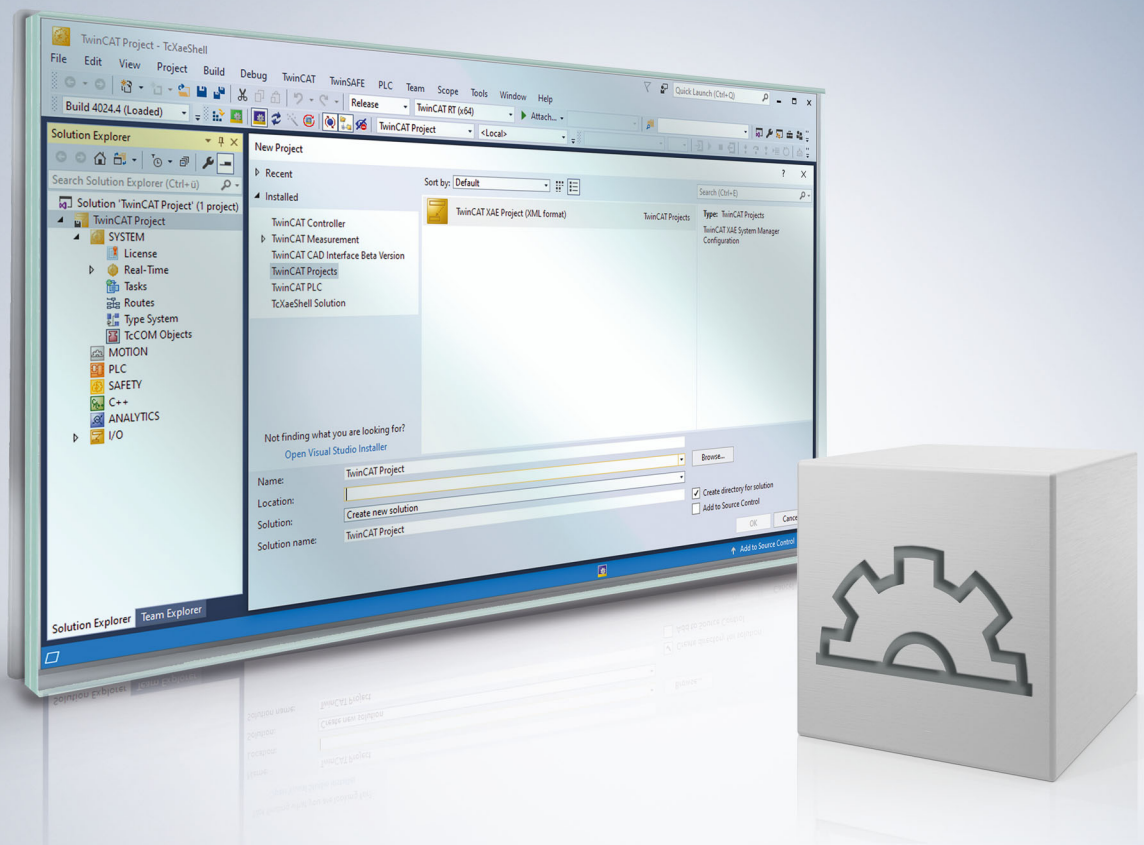
BECKHOFF New Automation Technology

Handbuch | DE

# TE1000

## TwinCAT 3 | PLC-Bibliothek: Tc3_Physics




11.03.2024 | Version: 1.3.2

# Inhaltsverzeichnis

# 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, stets die aktuell gültige 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 weiterentwickelt.
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 gelieferter Produkte geltend gemacht werden.

### Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
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:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
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

## 1.2 Zu Ihrer Sicherheit

### Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

### 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.

### Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

### Warnungen vor Personenschäden

**⚠ GEFAHR**

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

**⚠ WARNUNG**

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

**⚠ VORSICHT**

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

### Warnung vor Umwelt- oder Sachschäden

***HINWEIS***

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

### Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

## 1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/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<br>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<br>Advanced Motion Pack V3.2.27 | PC or CX (x64) | Tc3_Physics |

# 4 Function Blocks

## 4.1 Dynamics

Das Interface IPlcDynamicConstraint wird von vielen Verfahrbefehlen als optionaler Input akzeptiert, um die erlaubten Werte für Geschwindigkeit, Beschleunigung, Verzögerung oder Ruck während der Fahrt zu beschränken. Es gibt verschiedene Typen von Beschränkungen, häufig werden auch Kombinationen verschiedener Typen akzeptiert, welche in einem DynamicConstraint_Container [▶ 16] zusammengefasst werden:

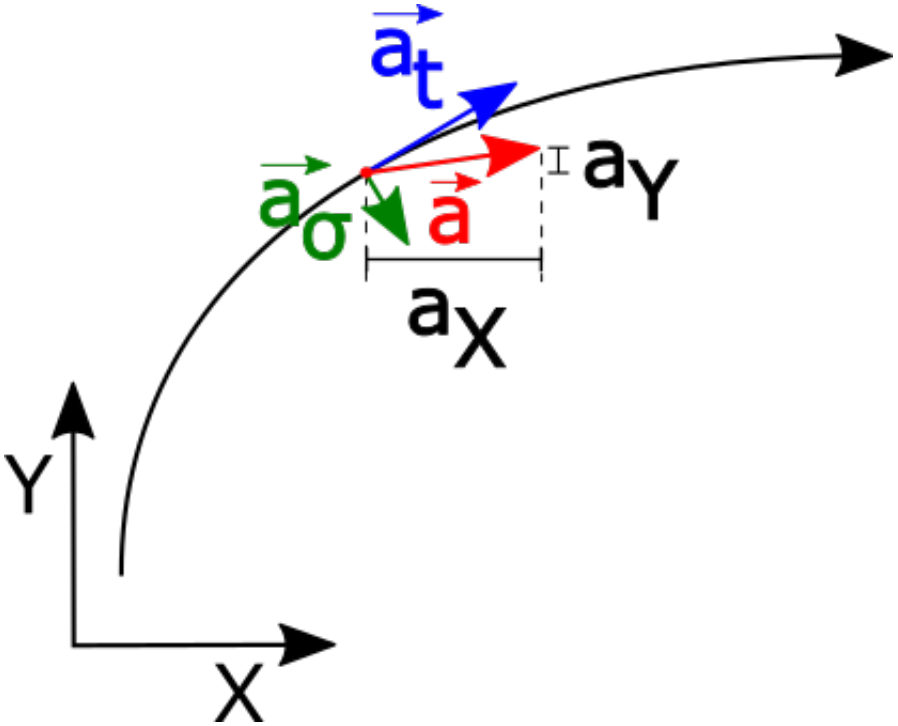
- Beschränkungen einzelner Koordinaten
  - DynamicConstraint_Coordinates [▶ 18]
- Beschränkungen in Fahrtrichtung
  - DynamicConstraint_PathXY [▶ 22]
  - DynamicConstraint_PathXYZ [▶ 23]
- Beschränkungen symmetrisch in alle Richtungen
  - DynamicConstraint_CartesianXY [▶ 14]
  - DynamicConstraint_CartesianXYZ [▶ 15]

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 speziellen 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.

