;
END_VAR
```
## 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

<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>

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

**Syntax**

**Definition:**
### Method: AddLimit

**Variables Input**

- **Coordinate**: CoordinateType
- **V**: MC_LREAL
- **A**: MC_LREAL
- **D**: MC_LREAL
- **J**: MC_LREAL

### 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>Maximal velocity.</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
<td>Maximal acceleration.</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
<td>Maximal deceleration.</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
<td>Maximal jerk.</td>
</tr>
</tbody>
</table>

### Syntax

**Definition:**

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

### SetLimit

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

**Syntax**

**Definition:**

```plaintext
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>Maximal velocity.</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
<td>Maximal acceleration.</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
<td>Maximal deceleration.</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
<td>Maximal jerk.</td>
</tr>
</tbody>
</table>

### RemoveLimit

Removes limits for a specific coordinate.

**Syntax**

**Definition:**

```plaintext
METHOD RemoveLimit
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>
</tbody>
</table>
**Syntax**

**Definition:**

```plaintext
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:**

```plaintext
METHOD Clear
```

**3.1.2.5 GetV**

GetV

```plaintext
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:**

```plaintext
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

```plaintext
Coordinate : CoordinateType
MC_LREAL GetA
```

---

**Function Blocks**

**Version:** 1.2

**TE1000**
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

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

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

**Definition:**

```plaintext
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</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

SetValuesVADJ

<table>
<thead>
<tr>
<th>V</th>
<th>MC_LREAL</th>
</tr>
</thead>
<tbody>
<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>

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>maximal velocity.</td>
</tr>
<tr>
<td>A</td>
<td>MC_LREAL</td>
<td>maximal acceleration.</td>
</tr>
<tr>
<td>D</td>
<td>MC_LREAL</td>
<td>maximal deceleration.</td>
</tr>
<tr>
<td>J</td>
<td>MC_LREAL</td>
<td>maximal 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>
### 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 SetZero

**SetZero**

Set the coordinates of this position to '0.0'.

**Syntax**

**Definition:**

METHOD SetZero

**Beispiel: PositionXY.SetZero**

**PLC Deklaration**

VAR

  position : PositionXY;

VAR_END

**PLC Implementierung**

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

**Erwartetes Ergebnis**

position.x = 0
position.y = 0

#### 3.2.1.2 SetValues

**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>
<tbody>
<tr>
<td>positionXY</td>
<td>PositionXY</td>
<td>The coordinate values to set.</td>
</tr>
</tbody>
</table>
Beispiel: PositionXY.SetValues

PLC Deklaration

VAR
  position0 : PositionXY;
  position1 : PositionXY;
VAR_END

PLC Implementierung

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

Erwartetes Ergebnis

position1.x = 5.6
position1.y = 0.2

3.2.1.1.3  SetValuesXY

Set the xy-coordinates of this position.

Syntax

Definition:

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

Inputs

<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>

Beispiel: PositionXY.SetValuesXY

PLC Deklaration

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

PLC Implementierung

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

Erwartetes Ergebnis

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>

Beispiel: PositionXY.SetValuesRP

PLC Deklaration

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

PLC Implementierung

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

Erwartetes Ergebnis

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
Beispiel: PositionXY.GetRadius

PLC Deklaration

VAR
   position : PositionXY;
   distance : LREAL;
VAR_END

PLC Implementierung

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

Erwartetes Ergebnis

distance = 5.604

3.2.1.1.6 GetPhi

Get the polar angle scaled in degrees [°].

Syntax

Definition:
METHOD GetPhi : LREAL

Return value

LREAL

Beispiel: PositionXY.GetPhi

PLC Deklaration

VAR
   position : PositionXY;
   polarAngle : LREAL;
VAR_END

PLC Implementierung

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

Erwartetes Ergebnis

polarAngle = 2.045

3.2.1.1.7 Invert

Invert this position.

Syntax

Definition:
METHOD Invert
Example: PositionXY.Invert

PLC Declaration

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

PLC Implementation

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

Expected Result

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

### 3.2.1.1.8 ConcatenateWith

```
ConcatenateWith
_ _ _
position PositionXY
```

Concatenate with a 2nd position.

**Syntax**

**Definition:**

```plaintext
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>
</table>

Example: PositionXY.ConcatenateWith

PLC Declaration

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

PLC Implementation

```plaintext
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