**Beispiel Plot für Beschleunigung in 2D**




2D-Geschwindigkeit (nicht eingezeichnet, in Fahrtrichtung):

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

2D-Beschleunigung:

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

Tangentiale Beschleunigung in Fahrtrichtung:

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

Mögliche Beschränkungen für die Beschleunigung:

DynamicConstraint_PathXY [▶ 22]:

$$a_{tang} \leq A$$

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

DynamicConstraint_CartesianXY [▶ 14]:

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

DynamicConstraint_Coordinates [▶ 18]:

$$a_x \leq A_x$$

$$-a_x \leq D_x$$

$$a_y \leq A_y$$

$$-a_y \leq D_y$$

**Beispiel 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. |

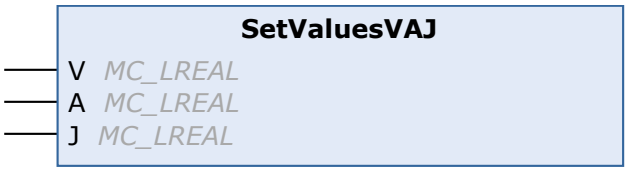
#### Weiterführende Informationen

Der Funktionsbaustein `DynamicConstraint_CartesianXY` beschränkt die Dynamikwerte einer Bewegung in der XY-Ebene. V, A und J sind dabei jeweils die maximalen Werte für die Geschwindigkeit (V), Beschleunigung (A) und Ruck (J) innerhalb der XY-Ebene. Im Gegensatz zu DynamicConstraint_PathXY [▶ 22] gelten diese Beschränkungen nicht nur in Fahrtrichtung, sondern symmetrisch in alle XY-Richtungen.

#### System Requirements

| Development environment | Target system type | PLC libraries to include |
|---|---|---|
| TwinCAT V3.1.4024.35<br>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. |

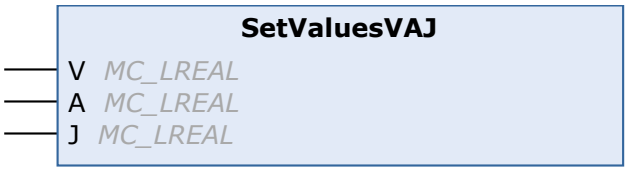
### Weiterführende Informationen

Der Funktionsbaustein DynamicConstraint_CartesianXYZ beschränkt die Dynamikwerte einer Bewegung im XYZ-Raum. V, A und J sind dabei jeweils die maximalen Werte für die Geschwindigkeit (V), Beschleunigung (A) und Ruck (J) innerhalb des XYZ-Raums. Im Gegensatz zu DynamicConstraint_PathXYZ [▶ 23] gelten diese Beschränkungen nicht nur in Fahrtrichtung, sondern symmetrisch in alle XYZ-Richtungen.

### System Requirements

| Development environment | Target system type | PLC libraries to include |
|---|---|---|
| TwinCAT V3.1.4024.35<br>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. |

### Weiterführende Informationen

Der Funktionsbaustein `DynamicConstraint_Container` definiert keine eigenen Beschränkungen. Sein Zweck ist es, mehrere andere Dynamikbeschränkungen in einem Objekt zusammenzufassen. Dies kann zum Beispiel nötig werden, wenn für eine Bewegung sowohl die Dynamikwerte auf dem Pfad als auch die Dynamikwerte einer einzelnen Koordinate beschränkt werden sollen.

Einer Instanz des `DynamicConstraint_Container` können sowohl Kopien als auf Referenzen von Beschränkungen hinzugefügt werden:

AddConstraint [▶ 17]

Der Instanz des `DynamicConstraint_Container` wird eine Kopie einer Dynamikbeschränkung hinzugefügt. Änderungen der Ursprungsbeschränkung haben keinen Einfluss auf die Kopie der Beschränkung in der Instanz des `DynamicConstraint_Container`.

AddConstraintByReference [▶ 18]

Der Instanz des `DynamicConstraint_Container` wird eine Referenz auf eine Dynamikbeschränkung hinzugefügt. Änderungen der Beschränkung werden von der Instanz des `DynamicConstraint_Container` berücksichtigt.

### System Requirements

| Development environment | Target system type | PLC libraries to include |
|---|---|---|
| TwinCAT V3.1.4024.12<br>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. |

### Weiterführende Informationen

Der Funktionsbaustein 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.

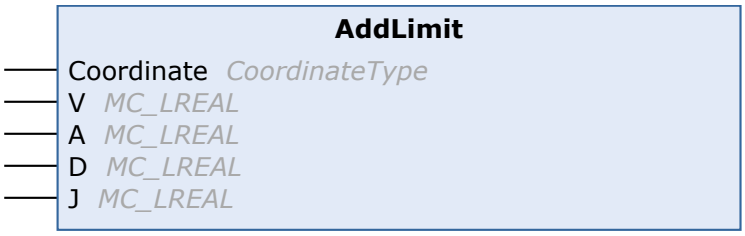
### Required License

TC3 Physics Base

### System Requirements

| Development environment | Target system type | PLC libraries to include |
|---|---|---|
| TwinCAT V3.1.4024.12<br>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

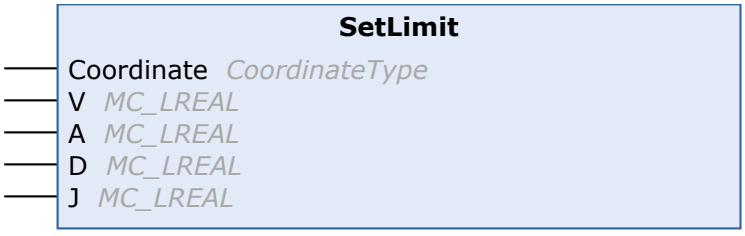
Definition:

```
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

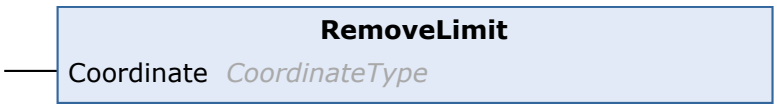
Definition:

```
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<br>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. |

#### Weiterführende Informationen

Der Funktionsbaustein DynamicConstraint_PathXY beschränkt die tangentialen Dynamikwerte einer Bewegung in der XY-Ebene. V ist dabei der maximale Wert für die Geschwindigkeit innerhalb der XY-Ebene. A, D und J sind jeweils die maximalen Werte für Beschleunigung (A), Verzögerung (D) und Ruck (J) in Richtung der aktuellen XY-Geschwindigkeit.

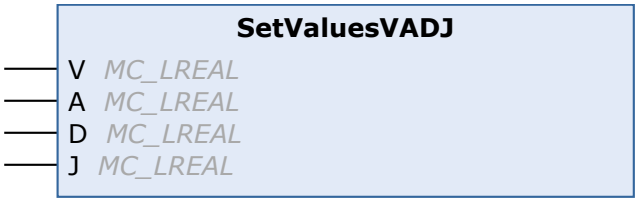
Zu beachten ist, dass die Gesamtbeschleunigung (-Ruck) in XY die tangentiale Beschleunigung (Ruck) übersteigen kann, wenn der Pfad in der XY-Ebene keine Gerade ist.

Sollen die Beschränkungen nicht nur in Fahrtrichtung, sondern symmetrisch in alle XY-Richtungen wirken, ist der Funktionsbaustein DynamicConstraint_CartesianXY [▶ 14] zu verwenden.

#### System Requirements

| Development environment | Target system type | PLC libraries to include |
|---|---|---|
| TwinCAT V3.1.4024.12<br>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. |

### Weiterführende Informationen

Der Funktionsbaustein DynamicConstraint_PathXYZ beschränkt die tangentialen Dynamikwerte einer Bewegung im XYZ-Raum. V ist dabei der maximale Wert für die Geschwindigkeit innerhalb des XYZ-Raums. A, D und J sind jeweils die maximalen Werte für Beschleunigung, Verzögerung und Ruck in Richtung der aktuellen XYZ-Geschwindigkeit.