```
ShiftByXY
_ _ _
dx  LREAL
_ _ _
dy  LREAL
```

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>

Beispiel: PositionXY.ShiftByXY

PLC Deklaration

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

PLC Implementierung

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

Erwartetes Ergebnis

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 [22]</td>
<td>Set the components of this vector to '0.0'.</td>
</tr>
<tr>
<td>SetValues [22]</td>
<td>Set the components of this position.</td>
</tr>
<tr>
<td>SetValuesXY [23]</td>
<td>Set the spatial components of this position.</td>
</tr>
<tr>
<td>SetValuesXYC [24]</td>
<td>Set the components of this position.</td>
</tr>
<tr>
<td>Invert [24]</td>
<td>Invert this position.</td>
</tr>
<tr>
<td>ConcatenateWith [25]</td>
<td>Multiply this position by a 2nd from the right.</td>
</tr>
<tr>
<td>StepByLocalXY [25]</td>
<td>Apply a local step in respect to the local coordinate frame without changing the orientation.</td>
</tr>
<tr>
<td>ShiftByXY [26]</td>
<td>Displace the components of this position.</td>
</tr>
<tr>
<td>TurnByC [27]</td>
<td>Rotate the attached orientation without changing the Cartesian coordinates.</td>
</tr>
<tr>
<td>Transform [27]</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

**SetZero**

Set the components of this vector to '0.0'.

**Syntax**

**Definition:**

```
METHOD SetZero
```

**Beispiel: PositionXYC.SetZero**

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

#### 3.2.1.2.2 SetValues

**SetValues**

```
positionXYC  PositionXYC
```

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>
</table>
Beispiel: PositionXYC.SetValues

PLC Deklaration

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

PLC Implementierung

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

Erwartetes Ergebnis

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

3.2.1.2.3 SetValuesXY

SetValuesXY

<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>

Syntax

Definition:

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

Inputs

Beispiel: PositionXYC.SetValuesXY

PLC Deklaration

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

PLC Implementierung

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

Erwartetes Ergebnis

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

SetValuesXYC

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

**Beispiel: PositionXYC.SetValuesXYC**

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

3.2.1.2.5 Invert

Invert

Invert this position.

**Syntax**

**Definition:**

```plaintext
METHOD Invert
```
### Beispiel: PositionXYC.Invert

**PLC Deklaration**

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

**PLC Implementierung**

```plaintext
position.SetValueXYC(1, 3, 10);
position.Invert();
```

**Erwartetes Ergebnis**

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

### 3.2.1.2.6 ConcatenateWith

Multiply this position by a 2nd from the right.

**Syntax**

**Definition:**

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

**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>

### Beispiel: PositionXYC.ConcatenateWith

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

### 3.2.1.2.7 StepByLocalXY

```plaintext
StepByLocalXY
  dx LREAL
  dy LREAL
ENDMETHOD
```
Apply a local step in respect to the local coordinate frame without changing the orientation.

**Syntax**

**Definition:**

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

**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>

**Beispiel: PositionXYC.StepByLocalXY**

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

### 3.2.1.2.8 ShiftByXY

**Syntax**

**Definition:**

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

**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>
### Beispiel: PositionXYC.ShiftByXY

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

### 3.2.1.2.9 TurnByC

**TurnByC**

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

**Syntax**

**Definition:**

Rotate the attached orientation without changing the Cartesian coordinates.

**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>

### Beispiel: PositionXYC.TurnByC

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

### 3.2.1.2.10 Transform

**Transform**

```plc
METHOD Transform
  VAR_INPUT
    positionXY : PositionXY;
  END_VAR
```

**Syntax**

**Definition:**

Transform 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>
</table>

**Beispiel: PositionXYC.Transform**

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

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

4.1 Coordinates

4.1.1 GetMcsCoordinateType

GetMcsCoordinateType

| mcsType | E_McsCoordType | CoordinateType | GetMcsCoordinateType |

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[30]</td>
<td>Mcs coordinate type.</td>
</tr>
</tbody>
</table>

_Return value

CoordinateType[32]

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>
</tbody>
</table>
5 Data Types

5.1 Enums

5.1.1 E_CoordCategory

Enumeration coordinate categories.

Syntax

Definition:

```
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:

```
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:**

```
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:

```plaintext
TYPE CoordinateType :
  STRUCT
    _reserved0 : ULLINT;
  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>ULLINT</td>
<td>0</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>
Mehr Informationen:
www.beckhoff.de/TE1000