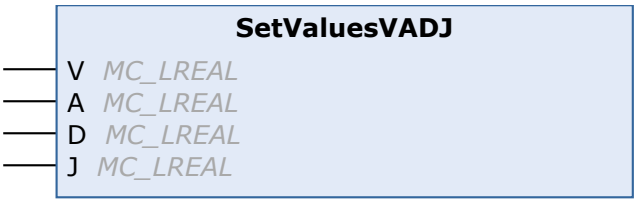
Zu beachten ist, dass die Gesamtbeschleunigung (-Ruck) in XYZ die tangentiale Beschleunigung (Ruck) übersteigen kann, wenn der Pfad in XYZ keine Gerade ist.

Sollen die Beschränkungen nicht nur in Fahrtrichtung, sondern symmetrisch in alle XYZ-Richtungen wirken, ist der Funktionsbaustein DynamicConstraint_CartesianXYZ [▶ 15] zu verwenden.

### System Requirements

| Development environment | Target system type | PLC libraries to include |
|---|---|---|
| TwinCAT V3.1.4024.35<br>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<br>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
```

**Beispiel: PositionXY.SetZero**

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

```
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. |

#### Beispiel: PositionXY.SetValues

#### PLC Deklaration

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

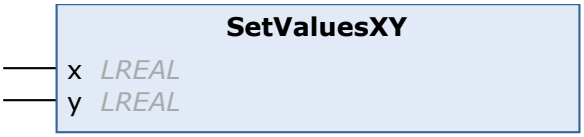
#### PLC Implementierung

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

#### Erwartetes Ergebnis

```
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. |

### Beispiel: PositionXY.SetValuesXY

#### PLC Deklaration

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

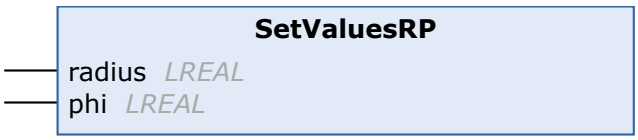
#### PLC Implementierung

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

#### Erwartetes Ergebnis

```
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 [°]. |

### Beispiel: PositionXY.SetValuesRP

#### PLC Deklaration

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

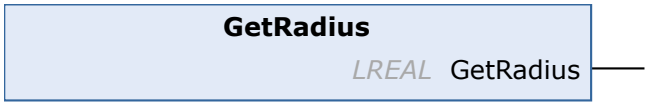
#### PLC Implementierung

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

#### Erwartetes Ergebnis

```
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

#### Beispiel: PositionXY.GetRadius

#### PLC Deklaration

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

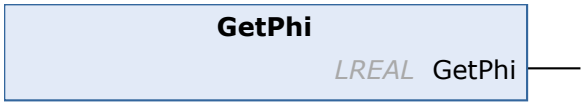
#### PLC Implementierung

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

#### Erwartetes Ergebnis

```
distance = 5.604
```

### 4.2.1.1.6 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;
END_VAR
```

#### PLC Implementierung

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

#### Erwartetes Ergebnis

```
polarAngle = 2.045
```

### 4.2.1.1.7 Invert


Invert this position.

#### Syntax

Definition:

```
METHOD Invert
```

#### Beispiel: PositionXY.Invert

#### PLC Deklaration

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

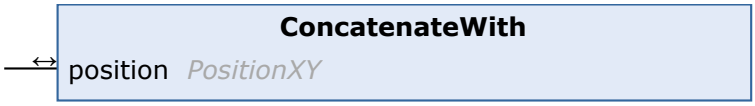
#### PLC Implementierung

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

#### Erwartetes Ergebnis

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

### 4.2.1.1.8 ConcatenateWith




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. |

#### Beispiel: PositionXY.ConcatenateWith

#### PLC Deklaration

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

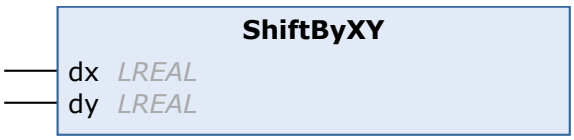
#### PLC Implementierung

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

### Erwartetes Ergebnis

```
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. |

### Beispiel: PositionXY.ShiftByXY

### PLC Deklaration

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

### PLC Implementierung

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

### Erwartetes Ergebnis

```
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<br>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
```

### Beispiel: PositionXYC.SetZero

### PLC Deklaration

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

### 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
```

## 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. |

#### Beispiel: PositionXYC.SetValues

#### PLC Deklaration

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

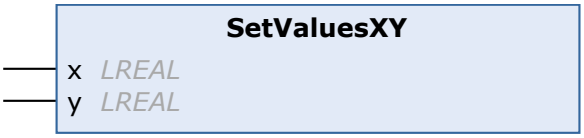
#### PLC Implementierung

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

#### Erwartetes Ergebnis

```
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. |

#### Beispiel: PositionXYC.SetValuesXY

#### PLC Deklaration

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

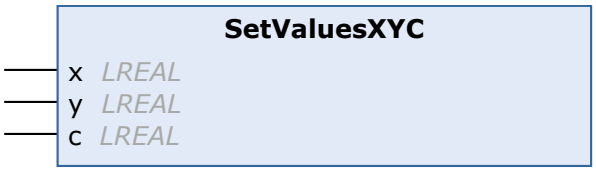
#### PLC Implementierung

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

### Erwartetes Ergebnis

```
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 [°]. |

### Beispiel: PositionXYC.SetValuesXYC

### PLC Deklaration

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

### PLC Implementierung

```
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
```

## 4.2.1.2.5 Invert


Invert this position.

### Syntax

Definition:

```
METHOD Invert
```

**Beispiel: PositionXYC.Invert**

**PLC Deklaration**

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

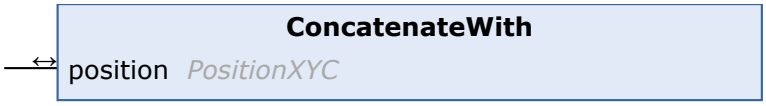
**PLC Implementierung**

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

**Erwartetes Ergebnis**

```
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. |

**Beispiel: PositionXYC.ConcatenateWith**

**PLC Deklaration**

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

**PLC Implementierung**

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

**Erwartetes Ergebnis**

```
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. |

### Beispiel: PositionXYC.StepByLocalXY

### PLC Deklaration

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

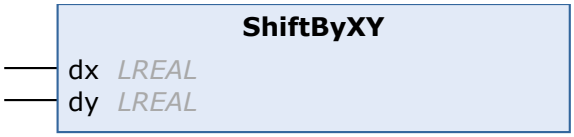
### PLC Implementierung

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

### Erwartetes Ergebnis

```
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. |

### Beispiel: PositionXYC.ShiftByXY

### PLC Deklaration

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

### PLC Implementierung

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

### Erwartetes Ergebnis

```
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 [°]. |

### Beispiel: PositionXYC.TurnByC

### PLC Deklaration

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

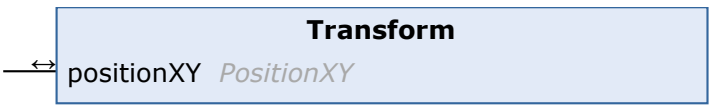
### PLC Implementierung

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

### Erwartetes Ergebnis

```
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. |

### Beispiel: PositionXYC.Transform

### PLC Deklaration

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

### PLC Implementierung

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

### Erwartetes Ergebnis

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

# 5 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

### Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

### Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

### Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

| | |
|---|---|
| Hotline: | +49 5246 963-157 |
| E-Mail: | support@beckhoff.com |

### Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

| | |
|---|---|
| Hotline: | +49 5246 963-460 |
| E-Mail: | service@beckhoff.com |

### Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

| | |
|---|---|
| Telefon: | +49 5246 963-0 |
| E-Mail: | info@beckhoff.com |
| Internet: | www.beckhoff.com |

Mehr Informationen:
**www.beckhoff.com/te1000**

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